Source Code of Jamella's Diablo 2 Editor 2001-03-17

This source code is my intellectual property. It was completely written by myself. I have published it so fellow programmers and other interested people can have a look into the internals of my editor.

If you use parts of the code consider that I needed over half a year to build this program and mention me in your info box if you like.

Other than that your will find some modifications I made after releasing version 3.0 of the editor, so don't be surprised if you find some extra functions and useless pieces of code.

And please don't email me and ask me how the code works, because it does and I don't have time to teach the whole world object-orientated C++.

Now dig into it by first reading the header file.

Jamella

http://jamella.cjb.net

mailto: jamella@gmx.net

```
// D2E.h Diablo 2 Editor Header File

/*
* Central Configurations for all Projects
*/

#define PUB                 0
#define DEV                 1
#define DEVPUB              2

#define RELEASE             DEV
#define RELEASECODE         "PIPER"

#define JEPRELOADITEMIMAGES 0

#if RELEASE == PUB

#define JEVERSION           "Version 3.0"
#define BUGGYMESSAGES       1

#elif RELEASE == DEVPUB

#define JEVERSION           "Public Development Version"
#define BUGGYMESSAGES       1

#else

#define VERSION             "Development Version"
#define BUGGYMESSAGES       0

#endif

#if _DEBUG
#define SHOWMAGICCODE       1
#else
#define SHOWMAGICCODE       0
#endif

#if defined(JAMELLAEDITOR)

#define PROGRAMNAME         "Jamella's Diablo 2 Hero Editor"

#define _WIN32_IE           0x0500
#define IDT_TIMER           12345

#include <windows.h>
#include <stdio.h>
#include <string.h>
#include <stdarg.h>
#include <shlobj.h>
#include <richedit.h>

// Resource Identifiers
#include "ResourceIDs.h"

#else if defined(JAMELLACOMSERVER)

#define PROGRAMNAME         "Jamella's COM Object"
#define COMPROGID           "Jamella.Server"
#define COMVERID            "Jamella.Server.3"
#define COMDESCRIPTION      "Jamella Diablo 2 COM Server"

// Includes
#include <windows.h>
#include <ole2.h>
#include <commctrl.h>
#include <stdio.h>
#include <shlobj.h>

// Resource Identifiers
#include "ResourceIDs.h"

#endif

#define REGISTRYKEY         "Software\\Jamella's Diablo 2 Editor"
#define D2SPROGID           "JamellaD2Editor.Character"
#define D2IPROGID           "JamellaD2Editor.Item"
#define D2SDESCRIPTION      "Diablo II Character"
#define D2IDESCRIPTION      "Diablo II Item"
```

```
#define PERMANENTURL        "http://jamella.cjb.net"
#define EMAIL               "jamella@gmx.net"
#define HTMLHELPFILE        "JamellaHelp.chm"
#define CLIPBOARDFORMAT     "JamellaD2EItem30"
#define WM_VALIDATE         WM_USER
#define SEARCHPOLLER        250
#define EXPERTBOXPOLLER     500
#define DECODETIMEOUT       1000000
#define PROGRESSCOLOR       RGB(200,0,0)
#define HISTORYSTEPS        32
#define MAXMODLEVEL         99
#define BFASPEEDPAST        10
#define JEINVGRIDS          1

// Windowing Data
extern HINSTANCE        hInstance;
extern HWND             hMainDialog,hTabDialog;

// Structures
struct Damage
{
    int     Hands;
    int     OneHandMin,OneHandMax;
    int     TwoHandMin,TwoHandMax;
    bool    Magic;
};

struct ArmorClass
{
    int     BaseAC;
    int     AC;
    bool    Magic;
};

// MainDlg.cpp
struct maindlgtab
{
    int             iddialog;
    int             idicon;
    char*           text;
    void*           proc;
    DLGTEMPLATE*    dialogtemplate;
    int             imagelistindex;
};

// ToolTips
extern HWND hToolTip;
extern HWND CreateToolTipCtrl(HWND hParent,int ToolSet,bool Relay);

// TextSummary
extern void WriteTextSummary(HWND);

// General Resources
extern HBITMAP hBmpPlus;
extern HICON hIconHelp;
extern HMENU hBatchMenu;
extern void D2ELoadResources();
extern void D2EUnLoadResources();

// Tab0.cpp
extern HBITMAP hBmpJamella;

// Tab1.cpp
extern const char* CharClasses[5];
extern const char* Difficulties[4];

// Item Editor Commons
extern class Item*          SelItem;
extern struct ItemInfo*     SelInfo;

// Tab2.cpp
extern HBITMAP hBmpBodyWhole,hBmpNotPlaceable;
extern HCURSOR hCurMove,hCurMoveCopy,hCurCross,hCurNo;
extern HIMAGELIST hTVImagelist;
extern HBRUSH hBrushBlack,hBrushNull;
extern HPEN hPenWhite,hPenGreen,hPenBusy,hPenUnwearable;
extern void UpdateTab2();
```

```
// Tab2E.cpp
extern HWND hExpertBox;
extern void UpdateTab2E();

// Tab2Random.cpp
extern HWND hRandomBox;
extern int  RandomBoxDW;
extern void UpdateTab2Rnd();

// Tab2ItemList.cpp
extern HWND hItemListBox;
extern void UpdateTab2ItemList();

// Tab2Rare.cpp
extern HCURSOR hCurAdd;

// Tab2ItemFileLoadSave
extern bool LoadItemFile(HWND hWnd);
extern bool SaveItemFile(HWND hWnd);
extern Item *MakeItemFromData(BYTE* data, int size);
extern Item *MakeItemFromFile(HANDLE hFile);

// Tab3.cpp
struct skill
{
    int     ifoffset;
    char    *name;
    int     idstring;;
    int     idbitmap;
    HBITMAP hbitmap;
};
extern struct skill skills[5*30];

// Tab4.cpp
struct queststatus
{
    char    *text;
    WORD    value;
};
struct quest
{
    int         idstring;
    int         idbitmap;
    queststatus*stati;
    int         offset;
    HANDLE      hbitmap;
    char        string[64];
};
extern struct quest quests[21];

// Tab5.cpp
extern HBITMAP hBmpWaypointOn;
extern HBITMAP hBmpWaypointOff;

// Infobox.cpp
extern HBITMAP hBmpWebLink;
extern void     RTFStreamSend(HWND hWnd,int Ctrl,const char *Stream);

// Item Grid Managment Sizes
struct InvGrids
{
    int     xInventory,yInventory;
    int     xStash,yStash;
    int     xCube,yCube;
    int     xBelt,yBelt;
};

extern struct InvGrids      InvGrids;
extern void SetInvGridPreset(int i);

// Items Structure
struct ItemInfo
{
    char*   ItemName;
    char    Quality;
    char*   BaseItemName;
    char*   ItemType;
    DWORD   ItemCode;
    int     UniqueCode;
```

```
    int     DWBCode;
    char    ItemSearch;
    DWORD   IC;
    bool    Sm;
    int     BitmapID;
    int     TreeID;
    int     MagicMask;
    int     RareMask;
    int     QualityMask;
    int     SizeX,SizeY;
    int     BodyPlace;
    char    Magic;
    int     Durability;
    DWORD   MinStr;
    DWORD   MinDex;
    DWORD   MinLvl;
    int     Hands;
    int     OneHandDmgMin;
    int     OneHandDmgMax;
    int     TwoHandDmgMin;
    int     TwoHandDmgMax;
    int     MissileDmgMin;
    int     MissileDmgMax;
    int     UndeadBonus;
    int     ACMin;
    int     ACMax;
    int     Sockets;
    char    GemClass;
    int     Quantity;
    char*   Description;

    // Zero initialized
    HBITMAP hBmp;
    HTREEITEM hTree;
};
extern struct ItemInfo      ItemInfos[];
extern int                  nItemInfos;
extern struct ItemInfo      itemunknown;

extern struct ItemInfo*     SelInfo;

// Effect Structure
struct Effect
{
    DWORD           Code;
    int             Min;
    int             Max;
};

// Ring and Amulet Structures
struct ImageMap
{
    int         BmpID;
    HBITMAP     hBmp;
};
extern struct ImageMap      RingImages[];
extern struct ImageMap      AmuletImages[];

// Item Tree Structures
struct _ItemTree
{
    int         Depth;
    char*       Text;
    int         TreeID;
    HTREEITEM   hTree;
};
extern struct _ItemTree     ItemTree[];
extern int                  nItemTree;

// Magic Modifiers Tables
struct _MagicPreSuffix
{
    int     N;
    char    *Text;
    int     ModLevel;
    int     ELevel;
    int     Group;
    int     nMod;
    Effect  Mod[4];
```

```
    int     MagicMask;
    int     Transform;
    int     Transformcolor;
    char*   Description;
};
extern struct _MagicPreSuffix   MagicPrefixTable[],MagicSuffixTable[];
extern int                      nMagicPrefixTable,nMagicSuffixTable;

struct Modifier
{
    DWORD   Code;
    char    *Text;
};
extern const struct Modifier    Modifiers[];
extern int                      nModifiers;

// Superior Items Table
struct _SuperiorItem
{
    int     X;
    int     nMods;
    Effect  Mod[2];
    int     SuperiorMask;
    int     Level;
    int     Multiply;
    int     Add;
};
extern const struct _SuperiorItem   SuperiorItemTable[];
extern int                          nSuperiorItemTable;

// Magical Items PreSuffix Tree
struct _MagicPreSuffixTree
{
    int     Depth;
    char*   Text;
    int     ModID;
    HTREEITEM hTree;
};
extern struct _MagicPreSuffixTree   MagicPreSuffixTree[];
extern int                          nMagicPreSuffixTree;

// Rare Item PreSuffix Table
struct _RarePreSuffix
{
    char*   Text;
    int     RareMask;
};
extern const struct _RarePreSuffix  RarePrefixTable[],RareSuffixTable[];
extern int                          nRarePrefixTable,nRareSuffixTable;

// Unique Item Attributes Table
struct _UniqueItem
{
    char*   Name;
    int     nMod;
    Effect  Mod[7];
};
extern const struct _UniqueItem     UniqueItems[];
extern int                          nUniqueItems;

// Set Item Attributes Table
struct _SetItem
{
    char*   SetName;
    int     SetID;
    char*   Prefix;
    int     Level;
    int     nItems;
    int     nProperties;
    int     Transform;
    int     Transformcolor;
    struct
    {
        DWORD   Code;
        DWORD   IC;
        char*   Suffix;
    }
    Item[6];
```

```
    Effect  Mod[16];
};
extern const struct _SetItem        SetItems[];
extern int                          nSetItems;

struct GemInfo
{
    char    *Name;
    DWORD   ItemCode;
    DWORD   IC;
    int     Transform;
    int     nMods;

    Effect  WeaponMod[3];
    Effect  HelmMod[3];
    Effect  ShieldMod[3];
};
extern const struct GemInfo         GemInfos[];
extern int                          nGemInfos;

// Popup Help Structure
struct PopupHelp
{
    int             CtrlID;
    int             HelpID;
    const char*     HelpText;
};
extern const struct PopupHelp       PopupHelps[];
extern int                          nPopupHelps;

// Main.cpp
int MainDialog(const char* CmdLine);
int ErrorMessage();

// HelpBox.cpp
bool ToggleHelpBox(HWND hWnd,int HELPID);
void CloseHelpBox();

// CommandLine.cpp
extern void         ParseCommandLine();
extern const char   *CmdLineFile();
extern const char   *ProgramFilePath();

// Registry.cpp
struct _RegOptions
{
    int     CreateItemRecordFormat;
    int     AllItemsSocketable;
    int     A7Gems;
    int     ExceedQuantity;
    char    ItemPath[260];
    int     Associations;
    int     ToolTips;
    int     NoAnnoyingMsgs;
};
extern void         CheckShellRegistry();
extern void         SaveEditorRegistryValues();
extern void         LoadEditorRegistryValues();
extern _RegOptions  RegOptions;

LRESULT CALLBACK Tab0DialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam);
LRESULT CALLBACK Tab1DialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam);
LRESULT CALLBACK Tab2DialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam);
LRESULT CALLBACK Tab2InventoryProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam);
LRESULT CALLBACK Tab2EDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam);
LRESULT CALLBACK Tab2RndDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam);
LRESULT CALLBACK Tab2MagicDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam);
LRESULT CALLBACK Tab2RareDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam);
LRESULT CALLBACK Tab2SearchDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam);
LRESULT CALLBACK Tab2GemsDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam);
LRESULT CALLBACK Tab3DialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam);
LRESULT CALLBACK Tab4DialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam);
LRESULT CALLBACK Tab5DialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam);
LRESULT CALLBACK CowLevelDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam);
LRESULT CALLBACK InfoDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam);
LRESULT CALLBACK SaveDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam);
LRESULT CALLBACK NewDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam);
LRESULT CALLBACK RenameDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam);
LRESULT CALLBACK UOptionsDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam);
```

```
LRESULT CALLBACK EOptionsDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam);
LRESULT CALLBACK Tab2ItemListDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam);
#if INVGRIDS > 0
LRESULT CALLBACK Tab2ExGridDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam);
#endif


#pragma pack(1)

class fileclass
{
private:
    char        filename[260];

    // Private Structures
    struct
    {
        BYTE    W4[4];              // '01 77 34 00'
        BYTE    unknown[48];
    } w4;
    struct
    {
        BYTE    JM[2];              // 'JM'
        WORD    num;                // number of items
    } JMinv;

public:
    fileclass();
    ~fileclass();

    bool        loaded;

    // File operations
    void        clear();
    bool        loadfile(HWND hWnd,const char *filename);
    bool        savefile(HWND hWnd);
    bool        reloadfile(HWND hWnd);
    bool        discardfile(HWND hWnd);
    inline bool isloaded()                      { return loaded; }

    int         transferdata(BYTE *data,int size);
    void        setfilename(const char *file);

    // Public Structures
    struct
    {
        BYTE    signatur[8];        // '55 AA 55 AA 47 00 00 00'
        char    playername[16];
        BYTE    hardcore;
        BYTE    diff;
        WORD    unknown1[4];        // '00 00 DD 00 10 00 82 00'
        WORD    playerclass;
        WORD    level;
        BYTE    unknown2[50];
        BYTE    startingtown;
        BYTE    unknown3[41];
    } Header;
    struct
    {
        BYTE    Woo[8];             // '57 6F 6F 21 06 00 00 00'
        WORD    unknown1;           // '2A 01'
        WORD    quests1[48];
        WORD    quests2[48];
        WORD    quests3[48];
    } Woo;
    struct
    {
        BYTE    WS[8];              // '57 53 01 00 00 00 50 00'
        WORD    unknown1;           // '02 01'
        DWORD   waypoints1;
        BYTE    unused1[18];
        WORD    unknown2;           // '02 01'
        DWORD   waypoints2;
        BYTE    unused2[18];
        WORD    unknown3;           // '02 01'
        DWORD   waypoints3;
        BYTE    unused3[18];
    } WS;
    struct
```

```
    {
        BYTE    gf[2];              // 'gf'
        WORD    fields;             // Bit Field:    '7 6 5 4:3 2 1 0|7 6 5 4:3 2 1 0'
        BYTE    null1;              // Fields:           5 4         F E
        DWORD   strength;           // 0
        DWORD   energy;             // 1
        DWORD   dexterity;          // 2
        DWORD   vitality;           // 3
        DWORD   statsbonus;         // 4
        DWORD   skillbonus;         // 5
        DWORD   health;             // 6
        DWORD   healthmax;          // 7
        DWORD   mana;               // 8
        DWORD   manamax;            // 9
        DWORD   stamina;            // A
        DWORD   staminamax;         // B
        DWORD   level;              // C
        DWORD   experience;         // D
        DWORD   goldperson;         // E
        DWORD   goldstash;          // F
    } gf;
    struct
    {
        BYTE    _if[2];             // 'if'
        BYTE    skills[30];
    } IF;

    struct JMitem*  JMinvitems;
    int             JMinvnum;

};

enum { CNT_NONE, CNT_INVENTORY, CNT_BODY, CNT_STASH, CNT_CUBE, CNT_BELT, CNT_SOCKET, CNT_COPYBUFFER };

enum { CRUDEITEM=0x02,
       USUALITEM=0x04,
       SUPERIORITEM=0x06,
       MAGICITEM=0x08,
       SETITEM=0x0A,
       RAREITEM=0x0C,
       UNIQUEITEM=0x0E };

enum {
    DE_NONE=0,

    DE_MAGIC_PREFIX_MODULO_ZERO,
    DE_MAGIC_PREFIX_MODIFIER_VALMISSING,
    DE_MAGIC_SUFFIX_MODULO_ZERO,
    DE_MAGIC_SUFFIX_MODIFIER_VALMISSING,

    DE_RARE_NAMEPREFIX_MODULO_ZERO,
    DE_RARE_NAMESUFFIX_MODULO_ZERO,
    DE_RARE_PRESUFFIX_MODULO_ZERO,
    DE_RARE_PRESUFFIX_NOPOSSIBLE,
    DE_RARE_PRESUFFIX_MODIFIER_VALMISSING,

    DE_UNIQUE_ITEMCODE,
    DE_SET_ITEMCODE,
};
const char* DecodeErrorString(int Error);

#define PREFIX  0
#define SUFFIX  1

class MagicDecoder
{
public:
    bool        Quick;
    Item*       I;

    // Item magic decoding data is buffered for maximum speed
    DWORD       ItemBuffered;
    DWORD       RareItemCodeBuffered;
    int         ItemBufferedModLevel;

    // Magical Attributes Buffering

    int         nPrefixBuffer,nSuffixBuffer;
    _MagicPreSuffix
```

```
                **PrefixBuffer,**SuffixBuffer;
    void        BuildMagicBuffers();

    // Rare Name Attributes Buffering
    int         nRarePrefixBuffer,nRareSuffixBuffer;
    const _RarePreSuffix
                **RarePrefixBuffer,**RareSuffixBuffer;
    void        BuildRareBuffers();

    BYTE        ZeroMemoryStart;

    // Pointers to Magical Attributes with Magnitude

    // Crude Item Prefix
    const char* CrudePrefix;

    // Magical Items
    int         modMagicPrefix,modMagicSuffix;
    int         modpickMagicPrefix,modpickMagicSuffix;
    _MagicPreSuffix *MagicPrefix,*MagicSuffix;
    int         MagicPrefixMag[4];
    int         MagicSuffixMag[4];

    // Rare Items
    const _RarePreSuffix
                *RarePrefix,*RareSuffix;

    // Set & Unique Items
    int         nRareFix;
    bool        tRareFix[6];
    _MagicPreSuffix
                *RareFix[6];
    int         RareFixMag[6][4];

    const _UniqueItem
                *UniqueItem;
    const _SetItem
                *SetItem;
    int         SetItemNum;
    int         UniqueSetMag[16];

    BYTE        ZeroMemoryEnd;

    // Decoding Procs
    void        DecodeCrude();
    void        DecodeMagical();
    void        DecodeRare();
    void        DecodeUnique();
    void        DecodeSet();

    void        CollectModifiers();

    static char NameTmp[256];
    static char AttrTmp[2048];

public:
    MagicDecoder(Item *I);
    ~MagicDecoder();

    int         DecodeError;
    bool        Decode();
    bool        QuickDecode();

    // Collected Modifiers for public
    int         nMods;
    struct
    {
        DWORD   Code;
        int     Mag;
    }
    Mod[32];

    const char* Name();
    const char* RichAttributes();
};

class DWHistory
{
private:
```

```
    int     ML[HISTORYSTEPS];
    DWORD   DWA[HISTORYSTEPS];
    DWORD   DWB[HISTORYSTEPS];

    int     Top,Bottom,Ptr;
public:
    DWHistory();
    ~DWHistory();

    bool    isNext();
    bool    isBack();

    void    StepBack(Item *I);
    void    StepNext(Item *I);
    void    StepAdd(Item *I);
};

// Item Record IDs
#define IT_BASE         0x0000
#define IT_103          0x1030
#define IT_103EAR       0x1031
#define IT_104EX        0x1040
#define IT_104SM        0x1041
#define IT_104EAR       0x1042

// Base Class
class Item
{
protected:
    // Chained list pointers
    Item*           ListNext;
    Item*           ListPrev;

public:
    // States
    bool            Busy;

    // Constructor & Destructor
                    Item();
    virtual         ~Item();

    // Chained List Functions
    inline Item*    Next()
        { return ListNext; }
    inline Item*    Prev()
        { return ListPrev; }

    int             Count();
    Item*           GetOrdinal(int i);
    void            Attach(Item *i);
    void            Delete();

    // The following virtual functions are filled in derived classes
    // with the apropriate functions. Here only the most basic properties
    // are defined.

    // Item Data Loading and Saving virtual
    virtual bool    LoadItemRecord(BYTE *data);
    virtual BYTE*   GetItemRecord();
    virtual DWORD   ItemRecordLength();
    virtual void    BlankItem();
    virtual int     ItemRecordID()          { return IT_BASE; }
    virtual const char* ItemRecordName()    { return "Base Class"; }

    // Coordinates & Size virtuals
    virtual int     Container() const;
    virtual int     xPos() const;           // logical 0-9
    virtual int     yPos() const;           // logical 0-9
    int             xPixelPos() const;
    int             yPixelPos() const;
    virtual int     xSize();                // logical 1-4
    virtual int     ySize();                // logical 1-4
    int             xPixelSize();
    int             yPixelSize();
    bool            isInRegion(int x,int y);

    virtual bool    SetCoordinates(int Container,int xPos,int yPos);

    virtual int     BodyCode();
```

```
int             BodyPlace();

// ItemInfo handling and creation
ItemInfo*       Info;
virtual bool    FindInfo();
virtual HBITMAP GetBitmap();

// Virtual Simple Item Properties
virtual DWORD   ItemCode();
virtual void    SetItemCode(DWORD x);
virtual int     UniqueCode();
virtual void    SetUniqueCode(int x);

virtual int     Quality();
virtual void    SetQuality(int x);
virtual int     Quantity();
virtual void    SetQuantity(int i);

virtual int     Durability();
virtual void    SetDurability(int x);
virtual int     DurabilityMax();
virtual void    SetDurabilityMax(int x);

virtual DWORD   DWA();
virtual void    SetDWA(DWORD x);
virtual DWORD   DWB();
virtual void    SetDWB(DWORD x);
DWORD           DWARandomOffset(int x);
DWORD           DWBRandomOffset(int x);

virtual int     MagicLevel();
virtual void    SetMagicLevel(int x);

virtual int     GemNum();
virtual void    SetGemNum(int i);

// Virtual Item Property Flags
virtual bool    Socketed();
virtual void    SetSocketed(bool i);
bool            Socketable();
virtual bool    Identified();
virtual void    SetIdentified(bool i);
virtual bool    Starter();
virtual void    SetStarter(bool i);

// Gemmed Items
Item*           Gems;               // Chained List of Socketed Gems
const GemInfo*  GInfo;              // Info about Gem if this is one
virtual bool    FindGemInfo();

// Ear-specific Properties
virtual int     OpponentClass();
virtual void    SetOpponentClass(int x);
virtual const char* OpponentClassString();

virtual int     OpponentLevel();
virtual void    SetOpponentLevel(int x);

virtual const char* OpponentName();
virtual void    SetOpponentName(const char *s);

// Virtual Computed Item Properties
virtual int         BaseDefense();
virtual ArmorClass  Defense();

virtual Damage  WeaponDamage();

virtual unsigned int RequiredStrength();
virtual unsigned int RequiredDexterity();
virtual unsigned int RequiredELevel();

virtual bool    isWearable();

bool            Decoded;
virtual bool    Decode();
class MagicDecoder  *MD;

virtual const char* Name();
virtual const char* RichText();
```

```
    DWHistory       DWBHistory;

protected:
    static char     NameTmp[256];
    static char     RichTextTmp[2048];
    void            StartRTF(char *s);
};

class ItemMg : public Item
{
public:
    // Computed Item Properties
    virtual int BaseDefense();
    virtual ArmorClass Defense();

    virtual Damage WeaponDamage();

    virtual unsigned int RequiredStrength();
    virtual unsigned int RequiredDexterity();
    virtual unsigned int RequiredELevel();

    virtual bool Decode();

    virtual const char* Name();
    virtual const char* RichText();
};

// Item Structure from 1.03
class Item103 : public ItemMg
{
private:
    struct _JMItem
    {
        BYTE    JM[2];              // 'JM'
        WORD    bitfield1;          // contains socket and identification flags
        WORD    bitfield2;          // contains starting item flag
        BYTE    bitfield3;          // contains gem info
        BYTE    modlevel;
        WORD    itemcode;
        WORD    quantity;
        BYTE    quantityex;
        WORD    durability;
        BYTE    coordinates;
        BYTE    specialitemcode;
        DWORD   properties1;
        DWORD   properties2;
        WORD    container;
    } Data;
public:
    Item103();
    ~Item103();

    // Item Data Loading and Saving virtual
    bool        LoadItemRecord(BYTE *data);
    BYTE*       GetItemRecord();
    DWORD       ItemRecordLength();
    void        BlankItem();
    int         ItemRecordID()          { return IT_103; }
    const char* ItemRecordName()        { return "v1.03 Record"; }

    // Coordinates & Size
    int         Container() const;
    int         xPos() const;           // logical 0-9
    int         yPos() const;           // logical 0-9
    int         xSize();                // logical 1-4
    int         ySize();                // logical 1-4
    int         BodyCode();

    bool        SetCoordinates(int Container,int xPos,int yPos);

    // Iteminfo functions
    bool        FindInfo();
    HBITMAP     GetBitmap();

    // Item Properties
    DWORD       ItemCode();
    void        SetItemCode(DWORD x);
    int         UniqueCode();
```

```
    void        SetUniqueCode(int x);

    int         Quality();
    void        SetQuality(int x);
    int         Quantity();
    void        SetQuantity(int i);

    int         GemNum();
    void        SetGemNum(int i);

    int         Durability();
    void        SetDurability(int x);
    int         DurabilityMax();
    void        SetDurabilityMax(int x);

    // Image and Defence Specifier
    DWORD       DWA();
    void        SetDWA(DWORD x);

    // Magical Double Word & Magic Mod Level
    DWORD       DWB();
    void        SetDWB(DWORD x);

    int         MagicLevel();
    void        SetMagicLevel(int x);

    // Flags
    bool        Socketed();
    void        SetSocketed(bool i);
    bool        Socketable();
    bool        Identified();
    void        SetIdentified(bool i);
    bool        Starter();
    void        SetStarter(bool i);

    // Gem Functions
    bool        FindGemInfo();
};

// Ear Item Structure from 1.03
class Item103Ear : public Item
{
private:
    struct _JMItemEar
    {
        BYTE    JM[2];
        WORD    bitfield1;
        WORD    bitfield2;
        WORD    itemcode;
        WORD    coordinates;
        BYTE    container;
        BYTE    string[16];
    } Data;
public:
    Item103Ear();
    ~Item103Ear();

    // Item Data Loading and Saving virtual
    bool        LoadItemRecord(BYTE *data);
    BYTE*       GetItemRecord();
    DWORD       ItemRecordLength();
    void        BlankItem();
    int         ItemRecordID()          { return IT_103EAR; }
    const char* ItemRecordName()        { return "v1.03 Ear Record"; }

    // Coordinates & Size
    int         Container() const;
    int         xPos() const;           // logical 0-9
    int         yPos() const;           // logical 0-9
    int         xSize();                // logical 1-4
    int         ySize();                // logical 1-4

    bool        SetCoordinates(int Container,int xPos,int yPos);

    // Iteminfo functions
    bool        FindInfo();
    HBITMAP     GetBitmap();

    // Item Properties
```

```
    DWORD       ItemCode();
    void        SetItemCode(DWORD x);

    // Ear-specific Properties
    int         OpponentClass();
    void        SetOpponentClass(int x);
    const char* OpponentClassString();

    int         OpponentLevel();
    void        SetOpponentLevel(int x);

    char        OpponentNameChar(int n);
    void        SetOpponentNameChar(int n,char c);

    const char* OpponentName();
    void        SetOpponentName(const char *s);

    // Computed Properties
    const char* Name();
    const char* RichText();
};

class Item104Ex : public ItemMg
{
private:
    struct _JMItem
    {
        BYTE    JM[2];              // 'JM'
        WORD    bitfield1;
        WORD    bitfield2;          // contains starting item flag
        BYTE    bitfield3;
        DWORD   itemcode;
        BYTE    bodycode;
        WORD    magicrestr;
        WORD    quantity;
        BYTE    quantityex;
        WORD    durability;
        BYTE    coordinates;
        BYTE    uniquecode;
        DWORD   DwA;
        DWORD   DwB;
        WORD    container;
    } Data;
public:
    Item104Ex();
    ~Item104Ex();

    // Item Data Loading and Saving virtual
    bool        LoadItemRecord(BYTE *data);
    BYTE*       GetItemRecord();
    DWORD       ItemRecordLength();
    void        BlankItem();
    int         ItemRecordID()          { return IT_104EX; }
    const char* ItemRecordName()        { return "v1.04 Extended Record"; }

    // Coordinates & Size
    int         Container() const;
    int         xPos() const;           // logical 0-9
    int         yPos() const;           // logical 0-9
    int         xSize();                // logical 1-4
    int         ySize();                // logical 1-4
    int         BodyCode();

    bool        SetCoordinates(int Container,int xPos,int yPos);

    // Iteminfo functions
    bool        FindInfo();
    void        CreateItem(ItemInfo *info);
    HBITMAP     GetBitmap();

    // Item Properties
    DWORD       ItemCode();
    void        SetItemCode(DWORD x);
    int         UniqueCode();
    void        SetUniqueCode(int x);

    int         Quality();
    void        SetQuality(int x);
    int         Quantity();
```

```
    void        SetQuantity(int i);

    int         GemNum();
    void        SetGemNum(int i);

    int         Durability();
    void        SetDurability(int x);
    int         DurabilityMax();
    void        SetDurabilityMax(int x);

    // Image and Defence Specifier
    DWORD       DwA();
    void        SetDwA(DWORD x);

    // Magical Double Word & Magic Mod Level
    DWORD       DwB();
    void        SetDwB(DWORD x);

    int         MagicLevel();
    void        SetMagicLevel(int x);

    // Flags
    bool        Socketed();
    void        SetSocketed(bool i);
    bool        Socketable();
    bool        Identified();
    void        SetIdentified(bool i);
    bool        Starter();
    void        SetStarter(bool i);
};

class Item104Sm : public Item
{
private:
    struct _JMItem
    {
        BYTE    JM[2];              // 'JM'
        WORD    bitfield1;
        WORD    bitfield2;          // contains starting item flag
        WORD    bitfield3;
        WORD    coordinates;
        DWORD   itemcode;
        BYTE    zero;
    } Data;
public:
    Item104Sm();
    ~Item104Sm();

    // Item Data Loading and Saving virtual
    bool        LoadItemRecord(BYTE *data);
    BYTE*       GetItemRecord();
    DWORD       ItemRecordLength();
    void        BlankItem();
    int         ItemRecordID()          { return IT_104SM; }
    const char* ItemRecordName()        { return "v1.04 Simple Record"; }

    // Coordinates & Size
    int         Container() const;
    int         xPos() const;           // logical 0-9
    int         yPos() const;           // logical 0-9
    int         xSize();                // logical 1-4
    int         ySize();                // logical 1-4

    bool        SetCoordinates(int Container,int xPos,int yPos);

    // Iteminfo functions
    bool        FindInfo();
    HBITMAP     GetBitmap();

    // Gem Functions
    bool        FindGemInfo();

    // Item Properties
    DWORD       ItemCode();
    void        SetItemCode(DWORD x);

    const char* RichText();
};
```

```
// Ear Item Structure from 1.04
class Item104Ear : public Item
{
private:
    struct _JMItemEar
    {
        BYTE    JM[2];
        WORD    bitfield1;
        WORD    bitfield2;
        WORD    itemcode;
        WORD    coordinates;
        BYTE    string[16];
    } Data;
public:
    Item104Ear();
    ~Item104Ear();

    // Item Data Loading and Saving virtual
    bool        LoadItemRecord(BYTE *data);
    BYTE*       GetItemRecord();
    DWORD       ItemRecordLength();
    void        BlankItem();
    int         ItemRecordID()          { return IT_104EAR; }
    const char* ItemRecordName()        { return "v1.04 Ear Record"; }

    // Coordinates & Size
    int         Container() const;
    int         xPos() const;           // logical 0-9
    int         yPos() const;           // logical 0-9
    int         xSize();                // logical 1-4
    int         ySize();                // logical 1-4

    bool        SetCoordinates(int Container,int xPos,int yPos);

    // Iteminfo functions
    bool        FindInfo();
    HBITMAP     GetBitmap();

    // Item Properties
    DWORD       ItemCode();
    void        SetItemCode(DWORD x);

    // Ear-specific Properties
    int         OpponentClass();
    void        SetOpponentClass(int x);
    const char* OpponentClassString();

    int         OpponentLevel();
    void        SetOpponentLevel(int x);

    char        OpponentNameChar(int n);
    void        SetOpponentNameChar(int n,char c);

    const char* OpponentName();
    void        SetOpponentName(const char *s);

    // Computed Properties
    const char* Name();
    const char* RichText();
};

#pragma pack(4)

extern class fileclass fc;
extern class Item* Items;
extern class Item* CopyBuffer;

extern HBITMAP  ItemInfoGetBitmap(ItemInfo *Info);
extern Item     *CreateItem(Item **IChain,ItemInfo *Info);

extern char buffer[256];

// D2Decode.cpp
struct RAND
{
    DWORD   Seed;
    DWORD   Carry;
};
inline DWORD Random(RAND *rnd)
```

```
{
    DWORDLONG x = rnd->Seed;
    x *= 0x6AC690C5;
    x += rnd->Carry;

    rnd->Seed = DWORD(x);
    rnd->Carry = DWORD(x >> 32);
    return rnd->Seed;
}
extern int      StartRandoms(Item *I,RAND *r);

#pragma pack(1)

#define TAB2COMMON      \
    bool    Running;    \
    DWORD   Counter;    \
    HANDLE  Thread;     \
    HGLOBAL ThreadData; \
    HWND    Dialog;     \
    Item*   Item;       \
    bool    Advanced;   \
    bool    TraverseMagicLevels; \
    bool    HitsSelection; \
    SearchHit* Hits;    \
    DWORD   Past[BFASPEEDPAST];

// Tab2 Search Threads
struct SearchHit
{
    int     MagicLevel;
    int     DWA;
    int     DWB;
    SearchHit*  List;
};

struct SearchThread
{
    TAB2COMMON
};

struct MagicSearchThread
{
    TAB2COMMON

    bool    PrefixMatch;
    _MagicPreSuffix *Prefix;

    bool    SuffixMatch;
    _MagicPreSuffix *Suffix;

    bool    ForcePrefixMatch[4];
    int     ForcePrefixValue[4];

    bool    ForceSuffixMatch[1];
    int     ForceSuffixValue[1];
};

struct RareSearchThread
{
    TAB2COMMON

    int     NamePrefix,NameSuffix;

    _MagicPreSuffix *Attribute[6];
    int     iAttribute[6];
    int     tAttribute[6];
};

struct DefenseSearchThread
{
    TAB2COMMON

    int     SelectAC;
};

struct RingAmuletSearchThread
{
    TAB2COMMON
```

```
    bool    RingAmulet;     // Ring == false; Amulet == true;
    DWORD   Image;
};

#pragma pack(4)

// Macro Functions
inline void PollMessages()
{
    MSG msg;
    while(PeekMessage(&msg,(HWND) NULL,0,0,PM_REMOVE))
    {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
}
inline bool CheckPollMessages()
{
    MSG msg;
    while(PeekMessage(&msg,(HWND) NULL,0,0,PM_REMOVE))
    {
        if (msg.message == WM_QUIT)
            return true;

        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
    return false;
}

inline void ErrorBox(const char *error,HWND hWnd = hMainDialog)
{
    MessageBox(hWnd,error,PROGRAMNAME,MB_OK | MB_ICONHAND | MB_APPLMODAL);
}
#if DEVELOPMENT == 1
#define AssertBox(str)  ErrorBox(str)
#else
#define AssertBox(str)
#endif

inline const PopupHelp* FindPopupHelp(int ID)
{
    for(int z=0;z<nPopupHelps;z++)
    {
        if (PopupHelps[z].CtrlID == ID)
            return &PopupHelps[z];
    }
    return 0;
}

// Info Macros
inline int FindItemInfoByItemCode(DWORD ItemCode)
{
    for(int z=0;z<nItemInfos;z++)
    {
        if (ItemInfos[z].ItemCode == ItemCode)
            return z;
    }
    return -1;
}

inline int FindMagicPreSuffixByPointer(bool PreSuff,const _MagicPreSuffix* p)
{
    if (PreSuff == PREFIX) {
        for(int z=0;z<nMagicPrefixTable;z++)
        {
            if (&MagicPrefixTable[z] == p)
                return z;
        }
        return -1;
    }
    else {
        for(int z=0;z<nMagicSuffixTable;z++)
        {
            if (&MagicSuffixTable[z] == p)
                return z | 256;
        }
        return -1;
    }
```

```
}
inline int FindRareNamePrefixByPointer(const _RarePreSuffix* p)
{
    for(int z=0;z<nRarePrefixTable;z++)
    {
        if (&RarePrefixTable[z] == p)
            return z;
    }
    return -1;
}
inline int FindRareNameSuffixByPointer(const _RarePreSuffix* p)
{
    for(int z=0;z<nRareSuffixTable;z++)
    {
        if (&RareSuffixTable[z] == p)
            return z;
    }
    return -1;
}

inline const char* GetEffect(DWORD code)
{
    if (code == 0) return "Zero String error";
    for(int z=0;z<nModifiers;z++)
    {
        if (Modifiers[z].Code == code && Modifiers[z].Text)
            return Modifiers[z].Text;
    }
    return "Unknown Effect";
}
extern char CodeStringTemp[5];
inline const char* CodeString(DWORD code)
{
    CodeStringTemp[0] = char((code & 0x000000FF) >> 0);
    CodeStringTemp[1] = char((code & 0x0000FF00) >> 8);
    CodeStringTemp[2] = char((code & 0x00FF0000) >> 16);
    CodeStringTemp[3] = char((code & 0xFF000000) >> 24);
    CodeStringTemp[4] = 0;
    return CodeStringTemp;
}
inline const char* CodeStringRev(DWORD code)
{
    CodeStringTemp[0] = char((code & 0xFF000000) >> 24);
    CodeStringTemp[1] = char((code & 0x00FF0000) >> 16);
    CodeStringTemp[2] = char((code & 0x0000FF00) >> 8);
    CodeStringTemp[3] = char((code & 0x000000FF) >> 0);
    CodeStringTemp[4] = 0;
    return CodeStringTemp;
}

// Diablo 2 Directory Key
inline bool GetDiabloSaveDirectory(char *dest)
{
    HKEY regkey;
    if (RegOpenKeyEx(HKEY_CURRENT_USER,"Software\\Blizzard Entertainment\\Diablo II",0,KEY_READ,&regkey) == ERROR_SUCCES
S)
    {
        DWORD type = REG_SZ;
        DWORD fdirsize = 260;
        DWORD x = RegQueryValueEx(regkey,"Save Path",0,&type,(unsigned char*)dest,&fdirsize);
        if (x != ERROR_SUCCESS)
            return false;
        RegCloseKey(regkey);
        return true;
    }
    return false;
}

// strcat Multiple
inline void strmcat(char *d,...)
{
    va_list args;
    va_start(args,d);

    char *s;

    while( s = va_arg(args,char *) )
        strcat(d,s);
```

```
    va_end(args);
}

// ASCIItoRTF
inline void ASCIItoRTF(char *d,const char *s)
{
    while(*s)
    {
        if (*s == '\n') {
            *d++ = '\\';
            *d++ = 'p';
            *d++ = 'a';
            *d++ = 'r';
            *d++ = ' ';
            s++;
            if (*s == '\r') s++;
        }
        else
            *d++ = *s++;
    }
    *d = 0;
}

// Item Class Duplicator
inline Item *ItemDuplicate(Item *I)
{
    switch(I->ItemRecordID())
    {
    default:            return new Item;
    case IT_103:        return new Item103;
    case IT_103EAR:     return new Item103Ear;
    case IT_104EX:      return new Item104Ex;
    case IT_104SM:      return new Item104Sm;
    case IT_104EAR:     return new Item104Ear;
    }
}
```

```
// CommandLine.cpp

#include "JamellaD2E.h"

static char ProgramFilePathBuffer[260];
static char InitialLoadFileBuffer[260];

void ParseCommandLine()
{
    WCHAR *cl = GetCommandLineW();
    char BufferA[1024];

    WideCharToMultiByte(CP_ACP,0,cl,-1,BufferA,sizeof BufferA,NULL,NULL);

    char *s = BufferA;
    char *d = ProgramFilePathBuffer;
    int QuoteDepth = 0;
    int cln = 0;

    while(*s != 0)
    {
        if (*s == '"' && QuoteDepth <= 0)
        {
            QuoteDepth++;
            s++;
        }
        else if (*s == '"' && QuoteDepth > 0)
        {
            QuoteDepth--;
            s++;
        }
        else if (*s == ' ' && QuoteDepth <= 0)
        {
            if (d)
                *d = 0;

            cln++;

            if (cln == 1)
                d = InitialLoadFileBuffer;
            else
                d = 0;

            s++;
        }
        else
        {
            if (d)
                *d++ = *s++;
            else
                s++;
        }
    }

}

const char *CmdLineFile()
{
    if (InitialLoadFileBuffer[0]) {
        char *s = InitialLoadFileBuffer;

        while(*s != 0) {
            s++;
            if (toupper(*(s+0)) != '.') continue;
            if (toupper(*(s+1)) != 'D') continue;
            if (toupper(*(s+2)) != '2') continue;
            if (toupper(*(s+3)) != 'S') continue;
            return InitialLoadFileBuffer;
        }

        return 0;
    }
    else
        return 0;
}

const char *ProgramFilePath()
{
    return ProgramFilePathBuffer;
```

```
}
```

```
// CowLevel.cpp from D2E

#include "JamellaD2E.h"

LRESULT CALLBACK CowLevelDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
    switch(uMsg)
    {
    case WM_CREATE:
        return true;

    case WM_COMMAND:
        switch (LOWORD(wParam))
        {
        case IDOK:
        case IDCANCEL:
            EndDialog(hWnd,wParam);
            return true;
        }
        break;
    case WM_CLOSE:
        EndDialog(hWnd,wParam);
        return true;
    case WM_DESTROY:
        return false;
    }
    return false;
}
```

```
// CreateItem.cpp from D2E
// Contains the function that creates item from item info

#include "JamellaD2E.h"

int MaximumItemStructure()
{
    int m = -1;
    for(Item *I = Items;I != 0;I = I->Next())
        if (I->ItemRecordID() > m)
            m = I->ItemRecordID();
    return m;
}

Item* CreateItem(Item **IChain,ItemInfo *Info)
{
    Item *I = 0;

    int toCreate = IT_BASE;

    if (RegOptions.CreateItemRecordFormat == 1) // 1.03 Records
    {
        if (Info->IC != ' rae')
            toCreate = IT_103;
        else if (Info->IC == ' rae')
            toCreate = IT_103EAR;
    }
    else if (RegOptions.CreateItemRecordFormat == 2) // 1.04 Records
    {
        if (Info->IC != ' rae') {
            if (!Info->Sm)
                toCreate = IT_104EX;
            else
                toCreate = IT_104SM;
        }
        else if (Info->IC == ' rae')
            toCreate = IT_104EAR;
    }
    else // Auto-detect Record
    {
        int m = MaximumItemStructure();

        if (m < IT_104EX && Info->IC != ' rae')
            toCreate = IT_103;
        else if (m < IT_104EX && Info->IC == ' rae')
            toCreate = IT_103EAR;
        else if (m >= IT_104EX && Info->IC != ' rae') {
            if (!Info->Sm)
                toCreate = IT_104EX;
            else
                toCreate = IT_104SM;
        }
        else if (m >= IT_104EX && Info->IC == ' rae')
            toCreate = IT_104EAR;
    }

    switch(toCreate)
    {
    case IT_103:
        { // Create Item103

            I = new Item103;
            if (*IChain)
                (*IChain)->Attach(I);
            else
                *IChain = I;

            I->BlankItem();

            I->SetItemCode(Info->ItemCode);
            I->SetUniqueCode(Info->UniqueCode);
            I->SetIdentified(true);

            switch(Info->ItemSearch)
            {
            case 'N':   I->SetQuality(USUALITEM);           break;
            case 'M':   I->SetQuality(MAGICITEM);           break;
            case 'S':   I->SetQuality(SETITEM);             break;
            case 'U':   I->SetQuality(UNIQUEITEM);          break;
```

```
        }

        I->SetDurabilityMax(Info->Durability);
        I->SetDurability(Info->Durability);
        I->SetQuantity(Info->Quantity);

        I->SetMagicLevel((rand() % 90) + 1);

        I->SetDWA(rand() + (rand() << 16));
        I->SetDWB(rand() + (rand() << 16));

        I->Info = Info;

        if (Info->DWBCode != 0 && I->Quality() == SETITEM)
        {
            for(int z=0;z<20000;z++)
            {
                RAND RDWB = { I->DWB(), 666 };
                StartRandoms(I,&RDWB);

                int offset = Random(&RDWB) % 0x10;
                if (( Info->DWBCode & (1 << offset) ) != 0) break;

                I->SetDWB(rand() + (rand() << 16));
            }
        }

        I->FindInfo();

    } // Create Item103
    break;
case IT_103EAR:
    { // Create Item103Ear

        I = new Item103Ear;

        if (*IChain)
            (*IChain)->Attach(I);
        else
            *IChain = I;

        I->BlankItem();

        I->SetItemCode(Info->ItemCode);
        I->SetOpponentName("SetNewName");

    } // Create Item103Ear
    break;
case IT_104EX:
    { // Create Item104Ex

        I = new Item104Ex;
        if (*IChain)
            (*IChain)->Attach(I);
        else
            *IChain = I;

        I->BlankItem();

        I->SetItemCode(Info->IC);
        I->SetUniqueCode(Info->UniqueCode);
        I->SetIdentified(true);

        switch(Info->ItemSearch)
        {
        case 'N':   I->SetQuality(USUALITEM);           break;
        case 'M':   I->SetQuality(MAGICITEM);           break;
        case 'S':   I->SetQuality(SETITEM);             break;
        case 'U':   I->SetQuality(UNIQUEITEM);          break;
        }

        I->SetDurabilityMax(Info->Durability);
        I->SetDurability(Info->Durability);
        I->SetQuantity(Info->Quantity);

        I->SetMagicLevel((rand() % 90) + 1);

        I->SetDWA(rand() + (rand() << 16));
        I->SetDWB(rand() + (rand() << 16));
```

```
            I->Info = Info;

            if (Info->DWBCode != 0 && I->Quality() == SETITEM)
            {
                for(int z=0;z<20000;z++)
                {
                    RAND RDWB = { I->DWB(), 666 };
                    StartRandoms(I,&RDWB);

                    int offset = Random(&RDWB) % 0x10;
                    if (( Info->DWBCode & (1 << offset) ) != 0)
                        break;

                    I->SetDWB(rand() + (rand() << 16));
                }
            }

            I->FindInfo();

        } // Create Item104Ex
        break;
    case IT_104SM:
        { // Create Item104Sm

            I = new Item104Sm;
            if (*IChain)
                (*IChain)->Attach(I);
            else
                *IChain = I;

            I->BlankItem();

            I->SetItemCode(Info->IC);

            I->FindInfo();

        } // Create Item104Sm
        break;
    case IT_104EAR:
        { // Create Item104Ear

            I = new Item104Ear;

            if (*IChain)
                (*IChain)->Attach(I);
            else
                *IChain = I;

            I->BlankItem();
            I->SetOpponentName("SetNewName");

        } // Create Item104Ear
        break;
    }

    return I;
}
```

```
#include "JamellaD2E.h"

int StartRandoms(Item *I,RAND *r)
{
    if (!I->Info)
        if (!I->FindInfo())
            return 0;

    r->Seed = I->DWB();
    r->Carry = 666;

    int     UsualAdd[6] = { 1000, 200, 125, 30,  12, 4 };
    int     UsualDiv[6] = { 1,    1,   1,   16,  16, 8 };
    int     ExcepAdd[6] = { 600,  120, 100, 3,   4,  1 };
    int     ExcepDiv[6] = { 1,    1,   1,   100, 16, 8 };

    int     *Add = (I->Info->Quality == 'E') ? ExcepAdd : UsualAdd;
    int     *Div = (I->Info->Quality == 'E') ? ExcepDiv : UsualDiv;

    int rands = 0;
    for(int z=0;z<6;z++)
    {
        int modulo = Add[z] - (I->MagicLevel() / Div[z]);

        if (modulo <= 0) modulo = 1;

        rands++;
        if ((Random(r) % modulo) == 0) break;
    }
    return rands;
}
```

```
#include "JamellaD2E.h"

char buffer[256];

ItemInfo *SelInfo = 0;

const char* CharClasses[5] =
{ "Amazon","Sorceress","Necromancer","Paladin","Barbarian" };
const char* Difficulties[4] =
{ "Normal","Nightmare","Hell","Hell done" };
```

```
// FileClass.cpp from D2E

#include "JamellaD2E.h"

class fileclass fc;

// Character Structs
const char Sign_Header[9]       = "\x55\xAA\x55\xAA\x47\x00\x00\x00";
const char Sign_Woo[9]          = "\x57\x6F\x6F\x21\x06\x00\x00\x00";
const char Sign_WS[9]           = "\x57\x53\x01\x00\x00\x00\x50\x00";
const char Sign_w4[5]           = "\x01\x77\x34\x00";
const char Sign_gf[3]           = "\x67\x66";
const char Sign_if[3]           = "\x69\x66";
const char Sign_JM[3]           = "\x4A\x4D";

BYTE*           rest;
int             restsize;

fileclass::fileclass()
{
    clear();
}
fileclass::~fileclass()
{
    clear();
}
void fileclass::clear()
{
    loaded = false;
    ZeroMemory(&Header,sizeof(Header));
    ZeroMemory(&Woo,sizeof(Woo));
    ZeroMemory(&WS,sizeof(WS));
    ZeroMemory(&w4,sizeof(w4));
    ZeroMemory(&gf,sizeof(gf));
    ZeroMemory(&IF,sizeof(IF));
    ZeroMemory(&JMinv,sizeof(JMinv));
    if (Items)
    {
        delete Items;
        Items = 0;
    }
    if (rest)
    {
        delete [restsize] rest;
        rest = 0;
        restsize = 0;
    }
}
int fileclass::transferdata(BYTE *data,int size)
{
    int JMn = 0;

    for(BYTE* a = data;a < data+size;)
    {
        char* Sign = (char *)a;

        if (strncmp(Sign,Sign_Header,sizeof Sign_Header-1) == 0)
        {
            memcpy(&Header,a,sizeof Header);
            a += sizeof Header;
        }
        else if (strncmp(Sign,Sign_Woo,sizeof Sign_Woo-1) == 0)
        {
            memcpy(&Woo,a,sizeof Woo);
            a += sizeof Woo;
        }
        else if (strncmp(Sign,Sign_WS,sizeof Sign_WS-1) == 0)
        {
            memcpy(&WS,a,sizeof WS);
            a += sizeof WS;
        }
        else if (strncmp(Sign,Sign_w4,sizeof Sign_w4-1) == 0)
        {
            memcpy(&w4,a,sizeof w4);
            a += sizeof w4;
        }
        else if (strncmp(Sign,Sign_gf,sizeof Sign_gf-1) == 0)
        {
            memset(&gf,0,sizeof gf);
```

```
    memcpy(&gf,a,sizeof 5);     // sig & field info
    a += 5;
    WORD fields = gf.fields;

    for(int i=0;i<16;i++)
    {
        if (fields & 1)
        {
            ((DWORD*)&gf.strength)[i] = *(DWORD*)a;
            a += 4;
        }
        fields /= 2;
    }
    gf.health /= 256;
    gf.healthmax /= 256;
    gf.mana /= 256;
    gf.manamax /= 256;
    gf.stamina /= 256;
    gf.staminamax /= 256;
}
else if (strncmp(Sign,Sign_if,sizeof Sign_if-1) == 0)
{
    memcpy(&IF,a,sizeof IF);
    a += sizeof IF;
}
else if (strncmp(Sign,Sign_JM,sizeof Sign_JM-1) == 0)
{
    if (JMn == 0)
    { // first entry

        memcpy(&JMinv,a,sizeof JMinv);
        a += sizeof JMinv;
        JMn++;

    } // first entry
    else if (Items->Count() < JMinv.num)
    { // this belongs into the inventory

        struct
        {
            char    JM[2];
            WORD    unimportant;
            WORD    type;
        } ItemHead;

        memcpy(&ItemHead,a,sizeof ItemHead);

        class Item *I = 0;

        if ((ItemHead.type & 0x0039) == 0x0000) // 1.03 Item Data Type
        {
            I = new Item103;
        }
        else if ((ItemHead.type & 0x0039) == 0x0001) // 1.03 Ear Item Data Type
        {
            I = new Item103Ear;
        }
        else if ((ItemHead.type & 0x0039) == 0x0018) // 1.04 Extended Struct
        {
            I = new Item104Ex;
        }
        else if ((ItemHead.type & 0x0039) == 0x0038) // 1.04 Simple Struct
        {
            I = new Item104Sm;
        }
        else if ((ItemHead.type & 0x0039) == 0x0039) // 1.04 Ear Struct
        {
            I = new Item104Ear;
        }
        else {
            return a-data;
        }

        if (Items) Items->Attach(I);
        else Items = I;

        I->LoadItemRecord(a);
        a += I->ItemRecordLength();
```

```
                if (I->Socketed())
                {// Read additional socketed gem records

                    for(int g=0;g < I->GemNum();g++)
                    {
                        memcpy(&ItemHead,a,sizeof ItemHead);

                        class Item *G = 0;

                        if ((ItemHead.type & 0x0039) == 0x0000) // 1.03 Item Data Type
                        {
                            G = new Item103;
                        }
                        else if ((ItemHead.type & 0x0039) == 0x0038) // 1.04 Simple Struct
                        {
                            G = new Item104Sm;
                        }
                        else {
                            return a-data;
                        }

                        if (I->Gems == 0) I->Gems = G;
                        else I->Gems->Attach(G);

                        G->LoadItemRecord(a);
                        a += G->ItemRecordLength();
                    }

                }// Read additional socketed gem records

            } // this belongs into the inventory
            else
            { // rest data
                restsize = data+size-a;
                rest = new BYTE [restsize];
                memcpy(rest,a,restsize);
                a += restsize;
            } // rest data
        }
        else
            return a-data;
    }
    return -1;
}
bool fileclass::loadfile(HWND hWnd,const char* tempfilename)
{
    if (loaded)
    {
        return false;
    }

    // Open File
    HANDLE hFile = CreateFile(tempfilename,
        GENERIC_READ,FILE_SHARE_READ,NULL,
        OPEN_EXISTING,FILE_ATTRIBUTE_NORMAL,NULL);

    if (hFile == INVALID_HANDLE_VALUE)
    {
        MessageBox(hWnd,"Could not open file!",PROGRAMNAME,
            MB_OK | MB_ICONSTOP | MB_APPLMODAL);
        return false;
    }

    DWORD filesize = GetFileSize(hFile,NULL);

    BYTE* filedata = new BYTE [filesize];
    DWORD fileread;

    // read whole file
    ReadFile(hFile,filedata,filesize,&fileread,NULL);

    if (filesize != fileread)
    {
        delete filedata;
        CloseHandle(hFile);
        MessageBox(hWnd,"Could not read from file!",PROGRAMNAME,
            MB_OK | MB_ICONSTOP | MB_APPLMODAL);
        return false;
    }
```

```
    int erroffset = transferdata(filedata,filesize);

    if (erroffset >= 0)
    {
        delete filedata;
        CloseHandle(hFile);
        char text[80];

        sprintf(text,"Corrupt field encountered in save game file! (@ offset %i)",erroffset);
        MessageBox(hWnd,text,PROGRAMNAME,
            MB_OK | MB_ICONSTOP | MB_APPLMODAL);
        clear();
        return false;
    }

    delete filedata;
    CloseHandle(hFile);

    loaded = true;
    memcpy(filename,tempfilename,sizeof filename);
    return true;
}
bool fileclass::reloadfile(HWND hWnd)
{
    clear();
    return loadfile(hWnd,filename);
}
DWORD   offset;
inline bool check_written(HWND hWnd,DWORD a,DWORD b,HANDLE hFile)
{
    if (a != b)
    {
        sprintf(buffer,"Could not write to file at offset %i!",offset);
        MessageBox(hWnd,buffer,PROGRAMNAME,
            MB_OK | MB_ICONSTOP | MB_APPLMODAL);
        CloseHandle(hFile);
        return true;
    }
    else return false;
}

bool fileclass::savefile(HWND hWnd)
{
    if (!loaded)
    {
        MessageBox(hWnd,"You didn't open a file!",PROGRAMNAME,
            MB_OK| MB_ICONEXCLAMATION | MB_APPLMODAL);
        return false;
    }

    int user = DialogBox(hInstance,MAKEINTRESOURCE(IDD_SAVE),hWnd,(DLGPROC) SaveDialogProc);

    if (user == 0) return false;

    if (user == 2)
    { // Copy saved game file

        char newfilename[260];
        strcpy(newfilename,filename);
        strupr(newfilename);

        char *testd2s,*lastd2s = newfilename;
        while( testd2s = strstr(lastd2s,".D2S") )
            lastd2s = testd2s + 1;

        strncpy(lastd2s,"JAM",3);

        if (!CopyFile(filename,newfilename,FALSE))
        {
            MessageBox(hWnd,"Could not make backup copy!\nStopping save.",PROGRAMNAME,
                MB_OK| MB_ICONEXCLAMATION | MB_APPLMODAL);
            return false;
        }

    } // Copy saved game file

    HANDLE hFile = CreateFile(filename,
        GENERIC_WRITE,0,NULL,
```

```
    CREATE_ALWAYS,FILE_ATTRIBUTE_NORMAL,NULL);

if (hFile == INVALID_HANDLE_VALUE)
{
    MessageBox(hWnd,"Could not create file!",PROGRAMNAME,
        MB_OK | MB_ICONSTOP | MB_APPLMODAL);
    return false;
}

offset = 0;
DWORD written;

WriteFile(hFile,&Header,sizeof Header,&written,NULL);
if (check_written(hWnd,sizeof Header,written,hFile)) return false;
offset += written;

WriteFile(hFile,&Woo,sizeof Woo,&written,NULL);
if (check_written(hWnd,sizeof Woo,written,hFile)) return false;
offset += written;

WriteFile(hFile,&WS,sizeof WS,&written,NULL);
if (check_written(hWnd,sizeof WS,written,hFile)) return false;
offset += written;

WriteFile(hFile,&w4,sizeof w4,&written,NULL);
if (check_written(hWnd,sizeof w4,written,hFile)) return false;
offset += written;

{
    gf.health *= 256;
    gf.healthmax *= 256;
    gf.mana *= 256;
    gf.manamax *= 256;
    gf.stamina *= 256;
    gf.staminamax *= 256;

    DWORD *a = &gf.strength;

    BYTE gftmp[70];
    BYTE *d = (gftmp+5);

    *(WORD*)(&gftmp[0]) = 0x6667;
    *(WORD*)(&gftmp[2]) = 0x0000;

    WORD *fields = (WORD*)(&gftmp[2]);

    for(int i=0;i<16;i++)
    {
        if (a[i] != 0)
        {
            *fields |= 1 << i;
            *(DWORD*)d = a[i];
            d += 4;
        }
    }

    WriteFile(hFile,&gftmp,d - gftmp,&written,NULL);
    offset += written;

    gf.health /= 256;
    gf.healthmax /= 256;
    gf.mana /= 256;
    gf.manamax /= 256;
    gf.stamina /= 256;
    gf.staminamax /= 256;
}

WriteFile(hFile,&IF,sizeof IF,&written,NULL);
if (check_written(hWnd,sizeof IF,written,hFile)) return false;
offset += written;

{ // Write Inventory

    JMinv.num = Items->Count();

    WriteFile(hFile,&JMinv,sizeof JMinv,&written,NULL);
    if (check_written(hWnd,sizeof JMinv,written,hFile)) return false;
    offset += written;
```

```
        for(Item *i = Items;i != 0;i = i->Next())
        {
            WriteFile(hFile,i->GetItemRecord(),i->ItemRecordLength(),&written,NULL);
            if (check_written(hWnd,i->ItemRecordLength(),written,hFile)) return false;
            offset += written;

            if (i->Socketed())
            {
                Item *G = i->Gems;

                for(int g=0;g < i->GemNum();g++)
                {
                    if (!G)
                    {
                        MessageBox(hWnd,"FATAL Gemmed Item Error!",PROGRAMNAME,
                            MB_OK | MB_ICONSTOP | MB_APPLMODAL);
                        return false;
                    }

                    WriteFile(hFile,G->GetItemRecord(),G->ItemRecordLength(),&written,NULL);
                    if (check_written(hWnd,G->ItemRecordLength(),written,hFile)) return false;
                    offset += written;

                    G = G->Next();
                }
            }
        }
    }

    WriteFile(hFile,rest,restsize,&written,NULL);
    if (check_written(hWnd,restsize,written,hFile)) return false;
    offset += written;

    CloseHandle(hFile);
    MessageBeep(MB_ICONASTERISK);
    return true;
}

bool fileclass::discardfile(HWND hWnd)
{
    clear();
    return true;
}
void fileclass::setfilename(const char *file)
{
    char fdir[260];
    if (GetDiabloSaveDirectory(fdir))
    {
        sprintf(filename,"%s\\%s.d2s",fdir,file);
    }
}
```

```
#include "JamellaD2E.h"

const struct GemInfo GemInfos[] = {
{ "Empty",0x0000,0xFFFFFFFF,0,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },
{ "Chipped Amethyst",0x13C0,0x20766367,18,3,'ht00',15,19,0,0,0,0,0,0,'st00',3,3,0,0,0,0,0,0,'ar00',5,6,0,0,0,0,0,0 },
{ "Chipped Topaz",0x1410,0x20796367,14,3,'dl00',1,1,'dl01',6,6,0,0,0,'ib01',7,9,0,0,0,0,0,0,'rl00',10,12,0,0,0,0,0,0 },
{ "Chipped Sapphire",0x1460,0x20626367,5,3,'dc00',1,1,'dc01',3,3,'dc03',25,25,'mm00',6,8,0,0,0,0,0,0,'rc00',10,12,0,0,0,
0,0,0 },
{ "Chipped Emerald",0x14B0,0x20676367,11,3,'dp00',8,8,'dp01',8,8,'dp02',75,75,'dx00',3,3,0,0,0,0,0,0,'rp00',10,12,0,0,0,
0,0,0 },
{ "Chipped Ruby",0x1500,0x20726367,8,3,'df00',3,3,'df01',4,4,0,0,0,'mh00',6,8,0,0,0,0,0,0,'rf00',10,12,0,0,0,0,0,0 },
{ "Chipped Diamond",0x1550,0x20776367,1,3,'md01',25,29,0,0,0,0,0,0,'ht00',10,10,0,0,0,0,0,0,'ra00',5,6,0,0,0,0,0,0 },
{ "Chipped Skull",0x1640,0x20636B73,2,3,'mm03',1,1,'mh03',2,2,0,0,0,'lf00',2,2,'hl11',8,8,0,0,0,'at00',2,2,0,0,0,0,0,0 }
,
{ "Flawed Amethyst",0x13D0,0x20766667,18,3,'ht00',20,29,0,0,0,0,0,0,'st00',4,4,0,0,0,0,0,0,'ar00',7,8,0,0,0,0,0,0 },
{ "Flawed Topaz",0x1420,0x20796667,14,3,'dl00',1,1,'dl01',7,7,0,0,0,'ib01',11,13,0,0,0,0,0,0,'rl00',13,16,0,0,0,0,0,0 },
{ "Flawed Sapphire",0x1470,0x20626667,5,3,'dc00',2,2,'dc01',3,3,'dc03',35,35,'mm00',9,11,0,0,0,0,0,0,'rc00',13,16,0,0,0,
0,0,0 },
{ "Flawed Emerald",0x14C0,0x20676667,11,3,'dp00',11,11,'dp01',11,11,'dp02',75,75,'dx00',4,4,0,0,0,0,0,0,'rp00',13,16,0,0
,0,0,0,0 },
{ "Flawed Ruby",0x1510,0x20726667,8,3,'df00',3,3,'df01',5,5,0,0,0,'mh00',9,11,0,0,0,0,0,0,'rf00',13,16,0,0,0,0,0,0 },
{ "Flawed Diamond",0x1560,0x20776667,1,3,'md01',30,35,0,0,0,0,0,0,'ht00',15,15,0,0,0,0,0,0,'ra00',7,8,0,0,0,0,0,0 },
{ "Flawed Skull",0x1650,0x20666B73,2,3,'mm03',2,2,'mh03',2,2,0,0,0,'lf00',3,3,'hl11',8,8,0,0,0,'at00',3,3,0,0,0,0,0,0 },
{ "Amethyst",0x13E0,0x20767367,18,3,'ht00',30,39,0,0,0,0,0,0,'st00',5,6,0,0,0,0,0,0,'ar00',9,10,0,0,0,0,0,0 },
{ "Topaz",0x1430,0x20797367,14,3,'dl00',1,1,'dl01',8,8,0,0,0,'ib01',14,17,0,0,0,0,0,0,'rl00',17,20,0,0,0,0,0,0 },
{ "Sapphire",0x1480,0x20627367,5,3,'dc00',2,2,'dc01',4,4,'dc03',50,50,'mm00',12,15,0,0,0,0,0,0,'rc00',17,20,0,0,0,0,0,0
},
{ "Emerald",0x14D0,0x20677367,11,3,'dp00',14,14,'dp01',14,14,'dp02',75,75,'dx00',5,6,0,0,0,0,0,0,'rp00',17,20,0,0,0,0,0,
0 },
{ "Ruby",0x1520,0x20727367,8,3,'df00',4,4,'df01',5,5,0,0,0,'mh00',12,15,0,0,0,0,0,0,'rf00',17,20,0,0,0,0,0,0 },
{ "Diamond",0x1570,0x20777367,1,3,'md01',36,45,0,0,0,0,0,0,'ht00',20,20,0,0,0,0,0,0,'ra00',9,11,0,0,0,0,0,0 },
{ "Skull",0x1660,0x20756B73,2,3,'mm03',2,2,'mh03',3,3,0,0,0,'lf00',3,3,'hl11',12,12,0,0,0,'at00',4,4,0,0,0,0,0,0 },
{ "Flawless Amethyst",0x13F0,0x20767A67,18,3,'ht00',40,49,0,0,0,0,0,0,'st00',7,8,0,0,0,0,0,0,'ar00',11,13,0,0,0,0,0,0 },
{ "Flawless Topaz",0x1440,0x20796C67,14,3,'dl00',1,1,'dl01',10,10,0,0,0,'ib01',18,20,0,0,0,0,0,0,'rl00',21,25,0,0,0,0,0,
0 },
{ "Flawless Sapphire",0x1490,0x20626C67,5,3,'dc00',3,3,'dc01',5,5,'dc03',60,60,'mm00',16,20,0,0,0,0,0,0,'rc00',21,25,0,0
,0,0,0,0 },
{ "Flawless Emerald",0x14E0,0x20676C67,11,3,'dp00',17,17,'dp01',17,17,'dp02',75,75,'dx00',7,8,0,0,0,0,0,0,'rp00',21,25,0
,0,0,0,0,0 },
{ "Flawless Ruby",0x1530,0x20726C67,8,3,'df00',5,5,'df01',6,6,0,0,0,'mh00',16,20,0,0,0,0,0,0,'rf00',21,25,0,0,0,0,0,0 },
{ "Flawless Diamond",0x1580,0x20776C67,1,3,'md01',46,55,0,0,0,0,0,0,'ht00',25,25,0,0,0,0,0,0,'ra00',12,15,0,0,0,0,0,0 },
{ "Flawless Skull",0x1670,0x206C6B73,2,3,'mm03',3,3,'mh03',3,3,0,0,0,'lf00',4,4,'hl11',12,12,0,0,0,'at00',5,5,0,0,0,0,0,
0 },
{ "Perfect Amethyst",0x1400,0x20767067,17,3,'ht00',50,60,0,0,0,0,0,0,'st00',9,10,0,0,0,0,0,0,'ar00',15,17,0,0,0,0,0,0 },
{ "Perfect Topaz",0x1450,0x20797067,13,3,'dl00',1,1,'dl01',12,15,0,0,0,'ib01',21,25,0,0,0,0,0,0,'rl00',26,30,0,0,0,0,0,0
 },
{ "Perfect Sapphire",0x14A0,0x20627067,6,3,'dc00',3,3,'dc01',6,7,'dc03',75,75,'mm00',21,25,0,0,0,0,0,0,'rc00',26,30,0,0,
0,0,0,0 },
{ "Perfect Emerald",0x14F0,0x20677067,12,3,'dp00',20,20,'dp01',20,20,'dp02',75,75,'dx00',9,10,0,0,0,0,0,0,'rp00',26,30,0
,0,0,0,0,0 },
{ "Perfect Ruby",0x1540,0x20727067,9,3,'df00',6,6,'df01',7,10,0,0,0,'mh00',21,25,0,0,0,0,0,0,'rf00',26,30,0,0,0,0,0,0 },
{ "Perfect Diamond",0x1590,0x20777067,0,3,'md01',56,70,0,0,0,0,0,0,'ht00',30,30,0,0,0,0,0,0,'ra00',16,20,0,0,0,0,0,0 },
{ "Perfect Skull",0x1680,0x207A6B73,3,3,'mm03',3,3,'mh03',4,4,0,0,0,'lf00',4,5,'hl11',16,20,0,0,0,'at00',6,7,0,0,0,0,0,0
 },
};

int nGemInfos = sizeof GemInfos / sizeof GemInfos[0];
```

```
// HelpBox.cpp from D2E

#include "JamellaD2E.h"

HWND hHelpBox = 0;

LRESULT CALLBACK HelpDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
    switch(uMsg)
    {
    case WM_INITDIALOG:
        {
            // Position Help Box right of Parent Window
            {
                RECT ParentPos;

                GetWindowRect(GetParent(hWnd),&ParentPos);
                ParentPos.left = ParentPos.right;

                SetWindowPos(hWnd,HWND_TOP,
                    ParentPos.left,ParentPos.top,
                    0,0,SWP_NOSIZE | SWP_NOACTIVATE);
            }

            // Load Help File from Resources
            HRSRC rcsrc = FindResource(hInstance,MAKEINTRESOURCE(lParam),"CHELP");
            HGLOBAL hglb = LoadResource(hInstance,rcsrc);
            BYTE *mem = (BYTE*)LockResource(hglb);

            RTFStreamSend(hWnd,IDC_HELP_Text,(const char*)mem);

            CHARRANGE cf = { -1,0 };
            SendDlgItemMessage(hWnd,IDC_HELP_Text,EM_EXSETSEL,0,(LPARAM)&cf);

            ShowWindow(hWnd,SW_SHOWNA);
        }
        return true;
    case WM_WINDOWPOSCHANGED:
        {
            RECT Rect;
            GetClientRect(hWnd,&Rect);
            SetWindowPos(GetDlgItem(hWnd,IDC_HELP_Text),HWND_TOP,0,0,Rect.right - Rect.left,Rect.bottom - Rect.top,SWP_N
OACTIVATE);
        }
        return false;
    case WM_COMMAND:
        switch (LOWORD(wParam))
        {
        case IDC_HELP_Text:
            if (HIWORD(wParam) == EN_SETFOCUS)
            {
                CHARRANGE cf = { -1,0 };
                SendDlgItemMessage(hWnd,IDC_HELP_Text,EM_EXSETSEL,0,(LPARAM)&cf);
            }
            return true;
        case IDOK:
        case IDCANCEL:
            EndDialog(hWnd,wParam);
            hHelpBox = 0;
            return true;
        }
        break;
    case WM_CLOSE:
        EndDialog(hWnd,IDOK);
        hHelpBox = 0;
        CheckDlgButton(GetParent(hWnd),IDC_CHELP,BST_UNCHECKED);
        return true;
    case WM_DESTROY:
        return false;
    }
    return false;
}

bool ToggleHelpBox(HWND hWnd,int HELPID)
{
    if (hHelpBox) {
        CheckDlgButton(hWnd,IDC_CHELP,BST_UNCHECKED);
        CloseHelpBox();
        return false;
```

```
    }
    else {
        CheckDlgButton(hWnd,IDC_CHELP,BST_CHECKED);
        hHelpBox = CreateDialogParam(hInstance,MAKEINTRESOURCE(IDD_HELP),hWnd,(DLGPROC)&HelpDialogProc,HELPID);
        SetFocus(hWnd);
        return true;
    }
}

void CloseHelpBox()
{
    if (hHelpBox) {
        EndDialog(hHelpBox,0);
        hHelpBox = 0;
    }
}
```

```
// InfoBox.cpp from D2E

#include "JamellaD2E.h"

HBITMAP hBmpWebLink;

static int              RTFStreamOffset;
static const char*      RTFCharStream;

static DWORD CALLBACK RTFStreamCallback(DWORD dwCookie,LPBYTE pbBuff,LONG cb,LONG FAR *pcb)
{
    if (!RTFCharStream) return -1;
    if (strlen(RTFCharStream) - RTFStreamOffset <= 0) return -1;

    *pcb = min((LONG)strlen(RTFCharStream) - RTFStreamOffset,cb);
    strncpy((char *)pbBuff,RTFCharStream + RTFStreamOffset,*pcb);


    RTFStreamOffset += *pcb;

    return 0;
}

void RTFStreamSend(HWND hWnd,int CtrlID,const char* Stream)
{
    RTFStreamOffset = 0;
    RTFCharStream = Stream;

    EDITSTREAM edstr;
    edstr.dwCookie = 0;
    edstr.dwError = 0;
    edstr.pfnCallback = &RTFStreamCallback;

    SendDlgItemMessage(hWnd,CtrlID,EM_STREAMIN,SF_RTF,(LPARAM) &edstr);
}

LRESULT CALLBACK InfoDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
    switch(uMsg)
    {
    case WM_INITDIALOG:
        {
            sprintf(buffer,"%s\n%s",PROGRAMNAME,VERSION);
            SetDlgItemText(hWnd,IDC_INFO_Program,buffer);

            sprintf(buffer,"Built on %s at %s\nVersion Code: %s",__DATE__,__TIME__,RELEASECODE);
            SetDlgItemText(hWnd,IDC_INFO_Date,buffer);

            SetDlgItemText(hWnd,IDC_INFO_URL,PERMANENTURL);
            SetDlgItemText(hWnd,IDC_INFO_Email,EMAIL);

            SendDlgItemMessage(hWnd,IDC_INFO_LINK,BM_SETIMAGE,IMAGE_BITMAP,(LPARAM)hBmpWebLink);
            PlaySound(MAKEINTRESOURCE(IDR_WAVE_JamellaSound),hInstance,SND_RESOURCE | SND_ASYNC);
        }
        return true;
    case WM_COMMAND:
        switch (LOWORD(wParam))
        {
        case IDC_INFO_LINK:
            ShellExecute(NULL,"open",PERMANENTURL,NULL,NULL,SW_SHOWMAXIMIZED);
            return true;
        case IDOK:
        case IDCANCEL:
            EndDialog(hWnd,wParam);
            return true;
        }
        break;
    case WM_CLOSE:
        EndDialog(hWnd,wParam);
        return true;
    case WM_DESTROY:
        return false;
    }
    return false;
}
```

```
// Inventory Grid Options

#include "JamellaD2E.h"

#if INVGRIDS == 0

// D2 Standard Sizes
struct InvGrids InvGrids =
{
    10,4,       // X/Y Inventory
    6,4,        // X/Y Stash
    3,4,        // X/Y Cube
    4,4         // X/Y Belt
};

#else

// Extended Grids for Fusman
struct InvGrids InvGrids =
{
    10,6,       // X/Y Inventory
    10,7,       // X/Y Stash
    8,9,        // X/Y Cube
    4,4         // X/Y Belt
};

#endif

void SetInvGridPreset(int i)
{
}
```

```
// Item.cpp from D2E
// Contains methods of Item base class

#include "JamellaD2E.h"

class Item *Items = 0;

Item::Item()
{
    ListNext = ListPrev = 0;
    Busy = Decoded = false;
    Info = 0;
    MD = 0;

    Gems = 0;
    GInfo = 0;
}
Item::~Item()
{
    if (ListNext) {
        delete ListNext;
        ListNext = 0;
    }
}

// Chained List Functions
int Item::Count()
{
    if (this == 0) return 0;
    return ListNext->Count() + 1;
}
Item* Item::GetOrdinal(int i)
{
    if (i == 0) return this;
    else if (!ListNext) return 0;
    else return ListNext->GetOrdinal(i-1);
}
void Item::Attach(Item *i)
{
    if (this == 0)
    {
        Items = i;
        i->ListPrev = i->ListNext = 0;
    }
    else if(!ListNext)
    {
        ListNext = i;
        ListNext->ListPrev = this;
    }
    else ListNext->Attach(i);
}
void Item::Delete()
{
    if (this == 0) return;
    if (ListPrev == 0)
    {
        Items = ListNext;
        if (ListNext)
            ListNext->ListPrev = 0;

        ListNext = 0;
        delete this;
    }
    else
    {
        ListPrev->ListNext = ListNext;
        if (ListNext)
            ListNext->ListPrev = ListPrev;

        ListNext = 0;
        delete this;
    }
}
#undef AssertBox
#define AssertBox
// Virtual Functions
bool Item::LoadItemRecord(BYTE *data)
{
    AssertBox("Base class cannot be loaded with an item record!");
```

```
    return false;
}
BYTE* Item::GetItemRecord()
{
    AssertBox("Base class cannot give an item record!");
    return 0;
}
DWORD Item::ItemRecordLength()
{
    return 0;
}
void Item::BlankItem()
{
}
int Item::Container() const
{
//  AssertBox("Base class cannot return a container!");
    return CNT_NONE;
}
int Item::xPos() const
{
//  AssertBox("Base class cannot return a x-coordinate!");
    return 0;
}
int Item::yPos() const
{
//  AssertBox("Base class cannot return a y-coordinate!");
    return 0;
}
int Item::xPixelPos() const
{
    return xPos() * 29 - 1;
}
int Item::yPixelPos() const
{
    return yPos() * 29 - 1;
}
int Item::xSize()
{
//  AssertBox("Base class cannot return a x-size!");
    return 0;
}
int Item::ySize()
{
//  AssertBox("Base class cannot return a y-size!");
    return 0;
}
int Item::xPixelSize()
{
    return xSize() * 29 - 1;
}
int Item::yPixelSize()
{
    return ySize() * 29 - 1;
}
bool Item::isInRegion(int x,int y)
{
    if (x < xPixelPos() || x > xPixelPos() + xPixelSize()) return false;
    if (y < yPixelPos() || y > yPixelPos() + yPixelSize()) return false;
    return true;
}
bool Item::SetCoordinates(int Container,int xPos,int yPos)
{
    AssertBox("Base class cannot be set coordinates!");
    return false;
}
int Item::BodyCode()
{
//  AssertBox("Base class does not have valid BodyCode() method!");
    return 0;
}
int Item::BodyPlace()
{
    if (!this || !Info) return -1;
    return Info->BodyPlace;
}

bool Item::FindInfo()
{
```

```
    AssertBox("Base class cannot find item info!");
    return false;
}
HBITMAP Item::GetBitmap()
{
    return itemunknown.hBmp;
}
// Simple Item Properties
DWORD Item::ItemCode()
{
    AssertBox("Base class does not have valid ItemCode() method!");
    return 0;
}
void Item::SetItemCode(DWORD x)
{
    AssertBox("Base class does not have valid SetItemCode(DWORD) method!");
}
int Item::UniqueCode()
{
    AssertBox("Base class does not have valid UniqueCode() method!");
    return 0;
}
void Item::SetUniqueCode(int x)
{
    AssertBox("Base class does not have valid SetUniqueCode(int) method!");
}

int Item::Quality()
{
    return USUALITEM;
}
void Item::SetQuality(int x)
{
    AssertBox("Base class does not have valid SetQuality(int) method!");
}
int Item::Quantity()
{
    AssertBox("Base class does not have valid Quantity() method!");
    return 0;
}
void Item::SetQuantity(int i)
{
    AssertBox("Base class does not have valid SetQuantity(int) method!");
}

int Item::Durability()
{
    AssertBox("Base class does not have valid Durability() method!");
    return 0;
}
void Item::SetDurability(int x)
{
    AssertBox("Base class does not have valid SetDurability(int) method!");
}
int Item::DurabilityMax()
{
    AssertBox("Base class does not have valid DurabilityMax() method!");
    return 0;
}
void Item::SetDurabilityMax(int x)
{
    AssertBox("Base class does not have valid SetDurabilityMax(int) method!");
}
DWORD Item::DWA()
{
    AssertBox("Base class does not have valid DWA() method!");
    return 0;
}
void Item::SetDWA(DWORD x)
{
    AssertBox("Base class does not have valid SetDWA(DWORD) method!");
}
DWORD Item::DWB()
{
    AssertBox("Base class does not have valid DWB() method!");
    return 0;
}
void Item::SetDWB(DWORD x)
{
```

```
    AssertBox("Base class does not have valid SetDWB(DWORD) method!");
}
DWORD Item::DWARandomOffset(int x)
{
    RAND r = { DWA(), 666 };

    for(int z=0;z<x;z++)
        Random(&r);

    return r.Seed;
}
DWORD Item::DWBRandomOffset(int x)
{
    RAND r = { DWB(), 666 };

    for(int z=0;z<x;z++)
        Random(&r);

    return r.Seed;
}

int Item::MagicLevel()
{
    AssertBox("Base class does not have valid MagicLevel() method!");
    return 0;
}
void Item::SetMagicLevel(int x)
{
    AssertBox("Base class does not have valid SetMagicLevel(int) method!");
}

int Item::GemNum()
{
    AssertBox("Base class does not have valid GemNum() method!");
    return 0;
}
void Item::SetGemNum(int i)
{
    AssertBox("Base class does not have valid SetGemNum(int) method!");
}

bool Item::Socketed()
{
    AssertBox("Base class does not have valid Socketed() method!");
    return 0;
}
void Item::SetSocketed(bool i)
{
    AssertBox("Base class does not have valid SetSocketed(bool) method!");
}
bool Item::Socketable()
{
    if (!Info) FindInfo();
    if (Info) return (Info->Sockets > 0);
    else return 0;
}
bool Item::Identified()
{
    AssertBox("Base class does not have valid Identified() method!");
    return 0;
}
void Item::SetIdentified(bool i)
{
    AssertBox("Base class does not have valid SetIdentified(DWORD) method!");
}
bool Item::Starter()
{
    AssertBox("Base class does not have valid Starter() method!");
    return 0;
}
void Item::SetStarter(bool i)
{
    AssertBox("Base class does not have valid SetStarter(DWORD) method!");
}

// Ear-specific Properties
int Item::OpponentClass()
{
    AssertBox("Base class does not have valid OpponentClass() method!");
```

```
    return 0;
}
void Item::SetOpponentClass(int x)
{
    AssertBox("Base class does not have valid SetStarter(int) method!");
}
const char* Item::OpponentClassString()
{
    AssertBox("Base class does not have valid OpponentClassString() method!");
    return 0;
}
int Item::OpponentLevel()
{
    AssertBox("Base class does not have valid OpponentLevel() method!");
    return 0;
}
void Item::SetOpponentLevel(int x)
{
    AssertBox("Base class does not have valid SetOpponentLevel(int) method!");
}
const char* Item::OpponentName()
{
    AssertBox("Base class does not have valid OpponentName() method!");
    return 0;
}
void Item::SetOpponentName(const char *s)
{
    AssertBox("Base class does not have valid SetOpponentName(const char *) method!");
}

int Item::BaseDefense()
{
    AssertBox("Base class does not have valid BaseDefense() method!");
    return 0;
}
ArmorClass Item::Defense()
{
    AssertBox("Base class does not have valid BaseDefense() method!");
    ArmorClass AC = { 0, 0, false };
    return AC;
}
Damage Item::WeaponDamage()
{
    AssertBox("Base class does not have valid WeaponDamage() method!");
    Damage WD = { 0, 0,0, 0,0, false };
    return WD;
}
unsigned int Item::RequiredStrength()
{
    if (!Info)
        if (!FindInfo()) return 0;
    return Info->MinStr;
}
unsigned int Item::RequiredDexterity()
{
    if (!Info)
        if (!FindInfo()) return 0;
    return Info->MinDex;
}
unsigned int Item::RequiredELevel()
{
    AssertBox("Base class does not have valid RequiredELevel() method");
    return 0;
}
bool Item::isWearable()
{
#if defined(JAMELLAEDITOR)
    bool is = true;
    if (Info) {
        if (RequiredStrength() > fc.gf.strength) is = false;
        if (RequiredDexterity() > fc.gf.dexterity) is = false;
    }
    if (RequiredELevel() > fc.gf.level) is = false;

    return is;
#else
    return true;
#endif
}
```

```
bool Item::Decode()
{
//  AssertBox("Base class does not have valid Decode() method!");
    return false;
}
const char* Item::Name()
{
    if (!Info)
        if (!FindInfo())
            return 0;

    return Info->ItemName;
}
const char* Item::RichText()
{
//  AssertBox("Base class does not have valid RichText() method!");
    return 0;
}
bool Item::FindGemInfo()
{
    GInfo = 0;
    return false;
}

char Item::NameTmp[256];
char Item::RichTextTmp[2048];

void Item::StartRTF(char *s)
{
    *s = 0;

    // RTF Header
    strcat(s,"{\\rtf1\\ansi\\ansicpg1252\\deff0\\deflang1031");

    // RTF Font Table (just normal font)
    strcat(s,"{\\fonttbl{\\f0\\fswiss\\fcharset0 MS Sans Serif;}}");

    // Color Table:
    // 0 = Regular Items
    strmcat(s,"{\\colortbl ;",
        "\\red72\\green80\\blue184;",   // 1 = Magic Items
        "\\red255\\green255\\blue0;",   // 2 = Rare Items
        "\\red0\\green196\\blue0;",     // 3 = Set Items
        "\\red144\\green136\\blue88;",  // 4 = Unique Items
        "\\red124\\green124\\blue124;", // 5 = Socketed Items
        "\\red255\\green0\\blue0;}",    // 6 = Red (Error or Unidentified)
        0);

    // Start Paragraph
    strcat(s,"\\uc1\\pard\\qc\\b\\f0\\fs16");
}
```

```
// Item103.cpp from D2E
// Contains methods for the Class Item103 representing old item data structures from patch version 1.03

#include "JamellaD2E.h"

Item103::Item103()
{
    ZeroMemory(&Data,sizeof Data);
    MD = new class MagicDecoder(this);
}
Item103::~Item103()
{
    delete MD;
    if (ListNext) {
        delete ListNext;
        ListNext = 0;
    }
}
bool Item103::LoadItemRecord(BYTE *record)
{
    if (*(record+0) != 'J' && *(record+1) != 'M')
        return false;

    memcpy(&Data,record,sizeof Data);
    Info = 0;
    return true;
}
BYTE* Item103::GetItemRecord()
{
    return (BYTE*) &Data;
}
DWORD   Item103::ItemRecordLength()
{
    return sizeof Data;
}
void Item103::BlankItem()
{
    ZeroMemory(&Data,sizeof Data);

    Data.JM[0] = 0x4A;
    Data.JM[1] = 0x4D;
    Data.bitfield1 = 0x0010;
    Data.bitfield2 = 0x0000;
}
int Item103::Container() const
{
    if ((Data.container & 0x07F8) == 0x0000) return CNT_INVENTORY;
    if ((Data.container & 0x07F8) == 0x0020) return CNT_STASH;
    if ((Data.container & 0x07F8) == 0x0018) return CNT_CUBE;
    if ((Data.container & 0x07F8) == 0x07F8)
    { // On Body or On Belt
        if (Data.itemcode & 0x8000)
            return CNT_BELT;
        if (Data.itemcode & 0x4000)
            return CNT_BODY;
    }
    return CNT_NONE;
}
int Item103::xPos() const
{
    switch(Container())
    {
    default:
        return 0;
    case CNT_INVENTORY:
    case CNT_STASH:
    case CNT_CUBE:
        return (Data.coordinates & 0x3E) >> 1;
    case CNT_BELT:
        return (Data.coordinates & 0x06) >> 1;
    }
}
int Item103::yPos() const
{
    switch(Container())
    {
    default:
        return 0;
    case CNT_INVENTORY:
```

```
    case CNT_STASH:
    case CNT_CUBE:
        return (Data.coordinates & 0xC0) >> 6;
    case CNT_BELT:
        return 3 - ((Data.coordinates & 0x18) >> 3);
    }
}
int Item103::xSize()
{
    if (!this && SelInfo) return SelInfo->SizeX;
    if (!this) return 1;

    if (!Info) FindInfo();
    return Info->SizeX;
}
int Item103::ySize()
{
    if (!this && SelInfo) return SelInfo->SizeY;
    if (!this) return 1;

    if (!Info) FindInfo();
    return Info->SizeY;
}
bool Item103::SetCoordinates(int Container,int xPos,int yPos)
{
    switch(Container)
    {
    default:
        return false;
    case CNT_INVENTORY:
        {
            if (xPos < 0 || xPos >= InvGrids.xInventory) return false;
            if (yPos < 0 || yPos >= InvGrids.yInventory) return false;

            Data.container = (Data.container & 0xF807) | 0x0000;
            Data.coordinates = (Data.coordinates & 0xC1) | (xPos << 1);
            Data.coordinates = (Data.coordinates & 0x3F) | (yPos << 6);

            Data.bitfield3 &= 0xF0;
            Data.itemcode &= 0x3FFF;
            return true;
        }
    case CNT_STASH:
        {
            if (xPos < 0 || xPos >= InvGrids.xStash) return false;
            if (yPos < 0 || yPos >= InvGrids.yStash) return false;

            Data.container = (Data.container & 0xF807) | 0x0020;
            Data.coordinates = (Data.coordinates & 0xC1) | (xPos << 1);
            Data.coordinates = (Data.coordinates & 0x3F) | (yPos << 6);

            Data.bitfield3 &= 0xF0;
            Data.itemcode &= 0x3FFF;
            return true;
        }
    case CNT_CUBE:
        {
            if (xPos < 0 || xPos >= InvGrids.xCube) return false;
            if (yPos < 0 || yPos >= InvGrids.yCube) return false;

            Data.container = (Data.container & 0xF807) | 0x0018;
            Data.coordinates = (Data.coordinates & 0xC1) | (xPos << 1);
            Data.coordinates = (Data.coordinates & 0x3F) | (yPos << 6);

            Data.bitfield3 &= 0xF0;
            Data.itemcode &= 0x3FFF;
            return true;
        }
    case CNT_BELT:
        {
            if (xPos < 0 || xPos >= InvGrids.xBelt) return false;
            if (yPos < 0 || yPos >= InvGrids.yBelt) return false;

            Data.container = (Data.container & 0xF807) | 0x07F8;
            Data.coordinates = (Data.coordinates & 0xF9) | (xPos << 1);
            Data.coordinates = (Data.coordinates & 0xE7) | ((3-yPos) << 3);

            Data.bitfield3 = (Data.bitfield3 & 0xF0) | 0x00;
            Data.itemcode &= 0x3FFF;
```

```
            Data.itemcode |= 0x8000;
            return true;
        }
    case CNT_SOCKET:
        {
            Data.container = (Data.container & 0xF807) | 0x07F8;
            Data.coordinates = Data.coordinates & 0xF9;
            Data.coordinates = Data.coordinates & 0xE7;

            Data.bitfield3 = (Data.bitfield3 & 0xF0) | 0x00;
            Data.itemcode &= 0x3FFF;
            Data.itemcode |= 0x8000;
            return true;
        }
    case CNT_BODY:
        {
            if (xPos < 1 || xPos > 10) return false;

            Data.container = (Data.container & 0xF807) | 0x07F8;
            Data.coordinates &= 0x01;

            Data.bitfield3 = (Data.bitfield3 & 0xF0) | (xPos & 0x0F);
            Data.itemcode &= 0x3FFF;
            Data.itemcode |= 0x4000;
            return true;
        }
    }
}
bool Item103::FindInfo()
{
    Info = &itemunknown;

    RAND RDWB = { DWB(), 666 };
    StartRandoms(this,&RDWB);
    int DWB0 = Random(&RDWB);

    for(int z=0;z<nItemInfos;z++)
    {
        // Check Item Code
        if (ItemCode() != ItemInfos[z].ItemCode) continue;

        // Check if Set or Unique Item
        switch(ItemInfos[z].ItemSearch)
        {
        case 'U':
            if (Quality() != UNIQUEITEM) continue;

            // Check Set/Unique Item Class Code
            if (UniqueCode() != ItemInfos[z].UniqueCode) continue;

            break;
        case 'S':
            if (Quality() != SETITEM) continue;

            // Check Set/Unique Item Class Code

            if (UniqueCode() != ItemInfos[z].UniqueCode) continue;

            // Get correct set item when there are multiple possiblities
            if (ItemInfos[z].DWBCode != 0)
            {
                int offset = DWB0 % 0x10;
                if (( ItemInfos[z].DWBCode & (1 << offset) ) == 0) continue;
            }

            break;
        }

        Info = &ItemInfos[z];
    }

    if (!Info || (!Info->BitmapID && Info->ItemCode != 0x1170 && Info->ItemCode != 0x1190))
    {
        Info = &itemunknown;
        return false;
    }
    return true;
}
HBITMAP Item103::GetBitmap()
```

```
{
    if (!Info) FindInfo();

     // Amulets
    if (ItemCode() == 0x1170)
    {
        int x = DWARandomOffset(1) % 3;
        return AmuletImages[x].hBmp;
    }
    // Rings
    if (ItemCode() == 0x1190)
    {
        return RingImages[DWARandomOffset(1) % 5].hBmp;
    }

    return ItemInfoGetBitmap(Info);
}
DWORD Item103::ItemCode()
{
    return Data.itemcode & 0x3FF0;
}
void Item103::SetItemCode(DWORD x)
{
    Data.itemcode = (Data.itemcode & 0xC00F) | (WORD(x) & 0x3FF0);
    FindInfo();
}
int Item103::UniqueCode()
{
    return Data.specialitemcode;
}
void Item103::SetUniqueCode(int x)
{
    Data.specialitemcode = x;
    FindInfo();
}
int Item103::Quality()
{
    return Data.quantity & 0x000F;
}
void Item103::SetQuality(int x)
{
    Data.quantity = (Data.quantity & 0xFFF0) | (x & 0x000F);
    FindInfo();
}
bool Item103::Socketed()
{
    return (Data.bitfield1 & 0x0800) != 0;
}
void Item103::SetSocketed(bool i)
{
    Data.bitfield1 = (Data.bitfield1 & 0xF7FF) | (i << 11);
}
bool Item103::Socketable()
{
    if (!Info) FindInfo();
    return (Info->Sockets != 0);
}
bool Item103::Identified()
{
    return (Data.bitfield1 & 0x0010) != 0;
}
void Item103::SetIdentified(bool i)
{
    Data.bitfield1 = Data.bitfield1 & 0xFFEF | (i << 4);
}
bool Item103::Starter()
{
    return (Data.bitfield2 & 0x0002) != 0;
}
void Item103::SetStarter(bool i)
{
    Data.bitfield2 = Data.bitfield2 & 0xFFFD | (i << 1);
}
int Item103::Quantity()
{
    return ((Data.quantity & 0xFFE0) >> 5) +
           ((Data.quantityex & 0xFF) << 11);
//  return (Data.quantity & 0x1FE0) >> 5;
}
```

```
void Item103::SetQuantity(int i)
{
#if defined(JAMELLAEDITOR)

    if (!RegOptions.ExceedQuantity) {
        Data.quantity = (Data.quantity & 0xE01F) | ((i & 0x00FF) << 5);
    }
    else {
        Data.quantity = (Data.quantity & 0x001F) | ((i & 0x1FFF) << 5);
        Data.quantityex = (i >> 11);
    }

#else
    Data.quantity = (Data.quantity & 0xE01F) | ((i & 0x00FF) << 5);
#endif
}
int Item103::Durability()
{
    return (Data.durability & 0x01FE) >> 1;
}
void Item103::SetDurability(int i)
{
    if (i > DurabilityMax()) i = DurabilityMax();
    if (i > 255) i = 255;

    Data.durability = (Data.durability & 0xFE01) | ((i & 0xFF) << 1);
}
int Item103::DurabilityMax()
{
    return ((Data.durability & 0xFE00) >> 9) + ((Data.coordinates & 0x01) * 128);
}
void Item103::SetDurabilityMax(int i)
{
    if (i > 255) i = 255;

    Data.durability = (Data.durability & 0x01FF) | ((i & 0x7F) << 9);
    Data.coordinates = (Data.coordinates & 0xFE) | ( (i & 0x80) ? 0x01 : 0x00 );
}
int Item103::GemNum()
{
    return (Data.bitfield3 & 0x60) >> 5;
}
void Item103::SetGemNum(int i)
{
    Data.bitfield3 = (Data.bitfield3 & 0x1F) | ((i%8)  << 5);
}
int Item103::BodyCode()
{
    return Data.bitfield3 & 0x0F;
}
DWORD Item103::DWA()
{
    return ( (Data.properties2 & 0x07) << 29 ) |
           ( (Data.properties1 & 0xFFFFFFF8) >> 3 );
}
void Item103::SetDWA(DWORD x)
{
    Data.properties1 = (Data.properties1 & 0x00000007) | ( (x & 0x1FFFFFFF) << 3 );
    Data.properties2 = (Data.properties2 & 0xFFFFFFF8) | BYTE( (x & 0xE0000000) >> 29 );
}
DWORD Item103::DWB()
{
    return ( (Data.container & 0x07) << 29 ) |
           ( (Data.properties2 & 0xFFFFFFF8) >> 3 );
}
void Item103::SetDWB(DWORD x)
{
    Data.container = (Data.container & 0xFFF8) | BYTE( (x & 0xE0000000) >> 29 );
    Data.properties2 = (Data.properties2 & 0x00000007) | ( (x & 0x1FFFFFFF) << 3 );
}
int Item103::MagicLevel()
{
    return Data.modlevel;
}
void Item103::SetMagicLevel(int x)
{
    Data.modlevel = x;
}
bool Item103::FindGemInfo()
```

```
{
    GInfo = 0;
    for(int z=0;z<nGemInfos;z++)
    {
        // Check Item Code
        if (ItemCode() != GemInfos[z].ItemCode) continue;
        GInfo = &GemInfos[z];
        return true;
    }
    return false;
}
```

```
// Item103EarEar.cpp from D2E
// Contains methods for the Class Item103Ear representing old item data structures from patch version 1.03 for ears

#include "JamellaD2E.h"

Item103Ear::Item103Ear()
{
    ZeroMemory(&Data,sizeof Data);
}
Item103Ear::~Item103Ear()
{
    if (ListNext) {
        delete ListNext;
        ListNext = 0;
    }
}
bool Item103Ear::LoadItemRecord(BYTE *record)
{
    if (*(record+0) != 'J' && *(record+1) != 'M')
        return false;

    memcpy(&Data,record,sizeof Data);
    Info = 0;
    return true;
}
BYTE* Item103Ear::GetItemRecord()
{
    return (BYTE*) &Data;
}
DWORD   Item103Ear::ItemRecordLength()
{
    return sizeof Data;
}
void Item103Ear::BlankItem()
{
    ZeroMemory(&Data,sizeof Data);

    Data.JM[0] = 0x4A;
    Data.JM[1] = 0x4D;
    Data.bitfield1 = 0x0010;
    Data.bitfield2 = 0x0001;

    SetOpponentName("SetNewName");
}

int Item103Ear::Container() const
{
    if ((Data.container & 0x60) == 0x00) return CNT_INVENTORY;
    if ((Data.container & 0x60) == 0x40) return CNT_STASH;
    if ((Data.container & 0x60) == 0x60) return CNT_CUBE;
    return CNT_NONE;
}
int Item103Ear::xPos() const
{
    return (Data.coordinates & 0x007C) >> 2;
}
int Item103Ear::yPos() const
{
    return (Data.coordinates & 0x0180) >> 7;
}
int Item103Ear::xSize()
{
    if (!this && SelInfo) return SelInfo->SizeX;
    if (!this) return 1;

    if (!Info) FindInfo();
    return Info->SizeX;
}
int Item103Ear::ySize()
{
    if (!this && SelInfo) return SelInfo->SizeY;
    if (!this) return 1;

    if (!Info) FindInfo();
    return Info->SizeY;
}
bool Item103Ear::SetCoordinates(int Container,int xPos,int yPos)
{
    switch(Container)
```

```
    {
    default:
        return false;
    case CNT_INVENTORY:
        {
            if (xPos < 0 || xPos >= InvGrids.xInventory) return false;
            if (yPos < 0 || yPos >= InvGrids.yInventory) return false;

            Data.container = (Data.container & 0x9F) | 0x00;
            Data.coordinates = (Data.coordinates & 0xFF83) | (xPos << 2);
            Data.coordinates = (Data.coordinates & 0xFE7F) | (yPos << 7);
            return true;
        }
    case CNT_STASH:
        {
            if (xPos < 0 || xPos >= InvGrids.xStash) return false;
            if (yPos < 0 || yPos >= InvGrids.yStash) return false;

            Data.container = (Data.container & 0x9F) | 0x40;
            Data.coordinates = (Data.coordinates & 0xFF83) | (xPos << 2);
            Data.coordinates = (Data.coordinates & 0xFE7F) | (yPos << 7);
            return true;
        }
    case CNT_CUBE:
        {
            if (xPos < 0 || xPos >= InvGrids.xCube) return false;
            if (yPos < 0 || yPos >= InvGrids.yCube) return false;

            Data.container = (Data.container & 0x9F) | 0x60;
            Data.coordinates = (Data.coordinates & 0xFF83) | (xPos << 2);
            Data.coordinates = (Data.coordinates & 0xFE7F) | (yPos << 7);
            return true;
        }
    }
}
bool Item103Ear::FindInfo()
{
    Info = &itemunknown;

    for(int z=0;z<nItemInfos;z++)
    {
        // Check Item Code
        if (ItemCode() != ItemInfos[z].ItemCode) continue;

        // Check if Set or Unique Item
        if (ItemInfos[z].ItemSearch != 'N') continue;

        Info = &ItemInfos[z];
    }

    if (!Info || !Info->BitmapID)
    {
        Info = &itemunknown;
        return false;
    }
    return true;
}
HBITMAP Item103Ear::GetBitmap()
{
    if (!Info) FindInfo();

    return ItemInfoGetBitmap(Info);
}
DWORD Item103Ear::ItemCode()
{
    return (Data.itemcode & 0x7FE0) >> 1;
}
void Item103Ear::SetItemCode(DWORD x)
{
    Data.itemcode = (Data.itemcode & 0x801F) | ((WORD(x) << 1) & 0x7FE0);
    FindInfo();
}
int Item103Ear::OpponentClass()
{
    return (Data.coordinates & 0x1C00) >> 10;
}
const char *Item103Ear::OpponentClassString()
{
    int x = OpponentClass();
```

```
    if (x >= 0 && x < 5)
        return CharClasses[x];
    else
        return "Error!!!";
}
void Item103Ear::SetOpponentClass(int x)
{
    Data.coordinates = (Data.coordinates & 0xE3FF)  | ((x % 5) << 10);
}
int Item103Ear::OpponentLevel()
{
    return ((Data.string[0] & 0xF0) >> 4) +
           ((Data.string[1] & 0x0F) << 4);
}
void Item103Ear::SetOpponentLevel(int x)
{
    x %= 100;

    Data.string[0] = (Data.string[0] & 0x0F) | ((x & 0x0F) << 4);
    Data.string[1] = (Data.string[1] & 0xF0) | ((x & 0xF0) >> 4);
}
char Item103Ear::OpponentNameChar(int n)
{
    int bitoffset = 12 + n * 7;

    // Get First Part
    int B = bitoffset / 8;
    int V = bitoffset % 8;
    char c = (Data.string[B] >> V) & 0x7F;

    // Get Secound Part
    int rest = 8 - V;
    int mask = (0xFF << (V-1)) ^ 0xFF;
    c |= (Data.string[B+1] & mask) << rest;

    return c;
}
void Item103Ear::SetOpponentNameChar(int n,char c)
{
    int bitoffset = 12 + n * 7;

    // Get First Part
    int B = bitoffset / 8;
    int V = bitoffset % 8;
    int mask1 = (0xFF << V) ^ 0xFF;
    Data.string[B] = (Data.string[B] & mask1) | ( (c << V) & (mask1 ^ 0xFF) );

    int rest = 8 - V;
    int mask2 = 0xFF << (V-1);
    Data.string[B+1] = (Data.string[B+1] & mask2) | ( (c >> rest) & (mask2 ^ 0xFF) );
}
static char ItemOpponentNameTemp[20];
const char* Item103Ear::OpponentName()
{
    memset(ItemOpponentNameTemp,0,sizeof ItemOpponentNameTemp);

    for(int z=0;z<15;z++)
        ItemOpponentNameTemp[z] = OpponentNameChar(z);

    return ItemOpponentNameTemp;
}
void Item103Ear::SetOpponentName(const char *s)
{
    for(int z=0;z<15;z++)
        SetOpponentNameChar(z,*s++);

    SetOpponentNameChar(15,0);
}
const char* Item103Ear::Name()
{
    sprintf(NameTmp,"%s's Ear",OpponentName());
    return NameTmp;
}
const char* Item103Ear::RichText()
{
    char *s = RichTextTmp;
    StartRTF(s);

    strmcat(s,"\\cf0 ",Name(),0);
```

```
    char buffers[2][64];

    // Ear Descriptions
    strmcat(s,"\\par ",OpponentClassString(),0);
    sprintf(buffers[0],"\\par Level %i",OpponentLevel());
    strcat(s,buffers[0]);

    strcat(s,"\\par }");

    return s;
}
```

```
// Item103EarEar.cpp from D2E
// Contains methods for the Class Item104Ear representing old item data structures from patch version 1.03 for ears

#include "JamellaD2E.h"

Item104Ear::Item104Ear()
{
    ZeroMemory(&Data,sizeof Data);
}
Item104Ear::~Item104Ear()
{
    if (ListNext) {
        delete ListNext;
        ListNext = 0;
    }
}
bool Item104Ear::LoadItemRecord(BYTE *record)
{
    if (*(record+0) != 'J' && *(record+1) != 'M')
        return false;

    memcpy(&Data,record,sizeof Data);
    Info = 0;
    return true;
}
BYTE* Item104Ear::GetItemRecord()
{
    return (BYTE*) &Data;
}
DWORD   Item104Ear::ItemRecordLength()
{
    return sizeof Data;
}
void Item104Ear::BlankItem()
{
    ZeroMemory(&Data,sizeof Data);

    Data.JM[0] = 0x4A;
    Data.JM[1] = 0x4D;
    Data.bitfield1 = 0x0010;
    Data.bitfield2 = 0x0039;

    SetOpponentName("SetNewName");
}

int Item104Ear::Container() const
{
    if ((Data.coordinates & 0x1C00) == 0x0000) return CNT_INVENTORY;
    if ((Data.coordinates & 0x1C00) == 0x1000) return CNT_STASH;
    if ((Data.coordinates & 0x1C00) == 0x0C00) return CNT_CUBE;
    return CNT_NONE;
}
int Item104Ear::xPos() const
{
    return (Data.coordinates & 0x007C) >> 2;
}
int Item104Ear::yPos() const
{
    return (Data.coordinates & 0x0180) >> 7;
}
int Item104Ear::xSize()
{
    if (!this && SelInfo) return SelInfo->SizeX;
    if (!this) return 1;

    if (!Info) FindInfo();
    return Info->SizeX;
}
int Item104Ear::ySize()
{
    if (!this && SelInfo) return SelInfo->SizeY;
    if (!this) return 1;

    if (!Info) FindInfo();
    return Info->SizeY;
}
bool Item104Ear::SetCoordinates(int Container,int xPos,int yPos)
{
    switch(Container)
```

```
    {
    default:
        return false;
    case CNT_INVENTORY:
        {
            if (xPos < 0 || xPos >= InvGrids.xInventory) return false;
            if (yPos < 0 || yPos >= InvGrids.yInventory) return false;

            Data.coordinates = (Data.coordinates & 0xE3FF) | 0x0000;
            Data.coordinates = (Data.coordinates & 0xFF83) | (xPos << 2);
            Data.coordinates = (Data.coordinates & 0xFE7F) | (yPos << 7);
            return true;
        }
    case CNT_STASH:
        {
            if (xPos < 0 || xPos >= InvGrids.xStash) return false;
            if (yPos < 0 || yPos >= InvGrids.yStash) return false;

            Data.coordinates = (Data.coordinates & 0xE3FF) | 0x1000;
            Data.coordinates = (Data.coordinates & 0xFF83) | (xPos << 2);
            Data.coordinates = (Data.coordinates & 0xFE7F) | (yPos << 7);
            return true;
        }
    case CNT_CUBE:
        {
            if (xPos < 0 || xPos >= InvGrids.xCube) return false;
            if (yPos < 0 || yPos >= InvGrids.yCube) return false;

            Data.coordinates = (Data.coordinates & 0xE3FF) | 0x0C00;
            Data.coordinates = (Data.coordinates & 0xFF83) | (xPos << 2);
            Data.coordinates = (Data.coordinates & 0xFE7F) | (yPos << 7);
            return true;
        }
    }
    return true;
}
bool Item104Ear::FindInfo()
{
    Info = &itemunknown;

    for(int z=0;z<nItemInfos;z++)
    {
        // Check Item Code
        if (ItemCode() != ItemInfos[z].IC) continue;

        // Check if Set or Unique Item
        if (ItemInfos[z].ItemSearch != 'N') continue;

        Info = &ItemInfos[z];
    }

    if (!Info || !Info->BitmapID)
    {
        Info = &itemunknown;
        return false;
    }
    return true;
}
HBITMAP Item104Ear::GetBitmap()
{
    if (!Info) FindInfo();

    return ItemInfoGetBitmap(Info);
}
DWORD Item104Ear::ItemCode()
{
    return ' rae';
}
void Item104Ear::SetItemCode(DWORD x)
{
    // No Item Code
}
int Item104Ear::OpponentClass()
{
    return (Data.string[0] & 0x1C) >> 2;
}
void Item104Ear::SetOpponentClass(int x)
{
    Data.string[0] = (Data.string[0] & 0xE3)  | ((x % 5) << 2);
```

```
}
const char *Item104Ear::OpponentClassString()
{
    int x = OpponentClass();
    if (x >= 0 && x < 5)
        return CharClasses[x];
    else
        return "Error!!!";
}
int Item104Ear::OpponentLevel()
{
    return ((Data.string[0] & 0xE0) >> 5) +
           ((Data.string[1] & 0x1F) << 3);
}
void Item104Ear::SetOpponentLevel(int x)
{
    x %= 100;

    Data.string[0] = (Data.string[0] & 0x1F) | ((x & 0x07) << 5);
    Data.string[1] = (Data.string[1] & 0xE0) | ((x & 0xF8) >> 3);
}
char Item104Ear::OpponentNameChar(int n)
{
    int bitoffset = 13 + n * 7;

    // Get First Part
    int B = bitoffset / 8;
    int V = bitoffset % 8;
    char c = (Data.string[B] >> V) & 0x7F;

    // Get Secound Part
    int rest = 8 - V;
    int mask = (0xFF << (V-1)) ^ 0xFF;
    c |= (Data.string[B+1] & mask) << rest;

    return c;
}
void Item104Ear::SetOpponentNameChar(int n,char c)
{
    int bitoffset = 13 + n * 7;

    // Get First Part
    int B = bitoffset / 8;
    int V = bitoffset % 8;
    int mask1 = (0xFF << V) ^ 0xFF;
    Data.string[B] = (Data.string[B] & mask1) | ( (c << V) & (mask1 ^ 0xFF) );

    int rest = 8 - V;
    int mask2 = 0xFF << (V-1);
    Data.string[B+1] = (Data.string[B+1] & mask2) | ( (c >> rest) & (mask2 ^ 0xFF) );
}
static char ItemOpponentNameTemp[20];
const char* Item104Ear::OpponentName()
{
    memset(ItemOpponentNameTemp,0,sizeof ItemOpponentNameTemp);

    for(int z=0;z<15;z++)
        ItemOpponentNameTemp[z] = OpponentNameChar(z);

    return ItemOpponentNameTemp;
}
void Item104Ear::SetOpponentName(const char *s)
{
    for(int z=0;z<15;z++)
        SetOpponentNameChar(z,*s++);

    SetOpponentNameChar(15,0);
}
const char* Item104Ear::Name()
{
    sprintf(NameTmp,"%s's Ear",OpponentName());
    return NameTmp;
}
const char* Item104Ear::RichText()
{
    char *s = RichTextTmp;
    StartRTF(s);

    strmcat(s,"\\cf0 ",Name(),0);
```

```
    char buffers[2][64];

    // Ear Descriptions
    strmcat(s,"\\par ",OpponentClassString(),0);
    sprintf(buffers[0],"\\par Level %i",OpponentLevel());
    strcat(s,buffers[0]);

    strcat(s,"\\par }");

    return s;
}
```

```
// Item104Ex.cpp from D2E
// Contains methods for the new extended item structure past 1.03

#include "JamellaD2E.h"

Item104Ex::Item104Ex()
{
    ZeroMemory(&Data,sizeof Data);
    MD = new class MagicDecoder(this);
}
Item104Ex::~Item104Ex()
{
    delete MD;
    if (ListNext) {
        delete ListNext;
        ListNext = 0;
    }
}
bool Item104Ex::LoadItemRecord(BYTE *record)
{
    if (*(record+0) != 'J' && *(record+1) != 'M')
        return false;

    memcpy(&Data,record,sizeof Data);
    Info = 0;
    return true;
}
BYTE* Item104Ex::GetItemRecord()
{
    return (BYTE*) &Data;
}
DWORD   Item104Ex::ItemRecordLength()
{
    return sizeof Data;
}
void Item104Ex::BlankItem()
{
    ZeroMemory(&Data,sizeof Data);

    Data.JM[0] = 0x4A;
    Data.JM[1] = 0x4D;
    Data.bitfield1 = 0x0010;
    Data.bitfield2 = 0x0018;
}
int Item104Ex::Container() const
{
    if ((Data.container & 0x07F8) == 0x0000) return CNT_INVENTORY;
    if ((Data.container & 0x07F8) == 0x0020) return CNT_STASH;
    if ((Data.container & 0x07F8) == 0x0018) return CNT_CUBE;
    if ((Data.container & 0x07F8) == 0x07F8) return CNT_BODY;
    return CNT_NONE;
}
int Item104Ex::xPos() const
{
    return (Data.coordinates & 0x3E) >> 1;
}
int Item104Ex::yPos() const
{
    return (Data.coordinates & 0xC0) >> 6;
}
int Item104Ex::xSize()
{
    if (!this && SelInfo) return SelInfo->SizeX;
    if (!this) return 1;

    if (!Info)
        if (!FindInfo())
            return 1;

    return Info->SizeX;
}
int Item104Ex::ySize()
{
    if (!this && SelInfo) return SelInfo->SizeY;
    if (!this) return 1;

    if (!Info)
        if (!FindInfo())
            return 1;
```

```
    return Info->SizeY;
}
bool Item104Ex::SetCoordinates(int Container,int xPos,int yPos)
{
    switch(Container)
    {
    default:
        return false;
    case CNT_INVENTORY:
        {
            if (xPos < 0 || xPos >= InvGrids.xInventory) return false;
            if (yPos < 0 || yPos >= InvGrids.yInventory) return false;

            Data.container = (Data.container & 0xF807) | 0x0000;
            Data.coordinates = (Data.coordinates & 0xC1) | (xPos << 1);
            Data.coordinates = (Data.coordinates & 0x3F) | (yPos << 6);

            Data.itemcode &= 0xFFFFFFFC;
            Data.magicrestr &= 0xBFFF;
            return true;
        }
    case CNT_STASH:
        {
            if (xPos < 0 || xPos >= InvGrids.xStash) return false;
            if (yPos < 0 || yPos >= InvGrids.yStash) return false;

            Data.container = (Data.container & 0xF807) | 0x0020;
            Data.coordinates = (Data.coordinates & 0xC1) | (xPos << 1);
            Data.coordinates = (Data.coordinates & 0x3F) | (yPos << 6);

            Data.itemcode &= 0xFFFFFFFC;
            Data.magicrestr &= 0xBFFF;
            return true;
        }
    case CNT_CUBE:
        {
            if (xPos < 0 || xPos >= InvGrids.xCube) return false;
            if (yPos < 0 || yPos >= InvGrids.yCube) return false;

            Data.container = (Data.container & 0xF807) | 0x0018;
            Data.coordinates = (Data.coordinates & 0xC1) | (xPos << 1);
            Data.coordinates = (Data.coordinates & 0x3F) | (yPos << 6);

            Data.itemcode &= 0xFFFFFFFC;
            Data.magicrestr &= 0xBFFF;
            return true;
        }
    case CNT_BODY:
        {
            if (xPos < 1 || xPos > 10) return false;

            Data.container = (Data.container & 0xF807) | 0x07F8;
            Data.coordinates &= 0x01;

            Data.itemcode &= 0xFFFFFFFC;

            Data.bodycode = (Data.bodycode & 0xC3) | ((xPos & 0x0F) << 2);
            Data.magicrestr |= 0x4000;
            return true;
        }
    }
}

bool Item104Ex::FindInfo()
{
    Info = &itemunknown;

    RAND RDWB = { DWB(), 666 };
    StartRandoms(this,&RDWB);
    DWORD DWB0 = Random(&RDWB);

    for(int z=0;z<nItemInfos;z++)
    {
        // Check Item Code
        if (ItemInfos[z].IC == 0xFFFFFFFF) continue;
        if (ItemCode() != ItemInfos[z].IC) continue;

        // Check if Set or Unique Item
```

```
        switch(ItemInfos[z].ItemSearch)
        {
        case 'U':
            if (Quality() != UNIQUEITEM) continue;

            // Check Set/Unique Item Class Code
            if (UniqueCode() != ItemInfos[z].UniqueCode) continue;

            break;
        case 'S':
            if (Quality() != SETITEM) continue;

            // Check Set/Unique Item Class Code
            if (UniqueCode() != ItemInfos[z].UniqueCode) continue;

            // Get correct set item when there are multiple possiblities
            if (ItemInfos[z].DWBCode != 0)
            {
                int offset = DWB0 % 0x10;
                if (( ItemInfos[z].DWBCode & (1 << offset) ) == 0)
                    continue;
            }
            break;
        }

        Info = &ItemInfos[z];
    }

    if (!Info || (!Info->BitmapID && Info->IC != ' uma' && Info->IC != ' nir'))
    {
        Info = &itemunknown;
        return false;
    }
    return true;
}
HBITMAP Item104Ex::GetBitmap()
{
    if (!Info) FindInfo();

     // Amulets
    if (ItemCode() == ' uma')
    {
        int x = DWARandomOffset(1) % 3;
        return AmuletImages[x].hBmp;
    }
    // Rings
    if (ItemCode() == ' nir')
    {
        return RingImages[DWARandomOffset(1) % 5].hBmp;
    }

    return ItemInfoGetBitmap(Info);
}

DWORD Item104Ex::ItemCode()
{
    return ((Data.itemcode & 0xFFFFFFFC) >> 2) |
           ((Data.bodycode & 0x0003) << 30);
}
void Item104Ex::SetItemCode(DWORD x)
{
    Data.itemcode = (Data.itemcode & 0x00000003) | ((x << 2) & 0xFFFFFFFC);
    Data.bodycode = (Data.bodycode & 0xF7) | (BYTE(x >> 30) & 0x03);
    Info = 0;
}
int Item104Ex::UniqueCode()
{
    return Data.uniquecode;
}
void Item104Ex::SetUniqueCode(int x)
{
    Data.uniquecode = x;
    Info = 0;
}
int Item104Ex::Quality()
{
    return Data.quantity & 0x000F;
}
void Item104Ex::SetQuality(int x)
```

```
{
    Data.quantity = (Data.quantity & 0xFFF0) | (x & 0x000F);
    Info = 0;
}
int Item104Ex::GemNum()
{
    return ((Data.bodycode & 0x80) >> 7) |
           ((Data.magicrestr & 0x03) << 1);
}
void Item104Ex::SetGemNum(int i)
{
    Data.bodycode = (Data.bodycode & 0x7F) | ((i << 7) & 0x80);
    Data.magicrestr = (Data.magicrestr & 0xFFFC) | ((i >> 1) & 0x03);
}
bool Item104Ex::Socketed()
{
    return (Data.bitfield1 & 0x0800) != 0;
}
void Item104Ex::SetSocketed(bool i)
{
    Data.bitfield1 = (Data.bitfield1 & 0xF7FF) | (i << 11);
}
bool Item104Ex::Socketable()
{
    if (!Info) FindInfo();
    return (Info->Sockets != 0);
}
bool Item104Ex::Identified()
{
    return (Data.bitfield1 & 0x0010) != 0;
}
void Item104Ex::SetIdentified(bool i)
{
    Data.bitfield1 = Data.bitfield1 & 0xFFEF | (i << 4);
}
bool Item104Ex::Starter()
{
    return (Data.bitfield2 & 0x0002) != 0;
}
void Item104Ex::SetStarter(bool i)
{
    Data.bitfield2 = Data.bitfield2 & 0xFFFD | (i << 1);
}
int Item104Ex::Quantity()
{
    return ((Data.quantity & 0xFFE0) >> 5) +
           ((Data.quantityex & 0xFF) << 11);
//  return (Data.quantity & 0x1FE0) >> 5;
}
void Item104Ex::SetQuantity(int i)
{
#if defined(JAMELLAEDITOR)
    if (!RegOptions.ExceedQuantity) {
        Data.quantity = (Data.quantity & 0xE01F) | ((i & 0x00FF) << 5);
    }
    else {
        Data.quantity = (Data.quantity & 0x001F) | ((i & 0x1FFF) << 5);
        Data.quantityex = (i >> 11);
    }
#else
    Data.quantity = (Data.quantity & 0xE01F) | ((i & 0x00FF) << 5);
#endif
}
int Item104Ex::Durability()
{
    return (Data.durability & 0x01FE) >> 1;
}
void Item104Ex::SetDurability(int i)
{
    if (i > DurabilityMax()) i = DurabilityMax();
    if (i > 255) i = 255;

    Data.durability = (Data.durability & 0xFE01) | ((i & 0xFF) << 1);
}
int Item104Ex::DurabilityMax()
{
    return ((Data.durability & 0xFE00) >> 9) + ((Data.coordinates & 0x01) * 128);
}
void Item104Ex::SetDurabilityMax(int i)
```

```
{
    if (i > 255) i = 255;

    Data.durability = (Data.durability & 0x01FF) | ((i & 0x7F) << 9);
    Data.coordinates = (Data.coordinates & 0xFE) | ( (i & 0x80) ? 0x01 : 0x00 );
}
int Item104Ex::BodyCode()
{
    return (Data.bodycode >> 2) & 0x0F;
}
DWORD Item104Ex::DWA()
{
    return ( (Data.DWB & 0x07) << 29 ) |
           ( (Data.DWA & 0xFFFFFFF8) >> 3 );
}
void Item104Ex::SetDWA(DWORD x)
{
    Data.DWA = (Data.DWA & 0x00000007) | ( (x & 0x1FFFFFFF) << 3 );
    Data.DWB = (Data.DWB & 0xFFFFFFF8) | ( (x & 0xE0000000) >> 29 );
}
DWORD Item104Ex::DWB()
{
    return ( (Data.container & 0x07) << 29 ) |
           ( (Data.DWB & 0xFFFFFFF8) >> 3 );
}
void Item104Ex::SetDWB(DWORD x)
{
    Data.container = (Data.container & 0xFFF8) | BYTE( (x & 0xE0000000) >> 29 );
    Data.DWB = (Data.DWB & 0x00000007) | ( (x & 0x1FFFFFFF) << 3 );
}
int Item104Ex::MagicLevel()
{
    return (Data.magicrestr & 0x03FC) >> 2;
}
void Item104Ex::SetMagicLevel(int x)
{
    Data.magicrestr = (Data.magicrestr & 0xFC03) | ((x & 0xFF) << 2);
}
```

```
// Item104Sm.cpp from D2E
// Contains methods for the new short item structure past 1.03

#include "JamellaD2E.h"

Item104Sm::Item104Sm()
{
    ZeroMemory(&Data,sizeof Data);
}
Item104Sm::~Item104Sm()
{
    if (ListNext) {
        delete ListNext;
        ListNext = 0;
    }
}
bool Item104Sm::LoadItemRecord(BYTE *record)
{
    if (*(record+0) != 'J' && *(record+1) != 'M')
        return false;

    memcpy(&Data,record,sizeof Data);
    Info = 0;
    return true;
}
BYTE* Item104Sm::GetItemRecord()
{
    return (BYTE*) &Data;
}
DWORD   Item104Sm::ItemRecordLength()
{
    return sizeof Data;
}
void Item104Sm::BlankItem()
{
    ZeroMemory(&Data,sizeof Data);

    Data.JM[0] = 0x4A;
    Data.JM[1] = 0x4D;
    Data.bitfield1 = 0x0010;
    Data.bitfield2 = 0x0038;
}
int Item104Sm::Container() const
{
    if ((Data.coordinates & 0xFC01) == 0x0000 && (Data.itemcode & 0x00000003) == 0x00) return CNT_INVENTORY;
    if ((Data.coordinates & 0xFC01) == 0x1000 && (Data.itemcode & 0x00000003) == 0x00) return CNT_STASH;
    if ((Data.coordinates & 0xFC01) == 0x0C00 && (Data.itemcode & 0x00000003) == 0x00) return CNT_CUBE;
    if ((Data.coordinates & 0xFC01) == 0xFC01 && (Data.itemcode & 0x00000003) == 0x03) return CNT_BELT;
    return CNT_NONE;
}
int Item104Sm::xPos() const
{
    switch(Container())
    {
    default:
        return 0;
    case CNT_INVENTORY:
    case CNT_STASH:
    case CNT_CUBE:
        return (Data.coordinates & 0x007C) >> 2;
    case CNT_BELT:
        return (Data.coordinates & 0x000C) >> 2;
    }
}
int Item104Sm::yPos() const
{
    switch(Container())
    {
    default:
        return 0;
    case CNT_INVENTORY:
    case CNT_STASH:
    case CNT_CUBE:
        return (Data.coordinates & 0x0180) >> 7;
    case CNT_BELT:
        return 3 - ((Data.coordinates & 0x0030) >> 4);
    }
}
int Item104Sm::xSize()
```

```
{
    if (!this && SelInfo) return SelInfo->SizeX;
    if (!this) return 1;

    if (!Info)
        if (!FindInfo())
            return 1;

    return Info->SizeX;
}
int Item104Sm::ySize()
{
    if (!this && SelInfo) return SelInfo->SizeY;
    if (!this) return 1;

    if (!Info)
        if (!FindInfo())
            return 1;

    return Info->SizeY;
}
bool Item104Sm::SetCoordinates(int Container,int xPos,int yPos)
{
    switch(Container)
    {
    default:
        return false;
    case CNT_INVENTORY:
        {
            if (xPos < 0 || xPos >= InvGrids.xInventory) return false;
            if (yPos < 0 || yPos >= InvGrids.yInventory) return false;

            Data.coordinates = (Data.coordinates & 0x03FE) | 0x0000;
            Data.coordinates = (Data.coordinates & 0xFF83) | (xPos << 2);
            Data.coordinates = (Data.coordinates & 0xFE7F) | (yPos << 7);

            Data.itemcode &= 0xFFFFFFFC;
            return true;
        }
    case CNT_STASH:
        {
            if (xPos < 0 || xPos >= InvGrids.xStash) return false;
            if (yPos < 0 || yPos >= InvGrids.yStash) return false;

            Data.coordinates = (Data.coordinates & 0x03FE) | 0x1000;
            Data.coordinates = (Data.coordinates & 0xFF83) | (xPos << 2);
            Data.coordinates = (Data.coordinates & 0xFE7F) | (yPos << 7);

            Data.itemcode &= 0xFFFFFFFC;
            return true;
        }
    case CNT_CUBE:
        {
            if (xPos < 0 || xPos >= InvGrids.xCube) return false;
            if (yPos < 0 || yPos >= InvGrids.yCube) return false;

            Data.coordinates = (Data.coordinates & 0x03FE) | 0x0C00;
            Data.coordinates = (Data.coordinates & 0xFF83) | (xPos << 2);
            Data.coordinates = (Data.coordinates & 0xFE7F) | (yPos << 7);

            Data.itemcode &= 0xFFFFFFFC;
            return true;
        }
    case CNT_BELT:
        {
            if (xPos < 0 || xPos >= InvGrids.xBelt) return false;
            if (yPos < 0 || yPos >= InvGrids.yBelt) return false;

            Data.coordinates = (Data.coordinates & 0x03FE) | 0xFC01;
            Data.coordinates = (Data.coordinates & 0xFFF3) | (xPos << 2);
            Data.coordinates = (Data.coordinates & 0xFFCF) | ((3-yPos) << 4);

            Data.itemcode |= 0x00000003;
            return true;
        }
    case CNT_SOCKET:
        {
            if (xPos < 0 || xPos >= 7) return false;
```

```
            Data.coordinates = (Data.coordinates & 0x03E0) | 0xFC1B;
            Data.coordinates = (Data.coordinates & 0xFC7F) | (xPos << 7);

            Data.itemcode |= 0x00000003;
            return true;
        }
    }
}
bool Item104Sm::FindInfo()
{
    Info = &itemunknown;

    for(int z=0;z<nItemInfos;z++)
    {
        // Check Item Code
        if (ItemInfos[z].IC == 0xFFFFFFFF) continue;
        if (ItemCode() != ItemInfos[z].IC) continue;

        Info = &ItemInfos[z];
    }

    if (!Info || !Info->BitmapID)
    {
        Info = &itemunknown;
        return false;
    }
    return true;
}
HBITMAP Item104Sm::GetBitmap()
{
    if (!Info) FindInfo();

    return ItemInfoGetBitmap(Info);
}
bool Item104Sm::FindGemInfo()
{
    GInfo = 0;
    for(int z=0;z<nGemInfos;z++)
    {
        // Check Item Code
        if (ItemCode() != GemInfos[z].IC) continue;
        GInfo = &GemInfos[z];
        return true;
    }
    return false;
}
DWORD Item104Sm::ItemCode()
{
    return ( (Data.itemcode & 0xFFFFFFFC) >> 2 ) |
           ( (Data.zero & 0x03) << 30 );
}
void Item104Sm::SetItemCode(DWORD x)
{
    Data.itemcode = (Data.itemcode & 0x00000003) | ((x << 2) & 0xFFFFFFFC);
    Data.zero = (Data.zero & 0xF7) | (BYTE(x >> 30) & 0x03);
    Info = 0;
}
const char* Item104Sm::RichText()
{
    char *s = RichTextTmp;
    StartRTF(s);

    strmcat(s,"\\cf0 ",Name(),0);

    if (Info->Description)
    {
        strmcat(s,"\\par\\cf0 ",Info->Description,0);
    }

    strcat(s,"\\par }");

    return s;
}
```

```
#include "JamellaD2E.h"

struct ItemInfo itemunknown =
{ "Unknown Item",'X',0,"Unknown Item",0x0000,0x0,0,'N',0xFFFFFFFF,0,IDB_ITEM_UNKNOWN,0,0,0,0x0,1,1,0,' ',0,0,0,0,0,0,0,0
,0,0,0,100,0,0,0,0,0 };

struct ItemInfo ItemInfos[] = {
{ "Hand Axe",'N',0,"Weapon Axe",0x0000,0x0,0,'N',0x20786168,0,IDB_ITEM_HANDAXE,1102,2,16,0xB,1,3,4,' ',28,0,0,0,1,3,6,3,
6,0,0,100,0,0,2,'W',0,0 },
{ "The Gnasher",'U',"Hand Axe","Weapon Axe",0x0000,0x00,0,'U',0x20786168,0,IDB_ITEM_THEGNASHER,1202,0,0,0xB,1,3,4,' ',14
0,0,0,0,1,3,6,3,6,0,0,100,0,0,2,'W',0,0 },
{ "Axe",'N',0,"Weapon Axe",0x0010,0x0,0,'N',0x20657861,0,IDB_ITEM_AXE,1102,2,16,0xB,2,3,4,' ',24,32,0,0,1,3,11,3,11,0,0,
100,0,0,3,'W',0,0 },
{ "Deathspade",'U',"Axe","Weapon Axe",0x0010,0x02,0,'U',0x20657861,0,IDB_ITEM_DEATHSPADE,1202,0,0,0xB,2,3,4,' ',120,32,0
,0,1,3,11,3,11,0,0,100,0,0,3,'W',0,0 },
{ "Double Axe",'N',0,"Weapon Axe",0x0020,0x0,0,'N',0x20786132,0,IDB_ITEM_DOUBLEAXE,1102,2,16,0xF,2,3,4,' ',24,43,0,0,1,5
,12,5,12,0,0,100,0,0,3,'W',0,0 },
{ "Berserker's Hatchet",'S',"Double Axe","Weapon Axe",0x0020,0x16,0,'S',0x20786132,0,IDB_ITEM_BERSERKERSHATCHET,20004,0,
0,0xF,2,3,4,' ',48,43,0,0,1,5,12,5,12,0,0,100,0,0,3,'W',0,0 },
{ "Bladebone",'U',"Double Axe","Weapon Axe",0x0020,0x04,0,'U',0x20786132,0,IDB_ITEM_DOUBLEAXE,1202,0,0,0xF,2,3,4,' ',120
,43,0,0,1,5,12,5,12,0,0,100,0,0,3,'W',0,0 },
{ "Military Pick",'N',0,"Weapon Axe",0x0030,0x0,0,'N',0x2069706D,0,IDB_ITEM_MILITARYPICK,1102,2,16,0xF,2,3,4,' ',26,49,3
3,0,1,6,10,6,10,0,0,100,0,0,3,'W',0,0 },
{ "Tancred's Crowbill",'S',"Military Pick","Weapon Axe",0x0030,0x10,0,'S',0x2069706D,0,IDB_ITEM_TANCREDSCROWBILL,20015,0
,0,0xF,2,3,4,' ',52,49,33,0,1,6,10,6,10,0,0,100,0,0,3,'W',0,0 },
{ "Mindrend",'U',"Military Pick","Weapon Axe",0x0030,0x06,0,'U',0x2069706D,0,IDB_ITEM_MINDREND,1202,0,0,0xF,2,3,4,' ',13
0,49,33,0,1,6,10,6,10,0,0,100,0,0,3,'W',0,0 },
{ "War Axe",'N',0,"Weapon Axe",0x0040,0x0,0,'N',0x20786177,0,IDB_ITEM_WARAXE,1102,2,16,0xB,2,3,4,' ',26,67,0,0,1,8,14,8,
14,0,0,100,0,0,3,'W',0,0 },
{ "Rakescar",'U',"War Axe","Weapon Axe",0x0040,0x08,0,'U',0x20786177,0,IDB_ITEM_RAKESCAR,1202,0,0,0xB,2,3,4,' ',130,67,0
,0,1,8,14,8,14,0,0,100,0,0,3,'W',0,0 },
{ "Large Axe",'N',0,"Weapon Axe",0x0050,0x0,0,'N',0x2078616C,0,IDB_ITEM_LARGEAXE,1102,2,16,0xB,2,3,4,' ',30,85,0,0,2,6,1
3,6,13,0,0,100,0,0,3,'W',0,0 },
{ "Axe of Fechmar",'U',"Large Axe","Weapon Axe",0x0050,0x0A,0,'U',0x2078616C,0,IDB_ITEM_FECHMARSAXE,1202,0,0,0xB,2,3,4,'
',150,85,0,0,2,6,13,6,13,0,0,100,0,0,3,'W',0,0 },
{ "Broad Axe",'N',0,"Weapon Axe",0x0060,0x0,0,'N',0x20786162,0,IDB_ITEM_BROADAXE,1102,2,16,0xB,2,3,4,' ',35,48,0,0,2,9,1
7,10,18,0,0,100,0,0,3,'W',0,0 },
{ "Goreshovel",'U',"Broad Axe","Weapon Axe",0x0060,0x0C,0,'U',0x20786162,0,IDB_ITEM_GORESHOVEL,1202,0,0,0xB,2,3,4,' ',17
5,48,0,0,2,9,17,10,18,0,0,100,0,0,3,'W',0,0 },
{ "Battle Axe",'N',0,"Weapon Axe",0x0070,0x0,0,'N',0x20787462,0,IDB_ITEM_BATTLEAXE,1102,2,16,0xB,2,3,4,' ',40,54,0,0,2,1
0,28,12,28,0,0,100,0,0,3,'W',0,0 },
{ "The Chieftan",'U',"Battle Axe","Weapon Axe",0x0070,0x0E,0,'U',0x20787462,0,IDB_ITEM_THECHIEFTAN,1202,0,0,0xB,2,3,4,'
',200,54,0,0,2,10,28,12,28,0,0,100,0,0,3,'W',0,0 },
{ "Great Axe",'N',0,"Weapon Axe",0x0080,0x0,0,'N',0x20786167,0,IDB_ITEM_GREATAXE,1102,2,16,0xB,2,4,4,' ',50,63,39,0,2,6,
24,8,26,0,0,100,0,0,3,'W',0,0 },
{ "Brainhew",'U',"Great Axe","Weapon Axe",0x0080,0x10,0,'U',0x20786167,0,IDB_ITEM_BRAINHEW,1202,0,0,0xB,2,4,4,' ',250,63
,39,0,2,6,24,8,26,0,0,100,0,0,3,'W',0,0 },
{ "Giant Axe",'N',0,"Weapon Axe",0x0090,0x0,0,'N',0x20786967,0,IDB_ITEM_GIANTAXE,1102,2,16,0xB,2,3,4,' ',50,70,0,0,2,26,
38,26,38,0,0,100,0,0,3,'W',0,0 },
{ "The Humongous",'U',"Giant Axe","Weapon Axe",0x0090,0x12,0,'U',0x20786967,0,IDB_ITEM_THEHUMONGOUS,1202,0,0,0xB,2,3,4,'
',250,84,0,0,2,26,38,26,38,0,0,100,0,0,3,'W',0,0 },
{ "Wand",'N',0,"Weapon Wand",0x00A0,0x0,0,'N',0x20646E77,0,IDB_ITEM_WAND,1111,32,256,0xB,1,2,4,'N',15,0,0,0,1,2,4,2,4,0,
0,100,0,0,0,' ',0,0 },
{ "Iro's Torch",'U',"Wand","Weapon Wand",0x00A0,0x14,0,'U',0x20646E77,0,IDB_ITEM_IROSTORCH,1209,0,0,0xB,1,2,4,' ',75,0,0
,0,1,2,4,2,4,0,0,100,0,0,0,' ',0,0 },
{ "Yew Wand",'N',0,"Weapon Wand",0x00B0,0x0,0,'N',0x206E7779,0,IDB_ITEM_YEWWAND,1111,32,256,0xB,1,2,4,'N',15,0,0,0,1,2,8
,2,8,0,0,100,0,0,0,' ',0,0 },
{ "Maelstromwrath",'U',"Yew Wand","Weapon Wand",0x00B0,0x16,0,'U',0x206E7779,0,IDB_ITEM_MAELSTROMWRATH,1209,0,0,0xB,1,2,
4,' ',75,0,0,0,1,2,8,2,8,0,0,100,0,0,0,' ',0,0 },
{ "Bone Wand",'N',0,"Weapon Wand",0x00C0,0x0,0,'N',0x206E7762,0,IDB_ITEM_BONEWAND,1111,32,256,0xB,1,2,4,'N',15,0,0,0,1,3
,7,3,7,0,0,100,0,0,0,' ',0,0 },
{ "Gravenspine",'U',"Bone Wand","Weapon Wand",0x00C0,0x18,0,'U',0x206E7762,0,IDB_ITEM_GRAVENSPINE,1209,0,0,0xB,1,2,4,' '
,75,0,0,0,1,3,7,3,7,0,0,100,0,0,0,' ',0,0 },
{ "Grim Wand",'N',0,"Weapon Wand",0x00D0,0x0,0,'N',0x206E7767,0,IDB_ITEM_GRIMWAND,1111,32,256,0xF,1,2,4,'N',15,0,0,0,1,5
,11,5,11,0,0,100,0,0,0,' ',0,0 },
{ "Infernal Torch",'S',"Grim Wand","Weapon Wand",0x00D0,0x14,0,'S',0x206E7767,0,IDB_ITEM_INFERNALTORCH,20010,0,0,0xF,1,2
,4,' ',30,0,0,0,1,5,11,5,11,0,0,100,0,0,0,' ',0,0 },
{ "Ume's Lament",'U',"Grim Wand","Weapon Wand",0x00D0,0x1A,0,'U',0x206E7767,0,IDB_ITEM_UMESLAMENT,1209,0,0,0xF,1,2,4,' '
,75,0,0,0,1,5,11,5,11,0,0,100,0,0,0,' ',0,0 },
{ "Club",'N',0,"Weapon Mace",0x00E0,0x0,0,'N',0x20626C63,0,IDB_ITEM_CLUB,1103,2,32,0xB,1,3,4,' ',24,0,0,0,1,1,6,1,6,0,0,
150,0,0,2,'W',0,0 },
{ "Felloak",'U',"Club","Weapon Mace",0x00E0,0x1C,0,'U',0x20626C63,0,IDB_ITEM_FELLOAK,1203,0,0,0xB,1,3,4,' ',120,0,0,0,1,
1,6,1,6,0,0,150,0,0,2,'W',0,0 },
{ "Scepter",'N',0,"Weapon Scepter",0x00F0,0x0,0,'N',0x20706373,0,IDB_ITEM_SCEPTER,1112,16,128,0xB,1,3,4,'P',50,25,0,0,1,
5,8,5,8,0,0,150,0,0,2,'W',0,0 },
{ "Knell Striker",'U',"Scepter","Weapon Scepter",0x00F0,0x1E,0,'U',0x20706373,0,IDB_ITEM_KNELLSTRIKER,1210,0,0,0xB,1,3,4
,' ',250,25,0,0,1,5,8,5,8,0,0,150,0,0,2,'W',0,0 },
{ "Grand Scepter",'N',0,"Weapon Scepter",0x0100,0x0,0,'N',0x20637367,0,IDB_ITEM_GRANDSCEPTER,1112,16,128,0xF,1,3,4,'P',6
0,37,0,0,1,6,11,6,11,0,0,150,0,0,2,'W',0,0 },
```

```
{ "Civerb's Cudgel",'S',"Grand Scepter","Weapon Scepter",0x0100,0x00,0,'S',0x20637367,0,IDB_ITEM_CIVERBSCUDGEL,20006,0,0
,0xF,1,3,4,' ',120,37,0,0,1,6,11,6,11,0,0,150,0,0,2,'W',0,0 },
{ "Rusthandle",'U',"Grand Scepter","Weapon Scepter",0x0100,0x20,0,'U',0x20637367,0,IDB_ITEM_RUSTHANDLE,1210,0,0,0xF,1,3,
4,' ',255,37,0,0,1,6,11,6,11,0,0,150,0,0,2,'W',0,0 },
{ "War Scepter",'N',0,"Weapon Scepter",0x0110,0x0,0,'N',0x20707377,0,IDB_ITEM_WARSCEPTER,1112,16,128,0xF,2,3,4,'P',70,55
,0,0,1,7,14,7,14,0,0,150,0,0,2,'W',0,0 },
{ "Milabrega's Rod",'S',"War Scepter","Weapon Scepter",0x0110,0x0C,0,'S',0x20707377,0,IDB_ITEM_MILABREGASROD,20013,0,0,0
xF,2,3,4,' ',140,55,0,0,1,7,14,7,14,0,0,150,0,0,2,'W',0,0 },
{ "Stormeye",'U',"War Scepter","Weapon Scepter",0x0110,0x22,0,'U',0x20707377,0,IDB_ITEM_WARSCEPTER,1210,0,0,0xF,2,3,4,'
',255,55,0,0,1,7,14,7,14,0,0,150,0,0,2,'W',0,0 },
{ "Spiked Club",'N',0,"Weapon Mace",0x0120,0x0,0,'N',0x20637073,0,IDB_ITEM_SPIKEDCLUB,1103,2,32,0xB,1,3,4,' ',36,0,0,0,1
,5,6,5,6,0,0,150,0,0,2,'W',0,0 },
{ "Stoutnail",'U',"Spiked Club","Weapon Mace",0x0120,0x24,0,'U',0x20637073,0,IDB_ITEM_STOUTNAIL,1203,0,0,0xB,1,3,4,' ',1
80,0,0,0,1,5,6,5,6,0,0,150,0,0,2,'W',0,0 },
{ "Mace",'N',0,"Weapon Mace",0x0130,0x0,0,'N',0x2063616D,0,IDB_ITEM_MACE,1103,2,32,0xB,1,3,4,' ',60,27,0,0,1,3,10,3,10,0
,0,150,0,0,2,'W',0,0 },
{ "Crushflange",'U',"Mace","Weapon Mace",0x0130,0x26,0,'U',0x2063616D,0,IDB_ITEM_CRUSHFLANGE,1203,0,0,0xB,1,3,4,' ',255,
27,0,0,1,3,10,3,10,0,0,150,0,0,2,'W',0,0 },
{ "Morning Star",'N',0,"Weapon Mace",0x0140,0x0,0,'N',0x2074736D,0,IDB_ITEM_MORNINGSTAR,1103,2,32,0xB,1,3,4,' ',72,36,0,
0,1,5,12,5,12,0,0,150,0,0,2,'W',0,0 },
{ "Bloodrise",'U',"Morning Star","Weapon Mace",0x0140,0x28,0,'U',0x2074736D,0,IDB_ITEM_BLOODRISE,1203,0,0,0xB,1,3,4,' ',
255,36,0,0,1,5,12,5,12,0,0,150,0,0,2,'W',0,0 },
{ "Flail",'N',0,"Weapon Mace",0x0150,0x0,0,'N',0x20616C66,0,IDB_ITEM_FLAIL,1103,2,32,0xB,2,3,4,' ',30,41,35,0,1,1,15,1,1
5,0,0,150,0,0,3,'W',0,0 },
{ "The General's Tan Do Li Ga",'U',"Flail","Weapon Mace",0x0150,0x2A,0,'U',0x20616C66,0,IDB_ITEM_THEGENERALSTANDOLIGA,12
03,0,0,0xB,2,3,4,' ',197,41,35,0,1,1,15,1,15,0,0,150,0,0,3,'W',0,0 },
{ "War Hammer",'N',0,"Weapon Mace",0x0160,0x0,0,'N',0x206D6877,0,IDB_ITEM_WARHAMMER,1103,2,32,0xB,2,3,4,' ',55,53,0,0,1,
11,20,11,20,0,0,150,0,0,3,'W',0,0 },
{ "Ironstone",'U',"War Hammer","Weapon Mace",0x0160,0x2C,0,'U',0x206D6877,0,IDB_ITEM_IRONSTONE,1203,0,0,0xB,2,3,4,' ',25
5,53,0,0,1,11,20,11,20,0,0,150,0,0,3,'W',0,0 },
{ "Maul",'N',0,"Weapon Mace",0x0170,0x0,0,'N',0x2075616D,0,IDB_ITEM_MAUL,1103,2,32,0xB,2,4,4,' ',60,69,0,0,2,1,1,30,40,0
,0,150,0,0,3,'W',0,0 },
{ "Bonesnap",'U',"Maul","Weapon Mace",0x0170,0x2E,0,'U',0x2075616D,0,IDB_ITEM_BONESNAP,1203,0,0,0xB,2,4,4,' ',255,69,0,0
,2,1,1,30,40,0,0,150,0,0,3,'W',0,0 },
{ "Great Maul",'N',0,"Weapon Mace",0x0180,0x0,0,'N',0x20616D67,0,IDB_ITEM_GREATMAUL,1103,2,32,0xB,2,3,4,' ',60,99,0,0,2,
1,1,35,55,0,0,150,0,0,3,'W',0,0 },
{ "Steeldriver",'U',"Great Maul","Weapon Mace",0x0180,0x30,0,'U',0x20616D67,0,IDB_ITEM_STEELDRIVER,1203,0,0,0xB,2,3,4,'
',255,50,0,0,2,1,1,35,55,0,0,150,0,0,3,'W',0,0 },
{ "Short Sword",'N',0,"Weapon Sword",0x0190,0x0,0,'N',0x20647373,0,IDB_ITEM_SHORTSWORD,1101,2,8,0xB,1,3,4,' ',24,0,0,0,1
,2,7,2,7,0,0,100,0,0,2,'W',0,0 },
{ "Rixot's Keen",'U',"Short Sword","Weapon Sword",0x0190,0x32,0,'U',0x20647373,0,IDB_ITEM_RIXOTSKEEN,1201,0,0,0xB,1,3,4,
' ',120,0,0,0,1,2,7,2,7,0,0,100,0,0,2,'W',0,0 },
{ "Scimitar",'N',0,"Weapon Sword",0x01A0,0x0,0,'N',0x206D6373,0,IDB_ITEM_SCIMITAR,1101,2,8,0xB,1,3,4,' ',22,0,21,0,1,2,6
,2,6,0,0,100,0,0,2,'W',0,0 },
{ "Blood Crescent",'U',"Scimitar","Weapon Sword",0x01A0,0x34,0,'U',0x206D6373,0,IDB_ITEM_BLOODCRESCENT,1201,0,0,0xB,1,3,
4,' ',110,0,21,0,1,2,6,2,6,0,0,100,0,0,2,'W',0,0 },
{ "Sabre",'N',0,"Weapon Sword",0x01B0,0x0,0,'N',0x20726273,0,IDB_ITEM_SABRE,1101,2,8,0xF,1,3,4,' ',32,25,25,0,1,3,8,3,8,
0,0,100,0,0,2,'W',0,0 },
{ "Angelical Sickle",'S',"Sabre","Weapon Sword",0x01B0,0x1A,0,'S',0x20726273,0,IDB_ITEM_SABRE,20001,0,0,0xF,1,3,4,' ',64
,25,25,0,1,3,8,3,8,0,0,100,0,0,2,'W',0,0 },
{ "Krintizs Skewer",'U',"Sabre","Weapon Sword",0x01B0,0x36,0,'U',0x20726273,0,IDB_ITEM_KRINTIZSSKEWER,1201,0,0,0xF,1,3,4
,' ',160,25,25,0,1,3,8,3,8,0,0,100,0,0,2,'W',0,0 },
{ "Falchion",'N',0,"Weapon Sword",0x01C0,0x0,0,'N',0x20636C66,0,IDB_ITEM_FALCHION,1101,2,8,0xB,1,3,4,' ',32,33,0,0,1,7,1
5,7,15,0,0,100,0,0,2,'W',0,0 },
{ "Gleamscythe",'U',"Falchion","Weapon Sword",0x01C0,0x38,0,'U',0x20636C66,0,IDB_ITEM_GLEAMSCYTHE,1201,0,0,0xB,1,3,4,' '
,160,33,0,0,1,7,15,7,15,0,0,100,0,0,2,'W',0,0 },
{ "Crystal Sword",'N',0,"Weapon Sword",0x01D0,0x0,0,'N',0x20737263,0,IDB_ITEM_CRYSTALSWORD,1101,2,8,0xB,2,3,4,' ',10,43,
0,0,1,5,15,5,15,0,0,100,0,0,3,'W',0,0 },
{ "Azurewrath",'U',"Crystal Sword","Weapon Sword",0x01D0,0x3A,0,'U',0x20737263,0,IDB_ITEM_AZUREWRATH,1201,0,0,0xB,2,3,4,
' ',50,43,0,0,1,5,15,5,15,0,0,100,0,0,3,'W',0,0 },
{ "Broad Sword",'N',0,"Weapon Sword",0x01E0,0x0,0,'N',0x20647362,0,IDB_ITEM_BROADSWORD,1101,2,8,0xF,2,3,4,' ',32,48,0,0,
1,7,14,7,14,0,0,100,0,0,3,'W',0,0 },
{ "Isenhart's Lightbrand",'S',"Broad Sword","Weapon Sword",0x01E0,0x08,0,'S',0x20647362,0,IDB_ITEM_ISENHARTSLIGHTBRAND,2
0012,0,0,0xF,2,3,4,' ',64,48,0,0,1,7,14,7,14,0,0,100,0,0,3,'W',0,0 },
{ "Griswold's Edge",'U',"Broad Sword","Weapon Sword",0x01E0,0x3C,0,'U',0x20647362,0,IDB_ITEM_GRISWOLDSEDGE,1201,0,0,0xF,
2,3,4,' ',160,48,0,0,1,7,14,7,14,0,0,100,0,0,3,'W',0,0 },
{ "Long Sword",'N',0,"Weapon Sword",0x01F0,0x0,0,'N',0x2064736C,0,IDB_ITEM_LONGSWORD,1101,2,8,0xF,2,3,4,' ',44,55,39,0,1
,3,19,3,19,0,0,100,0,0,3,'W',0,0 },
{ "Cleglaw's Tooth",'S',"Long Sword","Weapon Sword",0x01F0,0x04,0,'S',0x2064736C,0,IDB_ITEM_CLEGLAWSTOOTH,20007,0,0,0xF,
2,3,4,' ',88,55,39,0,1,3,19,3,19,0,0,100,0,0,3,'W',0,0 },
{ "Hellplague",'U',"Long Sword","Weapon Sword",0x01F0,0x3E,0,'U',0x2064736C,0,IDB_ITEM_HELLPLAGUE,1201,0,0,0xF,2,3,4,' '
,220,55,39,0,1,3,19,3,19,0,0,100,0,0,3,'W',0,0 },
{ "War Sword",'N',0,"Weapon Sword",0x0200,0x0,0,'N',0x20647377,0,IDB_ITEM_WARSWORD,1101,2,8,0xF,1,3,4,' ',44,71,45,0,1,8
,20,8,20,0,0,100,0,0,2,'W',0,0 },
{ "Death's Touch",'S',"War Sword","Weapon Sword",0x0200,0x18,0,'S',0x20647377,0,IDB_ITEM_DEATHSTOUCH,20008,0,0,0xF,1,3,4
,' ',88,71,45,0,1,8,20,8,20,0,0,100,0,0,2,'W',0,0 },
{ "Culwen's Point",'U',"War Sword","Weapon Sword",0x0200,0x40,0,'U',0x20647377,0,IDB_ITEM_WARSWORD,1201,0,0,0xF,1,3,4,'
',220,71,45,0,1,8,20,8,20,0,0,100,0,0,2,'W',0,0 },
{ "Two Handed Sword",'N',0,"Weapon Sword",0x0210,0x0,0,'N',0x20736832,0,IDB_ITEM_2HSWORD,1101,2,8,0xB,1,4,4,' ',44,35,27
```

```
,0,2,2,9,8,17,0,0,100,0,0,3,'W',0,0 },
{ "Shadowfang","U","Two Handed Sword","Weapon Sword",0x0210,0x42,0,'U',0x20736832,0,IDB_ITEM_SHADOWFANG,1201,0,0,0xB,1,4
,4,' ',220,35,27,0,2,2,9,8,17,0,0,100,0,0,3,'W',0,0 },
{ "Claymore","N",0,"Weapon Sword",0x0220,0x0,0,'N',0x206D6C63,0,IDB_ITEM_CLAYMORE,1101,2,8,0xB,1,4,4,' ',50,47,0,0,2,5,1
2,13,30,0,0,100,0,0,3,'W',0,0 },
{ "Soulflay","U","Claymore","Weapon Sword",0x0220,0x44,0,'U',0x206D6C63,0,IDB_ITEM_SOULFLAY,1201,0,0,0xB,1,4,4,' ',250,4
7,0,0,2,5,12,13,30,0,0,100,0,0,3,'W',0,0 },
{ "Giant Sword","N",0,"Weapon Sword",0x0230,0x0,0,'N',0x20736967,0,IDB_ITEM_GIANTSWORD,1101,2,8,0xB,1,4,4,' ',50,56,34,0
,2,3,16,9,28,0,0,100,0,0,3,'W',0,0 },
{ "Kinemil's Awl","U","Giant Sword","Weapon Sword",0x0230,0x46,0,'U',0x20736967,0,IDB_ITEM_KINEMILSAWL,1201,0,0,0xB,1,4,
4,' ',250,56,34,0,2,3,16,9,28,0,0,100,0,0,3,'W',0,0 },
{ "Bastard Sword","N",0,"Weapon Sword",0x0240,0x0,0,'N',0x20777362,0,IDB_ITEM_BASTARDSWORD,1101,2,8,0xB,1,4,4,' ',40,62,
0,0,2,6,18,20,28,0,0,100,0,0,3,'W',0,0 },
{ "Blacktongue","U","Bastard Sword","Weapon Sword",0x0240,0x48,0,'U',0x20777362,0,IDB_ITEM_BLACKTONGUE,1201,0,0,0xB,1,4,
4,' ',200,62,0,0,2,6,18,20,28,0,0,100,0,0,3,'W',0,0 },
{ "Flamberge","N",0,"Weapon Sword",0x0250,0x0,0,'N',0x20626C66,0,IDB_ITEM_FLAMBERGE,1101,2,8,0xB,2,4,4,' ',50,70,49,0,2,
9,15,13,26,0,0,100,0,0,3,'W',0,0 },
{ "Ripsaw","U","Flamberge","Weapon Sword",0x0250,0x4A,0,'U',0x20626C66,0,IDB_ITEM_RIPSAW,1201,0,0,0xB,2,4,4,' ',250,70,4
9,0,2,9,15,13,26,0,0,100,0,0,3,'W',0,0 },
{ "Great Sword","N",0,"Weapon Sword",0x0260,0x0,0,'N',0x20647367,0,IDB_ITEM_GREATSWORD,1101,2,8,0xB,2,4,4,' ',50,100,60,
0,2,12,18,25,42,0,0,100,0,0,3,'W',0,0 },
{ "The Patriarch","U","Great Sword","Weapon Sword",0x0260,0x4C,0,'U',0x20647367,0,IDB_ITEM_THEPATRIARCH,1201,0,0,0xB,2,4
,4,' ',250,100,60,0,2,12,18,25,42,0,0,100,0,0,3,'W',0,0 },
{ "Dagger","N",0,"Weapon Dagger",0x0270,0x0,0,'N',0x20726764,0,IDB_ITEM_DAGGER,1107,2,8,0xB,1,2,4,' ',16,0,0,0,1,1,4,1,4
,0,0,100,0,0,1,'W',0,0 },
{ "Gull","U","Dagger","Weapon Dagger",0x0270,0x4E,0,'U',0x20726764,0,IDB_ITEM_DAGGER,1204,0,0,0xB,1,2,4,' ',120,0,0,0,1,
1,4,1,4,0,0,100,0,0,1,'W',0,0 },
{ "Dirk","N",0,"Weapon Dagger",0x0280,0x0,0,'N',0x20726964,0,IDB_ITEM_DIRK,1107,2,8,0xB,1,2,4,' ',20,0,25,0,1,3,7,3,7,0,
0,100,0,0,1,'W',0,0 },
{ "The Diggler","U","Dirk","Weapon Dagger",0x0280,0x50,0,'U',0x20726964,0,IDB_ITEM_THEDIGGLER,1204,0,0,0xB,1,2,4,' ',100
,0,25,0,1,3,7,3,7,0,0,100,0,0,1,'W',0,0 },
{ "Kris","N",0,"Weapon Dagger",0x0290,0x0,0,'N',0x2069726B,0,IDB_ITEM_KRIS,1107,2,8,0xB,1,3,4,' ',24,0,45,0,1,2,9,2,9,0,
0,100,0,0,2,'W',0,0 },
{ "The Jade Tan Do","U","Kris","Weapon Dagger",0x0290,0x52,0,'U',0x2069726B,0,IDB_ITEM_THEJADETANDO,1204,0,0,0xB,1,3,4,'
',120,0,45,0,1,2,9,2,9,0,0,100,0,0,2,'W',0,0 },
{ "Blade","N",0,"Weapon Dagger",0x02A0,0x0,0,'N',0x20646C62,0,IDB_ITEM_BLADE,1107,2,8,0xB,1,3,4,' ',24,35,51,0,1,4,12,4,
12,0,0,100,0,0,1,'W',0,0 },
{ "Irice's Shard","U","Blade","Weapon Dagger",0x02A0,0x54,0,'U',0x20646C62,0,IDB_ITEM_IRICESSHARD,1204,0,0,0xB,1,3,4,' '
,120,35,51,0,1,4,12,4,12,0,0,100,0,0,1,'W',0,0 },
{ "Throwing Knife","N",0,"Weapon Throwing",0x02B0,0x0,0,'N',0x20666B74,0,IDB_ITEM_THROWINGKNIFE,1108,0,0,0x1,1,2,4,' ',0
,0,21,0,1,2,3,2,3,4,9,100,0,0,0,' ',75,0 },
{ "Throwing Axe","N",0,"Weapon Throwing",0x02C0,0x0,0,'N',0x20786174,0,IDB_ITEM_THROWINGAXE,1108,0,0,0x1,1,2,4,' ',0,0,5
1,0,1,4,7,4,7,8,12,100,0,0,0,' ',32,0 },
{ "Balanced Knife","N",0,"Weapon Throwing",0x02D0,0x0,0,'N',0x20666B62,0,IDB_ITEM_BALANCEDKNIFE,1108,0,0,0x1,1,2,4,' ',0
,0,40,0,1,1,8,1,8,6,11,100,0,0,0,' ',60,0 },
{ "Balanced Axe","N",0,"Weapon Throwing",0x02E0,0x0,0,'N',0x206C6162,0,IDB_ITEM_BALANCEDAXE,1108,0,0,0x1,2,3,4,' ',0,0,5
7,0,1,5,10,5,10,12,15,100,0,0,0,' ',24,0 },
{ "Javelin","N",0,"Weapon Javelin",0x02F0,0x0,0,'N',0x2076616A,0,IDB_ITEM_JAVELIN,1104,0,0,0x1,1,3,4,'A',0,0,0,0,1,1,5,1
,5,6,14,100,0,0,0,' ',60,0 },
{ "Pilum","N",0,"Weapon Javelin",0x0300,0x0,0,'N',0x206C6970,0,IDB_ITEM_PILUM,1104,0,0,0x1,1,3,4,'A',0,0,45,0,1,4,9,4,9,
7,20,100,0,0,0,' ',50,0 },
{ "Short Spear","N",0,"Weapon Javelin",0x0310,0x0,0,'N',0x20707373,0,IDB_ITEM_SHORTSPEAR,1104,0,0,0x1,1,3,4,'A',0,40,40,
0,1,2,13,2,13,14,25,100,0,0,0,' ',40,0 },
{ "Glaive","N",0,"Weapon Javelin",0x0320,0x0,0,'N',0x20766C67,0,IDB_ITEM_GLAIVE,1104,0,0,0x1,1,4,4,'A',0,52,35,0,1,5,17,
5,17,2,10,100,0,0,0,' ',20,0 },
{ "Throwing Spear","N",0,"Weapon Javelin",0x0330,0x0,0,'N',0x20707374,0,IDB_ITEM_THROWINGSPEAR,1104,0,0,0x1,1,4,4,'A',0,
0,65,0,1,5,15,5,15,8,28,100,0,0,0,' ',80,0 },
{ "Spear","N",0,"Weapon Spear",0x0340,0x0,0,'N',0x20727073,0,IDB_ITEM_SPEAR,1105,2,64,0xB,2,4,4,' ',30,0,0,0,2,3,15,3,15
,0,0,100,0,0,3,'W',0,0 },
{ "The Dragon Chang","U","Spear","Weapon Spear",0x0340,0x56,0,'U',0x20727073,0,IDB_ITEM_THEDRAGONCHANG,1205,0,0,0xB,2,4,
4,' ',150,0,0,0,2,3,15,3,15,0,0,100,0,0,3,'W',0,0 },
{ "Trident","N",0,"Weapon Spear",0x0350,0x0,0,'N',0x20697274,0,IDB_ITEM_TRIDENT,1105,2,64,0xB,2,4,4,' ',35,38,0,0,2,9,15
,9,15,0,0,100,0,0,3,'W',0,0 },
{ "Razortine","U","Trident","Weapon Spear",0x0350,0x58,0,'U',0x20697274,0,IDB_ITEM_RAZORTINE,1205,0,0,0xB,2,4,4,' ',175,
38,0,0,2,9,15,9,15,0,0,100,0,0,3,'W',0,0 },
{ "Brandistock","N",0,"Weapon Spear",0x0360,0x0,0,'N',0x206E7262,0,IDB_ITEM_BRANDISTOCK,1105,2,64,0xB,2,4,4,' ',28,40,50
,0,2,7,17,7,17,0,0,100,0,0,3,'W',0,0 },
{ "Bloodthief","U","Brandistock","Weapon Spear",0x0360,0x5A,0,'U',0x206E7262,0,IDB_ITEM_BLOODTHIEF,1205,0,0,0xB,2,4,4,'
',140,40,50,0,2,7,17,7,17,0,0,100,0,0,3,'W',0,0 },
{ "Spetum","N",0,"Weapon Spear",0x0370,0x0,0,'N',0x20747073,0,IDB_ITEM_SPETUM,1105,2,64,0xB,2,4,4,' ',28,54,0,0,2,15,21,
15,21,0,0,100,0,0,3,'W',0,0 },
{ "Lance of Yaggai","U","Spetum","Weapon Spear",0x0370,0x5C,0,'U',0x20747073,0,IDB_ITEM_LANCEOFYAGGAI,1205,0,0,0xB,2,4,4
,' ',140,54,0,0,2,15,21,15,21,0,0,100,0,0,3,'W',0,0 },
{ "Pike","N",0,"Weapon Spear",0x0380,0x0,0,'N',0x206B6970,0,IDB_ITEM_PIKE,1105,2,64,0xB,2,4,4,' ',25,60,45,0,2,14,63,14,
63,0,0,100,0,0,3,'W',0,0 },
{ "The Tannr Gorerod","U","Pike","Weapon Spear",0x0380,0x5E,0,'U',0x206B6970,0,IDB_ITEM_THETANNRGOREROD,1205,0,0,0xB,2,4
,4,' ',125,60,45,0,2,14,63,14,63,0,0,100,0,0,3,'W',0,0 },
{ "Bardiche","N",0,"Weapon Polearm",0x0390,0x0,0,'N',0x20726162,0,IDB_ITEM_BARDICHE,1106,2,64,0xB,2,4,4,' ',50,40,0,0,2,
1,25,1,25,0,0,100,0,0,3,'W',0,0 },
```

```
{ "Dimoak's Hew",'U',"Bardiche","Weapon Polearm",0x0390,0x60,0,'U',0x20726162,0,IDB_ITEM_DIMOAKSHEW,1206,0,0,0xB,2,4,4,'
 ',250,40,0,0,2,1,25,1,25,0,0,100,0,0,3,'W',0,0 },
{ "Voulge",'N',0,"Weapon Polearm",0x03A0,0x0,0,'N',0x20756F76,0,IDB_ITEM_VOULGE,1106,2,64,0xB,2,4,4,' ',50,50,0,0,2,6,20
,6,20,0,0,100,0,0,3,'W',0,0 },
{ "Steelgoad",'U',"Voulge","Weapon Polearm",0x03A0,0x62,0,'U',0x20756F76,0,IDB_ITEM_STEELGOAD,1206,0,0,0xB,2,4,4,' ',250
,50,0,0,2,6,20,6,20,0,0,100,0,0,3,'W',0,0 },
{ "Scythe",'N',0,"Weapon Polearm",0x03B0,0x0,0,'N',0x20796373,0,IDB_ITEM_SCYTHE,1106,2,64,0xB,2,4,4,' ',65,47,41,0,2,8,2
0,8,20,0,0,100,0,0,3,'W',0,0 },
{ "Soul Harvest",'U',"Scythe","Weapon Polearm",0x03B0,0x64,0,'U',0x20796373,0,IDB_ITEM_SOULHARVEST,1206,0,0,0xB,2,4,4,'
',255,41,41,0,2,8,20,8,20,0,0,100,0,0,3,'W',0,0 },
{ "Pole Axe",'N',0,"Weapon Polearm",0x03C0,0x0,0,'N',0x20786170,0,IDB_ITEM_POLEAXE,1106,2,64,0xB,2,4,4,' ',65,62,0,0,2,1
8,30,18,30,0,0,100,0,0,3,'W',0,0 },
{ "The Battlebranch",'U',"Pole Axe","Weapon Polearm",0x03C0,0x66,0,'U',0x20786170,0,IDB_ITEM_THEBATTLEBRANCH,1206,0,0,0x
B,2,4,4,' ',255,62,0,0,2,18,30,18,30,0,0,100,0,0,3,'W',0,0 },
{ "Halberd",'N',0,"Weapon Polearm",0x03D0,0x0,0,'N',0x206C6168,0,IDB_ITEM_HALBERD,1106,2,64,0xB,2,4,4,' ',55,75,47,0,2,1
2,35,12,40,0,0,100,0,0,3,'W',0,0 },
{ "Woestave",'U',"Halberd","Weapon Polearm",0x03D0,0x68,0,'U',0x206C6168,0,IDB_ITEM_WOESTAVE,1206,0,0,0xB,2,4,4,' ',255,
75,47,0,2,12,35,12,40,0,0,100,0,0,3,'W',0,0 },
{ "War Scythe",'N',0,"Weapon Polearm",0x03E0,0x0,0,'N',0x20637377,0,IDB_ITEM_WARSCYTHE,1106,2,64,0xB,2,4,4,' ',55,80,80,
0,2,15,32,15,32,0,0,100,0,0,3,'W',0,0 },
{ "The Grim Reaper",'U',"War Scythe","Weapon Polearm",0x03E0,0x6A,0,'U',0x20637377,0,IDB_ITEM_WARSCYTHE,1206,0,0,0xB,2,4
,4,' ',255,80,80,0,2,15,32,15,32,0,0,100,0,0,3,'W',0,0 },
{ "Short Staff",'N',0,"Weapon Staff",0x03F0,0x0,0,'N',0x20747373,0,IDB_ITEM_SHORTSTAFF,1113,64,512,0xB,1,3,4,'S',20,0,0,
0,2,1,5,1,5,0,0,100,0,0,2,'W',0,0 },
{ "Bane Ash",'U',"Short Staff","Weapon Staff",0x03F0,0x6C,0,'U',0x20747373,0,IDB_ITEM_BANEASH,1211,0,0,0xB,1,3,4,' ',100
,0,0,0,2,1,5,1,5,0,0,100,0,0,2,'W',0,0 },
{ "Long Staff",'N',0,"Weapon Staff",0x0400,0x0,0,'N',0x2074736C,0,IDB_ITEM_LONGSTAFF,1113,64,512,0xB,1,4,4,'S',30,0,0,0,
2,2,8,2,8,0,0,100,0,0,3,'W',0,0 },
{ "Serpent Lord",'U',"Long Staff","Weapon Staff",0x0400,0x6E,0,'U',0x2074736C,0,IDB_ITEM_SERPENTLORD,1211,0,0,0xB,1,4,4,
' ',150,0,0,0,2,2,8,2,8,0,0,100,0,0,3,'W',0,0 },
{ "Gnarled Staff",'N',0,"Weapon Staff",0x0410,0x0,0,'N',0x20747363,0,IDB_ITEM_GNARLEDSTAFF,1113,64,512,0xB,1,4,4,'S',35,
0,0,0,2,4,12,4,12,0,0,100,0,0,3,'W',0,0 },
{ "Lazarus' Spire",'U',"Gnarled Staff","Weapon Staff",0x0410,0x70,0,'U',0x20747363,0,IDB_ITEM_LAZARUSSPIRE,1211,0,0,0xB,
1,4,4,' ',175,0,0,0,2,4,12,4,12,0,0,100,0,0,3,'W',0,0 },
{ "Battle Staff",'N',0,"Weapon Staff",0x0420,0x0,0,'N',0x20747362,0,IDB_ITEM_BATTLESTAFF,1113,64,512,0xF,1,4,4,'S',40,0,
0,0,2,6,13,6,13,0,0,100,0,0,3,'W',0,0 },
{ "Cathan's Rule",'S',"Battle Staff","Weapon Staff",0x0420,0x0E,0,'S',0x20747362,0,IDB_ITEM_CATHANSRULE,20005,0,0,0xF,1,
4,4,' ',80,0,0,0,2,6,13,6,13,0,0,100,0,0,3,'W',0,0 },
{ "The Salamander",'U',"Battle Staff","Weapon Staff",0x0420,0x72,0,'U',0x20747362,0,IDB_ITEM_THESALAMANDER,1211,0,0,0xF,
1,4,4,' ',200,0,0,0,2,6,13,6,13,0,0,100,0,0,3,'W',0,0 },
{ "War Staff",'N',0,"Weapon Staff",0x0430,0x0,0,'N',0x20747377,0,IDB_ITEM_WARSTAFF,1113,64,512,0xF,2,4,4,'S',50,0,0,0,2,
12,28,12,28,0,0,100,0,0,3,'W',0,0 },
{ "Arcanna's Deathwand",'S',"War Staff","Weapon Staff",0x0430,0x1E,0,'S',0x20747377,0,IDB_ITEM_ARCANNASDEATHWAND,20002,0
,0,0xF,2,4,4,' ',100,0,0,0,2,12,28,12,28,0,0,100,0,0,3,'W',0,0 },
{ "The Iron Lang Bong",'U',"War Staff","Weapon Staff",0x0430,0x74,0,'U',0x20747377,0,IDB_ITEM_THEIRONLANGBONG,1211,0,0,0
xF,2,4,4,' ',250,0,0,0,2,12,28,12,28,0,0,100,0,0,3,'W',0,0 },
{ "Short Bow",'N',0,"Weapon Bow",0x0440,0x0,0,'N',0x20776273,0,IDB_ITEM_SHORTBOW,1109,128,1024,0xB,2,3,4,' ',0,0,15,0,2,
1,4,1,4,1,4,100,0,0,3,'W',0,0 },
{ "Pluckeye",'U',"Short Bow","Weapon Bow",0x0440,0x76,0,'U',0x20776273,0,IDB_ITEM_PLUCKEYE,1207,0,0,0xB,2,3,4,' ',0,0,15
,0,2,1,4,1,4,1,4,100,0,0,3,'W',0,0 },
{ "Hunter's Bow",'N',0,"Weapon Bow",0x0450,0x0,0,'N',0x20776268,0,IDB_ITEM_HUNTERSBOW,1109,128,1024,0xB,2,3,4,' ',0,0,28
,0,2,2,7,2,7,2,6,100,0,0,3,'W',0,0 },
{ "Witherstring",'U',"Hunter's Bow","Weapon Bow",0x0450,0x78,0,'U',0x20776268,0,IDB_ITEM_WITHERSTRING,1207,0,0,0xB,2,3,4
,' ',0,0,28,0,2,2,7,2,7,2,6,100,0,0,3,'W',0,0 },
{ "Long Bow",'N',0,"Weapon Bow",0x0460,0x0,0,'N',0x2077626C,0,IDB_ITEM_LONGBOW,1109,128,1024,0xB,2,4,4,' ',0,22,19,0,2,3
,10,3,10,3,10,100,0,0,3,'W',0,0 },
{ "Raven's Claw",'U',"Long Bow","Weapon Bow",0x0460,0x7A,0,'U',0x2077626C,0,IDB_ITEM_RIMERAVEN,1207,0,0,0xB,2,4,4,' ',0,
22,19,0,2,3,10,3,10,3,10,100,0,0,3,'W',0,0 },
{ "Composite Bow",'N',0,"Weapon Bow",0x0470,0x0,0,'N',0x20776263,0,IDB_ITEM_COMPOSITEBOW,1109,128,1024,0xB,2,3,4,' ',0,2
5,35,0,2,4,8,4,8,4,8,100,0,0,3,'W',0,0 },
{ "Rogue's Bow",'U',"Composite Bow","Weapon Bow",0x0470,0x7C,0,'U',0x20776263,0,IDB_ITEM_PIERCERIB,1207,0,0,0xB,2,3,4,'
',0,25,35,0,2,4,8,4,8,4,8,100,0,0,3,'W',0,0 },
{ "Short Battle Bow",'N',0,"Weapon Bow",0x0480,0x0,0,'N',0x20626273,0,IDB_ITEM_SHORTBATTLEBOW,1109,128,1024,0xB,2,3,4,'
',0,30,40,0,2,5,11,5,11,5,11,100,0,0,3,'W',0,0 },
{ "Stormstrike",'U',"Short Battle Bow","Weapon Bow",0x0480,0x7E,0,'U',0x20626273,0,IDB_ITEM_PULLSPITE,1207,0,0,0xB,2,3,4
,' ',0,30,40,0,2,5,11,5,11,5,11,100,0,0,3,'W',0,0 },
{ "Long Battle Bow",'N',0,"Weapon Bow",0x0490,0x0,0,'N',0x2062626C,0,IDB_ITEM_LONGBATTLEBOW,1109,128,1024,0xF,2,4,4,' ',
0,40,50,0,2,3,18,3,18,3,18,100,0,0,3,'W',0,0 },
{ "Vidala's Barb",'S',"Long Battle Bow","Weapon Bow",0x0490,0x0A,0,'S',0x2062626C,0,IDB_ITEM_VIDALASBARB,20016,0,0,0xF,2
,4,4,' ',0,40,50,0,2,3,18,3,18,3,18,100,0,0,3,'W',0,0 },
{ "Wizendraw",'U',"Long Battle Bow","Weapon Bow",0x0490,0x80,0,'U',0x2062626C,0,IDB_ITEM_WIZENDRAW,1207,0,0,0xF,2,4,4,'
',0,40,50,0,2,3,18,3,18,3,18,100,0,0,3,'W',0,0 },
{ "Short War Bow",'N',0,"Weapon Bow",0x04A0,0x0,0,'N',0x20627773,0,IDB_ITEM_SHORTWARBOW,1109,128,1024,0xF,2,3,4,' ',0,35
,55,0,2,6,14,6,14,6,14,100,0,0,3,'W',0,0 },
{ "Arctic Horn",'S',"Short War Bow","Weapon Bow",0x04A0,0x1C,0,'S',0x20627773,0,IDB_ITEM_ARCTICHORN,20003,0,0,0xF,2,3,4,
' ',0,35,55,0,2,6,14,6,14,6,14,100,0,0,3,'W',0,0 },
{ "Hellclap",'U',"Short War Bow","Weapon Bow",0x04A0,0x82,0,'U',0x20627773,0,IDB_ITEM_HELLCLAP,1207,0,0,0xF,2,3,4,' ',0,
35,55,0,2,6,14,6,14,6,14,100,0,0,3,'W',0,0 },
{ "Long War Bow",'N',0,"Weapon Bow",0x04B0,0x0,0,'N',0x2062776C,0,IDB_ITEM_LONGWARBOW,1109,128,1024,0xB,2,4,4,' ',0,50,6
```

```
5,0,2,3,23,3,23,3,23,100,0,0,3,'W',0,0 },
{ "Blastbark",'U',"Long War Bow","Weapon Bow",0x04B0,0x84,0,'U',0x2062776C,0,IDB_ITEM_BLASTBARK,1207,0,0,0xB,2,4,4,' ',0
,50,65,0,2,3,23,3,23,3,23,100,0,0,3,'W',0,0 },
{ "Light Crossbow",'N',0,"Weapon Crossbow",0x04C0,0x0,0,'N',0x2062786C,0,IDB_ITEM_LIGHTCROSSBOW,1110,128,1024,0xB,2,3,4,
' ',0,21,27,0,2,6,9,6,9,6,9,100,0,0,3,'W',0,0 },
{ "Leadcrow",'U',"Light Crossbow","Weapon Crossbow",0x04C0,0x86,0,'U',0x2062786C,0,IDB_ITEM_LEADCROW,1208,0,0,0xB,2,3,4,
' ',0,21,27,0,2,6,9,6,9,6,9,100,0,0,3,'W',0,0 },
{ "Crossbow",'N',0,"Weapon Crossbow",0x04D0,0x0,0,'N',0x2062786D,0,IDB_ITEM_CROSSBOW,1110,128,1024,0xB,2,3,4,' ',0,40,33
,0,2,9,14,9,14,9,14,100,0,0,3,'W',0,0 },
{ "Ichorsting",'U',"Crossbow","Weapon Crossbow",0x04D0,0x88,0,'U',0x2062786D,0,IDB_ITEM_ICHORSTING,1208,0,0,0xB,2,3,4,'
',0,40,33,0,2,9,14,9,14,9,14,100,0,0,3,'W',0,0 },
{ "Heavy Crossbow",'N',0,"Weapon Crossbow",0x04E0,0x0,0,'N',0x20627868,0,IDB_ITEM_HEAVYCROSSBOW,1110,128,1024,0xB,2,4,4,
' ',0,60,40,0,2,12,20,12,20,12,20,100,0,0,3,'W',0,0 },
{ "Hellcast",'U',"Heavy Crossbow","Weapon Crossbow",0x04E0,0x8A,0,'U',0x20627868,0,IDB_ITEM_HELLCAST,1208,0,0,0xB,2,4,4,
' ',0,60,40,0,2,12,20,12,20,12,20,100,0,0,3,'W',0,0 },
{ "Repeating Crossbow",'N',0,"Weapon Crossbow",0x04F0,0x0,0,'N',0x20627872,0,IDB_ITEM_REPEATINGCROSSBOW,1110,128,1024,0x
B,2,3,4,' ',0,40,50,0,2,6,12,6,12,6,12,100,0,0,3,'W',0,0 },
{ "Doomspittle",'U',"Repeating Crossbow","Weapon Crossbow",0x04F0,0x8C,0,'U',0x20627872,0,IDB_ITEM_DOOMSPITTLE,1208,0,0,
0xB,2,3,4,' ',0,40,50,0,2,6,12,6,12,6,12,100,0,0,3,'W',0,0 },
{ "Rancid Gas Potion",'N',0,"Potion Throwing",0x0500,0x0,0,'N',0x20737067,0,IDB_ITEM_RANCIDGAS,1400,0,0,0x1,1,1,-1,' ',0
,0,0,0,0,0,1,0,1,0,0,100,0,0,0,' ',10,0 },
{ "Oil Potion",'N',0,"Potion Throwing",0x0510,0x0,0,'N',0x2073706F,0,IDB_ITEM_OIL,1400,0,0,0x1,1,1,-1,' ',0,0,0,0,0,0,1,
0,1,0,0,100,0,0,0,' ',10,0 },
{ "Choking Gas Potion",'N',0,"Potion Throwing",0x0520,0x0,0,'N',0x206D7067,0,IDB_ITEM_CHOKINGGAS,1400,0,0,0x1,1,1,-1,' '
,0,0,0,0,0,0,1,0,1,0,0,100,0,0,0,' ',10,0 },
{ "Exploding Potion",'N',0,"Potion Throwing",0x0530,0x0,0,'N',0x206D706F,0,IDB_ITEM_EXPLODING,1400,0,0,0x1,1,1,-1,' ',0,
0,0,0,0,0,1,0,1,0,0,100,0,0,0,' ',10,0 },
{ "Strangling Gas Potion",'N',0,"Potion Throwing",0x0540,0x0,0,'N',0x206C7067,0,IDB_ITEM_STRANGLINGGAS,1400,0,0,0x1,1,1,
-1,' ',0,0,0,0,0,0,1,0,1,0,0,100,0,0,0,' ',10,0 },
{ "Fulminating Potion",'N',0,"Potion Throwing",0x0550,0x0,0,'N',0x206C706F,0,IDB_ITEM_FULM,1400,0,0,0x1,1,1,-1,' ',0,0,0
,0,0,0,1,0,1,0,0,100,0,0,0,' ',10,0 },
{ "Decoy Gidbinn",'Q',0,"Special Weapon",0x0560,0x0,0,'N',0x20333364,0,IDB_ITEM_DECOYGIDBINN,30003,2,8,0x3,1,2,4,' ',0,2
0,15,0,1,1,2,1,2,0,0,100,0,0,0,' ',0,0 },
{ "The Gidbinn",'Q',0,"Special Weapon",0x0570,0x0,0,'N',0x20333367,0,IDB_ITEM_THEGIDBINN,30003,2,8,0x3,1,2,4,' ',0,25,15
,0,1,3,7,3,7,0,0,100,0,0,0,' ',0,0 },
{ "Wirt's Leg",'Q',0,"Special Weapon",0x0580,0x0,0,'N',0x2067656C,0,IDB_ITEM_WIRTSLEG,30001,2,32,0x3,1,3,4,' ',0,0,0,0,1
,2,8,2,8,0,0,150,0,0,2,' ',0,0 },
{ "Horadric Malus",'Q',0,"Special Weapon",0x0590,0x0,0,'N',0x206D6468,0,IDB_ITEM_HORADRICMALUS,30001,2,8,0x3,1,2,4,' ',0
,15,15,0,1,6,15,6,15,0,0,150,0,0,0,' ',0,0 },
{ "Hellforge Hammer",'U',0,"Special Weapon",0x05A0,0xFC,0,'U',0x20686668,0,IDB_ITEM_HELLFORGEHAMMER,30004,2,16,0x8,2,3,4
,' ',0,0,0,0,0,1,6,15,6,15,0,0,150,0,0,0,' ',0,0 },
{ "Horadric Staff",'U',0,"Special Weapon",0x05B0,0xFA,0,'U',0x20747368,0,IDB_ITEM_HORADRICSTAFF,30002,64,512,0x8,1,4,4,'
 ',250,40,30,0,2,12,20,12,20,0,0,100,0,0,0,' ',0,0 },
{ "Shaft of the Horadric Staff",'U',0,"Special Weapon",0x05C0,0xF8,0,'U',0x2066736D,0,IDB_ITEM_SHAFTOFHORADRICSTAFF,3000
2,64,512,0x8,1,3,4,' ',0,35,25,0,2,10,15,10,15,0,0,100,0,0,0,' ',0,0 },
{ "Hatchet",'E',"Hand Axe","Weapon Axe",0x05D0,0x0,0,'N',0x20616839,0,IDB_ITEM_HANDAXE,1302,2,16,0x3,1,3,4,' ',28,25,25,
0,1,10,21,10,22,0,0,100,0,0,2,'W',0,0 },
{ "Cleaver",'E',"Axe","Weapon Axe",0x05E0,0x0,0,'N',0x20786139,0,IDB_ITEM_AXE,1302,2,16,0x3,2,3,4,' ',24,68,0,0,1,10,29,
10,30,0,0,100,0,0,3,'W',0,0 },
{ "Twin Axe",'E',"Double Axe","Weapon Axe",0x05F0,0x0,0,'N',0x20613239,0,IDB_ITEM_DOUBLEAXE,1302,2,16,0x3,2,3,4,' ',24,8
5,0,0,1,13,30,13,32,0,0,100,0,0,3,'W',0,0 },
{ "Crowbill",'E',"Military Pick","Weapon Axe",0x0600,0x0,0,'N',0x20706D39,0,IDB_ITEM_MILITARYPICK,1302,2,16,0x3,2,3,4,'
',26,94,70,0,1,14,27,14,29,0,0,100,0,0,3,'W',0,0 },
{ "Naga",'E',"War Axe","Weapon Axe",0x0610,0x0,0,'N',0x20617739,0,IDB_ITEM_WARAXE,1302,2,16,0x3,2,3,4,' ',26,121,0,0,1,1
8,34,18,35,0,0,100,0,0,3,'W',0,0 },
{ "Military Axe",'E',"Large Axe","Weapon Axe",0x0620,0x0,0,'N',0x20616C39,0,IDB_ITEM_LARGEAXE,1302,2,16,0x3,2,3,4,' ',30
,73,0,0,2,14,32,14,34,0,0,100,0,0,3,'W',0,0 },
{ "Bearded Axe",'E',"Broad Axe","Weapon Axe",0x0630,0x0,0,'N',0x20616239,0,IDB_ITEM_BROADAXE,1302,2,16,0x3,2,3,4,' ',35,
92,0,0,2,19,38,21,42,0,0,100,0,0,3,'W',0,0 },
{ "Tabar",'E',"Battle Axe","Weapon Axe",0x0640,0x0,0,'N',0x20746239,0,IDB_ITEM_BATTLEAXE,1302,2,16,0x3,2,3,4,' ',40,101,
0,0,2,21,56,24,58,0,0,100,0,0,3,'W',0,0 },
{ "Gothic Axe",'E',"Great Axe","Weapon Axe",0x0650,0x0,0,'N',0x20616739,0,IDB_ITEM_GREATAXE,1302,2,16,0x3,2,4,4,' ',50,1
15,79,0,2,14,50,18,54,0,0,100,0,0,3,'W',0,0 },
{ "Ancient Axe",'E',"Giant Axe","Weapon Axe",0x0660,0x0,0,'N',0x20696739,0,IDB_ITEM_GIANTAXE,1302,2,16,0x3,2,3,4,' ',50,
125,0,0,2,46,72,46,74,0,0,100,0,0,3,'W',0,0 },
{ "Burnt Wand",'E',"Wand","Weapon Wand",0x0670,0x0,0,'N',0x206E7739,0,IDB_ITEM_WAND,1311,32,256,0x3,1,2,4,'N',15,25,0,0,
1,8,18,8,19,0,0,100,0,0,0,' ',0,0 },
{ "Petrified Wand",'E',"Yew Wand","Weapon Wand",0x0680,0x0,0,'N',0x20777939,0,IDB_ITEM_YEWWAND,1311,32,256,0x3,1,2,4,'N'
,15,25,0,0,1,8,24,8,26,0,0,100,0,0,0,' ',0,0 },
{ "Tomb Wand",'E',"Bone Wand","Weapon Wand",0x0690,0x0,0,'N',0x20776239,0,IDB_ITEM_BONEWAND,1311,32,256,0x3,1,2,4,'N',15
,25,0,0,1,10,22,10,24,0,0,100,0,0,0,' ',0,0 },
{ "Grave Wand",'E',"Grim Wand","Weapon Wand",0x06A0,0x0,0,'N',0x20776739,0,IDB_ITEM_GRIMWAND,1311,32,256,0x3,1,2,4,'N',1
5,25,0,0,1,13,29,13,30,0,0,100,0,0,0,' ',0,0 },
{ "Cudgel",'E',"Club","Weapon Mace",0x06B0,0x0,0,'N',0x206C6339,0,IDB_ITEM_CLUB,1303,2,32,0x3,1,3,4,' ',24,25,0,0,1,6,21
,6,22,0,0,150,0,0,2,'W',0,0 },
{ "Rune Scepter",'E',"Scepter","Weapon Scepter",0x06C0,0x0,0,'N',0x20637339,0,IDB_ITEM_SCEPTER,1312,16,128,0x3,1,3,4,'P'
,50,58,0,0,1,13,24,13,26,0,0,150,0,0,2,'W',0,0 },
{ "Holy Water Sprinkler",'E',"Grand Scepter","Weapon Scepter",0x06D0,0x0,0,'N',0x20737139,0,IDB_ITEM_GRANDSCEPTER,1312,1
6,128,0x3,1,3,4,'P',60,76,25,0,1,14,29,14,30,0,0,150,0,0,2,'W',0,0 },
```

```
{ "Divine Scepter",'E',"War Scepter","Weapon Scepter",0x06E0,0x0,0,'N',0x20737739,0,IDB_ITEM_WARSCEPTER,1312,16,128,0x3,
2,3,4,'P',70,103,0,0,1,16,34,10,24,0,0,150,0,0,2,'W',0,0 },
{ "Barbed Club",'E',"Spiked Club","Weapon Mace",0x06F0,0x0,0,'N',0x20707339,0,IDB_ITEM_SPIKEDCLUB,1303,2,32,0x3,1,3,4,'
',36,20,0,0,1,13,21,13,22,0,0,150,0,0,2,'W',0,0 },
{ "Flanged Mace",'E',"Mace","Weapon Mace",0x0700,0x0,0,'N',0x20616D39,0,IDB_ITEM_MACE,1303,2,32,0x3,1,3,4,' ',60,61,0,0,
1,10,27,10,29,0,0,150,0,0,2,'W',0,0 },
{ "Jagged Star",'E',"Morning Star","Weapon Mace",0x0710,0x0,0,'N',0x20746D39,0,IDB_ITEM_MORNINGSTAR,1303,2,32,0x3,1,3,4,
' ',72,74,0,0,1,13,30,13,32,0,0,150,0,0,2,'W',0,0 },
{ "Knout",'E',"Flail","Weapon Mace",0x0720,0x0,0,'N',0x206C6639,0,IDB_ITEM_FLAIL,1303,2,32,0x3,2,3,4,' ',30,82,73,0,1,6,
35,6,37,0,0,150,0,0,3,'W',0,0 },
{ "Battle Hammer",'E',"War Hammer","Weapon Mace",0x0730,0x0,0,'N',0x20687739,0,IDB_ITEM_WARHAMMER,1303,2,32,0x3,2,3,4,'
',55,100,0,0,1,22,43,22,45,0,0,150,0,0,3,'W',0,0 },
{ "War Club",'E',"Maul","Weapon Mace",0x0740,0x0,0,'N',0x20396D39,0,IDB_ITEM_MAUL,1303,2,32,0x3,2,4,4,' ',60,124,25,0,2,
6,13,53,77,0,0,150,0,0,3,'W',0,0 },
{ "Martel De Fer",'E',"Great Maul","Weapon Mace",0x0750,0x0,0,'N',0x206D6739,0,IDB_ITEM_GREATMAUL,1303,2,32,0x3,2,3,4,'
',60,169,0,0,2,6,13,61,101,0,0,150,0,0,3,'W',0,0 },
{ "Gladius",'E',"Short Sword","Weapon Sword",0x0760,0x0,0,'N',0x20737339,0,IDB_ITEM_SHORTSWORD,1301,2,8,0x3,1,3,4,' ',24
,25,0,0,1,8,22,8,24,0,0,100,0,0,2,'W',0,0 },
{ "Cutlass",'E',"Scimitar","Weapon Sword",0x0770,0x0,0,'N',0x206D7339,0,IDB_ITEM_SCIMITAR,1301,2,8,0x3,1,3,4,' ',22,25,5
2,0,1,8,21,8,22,0,0,100,0,0,2,'W',0,0 },
{ "Shamshir",'E',"Sabre","Weapon Sword",0x0780,0x0,0,'N',0x20627339,0,IDB_ITEM_SABRE,1301,2,8,0x3,1,3,4,' ',32,58,58,0,1
,10,24,10,26,0,0,100,0,0,2,'W',0,0 },
{ "Tulwar",'E',"Falchion","Weapon Sword",0x0790,0x0,0,'N',0x20636639,0,IDB_ITEM_FALCHION,1301,2,8,0x3,1,3,4,' ',32,70,0,
0,1,16,35,16,37,0,0,100,0,0,2,'W',0,0 },
{ "Dimensional Blade",'E',"Crystal Sword","Weapon Sword",0x07A0,0x0,0,'N',0x20726339,0,IDB_ITEM_CRYSTALSWORD,1301,2,8,0x
3,2,3,4,' ',10,85,0,0,1,13,35,13,37,0,0,100,0,0,3,'W',0,0 },
{ "Battle Sword",'E',"Broad Sword","Weapon Sword",0x07B0,0x0,0,'N',0x20736239,0,IDB_ITEM_BROADSWORD,1301,2,8,0x3,2,3,4,'
',32,92,0,0,1,16,34,16,35,0,0,100,0,0,3,'W',0,0 },
{ "Rune Sword",'E',"Long Sword","Weapon Sword",0x07C0,0x0,0,'N',0x20736C39,0,IDB_ITEM_LONGSWORD,1301,2,8,0x3,2,3,4,' ',4
4,103,79,0,1,10,42,10,43,0,0,100,0,0,3,'W',0,0 },
{ "Ancient Sword",'E',"War Sword","Weapon Sword",0x07D0,0x0,0,'N',0x20647739,0,IDB_ITEM_WARSWORD,1301,2,8,0x3,1,3,4,' ',
44,127,88,0,1,18,43,18,45,0,0,100,0,0,2,'W',0,0 },
{ "Espandon",'E',"Two Handed Sword","Weapon Sword",0x07E0,0x0,0,'N',0x20683239,0,IDB_ITEM_2HSWORD,1301,2,8,0x3,1,4,4,' '
,44,73,61,0,2,8,26,18,40,0,0,100,0,0,3,'W',0,0 },
{ "Dacian Falx",'E',"Claymore","Weapon Sword",0x07F0,0x0,0,'N',0x206D6339,0,IDB_ITEM_CLAYMORE,1301,2,8,0x3,1,4,4,' ',50,
91,20,0,2,13,30,26,61,0,0,100,0,0,3,'W',0,0 },
{ "Tusk Sword",'E',"Giant Sword","Weapon Sword",0x0800,0x0,0,'N',0x20736739,0,IDB_ITEM_GIANTSWORD,1301,2,8,0x3,1,4,4,' '
,50,104,71,0,2,10,37,19,58,0,0,100,0,0,3,'W',0,0 },
{ "Gothic Sword",'E',"Bastard Sword","Weapon Sword",0x0810,0x0,0,'N',0x20396239,0,IDB_ITEM_BASTARDSWORD,1301,2,8,0x3,1,4
,4,' ',40,113,20,0,2,14,40,37,58,0,0,100,0,0,3,'W',0,0 },
{ "Zweihander",'E',"Flamberge","Weapon Sword",0x0820,0x0,0,'N',0x20626639,0,IDB_ITEM_FLAMBERGE,1301,2,8,0x3,2,4,4,' ',50
,125,94,0,2,19,35,26,54,0,0,100,0,0,3,'W',0,0 },
{ "Executioner Sword",'E',"Great Sword","Weapon Sword",0x0830,0x0,0,'N',0x20646739,0,IDB_ITEM_GREATSWORD,1301,2,8,0x3,2,
4,4,' ',50,170,110,0,2,24,40,45,80,0,0,100,0,0,3,'W',0,0 },
{ "Poignard",'E',"Dagger","Weapon Dagger",0x0840,0x0,0,'N',0x20676439,0,IDB_ITEM_DAGGER,1307,2,8,0x3,1,2,4,' ',16,25,0,0
,1,6,18,6,19,0,0,100,0,0,1,'W',0,0 },
{ "Rondel",'E',"Dirk","Weapon Dagger",0x0850,0x0,0,'N',0x20696439,0,IDB_ITEM_DIRK,1307,2,8,0x3,1,2,4,' ',20,25,58,0,1,10
,22,10,24,0,0,100,0,0,1,'W',0,0 },
{ "Ciquedeas",'E',"Kris","Weapon Dagger",0x0860,0x0,0,'N',0x20726B39,0,IDB_ITEM_KRIS,1307,2,8,0x3,1,3,4,' ',24,25,88,0,1
,8,26,8,27,0,0,100,0,0,2,'W',0,0 },
{ "Stiletto",'E',"Blade","Weapon Dagger",0x0870,0x0,0,'N',0x206C6239,0,IDB_ITEM_BLADE,1307,2,8,0x3,1,3,4,' ',24,73,97,0,
1,11,30,11,32,0,0,100,0,0,1,'W',0,0 },
{ "Battle Dart",'E',"Throwing Knife","Weapon Throwing",0x0880,0x0,0,'N',0x206B7439,0,IDB_ITEM_THROWINGKNIFE,1308,0,0,0x1
,1,2,4,' ',0,25,52,0,1,8,16,8,18,11,24,100,0,0,0,' ',75,0 },
{ "Francisca",'E',"Throwing Axe","Weapon Throwing",0x0890,0x0,0,'N',0x20617439,0,IDB_ITEM_THROWINGAXE,1308,0,0,0x1,1,2,4
,' ',0,25,97,0,1,11,22,11,24,18,29,100,0,0,0,' ',32,0 },
{ "War Dart",'E',"Balanced Knife","Weapon Throwing",0x08A0,0x0,0,'N',0x206B6239,0,IDB_ITEM_BALANCEDKNIFE,1308,0,0,0x1,1,
2,4,' ',0,25,80,0,1,6,24,13,26,14,27,100,0,0,0,' ',60,0 },
{ "Hurlbat",'E',"Balanced Axe","Weapon Throwing",0x08B0,0x0,0,'N',0x20386239,0,IDB_ITEM_BALANCEDAXE,1308,0,0,0x1,2,3,4,'
',0,25,106,0,1,13,27,13,29,24,34,100,0,0,0,' ',24,0 },
{ "War Javelin",'E',"Javelin","Weapon Javelin",0x08C0,0x0,0,'N',0x20616A39,0,IDB_ITEM_JAVELIN,1304,0,0,0x1,1,3,4,'A',0,2
5,25,0,1,6,19,6,21,14,32,100,0,0,0,' ',60,0 },
{ "Great Pilum",'E',"Pilum","Weapon Javelin",0x08D0,0x0,0,'N',0x20697039,0,IDB_ITEM_PILUM,1304,0,0,0x1,1,3,4,'A',0,25,88
,0,1,11,26,11,27,16,42,100,0,0,0,' ',50,0 },
{ "Simbilan",'E',"Short Spear","Weapon Javelin",0x08E0,0x0,0,'N',0x20397339,0,IDB_ITEM_SHORTSPEAR,1304,0,0,0x1,1,3,4,'A'
,0,80,80,0,1,8,32,8,34,27,50,100,0,0,0,' ',40,0 },
{ "Spiculum",'E',"Glaive","Weapon Javelin",0x08F0,0x0,0,'N',0x206C6739,0,IDB_ITEM_GLAIVE,1304,0,0,0x1,1,4,4,'A',0,93,73,
0,1,13,38,13,40,8,26,100,0,0,0,' ',20,0 },
{ "Harpoon",'E',"Throwing Spear","Weapon Javelin",0x0900,0x0,0,'N',0x20737439,0,IDB_ITEM_THROWINGSPEAR,1304,0,0,0x1,1,4,
4,'A',0,25,118,0,1,13,35,13,37,18,54,100,0,0,0,' ',80,0 },
{ "War Spear",'E',"Spear","Weapon Spear",0x0910,0x0,0,'N',0x20727339,0,IDB_ITEM_SPEAR,1305,2,64,0x1,2,4,4,' ',30,0,0,0,2
,10,35,10,37,0,0,100,0,0,3,'W',0,0 },
{ "Fuscina",'E',"Trident","Weapon Spear",0x0920,0x0,0,'N',0x20727439,0,IDB_ITEM_TRIDENT,1305,2,64,0x3,2,4,4,' ',35,77,0,
0,2,19,35,19,37,0,0,100,0,0,3,'W',0,0 },
{ "War Fork",'E',"Brandistock","Weapon Spear",0x0930,0x0,0,'N',0x20726239,0,IDB_ITEM_BRANDISTOCK,1305,2,64,0x3,2,4,4,' '
,28,80,95,0,2,16,38,16,40,0,0,100,0,0,3,'W',0,0 },
{ "Yari",'E',"Spetum","Weapon Spear",0x0940,0x0,0,'N',0x20747339,0,IDB_ITEM_SPETUM,1305,2,64,0x3,2,4,4,' ',28,0,28,0,2,2
9,45,29,46,0,0,100,0,0,3,'W',0,0 },
{ "Lance",'E',"Pike","Weapon Spear",0x0950,0x0,0,'N',0x20397039,0,IDB_ITEM_PIKE,1305,2,64,0x3,2,4,4,' ',25,110,88,0,2,27
```

```
,112,27,114,0,0,100,0,0,3,'W',0,0 },
{ "Lochaber Axe",'E',"Bardiche","Weapon Polearm",0x0960,0x0,0,'N',0x20376239,0,IDB_ITEM_BARDICHE,1306,2,64,0x3,2,4,4,' '
,50,80,0,0,2,6,51,6,56,0,0,100,0,0,3,'W',0,0 },
{ "Bill",'E',"Voulge","Weapon Polearm",0x0970,0x0,0,'N',0x206F7639,0,IDB_ITEM_VOULGE,1306,2,64,0x3,2,4,4,' ',50,95,0,0,2
,14,43,14,45,0,0,100,0,0,3,'W',0,0 },
{ "Battle Scythe",'E',"Scythe","Weapon Polearm",0x0980,0x0,0,'N',0x20387339,0,IDB_ITEM_SCYTHE,1306,2,64,0x3,2,4,4,' ',65
,82,82,0,2,18,43,18,45,0,0,100,0,0,3,'W',0,0 },
{ "Partizan",'E',"Pole Axe","Weapon Polearm",0x0990,0x0,0,'N',0x20617039,0,IDB_ITEM_POLEAXE,1306,2,64,0x3,2,4,4,' ',65,1
13,0,0,2,34,59,34,61,0,0,100,0,0,3,'W',0,0 },
{ "Bec-De-Corbin",'E',"Halberd","Weapon Polearm",0x09A0,0x0,0,'N',0x20396839,0,IDB_ITEM_HALBERD,1306,2,64,0x3,2,4,4,' ',
55,133,91,0,2,24,67,24,77,0,0,100,0,0,3,'W',0,0 },
{ "Grim Scythe",'E',"War Scythe","Weapon Polearm",0x09B0,0x0,0,'N',0x20637739,0,IDB_ITEM_WARSCYTHE,1306,2,64,0x3,2,4,4,'
 ',55,140,140,0,2,29,62,29,64,0,0,100,0,0,3,'W',0,0 },
{ "Jo Staff",'E',"Short Staff","Weapon Staff",0x09C0,0x0,0,'N',0x20737338,0,IDB_ITEM_SHORTSTAFF,1313,64,512,0x3,1,3,4,'S
',20,0,0,0,2,6,19,6,21,0,0,100,0,0,2,'W',0,0 },
{ "Quarterstaff",'E',"Long Staff","Weapon Staff",0x09D0,0x0,0,'N',0x20736C38,0,IDB_ITEM_LONGSTAFF,1313,64,512,0x3,1,4,4,
'S',30,0,0,0,2,8,24,8,26,0,0,100,0,0,3,'W',0,0 },
{ "Cedar Staff",'E',"Gnarled Staff","Weapon Staff",0x09E0,0x0,0,'N',0x20736338,0,IDB_ITEM_GNARLEDSTAFF,1313,64,512,0x3,1
,4,4,'S',35,0,0,0,2,11,30,11,32,0,0,100,0,0,3,'W',0,0 },
{ "Gothic Staff",'E',"Battle Staff","Weapon Staff",0x09F0,0x0,0,'N',0x20736238,0,IDB_ITEM_BATTLESTAFF,1313,64,512,0x3,1,
4,4,'S',40,0,0,0,2,14,32,14,34,0,0,100,0,0,3,'W',0,0 },
{ "Rune Staff",'E',"War Staff","Weapon Staff",0x0A00,0x0,0,'N',0x20737738,0,IDB_ITEM_WARSTAFF,1313,64,512,0x3,2,4,4,'S',
50,25,0,0,2,24,56,24,58,0,0,100,0,0,3,'W',0,0 },
{ "Edge Bow",'E',"Short Bow","Weapon Bow",0x0A10,0x0,0,'N',0x20627338,0,IDB_ITEM_SHORTBOW,1309,128,1024,0x3,2,3,4,' ',0,
25,43,0,2,6,18,6,19,6,19,100,0,0,3,'W',0,0 },
{ "Razor Bow",'E',"Hunter's Bow","Weapon Bow",0x0A20,0x0,0,'N',0x20626838,0,IDB_ITEM_HUNTERSBOW,1309,128,1024,0x3,2,3,4,
' ',0,0,62,0,2,8,22,8,22,8,21,100,0,0,3,'W',0,0 },
{ "Cedar Bow",'E',"Long Bow","Weapon Bow",0x0A30,0x0,0,'N',0x20626C38,0,IDB_ITEM_LONGBOW,1309,128,1024,0x3,2,4,4,' ',0,5
3,49,0,2,10,27,10,29,9,26,100,0,0,3,'W',0,0 },
{ "Double Bow",'E',"Composite Bow","Weapon Bow",0x0A40,0x0,0,'N',0x20626338,0,IDB_ITEM_COMPOSITEBOW,1309,128,1024,0x3,2,
3,4,' ',0,58,73,0,2,11,24,11,26,11,21,100,0,0,3,'W',0,0 },
{ "Short Siege Bow",'E',"Short Battle Bow","Weapon Bow",0x0A50,0x0,0,'N',0x20387338,0,IDB_ITEM_SHORTBATTLEBOW,1309,128,1
024,0x3,2,3,4,' ',0,65,80,0,2,13,29,13,30,12,26,100,0,0,3,'W',0,0 },
{ "Long Siege Bow",'E',"Long Battle Bow","Weapon Bow",0x0A60,0x0,0,'N',0x20386C38,0,IDB_ITEM_LONGBATTLEBOW,1309,128,1024
,0x3,2,4,4,' ',0,80,95,0,2,10,40,10,42,9,36,100,0,0,3,'W',0,0 },
{ "Rune Bow",'E',"Short War Bow","Weapon Bow",0x0A70,0x0,0,'N',0x20777338,0,IDB_ITEM_SHORTWARBOW,1309,128,1024,0x3,2,3,4
,' ',0,73,103,0,2,14,34,14,35,14,30,100,0,0,3,'W',0,0 },
{ "Gothic Bow",'E',"Long War Bow","Weapon Bow",0x0A80,0x0,0,'N',0x20776C38,0,IDB_ITEM_LONGWARBOW,1309,128,1024,0x3,2,4,4
,' ',0,95,118,0,2,10,48,10,50,9,44,100,0,0,3,'W',0,0 },
{ "Arbalest",'E',"Repeating Crossbow","Weapon Crossbow",0x0A90,0x0,0,'N',0x20786C38,0,IDB_ITEM_LIGHTCROSSBOW,1310,128,10
24,0x3,2,3,4,' ',0,80,95,0,2,14,26,14,27,14,23,100,0,0,3,'W',0,0 },
{ "Siege Crossbow",'E',"Light Crossbow","Weapon Crossbow",0x0AA0,0x0,0,'N',0x20786D38,0,IDB_ITEM_CROSSBOW,1310,128,1024,
0x3,2,3,4,' ',0,52,61,0,2,19,34,19,35,18,30,100,0,0,3,'W',0,0 },
{ "Ballista",'E',"Crossbow","Weapon Crossbow",0x0AB0,0x0,0,'N',0x20786838,0,IDB_ITEM_HEAVYCROSSBOW,1310,128,1024,0x3,2,4
,4,' ',0,80,70,0,2,24,43,24,45,23,39,100,0,0,3,'W',0,0 },
{ "Chu-Ko-Nu",'E',"Heavy Crossbow","Weapon Crossbow",0x0AC0,0x0,0,'N',0x20787238,0,IDB_ITEM_REPEATINGCROSSBOW,1310,128,1
024,0x3,2,3,4,' ',0,110,80,0,2,14,30,14,32,14,27,100,0,0,3,'W',0,0 },
{ "Khalim's Flail",'Q',"KhalimFlail","Special Weapon",0x0AD0,0x0,0,'N',0x20316671,0,IDB_ITEM_KHALIMSFLAIL,30003,2,32,0x3
,2,3,4,' ',0,35,25,0,2,1,15,1,15,0,0,150,0,0,0,' ',0,0 },
{ "Khalim's Will",'Q',"SuperKhalimFlail","Special Weapon",0x0AE0,0x0,0,'N',0x20326671,0,IDB_ITEM_KHALIMSWILL,30003,2,32,
0x3,2,3,4,' ',0,0,0,0,2,1,15,1,15,0,0,150,0,0,0,' ',0,0 },
{ "Cap",'N',0,"Helm",0x0AF0,0x0,0,'N',0x20706163,0,IDB_ITEM_CAP,2100,1,2,0xF,2,2,1,' ',12,0,0,0,0,0,0,0,0,0,0,3,5,2,'H
',0,0 },
{ "Infernal Cranium",'S',"Cap","Helm",0x0AF0,0x14,0,'S',0x20706163,0,IDB_ITEM_INFERNALCRANIUM,20010,0,0,0xF,2,2,1,' ',24
,0,0,0,0,0,0,0,0,0,0,0,3,5,2,'H',0,0 },
{ "Biggin's Bonnet",'U',"Cap","Helm",0x0AF0,0x8E,0,'U',0x20706163,0,IDB_ITEM_WARBONET,2200,0,0,0xF,2,2,1,' ',60,0,0,0,0,
0,0,0,0,0,0,0,0,3,5,2,'H',0,0 },
{ "Skull Cap",'N',0,"Helm",0x0B00,0x0,0,'N',0x20706B73,0,IDB_ITEM_SKULLCAP,2100,1,2,0xF,2,2,1,' ',18,15,0,0,0,0,0,0,0,0,
0,0,8,11,2,'H',0,0 },
{ "Arcanna's Head",'S',"Skull Cap","Helm",0x0B00,0x1E,0,'S',0x20706B73,0,IDB_ITEM_ARCANNASHEAD,20002,0,0,0xF,2,2,1,' ',3
6,35,0,0,0,0,0,0,0,0,0,0,0,8,11,2,'H',0,0 },
{ "Tarnhelm",'U',"Skull Cap","Helm",0x0B00,0x90,0,'U',0x20706B73,0,IDB_ITEM_TARNHELM,2200,0,0,0xF,2,2,1,' ',90,15,0,0,0,
0,0,0,0,0,0,0,0,8,11,2,'H',0,0 },
{ "Helm",'N',0,"Helm",0x0B10,0x0,0,'N',0x206D6C68,0,IDB_ITEM_HELM,2100,1,2,0xF,2,2,1,' ',24,26,0,0,0,0,0,0,0,0,0,0,15,18
,2,'H',0,0 },
{ "Berserker's Headgear",'S',"Helm","Helm",0x0B10,0x16,0,'S',0x206D6C68,0,IDB_ITEM_BERSERKERSHEADGEAR,20004,0,0,0xF,2,2,
1,' ',48,26,0,0,0,0,0,0,0,0,0,0,15,18,2,'H',0,0 },
{ "Coif of Glory",'U',"Helm","Helm",0x0B10,0x92,0,'U',0x206D6C68,0,IDB_ITEM_COIFOFGLORY,2200,0,0,0xF,2,2,1,' ',120,26,0,
0,0,0,0,0,0,0,0,15,18,2,'H',0,0 },
{ "Full Helm",'N',0,"Helm",0x0B20,0x0,0,'N',0x206C6866,0,IDB_ITEM_FULLHELM,2100,1,2,0xF,2,2,1,' ',30,41,0,0,0,0,0,0,0,0,
0,0,23,26,2,'H',0,0 },
{ "Isenhart's Horns",'S',"Full Helm","Helm",0x0B20,0x08,0,'S',0x206C6866,0,IDB_ITEM_ISENHARTSHORNS,20012,0,0,0xF,2,2,1,'
 ',60,41,0,0,0,0,0,0,0,0,0,0,23,26,2,'H',0,0 },
{ "Duskdeep",'U',"Full Helm","Helm",0x0B20,0x94,0,'U',0x206C6866,0,IDB_ITEM_DUSKDEEP,2200,0,0,0xF,2,2,1,' ',150,41,0,0,0
,0,0,0,0,0,0,0,0,23,26,2,'H',0,0 },
{ "Great Helm",'N',0,"Helm",0x0B30,0x0,0,'N',0x206D6867,0,IDB_ITEM_GREATHELM,2100,1,2,0xF,2,2,1,' ',40,63,0,0,0,0,0,0,0,
0,0,0,30,35,2,'H',0,0 },
{ "Sigon's Visor",'S',"Great Helm","Helm",0x0B30,0x12,0,'S',0x206D6867,0,IDB_ITEM_SIGONSVISOR,20014,0,0,0xF,2,2,1,' ',80
,63,0,0,0,0,0,0,0,0,0,0,30,35,2,'H',0,0 },
```

```
{ "Howltusk",'U',"Great Helm","Helm",0x0B30,0x98,0,'U',0x206D6867,0,IDB_ITEM_HOWLTUSK,2200,0,0,0xF,2,2,1,' ',200,63,0,0,
0,0,0,0,0,0,0,0,30,35,2,'H',0,0 },
{ "Crown",'N',0,"Helm",0x0B40,0x0,0,'N',0x206E7263,0,IDB_ITEM_CROWN,2100,1,2,0xF,2,2,1,' ',50,55,0,0,0,0,0,0,0,0,0,25,
45,2,'H',0,0 },
{ "Iratha's Coil",'S',"Crown","Helm",0x0B40,0x06,0x0070,'S',0x206E7263,0,IDB_ITEM_IRATHASCOIL,20011,0,0,0xF,2,2,1,' ',10
0,55,0,0,0,0,0,0,0,0,0,0,25,45,2,'H',0,0 },
{ "Milabrega's Diadem",'S',"Crown","Helm",0x0B40,0x0C,0xFF8F,'S',0x206E7263,0,IDB_ITEM_MILABREGASDIADEM,20013,0,0,0xF,2,
2,1,' ',100,55,0,0,0,0,0,0,0,0,0,0,25,45,2,'H',0,0 },
{ "Undead Crown",'U',"Crown","Helm",0x0B40,0x9A,0,'U',0x206E7263,0,IDB_ITEM_UNDEADCROWN,2200,0,0,0xF,2,2,1,' ',250,55,0,
0,0,0,0,0,0,0,0,0,25,45,2,'H',0,0 },
{ "Mask",'N',0,"Helm",0x0B50,0x0,0,'N',0x206B736D,0,IDB_ITEM_MASK,2100,1,2,0xF,2,2,1,' ',20,23,0,0,0,0,0,0,0,0,0,9,27,
2,'H',0,0 },
{ "Cathan's Visage",'S',"Mask","Helm",0x0B50,0x0E,0,'S',0x206B736D,0,IDB_ITEM_MASK,20005,0,0,0xF,2,2,1,' ',60,23,0,0,0,0
,0,0,0,0,0,0,9,27,2,'H',0,0 },
{ "The Face of Horror",'U',"Mask","Helm",0x0B50,0x9C,0,'U',0x206B736D,0,IDB_ITEM_MASK,2200,0,0,0xF,2,2,1,' ',100,23,0,0,
0,0,0,0,0,0,0,0,9,27,2,'H',0,0 },
{ "Quilted Armor",'N',0,"Armor",0x0B60,0x0,0,'N',0x20697571,0,IDB_ITEM_QUILTED,3100,1,1,0xF,2,3,3,' ',20,12,0,0,0,0,0,0,
0,0,0,0,8,11,0,' ',0,0 },
{ "Arctic Furs",'S',"Quilted Armor","Armor",0x0B60,0x1C,0,'S',0x20697571,0,IDB_ITEM_ARCTICFURS,20003,0,0,0xF,2,3,3,' ',4
0,12,0,0,0,0,0,0,0,0,0,0,8,11,0,' ',0,0 },
{ "Greyform",'U',"Quilted Armor","Armor",0x0B60,0x9E,0,'U',0x20697571,0,IDB_ITEM_GREYFORM,3200,0,0,0xF,2,3,3,' ',100,12,
0,0,0,0,0,0,0,0,0,0,8,11,0,' ',0,0 },
{ "Leather Armor",'N',0,"Armor",0x0B70,0x0,0,'N',0x2061656C,0,IDB_ITEM_LEATHER,3100,1,1,0xF,2,3,3,' ',24,15,0,0,0,0,0,0,
0,0,0,0,14,17,0,' ',0,0 },
{ "Vidala's Ambush",'S',"Leather Armor","Armor",0x0B70,0x0A,0,'S',0x2061656C,0,IDB_ITEM_VIDALASAMBUSH,20016,0,0,0xF,2,3,
3,' ',48,15,0,0,0,0,0,0,0,0,0,0,14,17,0,' ',0,0 },
{ "Blinkbat's Form",'U',"Leather Armor","Armor",0x0B70,0xA0,0,'U',0x2061656C,0,IDB_ITEM_BLINKBATSFORM,3200,0,0,0xF,2,3,3
,' ',120,15,0,0,0,0,0,0,0,0,0,0,14,17,0,' ',0,0 },
{ "Hard Leather Armor",'N',0,"Armor",0x0B80,0x0,0,'N',0x20616C68,0,IDB_ITEM_HARDLEATHER,3100,1,1,0xF,2,3,3,' ',28,20,0,0
,0,0,0,0,0,0,0,0,21,24,0,' ',0,0 },
{ "The Centurion",'U',"Hard Leather Armor","Armor",0x0B80,0xA2,0,'U',0x20616C68,0,IDB_ITEM_THECENTURION,3200,0,0,0xB,2,3
,3,' ',100,20,0,0,0,0,0,0,0,0,0,0,21,24,0,' ',0,0 },
{ "Studded Leather",'N',0,"Armor",0x0B90,0x0,0,'N',0x20757473,0,IDB_ITEM_STUDDEDLEATHER,3100,1,1,0xB,2,3,3,' ',32,27,0,0
,0,0,0,0,0,0,0,0,32,35,0,' ',0,0 },
{ "Twitchthroe",'U',"Studded Leather","Armor",0x0B90,0xA4,0,'U',0x20757473,0,IDB_ITEM_TWITCHTHROE,3200,0,0,0xF,2,3,3,' '
,160,27,0,0,0,0,0,0,0,0,0,0,32,35,0,' ',0,0 },
{ "Ring Mail",'N',0,"Armor",0x0BA0,0x0,0,'N',0x20676E72,0,IDB_ITEM_RINGMAIL,3100,1,1,0xF,2,3,3,' ',26,27,0,0,0,0,0,0,0,0
,0,0,45,48,0,' ',0,0 },
{ "Angelical Mantle",'S',"Ring Mail","Armor",0x0BA0,0x1A,0,'S',0x20676E72,0,IDB_ITEM_CHAINMAIL,20001,0,0,0xF,2,3,3,' ',1
20,65,0,0,0,0,0,0,0,0,0,0,45,48,0,' ',0,0 },
{ "Darkglow",'U',"Ring Mail","Armor",0x0BA0,0xA6,0,'U',0x20676E72,0,IDB_ITEM_DARKGLOW,3200,0,0,0xF,2,3,3,' ',130,36,0,0,
0,0,0,0,0,0,0,0,45,48,0,' ',0,0 },
{ "Scale Mail",'N',0,"Armor",0x0BB0,0x0,0,'N',0x206C6373,0,IDB_ITEM_SCALEMAIL,3100,1,1,0xF,2,3,3,' ',36,36,0,0,0,0,0,0,0
,0,0,0,57,60,0,' ',0,0 },
{ "Hawkmail",'U',"Scale Mail","Armor",0x0BB0,0xA8,0,'U',0x206C6373,0,IDB_ITEM_HAWKMAIL,3200,0,0,0xF,2,3,3,' ',180,44,0,0
,0,0,0,0,0,0,0,0,57,60,0,' ',0,0 },
{ "Chain Mail",'N',0,"Armor",0x0BC0,0x0,0,'N',0x206E6863,0,IDB_ITEM_CHAINMAIL,3100,1,1,0xF,2,3,3,' ',45,48,0,0,0,0,0,0,0,0
,0,0,0,72,75,0,' ',0,0 },
{ "Cathan's Mesh",'S',"Chain Mail","Armor",0x0BC0,0x0E,0,'S',0x206E6863,0,IDB_ITEM_CATHANSMESH,20005,0,0,0xF,2,3,3,' ',9
0,24,0,0,0,0,0,0,0,0,0,0,72,75,0,' ',0,0 },
{ "Sparking Mail",'U',"Chain Mail","Armor",0x0BC0,0xAA,0,'U',0x206E6863,0,IDB_ITEM_SPARKLINGMAIL,3200,0,0,0xF,2,3,3,' ',
250,30,0,0,0,0,0,0,0,0,0,0,72,75,0,' ',0,0 },
{ "Breast Plate",'N',0,"Armor",0x0BD0,0x0,0,'N',0x20737262,0,IDB_ITEM_BREASTPLATE,3100,1,1,0xF,2,3,3,' ',50,30,0,0,0,0,0,0
,0,0,0,0,0,65,68,0,' ',0,0 },
{ "Isenhart's Case",'S',"Breast Plate","Armor",0x0BD0,0x08,0,'S',0x20737262,0,IDB_ITEM_ISENHARTSCASE,20012,0,0,0xF,2,3,3
,' ',100,30,0,0,0,0,0,0,0,0,0,0,65,68,0,' ',0,0 },
{ "Venomsward",'U',"Breast Plate","Armor",0x0BD0,0xAC,0,'U',0x20737262,0,IDB_ITEM_VENOMSWARD,3200,0,0,0xF,2,3,3,' ',225,
48,0,0,0,0,0,0,0,0,0,0,65,68,0,' ',0,0 },
{ "Splint Mail",'N',0,"Armor",0x0BE0,0x0,0,'N',0x206C7073,0,IDB_ITEM_SPLINTMAIL,3100,1,1,0xF,2,3,3,' ',30,51,0,0,0,0,0,0
,0,0,0,0,90,95,0,' ',0,0 },
{ "Berserker's Hauberk",'S',"Splint Mail","Armor",0x0BE0,0x16,0,'S',0x206C7073,0,IDB_ITEM_BERSERKERSHAUBERK,20004,0,0,0x
F,2,3,3,' ',60,51,0,0,0,0,0,0,0,0,0,0,90,95,0,' ',0,0 },
{ "Iceblink",'U',"Splint Mail","Armor",0x0BE0,0xAE,0,'U',0x206C7073,0,IDB_ITEM_ICEBLINK,3200,0,0,0xF,2,3,3,' ',150,51,0,
0,0,0,0,0,0,0,0,0,90,95,0,' ',0,0 },
{ "Plate Mail",'N',0,"Armor",0x0BF0,0x0,0,'N',0x20746C70,0,IDB_ITEM_PLATEMAIL,3100,1,1,0xB,2,3,3,' ',60,65,0,0,0,0,0,0,0
,0,0,0,108,116,0,' ',0,0 },
{ "Boneflesh",'U',"Plate Mail","Armor",0x0BF0,0xB0,0,'U',0x20746C70,0,IDB_ITEM_BONEFLESH,3200,0,0,0xB,2,3,3,' ',255,65,0
,0,0,0,0,0,0,0,0,0,108,116,0,' ',0,0 },
{ "Field Plate",'N',0,"Armor",0x0C00,0x0,0,'N',0x20646C66,0,IDB_ITEM_FIELDPLATE,3100,1,1,0xB,2,3,3,' ',48,55,0,0,0,0,0,0
,0,0,0,0,101,105,0,' ',0,0 },
{ "Rockfleece",'U',"Field Plate","Armor",0x0C00,0xB2,0,'U',0x20646C66,0,IDB_ITEM_ROCKFLEECE,3200,0,0,0xB,2,3,3,' ',240,5
0,0,0,0,0,0,0,0,0,0,0,101,105,0,' ',0,0 },
{ "Gothic Plate",'N',0,"Armor",0x0C10,0x0,0,'N',0x20687467,0,IDB_ITEM_GOTHICPLATE,3100,1,1,0xF,2,3,3,' ',55,70,0,0,0,0,0,0
,0,0,0,0,0,128,135,0,' ',0,0 },
{ "Sigon's Shelter",'S',"Gothic Plate","Armor",0x0C10,0x12,0,'S',0x20687467,0,IDB_ITEM_SIGONSSHELTER,20014,0,0,0xF,2,3,3
,' ',110,70,0,0,0,0,0,0,0,0,0,0,128,135,0,' ',0,0 },
{ "Rattlecage",'U',"Gothic Plate","Armor",0x0C10,0xB4,0,'U',0x20687467,0,IDB_ITEM_RATTLECAGE,3200,0,0,0xF,2,3,3,' ',255,
70,0,0,0,0,0,0,0,0,0,0,128,135,0,' ',0,0 },
{ "Full Plate",'N',0,"Armor",0x0C20,0x0,0,'N',0x206C7566,0,IDB_ITEM_FULLPLATE,3100,1,1,0xF,2,3,3,' ',70,80,0,0,0,0,0,0,0
```

```
,0,0,0,150,161,0,' ',0,0 },
{ "Tancred's Spine",'S',"Full Plate","Armor",0x0C20,0x10,0,'S',0x206C7566,0,IDB_ITEM_TANCREDSSPINE,20015,0,0,0xF,2,3,3,'
 ',140,80,0,0,0,0,0,0,0,0,0,0,150,161,0,' ',0,0 },
{ "Goldskin",'U',"Full Plate","Armor",0x0C20,0xB6,0,'U',0x206C7566,0,IDB_ITEM_GOLDSKIN,3200,0,0,0xF,2,3,3,' ',255,80,0,0
,0,0,0,0,0,0,0,0,150,161,0,' ',0,0 },
{ "Ancient Armor",'N',0,"Armor",0x0C30,0x0,0,'N',0x20726161,0,IDB_ITEM_ANCIENTARMOR,3100,1,1,0xF,2,3,3,' ',60,100,0,0,0,
0,0,0,0,0,0,0,218,233,0,' ',0,0 },
{ "Milabrega's Robe",'S',"Ancient Armor","Armor",0x0C30,0x0C,0,'S',0x20726161,0,IDB_ITEM_MILABREGASROBE,20013,0,0,0xF,2,
3,3,' ',120,100,0,0,0,0,0,0,0,0,0,0,218,233,0,' ',0,0 },
{ "Victor's Silk",'U',"Ancient Armor","Armor",0x0C30,0xB8,0,'U',0x20726161,0,IDB_ITEM_VICTORSSILK,3200,0,0,0xF,2,3,3,' '
,255,100,0,0,0,0,0,0,0,0,0,0,218,233,0,' ',0,0 },
{ "Light Plate",'N',0,"Armor",0x0C40,0x0,0,'N',0x2070746C,0,IDB_ITEM_LIGHTPLATE,3100,1,1,0xF,2,3,3,' ',60,41,0,0,0,0,0,0
,0,0,0,0,90,107,0,' ',0,0 },
{ "Arcanna's Flesh",'S',"Light Plate","Armor",0x0C40,0x1E,0,'S',0x2070746C,0,IDB_ITEM_ARCANNASFLESH,20002,0,0,0xF,2,3,3,
' ',120,41,0,0,0,0,0,0,0,0,0,0,90,107,0,' ',0,0 },
{ "Heavenly Garb",'U',"Light Plate","Armor",0x0C40,0xBA,0,'U',0x2070746C,0,IDB_ITEM_HEAVENLYGARB,3200,0,0,0xF,2,3,3,' ',
255,41,0,0,0,0,0,0,0,0,0,0,90,107,0,' ',0,0 },
{ "Buckler",'N',0,"Shield",0x0C50,0x0,0,'N',0x20637562,0,IDB_ITEM_BUCKLER,4100,4,4,0xF,2,2,5,' ',12,12,0,0,0,0,0,0,0,0,0
,0,4,6,1,'S',0,0 },
{ "Hsarus' Iron Fist",'S',"Buckler","Shield",0x0C50,0x02,0,'S',0x20637562,0,IDB_ITEM_HSARUSIRONFIST,20009,0,0,0xF,2,2,5,
' ',24,12,0,0,0,0,0,0,0,0,0,0,4,6,1,'S',0,0 },
{ "Pelta Lunata",'U',"Buckler","Shield",0x0C50,0xBC,0,'U',0x20637562,0,IDB_ITEM_PELTALUNATA,4200,0,0,0xF,2,2,5,' ',60,12
,0,0,0,0,0,0,0,0,0,0,4,6,1,'S',0,0 },
{ "Small Shield",'N',0,"Shield",0x0C60,0x0,0,'N',0x206C6D73,0,IDB_ITEM_SMALLSHIELD,4100,4,4,0xF,2,2,5,' ',16,22,0,0,0,0,
0,0,0,0,0,0,8,10,2,'S',0,0 },
{ "Cleglaw's Claw",'S',"Small Shield","Shield",0x0C60,0x04,0,'S',0x206C6D73,0,IDB_ITEM_CLEGLAWSCLAW,20007,0,0,0xF,2,2,5,
' ',32,22,0,0,0,0,0,0,0,0,0,0,8,10,2,'S',0,0 },
{ "Umbral Disk",'U',"Small Shield","Shield",0x0C60,0xBE,0,'U',0x206C6D73,0,IDB_ITEM_UMBRALDISK,4200,0,0,0xF,2,2,5,' ',80
,22,0,0,0,0,0,0,0,0,0,0,8,10,2,'S',0,0 },
{ "Large Shield",'N',0,"Shield",0x0C70,0x0,0,'N',0x2067726C,0,IDB_ITEM_LARGESHIELD,4100,4,4,0xF,2,3,5,' ',24,34,0,0,0,0,
0,0,0,0,0,0,12,14,3,'S',0,0 },
{ "Civerb's Ward",'S',"Large Shield","Shield",0x0C70,0x00,0,'S',0x2067726C,0,IDB_ITEM_LARGESHIELD,20006,0,0,0xF,2,3,5,'
',48,34,0,0,0,0,0,0,0,0,0,0,12,14,3,'S',0,0 },
{ "Stormguild",'U',"Large Shield","Shield",0x0C70,0xC0,0,'U',0x2067726C,0,IDB_ITEM_STORMGUILD,4200,0,0,0xF,2,3,5,' ',120
,34,0,0,0,0,0,0,0,0,0,0,12,14,3,'S',0,0 },
{ "Kite Shield",'N',0,"Shield",0x0C80,0x0,0,'N',0x2074696B,0,IDB_ITEM_KITESHIELD,4100,4,4,0xF,2,3,5,' ',30,47,0,0,0,0,0,
0,0,0,0,0,16,18,3,'S',0,0 },
{ "Milabrega's Orb",'S',"Kite Shield","Shield",0x0C80,0x0C,0,'S',0x2074696B,0,IDB_ITEM_MILABREGASORB,20013,0,0,0xF,2,3,5
,' ',60,47,0,0,0,0,0,0,0,0,0,0,16,18,3,'S',0,0 },
{ "Steelclash",'U',"Kite Shield","Shield",0x0C80,0xC6,0,'U',0x2074696B,0,IDB_ITEM_STEELCLASH,4200,0,0,0xF,2,3,5,' ',150,
47,0,0,0,0,0,0,0,0,0,0,16,18,3,'S',0,0 },
{ "Tower Shield",'N',0,"Shield",0x0C90,0x0,0,'N',0x20776F74,0,IDB_ITEM_TOWERSHIELD,4100,4,4,0xF,2,3,5,' ',60,75,0,0,0,0,
0,0,0,0,0,0,22,25,3,'S',0,0 },
{ "Sigon's Guard",'S',"Tower Shield","Shield",0x0C90,0x12,0,'S',0x20776F74,0,IDB_ITEM_SIGONSGUARD,20014,0,0,0xF,2,3,5,'
',120,75,0,0,0,0,0,0,0,0,0,0,22,25,3,'S',0,0 },
{ "Bverrit Keep",'U',"Tower Shield","Shield",0x0C90,0xC8,0,'U',0x20776F74,0,IDB_ITEM_BVERRITKEEP,4200,0,0,0xF,2,3,5,' ',
255,75,0,0,0,0,0,0,0,0,0,0,22,25,3,'S',0,0 },
{ "Gothic Shield",'N',0,"Shield",0x0CA0,0x0,0,'N',0x20737467,0,IDB_ITEM_GOTHICSHIELD,4100,4,4,0xF,2,4,5,' ',40,60,0,0,0,
0,0,0,0,0,0,0,30,35,3,'S',0,0 },
{ "Isenhart's Parry",'S',"Gothic Shield","Shield",0x0CA0,0x08,0,'S',0x20737467,0,IDB_ITEM_ISENHARTSPARRY,20012,0,0,0xF,2
,4,5,' ',80,60,0,0,0,0,0,0,0,0,0,0,30,35,3,'S',0,0 },
{ "The Ward",'U',"Gothic Shield","Shield",0x0CA0,0xCA,0,'U',0x20737467,0,IDB_ITEM_THEWARD,4200,0,0,0xF,2,4,5,' ',200,60,
0,0,0,0,0,0,0,0,0,0,30,35,3,'S',0,0 },
{ "Leather Gloves",'N',0,"Gloves",0x0CB0,0x0,0,'N',0x206C676C,0,IDB_ITEM_LEATHERGLOVES,5100,512,4096,0xF,2,2,10,' ',12,0
,0,0,0,0,0,0,0,0,0,0,2,3,0,' ',0,0 },
{ "Death's Hand",'S',"Leather Gloves","Gloves",0x0CB0,0x18,0,'S',0x206C676C,0,IDB_ITEM_DEATHSHAND,20008,0,0,0xF,2,2,10,'
 ',24,0,0,0,0,0,0,0,0,0,0,0,2,3,0,' ',0,0 },
{ "The Hand of Broc",'U',"Leather Gloves","Gloves",0x0CB0,0xCC,0,'U',0x206C676C,0,IDB_ITEM_THEHANDOFBROC,5200,0,0,0xF,2,
2,10,' ',60,0,0,0,0,0,0,0,0,0,0,0,2,3,0,' ',0,0 },
{ "Heavy Gloves",'N',"Heavy Gloves","Gloves",0x0CC0,0x0,0,'N',0x206C6776,0,IDB_ITEM_HEAVYGLOVES,5100,512,4096,0xB,2,2,10
,' ',14,0,0,0,0,0,0,0,0,0,0,0,5,6,0,' ',0,0 },
{ "Bloodfist",'U',"Heavy Gloves","Gloves",0x0CC0,0xCE,0,'U',0x206C6776,0,IDB_ITEM_BLOODFIST,5200,0,0,0xB,2,2,10,' ',70,0
,0,0,0,0,0,0,0,0,0,0,5,6,0,' ',0,0 },
{ "Chain Gloves",'N',0,"Gloves",0x0CD0,0x0,0,'N',0x206C676D,0,IDB_ITEM_CHAINGLOVES,5100,512,4096,0xF,2,2,10,' ',16,25,0,
0,0,0,0,0,0,0,0,0,8,9,0,' ',0,0 },
{ "Cleglaw's Pincers",'S',"Chain Gloves","Gloves",0x0CD0,0x04,0,'S',0x206C676D,0,IDB_ITEM_CLEGLAWSPINCERS,20007,0,0,0xF,
2,2,10,' ',32,25,0,0,0,0,0,0,0,0,0,0,8,9,0,' ',0,0 },
{ "Chance Guards",'U',"Chain Gloves","Gloves",0x0CD0,0xD0,0,'U',0x206C676D,0,IDB_ITEM_CHANCEGUARDS,5200,0,0,0xF,2,2,10,'
 ',80,25,0,0,0,0,0,0,0,0,0,0,8,9,0,' ',0,0 },
{ "Light Gauntlets",'N',0,"Gloves",0x0CE0,0x0,0,'N',0x206C6774,0,IDB_ITEM_LIGHTGAUNTLETS,5100,512,4096,0xF,2,2,10,' ',18
,45,0,0,0,0,0,0,0,0,0,0,9,11,0,' ',0,0 },
{ "Iratha's Cuff",'S',"Light Gauntlets","Gloves",0x0CE0,0x06,0x7FF0,'S',0x206C6774,0,IDB_ITEM_IRATHASCUFF,20011,0,0,0xF,
2,2,10,' ',36,45,0,0,0,0,0,0,0,0,0,0,9,11,0,' ',0,0 },
{ "Arctic Mitts",'S',"Light Gauntlets","Gloves",0x0CE0,0x1C,0x800F,'S',0x206C6774,0,IDB_ITEM_ARCTICMITTS,20003,0,0,0xF,2
,2,10,' ',36,45,0,0,0,0,0,0,0,0,0,0,9,11,0,' ',0,0 },
{ "Magefist",'U',"Light Gauntlets","Gloves",0x0CE0,0xD2,0,'U',0x206C6774,0,IDB_ITEM_MAGEFIST,5200,0,0,0xF,2,2,10,' ',90,
45,0,0,0,0,0,0,0,0,0,0,9,11,0,' ',0,0 },
{ "Gauntlets",'N',0,"Gloves",0x0CF0,0x0,0,'N',0x206C6768,0,IDB_ITEM_GAUNTLETS,5100,512,4096,0xF,2,2,10,' ',24,60,0,0,0,0
,0,0,0,0,0,0,12,15,0,' ',0,0 },
```

```
{ "Sigon's Gage",'S',"Gauntlets","Gloves",0x0CF0,0x12,0,'S',0x206C6768,0,IDB_ITEM_SIGONSGAUNTLETS,20014,0,0,0xF,2,2,10,'
 ',48,60,0,0,0,0,0,0,0,0,0,0,12,15,0,' ',0,0 },
{ "Frostburn",'U',"Gauntlets","Gloves",0x0CF0,0xD4,0,'U',0x206C6768,0,IDB_ITEM_FROSTBURN,5200,0,0,0xF,2,2,10,' ',120,60,
0,0,0,0,0,0,0,0,0,0,12,15,0,' ',0,0 },
{ "Boots",'N',0,"Boots",0x0D00,0x0,0,'N',0x2074626C,0,IDB_ITEM_LEATHERBOOTS,6100,256,2048,0xF,2,2,9,' ',12,0,0,0,0,0,0,0
,0,0,0,0,2,3,0,' ',0,0 },
{ "Tancred's Hobnails",'S',"Boots","Boots",0x0D00,0x10,0,'S',0x2074626C,0,IDB_ITEM_TANCREDSHOBNAILS,20015,0,0,0xF,2,2,9,
' ',24,24,0,0,0,0,0,0,0,0,0,0,2,3,0,' ',0,0 },
{ "Hotspur",'U',"Boots","Boots",0x0D00,0xD6,0,'U',0x2074626C,0,IDB_ITEM_HOTSPUR,6200,0,0,0xF,2,2,9,' ',60,0,0,0,0,0,0,0,
0,0,0,0,2,3,0,' ',0,0 },
{ "Heavy Boots",'N',0,"Boots",0x0D10,0x0,0,'N',0x20746276,0,IDB_ITEM_HEAVYBOOTS,6100,256,2048,0xB,2,2,9,' ',14,18,0,0,0,
0,0,0,0,0,0,0,5,6,0,' ',0,0 },
{ "Gorefoot",'U',"Heavy Boots","Boots",0x0D10,0xD8,0,'U',0x20746276,0,IDB_ITEM_GOREFOOT,6200,0,0,0xB,2,2,9,' ',70,18,0,0
,0,0,0,0,0,0,0,0,5,6,0,' ',0,0 },
{ "Chain Boots",'N',0,"Boots",0x0D20,0x0,0,'N',0x2074626D,0,IDB_ITEM_CHAINBOOTS,6100,256,2048,0xF,2,2,9,' ',16,30,0,0,0,
0,0,0,0,0,0,0,8,9,0,' ',0,0 },
{ "Hsarus' Iron Heel",'S',"Chain Boots","Boots",0x0D20,0x02,0,'S',0x2074626D,0,IDB_ITEM_HSARUSIRONHEEL,20009,0,0,0xF,2,2
,9,' ',32,30,0,0,0,0,0,0,0,0,0,0,8,9,0,' ',0,0 },
{ "Treads of Cthon",'U',"Chain Boots","Boots",0x0D20,0xDA,0,'U',0x2074626D,0,IDB_ITEM_TREADSOFCTHON,6200,0,0,0xF,2,2,9,'
 ',80,30,0,0,0,0,0,0,0,0,0,0,8,9,0,' ',0,0 },
{ "Light Plate",'N',0,"Boots",0x0D30,0x0,0,'N',0x20746274,0,IDB_ITEM_LIGHTPLATEBOOTS,6100,256,2048,0xF,2,2,9,' ',18,50,0
,0,0,0,0,0,0,0,0,0,9,11,0,' ',0,0 },
{ "Vidala's Fetlock",'S',"Light Plate","Boots",0x0D30,0x0A,0,'S',0x20746274,0,IDB_ITEM_VIDALASFETLOCK,20016,0,0,0xF,2,2,
9,' ',30,0,0,0,0,0,0,0,0,0,0,0,9,11,0,' ',0,0 },
{ "Goblin Toe",'U',"Light Plate","Boots",0x0D30,0xDC,0,'U',0x20746274,0,IDB_ITEM_GOBLINTOE,6200,0,0,0xF,2,2,9,' ',90,50,
0,0,0,0,0,0,0,0,0,0,9,11,0,' ',0,0 },
{ "Greaves",'N',0,"Boots",0x0D40,0x0,0,'N',0x20746268,0,IDB_ITEM_PLATEBOOTS,6100,256,2048,0xF,2,2,9,' ',24,70,0,0,0,0,0,
0,0,0,0,0,12,15,0,' ',0,0 },
{ "Sigon's Sabot",'S',"Greaves","Boots",0x0D40,0x12,0,'S',0x20746268,0,IDB_ITEM_SIGONSGREAVES,20014,0,0,0xF,2,2,9,' ',48
,70,0,0,0,0,0,0,0,0,0,0,12,15,0,' ',0,0 },
{ "Tearhaunch",'U',"Greaves","Boots",0x0D40,0xDE,0,'U',0x20746268,0,IDB_ITEM_TEARHAUNCH,6200,0,0,0xF,2,2,9,' ',120,70,0,
0,0,0,0,0,0,0,0,0,12,15,0,' ',0,0 },
{ "Sash",'N',0,"Belt",0x0D50,0x0,0,'N',0x206C626C,0,IDB_ITEM_SASH,7100,1024,8192,0xF,2,1,8,' ',12,0,0,0,0,0,0,0,0,0,0,0,
2,2,0,' ',0,0 },
{ "Death's Guard",'S',"Sash","Belt",0x0D50,0x18,0,'S',0x206C626C,0,IDB_ITEM_DEATHSGUARD,20008,0,0,0xF,2,1,8,' ',24,0,0,0
,0,0,0,0,0,0,0,2,2,0,' ',0,0 },
{ "Lenymo",'U',"Sash","Belt",0x0D50,0xE0,0,'U',0x206C626C,0,IDB_ITEM_LENYMSCORD,7200,0,0,0xF,2,1,8,' ',60,0,0,0,0,0,0,0,
0,0,0,0,2,2,0,' ',0,0 },
{ "Light Belt",'N',0,"Belt",0x0D60,0x0,0,'N',0x206C6276,0,IDB_ITEM_LIGHTBELT,7100,1024,8192,0xF,2,1,8,' ',14,0,0,0,0,0,0
,0,0,0,0,0,3,3,0,' ',0,0 },
{ "Arctic Binding",'S',"Light Belt","Belt",0x0D60,0x1C,0,'S',0x206C6276,0,IDB_ITEM_ARCTICBINDING,20003,0,0,0xF,2,1,8,' '
,28,0,0,0,0,0,0,0,0,0,0,0,3,3,0,' ',0,0 },
{ "Snakecord",'U',"Light Belt","Belt",0x0D60,0xE2,0,'U',0x206C6276,0,IDB_ITEM_SNAKECORD,7200,0,0,0xF,2,1,8,' ',70,0,0,0,
0,0,0,0,0,0,0,0,3,3,0,' ',0,0 },
{ "Belt",'N',0,"Belt",0x0D70,0x0,0,'N',0x206C626D,0,IDB_ITEM_BELT,7100,1024,8192,0xF,2,1,8,' ',16,25,0,0,0,0,0,0,0,0,0,0
,5,5,0,' ',0,0 },
{ "Hsarus' Iron Stay",'S',"Belt","Belt",0x0D70,0x02,0,'S',0x206C626D,0,IDB_ITEM_HSARUSIRONSTAY,20009,0,0,0xF,2,1,8,' ',3
2,25,0,0,0,0,0,0,0,0,0,0,5,5,0,' ',0,0 },
{ "Nightsmoke",'U',"Belt","Belt",0x0D70,0xE4,0,'U',0x206C626D,0,IDB_ITEM_NIGHTSMOKE,7200,0,0,0xF,2,1,8,' ',80,25,0,0,0,0
,0,0,0,0,0,0,5,5,0,' ',0,0 },
{ "Heavy Belt",'N',0,"Belt",0x0D80,0x0,0,'N',0x206C6274,0,IDB_ITEM_HEAVYBELT,7100,1024,8192,0xF,2,1,8,' ',18,45,0,0,0,0,
0,0,0,0,0,0,6,6,0,' ',0,0 },
{ "Iratha's Cord",'S',"Heavy Belt","Belt",0x0D80,0x06,0x07F0,'S',0x206C6274,0,IDB_ITEM_IRATHASCORD,20011,0,0,0xF,2,1,8,'
 ',36,45,0,0,0,0,0,0,0,0,0,0,6,6,0,' ',0,0 },
{ "Infernal Buckle",'S',"Heavy Belt","Belt",0x0D80,0x14,0xF80F,'S',0x206C6274,0,IDB_ITEM_INFERNALBUCKLE,20010,0,0,0xF,2,
1,8,' ',36,45,0,0,0,0,0,0,0,0,0,0,6,6,0,' ',0,0 },
{ "Goldwrap",'U',"Heavy Belt","Belt",0x0D80,0xE6,0,'U',0x206C6274,0,IDB_ITEM_GOLDWRAP,7200,0,0,0xF,2,1,8,' ',90,45,0,0,0
,0,0,0,0,0,0,0,6,6,0,' ',0,0 },
{ "Plated Belt",'N',0,"Belt",0x0D90,0x0,0,'N',0x206C6268,0,IDB_ITEM_GIRDLE,7100,1024,8192,0xF,2,1,8,' ',24,60,0,0,0,0,0,0,
0,0,0,0,0,8,11,0,' ',0,0 },
{ "Sigon's Wrap",'S',"Plated Belt","Belt",0x0D90,0x12,0,'S',0x206C6268,0,IDB_ITEM_SIGONSWRAP,20014,0,0,0xF,2,1,8,' ',48,
60,0,0,0,0,0,0,0,0,0,0,8,11,0,' ',0,0 },
{ "Bladebuckle",'U',"Plated Belt","Belt",0x0D90,0xE8,0,'U',0x206C6268,0,IDB_ITEM_BLADEBUCKLE,7200,0,0,0xF,2,1,8,' ',120,
60,0,0,0,0,0,0,0,0,0,0,8,11,0,' ',0,0 },
{ "Bone Helm",'N',0,"Helm",0x0DA0,0x0,0,'N',0x206D6862,0,IDB_ITEM_BONEHELM,2100,1,2,0xF,2,2,1,' ',40,25,0,0,0,0,0,0,0,0,
0,0,33,36,0,' ',0,0 },
{ "Tancred's Skull",'S',"Bone Helm","Helm",0x0DA0,0x10,0,'S',0x206D6862,0,IDB_ITEM_TANCREDSSKULL,20015,0,0,0xF,2,2,1,' '
,80,25,0,0,0,0,0,0,0,0,0,0,33,36,0,' ',0,0 },
{ "Wormskull",'U',"Bone Helm","Helm",0x0DA0,0x96,0,'U',0x206D6862,0,IDB_ITEM_WORMSKULL,2200,0,0,0xF,2,2,1,' ',200,25,0,0
,0,0,0,0,0,0,0,0,33,36,0,' ',0,0 },
{ "Bone Shield",'N',0,"Shield",0x0DB0,0x0,0,'N',0x20687362,0,IDB_ITEM_BONESHIELD,4100,4,4,0xB,2,3,5,' ',40,25,0,0,0,0,0,
0,0,0,0,0,10,30,0,' ',0,0 },
{ "Wall of the Eyeless",'U',"Bone Shield","Shield",0x0DB0,0xC2,0,'U',0x20687362,0,IDB_ITEM_WALLOFTHEEYELESS,4200,0,0,0xB
,2,3,5,' ',200,25,0,0,0,0,0,0,0,0,0,0,10,30,0,' ',0,0 },
{ "Spiked Shield",'N',0,"Shield",0x0DC0,0x0,0,'N',0x206B7073,0,IDB_ITEM_SPIKEDSHIELD,4100,4,4,0xB,2,3,5,' ',40,30,0,0,0,
0,0,0,0,0,0,0,15,25,0,' ',0,0 },
{ "Swordback Hold",'U',"Spiked Shield","Shield",0x0DC0,0xC4,0,'U',0x206B7073,0,IDB_ITEM_SWORDBACKHOLD,4200,0,0,0xB,2,3,5
,' ',200,30,0,0,0,0,0,0,0,0,0,0,15,25,0,' ',0,0 },
{ "War Hat",'E',"Cap","Helm",0x0DD0,0x0,0,'N',0x20706178,0,IDB_ITEM_CAP,2300,1,2,0x3,2,2,1,' ',12,20,0,0,0,0,0,0,0,0,0,0
```

```
,45,53,2,'H',0,0 },
{ "Sallet",'E',"Skull Cap","Helm",0x0DE0,0x0,0,'N',0x20706B78,0,IDB_ITEM_SKULLCAP,2300,1,2,0x3,2,2,1,' ',18,43,0,0,0,0,0
,0,0,0,0,0,52,62,2,'H',0,0 },
{ "Casque",'E',"Helm","Helm",0x0DF0,0x0,0,'N',0x206D6C78,0,IDB_ITEM_HELM,2300,1,2,0x3,2,2,1,' ',24,59,0,0,0,0,0,0,0,0,
0,63,72,2,'H',0,0 },
{ "Basinet",'E',"Full Helm","Helm",0x0E00,0x0,0,'N',0x206C6878,0,IDB_ITEM_FULLHELM,2300,1,2,0x3,2,2,1,' ',30,82,0,0,0,0,
0,0,0,0,0,0,75,84,2,'H',0,0 },
{ "Winged Helm",'E',"Great Helm","Helm",0x0E10,0x0,0,'N',0x206D6878,0,IDB_ITEM_GREATHELM,2300,1,2,0x3,2,2,1,' ',40,115,0
,0,0,0,0,0,0,0,0,0,85,98,2,'H',0,0 },
{ "Grand Crown",'E',"Crown","Helm",0x0E20,0x0,0,'N',0x206E7278,0,IDB_ITEM_CROWN,2300,1,2,0x3,2,2,1,' ',50,103,0,0,0,0,0,
0,0,0,0,0,78,113,2,'H',0,0 },
{ "Death Mask",'E',"Mask","Helm",0x0E30,0x0,0,'N',0x206B7378,0,IDB_ITEM_MASK,2300,1,2,0x3,2,2,1,' ',40,55,0,0,0,0,0,0,0,
0,0,0,54,86,2,'H',0,0 },
{ "Ghost Armor",'E',0,"Armor",0x0E40,0x0,0,'N',0x20697578,0,IDB_ITEM_QUILTED,3300,1,1,0x3,2,3,3,' ',20,38,0,0,0,0,0,0,0,
0,0,0,102,117,0,' ',0,0 },
{ "Serpent Skin Armor",'E',0,"Armor",0x0E50,0x0,0,'N',0x20616578,0,IDB_ITEM_LEATHER,3300,1,1,0x3,2,3,3,' ',24,43,0,0,0,0
,0,0,0,0,0,0,111,126,0,' ',0,0 },
{ "Demonhide Armor",'E',0,"Armor",0x0E60,0x0,0,'N',0x20616C78,0,IDB_ITEM_HARDLEATHER,3300,1,1,0x3,2,3,3,' ',28,50,0,0,0,
0,0,0,0,0,0,0,122,136,0,' ',0,0 },
{ "Trellised Armor",'E',0,"Armor",0x0E70,0x0,0,'N',0x20757478,0,IDB_ITEM_STUDDEDLEATHER,3300,1,1,0x3,2,3,3,' ',32,61,0,0
,0,0,0,0,0,0,0,0,138,153,0,' ',0,0 },
{ "Linked Mail",'E',0,"Armor",0x0E80,0x0,0,'N',0x20676E78,0,IDB_ITEM_RINGMAIL,3300,1,1,0x3,2,3,3,' ',26,74,0,0,0,0,0,0,0
,0,0,0,158,172,0,' ',0,0 },
{ "Tigulated Mail",'E',0,"Armor",0x0E90,0x0,0,'N',0x206C6378,0,IDB_ITEM_SCALEMAIL,3300,1,1,0x3,2,3,3,' ',36,86,0,0,0,0,0
,0,0,0,0,0,176,190,0,' ',0,0 },
{ "Mesh Armor",'E',0,"Armor",0x0EA0,0x0,0,'N',0x206E6878,0,IDB_ITEM_CHAINMAIL,3300,1,1,0x3,2,3,3,' ',45,92,0,0,0,0,0,0,0
,0,0,0,198,213,0,' ',0,0 },
{ "Curiass",'E',0,"Armor",0x0EB0,0x0,0,'N',0x20737278,0,IDB_ITEM_BREASTPLATE,3300,1,1,0x3,2,3,3,' ',50,65,0,0,0,0,0,0,0,
0,0,0,188,202,0,' ',0,0 },
{ "Russet Armor",'E',0,"Armor",0x0EC0,0x0,0,'N',0x206C7078,0,IDB_ITEM_SPLINTMAIL,3300,1,1,0x3,2,3,3,' ',30,97,0,0,0,0,0,
0,0,0,0,0,225,243,0,' ',0,0 },
{ "Templar Coat",'E',0,"Armor",0x0ED0,0x0,0,'N',0x20746C78,0,IDB_ITEM_PLATEMAIL,3300,1,1,0x3,2,3,3,' ',60,118,0,0,0,0,0,
0,0,0,0,0,252,274,0,' ',0,0 },
{ "Sharktooth Armor",'E',0,"Armor",0x0EE0,0x0,0,'N',0x20646C78,0,IDB_ITEM_FIELDPLATE,3300,1,1,0x3,2,3,3,' ',48,103,0,0,0
,0,0,0,0,0,0,0,242,258,0,' ',0,0 },
{ "Embossed Plate",'E',0,"Armor",0x0EF0,0x0,0,'N',0x20687478,0,IDB_ITEM_GOTHICPLATE,3300,1,1,0x3,2,3,3,' ',55,125,0,0,0,
0,0,0,0,0,0,0,282,303,0,' ',0,0 },
{ "Chaos Armor",'E',0,"Armor",0x0F00,0x0,0,'N',0x206C7578,0,IDB_ITEM_FULLPLATE,3300,1,1,0x3,2,3,3,' ',70,140,0,0,0,0,0,0,0
,0,0,0,0,315,342,0,' ',0,0 },
{ "Ornate Plate",'E',0,"Armor",0x0F10,0x0,0,'N',0x20726178,0,IDB_ITEM_ANCIENTARMOR,3300,1,1,0x3,2,3,3,' ',60,170,0,0,0,0,0
,0,0,0,0,0,0,417,450,0,' ',0,0 },
{ "Mage Plate",'E',0,"Armor",0x0F20,0x0,0,'N',0x20707478,0,IDB_ITEM_LIGHTPLATE,3300,1,1,0x3,2,3,3,' ',60,55,0,0,0,0,0,0,0,
0,0,0,0,225,261,0,' ',0,0 },
{ "Defender",'E',"Buckler","Shield",0x0F30,0x0,0,'N',0x20637578,0,IDB_ITEM_BUCKLER,4300,4,4,0x3,2,2,5,' ',12,38,0,0,0,0,
0,0,0,0,0,0,41,49,1,'S',0,0 },
{ "Round Shield",'E',"Small Shield","Shield",0x0F40,0x0,0,'N',0x206C6D78,0,IDB_ITEM_SMALLSHIELD,4300,4,4,0x3,2,2,5,' ',1
6,53,0,0,0,0,0,0,0,0,0,0,0,47,55,2,'S',0,0 },
{ "Scutum",'E',"Large Shield","Shield",0x0F50,0x0,0,'N',0x20677278,0,IDB_ITEM_LARGESHIELD,4300,4,4,0x3,2,3,5,' ',24,71,0
,0,0,0,0,0,0,0,0,0,53,61,3,'S',0,0 },
{ "Dragon Shield",'E',"Kite Shield","Shield",0x0F60,0x0,0,'N',0x20746978,0,IDB_ITEM_KITESHIELD,4300,4,4,0x3,2,3,5,' ',30
,91,0,0,0,0,0,0,0,0,0,0,59,67,3,'S',0,0 },
{ "Pavise",'E',"Tower Shield","Shield",0x0F70,0x0,0,'N',0x20776F78,0,IDB_ITEM_TOWERSHIELD,4300,4,4,0x3,2,3,5,' ',60,133,
0,0,0,0,0,0,0,0,0,0,68,78,3,'S',0,0 },
{ "Ancient Shield",'E',"Gothic Shield","Shield",0x0F80,0x0,0,'N',0x20737478,0,IDB_ITEM_GOTHICSHIELD,4300,4,4,0x3,2,4,5,'
 ',40,110,0,0,0,0,0,0,0,0,0,0,80,93,3,'S',0,0 },
{ "Demonhide Gloves",'E',"Leather Gloves","Gloves",0x0F90,0x0,0,'N',0x20676C78,0,IDB_ITEM_LEATHERGLOVES,5300,512,4096,0x
3,2,2,10,' ',12,20,0,0,0,0,0,0,0,0,0,0,28,35,0,' ',0,0 },
{ "Sharkskin Gloves",'E',"Heavy Gloves","Gloves",0x0FA0,0x0,0,'N',0x20677678,0,IDB_ITEM_HEAVYGLOVES,5300,512,4096,0x3,2,
2,10,' ',14,20,0,0,0,0,0,0,0,0,0,0,33,39,0,' ',0,0 },
{ "Heavy Bracers",'E',"Chain Gloves","Gloves",0x0FB0,0x0,0,'N',0x20676D78,0,IDB_ITEM_CHAINGLOVES,5300,512,4096,0x3,2,2,1
0,' ',16,58,0,0,0,0,0,0,0,0,0,0,37,44,0,' ',0,0 },
{ "Battle Gauntlets",'E',"Light Gauntlets","Gloves",0x0FC0,0x0,0,'N',0x20677478,0,IDB_ITEM_LIGHTGAUNTLETS,5300,512,4096,
0x3,2,2,10,' ',18,88,0,0,0,0,0,0,0,0,0,0,39,47,0,' ',0,0 },
{ "War Gauntlets",'E',"Gauntlets","Gloves",0x0FD0,0x0,0,'N',0x20676878,0,IDB_ITEM_GAUNTLETS,5300,512,4096,0x3,2,2,10,' '
,24,110,0,0,0,0,0,0,0,0,0,0,43,53,0,' ',0,0 },
{ "Demonhide Boots",'E',"Boots","Boots",0x0FE0,0x0,0,'N',0x20626C78,0,IDB_ITEM_LEATHERBOOTS,6300,256,2048,0x3,2,2,9,' ',
12,20,0,0,0,0,0,0,0,0,0,0,28,35,0,' ',0,0 },
{ "Sharkskin Boots",'E',"Heavy Boots","Boots",0x0FF0,0x0,0,'N',0x20627678,0,IDB_ITEM_HEAVYBOOTS,6300,256,2048,0x3,2,2,9,
' ',14,47,0,0,0,0,0,0,0,0,0,0,33,39,0,' ',0,0 },
{ "Mesh Boots",'E',"Chain Boots","Boots",0x1000,0x0,0,'N',0x20626D78,0,IDB_ITEM_CHAINBOOTS,6300,256,2048,0x3,2,2,9,' ',1
5,65,0,0,0,0,0,0,0,0,0,0,37,44,0,' ',0,0 },
{ "Battle Boots",'E',"Light Plate","Boots",0x1010,0x0,0,'N',0x20627478,0,IDB_ITEM_LIGHTPLATEBOOTS,6300,256,2048,0x3,2,2,
9,' ',18,95,0,0,0,0,0,0,0,0,0,0,39,47,0,' ',0,0 },
{ "War Boots",'E',"Greaves","Boots",0x1020,0x0,0,'N',0x20626878,0,IDB_ITEM_PLATEBOOTS,6300,256,2048,0x3,2,2,9,' ',24,125
,0,0,0,0,0,0,0,0,0,0,43,53,0,' ',0,0 },
{ "Demonhide Sash",'E',"Sash","Belt",0x1030,0x0,0,'N',0x20626C7A,0,IDB_ITEM_SASH,7300,1024,8192,0x3,2,1,8,' ',12,20,0,0,
0,0,0,0,0,0,0,0,29,34,0,' ',0,0 },
{ "Sharkskin Belt",'E',"Light Belt","Belt",0x1040,0x0,0,'N',0x2062767A,0,IDB_ITEM_LIGHTBELT,7300,1024,8192,0x3,2,1,8,' '
,14,20,0,0,0,0,0,0,0,0,0,0,31,36,0,' ',0,0 },
```

```
{ "Mesh Belt",'E',"Belt","Belt",0x1050,0x0,0,'N',0x20626D7A,0,IDB_ITEM_BELT,7300,1024,8192,0x3,2,1,8,' ',15,58,0,0,0,0,0
,0,0,0,0,0,35,40,0,' ',0,0 },
{ "Battle Belt",'E',"Heavy Belt","Belt",0x1060,0x0,0,'N',0x2062747A,0,IDB_ITEM_HEAVYBELT,7300,1024,8192,0x3,2,1,8,' ',18
,88,0,0,0,0,0,0,0,0,0,0,37,42,0,' ',0,0 },
{ "War Belt",'E',"Plated Belt","Belt",0x1070,0x0,0,'N',0x2062687A,0,IDB_ITEM_GIRDLE,7300,1024,8192,0x3,2,1,8,' ',24,110,
0,0,0,0,0,0,0,0,0,0,41,52,0,' ',0,0 },
{ "Grim Helm",'E',"Bone Helm","Helm",0x1080,0x0,0,'N',0x20396878,0,IDB_ITEM_BONEHELM,2300,1,2,0x3,2,2,1,' ',250,58,0,0,0
,0,0,0,0,0,0,0,60,125,0,' ',0,0 },
{ "Grim Shield",'E',"Bone Shield","Shield",0x1090,0x0,0,'N',0x20687378,0,IDB_ITEM_BONESHIELD,4300,4,4,0x3,2,3,5,' ',40,5
8,0,0,0,0,0,0,0,0,0,0,50,150,0,' ',0,0 },
{ "Barbed Shield",'E',"Spiked Shield","Shield",0x10A0,0x0,0,'N',0x206B7078,0,IDB_ITEM_SPIKEDSHIELD,4300,4,4,0x3,2,3,5,'
',40,65,0,0,0,0,0,0,0,0,0,0,58,78,0,' ',0,0 },
{ "Elixer of Vitality",'Q',0,"Special",0x10B0,0x0,0,'N',0x20786C65,1,IDB_ITEM_ELIXEROFVITALITY,40000,0,0,0x0,1,1,-1,' ',
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,' ',0,0 },
{ "HPO not translated",'X',0,"Trash",0x10C0,0x0,0,'N',0x206F7068,1,IDB_ITEM_HPO,90000,0,0,0x0,1,1,0,' ',0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,' ',0,0 },
{ "MPO not translated",'X',0,"Trash",0x10D0,0x0,0,'N',0x206F706D,1,IDB_ITEM_MPO,90000,0,0,0x0,1,1,0,' ',0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,' ',0,0 },
{ "HPF not translated",'X',0,"Trash",0x10E0,0x0,0,'N',0x20667068,1,IDB_ITEM_HPF,90000,0,0,0x0,1,1,0,' ',0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,' ',0,0 },
{ "MPF not translated",'X',0,"Trash",0x10F0,0x0,0,'N',0x2066706D,1,IDB_ITEM_MPF,90000,0,0,0x0,1,1,0,' ',0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,' ',0,0 },
{ "Stamina Potion",'A',0,"Potion",0x1100,0x0,0,'N',0x20737076,1,IDB_ITEM_STAMINA,10100,0,0,0x0,1,1,0,' ',0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,' ',0,0 },
{ "Antidote Potion",'A',0,"Potion",0x1110,0x0,0,'N',0x20737079,1,IDB_ITEM_ANTIDOTE,10100,0,0,0x0,1,1,0,' ',0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,' ',0,0 },
{ "Rejuvenation Potion",'A',0,"Potion",0x1120,0x0,0,'N',0x20737672,1,IDB_ITEM_REJUV,10100,0,0,0x0,1,1,0,' ',0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,' ',0,"Heals 35% Life and Mana" },
{ "Full Rejuvenation Potion",'A',0,"Potion",0x1130,0x0,0,'N',0x206C7672,1,IDB_ITEM_FULLREJUV,10100,0,0,0x0,1,1,0,' ',0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,' ',0,"Heals 100% Life and Mana" },
{ "Thawing Potion",'A',0,"Potion",0x1140,0x0,0,'N',0x20736D77,1,IDB_ITEM_THAWING,10100,0,0,0x0,1,1,0,' ',0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,' ',0,0 },
{ "Tome of Town Portal",'A',0,"Scroll Tome",0x1150,0x0,0,'N',0x206B6274,0,IDB_ITEM_TOMEBLUE,10200,0,0,0x0,1,2,-1,' ',0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,' ',20,0 },
{ "Tome of Identify",'A',0,"Scroll Tome",0x1160,0x0,0,'N',0x206B6269,0,IDB_ITEM_TOMERED,10200,0,0,0x0,1,2,-1,' ',0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,' ',20,0 },
{ "Amulet",'M',0,"Amulet",0x1170,0x0,0,'M',0x20756D61,0,0,9100,4096,32768,0xE,1,1,2,' ',0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,'
',0,0 },
{ "Civerb's Icon",'S',"Amulet","Amulet",0x1170,0x00,0x000E,'S',0x20756D61,0,0,20006,0,0,0xE,1,1,2,' ',0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,' ',0,0 },
{ "Iratha's Collar",'S',"Amulet","Amulet",0x1170,0x06,0x0030,'S',0x20756D61,0,0,20011,0,0,0xE,1,1,2,' ',0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,' ',0,0 },
{ "Vidala's Snare",'S',"Amulet","Amulet",0x1170,0x0A,0x00C0,'S',0x20756D61,0,0,20016,0,0,0xE,1,1,2,' ',0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,' ',0,0 },
{ "Cathan's Sigil",'S',"Amulet","Amulet",0x1170,0x0E,0x0100,'S',0x20756D61,0,0,20005,0,0,0xE,1,1,2,' ',0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,' ',0,0 },
{ "Tancred's Weird",'S',"Amulet","Amulet",0x1170,0x10,0x3E00,'S',0x20756D61,0,0,20015,0,0,0xE,1,1,2,' ',0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,' ',0,0 },
{ "Angelical Wings",'S',"Amulet","Amulet",0x1170,0x1A,0xC000,'S',0x20756D61,0,0,20001,0,0,0xE,1,1,2,' ',0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,' ',0,0 },
{ "Arcanna's Sign",'S',"Amulet","Amulet",0x1170,0x1E,0x0001,'S',0x20756D61,0,0,20002,0,0,0xE,1,1,2,' ',0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,' ',0,0 },
{ "Nokozan Relic",'U',"Amulet","Amulet",0x1170,0xEA,0,'U',0x20756D61,0,0,9200,0,0,0xE,1,1,2,' ',0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,' ',0,0 },
{ "The Eye of Etlich",'U',"Amulet","Amulet",0x1170,0xEC,0,'U',0x20756D61,0,0,9200,0,0,0xE,1,1,2,' ',0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,' ',0,0 },
{ "The Mahim-Oak Curio",'U',"Amulet","Amulet",0x1170,0xEE,0,'U',0x20756D61,0,0,9200,0,0,0xE,1,1,2,' ',0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,' ',0,0 },
{ "Top of Horadric Staff of Craft",'Q',0,"Special",0x1180,0x0,0,'N',0x20706976,0,IDB_ITEM_TOPOFHORADRICSTAFF,30002,0,0,0
x1,1,1,-1,' ',0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,' ',0,0 },
{ "Ring",'M',0,"Ring",0x1190,0x0,0,'M',0x206E6972,0,0,8100,2048,16384,0xE,1,1,6,' ',0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,' ',0,
0 },
{ "Cathan's Seal",'S',"Ring","Ring",0x1190,0x0E,0x3F00,'S',0x206E6972,0,0,20005,0,0,0xE,1,1,6,' ',0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,' ',0,0 },
{ "Angelical Halo",'S',"Ring","Ring",0x1190,0x1A,0xC0FF,'S',0x206E6972,0,0,20001,0,0,0xE,1,1,6,' ',0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,' ',0,0 },
{ "Nagelring",'U',"Ring","Ring",0x1190,0xF0,0,'U',0x206E6972,0,0,8200,0,0,0xE,1,1,6,' ',0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,'
',0,0 },
{ "Manald Heal",'U',"Ring","Ring",0x1190,0xF2,0,'U',0x206E6972,0,0,8200,0,0,0xE,1,1,6,' ',0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
' ',0,0 },
{ "The Stone of Jordan",'U',"Ring","Ring",0x1190,0xF4,0,'U',0x206E6972,0,0,8200,0,0,0xE,1,1,6,' ',0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,' ',0,0 },
{ "Amulet of the Viper",'U',"Ring","Ring",0x1170,0xF6,0,'U',0x206E6972,0,0,0,0,0,0xE,1,1,6,' ',0,0,0,0,0,0,0,0,0,0,0,0
,0,0,' ',0,0 },
{ "Gold",'X',0,"Trash",0x11A0,0x0,0,'N',0x20646C67,0,IDB_ITEM_GOLD,90000,0,0,0x0,1,1,-1,' ',0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,' ',0,0 },
{ "Scroll of Inifuss",'Q',0,"Special",0x11B0,0x0,0,'N',0x20736B62,0,IDB_ITEM_SCROLLOFINIFUSS,30001,0,0,0x0,2,2,-1,' ',0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,' ',0,0 },
{ "Key to the Cairn Stones",'Q',0,"Special",0x11C0,0x0,0,'N',0x20646B62,0,IDB_ITEM_SCROLLOFINIFUSS,30001,0,0,0x0,2,2,-1,
```

```
' ',0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,' ',0,0 },
{ "Arrows",'A',0,"Weapon Access",0x11D0,0x0,0,'N',0x20767161,0,IDB_ITEM_ARROWS,10000,0,0,0x0,1,3,5,' ',0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,' ',250,0 },
{ "Torch",'A',0,"Accessories",0x11E0,0x0,0,'N',0x20686374,0,IDB_ITEM_TORCH,90000,0,0,0x0,1,2,-1,' ',0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,' ',0,0 },
{ "Bolts",'A',0,"Weapon Access",0x11F0,0x0,0,'N',0x20767163,0,IDB_ITEM_BOLTS,10000,0,0,0x0,1,3,5,' ',0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,' ',150,0 },
{ "Scroll of Town Portal",'N',0,"Scroll",0x1200,0x0,0,'N',0x20637374,1,IDB_ITEM_SCROLLBLUE,10200,0,0,0x0,1,1,-1,' ',0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,' ',0,0 },
{ "Scroll of Identify",'N',0,"Scroll",0x1210,0x0,0,'N',0x20637369,1,IDB_ITEM_SCROLLRED,10200,0,0,0x0,1,1,-1,' ',0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,' ',0,0 },
{ "Skeleton HRT not translated",'X',0,"Trash",0x1220,0x0,0,'N',0x20747268,1,IDB_ITEM_HEART,90100,0,0,0x0,1,1,-1,' ',0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,' ',0,0 },
{ "Skeleton BRN not xlated",'X',0,"Trash",0x1230,0x0,0,'N',0x207A7262,1,IDB_ITEM_BRAIN,90100,0,0,0x0,1,1,-1,' ',0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,' ',0,0 },
{ "Skeleton JAW not xlated",'X',0,"Trash",0x1240,0x0,0,'N',0x2077616A,1,IDB_ITEM_SKELETONJAW,90100,0,0,0x0,1,1,-1,' ',0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,' ',0,0 },
{ "Skeleton EYE not translated",'X',0,"Trash",0x1250,0x0,0,'N',0x207A7965,1,IDB_ITEM_EYE,90100,0,0,0x0,1,1,-1,' ',0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,' ',0,0 },
{ "Skeleton HRN not translated",'X',0,"Trash",0x1260,0x0,0,'N',0x206E7268,1,IDB_ITEM_SKELETONHRN,90100,0,0,0x0,1,1,-1,'
',0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,' ',0,0 },
{ "Skeleton TAL not translated",'X',0,"Trash",0x1270,0x0,0,'N',0x206C6174,1,IDB_ITEM_SKELETONTAL,90100,0,0,0x0,1,1,-1,'
',0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,' ',0,0 },
{ "Skeleton FLG not translated",'X',0,"Trash",0x1280,0x0,0,'N',0x20676C66,1,IDB_ITEM_SKELETONFLG,90100,0,0,0x0,1,1,-1,'
',0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,' ',0,0 },
{ "Skeleton FAN not translated",'X',0,"Trash",0x1290,0x0,0,'N',0x20676E66,1,IDB_ITEM_SKELETONFNG,90100,0,0,0x0,1,1,-1,'
',0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,' ',0,0 },
{ "Skeleton QLL not translated",'X',0,"Trash",0x12A0,0x0,0,'N',0x206C6C71,1,IDB_ITEM_SKELETONQLL,90100,0,0,0x0,1,1,-1,'
',0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,' ',0,0 },
{ "Skeleton SOL not translated",'X',0,"Trash",0x12B0,0x0,0,'N',0x206C6F73,1,IDB_ITEM_SKELETONSOL,90100,0,0,0x0,1,1,-1,'
',0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,' ',0,0 },
{ "Skeleton SCZ not translated",'X',0,"Trash",0x12C0,0x0,0,'N',0x207A6373,1,IDB_ITEM_SKELETONSCZ,90100,0,0,0x0,1,1,-1,'
',0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,' ',0,0 },
{ "Skeleton SPE not translated",'X',0,"Trash",0x12D0,0x0,0,'N',0x20657073,1,IDB_ITEM_SKELETONSOL,90100,0,0,0x0,1,1,-1,'
',0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,' ',0,0 },
{ "Key",'A',0,"Accessories",0x12E0,0x0,0,'N',0x2079656B,0,IDB_ITEM_KEY,10000,0,0,0x0,1,1,-1,' ',0,0,0,0,0,0,0,0,0,0,0,
0,0,0,' ',6,0 },
{ "Black Tower Key",'X',0,"Trash",0x12F0,0x0,0,'N',0x2076756C,0,IDB_ITEM_BLACKTOWERKEY,90000,0,0,0x0,1,2,-1,' ',0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,' ',0,0 },
{ "Potion of Life",'Q',0,"Special",0x1300,0x0,0,'N',0x207A7978,0,IDB_ITEM_POTIONOFLIFE,30003,0,0,0x0,1,1,-1,' ',0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,' ',0,0 },
{ "Jade Figurine",'Q',0,"Special",0x1310,0x0,0,'N',0x2034336A,0,IDB_ITEM_JADEFIGURINE,30003,0,0,0x0,1,2,-1,' ',0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,' ',0,0 },
{ "Golden Bird",'Q',0,"Special",0x1320,0x0,0,'N',0x20343367,0,IDB_ITEM_GOLDENBIRD,30003,0,0,0x0,1,2,-1,' ',0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,' ',0,0 },
{ "Lam Eseu's Tome",'Q',0,"Special",0x1330,0x0,0,'N',0x20626262,0,IDB_ITEM_LAMESEUSTOME,30003,0,0,0x0,2,2,-1,' ',0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,' ',0,0 },
{ "Horadric Cube",'Q',0,"Special",0x1340,0x0,0,'N',0x20786F62,0,IDB_ITEM_HORADRICCUBE,30002,0,0,0x0,2,2,-1,' ',0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,' ',0,0 },
{ "Horadric Scroll",'Q',0,"Special",0x1350,0x0,0,'N',0x20317274,0,IDB_ITEM_HORADRICSCROLL,30002,0,0,0x0,2,2,-1,' ',0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,' ',0,0 },
{ "Mephisto's Soulstone",'Q',0,"Special",0x1360,0x0,0,'N',0x2073736D,0,IDB_ITEM_MOPHISTOSSOULSTONE,30004,0,0,0x0,1,1,-1,
' ',0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,' ',0,0 },
{ "Book of Skill",'Q',0,"Special",0x1370,0x0,0,'N',0x20737361,0,IDB_ITEM_BOOKOFSKILL,30002,0,0,0x0,2,2,-1,' ',0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,' ',0,0 },
{ "Khalim's Eye",'Q',0,"Special",0x1380,0x0,0,'N',0x20796571,0,IDB_ITEM_EYE,30003,0,0,0x0,1,1,-1,' ',0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,' ',0,0 },
{ "Khalim's Heart",'Q',0,"Special",0x1390,0x0,0,'N',0x20726871,0,IDB_ITEM_HEART,30003,0,0,0x0,1,1,-1,' ',0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,' ',0,0 },
{ "Khalim's Brain",'Q',0,"Special",0x13A0,0x0,0,'N',0x20726271,0,IDB_ITEM_BRAIN,30003,0,0,0x0,1,1,-1,' ',0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,' ',0,0 },
{ "Ear",'R',0,"Special",0x13B0,0x0,0,'N',0x20726165,0,IDB_ITEM_EAR,40000,0,0,0x0,1,1,-1,' ',0,0,0,0,0,0,0,0,0,0,0,0,0,
0,' ',0,0 },
{ "Chipped Amethyst",'G',0,"Gem",0x13C0,0x0,0,'N',0x20766367,1,IDB_ITEM_CHIPPEDAMETHYST,10301,0,0,0x0,1,1,-1,' ',0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,' ',0,"Can Be Inserted in Socketed\\par Weapons, Shields or Helms\\par\\par Weapon: adds to attac
k rating\\par Shield: adds to shield's defense rating\\par Helm: adds to strength" },
{ "Flawed Amethyst",'G',0,"Gem",0x13D0,0x0,0,'N',0x20766667,1,IDB_ITEM_FLAWEDAMETHYST,10302,0,0,0x0,1,1,-1,' ',0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,' ',0,"Can Be Inserted in Socketed\\par Weapons, Shields or Helms\\par\\par Weapon: adds to attack
rating\\par Shield: adds to shield's defense rating\\par Helm: adds to strength" },
{ "Amethyst",'G',0,"Gem",0x13E0,0x0,0,'N',0x20767367,1,IDB_ITEM_AMETHYST,10303,0,0,0x0,1,1,-1,' ',0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,' ',0,"Can Be Inserted in Socketed\\par Weapons, Shields or Helms\\par\\par Weapon: adds to attack rating\\par S
hield: adds to shield's defense rating\\par Helm: adds to strength" },
{ "Flawless Amethyst",'G',0,"Gem",0x13F0,0x0,0,'N',0x20767A67,1,IDB_ITEM_FLAWLESSAMETHYST,10304,0,0,0x0,1,1,-1,' ',0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,' ',0,"Can Be Inserted in Socketed\\par Weapons, Shields or Helms\\par\\par Weapon: adds to att
ack rating\\par Shield: adds to shield's defense rating\\par Helm: adds to strength" },
{ "Perfect Amethyst",'G',0,"Gem",0x1400,0x0,0,'N',0x20767067,1,IDB_ITEM_PERFECTAMETHYST,10305,0,0,0x0,1,1,-1,' ',0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,' ',0,"Can Be Inserted in Socketed\\par Weapons, Shields or Helms\\par\\par Weapon: adds to attac
k rating\\par Shield: adds to shield's defense rating\\par Helm: adds to strength" },
{ "Chipped Topaz",'G',0,"Gem",0x1410,0x0,0,'N',0x20796367,1,IDB_ITEM_CHIPPEDTOPAZ,10301,0,0,0x0,1,1,-1,' ',0,0,0,0,0,0,0
```

```
,0,0,0,0,0,0,0,0,' ',0,"Can Be Inserted in Socketed\\par Weapons, Shields or Helms\\par\\par Weapon: adds lightning dama
ge to attack\\par Shield: adds resistance to lightning\\par Helm: adds to chance to find magic items" },
{ "Flawed Topaz",'G',0,"Gem",0x1420,0x0,0,'N',0x20796667,1,IDB_ITEM_FLAWEDTOPAZ,10302,0,0,0x0,1,1,-1,' ',0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,' ',0,"Can Be Inserted in Socketed\\par Weapons, Shields or Helms\\par\\par Weapon: adds lightning damage
 to attack\\par Shield: adds resistance to lightning\\par Helm: adds to chance to find magic items" },
{ "Topaz",'G',0,"Gem",0x1430,0x0,0,'N',0x20797367,1,IDB_ITEM_TOPAZ,10303,0,0,0x0,1,1,-1,' ',0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,' ',0,"Can Be Inserted in Socketed\\par Weapons, Shields or Helms\\par\\par Weapon: adds lightning damage to attack\\p
ar Shield: adds resistance to lightning\\par Helm: adds to chance to find magic items" },
{ "Flawless Topaz",'G',0,"Gem",0x1440,0x0,0,'N',0x20796C67,1,IDB_ITEM_FLAWLESSTOPAZ,10304,0,0,0x0,1,1,-1,' ',0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,' ',0,"Can Be Inserted in Socketed\\par Weapons, Shields or Helms\\par\\par Weapon: adds lightning da
mage to attack\\par Shield: adds resistance to lightning\\par Helm: adds to chance to find magic items" },
{ "Perfect Topaz",'G',0,"Gem",0x1450,0x0,0,'N',0x20797067,1,IDB_ITEM_PERFECTTOPAZ,10305,0,0,0x0,1,1,-1,' ',0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,' ',0,"Can Be Inserted in Socketed\\par Weapons, Shields or Helms\\par\\par Weapon: adds lightning dama
ge to attack\\par Shield: adds resistance to lightning\\par Helm: adds to chance to find magic items" },
{ "Chipped Sapphire",'G',0,"Gem",0x1460,0x0,0,'N',0x20626367,1,IDB_ITEM_CHIPPEDSAPPHIRE,10301,0,0,0x0,1,1,-1,' ',0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,' ',0,"Can Be Inserted in Socketed\\par Weapons, Shields or Helms\\par\\par Weapon: adds cold dam
age to attack\\par Shield: adds resistance to cold\\par Helm: adds to maximum mana" },
{ "Flawed Sapphire",'G',0,"Gem",0x1470,0x0,0,'N',0x20626667,1,IDB_ITEM_FLAWEDSAPPHIRE,10302,0,0,0x0,1,1,-1,' ',0,0,0,0,0
,0,0,0,0,0,0,0,0,0,' ',0,"Can Be Inserted in Socketed\\par Weapons, Shields or Helms\\par\\par Weapon: adds cold damag
e to attack\\par Shield: adds resistance to cold\\par Helm: adds to maximum mana" },
{ "Sapphire",'G',0,"Gem",0x1480,0x0,0,'N',0x20627367,1,IDB_ITEM_SAPPHIRE,10303,0,0,0x0,1,1,-1,' ',0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,' ',0,"Can Be Inserted in Socketed\\par Weapons, Shields or Helms\\par\\par Weapon: adds cold damage to attack\\
par Shield: adds resistance to cold\\par Helm: adds to maximum mana" },
{ "Flawless Sapphire",'G',0,"Gem",0x1490,0x0,0,'N',0x20626C67,1,IDB_ITEM_FLAWLESSSAPPHIRE,10304,0,0,0x0,1,1,-1,' ',0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,' ',0,"Can Be Inserted in Socketed\\par Weapons, Shields or Helms\\par\\par Weapon: adds cold d
amage to attack\\par Shield: adds resistance to cold\\par Helm: adds to maximum mana" },
{ "Perfect Sapphire",'G',0,"Gem",0x14A0,0x0,0,'N',0x20627067,1,IDB_ITEM_PERFECTSAPPHIRE,10305,0,0,0x0,1,1,-1,' ',0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,' ',0,"Can Be Inserted in Socketed\\par Weapons, Shields or Helms\\par\\par Weapon: adds cold dam
age to attack\\par Shield: adds resistance to cold\\par Helm: adds to maximum mana" },
{ "Chipped Emerald",'G',0,"Gem",0x14B0,0x0,0,'N',0x20676367,1,IDB_ITEM_CHIPPEDEMERALD,10301,0,0,0x0,1,1,-1,' ',0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,' ',0,"Can Be Inserted in Socketed\\par Weapons, Shields or Helms\\par\\par Weapon: adds poison dam
age to attack\\par Shield: adds resistance to poison\\par Helm: adds to dexterity" },
{ "Flawed Emerald",'G',0,"Gem",0x14C0,0x0,0,'N',0x20676667,1,IDB_ITEM_FLAWEDEMERALD,10302,0,0,0x0,1,1,-1,' ',0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,' ',0,"Can Be Inserted in Socketed\\par Weapons, Shields or Helms\\par\\par Weapon: adds poison damag
e to attack\\par Shield: adds resistance to poison\\par Helm: adds to dexterity" },
{ "Emerald",'G',0,"Gem",0x14D0,0x0,0,'N',0x20677367,1,IDB_ITEM_EMERALD,10303,0,0,0x0,1,1,-1,' ',0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,' ',0,"Can Be Inserted in Socketed\\par Weapons, Shields or Helms\\par\\par Weapon: adds poison damage to attack\\
par Shield: adds resistance to poison\\par Helm: adds to dexterity" },
{ "Flawless Emerald",'G',0,"Gem",0x14E0,0x0,0,'N',0x20676C67,1,IDB_ITEM_FLAWLESSEMERALD,10304,0,0,0x0,1,1,-1,' ',0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,' ',0,"Can Be Inserted in Socketed\\par Weapons, Shields or Helms\\par\\par Weapon: adds poison d
amage to attack\\par Shield: adds resistance to poison\\par Helm: adds to dexterity" },
{ "Perfect Emerald",'G',0,"Gem",0x14F0,0x0,0,'N',0x20677067,1,IDB_ITEM_PERFECTEMERALD,10305,0,0,0x0,1,1,-1,' ',0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,' ',0,"Can Be Inserted in Socketed\\par Weapons, Shields or Helms\\par\\par Weapon: adds poison dam
age to attack\\par Shield: adds resistance to poison\\par Helm: adds to dexterity" },
{ "Chipped Ruby",'G',0,"Gem",0x1500,0x0,0,'N',0x20726367,1,IDB_ITEM_CHIPPEDRUBY,10301,0,0,0x0,1,1,-1,' ',0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,' ',0,"Can Be Inserted in Socketed\\par Weapons, Shields or Helms\\par\\par Weapon: adds fire damage to a
ttack\\par Shield: adds resistance to fire\\par Helm: adds to maximum life" },
{ "Flawed Ruby",'G',0,"Gem",0x1510,0x0,0,'N',0x20726667,1,IDB_ITEM_FLAWEDRUBY,10302,0,0,0x0,1,1,-1,' ',0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,' ',0,"Can Be Inserted in Socketed\\par Weapons, Shields or Helms\\par\\par Weapon: adds fire damage to att
ack\\par Shield: adds resistance to fire\\par Helm: adds to maximum life" },
{ "Ruby",'G',0,"Gem",0x1520,0x0,0,'N',0x20727367,1,IDB_ITEM_RUBY,10303,0,0,0x0,1,1,-1,' ',0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
' ',0,"Can Be Inserted in Socketed\\par Weapons, Shields or Helms\\par\\par Weapon: adds fire damage to attack\\par Shie
ld: adds resistance to fire\\par Helm: adds to maximum life" },
{ "Flawless Ruby",'G',0,"Gem",0x1530,0x0,0,'N',0x20726C67,1,IDB_ITEM_FLAWLESSRUBY,10304,0,0,0x0,1,1,-1,' ',0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,' ',0,"Can Be Inserted in Socketed\\par Weapons, Shields or Helms\\par\\par Weapon: adds fire damage to
 attack\\par Shield: adds resistance to fire\\par Helm: adds to maximum life" },
{ "Perfect Ruby",'G',0,"Gem",0x1540,0x0,0,'N',0x20727067,1,IDB_ITEM_PERFECTRUBY,10305,0,0,0x0,1,1,-1,' ',0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,' ',0,"Can Be Inserted in Socketed\\par Weapons, Shields or Helms\\par\\par Weapon: adds fire damage to a
ttack\\par Shield: adds resistance to fire\\par Helm: adds to maximum life" },
{ "Chipped Diamond",'G',0,"Gem",0x1550,0x0,0,'N',0x20776367,1,IDB_ITEM_CHIPPEDDIAMOND,10301,0,0,0x0,1,1,-1,' ',0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,' ',0,"Can Be Inserted in Socketed\\par Weapons, Shields or Helms\\par\\par Weapon: adds to damage
 vs. undead\\par Shield: adds to all resistances\\par Helm: adds to attack rating" },
{ "Flawed Diamond",'G',0,"Gem",0x1560,0x0,0,'N',0x20776667,1,IDB_ITEM_FLAWEDDIAMOND,10302,0,0,0x0,1,1,-1,' ',0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,' ',0,"Can Be Inserted in Socketed\\par Weapons, Shields or Helms\\par\\par Weapon: adds to damage vs
. undead\\par Shield: adds to all resistances\\par Helm: adds to attack rating" },
{ "Diamond",'G',0,"Gem",0x1570,0x0,0,'N',0x20777367,1,IDB_ITEM_DIAMOND,10303,0,0,0x0,1,1,-1,' ',0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,' ',0,"Can Be Inserted in Socketed\\par Weapons, Shields or Helms\\par\\par Weapon: adds to damage vs. undead\\par
 Shield: adds to all resistances\\par Helm: adds to attack rating" },
{ "Flawless Diamond",'G',0,"Gem",0x1580,0x0,0,'N',0x20776C67,1,IDB_ITEM_FLAWLESSDIAMOND,10304,0,0,0x0,1,1,-1,' ',0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,' ',0,"Can Be Inserted in Socketed\\par Weapons, Shields or Helms\\par\\par Weapon: adds to damag
e vs. undead\\par Shield: adds to all resistances\\par Helm: adds to attack rating" },
{ "Perfect Diamond",'G',0,"Gem",0x1590,0x0,0,'N',0x20777067,1,IDB_ITEM_PERFECTDIAMOND,10305,0,0,0x0,1,1,-1,' ',0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,' ',0,"Can Be Inserted in Socketed\\par Weapons, Shields or Helms\\par\\par Weapon: adds to damage
 vs. undead\\par Shield: adds to all resistances\\par Helm: adds to attack rating" },
{ "Minor Healing Potion",'A',0,"Potion",0x15A0,0x0,0,'N',0x20317068,1,IDB_ITEM_MINORHEALING,10101,0,0,0x0,1,1,0,' ',0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,' ',0,"Points: 20" },
{ "Light Healing Potion",'A',0,"Potion",0x15B0,0x0,0,'N',0x20327068,1,IDB_ITEM_LIGHTHEALING,10101,0,0,0x0,1,1,0,' ',0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,' ',0,"Points: 40" },
{ "Healing Potion",'A',0,"Potion",0x15C0,0x0,0,'N',0x20337068,1,IDB_ITEM_HEALING,10101,0,0,0x0,1,1,0,' ',0,0,0,0,0,0,0,0
```

```
,0,0,0,0,0,0,0,' ',0,"Points: 60" },
{ "Greater Healing Potion",'A',0,"Potion",0x15D0,0x0,0,'N',0x20347068,1,IDB_ITEM_GREATERHEALING,10101,0,0,0x0,1,1,0,' ',
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,' ',0,"Points: 80" },
{ "Super Healing Potion",'A',0,"Potion",0x15E0,0x0,0,'N',0x20357068,1,IDB_ITEM_SUPERHEALING,10101,0,0,0x0,1,1,0,' ',0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,' ',0,"Points: 100" },
{ "Minor Mana Potion",'A',0,"Potion",0x15F0,0x0,0,'N',0x2031706D,1,IDB_ITEM_MINORMANA,10102,0,0,0x0,1,1,0,' ',0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,' ',0,"Points: 20" },
{ "Light Mana Potion",'A',0,"Potion",0x1600,0x0,0,'N',0x2032706D,1,IDB_ITEM_LIGHTMANA,10102,0,0,0x0,1,1,0,' ',0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,' ',0,"Points: 40" },
{ "Mana Potion",'A',0,"Potion",0x1610,0x0,0,'N',0x2033706D,1,IDB_ITEM_MANA,10102,0,0,0x0,1,1,0,' ',0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,' ',0,"Points: 60" },
{ "Greater Mana Potion",'A',0,"Potion",0x1620,0x0,0,'N',0x2034706D,1,IDB_ITEM_GREATERMANA,10102,0,0,0x0,1,1,0,' ',0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,' ',0,"Points: 80" },
{ "Super Mana Potion",'A',0,"Potion",0x1630,0x0,0,'N',0x2035706D,1,IDB_ITEM_SUPERMANA,10102,0,0,0x0,1,1,0,' ',0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,' ',0,"Points: 100" },
{ "Chipped Skull",'G',0,"Gem",0x1640,0x0,0,'N',0x20636B73,1,IDB_ITEM_CHIPPEDSKULL,10301,0,0,0x0,1,1,-1,' ',0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,' ',0,"Can Be Inserted in Socketed\\par Weapons, Shields or Helms\\par\\par Weapon: adds mana and life
steal to attack\\par Shield: adds attacker takes damage\\par Helm: adds mana and life regeneration" },
{ "Flawed Skull",'G',0,"Gem",0x1650,0x0,0,'N',0x20666B73,1,IDB_ITEM_FLAWEDSKULL,10302,0,0,0x0,1,1,-1,' ',0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,' ',0,"Can Be Inserted in Socketed\\par Weapons, Shields or Helms\\par\\par Weapon: adds mana and life st
eal to attack\\par Shield: adds attacker takes damage\\par Helm: adds mana and life regeneration" },
{ "Skull",'G',0,"Gem",0x1660,0x0,0,'N',0x20756B73,1,IDB_ITEM_SKULL,10303,0,0,0x0,1,1,-1,' ',0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,' ',0,"Can Be Inserted in Socketed\\par Weapons, Shields or Helms\\par\\par Weapon: adds mana and life steal to attack
\\par Shield: adds attacker takes damage\\par Helm: adds mana and life regeneration" },
{ "Flawless Skull",'G',0,"Gem",0x1670,0x0,0,'N',0x206C6B73,1,IDB_ITEM_FLAWLESSSKULL,10304,0,0,0x0,1,1,-1,' ',0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,' ',0,"Can Be Inserted in Socketed\\par Weapons, Shields or Helms\\par\\par Weapon: adds mana and lif
e steal to attack\\par Shield: adds attacker takes damage\\par Helm: adds mana and life regeneration" },
{ "Perfect Skull",'G',0,"Gem",0x1680,0x0,0,'N',0x207A6B73,1,IDB_ITEM_PERFECTSKULL,10305,0,0,0x0,1,1,-1,' ',0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,' ',0,"Can Be Inserted in Socketed\\par Weapons, Shields or Helms\\par\\par Weapon: adds mana and life
steal to attack\\par Shield: adds attacker takes damage\\par Helm: adds mana and life regeneration" },
};

int nItemInfos = sizeof ItemInfos / sizeof ItemInfos[0];;

HBITMAP ItemInfoGetBitmap(ItemInfo *Info)
{
#if defined(JAMELLAEDITOR)

    if (Info->hBmp) return Info->hBmp;

    if (Info->BitmapID)
    {
        Info->hBmp = LoadBitmap(hInstance,MAKEINTRESOURCE(Info->BitmapID));

        if (Info->hBmp)
        {
            BITMAP bmpinfo;
            if (GetObject(Info->hBmp,sizeof bmpinfo,&bmpinfo) == 0)
            {
                sprintf(buffer,"Bitmap Resource Error ID = %i\n",Info->BitmapID);
                MessageBox(NULL,buffer,PROGRAMNAME,
                    MB_OK | MB_ICONSTOP | MB_APPLMODAL);
                exit(0);
            }

            if (bmpinfo.bmWidth != Info->SizeX * 29 - 1 ||
                bmpinfo.bmHeight != Info->SizeY * 29 - 1)
            {
                sprintf(buffer,"Resource Error [%i|%i]\nID = %i",bmpinfo.bmWidth,bmpinfo.bmHeight,Info->BitmapID);
                MessageBox(NULL,buffer,PROGRAMNAME,
                    MB_OK | MB_ICONSTOP | MB_APPLMODAL);
                exit(0);
            }
        }
        else
        {
            sprintf(buffer,"LoadBitmap Error ID = %i\n",Info->BitmapID);
            MessageBox(NULL,buffer,PROGRAMNAME,
                MB_OK | MB_ICONSTOP | MB_APPLMODAL);
            exit(0);
        }
    }
    else
    {
        return itemunknown.hBmp;
    }
    return Info->hBmp;

#else
```

```
    return 0;

#endif
}
```

```
// ItemMg.cpp

#include "JamellaD2E.h"

int ItemMg::BaseDefense()
{
    if (!Info) FindInfo();

    int BaseAC = Info->ACMin;
    if (Info->ACMax - Info->ACMin > 0)
        BaseAC += DWARandomOffset(2) % (Info->ACMax - Info->ACMin);
    return BaseAC;
}
ArmorClass ItemMg::Defense()
{
    ArmorClass AC;

    AC.BaseAC = AC.AC = BaseDefense();
    AC.Magic = false;

    if (Decoded) {
        for(int z=0;z<MD->nMods;z++)
        {
            if (!MD->Mod[z].Code) continue;

            if (MD->Mod[z].Code == 'ar00')
            {
                AC.AC += MD->Mod[z].Mag;
                AC.Magic = true;
            }
            else if (MD->Mod[z].Code == 'ar02')
            {
                AC.AC += (AC.AC * MD->Mod[z].Mag) / 100;
                AC.Magic = true;
            }
        }
    }

    return AC;
}
Damage ItemMg::WeaponDamage()
{
    Damage  Dmg;

    Dmg.Hands = Info->Hands;
    Dmg.Magic = false;

    if (Decoded) {

        int Percent = 100;
        int BoostMax = 0;
        int BoostMin = 0;

        for(int z=0;z<MD->nMods;z++)
        {
            if (MD->Mod[z].Code == 'dm00')
            {
                BoostMax += MD->Mod[z].Mag;
                Dmg.Magic = true;
            }
            else if (MD->Mod[z].Code == 'dm01')
            {
                BoostMin += MD->Mod[z].Mag;
                Dmg.Magic = true;
            }
            else if (MD->Mod[z].Code == 'dm02')
            {
                Percent += MD->Mod[z].Mag;
                Dmg.Magic = true;
            }
        }

        Dmg.OneHandMin = (Info->OneHandDmgMin * Percent)/100 + BoostMin;
        Dmg.OneHandMax = (Info->OneHandDmgMax * Percent)/100 + BoostMax;
        Dmg.TwoHandMin = (Info->TwoHandDmgMin * Percent)/100 + BoostMin;
        Dmg.TwoHandMax = (Info->TwoHandDmgMax * Percent)/100 + BoostMax;
    }
    return Dmg;
}
```

```
unsigned int ItemMg::RequiredStrength()
{
    if (!Info)
        if (!FindInfo()) return 0;
    if (!Decoded)
        Decode();

    int MinStr = Info->MinStr;

    if (Decoded) {

        int Percent = 100;

        for(int z=0;z<MD->nMods;z++)
        {
            if (MD->Mod[z].Code == 'rq00')
            {
                Percent += MD->Mod[z].Mag;
            }
        }

        MinStr = (MinStr * Percent) / 100;
    }
    return MinStr;
}
unsigned int ItemMg::RequiredDexterity()
{
    if (!Info)
        if (!FindInfo()) return 0;
    if (!Decoded)
        Decode();

    return Info->MinDex;
}
unsigned int ItemMg::RequiredELevel()
{
    int ELevel = 0;

    if (Quality() == MAGICITEM && Decoded)
    {
        if (MD->MagicPrefix)
            ELevel = max(ELevel,MD->MagicPrefix->ELevel);
        if (MD->MagicSuffix)
            ELevel = max(ELevel,MD->MagicSuffix->ELevel);
    }
    else if (Quality() == RAREITEM && Decoded)
    {
        for(int z=0;z<6;z++) {
            if (MD->RareFix[z])
                ELevel = max(ELevel,MD->RareFix[z]->ELevel);
        }
    }

    return ELevel;
}
bool ItemMg::Decode()
{
    return MD->Decode();
}
const char* ItemMg::Name()
{
    if (!Decoded) Decode();
    return MD->Name();
}
const char* ItemMg::RichText()
{
    char *s = RichTextTmp;
    StartRTF(s);

    if (!Decoded) Decode();

    char buffers[2][64];

    switch(Quality())
    {
    case CRUDEITEM:
        strmcat(s,"\\cf0 ",MD->Name(),0);
        break;
    default:
```

```
        if (Socketed()) {
            if (!GemNum())
                strmcat(s,"\\cf5 ",MD->Name(),0);
            else
                strmcat(s,"\\cf5 Gemmed ",MD->Name(),0);
        }
        else
            strmcat(s,"\\cf0 ",MD->Name(),0);
        break;
    case SUPERIORITEM:
        strmcat(s,"\\cf0 ",MD->Name(),0);
        break;
    case MAGICITEM:
        strmcat(s,"\\cf1 ",MD->Name(),0);
        break;
    case RAREITEM:
        strmcat(s,"\\cf2 ",MD->Name(),"\\par ",Info->ItemName,0);
        break;
    case SETITEM:
        strmcat(s,"\\cf3 ",MD->Name(),"\\par ",Info->BaseItemName,0);
        break;
    case UNIQUEITEM:
        strmcat(s,"\\cf4 ",MD->Name(),"\\par ",Info->BaseItemName,0);
        break;
    }

    if (Info->Hands > 0)
    {
        Damage Dmg = WeaponDamage();

        if (Dmg.Hands >= 1)
        {
            if (Dmg.Magic) {
                strmcat(s,"\\par\\cf0 One-Hand Damage: \\cf1 ",itoa(Dmg.OneHandMin,buffers[0],10),0);
                strmcat(s," to ",itoa(Dmg.OneHandMax,buffers[0],10),0);
            }
            else {
                strmcat(s,"\\par\\cf0 One-Hand Damage: ",itoa(Dmg.OneHandMin,buffers[0],10),0);
                strmcat(s," to ",itoa(Dmg.OneHandMax,buffers[0],10),0);
            }
        }
        if (Dmg.Hands >= 2)
        {
            if (Dmg.Magic) {
                strmcat(s,"\\par\\cf0 Two-Hand Damage: \\cf1 ",itoa(Dmg.TwoHandMin,buffers[0],10),0);
                strmcat(s," to ",itoa(Dmg.TwoHandMax,buffers[0],10),0);
            }
            else {
                strmcat(s,"\\par\\cf0 Two-Hand Damage: ",itoa(Dmg.TwoHandMin,buffers[0],10),0);
                strmcat(s," to ",itoa(Dmg.TwoHandMax,buffers[0],10),0);
            }
        }
    }

    if (Info->ACMin > 0)
    {
        ArmorClass AC = Defense();
        if (AC.Magic)
            strmcat(s,"\\par\\cf0 Defense: \\cf1 ",itoa(AC.AC,buffers[0],10),0);
        else
            strmcat(s,"\\par\\cf0 Defense: ",itoa(AC.AC,buffers[0],10),0);
    }

    if (Info->Quantity > 0)
    {
        strmcat(s,"\\par\\cf0 Quantity: ",itoa(Quantity(),buffers[0],10),0);
    }

    if (Info->Durability > 0)
    {
        strmcat(s,"\\par\\cf0 Durability: ",itoa(Durability(),buffers[0],10),0);
        strmcat(s," of ",itoa(DurabilityMax(),buffers[0],10),0);
    }

#if defined(JAMELLAEDITOR)
    if (RequiredStrength() > 0)
    {
        strmcat(s,"\\par",RequiredStrength() > fc.gf.strength ? "\\cf6" : "\\cf0"," Required Strength: ",itoa(RequiredSt
rength(),buffers[0],10),0);
```

```
    }
    if (RequiredDexterity() > 0)
    {
        strmcat(s,"\\par",RequiredDexterity() > fc.gf.dexterity ? "\\cf6" : "\\cf0"," Required Dexterity: ",itoa(Require
dDexterity(),buffers[0],10),0);
    }
    if (RequiredELevel() > 0)
    {
        strmcat(s,"\\par",RequiredELevel() > fc.gf.level ? "\\cf6" : "\\cf0"," Required Level: ",itoa(RequiredELevel(),b
uffers[0],10),0);
    }
#else
    if (RequiredStrength() > 0)
    {
        strmcat(s,"\\par\\cf0 Required Strength: ",itoa(RequiredStrength(),buffers[0],10),0);
    }
    if (RequiredDexterity() > 0)
    {
        strmcat(s,"\\par\\cf0 Required Dexterity: ",itoa(RequiredDexterity(),buffers[0],10),0);
    }
    if (RequiredELevel() > 0)
    {
        strmcat(s,"\\par\\cf0 Required Level: ",itoa(RequiredELevel(),buffers[0],10),0);
    }
#endif
    switch(Quality())
    {
    default:
        if (Socketed()) {
            strmcat(s,"\\cf1 ",MD->RichAttributes(),0);
        }
        break;
    case MAGICITEM:
    case SETITEM:
    case RAREITEM:
    case UNIQUEITEM:
        strmcat(s,"\\cf1 ",MD->RichAttributes(),0);
        break;
    }

    if (Info->UndeadBonus > 100)
    {
        strmcat(s,"\\par\\cf1 ",itoa(Info->UndeadBonus,buffers[0],10),"% Damage to Undead ",0);
    }

    if (Socketed())
    {
        sprintf(buffers[0],"Socketed [%i Gems]",GemNum());
        strmcat(s,"\\par\\cf1 ",buffers[0],0);
    }

    if (Info->Description)
    {
        strmcat(s,"\\par\\cf0 ",Info->Description,0);
    }

    strcat(s,"\\par }");

    return s;
}
```

```
#include "JamellaD2E.h"

struct _ItemTree ItemTree[] = {
{ 1,"Weapons",1000 },
{ 2,"Regular",1100 },
{ 3,"Swords",1101 },
{ 3,"Axes",1102 },
{ 3,"Maces",1103 },
{ 3,"Javelins",1104 },
{ 3,"Spears",1105 },
{ 3,"Polearms",1106 },
{ 3,"Daggers",1107 },
{ 3,"Throwing",1108 },
{ 3,"Bows",1109 },
{ 3,"Crossbows",1110 },
{ 3,"Wands",1111 },
{ 3,"Scepters",1112 },
{ 3,"Staves",1113 },
{ 2,"Unique",1200 },
{ 3,"Swords",1201 },
{ 3,"Axes",1202 },
{ 3,"Maces",1203 },
{ 3,"Daggers",1204 },
{ 3,"Spears",1205 },
{ 3,"Polearms",1206 },
{ 3,"Bows",1207 },
{ 3,"Crossbows",1208 },
{ 3,"Wands",1209 },
{ 3,"Scepters",1210 },
{ 3,"Staves",1211 },
{ 2,"Exceptional",1300 },
{ 3,"Swords",1301 },
{ 3,"Axes",1302 },
{ 3,"Maces",1303 },
{ 3,"Javelins",1304 },
{ 3,"Spears",1305 },
{ 3,"Polearms",1306 },
{ 3,"Daggers",1307 },
{ 3,"Throwing",1308 },
{ 3,"Bows",1309 },
{ 3,"Crossbows",1310 },
{ 3,"Wands",1311 },
{ 3,"Scepters",1312 },
{ 3,"Staves",1313 },
{ 2,"Potions",1400 },
{ 1,"Helms",2000 },
{ 2,"Regular",2100 },
{ 2,"Unique",2200 },
{ 2,"Exceptional",2300 },
{ 1,"Armor",3000 },
{ 2,"Regular",3100 },
{ 2,"Unique",3200 },
{ 2,"Exceptional",3300 },
{ 1,"Shields",4000 },
{ 2,"Regular",4100 },
{ 2,"Unique",4200 },
{ 2,"Exceptional",4300 },
{ 1,"Gloves",5000 },
{ 2,"Regular",5100 },
{ 2,"Unique",5200 },
{ 2,"Exceptional",5300 },
{ 1,"Boots",6000 },
{ 2,"Regular",6100 },
{ 2,"Unique",6200 },
{ 2,"Exceptional",6300 },
{ 1,"Belts",7000 },
{ 2,"Regular",7100 },
{ 2,"Unique",7200 },
{ 2,"Exceptional",7300 },
{ 1,"Rings",8000 },
{ 2,"Magical",8100 },
{ 2,"Unique",8200 },
{ 1,"Amulets",9000 },
{ 2,"Magical",9100 },
{ 2,"Unique",9200 },
{ 1,"Accessories",10000 },
{ 2,"Potions",10100 },
{ 3,"Healing",10101 },
{ 3,"Mana",10102 },
```

```
{ 2,"Scrolls & Tomes",10200 },
{ 2,"Gems",10300 },
{ 3,"Chipped",10301 },
{ 3,"Flawed",10302 },
{ 3,"Regular",10303 },
{ 3,"Flawless",10304 },
{ 3,"Perfect",10305 },
{ 1,"Set Items",20000 },
{ 2,"Angelic Raiment",20001 },
{ 2,"Arcanna's Tricks",20002 },
{ 2,"Arctic Gear",20003 },
{ 2,"Berserker's Arsenal",20004 },
{ 2,"Cathan's Traps",20005 },
{ 2,"Civerb's Vestments",20006 },
{ 2,"Cleglaw's Brace",20007 },
{ 2,"Death's Disguise",20008 },
{ 2,"Hsaru's Defense",20009 },
{ 2,"Infernal Tools",20010 },
{ 2,"Iratha's Finery",20011 },
{ 2,"Isenhart's Armory",20012 },
{ 2,"Milabrega's Regalia",20013 },
{ 2,"Sigon's Complete Steel",20014 },
{ 2,"Tancred's Battlegear",20015 },
{ 2,"Vidala's Rig",20016 },
{ 1,"Quest Items",30000 },
{ 2,"Act I",30001 },
{ 2,"Act II",30002 },
{ 2,"Act III",30003 },
{ 2,"Act IV",30004 },
{ 1,"Special Items",40000 },
{ 1,"Useless Junk (but fun)",90000 },
{ 2,"Skeleton Collection",90100 },
};
int nItemTree = sizeof ItemTree / sizeof ItemTree[0];
```

```
// MagicDecoder.cpp from D2E
// Contains methods for the Class MagicDecoder

#include "JamellaD2E.h"

MagicDecoder::MagicDecoder(Item *IP)
{
    ZeroMemory(this,sizeof *this);
    I = IP;

    PrefixBuffer = new _MagicPreSuffix* [nMagicPrefixTable];
    SuffixBuffer = new _MagicPreSuffix* [nMagicSuffixTable];
    RarePrefixBuffer = new const _RarePreSuffix* [nRarePrefixTable];
    RareSuffixBuffer = new const _RarePreSuffix* [nRareSuffixTable];
}
MagicDecoder::~MagicDecoder()
{
    if (PrefixBuffer) delete [] PrefixBuffer;
    if (SuffixBuffer) delete [] SuffixBuffer;
    if (RarePrefixBuffer) delete [] RarePrefixBuffer;
    if (RareSuffixBuffer) delete [] RareSuffixBuffer;
}
// Buffer Builders
void MagicDecoder::BuildMagicBuffers()
{
    if (PrefixBuffer && SuffixBuffer)
        if (ItemBuffered == I->ItemCode() && ItemBufferedModLevel == I->MagicLevel())
            return;

    ItemBuffered = I->ItemCode();
    ItemBufferedModLevel = I->MagicLevel();

    _MagicPreSuffix *Buffer[128];

    // Build Prefix Buffer
    nPrefixBuffer = 0;
    for(int z=0;z<nMagicPrefixTable;z++)
    {
        if (MagicPrefixTable[z].ModLevel - 2 > I->MagicLevel()) continue;
        if ((MagicPrefixTable[z].MagicMask & I->Info->MagicMask) == 0) continue;

        Buffer[nPrefixBuffer++] = &MagicPrefixTable[z];
    }

    memcpy(PrefixBuffer,Buffer,sizeof Buffer[0] * nPrefixBuffer);

    // Build Suffix Buffer
    nSuffixBuffer = 0;
    for(z=0;z<nMagicSuffixTable;z++)
    {
        if (MagicSuffixTable[z].ModLevel - 2 > I->MagicLevel()) continue;
        if ((MagicSuffixTable[z].MagicMask & I->Info->MagicMask) == 0) continue;

        Buffer[nSuffixBuffer++] = &MagicSuffixTable[z];
    }

    memcpy(SuffixBuffer,Buffer,sizeof Buffer[0] * nSuffixBuffer);
}
void MagicDecoder::BuildRareBuffers()
{
    if (RarePrefixBuffer && RareSuffixBuffer)
        if (RareItemCodeBuffered == I->ItemCode())
            return;

    RareItemCodeBuffered = I->ItemCode();

    const _RarePreSuffix *Buffer[128];

    // Build Prefix Buffer
    nRarePrefixBuffer = 0;
    for(int z=0;z<nRarePrefixTable;z++)
    {
        if ((RarePrefixTable[z].RareMask & I->Info->RareMask) == 0) continue;
        Buffer[nRarePrefixBuffer++] = &RarePrefixTable[z];
    }

    memcpy(RarePrefixBuffer,Buffer,sizeof Buffer[0] * nRarePrefixBuffer);

    // Build Suffix Buffer
```

```
    nRareSuffixBuffer = 0;
    for(z=0;z<nRareSuffixTable;z++)
    {
        if ((RareSuffixTable[z].RareMask & I->Info->RareMask) == 0) continue;
        Buffer[nRareSuffixBuffer++] = &RareSuffixTable[z];
    }

    memcpy(RareSuffixBuffer,Buffer,sizeof Buffer[0] * nRareSuffixBuffer);
}

void MagicDecoder::DecodeCrude()
{
    RAND rnd;
    StartRandoms(I,&rnd);

    const char *CrudePrefixes[4] =
    { "Crude","Cracked","Damaged","Low Quality" };

    CrudePrefix = CrudePrefixes[Random(&rnd) % 4];
}
void MagicDecoder::DecodeMagical()
{
    RAND rnd;
    StartRandoms(I,&rnd);

    if (!I->Info) I->FindInfo();

    BuildMagicBuffers();

    memset(&MagicPrefixMag,0,sizeof MagicPrefixMag);
    memset(&MagicSuffixMag,0,sizeof MagicSuffixMag);

    // Got Prefix ?
    if ((Random(&rnd) % 2) == 1)
    {
        if (nPrefixBuffer == 0)
        {
            DecodeError = DE_MAGIC_PREFIX_MODULO_ZERO;
            return;
        }

        modMagicPrefix = nPrefixBuffer;

        int Index = modpickMagicPrefix = Random(&rnd) % nPrefixBuffer;

        MagicPrefix = PrefixBuffer[Index];

        int RndGets = MagicPrefix->nMod;
        for(int z=0;z<4;z++)
        {
            if (MagicPrefix->Mod[z].Code == 0) continue;

            int diff = MagicPrefix->Mod[z].Max - MagicPrefix->Mod[z].Min;

            if (diff > 0)
            {
                if (RndGets <= 0)
                {
                    DecodeError = DE_MAGIC_PREFIX_MODIFIER_VALMISSING;
                    return;
                }

                MagicPrefixMag[z] = Random(&rnd) % diff;
                RndGets--;
            }
        }
        for(;RndGets > 0;RndGets--)
            Random(&rnd);
    }

    // Get Suffix ?
    int GetSuffix = 0;
    GetSuffix = Random(&rnd) % 2;
    if (GetSuffix || !MagicPrefix)
    {
        if (nSuffixBuffer == 0)
        {
            DecodeError = DE_MAGIC_SUFFIX_MODULO_ZERO;
            return;
```

```
        }

        modMagicSuffix = nSuffixBuffer;

        int Index = modpickMagicSuffix = Random(&rnd) % nSuffixBuffer;

        MagicSuffix = SuffixBuffer[Index];

        int RndGets = MagicSuffix->nMod;
        for(int z=0;z<4;z++)
        {
            if (MagicSuffix->Mod[z].Code == 0) continue;

            int diff = MagicSuffix->Mod[z].Max - MagicSuffix->Mod[z].Min;

            if (diff > 0)
            {
                if (RndGets <= 0)
                {
                    DecodeError = DE_MAGIC_SUFFIX_MODIFIER_VALMISSING;
                    return;
                }

                MagicSuffixMag[z] = Random(&rnd) % diff;
                RndGets--;
            }
        }
        for(;RndGets > 0;RndGets--)
            Random(&rnd);
    }
}
void MagicDecoder::DecodeRare()
{
    RAND rnd;
    StartRandoms(I,&rnd);

    if (!I->Info) I->FindInfo();

    BuildMagicBuffers();
    BuildRareBuffers();

    { // Name Prefix

        if (nRarePrefixBuffer == 0)
        {
            DecodeError = DE_RARE_NAMEPREFIX_MODULO_ZERO;
            return;
        }

        int iRarePrefix = Random(&rnd) % nRarePrefixBuffer;
        RarePrefix = RarePrefixBuffer[iRarePrefix];

    } // Name Prefix

    { // Name Suffix

        if (nRareSuffixBuffer == 0)
        {
            DecodeError = DE_RARE_NAMESUFFIX_MODULO_ZERO;
            return;
        }

        int iRareSuffix = Random(&rnd) % nRareSuffixBuffer;
        RareSuffix = RareSuffixBuffer[iRareSuffix];

    } // Name Suffix

    { // Number of PreSuffixes

        nRareFix = (Random(&rnd) % 3) + 4;

    } // Number of PreSuffixes

    int nPrefixes = 0,
        nSuffixes = 0;

    if (nPrefixBuffer == 0 || nSuffixBuffer == 0)
    {
        DecodeError = DE_RARE_PRESUFFIX_MODULO_ZERO;
```

```
        return;
    }

    int Timeout = 0;

    { // PreSuffixes

        for(int n=0;n<nRareFix;n++)
        {
            // Select Pre or Suffix
            int PreSuf = Random(&rnd) % 2;

            // Kick random Value
            Random(&rnd);

            if ((PreSuf == 0 && nPrefixes < 3) || nSuffixes > 2)
            {
reselectprefix:
                int Index = Random(&rnd) % nPrefixBuffer;

                if (Timeout++ > DECODETIMEOUT)
                {
                    //DecodeError = DE_RARE_PRESUFFIX_NOPOSSIBLE;
                    return;
                }

                // Check Groups
                for(int m=n-1;m>=0;m--)
                {
                    if (!RareFix[m]) continue;
                    if (RareFix[m]->Text == 0) continue;
                    if (RareFix[m]->Group != PrefixBuffer[Index]->Group) continue;

                    // Kick random Value
                    Random(&rnd);

                    goto reselectprefix;
                }

                nPrefixes++;
                RareFix[n] = PrefixBuffer[Index];
                tRareFix[n] = PREFIX;
            }
            else
            {
reselectsuffix:
                int Index = Random(&rnd) % nSuffixBuffer;

                if (Timeout++ > DECODETIMEOUT)
                {
                    //DecodeError = DE_RARE_PRESUFFIX_NOPOSSIBLE;
                    return;
                }

                // Check Groups
                for(int m=n-1;m>=0;m--)
                {
                    if (!RareFix[m]) continue;
                    if (RareFix[m]->Text == 0) continue;
                    if (RareFix[m]->Group != SuffixBuffer[Index]->Group) continue;

                    // Kick random Value
                    Random(&rnd);

                    goto reselectsuffix;
                }

                nSuffixes++;
                RareFix[n] = SuffixBuffer[Index];
                tRareFix[n] = SUFFIX;
            }
        }

        // Get Magnitudes of Modifiers
        if (!Quick)
        for(n=0;n<nRareFix;n++)
        {
            if (!RareFix[n]) continue;
```

```
            int RndGets = RareFix[n]->nMod;
            for(int z=0;z<4;z++)
            {
                if (RareFix[n]->Mod[z].Code == 0) continue;

                int diff = RareFix[n]->Mod[z].Max - RareFix[n]->Mod[z].Min;

                if (diff > 0)
                {
                    if (RndGets <= 0)
                    {
                        DecodeError = DE_RARE_PRESUFFIX_MODIFIER_VALMISSING;
                        return;
                    }

                    RareFixMag[n][z] = Random(&rnd) % diff;
                    RndGets--;
                }
            }
            for(;RndGets > 0;RndGets--)
                Random(&rnd);
        }


    } // PreSuffixes
}
void MagicDecoder::DecodeUnique()
{
    if (I->UniqueCode() >= 0 && I->UniqueCode() < nUniqueItems*2)
    {
        UniqueItem = &UniqueItems[I->UniqueCode() / 2];
    }
    else
        DecodeError = DE_UNIQUE_ITEMCODE;
}
void MagicDecoder::DecodeSet()
{
    if (I->UniqueCode() >= 0 && I->UniqueCode() < nSetItems*2)
    {
        SetItem = &SetItems[I->UniqueCode() / 2];
        SetItemNum = -1;

        for(int z=0;z<SetItem->nItems;z++)
        {
            if (I->ItemCode() != SetItem->Item[z].Code && I->ItemCode() != SetItem->Item[z].IC ) continue;
            SetItemNum = z;
        }

        if (SetItemNum < 0)
            DecodeError = DE_SET_ITEMCODE;
    }
    else
        DecodeError = DE_SET_ITEMCODE;
}
void MagicDecoder::CollectModifiers()
{
    nMods = 0;
    memset(Mod,0,sizeof Mod);

    if (DecodeError) return;

    switch(I->Quality())
    {
    default:
        break;
    case MAGICITEM:
        {
            if (MagicPrefix)
            {
                for(int z=0;z<3;z++)
                {
                    if (!MagicPrefix->Mod[z].Code) continue;
                    Mod[nMods].Code = MagicPrefix->Mod[z].Code;
                    Mod[nMods++].Mag = MagicPrefix->Mod[z].Min + MagicPrefixMag[z];
                }
            }
            if (MagicSuffix)
            {
                for(int z=0;z<3;z++)
```

```
                {
                    if (!MagicSuffix->Mod[z].Code) continue;
                    Mod[nMods].Code = MagicSuffix->Mod[z].Code;
                    Mod[nMods++].Mag = MagicSuffix->Mod[z].Min + MagicSuffixMag[z];
                }
            }
        }
        break;
    case SETITEM:
        {
            for(int z = SetItemNum * 2;z < SetItemNum * 2 + 2;z++)
            {
                if (SetItem->Mod[z].Code == 0) continue;

                Mod[nMods].Code = SetItem->Mod[z].Code;
                Mod[nMods++].Mag = SetItem->Mod[z].Min;
            }
        }
        break;
    case RAREITEM:
        {
            for(int z = 0;z < 6;z++)
            {
                if (!RareFix[z]) continue;

                for(int y=0;y<3;y++)
                {
                    if (!RareFix[z]->Mod[y].Code) continue;

                    Mod[nMods].Code = RareFix[z]->Mod[y].Code;
                    Mod[nMods++].Mag = RareFix[z]->Mod[y].Min + RareFixMag[z][y];
                }
            }
        }
        break;
    case UNIQUEITEM:
        {
            for(int z=0;z<7;z++)
            {
                if (UniqueItem->Mod[z].Code == 0) continue;

                Mod[nMods].Code = UniqueItem->Mod[z].Code;
                Mod[nMods++].Mag = UniqueItem->Mod[z].Min;
            }
        }
        break;
    }

}
bool MagicDecoder::Decode()
{
    Quick = false;
    I->Decoded = false;

    DecodeError = 0;

    ZeroMemory(&ZeroMemoryStart,(&ZeroMemoryEnd - &ZeroMemoryStart));

    switch(I->Quality())
    {
    case CRUDEITEM:
        DecodeCrude();
        break;
    case MAGICITEM:
        DecodeMagical();
        break;
    case SETITEM:
        DecodeSet();
        break;
    case RAREITEM:
        DecodeRare();
        break;
    case UNIQUEITEM:
        DecodeUnique();
        break;
    }

    CollectModifiers();
    I->Decoded = true;
```

```
    return I->Decoded;
}
bool MagicDecoder::QuickDecode()
{
    Quick = true;
    bool ODecoded = I->Decoded;
    I->Decoded = false;

    DecodeError = 0;

    ZeroMemory(&ZeroMemoryStart,(&ZeroMemoryEnd - &ZeroMemoryStart));

    switch(I->Quality())
    {
    case CRUDEITEM:
        DecodeCrude();
        break;
    case MAGICITEM:
        DecodeMagical();
        break;
    case SETITEM:
        DecodeSet();
        break;
    case RAREITEM:
        DecodeRare();
        break;
    case UNIQUEITEM:
        DecodeUnique();
        break;
    }

    I->Decoded = ODecoded;
    return true;
}

char MagicDecoder::NameTmp[256];
const char* MagicDecoder::Name()
{
    char *s = NameTmp;
    *s = 0;
    if (!I->Info) I->FindInfo();

    if (!I->Decoded) Decode();

    switch(I->Quality())
    {
    default:
        strcat(s,I->Info->ItemName);
        break;
    case CRUDEITEM:
        strmcat(s,CrudePrefix," ",I->Info->ItemName,0);
        break;
    case SUPERIORITEM:
        strmcat(s,"Superior ",I->Info->ItemName,0);
        break;
    case MAGICITEM:
        if (MagicPrefix)
        {
            strcat(s,MagicPrefix->Text);
            strcat(s," ");
        }

        strcat(s,I->Info->ItemName);

        if (MagicSuffix)
        {
            strcat(s," ");
            strcat(s,MagicSuffix->Text);
        }
        break;
    case RAREITEM:
        if (RarePrefix)
            strcat(s,RarePrefix->Text);

        strcat(s," ");

        if (RareSuffix)
            strcat(s,RareSuffix->Text);
        break;
```

```
    case SETITEM:
        strcat(s,I->Info->ItemName);
        break;
    case UNIQUEITEM:
        strcat(s,I->Info->ItemName);
        break;
    }

    return s;
}
char MagicDecoder::AttrTmp[2048];
const char* MagicDecoder::RichAttributes()
{
    *AttrTmp = 0;

    if (DecodeError)
    {
        strcat(AttrTmp,"\\par\\cf6 ");
        strcat(AttrTmp,DecodeErrorString(DecodeError));
        return AttrTmp;
    }

    switch(I->Quality())
    {
    default:
        {
            return "\\par\\cf6 No Attributes!";
        }
    case USUALITEM:
        {
            if (I->Socketed()) {

                for(Item *G = I->Gems;G != 0;G = G->Next())
                {
                    if (!G->FindGemInfo()) break;

                    switch(I->Info->GemClass)
                    {
                    case 'W':
                        {
                            for(int z=0;z<3;z++)
                            {
                                if (!G->GInfo->WeaponMod[z].Code) continue;

                                strcat(AttrTmp,"\\par ");
                                sprintf(buffer,GetEffect(G->GInfo->WeaponMod[z].Code),G->GInfo->WeaponMod[z].Min);
                                strcat(AttrTmp,buffer);
                            }
                        }
                        break;
                    case 'H':
                        {
                            for(int z=0;z<3;z++)
                            {
                                if (!G->GInfo->HelmMod[z].Code) continue;

                                strcat(AttrTmp,"\\par ");
                                sprintf(buffer,GetEffect(G->GInfo->HelmMod[z].Code),G->GInfo->HelmMod[z].Min);
                                strcat(AttrTmp,buffer);
                            }
                        }
                        break;
                    case 'S':
                        {
                            for(int z=0;z<3;z++)
                            {
                                if (!G->GInfo->ShieldMod[z].Code) continue;

                                strcat(AttrTmp,"\\par ");
                                sprintf(buffer,GetEffect(G->GInfo->ShieldMod[z].Code),G->GInfo->ShieldMod[z].Min);
                                strcat(AttrTmp,buffer);
                            }
                        }
                        break;
                    }
                }
                break;
            }
            else {
```

```
                return "\\par\\cf6 Non-Socketed Item has no attributes!";
            }
        }
    case MAGICITEM:
        {
            if (MagicPrefix)
            {
                for(int z=0;z<4;z++)
                {
                    if (MagicPrefix->Mod[z].Code == 0) continue;

                    strcat(AttrTmp,"\\par ");
                    sprintf(buffer,GetEffect(MagicPrefix->Mod[z].Code),MagicPrefix->Mod[z].Min + MagicPrefixMag[z]);
                    strcat(AttrTmp,buffer);
#if SHOWMAGICCODE == 1
                    strmcat(AttrTmp,"[",CodeStringRev(MagicPrefix->Mod[z].Code),"]",0);
#endif
                }
            }
            if (MagicSuffix)
            {
                for(int z=0;z<4;z++)
                {
                    if (MagicSuffix->Mod[z].Code == 0) continue;

                    strcat(AttrTmp,"\\par ");
                    sprintf(buffer,GetEffect(MagicSuffix->Mod[z].Code),MagicSuffix->Mod[z].Min + MagicSuffixMag[z]);
                    strcat(AttrTmp,buffer);
#if SHOWMAGICCODE == 1
                    strmcat(AttrTmp,"[",CodeStringRev(MagicSuffix->Mod[z].Code),"]",0);
#endif
                }
            }
        }
        break;
    case RAREITEM:
        {
            for(int n=0;n<nRareFix;n++)
            {
                for(int z=0;z<4;z++)
                {
                    if (!RareFix[n]) continue;
                    if (RareFix[n]->Mod[z].Code == 0) continue;

                    strcat(AttrTmp,"\\par ");
                    sprintf(buffer,GetEffect(RareFix[n]->Mod[z].Code),RareFix[n]->Mod[z].Min + RareFixMag[n][z]);
                    strcat(AttrTmp,buffer);
#if SHOWMAGICCODE == 1
                    strmcat(AttrTmp,"[",RareFix[n]->Text,"]",0);
#endif
                }
            }
        }
        break;
    case SETITEM:
        {
            if (!SetItem || SetItemNum < 0)
            {
                return "Set Item decoding error!";
            }
            for(int z = SetItemNum * 2;z < SetItemNum * 2 + 2;z++)
            {
                if (SetItem->Mod[z].Code == 0) continue;

                int y = z - SetItemNum * 2;

                strcat(AttrTmp,"\\par ");
                sprintf(buffer,GetEffect(SetItem->Mod[z].Code),SetItem->Mod[z].Min + UniqueSetMag[y]);
                strcat(AttrTmp,buffer);
#if SHOWMAGICCODE == 1
                strmcat(AttrTmp,"[",CodeStringRev(SetItem->Mod[z].Code),"]",0);
#endif
            }
        }
        break;
    case UNIQUEITEM:
        {
            if (!UniqueItem)
            {
```

```
                return "Unique Item decoding error!";
            }
            for(int z=0;z<7;z++)
            {
                if (UniqueItem->Mod[z].Code == 0) continue;

                strcat(AttrTmp,"\\par ");
                sprintf(buffer,GetEffect(UniqueItem->Mod[z].Code),UniqueItem->Mod[z].Min + UniqueSetMag[z]);
                strcat(AttrTmp,buffer);
#if SHOWMAGICCODE == 1
                strmcat(AttrTmp,"[",CodeStringRev(UniqueItem->Mod[z].Code),"]",0);
#endif
            }
        }
        break;
    }
    return AttrTmp;
}
const char* DecodeErrorString(int Error)
{
    switch(Error)
    {
    default:
        return "No Error";

    case DE_MAGIC_PREFIX_MODULO_ZERO:
        return "Error decoding Magical Prefix! (Modulo is zero)";
    case DE_MAGIC_PREFIX_MODIFIER_VALMISSING:
        return "Error decoding Magical Prefix! (Modifier Specifier missing)";
    case DE_MAGIC_SUFFIX_MODULO_ZERO:
        return "Error decoding Magical Suffix! (Modulo is zero)";
    case DE_MAGIC_SUFFIX_MODIFIER_VALMISSING:
        return "Error decoding Magical Suffix! (Modifier Specifier missing)";

    case DE_RARE_NAMEPREFIX_MODULO_ZERO:
        return "Error decoding Rare Name Prefix! (Modulo is zero)";
    case DE_RARE_NAMESUFFIX_MODULO_ZERO:
        return "Error decoding Rare Name Suffix! (Modulo is zero)";
    case DE_RARE_PRESUFFIX_MODULO_ZERO:
        return "Error decoding Rare Item Attributes! (Buffer Modulo is zero)";
    case DE_RARE_PRESUFFIX_NOPOSSIBLE:
        return "Error decoding Rare Item Attributes! (No more Modifiers available)";
    case DE_RARE_PRESUFFIX_MODIFIER_VALMISSING:
        return "Error decoding Rare Item Attributes! (Modifier Specifier missing)";

    case DE_UNIQUE_ITEMCODE:
        return "Error decoding Unique Item Code! (Invalid Code)";
    case DE_SET_ITEMCODE:
        return "Error decoding Set Item Code! (Invalid Code)";
    }
}

// Double Word History Class

DWHistory::DWHistory()
{
    Top = Bottom = Ptr = 0;
}
DWHistory::~DWHistory()
{
}

bool DWHistory::isNext()
{
    return !(Top == Ptr);
}
bool DWHistory::isBack()
{
    return !(Ptr == Bottom);
}
void DWHistory::StepAdd(Item *I)
{
    ML[Ptr] = I->MagicLevel();
    DWA[Ptr] = I->DWA();
    DWB[Ptr] = I->DWB();

    if (++Ptr >= HISTORYSTEPS)
        Ptr = 0;
```

```
    Top = Ptr;

    if (Top == Bottom)
        if (++Bottom >= HISTORYSTEPS)
            Bottom = 0;
}
void DWHistory::StepBack(Item *I)
{
    if (Ptr == Bottom) return;

    ML[Ptr] = I->MagicLevel();
    DWA[Ptr] = I->DWA();
    DWB[Ptr] = I->DWB();

    Ptr--;
    if (Ptr < 0)
        Ptr = HISTORYSTEPS-1;

    I->SetMagicLevel(ML[Ptr]);
    I->SetDWA(DWA[Ptr]);
    I->SetDWB(DWB[Ptr]);

    I->Info = 0;
    I->Decoded = 0;
}
void DWHistory::StepNext(Item *I)
{
    if (Ptr == Top) return;

    if (++Ptr > HISTORYSTEPS)
        Ptr = 0;

    I->SetMagicLevel(ML[Ptr]);
    I->SetDWA(DWA[Ptr]);
    I->SetDWB(DWB[Ptr]);

    I->Info = 0;
    I->Decoded = 0;
}
```

```
#include "JamellaD2E.h"

struct _MagicPreSuffix MagicPrefixTable[] =
{
{ 0,"Resiliant",1,0,101,0,'ar00',1,1,0,0,0,0,0,0,0,0,8192,0,16,"Not Available" },
{ 1,"Sturdy",4,3,102,1,'ar02',20,30,0,0,0,0,0,0,0,0,1797,0,16,"+20-30% to Defense" },
{ 2,"Strong",9,6,102,1,'ar02',31,40,0,0,0,0,0,0,0,0,1797,0,16,"+31-40% to Defense" },
{ 3,"Glorious",19,14,102,1,'ar02',41,50,0,0,0,0,0,0,0,0,1797,0,16,"+41-50% to Defense" },
{ 4,"Blessed",25,18,102,1,'ar02',51,65,0,0,0,0,0,0,0,0,1797,0,16,"+51-60% to Defense" },
{ 5,"Saintly",31,23,102,1,'ar02',66,80,0,0,0,0,0,0,0,0,1797,1,16,"+61-80% to Defense" },
{ 6,"Holy",36,27,102,1,'ar02',81,100,0,0,0,0,0,0,0,0,1797,1,16,"+81-100% to Defense" },
{ 7,"Devious",7,5,103,0,'ar03',1,1,0,0,0,0,0,0,0,0,4,1,5,"Magic Damage Reduced by 1" },
{ 8,"Fortified",14,10,103,0,'ar03',2,2,0,0,0,0,0,0,0,0,4,1,5,"Magic Damage Reduced by 2" },
{ 9,"Urgent",9,6,104,0,'at02',10,10,'tm00',150,150,0,0,0,0,0,0,0,0,3,"Not Available" },
{ 10,"Fleet",6,4,104,0,'at03',20,20,'tm00',150,150,0,0,0,0,0,0,0,0,3,"Not Available" },
{ 11,"Muscular",12,9,105,0,'dm01',1,1,0,0,0,0,0,0,0,0,8192,0,3,"Not Available" },
{ 12,"Jagged",1,0,106,2,'dm02',10,20,0,0,0,0,0,0,0,0,242,0,16,"+20-30% to Damage" },
{ 13,"Deadly",5,3,106,2,'dm02',21,30,0,0,0,0,0,0,0,0,8434,0,16,"+21-30% to Damage" },
{ 14,"Vicious",8,6,106,2,'dm02',31,40,0,0,0,0,0,0,0,0,242,0,16,"+31-40% to Damage" },
{ 15,"Brutal",14,10,106,2,'dm02',41,50,0,0,0,0,0,0,0,0,242,0,16,"+41-50% to Damage" },
{ 16,"Massive",20,15,106,2,'dm02',51,65,0,0,0,0,0,0,0,0,242,1,16,"+51-65% to Damage" },
{ 17,"Savage",26,19,106,2,'dm02',66,80,0,0,0,0,0,0,0,0,146,1,16,"+66-80% to Damage" },
{ 18,"Merciless",32,24,106,2,'dm02',81,100,0,0,0,0,0,0,0,0,146,1,16,"+81-100% to Damage" },
{ 19,"Vulpine",9,6,107,0,'dm03',10,10,0,0,0,0,0,0,0,0,4100,1,6,"Damage Generates 10% Mana" },
{ 20,"Swift",4,3,108,0,'dx00',1,1,0,0,0,0,0,0,0,0,16,"Not Available" },
{ 21,"Artful",3,2,108,0,'dx00',20,20,'tm00',150,150,0,0,0,0,0,0,0,0,3,"Not Available" },
{ 22,"Skillful",23,17,108,0,'dx00',50,50,'tm00',150,150,0,0,0,0,0,0,0,0,3,"Not Available" },
{ 23,"Adroit",26,19,108,0,'dx00',100,100,'tm00',150,150,0,0,0,0,0,0,0,0,3,"Not Available" },
{ 24,"Tireless",14,10,109,0,'hl08',50,50,0,0,0,0,0,0,0,0,256,0,2,"+ Stamina Regeneration" },
{ 25,"Rugged",14,10,110,1,'hl12',5,10,0,0,0,0,0,0,0,0,16128,0,2,"+ 5-10 to Max Stamina" },
{ 26,"Bronze",1,0,111,1,'ht00',10,20,0,0,0,0,0,0,0,0,6898,0,15,"+10-20 to Attack Rating" },
{ 27,"Iron",4,3,111,1,'ht00',21,40,0,0,0,0,0,0,0,0,2290,0,15,"+21-40 to Attack Rating" },
{ 28,"Steel",8,6,111,1,'ht00',41,60,0,0,0,0,0,0,0,0,2290,0,15,"+41-60 to Attack Rating" },
{ 29,"Silver",12,9,111,1,'ht00',61,80,0,0,0,0,0,0,0,0,2290,0,15,"+61-80 to Attack Rating" },
{ 30,"Silver",17,12,111,1,'ht00',40,80,0,0,0,0,0,0,0,0,0,0,15,"Not Available" },
{ 31,"Gold",17,12,111,1,'ht00',81,100,0,0,0,0,0,0,0,0,242,1,15,"+81-100 to Attack Rating" },
{ 32,"Platinum",22,16,111,1,'ht00',101,120,0,0,0,0,0,0,0,0,2290,1,15,"+101-120 to Attack Rating" },
{ 33,"Meteoric",27,20,111,1,'ht00',121,150,0,0,0,0,0,0,0,0,2194,1,15,"+121-150 to Attack Rating" },
{ 34,"Sharp",5,3,112,3,'ht00',10,20,'dm02',10,20,0,0,0,0,0,0,242,0,16,"+10-20 to AR & +10-20 to Damage" },
{ 35,"Fine",9,6,112,3,'ht00',21,40,'dm02',21,30,0,0,0,0,0,0,242,0,16,"+21-40 to AR & +21-30 to Damage" },
{ 36,"Warrior's",15,11,112,3,'ht00',41,60,'dm02',31,40,0,0,0,0,0,0,242,0,16,"+41-60 to AR & +31-40 to Damage" },
{ 37,"Soldier's",21,15,112,3,'ht00',61,80,'dm02',41,50,0,0,0,0,0,0,242,0,16,"+61-80 to AR & +41-50 to Damage" },
{ 38,"Knight's",25,18,112,3,'ht00',81,100,'dm02',51,65,0,0,0,0,0,0,242,1,16,"+81-100 to AR & +51-65 to Damage" },
{ 39,"Lord's",30,22,112,3,'ht00',101,120,'dm02',66,80,0,0,0,0,0,0,146,1,16,"+101-120 to AR & +66-80 to Damage" },
{ 40,"King's",35,26,112,3,'ht00',121,150,'dm02',81,100,0,0,0,0,0,0,146,1,16,"+121-150 to AR & +81-100 to Damage" },
{ 41,"Howling",16,12,113,0,'hw00',1,1,0,0,0,0,0,0,0,0,114,1,19,"Make Monsters Flee" },
{ 42,"Fortuitous",5,3,114,1,'ib01',10,15,0,0,0,0,0,0,0,0,14336,0,15,"10-15% Better Chance of Getting Magic Items" },
{ 43,"Brilliant",4,3,115,0,'mm00',10,10,0,0,0,0,0,0,0,0,0,0,16,"Not Available" },
{ 44,"Omniscient",1,0,115,0,'in00',10,10,'tm00',150,150,0,0,0,0,0,0,16384,0,3,"Not Available" },
{ 45,"Sage",8,6,115,0,'in00',20,20,'tm00',150,150,0,0,0,0,0,0,16384,1,3,"Not Available" },
{ 46,"Shrewd",23,17,115,0,'in00',50,50,'tm00',150,150,0,0,0,0,0,0,16384,1,3,"Not Available" },
{ 47,"Vivid",27,20,115,0,'in00',100,100,'tm00',150,150,0,0,0,0,0,0,16384,1,3,"Not Available" },
{ 48,"Glimmering",1,0,116,0,'lt00',1,1,0,0,0,0,0,0,0,0,32613,0,13,"+1 to Light Radius" },
{ 49,"Glowing",6,4,116,0,'lt00',2,2,0,0,0,0,0,0,0,0,32613,1,13,"+2 to Light Radius" },
{ 50,"Bright",9,6,117,0,'mh01',50,50,'tm00',150,150,0,0,0,0,0,0,0,0,3,"Not Available" },
{ 51,"Solar",20,15,117,0,'mh01',100,100,'tm00',150,150,0,0,0,0,0,0,0,0,3,"Not Available" },
{ 52,"Lizard's",3,2,118,1,'mm00',1,5,0,0,0,0,0,0,0,0,31861,0,6,"+1-5 to Mana" },
{ 53,"Forceful",13,-1,118,0,'mm00',5,5,0,0,0,0,0,0,0,0,24576,0,6,"Not Available" },
{ 54,"Snake's",6,4,118,1,'mm00',5,10,0,0,0,0,0,0,0,0,23668,0,6,"+5-10 to Mana" },
{ 55,"Serpent's",14,10,118,1,'mm00',11,20,0,0,0,0,0,0,0,0,23668,0,6,"+11-20 to Mana" },
{ 56,"Serpent's",37,27,118,1,'mm00',11,20,0,0,0,0,0,0,0,0,771,0,6,"+11-20 to Mana" },
{ 57,"Drake's",20,15,118,1,'mm00',21,30,0,0,0,0,0,0,0,0,23664,1,6,"+21-30 to Mana" },
{ 58,"Dragon's",24,18,118,1,'mm00',31,40,0,0,0,0,0,0,0,0,22640,1,6,"+31-40 to Mana" },
{ 59,"Dragon's",52,39,118,1,'mm00',31,40,0,0,0,0,0,0,0,0,769,1,6,"+31-40 to Mana" },
{ 60,"Wyrm's",30,22,118,1,'mm00',41,60,0,0,0,0,0,0,0,0,22640,1,6,"+41-50 to Mana" },
{ 61,"Dazzling",9,6,119,0,'mm01',50,50,'tm00',150,150,0,0,0,0,0,0,16384,0,3,"Not Available" },
{ 62,"Facinating",20,15,119,0,'mm01',100,100,'tm00',150,150,0,0,0,0,0,0,16384,0,3,"Not Available" },
{ 63,"Prismatic",27,20,120,4,'rf00',15,25,'rl00',15,25,'rc00',15,25,'rp00',15,25,4096,1,17,"+15-25% to all Resistances"
},
{ 64,"Prismatic",62,46,120,0,'ra00',15,15,0,0,0,0,0,0,0,0,2048,1,17,"+15% to all Resistances" },
{ 65,"Azure",5,3,121,1,'rc00',5,10,0,0,0,0,0,0,0,0,32757,0,4,"+5-10% to Cold Resist" },
{ 66,"Lapis",12,9,121,1,'rc00',11,20,0,0,0,0,0,0,0,0,24437,0,4,"+11-20% to Cold Resist" },
{ 67,"Lapis",35,26,121,1,'rc00',11,20,0,0,0,0,0,0,0,0,642,0,4,"+11-20% to Cold Resist" },
{ 68,"Cobalt",18,13,121,1,'rc00',21,30,0,0,0,0,0,0,0,0,23925,0,4,"+21-30% to Cold Resist" },
{ 69,"Cobalt",55,41,121,1,'rc00',21,30,0,0,0,0,0,0,0,0,642,0,4,"+21-30% to Cold Resist" },
{ 70,"Indigo",3,2,121,0,'rc00',25,25,'tm00',150,150,0,0,0,0,0,0,16384,0,3,"Not Available" },
{ 71,"Sapphire",25,18,121,1,'rc00',31,50,0,0,0,0,0,0,0,0,31200,1,4,"+31-50% to Cold Resist" },
{ 72,"Cerulean",7,5,121,0,'rc00',50,50,'tm00',150,150,0,0,0,0,0,0,16384,1,3,"Not Available" },
```

```
{ 73,"Red",7,5,122,0,'rf00',2,2,0,0,0,0,0,0,0,0,24576,0,7,"Not Available" },
{ 74,"Crimson",5,3,122,1,'rf00',5,10,0,0,0,0,0,0,0,0,24565,0,7,"+5-10% to Fire Resist" },
{ 75,"Burgundy",12,9,122,1,'rf00',11,20,0,0,0,0,0,0,0,0,24437,0,7,"+11-20% to Fire Resist" },
{ 76,"Burgundy",35,26,122,1,'rf00',11,20,0,0,0,0,0,0,0,0,130,0,7,"+11-20% to Fire Resist" },
{ 77,"Garnet",18,13,122,1,'rf00',21,30,0,0,0,0,0,0,0,0,23925,0,7,"+21-30% to Fire Resist" },
{ 78,"Garnet",55,41,122,3,'rf00',21,30,0,0,0,0,0,0,0,0,642,0,7,"+21-30% to Fire Resist" },
{ 79,"Russet",3,2,122,0,'rf00',25,25,'tm00',150,150,0,0,0,0,0,0,16384,0,3,"Not Available" },
{ 80,"Ruby",25,18,122,1,'rf00',31,50,0,0,0,0,0,0,0,0,22880,1,7,"+31-50% to Fire Resist" },
{ 81,"Vermilion",7,5,122,0,'rf00',50,50,'tm00',150,150,0,0,0,0,0,0,16384,1,3,"Not Available" },
{ 82,"Orange",7,5,123,0,'rl00',2,2,0,0,0,0,0,0,0,0,24576,0,13,"Not Available" },
{ 83,"Ocher",5,3,123,1,'rl00',5,10,0,0,0,0,0,0,0,0,24565,0,13,"+5-10% to Lightning Resist" },
{ 84,"Tangerine",12,9,123,1,'rl00',11,20,0,0,0,0,0,0,0,0,24437,0,13,"+11-20% to Lightning Resist" },
{ 85,"Tangerine",35,26,123,1,'rl00',11,20,0,0,0,0,0,0,0,0,130,0,13,"+11-20% to Lightning Resist" },
{ 86,"Coral",18,13,123,1,'rl00',21,30,0,0,0,0,0,0,0,0,23925,0,13,"+21-30% to Lightning Resist" },
{ 87,"Coral",55,41,123,1,'rl00',21,30,0,0,0,0,0,0,0,0,642,0,13,"+21-30% to Lightning Resist" },
{ 88,"Crackling",3,2,123,0,'rl00',25,25,'tm00',150,150,0,0,0,0,0,0,16384,0,3,"Not Available" },
{ 89,"Amber",25,18,123,1,'rl00',31,50,0,0,0,0,0,0,0,0,23905,1,13,"+31-50% to Lightning Resist" },
{ 90,"Forked",21,15,123,0,'rl00',50,50,'tm00',150,150,0,0,0,0,0,0,16384,1,3,"Not Available" },
{ 91,"Green",7,5,124,0,'rp00',2,2,0,0,0,0,0,0,0,0,8192,0,10,"Not Available" },
{ 92,"Beryl",5,3,124,1,'rp00',5,10,0,0,0,0,0,0,0,0,8149,0,10,"+5-10% to Poison Resist" },
{ 93,"Jade",12,9,124,1,'rp00',11,20,0,0,0,0,0,0,0,0,8021,0,10,"+11-20% to Poison Resist" },
{ 94,"Jade",35,26,124,1,'rp00',11,20,0,0,0,0,0,0,0,0,2210,0,10,"+11-20% to Poison Resist" },
{ 95,"Viridian",18,13,124,1,'rp00',21,30,0,0,0,0,0,0,0,0,6229,0,10,"+21-30% to Poison Resist" },
{ 96,"Viridian",55,41,124,1,'rp00',21,30,0,0,0,0,0,0,0,0,1954,0,10,"+21-30% to Poison Resist" },
{ 97,"Vital",11,8,124,0,'rp00',25,25,'tm00',150,150,0,0,0,0,0,0,0,3,"Not Available" },
{ 98,"Emerald",25,18,124,1,'rp00',31,50,0,0,0,0,0,0,0,0,6160,1,10,"+31-50% to Poison Resist" },
{ 99,"Enduring",14,10,124,0,'rp00',50,50,'tm00',150,150,0,0,0,0,0,0,0,3,"Not Available" },
{ 100,"Fletcher's",30,22,125,0,'sk00',1,1,0,0,0,0,0,0,0,0,4224,0,12,"+1 to Amazon Skill Levels" },
{ 101,"Archer's",40,30,125,0,'sk00',2,2,0,0,0,0,0,0,0,0,128,1,12,"+2 to Amazon Skill Levels" },
{ 102,"Archer's",90,67,125,0,'sk00',2,2,0,0,0,0,0,0,0,0,4096,1,12,"+2 to Amazon Skill Levels" },
{ 103,"Monk's",30,22,126,0,'sk01',1,1,0,0,0,0,0,0,0,0,4112,0,12,"+1 to Paladin Skill Levels" },
{ 104,"Priest's",40,30,126,0,'sk01',2,2,0,0,0,0,0,0,0,0,16,1,12,"+2 to Paladin Skill Levels" },
{ 105,"Priest's",90,67,126,0,'sk01',2,2,0,0,0,0,0,0,0,0,4096,1,12,"+2 to Paladin Skill Levels" },
{ 106,"Summoner's",30,22,127,0,'sk02',1,1,0,0,0,0,0,0,0,0,4128,0,12,"+1 to Necromancer Skill Levels" },
{ 107,"Necromancer's",40,30,127,0,'sk02',2,2,0,0,0,0,0,0,0,0,32,1,12,"+2 to Necromancer Skill Levels" },
{ 108,"Necromancer's",90,67,127,0,'sk02',2,2,0,0,0,0,0,0,0,0,4096,1,12,"+2 to Necromancer Skill Levels" },
{ 109,"Angel's",30,22,128,0,'sk03',1,1,0,0,0,0,0,0,0,0,20544,0,12,"+1 to Sorceress Skill Levels" },
{ 110,"Arch-Angel's",40,30,128,0,'sk03',2,2,0,0,0,0,0,0,0,0,64,1,12,"+2 to Sorceress Skill Levels" },
{ 111,"Arch-Angel's",90,67,128,0,'sk03',2,2,0,0,0,0,0,0,0,0,20480,1,12,"+2 to Sorceress Skill Levels" },
{ 112,"Slayer's",30,22,129,0,'sk04',1,1,0,0,0,0,0,0,0,0,4098,0,12,"+1 to Barbarian Skill Levels" },
{ 113,"Berserker's",40,30,129,0,'sk04',2,2,0,0,0,0,0,0,0,0,2,1,12,"+2 to Barbarian Skill Levels" },
{ 114,"Berserker's",90,67,129,0,'sk04',2,2,0,0,0,0,0,0,0,0,4096,1,12,"+2 to Barbarian Skill Levels" },
{ 115,"unused",9,6,130,0,'sk05',1,1,0,0,0,0,0,0,0,0,0,0,"Not Available" },
{ 116,"Kicking",3,2,131,1,'sp04',1,3,0,0,0,0,0,0,0,0,0,3,"Not Available" },
{ 117,"Triumphant",3,2,132,0,'sp05',1,1,0,0,0,0,0,0,0,0,2162,0,6,"+1 to Life per Kill" },
{ 118,"Mighty",4,3,133,0,'st00',1,1,0,0,0,0,0,0,0,0,0,16,"Not Available" },
{ 119,"Energizing",3,2,133,0,'st00',20,20,'tm00',150,150,0,0,0,0,0,0,0,3,"Not Available" },
{ 120,"Strengthening",23,17,133,0,'st00',50,50,'tm00',150,150,0,0,0,0,0,0,0,0,3,"Not Available" },
{ 121,"Empowering",26,19,133,0,'st00',100,100,'tm00',150,150,0,0,0,0,0,0,0,0,3,"Not Available" },
{ 122,"Brisk",5,3,134,0,'ve01',10,10,'tm00',150,150,0,0,0,0,0,0,0,0,3,"Not Available" },
{ 123,"Swift",19,14,134,0,'ve02',20,20,'tm00',150,150,0,0,0,0,0,0,0,0,3,"Not Available" },
{ 124,"Tough",4,3,135,0,'mh00',10,10,0,0,0,0,0,0,0,0,0,0,16,"Not Available" },
{ 125,"Sturdy",11,8,135,0,'mh00',20,20,'tm00',150,150,0,0,0,0,0,0,0,0,3,"Not Available" },
{ 126,"Hardy",23,17,135,0,'mh00',50,50,'tm00',150,150,0,0,0,0,0,0,0,0,3,"Not Available" },
{ 127,"Robust",27,20,135,0,'mh00',100,100,'tm00',150,150,0,0,0,0,0,0,0,0,3,"Not Available" },
};
struct _MagicPreSuffix MagicSuffixTable[] =
{
{ 0,"of Health",7,5,1,0,'ar01',1,1,0,0,0,0,0,0,0,0,6149,0,5,"Damage Reduction by 1" },
{ 1,"of Protection",18,13,1,0,'ar01',2,2,0,0,0,0,0,0,0,0,6144,0,5,"Damage Reduction by 2" },
{ 2,"of Absorption",26,19,1,0,'ar01',3,3,0,0,0,0,0,0,0,0,4096,0,5,"Damage Reduction by 3" },
{ 3,"of Life",35,26,1,0,'ar01',4,4,0,0,0,0,0,0,0,0,12288,1,5,"Damage Reduction by 4" },
{ 4,"of Life",35,26,1,0,'ar01',4,4,0,0,0,0,0,0,0,0,1,5,"Damage Reduction by 4" },
{ 5,"of Warding",7,5,2,0,'ar03',1,1,0,0,0,0,0,0,0,0,14341,0,5,"Magic Damage Reduction by 1" },
{ 6,"of the Sentinel",18,13,2,0,'ar03',2,2,0,0,0,0,0,0,0,0,6144,0,5,"Magic Damage Reduction by 2" },
{ 7,"of Guarding",26,19,2,0,'ar03',3,3,0,0,0,0,0,0,0,0,4096,0,5,"Magic Damage Reduction by 3" },
{ 8,"of Negation",35,26,2,0,'ar03',4,4,0,0,0,0,0,0,0,0,4096,0,5,"Magic Damage Reduction by 4" },
{ 9,"unused",35,26,2,1,'ar03',5,6,0,0,0,0,0,0,0,0,1,5,"Not Available" },
{ 10,"of Piercing",25,18,3,0,'ar04',1,1,0,0,0,0,0,0,0,0,112,0,19,"Attack Ignores Target's Defense" },
{ 11,"of Bashing",16,12,4,1,'ar05',-25,-40,0,0,0,0,0,0,0,0,18,0,19,"-25-40 of Monster's Defense" },
{ 12,"of Puncturing",6,4,4,1,'ar05',-10,-20,0,0,0,0,0,0,0,0,18,0,19,"-10-20 of Monster's Defense" },
{ 13,"of Thorns",14,10,5,1,'at00',1,3,0,0,0,0,0,0,0,0,1029,0,19,"Attack Takes 1-4 Damage" },
{ 14,"of Spikes",21,15,5,1,'at00',2,6,0,0,0,0,0,0,0,0,1029,1,19,"Attack Takes 2-6 Damage" },
{ 15,"of Readiness",1,0,6,0,'at01',10,10,0,0,0,0,0,0,0,0,754,0,14,"Slightly Increased Weapon Speed" },
{ 16,"of Alacrity",8,6,6,0,'at02',20,20,0,0,0,0,0,0,0,0,242,0,14,"Increased Weapon Speed" },
{ 17,"of Swiftness",19,14,6,0,'at02',30,30,0,0,0,0,0,0,0,0,18,0,14,"Increased Weapon Speed" },
{ 18,"of Quickness",25,18,6,0,'at03',40,40,0,0,0,0,0,0,0,0,18,1,14,"Greatly Increased Weapon Speed" },
{ 19,"of Blocking",1,0,7,0,'bl00',10,10,0,0,0,0,0,0,0,0,4,0,14,"+10% Chance to Block" },
{ 20,"of Deflecting",11,8,7,0,'bl00',20,20,0,0,0,0,0,0,0,0,4,0,14,"+20% Chance to Block" },
```

```
{ 21,"of the Apprentice",5,3,8,0,'ca00',10,10,0,0,0,0,0,0,0,0,22640,0,14,"Fast Cast Time" },
{ 22,"of the Magus",17,12,8,0,'ca02',20,20,0,0,0,0,0,0,0,0,16496,0,14,"Fastest Cast Time" },
{ 23,"of Frost",4,3,9,1,'dc00',1,1,'dc01',1,4,'dc03',125,125,0,0,0,242,0,5,"+1-4 Cold Damage" },
{ 24,"of the Glacier",18,13,9,1,'dc00',4,4,'dc01',4,12,'dc03',150,150,0,0,0,114,0,5,"+4-12 Cold Damage" },
{ 25,"of Frost",55,41,9,1,'dc00',1,1,'dc01',1,4,'dc03',125,125,0,0,0,5120,0,5,"+1-4 Cold Damage" },
{ 26,"of Warmth/Thawing",10,7,10,0,'dc02',1,1,0,0,0,0,0,0,0,0,0,23300,0,4,"Freeze Length Reduction" },
{ 27,"of Flame",4,3,11,1,'df00',1,1,'df01',2,6,0,0,0,0,0,0,242,0,8,"+2-6 Fire Damage" },
{ 28,"of Fire",15,11,11,1,'df00',2,2,'df01',6,11,0,0,0,0,0,0,242,0,8,"+6-11 Fire Damage" },
{ 29,"of Burning",25,18,11,1,'df00',10,10,'df01',10,20,0,0,0,0,0,0,242,1,8,"+10-20 Fire Damage" },
{ 30,"of Flame",40,30,11,1,'df00',1,1,'df01',2,6,0,0,0,0,0,0,6656,0,8,"+2-6 Fire Damage" },
{ 31,"of Shock",4,3,12,0,'dl00',1,1,'dl01',8,8,0,0,0,0,0,0,242,0,14,"+1-8 Lightning Damage" },
{ 32,"of Lightning",15,11,12,0,'dl00',1,1,'dl01',16,16,0,0,0,0,0,0,242,0,14,"+1-16 Lightning Damage" },
{ 33,"of Thunder",25,18,12,0,'dl00',1,1,'dl01',32,32,0,0,0,0,0,0,242,1,14,"+1-32 Lightning Damage" },
{ 34,"of Shock",50,37,12,0,'dl00',1,1,'dl01',8,8,0,0,0,0,0,0,6402,0,14,"+1-8 Lightning Damage" },
{ 35,"of Craftsmanship",1,0,13,1,'dm00',1,2,0,0,0,0,0,0,0,0,0,6386,0,3,"+1-2 to Maximum Damage" },
{ 36,"of Quality",4,3,13,1,'dm00',2,2,0,0,0,0,0,0,0,0,0,242,0,3,"+2 to Maximum Damage" },
{ 37,"of Maiming",7,5,13,1,'dm00',3,4,0,0,0,0,0,0,0,0,0,242,0,3,"+3-4 to Maximum Damage" },
{ 38,"of Slaying",11,8,13,1,'dm00',5,7,0,0,0,0,0,0,0,0,0,242,0,3,"+5-7 to Maximum Damage" },
{ 39,"of Gore",14,10,13,1,'dm00',8,10,0,0,0,0,0,0,0,0,0,242,0,3,"+8-10 to Maximum Damage" },
{ 40,"of Carnage",19,14,13,1,'dm00',11,14,0,0,0,0,0,0,0,0,0,146,1,3,"+11-15 to Maximum Damage" },
{ 41,"of Slaughter",25,18,13,1,'dm00',15,20,0,0,0,0,0,0,0,0,0,146,1,3,"+15-20 to Maximum Damage" },
{ 42,"of Maiming",42,31,13,1,'dm00',3,4,0,0,0,0,0,0,0,0,0,6148,0,3,"+3-4 to Maximum Damage" },
{ 43,"of Worth",2,0,14,0,'dm01',1,1,0,0,0,0,0,0,0,0,0,8322,0,3,"+1 to Minimum Damage" },
{ 44,"of Measure",6,4,14,0,'dm01',2,2,0,0,0,0,0,0,0,0,0,242,0,3,"+2 to Minimum Damage" },
{ 45,"of Excellence",12,9,14,0,'dm01',3,3,0,0,0,0,0,0,0,0,0,6386,0,3,"+3 to Minimum Damage" },
{ 46,"of Performance",18,13,14,1,'dm01',4,5,0,0,0,0,0,0,0,0,0,242,1,3,"+4-5 to Minimum Damage" },
{ 47,"of Measure",37,27,14,0,'dm01',2,2,0,0,0,0,0,0,0,0,0,6145,0,3,"+2 to Minimum Damage" },
{ 48,"of Blight",5,3,15,0,'dp00',8,8,'dp01',24,24,'dp02',75,75,0,0,0,226,0,11,"+2-7 to Poison Damage" },
{ 49,"of Venom",15,11,15,0,'dp00',16,16,'dp01',48,48,'dp02',75,75,0,0,0,226,0,11,"+4-14 to Poison Damage" },
{ 50,"of Pestilence",25,18,15,0,'dp00',32,32,'dp01',72,72,'dp02',100,100,0,0,0,226,1,11,"+12-28 to Poison Damage" },
{ 51,"of Blight",45,33,15,0,'dp00',8,8,'dp01',24,24,'dp02',75,75,0,0,0,6144,0,11,"+2-7 to Poison Damage" },
{ 52,"of Dexterity",4,3,16,0,'dx00',1,1,0,0,0,0,0,0,0,0,0,14336,0,16,"+1-3 to Dexterity" },
{ 53,"of Dexterity",5,3,16,1,'dx00',1,3,0,0,0,0,0,0,0,0,0,7056,0,16,"+1-3 to Dexterity" },
{ 54,"of Skill",10,7,16,1,'dx00',4,6,0,0,0,0,0,0,0,0,0,7056,0,16,"+4-6 to Dexterity" },
{ 55,"of Skill",45,33,16,1,'dx00',4,6,0,0,0,0,0,0,0,0,0,1029,0,16,"+4-6 to Dexterity" },
{ 56,"of Accuracy",18,13,16,1,'dx00',7,10,0,0,0,0,0,0,0,0,0,7056,0,16,"+7-10 to Dexterity" },
{ 57,"of Precision",22,16,16,1,'dx00',11,15,0,0,0,0,0,0,0,0,0,4240,0,16,"+11-15 to Dexterity" },
{ 58,"of Precision",60,45,16,1,'dx00',11,15,0,0,0,0,0,0,0,0,0,3845,0,16,"+11-15 to Dexterity" },
{ 59,"of Perfection",30,22,16,1,'dx00',16,20,0,0,0,0,0,0,0,0,0,4736,1,16,"+16-20 to Dexterity" },
{ 60,"of Balance",5,3,17,0,'gh00',10,10,0,0,0,0,0,0,0,0,0,9477,0,14,"Fast Hit Recovery" },
{ 61,"of Stability",9,6,17,0,'gh02',20,20,0,0,0,0,0,0,0,0,0,1285,1,14,"Fastest Hit Recovery" },
{ 62,"of the Horse",1,0,18,0,'hl08',50,50,0,0,0,0,0,0,0,0,0,16384,0,15,"Not Available" },
{ 63,"of Regeneration",10,7,19,0,'lf00',3,3,0,0,0,0,0,0,0,0,0,7184,0,9,"+3 to Life Regeneration" },
{ 64,"of Regeneration",40,30,19,0,'lf00',3,3,0,0,0,0,0,0,0,0,0,516,0,9,"+3 to Life Regeneration" },
{ 65,"of Regeneration",70,52,19,0,'lf00',3,3,0,0,0,0,0,0,0,0,0,483,0,9,"+3 to Life Regeneration" },
{ 66,"of Regrowth",17,12,19,0,'lf00',5,5,0,0,0,0,0,0,0,0,0,4112,1,9,"+5 to Life Regeneration" },
{ 67,"of Regrowth",55,41,19,0,'lf00',5,5,0,0,0,0,0,0,0,0,0,3072,1,9,"+5 to Life Regeneration" },
{ 68,"of Vileness",9,6,20,0,'hl10',1,1,0,0,0,0,0,0,0,0,0,98,0,3,"Prevent Monster Healing" },
{ 69,"of the Ox",1,0,21,1,'hl12',3,5,0,0,0,0,0,0,0,0,0,0,2,"Not Available" },
{ 70,"of Greed",2,0,22,1,'ib00',40,60,0,0,0,0,0,0,0,0,0,6144,0,15,"40-60% More Gold" },
{ 71,"of Wealth",17,12,22,1,'ib00',80,120,0,0,0,0,0,0,0,0,0,5888,1,15,"80-120% More Gold" },
{ 72,"of Chance",12,9,23,1,'ib01',10,19,0,0,0,0,0,0,0,0,0,30720,0,15,"10-19% Better Chance of Getting Magic Item" },
{ 73,"of Fortune",16,12,23,1,'ib01',20,35,0,0,0,0,0,0,0,0,0,13056,1,15,"20-35% Better Chance of Getting Magic Item" },
{ 74,"of Energy",4,3,24,0,'mm00',1,1,0,0,0,0,0,0,0,0,0,30720,0,16,"+1-5 to Energy" },
{ 75,"of Energy",5,3,24,1,'mm00',1,5,0,0,0,0,0,0,0,0,0,22640,0,16,"+1-5 to Energy" },
{ 76,"of the Mind",10,7,24,1,'mm00',6,10,0,0,0,0,0,0,0,0,0,22640,0,16,"+6-10 to Energy" },
{ 77,"of Brilliance",18,13,24,1,'mm00',11,15,0,0,0,0,0,0,0,0,0,22640,0,16,"+11-15 to Energy" },
{ 78,"of Sorcery",22,16,24,1,'mm00',16,20,0,0,0,0,0,0,0,0,0,20592,1,16,"+16-20 to Energy" },
{ 79,"of Wizardry",30,22,24,1,'mm00',21,30,0,0,0,0,0,0,0,0,0,22640,1,16,"+21-30 to Energy" },
{ 80,"of the Bear",8,6,25,0,'ko00',1,1,0,0,0,0,0,0,0,0,0,114,0,19,"Monster Knockback" },
{ 81,"of Light",6,4,26,0,'lt00',1,1,0,0,0,0,0,0,0,0,0,32757,0,13,"+1 to Light Radius" },
{ 82,"of Radiance",15,11,26,0,'lt00',3,3,0,0,0,0,0,0,0,0,0,22768,0,13,"+2 to Light Radius" },
{ 83,"of the Sun",17,12,26,0,'lt00',5,5,0,0,0,0,0,0,0,0,0,31472,0,13,"+5 to Light Radius" },
{ 84,"of Life",23,17,27,0,'mh00',5,5,0,0,0,0,0,0,0,0,0,13317,0,9,"+5 to Life" },
{ 85,"of the Jackal",3,2,27,1,'mh00',1,5,0,0,0,0,0,0,0,0,0,16245,0,9,"+1-5 to Life" },
{ 86,"of the Fox",6,4,27,1,'mh00',5,10,0,0,0,0,0,0,0,0,0,8053,0,9,"+5-10 to Life" },
{ 87,"of the Wolf",14,10,27,1,'mh00',11,20,0,0,0,0,0,0,0,0,0,8053,0,9,"+11-20 to Life" },
{ 88,"of the Wolf",45,33,27,1,'mh00',11,20,0,0,0,0,0,0,0,0,0,130,0,9,"+11-20 to Life" },
{ 89,"of the Tiger",20,15,27,1,'mh00',21,30,0,0,0,0,0,0,0,0,0,7941,0,9,"+21-30 to Life" },
{ 90,"of the Mammoth",24,18,27,1,'mh00',31,40,0,0,0,0,0,0,0,0,0,6149,1,9,"+31-40 to Life" },
{ 91,"of the Mammoth",60,45,27,1,'mh00',31,40,0,0,0,0,0,0,0,0,0,1538,1,9,"+31-40 to Life" },
{ 92,"of the Colosuss",30,22,27,1,'mh00',41,60,0,0,0,0,0,0,0,0,0,13317,1,9,"+41-60 to Life" },
{ 93,"of the Leech",6,4,28,1,'mh03',4,7,0,0,0,0,0,0,0,0,0,6386,0,9,"+4-7% Life Stolen" },
{ 94,"of the Locust",20,15,28,1,'mh03',8,10,0,0,0,0,0,0,0,0,0,114,1,9,"+8-10% Life Stolen" },
{ 95,"of the Bat",6,4,29,1,'mm03',4,8,0,0,0,0,0,0,0,0,0,6386,0,6,"+4-8% Mana Stolen" },
{ 96,"of the Vampire",20,15,29,1,'mm03',9,12,0,0,0,0,0,0,0,0,0,114,1,6,"+8-10% Mana Stolen" },
{ 97,"of Defiance",25,18,30,0,'pd00',75,75,0,0,0,0,0,0,0,0,0,4101,1,8,"75% Poison Length Reduction" },
{ 98,"of Amelioration",18,13,30,0,'pd00',50,50,0,0,0,0,0,0,0,0,0,4101,0,8,"50% Poison Length Reduction" },
{ 99,"of Remedy",7,5,30,0,'pd00',25,25,0,0,0,0,0,0,0,0,0,7941,0,8,"25% Poison Length Reduction" },
```

```
{ 100,"of Ice",9,6,31,0,'rc00',2,2,0,0,0,0,0,0,0,0,0,0,0,5,"Not Available" },
{ 101,"of Simplicity",25,18,32,0,'rq00',-40,-40,0,0,0,0,0,0,0,0,0,135,0,0,"-40% Item Requirements" },
{ 102,"of Ease",15,11,32,0,'rq00',-20,-20,0,0,0,0,0,0,0,0,0,135,0,0,"-20% Item Requirements" },
{ 103,"of the Mule",3,2,33,1,'sp04',1,3,0,0,0,0,0,0,0,0,0,0,3,"Not Available" },
{ 104,"of Strength",5,3,34,1,'st00',1,3,0,0,0,0,0,0,0,0,0,0,7168,0,16,"+1-3 to Strength" },
{ 105,"of Might",10,7,34,1,'st00',4,6,0,0,0,0,0,0,0,0,0,0,14358,0,16,"+4-6 to Strength" },
{ 106,"of the Ox",18,13,34,1,'st00',7,10,0,0,0,0,0,0,0,0,0,0,7186,0,16,"+7-10 to Strength" },
{ 107,"of the Ox",47,35,34,1,'st00',7,10,0,0,0,0,0,0,0,0,0,0,772,0,16,"+7-10 to Strength" },
{ 108,"of the Giant",22,16,34,1,'st00',11,15,0,0,0,0,0,0,0,0,0,0,5138,1,16,"+11-15 to Strength" },
{ 109,"of the Giant",62,46,34,1,'st00',11,15,0,0,0,0,0,0,0,0,0,0,2820,1,16,"+11-15 to Strength" },
{ 110,"of the Titan",30,22,34,1,'st00',16,20,0,0,0,0,0,0,0,0,0,0,5376,1,16,"+16-20 to Strength" },
{ 111,"of Pacing",2,0,35,0,'ve00',10,10,0,0,0,0,0,0,0,0,0,0,256,0,14,"Fast Walking/Running" },
{ 112,"of Haste",12,9,35,0,'ve01',20,20,0,0,0,0,0,0,0,0,0,0,256,0,14,"Faster Walking/Running" },
{ 113,"of Speed",22,16,35,0,'ve02',30,30,0,0,0,0,0,0,0,0,0,0,256,0,14,"Fastest Walking/Running" },
};

int nMagicPrefixTable = sizeof MagicPrefixTable / sizeof MagicPrefixTable[0];
int nMagicSuffixTable = sizeof MagicSuffixTable / sizeof MagicSuffixTable[0];
```

```
#include "JamellaD2E.h"

struct _MagicPreSuffixTree MagicPreSuffixTree[] =
{
{ 1,"+ Attack Rating",-1 },
{ 2,"Bronze (10-20)",26 },
{ 2,"Iron (21-40)",27 },
{ 2,"Steel (41-60)",28 },
{ 2,"Silver (61-80)",29 },
{ 2,"Silver (61-80)",30 },
{ 2,"Gold (81-100)",31 },
{ 2,"Platinum (101-120)",32 },
{ 2,"Meteoric (121-150)",33 },
{ 1,"+ Damage",-1 },
{ 2,"Jagged (110-120%)",12 },
{ 2,"Deadly (121-130%)",13 },
{ 2,"Vicious (131-140%)",14 },
{ 2,"Brutal (141-150%)",15 },
{ 2,"Massive (151-165%)",16 },
{ 2,"Savage (166-180%)",17 },
{ 2,"Merciless (181-200%)",18 },
{ 1,"+ Attack Rating & + Damage",-1 },
{ 2,"Sharp (10-20 & 110-120%)",34 },
{ 2,"Fine (21-40 & 121-130%)",35 },
{ 2,"Warrior's (41-60 & 131-140%)",36 },
{ 2,"Soldier's (61-80 & 141-150%)",37 },
{ 2,"Knight's (81-100 & 151-165%)",38 },
{ 2,"Lord's (101-120 & 166-180%)",39 },
{ 2,"King's (121-150 & 181-200%)",40 },
{ 1,"+ Defense",-1 },
{ 2,"Sturdy (120-130%)",1 },
{ 2,"Strong (131-140%)",2 },
{ 2,"Glorious (141-150%)",3 },
{ 2,"Blessed (151-160%)",4 },
{ 2,"Saintly (161-180%)",5 },
{ 2,"Holy (181-200%)",6 },
{ 1,"- Magic Damage",-1 },
{ 2,"Devious (Reduced by 1)",7 },
{ 2,"Fortified (Reduced by 2)",8 },
{ 1,"+ Mana",-1 },
{ 2,"Lizard's (+1-5)",52 },
{ 2,"Snake's (+5-10)",54 },
{ 2,"Serpent's (+11-20)",55 },
{ 2,"Serpent's (+11-20)",56 },
{ 2,"Drake's (+21-30)",57 },
{ 2,"Dragon's (+31-40)",58 },
{ 2,"Dragon's (+31-40)",59 },
{ 2,"Wyrm's (+41-60)",60 },
{ 1,"+ Light Radius",-1 },
{ 2,"Glimmering (+1)",48 },
{ 2,"Glowing (+2)",49 },
{ 1,"+ Fire Resist",-1 },
{ 2,"Crimson (5-10%)",74 },
{ 2,"Burgundy (11-20%)",75 },
{ 2,"Burgundy (11-20%)",76 },
{ 2,"Garnet (21-30%)",77 },
{ 2,"Garnet (21-30%)",78 },
{ 2,"Ruby (31-50%)",80 },
{ 1,"+ Cold Resist",-1 },
{ 2,"Azure (5-10%)",65 },
{ 2,"Lapis (11-20%)",66 },
{ 2,"Lapis (11-20%)",67 },
{ 2,"Cobalt (21-30%)",68 },
{ 2,"Cobalt (21-30%)",69 },
{ 2,"Sapphire (31-50%)",71 },
{ 1,"+ Lightning Resist",-1 },
{ 2,"Ocher (5-10%)",83 },
{ 2,"Tangerine (11-20%)",84 },
{ 2,"Tangerine (11-20%)",85 },
{ 2,"Coral (21-30%)",86 },
{ 2,"Coral (21-30%)",87 },
{ 2,"Amber (31-50%)",89 },
{ 1,"+ Poison Resist",-1 },
{ 2,"Beryl (5-10%)",92 },
{ 2,"Jade (11-20%)",93 },
{ 2,"Jade (11-20%)",94 },
{ 2,"Viridian (21-30%)",95 },
{ 2,"Viridian (21-30%)",96 },
```

```
{ 2,"Emerald (31-50%)",98 },
{ 1,"+ Resist All",-1 },
{ 2,"Prismatic (15-25%)",63 },
{ 2,"Prismatic (15%)",64 },
{ 1,"+ Skill Levels",-1 },
{ 2,"Fletcher's (+1 to Amazon Skill Levels)",100 },
{ 2,"Archer's (+2 to Amazon Skill Levels)",101 },
{ 2,"Archer's (+2 to Amazon Skill Levels)",102 },
{ 2,"Slayer's (+1 to Barbarian Skill Levels)",112 },
{ 2,"Berserker's (+2 to Barbarian Skill Levels)",113 },
{ 2,"Berserker's (+2 to Barbarian Skill Levels)",114 },
{ 2,"Monk's (+1 to Paladin Skill Levels)",103 },
{ 2,"Priest's (+2 to Paladin Skill Levels)",104 },
{ 2,"Priest's (+2 to Paladin Skill Levels)",105 },
{ 2,"Summoner's (+1 to Necromancer Skill Levels)",106 },
{ 2,"Necromancer's (+2 to Necromancer Skill Levels)",107 },
{ 2,"Necromancer's (+2 to Necromancer Skill Levels)",108 },
{ 2,"Angel's (+1 to Sorceress Skill Levels)",109 },
{ 2,"Arch-Angel's (+2 to Sorceress Skill Levels)",110 },
{ 2,"Arch-Angel's (+2 to Sorceress Skill Levels)",111 },
{ 1,"Damage Generates Mana",-1 },
{ 2,"Vulpine (10%)",19 },
{ 1,"Each Kill Regenerates Mana",-1 },
{ 2,"Triumphant (+1 per Kill)",117 },
{ 1,"Better Chance of Getting Magic Items",-1 },
{ 2,"Fortuitous (10-15%)",42 },
{ 1,"+ Max Stamina",-1 },
{ 2,"Rugged (+5-10)",25 },
{ 1,"+ Stamina Regeneration",-1 },
{ 2,"Tireless",24 },
{ 1,"Make Monsters Flee",-1 },
{ 2,"Howling",41 },

{ 1,"+ Strength",-1 },
{ 2,"of Strength (+1-3)",360 },
{ 2,"of Might (+4-6)",361 },
{ 2,"of the Ox (+7-10)",362 },
{ 2,"of the Ox (+7-10)",363 },
{ 2,"of the Giant (+11-15)",364 },
{ 2,"of the Giant (+11-15)",365 },
{ 2,"of the Titan (+16-20)",366 },
{ 1,"+ Dexterity",-1 },
{ 2,"of Dexterity (+1-3)",308 },
{ 2,"of Dexterity (+1-3)",309 },
{ 2,"of Skill (+4-6)",310 },
{ 2,"of Skill (+4-6)",311 },
{ 2,"of Accuracy (+7-10)",312 },
{ 2,"of Precision (+11-15)",313 },
{ 2,"of Precision (+11-15)",314 },
{ 2,"of Perfection (+16-20)",315 },
{ 1,"+ Energy",-1 },
{ 2,"of Energy (+1-5)",330 },
{ 2,"of Energy (+1-5)",331 },
{ 2,"of Mind (+6-10)",332 },
{ 2,"of Brilliance (+11-15)",333 },
{ 2,"of Sorcery (+16-20)",334 },
{ 2,"of Wizardy (+21-30)",335 },
{ 1,"+ Life",-1 },
{ 2,"of Life (+5)",340 },
{ 2,"of the Jackal (+1-5)",341 },
{ 2,"of the Fox (+5-10)",342 },
{ 2,"of the Wolf (+11-20)",343 },
{ 2,"of the Wolf (+11-20)",344 },
{ 2,"of the Tiger (+21-30)",345 },
{ 2,"of the Mammoth (+31-40)",346 },
{ 2,"of the Mammoth (+31-40)",347 },
{ 2,"of the Colosuss (+41-60)",348 },
{ 1,"+ Light Radius",-1 },
{ 2,"of Light (+1)",337 },
{ 2,"of Radiance (+3)",338 },
{ 2,"of the Sun (+5)",339 },
{ 1,"+ Minimum Damage",-1 },
{ 2,"of Worth (+1)",299 },
{ 2,"of Measure (+2)",300 },
{ 2,"of Measure (+2)",303 },
{ 2,"of Excellence (+3)",301 },
{ 2,"of Performance (+4-5)",302 },
{ 1,"+ Maximum Damage",-1 },
{ 2,"of Craftsmanship (+1-2)",291 },
```

```
{ 2,"of Quality (+2)",292 },
{ 2,"of Maiming (+3-4)",293 },
{ 2,"of Maiming (+3-4)",298 },
{ 2,"of Slaying (+5-7)",294 },
{ 2,"of Gore (+8-10)",295 },
{ 2,"of Carnage (+11-15)",296 },
{ 2,"of Slaughter (+15-20)",297 },
{ 1,"+ Fire Damage",-1 },
{ 2,"of Flame (2-6)",283 },
{ 2,"of Flame (2-6)",286 },
{ 2,"of Fire (6-11)",284 },
{ 2,"of Burning (10-20)",285 },
{ 1,"+ Cold Damage",-1 },
{ 2,"of Frost (1-4)",279 },
{ 2,"of Frost (1-4)",281 },
{ 2,"of the Glacier (4-12)",280 },
{ 1,"+ Lightning Damage",-1 },
{ 2,"of Shock (1-8)",287 },
{ 2,"of Shock (1-8)",290 },
{ 2,"of Lightning (1-16)",288 },
{ 2,"of Thunder (1-32)",289 },
{ 1,"+ Poison Damage",-1 },
{ 2,"of Blight (2-7)",304 },
{ 2,"of Blight (2-7)",307 },
{ 2,"of Venom (4-14)",305 },
{ 2,"of Pestilence (12-28)",306 },
{ 1,"Poison Length Reduction",-1 },
{ 2,"of Remedy (25%)",355 },
{ 2,"of Amelioration (50%)",354 },
{ 2,"of Defiance (75%)",353 },
{ 1,"+ Life Stolen",-1 },
{ 2,"of the Leech (4-7%)",349 },
{ 2,"of the Locust (8-10%)",350 },
{ 1,"+ Mana Stolen",-1 },
{ 2,"of the Bat (4-8%)",351 },
{ 2,"of the Vampire (8-10%)",352 },
{ 1,"+ Life Regeneration",-1 },
{ 2,"of Regeneration (+3)",319 },
{ 2,"of Regeneration (+3)",320 },
{ 2,"of Regeneration (+3)",321 },
{ 2,"of Regrowth (+5)",322 },
{ 2,"of Regrowth (+5)",323 },
{ 1,"+ Weapon Speeds",-1 },
{ 2,"of Readiness (Slight Increase)",271 },
{ 2,"of Alacrity (Increase)",272 },
{ 2,"of Swiftness (Increase)",273 },
{ 2,"of Quickness (Greater Increase)",274 },
{ 1,"+ Chance to Block",-1 },
{ 2,"of Blocking (+10%)",275 },
{ 2,"of Deflecting (+20%)",276 },
{ 1,"Faster Hit Recovery",-1 },
{ 2,"of Balance (Fast)",316 },
{ 2,"of Stability (Fastest)",317 },
{ 1,"Damage Reduction",-1 },
{ 2,"of Health (by 1)",256 },
{ 2,"of Protection (by 2)",257 },
{ 2,"of Absorption (by 3)",258 },
{ 2,"of Life (by 4)",259 },
{ 2,"of Life (by 4)",260 },
{ 1,"Magic Damage Reduction",-1 },
{ 2,"of Warding (by 1)",261 },
{ 2,"of the Sentinel (by 2)",263 },
{ 2,"of Guarding (by 3)",262 },
{ 2,"of Negation (by 4)",264 },
{ 1,"More Gold",-1 },
{ 2,"of Greed (40-60%)",326 },
{ 2,"of Wealth (80-120%)",327 },
{ 1,"Better Chance of Getting Magic Item",-1 },
{ 2,"of Chance (10-19%)",328 },
{ 2,"of Fortune (20-35%)",329 },
{ 1,"Monster Defense",-1 },
{ 2,"of Puncturing (-10-20)",268 },
{ 2,"of Bashing (-25-40)",267 },
{ 1,"Cast Time",-1 },
{ 2,"of the Apprentice (Fast)",277 },
{ 2,"of the Magus (Fastest)",278 },
{ 1,"Attacker Takes Damage",-1 },
{ 2,"of Thorns (1-6)",269 },
{ 2,"of Spikes (2-6)",270 },
```

```
{ 1,"- Item Requirements",-1 },
{ 2,"of Ease (-20%)",358 },
{ 2,"of Simplicity (-40%)",357 },
{ 1,"Faster Walking/Running",-1 },
{ 2,"of Pacing (Fast)",367 },
{ 2,"of Haste (Faster)",368 },
{ 2,"of Speed (Fastest)",369 },
{ 1,"Monster Knockback",-1 },
{ 2,"of the Bear",336 },
{ 1,"Attack Ignores Target's Defense",-1 },
{ 2,"of Piercing",266 },
{ 1,"Prevent Moster Healing",-1 },
{ 2,"of Vileness",324 },
{ 1,"Freeze Length Reduction",-1 },
{ 2,"of Warmth/Thawing",282 },
};

//int nMagicPrefixTree = sizeof MagicPrefixTree / sizeof MagicPrefixTree[0];
//int nMagicSuffixTree = sizeof MagicSuffixTree / sizeof MagicSuffixTree[0];
int nMagicPreSuffixTree = sizeof MagicPreSuffixTree / sizeof MagicPreSuffixTree[0];
```

```
// MainDlg.cpp from D2E

#include "JamellaD2E.h"

HWND    hMainDialog = 0;
RECT    rMainDialog;

// Tab Control
HWND    hTab;
RECT    rTab,rTabDialog;
int     TabSelection = 0;
HWND    hTabDialog;

// Procedure Prototypes
LRESULT CALLBACK MainDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam);
LRESULT CALLBACK TabDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam);

struct maindlgtab Tabs[] =
{
    { IDD_TAB1,         IDB_TAB1, "Stats",      &Tab1DialogProc },
    { IDD_TAB2,         IDB_TAB2, "Inventory",  &Tab2DialogProc },
/*
#if INVGRIDS == 0
    { IDD_TAB2,         IDB_TAB2, "Inventory",  &Tab2DialogProc },
#else
    { IDD_TAB2ExGrid,   IDB_TAB2, "Inventory",  &Tab2DialogProc },
#endif
*/
    { IDD_TAB3,         IDB_TAB3, "Skills",     &Tab3DialogProc },
    { IDD_TAB4,         IDB_TAB4, "Quests",     &Tab4DialogProc },
    { IDD_TAB5,         IDB_TAB5, "Waypoints",  &Tab5DialogProc },
};

struct maindlgtab JamellaTab =
    { IDD_TAB0, 0,         "Jamella",   &Tab0DialogProc };

const int Tabsn = sizeof Tabs / sizeof Tabs[0];

void ResizeDialog()
{
    // Calculate Tab Control Size
    GetWindowRect(hTabDialog,&rTab);

    TabCtrl_AdjustRect(hTab,TRUE,&rTab);
    rTab.right -= rTab.left;
    rTab.bottom -= rTab.top;
    rTab.top = 0;
    rTab.left = 0;

    SetWindowPos(hTab,NULL,
        0,0,
        rTab.right,rTab.bottom,
        SWP_NOZORDER);

    // Adjust for Dialog Size
    CopyRect(&rTabDialog,&rTab);
    TabCtrl_AdjustRect(hTab,FALSE,&rTabDialog);

    // Center and resize Main Dialog Box
    {
        RECT WorkArea;
        SystemParametersInfo(SPI_GETWORKAREA ,0,&WorkArea,0);

        int width = rTab.right + 2 * GetSystemMetrics(SM_CXDLGFRAME);
        int height = rTab.bottom + 2 * GetSystemMetrics(SM_CXDLGFRAME)
            + GetSystemMetrics(SM_CYMENU)
            + GetSystemMetrics(SM_CYCAPTION);

        int top = (WorkArea.bottom - WorkArea.top - height) / 2;
        int left = (WorkArea.right - WorkArea.left - width) / 2;

        SetWindowPos(hMainDialog,NULL,
            left,top,
            width,height,
            SWP_NOZORDER);
    }

}
```

```
void switchtab(int i)
{
    if (i < 0 && TabSelection >= 0) {

        if (hTabDialog) {
            DestroyWindow(hTabDialog);
        }

        // Show Jamella
        TabCtrl_SetCurSel(hTab,-1);
        TabSelection = -1;

        hTabDialog = CreateDialogIndirect(
            hInstance,
            JamellaTab.dialogtemplate,
            hTab,
            (DLGPROC)JamellaTab.proc);

        SetWindowPos(hTabDialog,HWND_TOP,
            rTabDialog.left,rTabDialog.top,0,0,
            SWP_NOSIZE);

        ResizeDialog();

        ShowWindow(hTabDialog,SW_SHOW);
    }
    else {
        TabCtrl_SetCurSel(hTab,i);
        TabSelection = i;
    }
}

bool validatetab()
{
    if (hTabDialog)
    if (!SendMessage(hTabDialog,WM_VALIDATE,0,0))
    {
        return true;
    }
    return false;
}

int MainDialog(const char *CmdLine)
{
    // First register a Window Class
    WNDCLASSEX wcx;
    wcx.cbSize = sizeof(wcx);
    wcx.style = CS_HREDRAW | CS_VREDRAW;
    wcx.lpfnWndProc = MainDialogProc;
    wcx.cbClsExtra = 0;
    wcx.cbWndExtra = 0;
    wcx.hInstance = hInstance;
    wcx.hIcon = LoadIcon(hInstance,MAKEINTRESOURCE(IDI_ICON));
    wcx.hCursor = LoadCursor(NULL,IDC_ARROW);
    wcx.hbrBackground = (HBRUSH)(COLOR_APPWORKSPACE + 1);
    wcx.lpszMenuName =  NULL;
    wcx.lpszClassName = "D2EMainDialog";
    wcx.hIconSm = wcx.hIcon;
    RegisterClassEx(&wcx);

    HMENU hMainMenu = LoadMenu(NULL,MAKEINTRESOURCE(IDR_MAINDIALOG));

    // Secondly create the Main Dialog Box
    hMainDialog = CreateWindowEx(
        0,
        "D2EMainDialog",PROGRAMNAME,
        WS_CAPTION | WS_SYSMENU | WS_CLIPCHILDREN | WS_MINIMIZEBOX,
        CW_USEDEFAULT,
        CW_USEDEFAULT,
        500,
        500,
        (HWND) NULL,
        hMainMenu,
        hInstance,
        (LPVOID) NULL);

    if (!hMainDialog) return ErrorMessage();

    // Setup Tab Control
```

```
HIMAGELIST himl = ImageList_Create(17,16,ILC_COLORDDB,Tabsn,0);
for(int i=0;i<Tabsn;i++)
{
    HBITMAP hBmp = LoadBitmap(hInstance,MAKEINTRESOURCE(Tabs[i].idicon));
    Tabs[i].imagelistindex = ImageList_Add(himl,hBmp,NULL);
    DeleteObject(hBmp);
}

GetClientRect(hMainDialog,&rMainDialog);

// Create Tab Control
hTab = CreateWindow(WC_TABCONTROL,"",
    WS_CHILD | WS_CLIPSIBLINGS | WS_VISIBLE | TCS_HOTTRACK,
    rMainDialog.left,rMainDialog.top,
    rMainDialog.right - rMainDialog.left,rMainDialog.bottom - rMainDialog.top,
    hMainDialog,NULL,hInstance,NULL);

if (!hTab) {
    ErrorMessage();
    DestroyWindow(hMainDialog);
    return false;
}

TabCtrl_SetImageList(hTab,himl);

// Set nice font
HFONT fTab = CreateFont(8,0,
    0,0,0,
    FALSE,FALSE,FALSE,
    DEFAULT_CHARSET,OUT_DEFAULT_PRECIS,CLIP_CHARACTER_PRECIS,
    DEFAULT_QUALITY,DEFAULT_PITCH,
    "MS Sans Serif");

PostMessage(hTab,WM_SETFONT,(WPARAM) fTab,TRUE);

// Load Tab Dialogs
for(int j=0;j<Tabsn;j++)
{
    TC_ITEM tie;
    tie.mask = TCIF_TEXT | (Tabs[j].idicon ? TCIF_IMAGE : NULL);
    tie.iImage = Tabs[j].imagelistindex;
    tie.pszText = Tabs[j].text;
    TabCtrl_InsertItem(hTab,j,&tie);

    HRSRC hrsrc = FindResource(NULL,MAKEINTRESOURCE(Tabs[j].iddialog),RT_DIALOG);
    HGLOBAL hglb = LoadResource(hInstance,hrsrc);
    Tabs[j].dialogtemplate = (DLGTEMPLATE *) LockResource(hglb);
}

// Load Jamella Dialog
{
    HRSRC hrsrc = FindResource(NULL,MAKEINTRESOURCE(JamellaTab.iddialog),RT_DIALOG);
    HGLOBAL hglb = LoadResource(hInstance,hrsrc);
    JamellaTab.dialogtemplate = (DLGTEMPLATE *) LockResource(hglb);
}

// Load Progress Dialog
{
    HRSRC hrsrc = FindResource(NULL,MAKEINTRESOURCE(IDD_PROGRESS),RT_DIALOG);
    HGLOBAL hglb = LoadResource(hInstance,hrsrc);
    DLGTEMPLATE* dlgtemp = (DLGTEMPLATE *) LockResource(hglb);

    hTabDialog = CreateDialogIndirect(
        hInstance,
        dlgtemp,
        hTab,
        (DLGPROC)TabDialogProc);
}

TabCtrl_SetCurSel(hTab,-1);
TabSelection = -1;

// Calculate Tab Control Size
GetWindowRect(hTabDialog,&rTab);

TabCtrl_AdjustRect(hTab,TRUE,&rTab);
rTab.right -= rTab.left;
rTab.bottom -= rTab.top;
rTab.top = 0;
```

```
    rTab.left = 0;

    SetWindowPos(hTab,NULL,
        0,0,
        rTab.right,rTab.bottom,
        SWP_NOZORDER);

    // Adjust for Dialog Size
    CopyRect(&rTabDialog,&rTab);
    TabCtrl_AdjustRect(hTab,FALSE,&rTabDialog);

    SetWindowPos(hTabDialog,HWND_TOP,
                rTabDialog.left,rTabDialog.top,0,0,
                SWP_NOSIZE);

    ResizeDialog();

    // Set initial states of main controls
    SendDlgItemMessage(hTabDialog,IDC_PROGRESS_Bar,PBM_SETBARCOLOR,0,PROGRESSCOLOR);
    EnableWindow(hTab,FALSE);
    for(int z=0;z < GetMenuItemCount(hMainMenu);z++) {
        EnableMenuItem(hMainMenu,z,MF_BYPOSITION | MF_GRAYED);
    }

    // Show Progress Dialog and Load Resources
    ShowWindow(hMainDialog,SW_SHOW);
    ShowWindow(hTabDialog,SW_SHOW);

    D2ELoadResources();

    CheckShellRegistry();

    // Load inital file
    if (CmdLineFile())
        fc.loadfile(hTabDialog,CmdLineFile());

    SetInvGridPreset(1);

    // Set working states of main controls
    EnableWindow(hTab,TRUE);
    for(z=0;z < GetMenuItemCount(hMainMenu);z++) {
        EnableMenuItem(hMainMenu,z,MF_BYPOSITION | MF_ENABLED);
    }

    // Switch to Jamella's Dialog
    TabSelection = 0;
    switchtab(-1);

    // Load Accelerators
    HACCEL accel = LoadAccelerators(hInstance,MAKEINTRESOURCE(IDR_MAIN));

    // Start the message loop
    MSG msg;
    while (GetMessage(&msg,(HWND) NULL,0,0))
    {
        if (!TranslateAccelerator(hMainDialog,accel,&msg)) {
            TranslateMessage(&msg);
        }
        DispatchMessage(&msg);
    }

    D2EUnLoadResources();

    // Free Main Dialog Resources
    DeleteObject(fTab);
    ImageList_Destroy(himl);
    DestroyMenu(hMainMenu);

    return msg.wParam;
}

char QueryFilename[260];
int QueryOpenFileName(HWND hWnd)
{
    // Retrieve Registry Key from Diablo 2
    char fdir[260];
    {
        HKEY regkey;
        if (RegOpenKeyEx(HKEY_CURRENT_USER,"Software\\Blizzard Entertainment\\Diablo II",0,KEY_READ,&regkey) == ERROR_SU
```

```
CCESS)
        {
            DWORD type = REG_SZ;
            DWORD fdirsize = 260;
            DWORD x = RegQueryValueEx(regkey,"Save Path",0,&type,(unsigned char*)fdir,&fdirsize);
            if (x != ERROR_SUCCESS)
            {
                memset(fdir,0,sizeof fdir);
            }
            RegCloseKey(regkey);
        }
    }

    // common dialog box structure
    OPENFILENAME ofn;

    {   // Initialize OPENFILENAME
        ZeroMemory(&ofn,sizeof(OPENFILENAME));
        ofn.lStructSize = sizeof(OPENFILENAME);
        ofn.hwndOwner = hWnd;
        ofn.lpstrFilter = "D2 Save Game (*.d2s)\0*.d2s\0";
        ofn.nFilterIndex = 0;
        ofn.lpstrFile = QueryFilename;
        ofn.nMaxFile = sizeof(QueryFilename);
        ofn.lpstrFileTitle = NULL;
        ofn.nMaxFileTitle = 0;

        if (fdir[0]) ofn.lpstrInitialDir = fdir;

        ofn.Flags = OFN_PATHMUSTEXIST | OFN_FILEMUSTEXIST | OFN_NOREADONLYRETURN | OFN_HIDEREADONLY;

        ZeroMemory(&QueryFilename,sizeof(QueryFilename));
    }

    // Display the Open Dialog Box
    return GetOpenFileName(&ofn);
}

LRESULT CALLBACK MainDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
    switch(uMsg)
    {
    case WM_CREATE:
        return true;

    case WM_COMMAND:
        switch (LOWORD(wParam))
        {
        case IDR_OPEN:
            if (!validatetab())
            {
                if (fc.isloaded())
                {
                    if (!RegOptions.NoAnnoyingMsgs) {
                        int user = MessageBox(hWnd,"Do you really want to discard all changes?",PROGRAMNAME,
                            MB_YESNO | MB_ICONQUESTION | MB_APPLMODAL);

                        if (user != IDYES) break;
                    }

                    fc.discardfile(hWnd);
                    switchtab(-1);
                }
                if (QueryOpenFileName(hWnd))
                {
                    fc.loadfile(hWnd,QueryFilename);
                }
            }
            return true;
        case IDR_NEW:
            if (!validatetab())
            {
                if (fc.isloaded())
                {
                    if (!RegOptions.NoAnnoyingMsgs) {

                        int user = MessageBox(hWnd,"Do you really want to discard all changes?",PROGRAMNAME,
                            MB_YESNO | MB_ICONQUESTION | MB_APPLMODAL);
```

```
                if (user != IDYES) break;
            }

            fc.discardfile(hWnd);
            switchtab(-1);
        }

        DialogBox(hInstance,MAKEINTRESOURCE(IDD_NEW),hWnd,(DLGPROC) NewDialogProc);
    }
    return true;
case IDR_SAVE:
    if (!validatetab())
    {
        if (fc.isloaded())
        {
            SendMessage(hTabDialog,WM_DESTROY,0,0);
            fc.savefile(hWnd);
        }
        else
        {
            MessageBox(hWnd,"You didn't open a file!",PROGRAMNAME,
                MB_OK| MB_ICONEXCLAMATION | MB_APPLMODAL);
        }
    }
    return true;
case IDR_RELOAD:
    {
        if (fc.isloaded())
        {
            if (!RegOptions.NoAnnoyingMsgs) {
                int user = MessageBox(hWnd,"Do you really want to discard all changes?",PROGRAMNAME,
                    MB_YESNO | MB_ICONQUESTION | MB_APPLMODAL);

                if (user != IDYES) break;
            }

            switchtab(-1);

            fc.reloadfile(hWnd);
        }
        else
        {
            MessageBox(hWnd,"You didn't open a file!",PROGRAMNAME,
                MB_OK| MB_ICONEXCLAMATION | MB_APPLMODAL);
        }
    }
    return true;
case IDR_CLOSE:
    {
        if (fc.isloaded())
        {
            if (!RegOptions.NoAnnoyingMsgs) {
                int user = MessageBox(hWnd,"Do you really want to discard all changes?",PROGRAMNAME,
                    MB_YESNO | MB_ICONQUESTION | MB_APPLMODAL);

                if (user != IDYES) break;
            }

            fc.discardfile(hWnd);
            switchtab(-1);
        }
        else
        {
            MessageBox(hWnd,"You didn't open a file!",PROGRAMNAME,
                MB_OK| MB_ICONEXCLAMATION | MB_APPLMODAL);
        }
    }
    return true;
case IDR_UOPTIONS:
    DialogBox(hInstance,MAKEINTRESOURCE(IDD_UOPTIONS),hWnd,(DLGPROC) UOptionsDialogProc);
    return true;
case IDR_EOPTIONS:
    DialogBox(hInstance,MAKEINTRESOURCE(IDD_EOPTIONS),hWnd,(DLGPROC) EOptionsDialogProc);
    return true;
case IDR_TEXTFILE:
    WriteTextSummary(hWnd);
    return true;
case IDR_INFO:
    DialogBox(hInstance,MAKEINTRESOURCE(IDD_INFO),hWnd,(DLGPROC) InfoDialogProc);
```

```
            return true;
        case IDCANCEL:
        case IDR_EXIT:
            if (fc.isloaded())
            {
                if (RegOptions.NoAnnoyingMsgs || MessageBox(hMainDialog,"Are you sure you want to exit and discard all c
hanges?",
                    PROGRAMNAME,MB_YESNO) == IDYES)
                {
                    PostQuitMessage(0);
                    DestroyWindow(hMainDialog);
                }
            }
            else
            {
                PostQuitMessage(0);
                DestroyWindow(hMainDialog);
            }
            return true;
        }
        break;
    case WM_NOTIFY:
        {
            NMHDR nmh = *(LPNMHDR)lParam;
            if (nmh.hwndFrom == hTab)
            {
                switch(nmh.code)
                {
                case TCN_SELCHANGING:
                    if (!fc.isloaded())
                    {
                        if (QueryOpenFileName(hWnd))
                        {
                            fc.loadfile(hWnd,QueryFilename);
                        }
                        return true;
                    }
                    if (hTabDialog)
                    if (!SendMessage(hTabDialog,WM_VALIDATE,0,0))
                    {
                        return true;
                    }
                    return false;

                case TCN_SELCHANGE:
                    {
                        // Get Selection
                        TabSelection = TabCtrl_GetCurSel(hTab);

                        // Destroy the current child dialog box, if any.
                        if (hTabDialog != NULL)
                            DestroyWindow(hTabDialog);

                        // Create the new child dialog box.
                        if (Tabs[TabSelection].dialogtemplate)
                        {
                            hTabDialog = CreateDialogIndirect(
                                hInstance,
                                Tabs[TabSelection].dialogtemplate,
                                hTab,
                                (DLGPROC) (Tabs[TabSelection].proc ? Tabs[TabSelection].proc : TabDialogProc));

                            SetWindowPos(hTabDialog,HWND_TOP,
                                rTabDialog.left,rTabDialog.top,0,0,
                                SWP_NOSIZE);

                            ResizeDialog();
                            ShowWindow(hTabDialog,SW_SHOW);
                        }
                        return true;
                    }
                }
            }
        }
        break;
    case WM_CLOSE:
        if (fc.isloaded())
        {
            if (RegOptions.NoAnnoyingMsgs || MessageBox(hMainDialog,"Are you sure you want to exit and discard all chang
```

```
es?",
                PROGRAMNAME,MB_YESNO) == IDYES)
            {
                PostQuitMessage(0);
                DestroyWindow(hMainDialog);
            }
        }
        else
        {
            PostQuitMessage(0);
            DestroyWindow(hMainDialog);
        }
        return true;
    case WM_DESTROY:
        PostQuitMessage(0);
        return false;
    case WM_CHAR:
        PostMessage(hTabDialog,WM_CHAR,wParam,lParam);
        return false;
    }
    return DefWindowProc(hWnd,uMsg,wParam,lParam);
}

LRESULT CALLBACK TabDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
    switch(uMsg)
    {
    case WM_INITDIALOG:
        {
        }
        return true;
    }
    return false;
}
```

```
#include "JamellaD2E.h"

const struct Modifier Modifiers[] =
{
{ 'ar00',"%+d Defense" },
{ 'ar01',"Damage Reduced by %d" },
{ 'ar02',"%+d%% to Defense" },
{ 'ar03',"Magic Damage Reduced by %d" },
{ 'ar04',"Ignores Target's Defense" },
{ 'ar05',"%d to Monster AC per Hit" },
{ 'ar06',"%d%% Target Defense" },
{ 'ar08',"%+d Defense vs Missile" },
{ 'ar09',"%+d Defense vs Melee" },
{ 'at00',"Attacker Takes Damage of %d" },
{ 'at01',"Slight Attack Rate Increase (%d Points)" },
{ 'at02',"Attack Rate Increase (%d Points)" },
{ 'at03',"Greatly Attack Rate Increase (%d Points)" },
{ 'at04',"Attacker Takes Lightning Damage of %d" },
{ 'bl00',"%+d%% Increased Blocking" },
{ 'bl02',"Faster Block Rate" },
{ 'ca00',"Fast Cast Rate" },
{ 'ca01',"Faster Cast Rate" },
{ 'ca02',"Fastest Cast Rate" },
{ 'dc00',"%+d Cold Damage Min" },
{ 'dc01',"%+d Cold Damage Max" },
{ 'dc02',"Half Freeze Duration" },
{ 'dc03',"Cold Duration: %d seconds" },
{ 'df00',"%+d Fire Damage Min" },
{ 'df01',"%+d Fire Damage Max" },
{ 'dl00',"%+d Lightning Damage Min" },
{ 'dl01',"%+d Lightning Damage Max" },
{ 'dm00',"%+d to Maximum Damage" },
{ 'dm01',"%+d to Minimum Damage" },
{ 'dm02',"%+d%% to Damage" },
{ 'dm03',"%d Damage taken Goes to Mana" },
{ 'dp00',"%+d Poison Damage Min" },
{ 'dp01',"%+d Poison Damage Max" },
{ 'dp02',"Poison Length %d seconds" },
{ 'ds00',"%d%% Deadly Strike" },
{ 'dx00',"%+d to Dexterity" },
{ 'gh00',"Fast Hit Recovery" },
{ 'gh01',"Faster Hit Recovery" },
{ 'gh02',"Fastest Hit Recovery" },
{ 'hl08',"Heal Stamina %+d%%" },
{ 'hl10',"Prevent Monster Heal" },
{ 'hl11',"Regenerate Mana %d%%" },
{ 'hl12',"%+d to Max Stamina" },
{ 'hl13',"%d%% Stamina Drain" },
{ 'ht00',"%+d to Attack Rating" },
{ 'ht01',"%d%% Bonus to Attack Rating" },
{ 'hw00',"Hit Causes Monster to Flee" },
{ 'hw01',"Hit blinds target" },
{ 'ib00',"%d%% Extra Gold From Monsters" },
{ 'ib01',"%d%% Better Chance of Getting Magic Item" },
{ 'in00',0 },
{ 'ko00',"Knockback" },
{ 'lf00',"Replenish Life %+d" },
{ 'lt00',"%+d to Light Radius" },
{ 'md01',"%+d%% Damage to Undead" },
{ 'md03',"%+d to Attack Rating against Undead" },
{ 'mh00',"%+d to Life" },
{ 'mh01',"%+d%% to Max Life" },
{ 'mh03',"%d%% Life Stolen Per Hit" },
{ 'mm00',"%+d to Mana" },
{ 'mm01',"%+d%% to Max Mana" },
{ 'mm03',"%d%% Mana Stolen Per Hit" },
{ 'pd00',"Poison Length Reduced by %d%%" },
{ 'ra00',"%+d to each Resistance" },
{ 'ra01',"%d%% To Maximum of each Resistance" },
{ 'rc00',"Cold Resist %d%%" },
{ 'rc01',"%d%% To Maximum Cold Resist" },
{ 'rf00',"Fire Resist %d%%" },
{ 'rf01',"%d%% To Maximum Fire Resist" },
{ 'rl00',"Lightning Resist %d%%" },
{ 'rl01',"%d%% To Maximum Lightning Resist" },
{ 'rp00',"Poison Resist %d%%" },
{ 'rp01',"%d%% To Maximum Poison Resist" },
{ 'rq00',"Requirements %d%%" },
{ 'sk00',"%+d to Amazon Skill Levels" },
```

```
{ 'sk01',"%+d to Paladin Skill Levels" },
{ 'sk02',"%+d to Necromancer Skill Levels" },
{ 'sk03',"%+d to Sorceress Skill Levels" },
{ 'sk04',"%+d to Barbarian Skill Levels" },
{ 'sk05',0 },
{ 'sk07',"%+d To Fire Skills" },
{ 'sk08',"%+d To All Skills" },
{ 'sp01',"Freezes Target" },
{ 'sp02',"%d%% Chance of Open Wounds" },
{ 'sp03',"%d%% Chance of Crushing Blow" },
{ 'sp04',0 },
{ 'sp05',"%+d Points of Mana After Each Kill" },
{ 'sp06',0 },
{ 'sp07',"Slows Target By %d%%" },
{ 'sp10',"Cannot Be Frozen" },
{ 'sp13',"Piercing Attack" },
{ 'sp14',"Fires Magic Arrows" },
{ 'sp15',"Fires Explosive Arrows" },
{ 'st00',"%+d to Strength" },
{ 've00',"Fast Run/Walk" },
{ 've01',"Faster Run/Walk" },
{ 've02',"Fastest Run/Walk" },
};

int nModifiers = sizeof Modifiers / sizeof Modifiers[0];

char CodeStringTemp[5];
```

```
// NewBox.cpp from D2E

#include "JamellaD2E.h"

inline bool ValidateName(const char *s)
{
    if (*s == 0) return false;

    while(*s)
    {
        if (!isalpha(*s))
            return false;
        s++;
    }
    return true;
}

inline void MangleNames(char *d, const char *file, const char* nname,const char *oname)
{
    int ol = strlen(oname);

    d += strlen(d);

    while(*file)
    {
        if (oname && *file == *oname && strncmp(file,oname,ol) == 0) {
            strcpy(d,nname);
            d += strlen(nname);
            file += ol;
            oname = 0;
        }
        else {
            *d++ = *file++;
            *d = 0;
        }
    }
}

struct
{
    char    *name;
    char    *rc;
    char    *description;
}
CharTemplates[] =
{
    { "Tyrael", MAKEINTRESOURCE(IDN_SetTyrael), "Fully equipped set: Angelic Raidament" }
};

LRESULT CALLBACK NewDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
    switch(uMsg)
    {
    case WM_INITDIALOG:
        {
            SendDlgItemMessage(hWnd,IDC_NEW_Class,CB_RESETCONTENT,0,0);
            for (int i=0;i<5;i++)
                SendDlgItemMessage(hWnd,IDC_NEW_Class,CB_ADDSTRING,0,(LPARAM) CharClasses[i]);
            SendDlgItemMessage(hWnd,IDC_NEW_Class,CB_SETCURSEL,0,0);

            //for(int z=0;z
            //IDC_NEW_Templates
        }
        return true;
    case WM_COMMAND:
        switch (LOWORD(wParam))
        {
        case IDC_NEW_CreateNewbie:
            {
                GetDlgItemText(hWnd,IDC_NEW_Name,buffer,256);
                if (!ValidateName(buffer))
                {
                    ErrorBox("Invalid Name:\nOnly A-Z & a-z allowed!");
                    return false;
                }

                int Class = SendDlgItemMessage(hWnd,IDC_NEW_Class,CB_GETCURSEL,0,0);

                const char* rc =    (Class == 0) ? MAKEINTRESOURCE(IDN_Amazon) :
```

```
                    (Class == 1) ? MAKEINTRESOURCE(IDN_Sorceress) :
                    (Class == 2) ? MAKEINTRESOURCE(IDN_Necromancer) :
                    (Class == 3) ? MAKEINTRESOURCE(IDN_Paladin) :
                    (Class == 4) ? MAKEINTRESOURCE(IDN_Barbarian) : 0;

                HRSRC rcsrc = FindResource(hInstance,rc,"D2S");
                HGLOBAL hglb = LoadResource(hInstance,rcsrc);
                int size = SizeofResource(hInstance,rcsrc);
                BYTE *mem = (BYTE*)LockResource(hglb);

                int erroffset = fc.transferdata(mem,size);

                if (erroffset < 0)
                {
                    fc.loaded = true;

                    fc.setfilename(buffer);
                    strcpy(fc.Header.playername,buffer);
                }
                else
                {
                    char text[80];

                    sprintf(text,"Corrupt field encountered in save game data! (@ offset %i)",erroffset);
                    MessageBox(hWnd,text,PROGRAMNAME,MB_OK | MB_ICONSTOP | MB_APPLMODAL);
                    fc.clear();
                }

                EndDialog(hWnd,IDOK);
                return true;
            }
        case IDCANCEL:
            EndDialog(hWnd,wParam);
            return true;
        }
        break;
    case WM_CLOSE:
        EndDialog(hWnd,wParam);
        return true;
    case WM_DESTROY:
        return false;
    }
    return false;
}

LRESULT CALLBACK RenameDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
    switch(uMsg)
    {
    case WM_INITDIALOG:
        {
            SetDlgItemText(hWnd,IDC_RENAME_Name,fc.Header.playername);
            CheckRadioButton(hWnd,IDC_RENAME_Move,IDC_RENAME_Copy,IDC_RENAME_Move);
        }
        return true;
    case WM_COMMAND:
        switch (LOWORD(wParam))
        {
        case IDOK:
            {
                char NewName[32];

                GetDlgItemText(hWnd,IDC_RENAME_Name,NewName,32);
                if (!ValidateName(NewName))
                {
                    ErrorBox("Invalid Name:\nOnly A-Z & a-z allowed!");
                    return false;
                }

                char sdir[260];
                if (GetDiabloSaveDirectory(sdir))
                {
                    char srchdir[260];
                    sprintf(srchdir,"%s\\%s.*",sdir,NewName);

                    WIN32_FIND_DATA FData;
                    HANDLE hFind = FindFirstFile(srchdir,&FData);

                    if (hFind != INVALID_HANDLE_VALUE)
```

```
                    {
                        if (MessageBox(hWnd,"A character with the name already exists!\nDo you want to delete him/her?",
PROGRAMNAME,MB_YESNO | MB_ICONEXCLAMATION) != IDYES) {
                            FindClose(hFind);
                            return false;
                        }

                        int fLoop = true;
                        while(fLoop)
                        {
                            char dfile[260];

                            sprintf(dfile,"%s\\%s",sdir,FData.cFileName);
                            DeleteFile(dfile);

                            fLoop = FindNextFile(hFind, &FData);
                        }

                        FindClose(hFind);
                    }
                }

                if (IsDlgButtonChecked(hWnd,IDC_RENAME_Move) == BST_CHECKED)
                {
                    char srch[260];
                    sprintf(srch,"%s\\%s.*",sdir,fc.Header.playername);

                    WIN32_FIND_DATA FData;
                    HANDLE hFind = FindFirstFile(srch,&FData);

                    if (hFind != INVALID_HANDLE_VALUE)
                    {
                        int fLoop = true;
                        while(fLoop)
                        {
                            char nfile[260];
                            strcpy(nfile,sdir);
                            strcat(nfile,"\\");

                            MangleNames(nfile,FData.cFileName,NewName,fc.Header.playername);

                            if (!MoveFile(FData.cFileName,nfile))
                            {
                                sprintf(buffer,"Error moving file '%s' to '%s'!\nPress OK to continue.",FData.cFileName,
nfile);

                                ErrorBox(buffer);
                            }

                            fLoop = FindNextFile(hFind, &FData);
                        }

                        FindClose(hFind);
                    }

                    fc.setfilename(NewName);
                    strcpy(fc.Header.playername,NewName);
                }
                else
                {
                    fc.setfilename(NewName);
                    strcpy(fc.Header.playername,NewName);
                }

                EndDialog(hWnd,IDOK);
                return true;
            }
        case IDCANCEL:
            EndDialog(hWnd,wParam);
            return true;
        }
        break;
    case WM_CLOSE:
        EndDialog(hWnd,wParam);
        return true;
    case WM_DESTROY:
        return false;
    }
    return false;
}
```

```
// OptionsBox.cpp from D2E

#include "JamellaD2E.h"

static struct
{
    bool    Expert;
    int     DlgID;
    int     *Value;
}
CheckBoxes[] =
{
    { 0,    IDC_UOPTIONS_Associations,          &RegOptions.Associations },
    { 0,    IDC_UOPTIONS_ExceedQuantity,        &RegOptions.ExceedQuantity },
    { 0,    IDC_UOPTIONS_Tooltips,              &RegOptions.ToolTips },
    { 0,    IDC_UOPTIONS_AnnoyingMsgs,          &RegOptions.NoAnnoyingMsgs },

    { 1,    IDC_UOPTIONS_AllSocktable,          &RegOptions.AllItemsSocketable },
    { 1,    IDC_EOPTIONS_7Gems,                 &RegOptions.A7Gems },
};


LRESULT CALLBACK UOptionsDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
    switch(uMsg)
    {
    case WM_INITDIALOG:
        {
            for(int u = 0;u < sizeof CheckBoxes / sizeof CheckBoxes[0];u++)
            {
                if (CheckBoxes[u].Expert) continue;

                CheckDlgButton(hWnd,CheckBoxes[u].DlgID,
                    *CheckBoxes[u].Value ? BST_CHECKED : BST_UNCHECKED);
            }
        }
        return true;
    case WM_COMMAND:
        switch (LOWORD(wParam))
        {
        case IDOK:
            {
                for(int u = 0;u < sizeof CheckBoxes / sizeof CheckBoxes[0];u++)
                {
                    if (CheckBoxes[u].Expert) continue;

                    if (IsDlgButtonChecked(hWnd,CheckBoxes[u].DlgID) == BST_CHECKED)
                        *CheckBoxes[u].Value = true;
                    else
                        *CheckBoxes[u].Value = false;
                }

                SaveEditorRegistryValues();
            }
            EndDialog(hWnd,IDOK);
            return true;
        case IDCANCEL:
            EndDialog(hWnd,IDCANCEL);
            return true;
        }
        break;
    case WM_CLOSE:
        EndDialog(hWnd,wParam);
        return true;
    case WM_DESTROY:
        return false;
    }
    return false;
}

LRESULT CALLBACK EOptionsDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
    switch(uMsg)
    {
    case WM_INITDIALOG:
        {
            CheckRadioButton(hWnd,IDC_OPTIONS_IFormat1,IDC_OPTIONS_IFormat3,
                IDC_OPTIONS_IFormat1 + RegOptions.CreateItemRecordFormat);
```

```
            CheckDlgButton(hWnd,IDC_UOPTIONS_AllSocktable,
                RegOptions.AllItemsSocketable ? BST_CHECKED : BST_UNCHECKED);
        }
        return true;
    case WM_COMMAND:
        switch (LOWORD(wParam))
        {
        case IDOK:
            {
                if (IsDlgButtonChecked(hWnd,IDC_UOPTIONS_AllSocktable) == BST_CHECKED)
                    RegOptions.AllItemsSocketable = true;
                else
                    RegOptions.AllItemsSocketable = false;


                RegOptions.CreateItemRecordFormat = 0;
                if (IsDlgButtonChecked(hWnd,IDC_OPTIONS_IFormat2) == BST_CHECKED)
                    RegOptions.CreateItemRecordFormat = 1;
                if (IsDlgButtonChecked(hWnd,IDC_OPTIONS_IFormat3) == BST_CHECKED)
                    RegOptions.CreateItemRecordFormat = 2;

                SaveEditorRegistryValues();
            }
            EndDialog(hWnd,IDOK);
            return true;
        case IDCANCEL:
            EndDialog(hWnd,IDCANCEL);
            return true;
        }
        break;
    case WM_CLOSE:
        EndDialog(hWnd,wParam);
        return true;
    case WM_DESTROY:
        return false;
    }
    return false;
}
```

```
#include "JamellaD2E.h"

const struct _RarePreSuffix RarePrefixTable[] =
{
{ "Beast", 65535 },
{ "Eagle", 65535 },
{ "Raven", 65535 },
{ "Viper", 65535 },
{ "Ghoul", 65535 },
{ "Skull", 65535 },
{ "Blood", 65535 },
{ "Dread", 65535 },
{ "Doom", 65535 },
{ "Grim", 65535 },
{ "Bone", 65535 },
{ "Death", 65535 },
{ "Shadow", 65535 },
{ "Storm", 65535 },
{ "Rune", 65535 },
{ "Plague", 65535 },
{ "Stone", 65535 },
{ "Wraith", 65535 },
{ "Spirit", 15239 },
{ "Storm", 16383 },
{ "Demon", 16383 },
{ "Cruel", 16383 },
{ "Empyrian", 896 },
{ "Bramble", 16383 },
{ "Pain", 16383 },
{ "Loath", 16383 },
{ "Glyph", 16383 },
{ "Imp", 16383 },
{ "Fiend", 16383 },
{ "Hailstone", 16383 },
{ "Gale", 16383 },
{ "Dire", 16383 },
{ "Soul", 16383 },
{ "Brimstone", 16383 },
{ "Corpse", 16383 },
{ "Carrion", 16383 },
{ "Armageddon", 31 },
{ "Havoc", 65535 },
{ "Bitter", 65535 },
{ "Entropy", 49152 },
{ "Chaos", 49152 },
{ "Order", 0 },
{ "Rule", 0 },
{ "Warp", 0 },
{ "Rift", 0 },
{ "Corruption", 49152 },
};

const struct _RarePreSuffix RareSuffixTable[] =
{
{ "Bite", 24 },
{ "Scratch", 72 },
{ "Scalpel", 8 },
{ "Fang", 72 },
{ "Gutter", 72 },
{ "Thirst", 1112 },
{ "Razor", 8 },
{ "Scythe", 24 },
{ "Edge", 24 },
{ "Saw", 8 },
{ "Splitter", 16 },
{ "Cleaver", 24 },
{ "Sever", 24 },
{ "Sunder", 16 },
{ "Rend", 16 },
{ "Mangler", 16 },
{ "Slayer", 16 },
{ "Reaver", 16 },
{ "Spawn", 16 },
{ "Gnash", 16 },
{ "Star", 416 },
{ "Blow", 416 },
{ "Smasher", 416 },
{ "Bane", 416 },
```

```
{ "Crusher", 416 },
{ "Breaker", 416 },
{ "Grinder", 416 },
{ "Crack", 416 },
{ "Mallet", 416 },
{ "Knell", 416 },
{ "Lance", 64 },
{ "Spike", 72 },
{ "Impaler", 72 },
{ "Skewer", 72 },
{ "Prod", 64 },
{ "Scourge", 64 },
{ "Wand", 0 },
{ "Wrack", 64 },
{ "Barb", 72 },
{ "Needle", 1096 },
{ "Dart", 1088 },
{ "Bolt", 1024 },
{ "Quarrel", 1024 },
{ "Fletch", 1024 },
{ "Flight", 1024 },
{ "Nock", 1024 },
{ "Horn", 1026 },
{ "Stinger", 1096 },
{ "Quill", 1024 },
{ "Goad", 576 },
{ "Branch", 1600 },
{ "Spire", 512 },
{ "Song", 1944 },
{ "Call", 896 },
{ "Cry", 896 },
{ "Spell", 896 },
{ "Chant", 896 },
{ "Weaver", 896 },
{ "Gnarl", 896 },
{ "Visage", 2 },
{ "Crest", 2 },
{ "Circlet", 2 },
{ "Veil", 2 },
{ "Hood", 2 },
{ "Mask", 2 },
{ "Brow", 2 },
{ "Casque", 2 },
{ "Visor", 2 },
{ "Cowl", 2 },
{ "Hide", 1 },
{ "Pelt", 1 },
{ "Carapace", 1 },
{ "Coat", 1 },
{ "Wrap", 1 },
{ "Suit", 1 },
{ "Cloak", 1 },
{ "Shroud", 1 },
{ "Jack", 1 },
{ "Mantle", 1 },
{ "Guard", 4 },
{ "Badge", 4 },
{ "Rock", 4 },
{ "Aegis", 4 },
{ "Ward", 4 },
{ "Tower", 4 },
{ "Shield", 4 },
{ "Wing", 4 },
{ "Mark", 4 },
{ "Emblem", 4 },
{ "Hand", 4096 },
{ "Fist", 4096 },
{ "Claw", 4096 },
{ "Clutches", 4096 },
{ "Grip", 20480 },
{ "Grasp", 20480 },
{ "Hold", 20480 },
{ "Touch", 20480 },
{ "Finger", 20480 },
{ "Knuckle", 20480 },
{ "Shank", 2048 },
{ "Spur", 2048 },
{ "Tread", 2048 },
{ "Stalker", 2048 },
```

```
{ "Greave", 2048 },
{ "Blazer", 2048 },
{ "Nails", 2112 },
{ "Trample", 2048 },
{ "Brogues", 2048 },
{ "Track", 2048 },
{ "Slippers", 2048 },
{ "Clasp", 8192 },
{ "Buckle", 8192 },
{ "Harness", 8192 },
{ "Lock", 8192 },
{ "Fringe", 8192 },
{ "Winding", 8192 },
{ "Chain", 8192 },
{ "Strap", 8192 },
{ "Lash", 8192 },
{ "Cord", 8192 },
{ "Knot", 16384 },
{ "Circle", 16384 },
{ "Loop", 16384 },
{ "Eye", 16384 },
{ "Turn", 16384 },
{ "Spiral", 16384 },
{ "Coil", 16384 },
{ "Gyre", 16384 },
{ "Band", 16384 },
{ "Whorl", 16384 },
{ "Talisman", 32768 },
{ "Heart", 32768 },
{ "Noose", 32768 },
{ "Necklace", 32768 },
{ "Collar", 32768 },
{ "Beads", 32768 },
{ "Torc", 32768 },
{ "Gorget", 32768 },
{ "Scarab", 32768 },
{ "Wood", 832 },
{ "Brand", 1016 },
{ "Bludgeon", 928 },
{ "Cudgel", 928 },
{ "Loom", 1024 },
{ "Harp", 1024 },
{ "Master", 49152 },
{ "Bar", 992 },
{ "Hew", 24 },
{ "Crook", 512 },
{ "Mar", 120 },
{ "Shell", 7 },
{ "Stake", 64 },
{ "Picket", 64 },
{ "Pale", 64 },
{ "Flange", 161 },
};

int nRarePrefixTable = sizeof RarePrefixTable / sizeof RarePrefixTable[0];
int nRareSuffixTable = sizeof RareSuffixTable / sizeof RareSuffixTable[0];
```

```
// Registry.cpp

#include "JamellaD2E.h"

static int GetUserConfirmation()
{
    return MessageBox(NULL,"The file types for *.d2s and *.d2i have not been registered in the explorer.\nDo you wish to
 do this now?\nThis will make the files have nice icons and double-click open features.",PROGRAMNAME,MB_YESNO | MB_ICONE
XCLAMATION);
}

static bool Confirmation;

static int CheckFileExtention(const char *Extention,const char *ProgID)
{
    char Query[260];
    DWORD Type;
    DWORD Size = sizeof Query;

    // Check Entry
    HKEY Key;
    if (RegOpenKeyEx(HKEY_CLASSES_ROOT,Extention,0,KEY_ALL_ACCESS,&Key) == ERROR_SUCCESS)
    { // Check Key Contents

        if (RegQueryValueEx(Key,NULL,NULL,&Type,(unsigned char*)&Query,&Size) == ERROR_SUCCESS &&
            Type == REG_SZ &&
            strcmp(Query,ProgID) == 0)
        {
        }
        else
        {
            if (!Confirmation) {
                if (GetUserConfirmation() != IDYES) return false;
                else Confirmation = true;
            }

            RegCreateKeyEx(HKEY_CLASSES_ROOT,Extention,0,NULL,REG_OPTION_NON_VOLATILE,KEY_ALL_ACCESS,0,&Key,NULL);
            RegSetValueEx(Key,NULL,0,REG_SZ,(unsigned char*)ProgID,strlen(ProgID));
        }

    } // Check Key Contents
    else
    { // Add new entry

        if (!Confirmation) {
            if (GetUserConfirmation() != IDYES) return false;
            else Confirmation = true;
        }

        RegCreateKeyEx(HKEY_CLASSES_ROOT,Extention,0,NULL,REG_OPTION_NON_VOLATILE,KEY_ALL_ACCESS,0,&Key,NULL);
        RegSetValueEx(Key,NULL,0,REG_SZ,(unsigned char*)ProgID,strlen(ProgID));

    } // Add new entry
    RegCloseKey(Key);
    return true;
}

static int CheckDescription(const char *ProgID,const char *Description)
{
    char Query[260];
    DWORD Type;
    DWORD Size = sizeof Query;

    // Check Entry
    HKEY Key;
    if (RegOpenKeyEx(HKEY_CLASSES_ROOT,ProgID,0,KEY_ALL_ACCESS,&Key) == ERROR_SUCCESS)
    { // Check Key Contents

        if (RegQueryValueEx(Key,NULL,NULL,&Type,(unsigned char*)&Query,&Size) == ERROR_SUCCESS &&
            Type == REG_SZ &&
            strcmp(Query,Description) == 0)
        {
        }
        else
        {
            if (!Confirmation) {
                if (GetUserConfirmation() != IDYES) return false;
                else Confirmation = true;
            }
```

```
            RegCreateKeyEx(HKEY_CLASSES_ROOT,ProgID,0,NULL,REG_OPTION_NON_VOLATILE,KEY_ALL_ACCESS,0,&Key,NULL);
            RegSetValueEx(Key,NULL,0,REG_SZ,(unsigned char*)Description,strlen(Description));
        }

    } // Check Key Contents
    else
    { // Add new entry

        if (!Confirmation) {
            if (GetUserConfirmation() != IDYES) return false;
            else Confirmation = true;
        }

        RegCreateKeyEx(HKEY_CLASSES_ROOT,ProgID,0,NULL,REG_OPTION_NON_VOLATILE,KEY_ALL_ACCESS,0,&Key,NULL);
        RegSetValueEx(Key,NULL,0,REG_SZ,(unsigned char*)Description,strlen(Description));

    } // Add new entry
    RegCloseKey(Key);
    return true;
}

static int CheckIcon(const char *ProgID,const char *IconPath,const int IconNum)
{
    char Query[260];
    DWORD Type;
    DWORD Size = sizeof Query;

    char KeyPath[260];
    sprintf(KeyPath,"%s\\DefaultIcon",ProgID);

    char FilePath[260];
    sprintf(FilePath,"%s,%i",IconPath,IconNum);

    // Check Entry
    HKEY Key;
    if (RegOpenKeyEx(HKEY_CLASSES_ROOT,KeyPath,0,KEY_ALL_ACCESS,&Key) == ERROR_SUCCESS)
    { // Check Key Contents

        if (RegQueryValueEx(Key,NULL,NULL,&Type,(unsigned char*)&Query,&Size) == ERROR_SUCCESS &&
            Type == REG_SZ &&
            strcmp(Query,FilePath) == 0)
        {
        }
        else
        {
            if (!Confirmation) {
                if (GetUserConfirmation() != IDYES) return false;
                else Confirmation = true;
            }

            RegCreateKeyEx(HKEY_CLASSES_ROOT,KeyPath,0,NULL,REG_OPTION_NON_VOLATILE,KEY_ALL_ACCESS,0,&Key,NULL);
            RegSetValueEx(Key,NULL,0,REG_SZ,(unsigned char*)FilePath,strlen(FilePath));
        }

    } // Check Key Contents
    else
    { // Add new entry

        if (!Confirmation) {
            if (GetUserConfirmation() != IDYES) return false;
            else Confirmation = true;
        }

        RegCreateKeyEx(HKEY_CLASSES_ROOT,KeyPath,0,NULL,REG_OPTION_NON_VOLATILE,KEY_ALL_ACCESS,0,&Key,NULL);
        RegSetValueEx(Key,NULL,0,REG_SZ,(unsigned char*)FilePath,strlen(FilePath));

    } // Add new entry
    RegCloseKey(Key);
    return true;
}

static int CheckShellVerb(const char *ProgID,const char *Verb,const char *Command)
{
    char Query[260];
    DWORD Type;
    DWORD Size = sizeof Query;

    char KeyPath[260];
```

```
    sprintf(KeyPath,"%s\\shell\\%s\\command",ProgID,Verb);

    // Check Entry
    HKEY Key;
    if (RegOpenKeyEx(HKEY_CLASSES_ROOT,KeyPath,0,KEY_ALL_ACCESS,&Key) == ERROR_SUCCESS)
    { // Check Key Contents

        if (RegQueryValueEx(Key,NULL,NULL,&Type,(unsigned char*)&Query,&Size) == ERROR_SUCCESS &&
            Type == REG_SZ &&
            strcmp(Query,Command) == 0)
        {
        }
        else
        {
            if (!Confirmation) {
                if (GetUserConfirmation() != IDYES) return false;
                else Confirmation = true;
            }

            sprintf(KeyPath,"%s\\shell",ProgID);
            RegCreateKeyEx(HKEY_CLASSES_ROOT,KeyPath,0,NULL,REG_OPTION_NON_VOLATILE,KEY_ALL_ACCESS,0,&Key,NULL);
            RegCloseKey(Key);

            sprintf(KeyPath,"%s\\shell\\%s",ProgID,Verb);
            RegCreateKeyEx(HKEY_CLASSES_ROOT,KeyPath,0,NULL,REG_OPTION_NON_VOLATILE,KEY_ALL_ACCESS,0,&Key,NULL);
            RegCloseKey(Key);

            sprintf(KeyPath,"%s\\shell\\%s\\command",ProgID,Verb);
            RegCreateKeyEx(HKEY_CLASSES_ROOT,KeyPath,0,NULL,REG_OPTION_NON_VOLATILE,KEY_ALL_ACCESS,0,&Key,NULL);
            RegSetValueEx(Key,NULL,0,REG_SZ,(unsigned char*)Command,strlen(Command));
        }

    } // Check Key Contents
    else
    { // Add new entry

        if (!Confirmation) {
            if (GetUserConfirmation() != IDYES) return false;
            else Confirmation = true;
        }

        sprintf(KeyPath,"%s\\shell",ProgID);
        RegCreateKeyEx(HKEY_CLASSES_ROOT,KeyPath,0,NULL,REG_OPTION_NON_VOLATILE,KEY_ALL_ACCESS,0,&Key,NULL);
        RegCloseKey(Key);

        sprintf(KeyPath,"%s\\shell\\%s",ProgID,Verb);
        RegCreateKeyEx(HKEY_CLASSES_ROOT,KeyPath,0,NULL,REG_OPTION_NON_VOLATILE,KEY_ALL_ACCESS,0,&Key,NULL);
        RegCloseKey(Key);

        sprintf(KeyPath,"%s\\shell\\%s\\command",ProgID,Verb);
        RegCreateKeyEx(HKEY_CLASSES_ROOT,KeyPath,0,NULL,REG_OPTION_NON_VOLATILE,KEY_ALL_ACCESS,0,&Key,NULL);
        RegSetValueEx(Key,NULL,0,REG_SZ,(unsigned char*)Command,strlen(Command));

    } // Add new entry
    RegCloseKey(Key);
    return true;
}

void CheckShellRegistry()
{
    if (!RegOptions.Associations) return;

    Confirmation = false;

    if (!CheckFileExtention(".d2s",D2SPROGID)) return;
    if (!CheckFileExtention(".d2i",D2IPROGID)) return;
    if (!CheckFileExtention(".item",D2IPROGID)) return;
    if (!CheckFileExtention(".ite",D2IPROGID)) return;
    if (!CheckFileExtention(".itm",D2IPROGID)) return;

    if (!CheckDescription(D2SPROGID,D2SDESCRIPTION)) return;
    if (!CheckDescription(D2IPROGID,D2IDESCRIPTION)) return;

    if (!CheckIcon(D2SPROGID,ProgramFilePath(),1)) return;
    if (!CheckIcon(D2IPROGID,ProgramFilePath(),2)) return;

    char command[260];
    sprintf(command,"\"%s\" \"%%1\"",ProgramFilePath());
```

```
    if (!CheckShellVerb(D2SPROGID,"open",command)) return;

    SHChangeNotify(SHCNE_ASSOCCHANGED,0,0,0);
}

struct _RegOptions      RegOptions;

struct
{
    char*       ValName;
    DWORD       Type;
    BYTE*       Ptr;
    DWORD       PtrSize;
    int         Default;
}
QueryValues[] =
{
    { "CreateItemRecordFormat",
        REG_DWORD,
        (BYTE*)&RegOptions.CreateItemRecordFormat,
        sizeof RegOptions.CreateItemRecordFormat,
        0 },

    { "AllItemsSocketable",
        REG_DWORD,
        (BYTE*)&RegOptions.AllItemsSocketable,
        sizeof RegOptions.AllItemsSocketable,
        0 },

    { "7Gems",
        REG_DWORD,
        (BYTE*)&RegOptions.A7Gems,
        sizeof RegOptions.A7Gems,
        0 },

    { "ExceedQuantity",
        REG_DWORD,
        (BYTE*)&RegOptions.ExceedQuantity,
        sizeof RegOptions.ExceedQuantity,
        0 },

    { "ItemPath",
        REG_SZ,
        (BYTE*)RegOptions.ItemPath,
        sizeof RegOptions.ItemPath,
        0 },

    { "Associations",
        REG_DWORD,
        (BYTE*)&RegOptions.Associations,
        sizeof RegOptions.Associations,
        1 },

    { "ToolTips",
        REG_DWORD,
        (BYTE*)&RegOptions.ToolTips,
        sizeof RegOptions.ToolTips,
        1 },

    { "AnnoyingMsgs",
        REG_DWORD,
        (BYTE*)&RegOptions.NoAnnoyingMsgs,
        sizeof RegOptions.NoAnnoyingMsgs,
        0 },
};


void LoadEditorRegistryValues()
{
    atexit(&SaveEditorRegistryValues);

    HKEY Key;
    if (RegOpenKeyEx(HKEY_CURRENT_USER,REGISTRYKEY,0,KEY_ALL_ACCESS,&Key) != ERROR_SUCCESS) {

        for(int z=0;z<sizeof QueryValues / sizeof QueryValues[0];z++)
        {
            if (QueryValues[z].Type == REG_DWORD) {
                *(DWORD *)QueryValues[z].Ptr = QueryValues[z].Default;
            }
```

```
        }
        return;
    }

    DWORD Type;

    for(int z=0;z<sizeof QueryValues / sizeof QueryValues[0];z++)
    {
        if (RegQueryValueEx(Key,QueryValues[z].ValName,0,&Type,NULL,NULL) != ERROR_SUCCESS)
        {
            if (QueryValues[z].Type == REG_DWORD) {
                *(DWORD *)QueryValues[z].Ptr = QueryValues[z].Default;
            }
        }
        else if (Type == QueryValues[z].Type)
        {
            if (RegQueryValueEx(Key,QueryValues[z].ValName,0,&Type,QueryValues[z].Ptr,&QueryValues[z].PtrSize) != ERROR_
SUCCESS) {

                if (QueryValues[z].Type == REG_DWORD) {
                    *(DWORD *)QueryValues[z].Ptr = QueryValues[z].Default;
                }

            }
        }
    }

    RegCloseKey(Key);
}

void SaveEditorRegistryValues()
{
    HKEY Key;
    if (RegCreateKeyEx(HKEY_CURRENT_USER,REGISTRYKEY,0,NULL,REG_OPTION_NON_VOLATILE,KEY_ALL_ACCESS,0,&Key,NULL) != ERROR
_SUCCESS) return;

    for(int z=0;z<sizeof QueryValues / sizeof QueryValues[0];z++)
    {
        RegSetValueEx(Key,QueryValues[z].ValName,0,QueryValues[z].Type,QueryValues[z].Ptr,QueryValues[z].PtrSize);
    }

    RegCloseKey(Key);
}
```

```
// File.cpp from D2E

#include "JamellaD2E.h"

// Resources
HICON       hIconHelp;

// Progress Bar Macros
inline void ProgressText(const char *text)
{
    SetDlgItemText(hTabDialog,IDC_PROGRESS_Text,text);
}
inline void ProgressBarRange(int min,int max)
{
    SendDlgItemMessage(hTabDialog,IDC_PROGRESS_Bar,PBM_SETRANGE,0,MAKELPARAM(min,max));
}
inline void ProgressBarSet(int i)
{
    SendDlgItemMessage(hTabDialog,IDC_PROGRESS_Bar,PBM_SETPOS,i,0);
}
inline void ProgressBarSetStep(int i)
{
    SendDlgItemMessage(hTabDialog,IDC_PROGRESS_Bar,PBM_SETSTEP,i,0);
}
inline void ProgressBarStep()
{
    SendDlgItemMessage(hTabDialog,IDC_PROGRESS_Bar,PBM_STEPIT,0,0);
    PollMessages();
}

HMENU hBatchMenu;

// Bitmaps
struct
{
    int         Type;
    int         ID;
    HANDLE      *Handle;
}
rc[] =
{
    { IMAGE_BITMAP, IDB_JAMELLA,            (HANDLE*)&hBmpJamella },
    { IMAGE_BITMAP, IDB_PLUS,               (HANDLE*)&hBmpPlus },
    { IMAGE_BITMAP, IDB_WAYPOINT_ON,        (HANDLE*)&hBmpWaypointOn },
    { IMAGE_BITMAP, IDB_WAYPOINT_OFF,       (HANDLE*)&hBmpWaypointOff },
    { IMAGE_BITMAP, IDB_ITEM_UNKNOWN,       (HANDLE*)&itemunknown.hBmp },
    { IMAGE_BITMAP, IDB_NOTPLACEABLE,       (HANDLE*)&hBmpNotPlaceable },
    { IMAGE_BITMAP, IDB_INV_WHOLE,          (HANDLE*)&hBmpBodyWhole },
    { IMAGE_BITMAP, IDB_WEBLINK,            (HANDLE*)&hBmpWebLink },
    { IMAGE_ICON,   IDI_CHELP,              (HANDLE*)&hIconHelp },
    { IMAGE_CURSOR, IDC_CUR_MOVE,           (HANDLE*)&hCurMove },
    { IMAGE_CURSOR, IDC_CUR_MOVECOPY,       (HANDLE*)&hCurMoveCopy },
    { IMAGE_CURSOR, IDC_CUR_CROSS,          (HANDLE*)&hCurCross },
    { IMAGE_CURSOR, IDC_CUR_ADD,            (HANDLE*)&hCurAdd },
    { IMAGE_CURSOR, IDC_CUR_NO,             (HANDLE*)&hCurNo },
};

void D2ELoadResources()
{
    ProgressBarSetStep(1);

    int ImgNum = 0;
    ImgNum += 30*5;
    ImgNum += 21;
#if PRELOADITEMIMAGES == 1
    ImgNum += nItemInfos;
#endif
    ImgNum += 5+3;
    ImgNum += sizeof rc / sizeof rc[0];

    ProgressBarRange(0,ImgNum);

    ProgressText("Loading Skill Bitmaps...");
    for(int z=0;z<30*5;z++)
    {
        skills[z].hbitmap = LoadBitmap(hInstance,MAKEINTRESOURCE(skills[z].idbitmap));
        ProgressBarStep();
    }
```

```
    ProgressText("Loading Quest Bitmaps...");
    for(z=0;z<21;z++)
    {
        quests[z].hbitmap = LoadBitmap(hInstance,MAKEINTRESOURCE(quests[z].idbitmap));
        if (!quests[z].hbitmap)
        {
            ErrorMessage();
            exit(0);
        }
        LoadString(hInstance,quests[z].idstring,quests[z].string,60);
        ProgressBarStep();
    }

    ProgressText("Loading Item Bitmaps...");
#if PRELOADITEMIMAGES == 1
    for(z=0;z<nItemInfos;z++)
    {
        if (ItemInfos[z].BitmapID)
        {
            ItemInfos[z].hBmp = LoadBitmap(hInstance,MAKEINTRESOURCE(ItemInfos[z].BitmapID));

            if (ItemInfos[z].hBmp)
            {
                BITMAP bmpinfo;
                if (GetObject(ItemInfos[z].hBmp,sizeof bmpinfo,&bmpinfo) == 0)
                {
                    ErrorMessage();
                    exit(0);
                }

                if (bmpinfo.bmWidth != ItemInfos[z].SizeX * 29 - 1 ||
                    bmpinfo.bmHeight != ItemInfos[z].SizeY * 29 - 1)
                {
                    sprintf(buffer,"Resource Error [%i|%i]\nID = %i",bmpinfo.bmWidth,bmpinfo.bmHeight,ItemInfos[z].Bitma
pID);
                    MessageBox(hTabDialog,buffer,PROGRAMNAME,
                        MB_OK | MB_ICONSTOP | MB_APPLMODAL);
                    exit(0);
                }
            }
            else
            {
                ErrorMessage();
                exit(0);
            }
        }
        ProgressBarStep();
    }
#endif

    for(z=0;z<5;z++)
    {
        RingImages[z].hBmp = LoadBitmap(hInstance,MAKEINTRESOURCE(RingImages[z].BmpID));
        if (!RingImages[z].hBmp)
        {
            ErrorMessage();
            exit(0);
        }
        ProgressBarStep();
    }
    for(z=0;z<3;z++)
    {
        AmuletImages[z].hBmp = LoadBitmap(hInstance,MAKEINTRESOURCE(AmuletImages[z].BmpID));
        if (!AmuletImages[z].hBmp)
        {
            ErrorMessage();
            exit(0);
        }
        ProgressBarStep();
    }

    ProgressText("Loading Miscellaneous Resources...");
    for(z=0;z<sizeof rc / sizeof rc[0];z++)
    {
        *rc[z].Handle = LoadImage(hInstance,MAKEINTRESOURCE(rc[z].ID),rc[z].Type,0,0,LR_DEFAULTCOLOR | LR_SHARED);
        if (!*rc[z].Handle)
        {
            ErrorMessage();
            exit(0);
```

```
        }
        ProgressBarStep();
    }


    hBatchMenu = LoadMenu(hInstance,MAKEINTRESOURCE(IDR_BATCH));

    hBrushBlack = (HBRUSH) GetStockObject(BLACK_BRUSH);
    hBrushNull = (HBRUSH) GetStockObject(NULL_BRUSH);
    hPenWhite = CreatePen(PS_SOLID,1,RGB(240,240,240));
    hPenGreen = CreatePen(PS_SOLID,2,RGB(0,255,0));
    hPenBusy = CreatePen(PS_SOLID,2,RGB(255,0,0));
    hPenUnwearable = CreatePen(PS_SOLID,1,RGB(255,0,0));
}

/*
            // Imagelist for ItemTree
            if ((hTVImagelist = ImageList_Create(16,16,ILC_COLORDDB,3,0)) == NULL) return false;

            HDC hdcbitmap = CreateCompatibleDC(NULL);
            HDC hdcbitmaps = CreateCompatibleDC(NULL);

            HBITMAP hbmp = LoadBitmap(hInstance,MAKEINTRESOURCE(IDB_ITEM_1170_S_00));
            SelectObject(hdcbitmap, hbmp);

            HBITMAP hsbmp = CreateCompatibleBitmap(hdcbitmap,16,16);
            SelectObject(hdcbitmaps, hsbmp);

            if(!StretchBlt(hdcbitmaps,0,0,16,16,hdcbitmap,0,0,28,28,SRCCOPY)) OutputDebugString("FUCK\n");

            DeleteDC(hdcbitmap);
            DeleteDC(hdcbitmaps);

            BITMAP info;
            GetObject(hsbmp,sizeof(info),&info);

            ImageList_Add(hTVImagelist,hsbmp,NULL);
            DeleteObject(hbmp);
            DeleteObject(hsbmp);
*/

void D2EUnLoadResources()
{
    for(int z=0;z<30*5;z++)
    {
        if (skills[z].hbitmap)
            DeleteObject(skills[z].hbitmap);
    }
    for(z=0;z<21;z++)
    {
        if (quests[z].hbitmap)
            DeleteObject(quests[z].hbitmap);
    }
    for(z=0;z<nItemInfos;z++)
    {
        if (ItemInfos[z].hBmp)
            DeleteObject(ItemInfos[z].hBmp);
    }
    for(z=0;z<5;z++)
    {
        if (RingImages[z].hBmp)
            DeleteObject(RingImages[z].hBmp);
    }
    for(z=0;z<3;z++)
    {
        if (AmuletImages[z].hBmp)
            DeleteObject(AmuletImages[z].hBmp);
    }

    for(z=0;z<sizeof rc / sizeof rc[0];z++)
    {
        if (rc[z].Handle)
            DeleteObject(*rc[z].Handle);
    }


    DestroyMenu(hBatchMenu);

    DeleteObject(hBrushBlack);
```

```
    DeleteObject(hPenWhite);
    DeleteObject(hPenGreen);
    DeleteObject(hPenBusy);
    DeleteObject(hPenUnwearable);
}
```

```
#include "JamellaD2E.h"

struct ImageMap RingImages[] =
{
    { IDB_ITEM_RING3 },
    { IDB_ITEM_RING4 },
    { IDB_ITEM_RING5 },
    { IDB_ITEM_RING1 },
    { IDB_ITEM_RING2 },
};
struct ImageMap AmuletImages[] =
{
    { IDB_ITEM_AMULET1 },
    { IDB_ITEM_AMULET2 },
    { IDB_ITEM_AMULET3 },
};
```

```
#include "JamellaD2E.h"

LRESULT CALLBACK SaveDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
    switch(uMsg)
    {
    case WM_INITDIALOG:
        {

        }
        return true;
    case WM_COMMAND:
        switch (LOWORD(wParam))
        {
        case IDOK:
            if (IsDlgButtonChecked(hWnd,IDC_SAVE_Backup) == BST_CHECKED)
                EndDialog(hWnd,2);
            else
                EndDialog(hWnd,1);

            return true;
        case IDCANCEL:
            EndDialog(hWnd,0);
            return true;
        }
        break;
    case WM_CLOSE:
        EndDialog(hWnd,0);
        return true;
    case WM_DESTROY:
        return false;
    }
    return false;
}
```

```
#include "JamellaD2E.h"

const struct _SetItem SetItems[] =
{
{ "Civerb's Vestments",0x0,"Civerb's",13,3,8,1,13,0x0C70,0x2067726C,"Ward",0x1170,0x20756D61,"Icon",0x0100,0x20637367,"C
udgel",0x0,0xFFFFFFFF,0,0x0,0xFFFFFFFF,0,0x0,0xFFFFFFFF,0,'ar00',15,15,'bl00',15,15,'hl11',40,40,'lf00',4,4,'ht00',75,75
,'dm00',17,23,'st00',15,15,'md01',200,200,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },
{ "Hsarus' Trim",0x2,"Hsarus'",4,3,9,1,8,0x0D20,0x2074626D,"Iron Heel",0x0C50,0x20637562,"Iron Fist",0x0D70,0x206C626D,"
Iron Stay",0x0,0xFFFFFFFF,0,0x0,0xFFFFFFFF,0,0x0,0xFFFFFFFF,0,'rf00',25,25,'ve00',10,10,'ar01',2,2,'st00',10,10,'rc00',2
0,20,'mh00',20,20,'dm00',5,5,'sp10',1,1,'rl00',25,25,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },
{ "Cleglaw's Brace",0x4,"Cleglaw's",6,3,9,1,7,0x01F0,0x2064736C,"Tooth",0x0C60,0x206C6D73,"Claw",0x0CD0,0x206C676D,"Pinc
ers",0x0,0xFFFFFFFF,0,0x0,0xFFFFFFFF,0,0x0,0xFFFFFFFF,0,'ht01',30,30,'ds00',50,50,'ar00',17,17,'pd00',75,75,'ko00',1,1,'
sp07',25,25,'ar00',50,50,'mm03',6,6,'sp03',35,35,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },
{ "Iratha's Finery",0x6,"Iratha's",21,4,14,1,2,0x1170,0x20756D61,"Collar",0x0CE0,0x206C6774,"Cuff",0x0B40,0x206E7263,"Co
il",0x0D80,0x206C6274,"Cord",0x0,0xFFFFFFFF,0,0x0,0xFFFFFFFF,0,'rp00',30,30,'pd00',75,75,'rc00',30,30,'dc02',1,1,'rf00',
30,30,'rl00',30,30,'ar00',25,25,'dm01',5,5,'ra00',20,20,'rf01',10,10,'rc01',10,10,'rl01',10,10,'rp01',10,10,'dx00',15,15
,0,0,0,0,0,0 },
{ "Isenhart's Armory",0x8,"Isenhart's",11,4,11,1,15,0x01E0,0x20647362,"Lightbrand",0x0CA0,0x20737467,"Parry",0x0BD0,0x20
737262,"Case",0x0B20,0x206C6866,"Horns",0x0,0xFFFFFFFF,0,0x0,0xFFFFFFFF,0,'dm01',10,10,'lt00',3,3,'ar00',40,40,'at04',4,
4,'ar00',40,40,'ar03',2,2,'dx00',6,6,'ar01',2,2,'mh03',5,5,'ra00',20,20,'ht01',35,35,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },
{ "Vidala's Rig",0xA,"Vidala's",19,4,11,1,4,0x0490,0x2062626C,"Barb",0x0D30,0x20746274,"Fetlock",0x0B70,0x2061656C,"Ambu
sh",0x1170,0x20756D61,"Snare",0x0,0xFFFFFFFF,0,0x0,0xFFFFFFFF,0,'dl00',1,1,'dl01',20,20,'hl12',150,150,'ve02',30,30,'ar0
0',50,50,'dx00',11,11,'mh00',15,15,'rc00',20,20,'dm02',100,100,'sp01',50,50,'sp13',50,50,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,
{ "Milabrega's Regalia",0xC,"Milabrega's",23,4,12,1,5,0x0C80,0x2074696B,"Orb",0x0110,0x20707377,"Rod",0x0B40,0x206E7263,
"Diadem",0x0C30,0x20726161,"Robe",0x0,0xFFFFFFFF,0,0x0,0xFFFFFFFF,0,'ib01',20,20,'ar00',25,25,'sk01',1,1,'dm02',50,50,'m
h00',15,15,'mm00',15,15,'at00',3,3,'ar01',2,2,'mh03',8,8,'sk01',2,2,'lt00',2,2,'rp00',15,15,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },
{ "Cathan's Traps",0xE,"Cathan's",15,5,15,1,11,0x0420,0x20747362,"Rule",0x0BC0,0x206E6863,"Mesh",0x0B50,0x206B736D,"Visa
ge",0x1170,0x20756D61,"Sigil",0x1190,0x206E6972,"Seal",0x0,0xFFFFFFFF,0,'sk07',1,1,'df01',10,10,'ar00',15,15,'rq00',-50,
-50,'mm00',20,20,'rc00',25,25,'gh00',10,10,'at04',5,5,'mh03',6,6,'ar01',2,2,'ht00',60,60,'ar03',3,3,'ra00',25,25,'ca00',
10,10,'mm00',20,20,0,0,0 },
{ "Tancred's Battlegear",0x10,"Tancred's",27,5,14,1,16,0x0030,0x2069706D,"Crowbill",0x0C20,0x206C7566,"Spine",0x0D00,0x2
074626C,"Hobnails",0x1170,0x20756D61,"Weird",0x0DA0,0x206D6862,"Skull",0x0,0xFFFFFFFF,0,'ht00',75,75,'dm02',80,80,'mh00'
,40,40,'st00',15,15,'hl08',25,25,'dx00',10,10,'ar01',2,2,'ar03',1,1,'dm02',10,10,'ht00',40,40,'ib00',75,75,'ra00',10,10,
'sp07',35,35,'mm03',5,5,0,0,0,0,0,0 },
{ "Sigon's Complete Steel",0x12,"Sigon's",9,6,16,1,0,0x0B30,0x206D6867,"Visor",0x0C10,0x20687467,"Shelter",0x0CF0,0x206C
6768,"Gage",0x0D40,0x20746268,"Sabot",0x0D90,0x206C6268,"Wrap",0x0C90,0x20776F74,"Guard",'mm00',30,30,'lt00',3,3,'ar02',
25,25,'rl00',30,30,'st00',10,10,'ht00',20,20,'ve01',20,20,'rc00',40,40,'rf00',20,20,'mh00',20,20,'sk08',1,1,'bl00',20,20
,'df00',12,12,'at00',12,12,'ar01',7,7,'df01',24,24 },
{ "Infernal Tools",0x14,"Infernal",7,3,9,1,13,0x0AF0,0x20706163,"Cranium",0x00D0,0x206E7767,"Torch",0x0D80,0x206C6274,"B
uckle",0x0,0xFFFFFFFF,0,0x0,0xFFFFFFFF,0,0x0,0xFFFFFFFF,0,'ra00',10,10,'dm03',20,20,'dm01',8,8,'sk02',1,1,'ar00',25,25,'
mh00',20,20,'ht01',20,20,'sk02',1,1,'sp02',20,20,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },
{ "Berserker's Garb",0x16,"Berserker's",5,3,10,1,8,0x0B10,0x206D6C68,"Headgear",0x0BE0,0x206C7073,"Hauberk",0x0020,0x207
86132,"Hatchet",0x0,0xFFFFFFFF,0,0x0,0xFFFFFFFF,0,0x0,0xFFFFFFFF,0,'ar00',15,15,'rf00',25,25,'ar03',2,2,'sk04',1,1,'ht01
',30,30,'mm03',5,5,'pd00',75,75,'dp00',16,16,'dp01',32,32,'dp02',75,75,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },
{ "Death's Disguise",0x18,"Death's",8,3,9,1,7,0x0CB0,0x206C676C,"Hand",0x0D50,0x206C626C,"Guard",0x0200,0x20647377,"Touc
h",0x0,0xFFFFFFFF,0,0x0,0xFFFFFFFF,0,0x0,0xFFFFFFFF,0,'rp00',50,50,'pd00',75,75,'ar00',20,20,'sp10',1,1,'dm02',25,25,'mh
03',4,4,'ht01',40,40,'dm01',10,10,'ra00',25,25,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },
{ "Angelical Raiment",0x1A,"Angelical",17,4,11,1,1,0x01B0,0x20726273,"Sickle",0x0BA0,0x20676E72,"Mantle",0x1190,0x206E69
72,"Halo",0x1170,0x20756D61,"Wings",0x0,0xFFFFFFFF,0,0x0,0xFFFFFFFF,0,'ht00',75,75,'md01',250,250,'ar01',3,3,'ar02',40,4
0,'lf00',6,6,'mh00',20,20,'lt00',3,3,'dm03',20,20,'ra00',25,25,'dc02',1,1,'ib01',40,40,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },
{ "Arctic Gear",0x1C,"Arctic",3,4,12,1,15,0x04A0,0x20627773,"Horn",0x0B60,0x20697571,"Furs",0x0D60,0x206C6276,"Binding",
0x0CE0,0x206C6774,"Mitts",0x0,0xFFFFFFFF,0,0x0,0xFFFFFFFF,0,'ht01',20,20,'dm02',20,20,'ar02',275,325,'ra00',10,10,'rc00'
,40,40,'ar00',30,30,'mh00',20,20,'at01',10,10,'dc00',6,6,'dc01',14,14,'dc03',100,100,'sp10',1,1,0,0,0,0,0,0,0,0,0,0,0,0
},
{ "Arcanna's Tricks",0x1E,"Arcanna's",20,4,11,1,4,0x1170,0x20756D61,"Sign",0x0430,0x20747377,"Deathwand",0x0B00,0x20706B
73,"Head",0x0C40,0x2070746C,"Flesh",0x0,0xFFFFFFFF,0,0x0,0xFFFFFFFF,0,'mm00',15,15,'hl11',20,20,'sk03',1,1,'ds00',25,25,
'lf00',4,4,'at00',2,2,'lt00',2,2,'ar01',3,3,'ca01',20,20,'mm03',5,5,'mm00',25,25,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },
};

int nSetItems = sizeof SetItems / sizeof SetItems[0];
```

```
#include "JamellaD2E.h"

const struct _SuperiorItem SuperiorItemTable[] =
{
{ 0,1,'ht00',1,3,0,0,0,250,1,2,100 },
{ 1,1,'dm02',5,15,0,0,0,250,1,3,250 },
{ 2,1,'ar02',5,15,0,0,0,1797,1,2,200 },
{ 3,2,'ht00',1,3,'dm02',5,15,250,3,5,350 },
{ 4,1,'du01',10,15,0,0,0,2047,1,2,100 },
{ 5,2,'ht00',1,3,'du01',10,15,250,3,5,350 },
{ 6,2,'dm02',5,15,'du01',10,15,250,3,5,350 },
{ 7,2,'ar02',5,15,'du01',10,15,1797,3,5,350 },
};

int nSuperiorItemTable = sizeof SuperiorItemTable / sizeof SuperiorItemTable[0];
```

```
// Tab0.cpp from D2E

#include "JamellaD2E.h"

HBITMAP hBmpJamella;

LRESULT CALLBACK Tab0DialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
    switch(uMsg)
    {
    case WM_INITDIALOG:
        {
#if RELEASE == PUB
            sprintf(buffer,"%s %s",PROGRAMNAME,VERSION);
#else
            sprintf(buffer,"%s %s [%s]",PROGRAMNAME,VERSION,__DATE__);
#endif
            SetDlgItemText(hWnd,IDC_TAB0_Version,buffer);
            SendDlgItemMessage(hWnd,IDC_TAB0_BmpJamella,STM_SETIMAGE,IMAGE_BITMAP,(LPARAM) hBmpJamella);
        }
        return true;
    }
    return false;
}
/*
bool ShowPopupHelp(HWND hWnd,LPARAM lParam)
{
    HELPINFO *HI = (LPHELPINFO) lParam;

    const PopupHelp *PH = 0;

    if ((HWND) HI->hItemHandle == hMainDialog) {

        RECT r;
        GetWindowRect(hTabDialog,&r);

        POINT spt = HI->MousePos;
        spt.x -= r.left;
        spt.y -= r.top;

        HWND hCtrl = ChildWindowFromPointEx(hTabDialog,spt,CWP_SKIPTRANSPARENT);

        if (!hCtrl) return false;

        int CtrlID = GetDlgCtrlID(hCtrl);
        PH = FindPopupHelp(CtrlID);
    }
    else {
        int CtrlID = GetDlgCtrlID((HWND) HI->hItemHandle);
        PH = FindPopupHelp(CtrlID);
    }

    if (!PH) {
        MessageBeep(MB_ICONHAND);
        return false;
    }

    HH_POPUP Popup;

    ZeroMemory(&Popup,sizeof Popup);
    Popup.cbStruct = sizeof Popup;
    Popup.idString = 0;
    Popup.pszText = PH->HelpText;
    Popup.pt = HI->MousePos;
    Popup.clrForeground = RGB(255,0,0);
    Popup.clrBackground = RGB(255,255,239);
    Popup.rcMargins.left = Popup.rcMargins.right = Popup.rcMargins.top = Popup.rcMargins.bottom = -1;
    Popup.pszFont = "MS Sans Serif, 8";

    HtmlHelp((HWND) HI->hItemHandle, 0,
        HH_DISPLAY_TEXT_POPUP, (DWORD) &Popup);

    return true;
}
*/
```

```
// Tab1.cpp from D2E

#include "JamellaD2E.h"

HBITMAP hBmpPlus;

/*
struct
{
    char    *text;
    int     value;
}
StartTowns[4*3] =
{
    { "Rogues' Encampment in Normal", 0x00 },
    { "Lut Gholein in Normal", 0x01 },
    { "Kurast Docks in Normal", 0x02 },
    { "Pandemonium Fortress in Normal", 0x03 },

    { "Rogues' Encampment in Nightmare", 0x10 },
    { "Lut Gholein in Nightmare", 0x11 },
    { "Kurast Docks in Nightmare", 0x12 },
    { "Pandemonium Fortress in Nightmare", 0x13 },

    { "Rogues' Encampment in Hell", 0x20 },
    { "Lut Gholein in Hell", 0x21 },
    { "Kurast Docks in Hell", 0x22 },
    { "Pandemonium Fortress in Hell", 0x23 }
};
*/
struct
{
    char    *text;
    int     value;
}
StartTowns[4] =
{
    { "Rogues' Encampment", 0x00 },
    { "Lut Gholein", 0x01 },
    { "Kurast Docks", 0x02 },
    { "Pandemonium Fortress", 0x03 },
};

const struct
{
    int     Vitality[3];
    int     Energy[3];
    int     Level[3];
}
Boosts[5] =
{
    { // Amazon
        { 3, 0, 1 },
        { 0, 2, 0 },
        { 2, 2, 1 }
    },
    { // Sorceress
        { 2, 0, 1 },
        { 0, 2, 0 },
        { 1, 2, 1 }
    },
    { // Necromancer
        { 2, 0, 1 },
        { 0, 2, 0 },
        { 2, 2, 1 }
    },
    { // Paladin
        { 3, 0, 1 },
        { 0, 2, 0 },
        { 2, 2, 1 }
    },
    { // Barbarian
        { 4, 0, 1 },
        { 0, 1, 0 },
        { 2, 1, 1 }
    }
};

const struct
```

```
{
    int     id;
    DWORD*  val;
}
values[] =
{
    { IDC_TAB1_Level,       &fc.gf.level        },
    { IDC_TAB1_Experience,  &fc.gf.experience   },
    { IDC_TAB1_Char1,       &fc.gf.strength     },
    { IDC_TAB1_Char2,       &fc.gf.dexterity    },
    { IDC_TAB1_Char3,       &fc.gf.vitality     },
    { IDC_TAB1_Char4,       &fc.gf.energy       },
    { IDC_TAB1_Char5,       &fc.gf.statsbonus   },
    { IDC_TAB1_Health,      &fc.gf.health       },
    { IDC_TAB1_HealthMax,   &fc.gf.healthmax    },
    { IDC_TAB1_Mana,        &fc.gf.mana         },
    { IDC_TAB1_ManaMax,     &fc.gf.manamax      },
    { IDC_TAB1_Stamina,     &fc.gf.stamina      },
    { IDC_TAB1_StaminaMax,  &fc.gf.staminamax   },
    { IDC_TAB1_GoldPerson,  &fc.gf.goldperson   },
    { IDC_TAB1_GoldStash,   &fc.gf.goldstash    }
};

extern DWORD experiencelevels[100];
extern int      goldstash[10];
bool editchanging = false;

DWORD level;
DWORD maxgoldperson,maxgoldstash;

void calc_maxgold(HWND hWnd)
{
    if (level > 0 && level <= 99)
    {
        maxgoldperson = level * 10000;
        maxgoldstash = goldstash[level / 10];

        sprintf(buffer,"/ %lu",maxgoldperson);
        SetDlgItemText(hWnd,IDC_TAB1_MaxGoldPerson,buffer);
        sprintf(buffer,"/ %lu",maxgoldstash);
        SetDlgItemText(hWnd,IDC_TAB1_MaxGoldStash,buffer);

        DWORD x = GetDlgItemInt(hWnd,IDC_TAB1_GoldPerson,0,FALSE);
        if (x > maxgoldperson)
            SetDlgItemInt(hWnd,IDC_TAB1_GoldPerson,maxgoldperson,FALSE);

        x = GetDlgItemInt(hWnd,IDC_TAB1_GoldStash,0,FALSE);
        if (x > maxgoldperson)
            SetDlgItemInt(hWnd,IDC_TAB1_GoldStash,maxgoldperson,FALSE);
    }
    else
    {
        SetDlgItemText(hWnd,IDC_TAB1_MaxGoldPerson,"/ ???????");
        SetDlgItemText(hWnd,IDC_TAB1_MaxGoldStash,"/ ???????");
    }
}

LRESULT CALLBACK Tab1DialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
    switch(uMsg)
    {
    case WM_INITDIALOG:
        {
            // Associate Bitmaps
            SendDlgItemMessage(hWnd,IDC_TAB1_Plus1,STM_SETIMAGE,IMAGE_BITMAP,(LPARAM) hBmpPlus);
            SendDlgItemMessage(hWnd,IDC_TAB1_Plus2,STM_SETIMAGE,IMAGE_BITMAP,(LPARAM) hBmpPlus);
            SendDlgItemMessage(hWnd,IDC_TAB1_Plus3,STM_SETIMAGE,IMAGE_BITMAP,(LPARAM) hBmpPlus);
            SendDlgItemMessage(hWnd,IDC_TAB1_Plus4,STM_SETIMAGE,IMAGE_BITMAP,(LPARAM) hBmpPlus);
            SendDlgItemMessage(hWnd,IDC_TAB1_Plus5,STM_SETIMAGE,IMAGE_BITMAP,(LPARAM) hBmpPlus);

            // Add Classes to Class List Box
            HWND hClass = GetDlgItem(hWnd,IDC_TAB1_Class);
            SendMessage(hClass,CB_RESETCONTENT,0,0);
            for (int i=0;i<5;i++)
                SendMessage(hClass,CB_ADDSTRING,0,(LPARAM) CharClasses[i]);

            // Add Difficulties to List Box
            HWND hDiff = GetDlgItem(hWnd,IDC_TAB1_Difficulty);
            SendMessage(hDiff,CB_RESETCONTENT,0,0);
```

```
        for (i=0;i<4;i++)
            SendMessage(hDiff,CB_ADDSTRING,0,(LPARAM) Difficulties[i]);

        // Add Towns to List Box
        HWND hTown = GetDlgItem(hWnd,IDC_TAB1_StartTown);
        SendMessage(hTown,CB_RESETCONTENT,0,0);
        for (i=0;i<4;i++)
            SendMessage(hTown,CB_ADDSTRING,0,(LPARAM) StartTowns[i].text);

        // Load Character values
        SetDlgItemText(hWnd,IDC_TAB1_Name,fc.Header.playername);
        SendMessage(hClass,CB_SETCURSEL,fc.Header.playerclass,0);

        for (i=0;i<4;i++)
        {
            if (StartTowns[i].value == (fc.Header.startingtown & 0x0F))
                SendMessage(hTown,CB_SETCURSEL,i,0);
        }

        for(i=0;i < sizeof values / sizeof values[0];i++)
            SetDlgItemInt(hWnd,values[i].id,*values[i].val,FALSE);

        level = fc.gf.level;
        calc_maxgold(hWnd);

        CheckDlgButton(hWnd,IDC_TAB1_Hardcore,
            fc.Header.hardcore & 0x04 ? BST_CHECKED : BST_UNCHECKED);
        CheckDlgButton(hWnd,IDC_TAB1_Dead,
            fc.Header.hardcore & 0x08 ? BST_CHECKED : BST_UNCHECKED);

        HWND hDead = GetDlgItem(hWnd,IDC_TAB1_Dead);
        EnableWindow(hDead,
            (IsDlgButtonChecked(hWnd,IDC_TAB1_Hardcore) == BST_CHECKED));

        int diff = 0;
        if ((fc.Header.diff & 0x0C) == 0x04) diff = 1;
        if ((fc.Header.diff & 0x0C) == 0x08) diff = 2;
        if ((fc.Header.diff & 0x0C) == 0x0C) diff = 3;
        SendMessage(hDiff,CB_SETCURSEL,diff,0);

        CheckDlgButton(hWnd,IDC_TAB1_StatsLink,BST_CHECKED);

        hToolTip = CreateToolTipCtrl(hWnd,IDD_TAB1,true);
    }
    return true;
case WM_COMMAND:
    {
        switch(LOWORD(wParam))
        {
        case IDC_TAB1_Plus1:
        case IDC_TAB1_Plus2:
        case IDC_TAB1_Plus3:
        case IDC_TAB1_Plus4:
        case IDC_TAB1_Plus5:
            switch(HIWORD(wParam))
            {
            case STN_CLICKED:
            case STN_DBLCLK:
                {
                    int id = LOWORD(wParam) - IDC_TAB1_Plus1;
                    DWORD x = GetDlgItemInt(hWnd,IDC_TAB1_Char1+id,NULL,FALSE);
                    SetDlgItemInt(hWnd,IDC_TAB1_Char1+id,x+1,FALSE);

                    int c = (BYTE)SendDlgItemMessage(hWnd,IDC_TAB1_Class,CB_GETCURSEL,0,0);

                    if (IsDlgButtonChecked(hWnd,IDC_TAB1_StatsLink) == BST_CHECKED)
                    {
                        switch(LOWORD(wParam))
                        {
                        case IDC_TAB1_Plus3:
                            x = GetDlgItemInt(hWnd,IDC_TAB1_HealthMax,NULL,FALSE);
                            x += Boosts[c].Vitality[0];
                            SetDlgItemInt(hWnd,IDC_TAB1_HealthMax,x,FALSE);

                            x = GetDlgItemInt(hWnd,IDC_TAB1_ManaMax,NULL,FALSE);
                            x += Boosts[c].Vitality[1];
                            SetDlgItemInt(hWnd,IDC_TAB1_ManaMax,x,FALSE);

                            x = GetDlgItemInt(hWnd,IDC_TAB1_StaminaMax,NULL,FALSE);
```

```
                    x += Boosts[c].Vitality[2];
                    SetDlgItemInt(hWnd,IDC_TAB1_StaminaMax,x,FALSE);
                    break;
                case IDC_TAB1_Plus4:
                    x = GetDlgItemInt(hWnd,IDC_TAB1_HealthMax,NULL,FALSE);
                    x += Boosts[c].Energy[0];
                    SetDlgItemInt(hWnd,IDC_TAB1_HealthMax,x,FALSE);

                    x = GetDlgItemInt(hWnd,IDC_TAB1_ManaMax,NULL,FALSE);
                    x += Boosts[c].Energy[1];
                    SetDlgItemInt(hWnd,IDC_TAB1_ManaMax,x,FALSE);

                    x = GetDlgItemInt(hWnd,IDC_TAB1_StaminaMax,NULL,FALSE);
                    x += Boosts[c].Energy[2];
                    SetDlgItemInt(hWnd,IDC_TAB1_StaminaMax,x,FALSE);
                    break;
                }
            }
        }
        break;
    }
    break;
case IDC_TAB1_Experience:
    if (editchanging) break;
    switch(HIWORD(wParam))
    {
    case EN_CHANGE:
        {
            char *end;

            editchanging = true;
            int anz = GetDlgItemText(hWnd,IDC_TAB1_Experience,buffer,32);
            DWORD x = strtoul(buffer,&end,10);
            if (end == buffer + anz && x < 4294967294)
            {
                for(int z=0;z<99;z++)
                {
                    if (experiencelevels[z] > x)
                    {
                        break;
                    }
                }
                SetDlgItemInt(hWnd,IDC_TAB1_Level,z,FALSE);
                level = z;
            }
            else
            {
                SetDlgItemText(hWnd,IDC_TAB1_Level,"???");
                level = 0;
            }
            calc_maxgold(hWnd);
            editchanging = false;
        }
    }
    break;
case IDC_TAB1_Level:
    if (editchanging) break;
    switch(HIWORD(wParam))
    {
    case EN_CHANGE:
        {
            char buffer[32];
            char *end;

            editchanging = true;
            int anz = GetDlgItemText(hWnd,IDC_TAB1_Level,buffer,32);
            DWORD x = strtoul(buffer,&end,10);

            if (end == buffer + anz && x > 0 && x <= 99)
            {
                DWORD lvl = experiencelevels[x-1] + 100;
                SetDlgItemInt(hWnd,IDC_TAB1_Experience,lvl,FALSE);
                level = x;
            }
            else
            {
                SetDlgItemText(hWnd,IDC_TAB1_Experience,"???");
                level = 0;
            }
```

```
            editchanging = false;
            calc_maxgold(hWnd);
        }
    }
    break;
case IDC_TAB1_Hardcore:
    switch(HIWORD(wParam))
    {
    case BN_CLICKED:
        {
            EnableWindow(GetDlgItem(hWnd,IDC_TAB1_Dead),
                (IsDlgButtonChecked(hWnd,IDC_TAB1_Hardcore) == BST_CHECKED));
        }
    }
    break;
case IDC_TAB1_Rename:
    if (HIWORD(wParam) == BN_CLICKED)
    {
        DialogBox(hInstance,MAKEINTRESOURCE(IDD_RENAME),hWnd,(DLGPROC) RenameDialogProc);
        SetDlgItemText(hWnd,IDC_TAB1_Name,fc.Header.playername);
    }
    break;
case IDC_TAB1_SetMaxGoldPerson:
    switch(HIWORD(wParam))
    {
    case BN_CLICKED:
        {
            SetDlgItemInt(hWnd,IDC_TAB1_GoldPerson,maxgoldperson,FALSE);
        }
    }
    break;
case IDC_TAB1_SetMaxGoldStash:
    switch(HIWORD(wParam))
    {
    case BN_CLICKED:
        {
            SetDlgItemInt(hWnd,IDC_TAB1_GoldStash,maxgoldstash,FALSE);
        }
    }
    break;
case IDC_TAB1_Batch:
    switch(HIWORD(wParam))
    {
    case BN_CLICKED:
        {
            HMENU hMenu = GetSubMenu(hBatchMenu,0);

            POINT Pos;
            GetCursorPos(&Pos);

            TrackPopupMenu(hMenu, TPM_LEFTALIGN | TPM_LEFTBUTTON,
                Pos.x, Pos.y, 0, hWnd, NULL);
        }
        break;
    }
break;
case IDR_TAB1_RestoreConstitution:
    if (HIWORD(wParam) == BN_CLICKED)
    {
        int X = GetDlgItemInt(hWnd,IDC_TAB1_HealthMax,NULL,FALSE);
        SetDlgItemInt(hWnd,IDC_TAB1_Health,X,FALSE);

        int Y = GetDlgItemInt(hWnd,IDC_TAB1_ManaMax,NULL,FALSE);
        SetDlgItemInt(hWnd,IDC_TAB1_Mana,Y,FALSE);

        int Z = GetDlgItemInt(hWnd,IDC_TAB1_StaminaMax,NULL,FALSE);
        SetDlgItemInt(hWnd,IDC_TAB1_Stamina,Z,FALSE);
    }
    break;
case IDR_TAB1_SetAllStats40:
case IDR_TAB1_SetAllStats60:
case IDR_TAB1_SetAllStats80:
case IDR_TAB1_SetAllStats100:
case IDR_TAB1_SetAllStats120:
case IDR_TAB1_SetAllStats140:
case IDR_TAB1_SetAllStats160:
case IDR_TAB1_SetAllStats180:
case IDR_TAB1_SetAllStats200:
case IDR_TAB1_SetAllStats250:
```

```
case IDR_TAB1_SetAllStats300:
case IDR_TAB1_SetAllStats350:
case IDR_TAB1_SetAllStats400:
case IDR_TAB1_SetAllStats450:
case IDR_TAB1_SetAllStats500:
    {
        struct
        {
            int ID;
            int Value;
        } Stats[] =
        {
            { IDR_TAB1_SetAllStats40,  40 },
            { IDR_TAB1_SetAllStats60,  60 },
            { IDR_TAB1_SetAllStats80,  80 },
            { IDR_TAB1_SetAllStats100, 100 },
            { IDR_TAB1_SetAllStats120, 120 },
            { IDR_TAB1_SetAllStats140, 140 },
            { IDR_TAB1_SetAllStats160, 160 },
            { IDR_TAB1_SetAllStats180, 180 },
            { IDR_TAB1_SetAllStats200, 200 },
            { IDR_TAB1_SetAllStats250, 250 },
            { IDR_TAB1_SetAllStats300, 300 },
            { IDR_TAB1_SetAllStats350, 350 },
            { IDR_TAB1_SetAllStats400, 400 },
            { IDR_TAB1_SetAllStats450, 450 },
            { IDR_TAB1_SetAllStats500, 500 }
        };

        int V = 0;
        for(int z=0;z<sizeof Stats / sizeof Stats[0];z++) {
            if (Stats[z].ID == LOWORD(wParam))
                V = Stats[z].Value;
        }

        SetDlgItemInt(hWnd,IDC_TAB1_Char1,V,FALSE);
        SetDlgItemInt(hWnd,IDC_TAB1_Char2,V,FALSE);
        SetDlgItemInt(hWnd,IDC_TAB1_Char3,V,FALSE);
        SetDlgItemInt(hWnd,IDC_TAB1_Char4,V,FALSE);
    }
    break;
case IDR_TAB1_SetAllConstitution400:
case IDR_TAB1_SetAllConstitution600:
case IDR_TAB1_SetAllConstitution800:
case IDR_TAB1_SetAllConstitution1000:
case IDR_TAB1_SetAllConstitution1500:
case IDR_TAB1_SetAllConstitution2000:
case IDR_TAB1_SetAllConstitution3000:
case IDR_TAB1_SetAllConstitution5000:
    {
        struct
        {
            int ID;
            int Value;
        } Consts[] =
        {
            { IDR_TAB1_SetAllConstitution400,  400 },
            { IDR_TAB1_SetAllConstitution600,  600 },
            { IDR_TAB1_SetAllConstitution800,  800 },
            { IDR_TAB1_SetAllConstitution1000, 1000 },
            { IDR_TAB1_SetAllConstitution1500, 1500 },
            { IDR_TAB1_SetAllConstitution2000, 2000 },
            { IDR_TAB1_SetAllConstitution3000, 3000 },
            { IDR_TAB1_SetAllConstitution5000, 5000 },
        };

        int V = 0;
        for(int z=0;z<sizeof Consts / sizeof Consts[0];z++) {
            if (Consts[z].ID == LOWORD(wParam))
                V = Consts[z].Value;
        }

        SetDlgItemInt(hWnd,IDC_TAB1_Health,V,FALSE);
        SetDlgItemInt(hWnd,IDC_TAB1_HealthMax,V,FALSE);
        SetDlgItemInt(hWnd,IDC_TAB1_Mana,V,FALSE);
        SetDlgItemInt(hWnd,IDC_TAB1_ManaMax,V,FALSE);
        SetDlgItemInt(hWnd,IDC_TAB1_Stamina,V,FALSE);
        SetDlgItemInt(hWnd,IDC_TAB1_StaminaMax,V,FALSE);
    }
```

```
                break;
            }
        }
        break;
    case WM_VALIDATE:
        {
            if (level == 0)
            {
                MessageBox(hWnd,"Level and Experience are invalid!!!\nPlease correct.",PROGRAMNAME,MB_OK);
                return true;
            }
            return false;
        }
    case WM_DESTROY:
        {
            // Retrieve User Changed Data
            GetDlgItemText(hWnd,IDC_TAB1_Name,fc.Header.playername,16);
            fc.Header.playerclass = (BYTE)SendDlgItemMessage(hWnd,IDC_TAB1_Class,CB_GETCURSEL,0,0);

            for(int i=0;i < sizeof values / sizeof values[0];i++)
            {
                *values[i].val = GetDlgItemInt(hWnd,values[i].id,NULL,FALSE);
            }

            fc.Header.level = GetDlgItemInt(hWnd,IDC_TAB1_Level,NULL,FALSE);

            if (IsDlgButtonChecked(hWnd,IDC_TAB1_Hardcore) == BST_CHECKED)
                fc.Header.hardcore |= 0x04;
            else
                fc.Header.hardcore &= 0x04 ^ 0xFF;

            if (IsDlgButtonChecked(hWnd,IDC_TAB1_Dead) == BST_CHECKED)
                fc.Header.hardcore |= 0x08;
            else
                fc.Header.hardcore &= 0x08 ^ 0xFF;

            fc.Header.diff &= 0x0C;

            switch((int)SendDlgItemMessage(hWnd,IDC_TAB1_Difficulty,CB_GETCURSEL,0,0))
            {
            case 0:
                fc.Header.diff |= 0x00;
                break;
            case 1:
                fc.Header.diff |= 0x04;
                break;
            case 2:
                fc.Header.diff |= 0x08;
                break;
            case 3:
                fc.Header.diff |= 0x0C;
                break;
            }

            int startingtown = (BYTE)SendDlgItemMessage(hWnd,IDC_TAB1_StartTown,CB_GETCURSEL,0,0);
            if (startingtown >= 0 && startingtown <= 3)
                fc.Header.startingtown = (fc.Header.startingtown & 0xF0) | (StartTowns[startingtown].value & 0x0F);

            if (hToolTip)
                DestroyWindow(hToolTip);
        }
        return false;
    }
    return false;
}

DWORD experiencelevels[100] =
{
    0,
        500,
        1500,
        3750,
        7875,
        14175,
        22680,
        32886,
        44396,
        57715,
```

```
72144,
90180,
112725,
140906,
176132,
220165,
275207,
344008,
430010,
537513,
671891,
839864,
1049830,
1312287,
1640359,
2050449,
2563061,
3203826,
3902260,
4663553,
5493363,
6397855,
7383752,
8458379,
9629723,
10906488,
12298162,
13815086,
15468534,
17270791,
19235252,
21376515,
23710491,
26254525,
29027522,
32050088,
35344686,
38935798,
42850109,
47116709,
51767302,
56836449,
62361819,
68384473,
74949165,
82104680,
89904191,
98405658,
107672256,
117772849,
128782495,
140783010,
153863570,
168121381,
183662396,
200602101,
219066380,
239192444,
261129853,
285041630,
311105466,
339515048,
370481492,
404234916,
441026148,
481128591,
524840254,
572485967,
624419793,
681027665,
742730244,
809986056,
883294891,
963201521,
1050299747,
1145236814,
1248718217,
1361512946,
1484459201,
```

```
        1618470619,
        1764543065,
        1923762030,
        2097310703,
        2286478756,
        2492671933,
        2717422497,
        2962400612,
        3229426756,
        3520485254,
        4000000000
};

int goldstash[10] =
{ 50000,100000,150000,200000,250000,300000,350000,400000,450000,500000 };
```

```
// Tab2.cpp from D2E

#include "JamellaD2E.h"

// Tab2 Controls
HWND hQuality;
// ItemTree
HWND        hTV;
HIMAGELIST hTVImagelist = 0;

// Inventory Parts
HWND hInv,hStash,hBelt,hCube,hBody,hCopyBuffer,hLoad,hSave;
RECT rCube,rBelt,rStash;

inline ContainerFromHWND(HWND hWnd)
{
    if (hWnd == hInv) return CNT_INVENTORY;
    if (hWnd == hStash) return CNT_STASH;
    if (hWnd == hBelt) return CNT_BELT;
    if (hWnd == hCube) return CNT_CUBE;
    if (hWnd == hBody) return CNT_BODY;
    if (hWnd == hCopyBuffer) return CNT_COPYBUFFER;
    return CNT_NONE;
}

HBITMAP hItemUnknown;
HBITMAP hBmpBodyWhole,hBmpNotPlaceable;
HCURSOR hCurMove,hCurMoveCopy,hCurCross,hCurNo;
HBRUSH hBrushBlack,hBrushNull;
HPEN hPenWhite,hPenGreen,hPenBusy,hPenUnwearable;

class Item*         SelItem;
class Item*         CopyBuffer = 0;

static UINT         ClipBoardFormat,ClipBoardFormatJohnDoe;
static HGLOBAL      ClipBoardData,ClipBoardDataJohnDoe;

bool MouseButtonDown = false;
bool MouseButtonMoved = false;
int  xStartPos,yStartPos;
int  xDragTol,yDragTol;
bool TVMouseDrag = false;

void MakeSelection(Item *i);
bool SelItemDeleteable();

// Item Context Menu Entries
#define IDR_TAB2_Delete                     50000
#define IDR_TAB2_Socketed                   50001
#define IDR_TAB2_Identified                 50002
#define IDR_TAB2_Starter                    50003
#define IDR_TAB2_SetQuantity                50004
#define IDR_TAB2_SetQuality                 50005
#define IDR_TAB2_SetDurability              50006
#define IDR_TAB2_SetDefense                 50007
#define IDR_TAB2_SetQualityCrude            50008
#define IDR_TAB2_SetQualityRegular          50009
#define IDR_TAB2_SetQualitySuperior         50010
#define IDR_TAB2_SetQualityMagical          50011
#define IDR_TAB2_SetQualityRare             50012
#define IDR_TAB2_SetQualitySet              50013
#define IDR_TAB2_SetQualityUnique           50014
#define IDR_TAB2_AttributeSelect            50015
#define IDR_TAB2_GemsEdit                   50016
#define IDR_TAB2_SetEarProperties           50017
#define IDR_TAB2_ChangeRingImage            50018
#define IDR_TAB2_ChangeAmuletImage          50019
#define IDR_TAB2_Cut                        50020
#define IDR_TAB2_Copy                       50021
#define IDR_TAB2_Paste                      50022

#pragma pack(1)

struct JamClipboardHead
{
    WORD    IID;
    BYTE    xSize;
    BYTE    ySize;
    BYTE    BodyPlace;
```

```
};

#pragma pack(4)

void LoadItemTree()
{
    TV_ITEM TVItem;
    TV_INSERTSTRUCT TVInsert;
    HTREEITEM hPrevBranches[5];
    hPrevBranches[0] = TVI_ROOT;

    for(int z1=0;z1<nItemTree;z1++)
    {
        ZeroMemory(&TVItem,sizeof TVItem);
        TVItem.mask = TVIF_TEXT | TVIF_PARAM;
        TVItem.pszText = ItemTree[z1].Text;
        TVItem.lParam = (LPARAM)-1;

        TVInsert.hParent = hPrevBranches[ItemTree[z1].Depth-1];
        TVInsert.hInsertAfter = TVI_LAST;
        TVInsert.item = TVItem;

        hPrevBranches[ItemTree[z1].Depth] =
        ItemTree[z1].hTree =
            TreeView_InsertItem(hTV,&TVInsert);

        for(int z2=0;z2<nItemInfos;z2++)
        {
            if (ItemInfos[z2].TreeID != ItemTree[z1].TreeID) continue;

            ZeroMemory(&TVItem,sizeof TVItem);
            TVItem.mask = TVIF_TEXT | TVIF_PARAM;
            TVItem.pszText = ItemInfos[z2].ItemName;
            TVItem.lParam = z2;

            TVInsert.hParent = hPrevBranches[ItemTree[z1].Depth];
            TVInsert.hInsertAfter = TVI_LAST;
            TVInsert.item = TVItem;

            ItemInfos[z2].hTree = TreeView_InsertItem(hTV,&TVInsert);
        }
    }
}

void MakeEmptyBelt(int Column=-1)
{
    // Delete Items in Belt
    for(Item *I = Items;I != 0;)
    {
        if (I->Container() == CNT_BELT)
        {
            Item *J = I->Next();
            if (Column < 0 || I->xPos() == Column) {
                I->Delete();
            }
            I = J;
        }
        else
        {
            I = I->Next();
        }
    }
}

LRESULT CALLBACK QuantityDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
    switch(uMsg)
    {
    case WM_INITDIALOG:
        {
            sprintf(buffer,"/  %i  Max",SelItem->Info->Quantity);
            SetDlgItemText(hWnd,IDC_TAB2Quantity_Max,buffer);
            SetDlgItemInt(hWnd,IDC_TAB2Quantity_Set,SelItem->Quantity(),TRUE);
        }
        return true;
    case WM_COMMAND:
        switch (LOWORD(wParam))
        {
        case IDOK: {
```

```
                BOOL nOK;
                int x = GetDlgItemInt(hWnd,IDC_TAB2Quantity_Set,&nOK,TRUE);
                if (nOK && x > 0 && (x <= 255 || RegOptions.ExceedQuantity))
                {
                    SelItem->SetQuantity(x);
                    EndDialog(hWnd,0);
                }
                else
                    MessageBeep(MB_ICONHAND);
                return true;
            }
        case IDCANCEL:
            EndDialog(hWnd,0);
            return true;
        }
        break;
    }
    return false;
}
LRESULT CALLBACK DurabilityDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
    switch(uMsg)
    {
    case WM_INITDIALOG:
        {
            SetDlgItemInt(hWnd,IDC_TAB2Durability_Set,SelItem->Durability(),TRUE);
            SetDlgItemInt(hWnd,IDC_TAB2Durability_Max,SelItem->DurabilityMax(),TRUE);
        }
        return true;
    case WM_COMMAND:
        switch (LOWORD(wParam))
        {
        case IDOK: {
                BOOL nOK1,nOK2;
                int Is = GetDlgItemInt(hWnd,IDC_TAB2Durability_Set,&nOK1,TRUE);
                int Max = GetDlgItemInt(hWnd,IDC_TAB2Durability_Max,&nOK2,TRUE);
                if (nOK1 && nOK2 && Is > 0 && Max > 0 && Is <= 255 && Max <= 255 && Is <= Max)
                {
                    SelItem->SetDurabilityMax(Max);
                    SelItem->SetDurability(Is);
                    EndDialog(hWnd,0);
                }
                else
                    MessageBeep(MB_ICONHAND);
                return true;
            }
        case IDCANCEL:
            EndDialog(hWnd,0);
            return true;
        }
        break;
    }
    return false;
}

static DWORD WINAPI DefenseAttack(LPVOID ThreadParameters)
{
    DefenseSearchThread *P = (DefenseSearchThread *)ThreadParameters;
    P->Advanced = false;

    P->Dialog = CreateDialogParam(hInstance,MAKEINTRESOURCE(IDD_TAB2SS),hMainDialog,(DLGPROC) Tab2SearchDialogProc, (LPA
RAM) P);
    if (!P->Dialog) return -1;

    Item* I = P->Item;
    I->Busy = true;

    while(!CheckPollMessages())
    {
        I->SetDWA(I->DWA() + 1);
        P->Counter++;

        if (I->BaseDefense() == P->SelectAC)
            break;
    }

    MessageBeep(MB_ICONASTERISK);
    I->Busy = false;
    EndDialog(P->Dialog,0);
```

```
    UpdateTab2();

    CloseHandle(P->Thread);
    GlobalFree(P->ThreadData);
    return 0;
}

LRESULT CALLBACK DefenseDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
    switch(uMsg)
    {
    case WM_INITDIALOG:
        {
            sprintf(buffer,"Range %i to %i",SelItem->Info->ACMin,SelItem->Info->ACMax-1);
            SetDlgItemText(hWnd,IDC_TAB2Defense_Range,buffer);
            SetDlgItemInt(hWnd,IDC_TAB2Defense_Value,SelItem->BaseDefense(),TRUE);
        }
        return true;
    case WM_COMMAND:
        switch (LOWORD(wParam))
        {
        case IDOK: {
                BOOL nOK;
                int Val = GetDlgItemInt(hWnd,IDC_TAB2Defense_Value,&nOK,TRUE);

                if (nOK && SelItem->Info->ACMin <= Val && Val <= SelItem->Info->ACMax-1)
                {
                    SelItem->DWBHistory.StepAdd(SelItem);

                    HGLOBAL TD = GlobalAlloc(GMEM_ZEROINIT,sizeof DefenseSearchThread);
                    struct DefenseSearchThread* NewThread = (DefenseSearchThread*)GlobalLock(TD);;
                    NewThread->ThreadData = TD;

                    NewThread->Item = SelItem;
                    NewThread->SelectAC = Val;

                    DWORD ThreadID;
                    NewThread->Thread = CreateThread(NULL,0,&DefenseAttack,NewThread,0,&ThreadID);

                    EndDialog(hWnd,0);
                }
                else
                    MessageBeep(MB_ICONHAND);
                return true;
            }
        case IDCANCEL:
            EndDialog(hWnd,0);
            return true;
        }
        break;
    }
    return false;
}
LRESULT CALLBACK EarDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
    switch(uMsg)
    {
    case WM_INITDIALOG:
        {
            // Set Name
            SetDlgItemText(hWnd,IDC_TAB2Ear_Name,SelItem->OpponentName());

            // Fill and set character class
            SendDlgItemMessage(hWnd,IDC_TAB2Ear_Class,CB_RESETCONTENT,0,0);
            for (int i=0;i<5;i++)
                SendDlgItemMessage(hWnd,IDC_TAB2Ear_Class,CB_ADDSTRING,0,(LPARAM) CharClasses[i]);
            SendDlgItemMessage(hWnd,IDC_TAB2Ear_Class,CB_SETCURSEL,SelItem->OpponentClass(),0);

            // Set Level
            SetDlgItemInt(hWnd,IDC_TAB2Ear_Level,SelItem->OpponentLevel(),FALSE);
        }
        return true;
    case WM_COMMAND:
        switch (LOWORD(wParam))
        {
        case IDOK: {
                GetDlgItemText(hWnd,IDC_TAB2Ear_Name,buffer,32);
                SelItem->SetOpponentName(buffer);
                SelItem->SetOpponentClass( SendDlgItemMessage(hWnd,IDC_TAB2Ear_Class,CB_GETCURSEL,0,0) );
```

```
                SelItem->SetOpponentLevel( GetDlgItemInt(hWnd,IDC_TAB2Ear_Level,NULL,FALSE) );

                EndDialog(hWnd,0);
                return true;
            }
        case IDCANCEL:
            EndDialog(hWnd,0);
            return true;
        }
        break;
    }
    return false;
}

static DWORD WINAPI RingAmuletAttack(LPVOID ThreadParameters)
{
    RingAmuletSearchThread *P = (RingAmuletSearchThread *)ThreadParameters;
    P->Advanced = false;

    P->Dialog = CreateDialogParam(hInstance,MAKEINTRESOURCE(IDD_TAB2SS),hMainDialog,(DLGPROC) Tab2SearchDialogProc,(LPAR
AM) P);
    if (!P->Dialog) return -1;

    Item* I = P->Item;
    I->Busy = true;

    while(!CheckPollMessages())
    {
        I->SetDWA(I->DWA() + 1);
        P->Counter++;

        if (P->RingAmulet == false) {
            if ( (I->DWARandomOffset(1) % 5) == P->Image)
                break;
        }
        else {
            if ( (I->DWARandomOffset(1) % 3) == P->Image)
                break;
        }
    }

    MessageBeep(MB_ICONASTERISK);
    I->Busy = false;
    I->Info = 0;

    EndDialog(P->Dialog,IDOK);
    InvalidateRect(hTabDialog,NULL,FALSE);

    UpdateTab2();

    CloseHandle(P->Thread);
    GlobalFree(P->ThreadData);
    return 0;
}

LRESULT CALLBACK RingImageDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
    switch(uMsg)
    {
    case WM_INITDIALOG:
        {
            // Fill in Images
            for (int z=0;z<5;z++)
                SendDlgItemMessage(hWnd,IDC_TAB2RingImage_Image1+z,BM_SETIMAGE,IMAGE_BITMAP,(LPARAM) RingImages[z].hBmp)
;

            int x = SelItem->DWARandomOffset(1) % 5;

            CheckRadioButton(hWnd,IDC_TAB2RingImage_Image1,IDC_TAB2RingImage_Image5,IDC_TAB2RingImage_Image1+x);
        }
        return true;
    case WM_COMMAND:
        switch (LOWORD(wParam))
        {
        case IDOK: {
                int s = -1;
                for(int z=0;z<5;z++) {
                    if (IsDlgButtonChecked(hWnd,IDC_TAB2RingImage_Image1+z) == BST_CHECKED)
```

```
                    s = z;
                }

                if (s < 0)
                    EndDialog(hWnd,IDOK);

                SelItem->DWBHistory.StepAdd(SelItem);

                HGLOBAL TD = GlobalAlloc(GMEM_ZEROINIT,sizeof RingAmuletSearchThread);
                struct RingAmuletSearchThread* NewThread = (RingAmuletSearchThread*)GlobalLock(TD);;
                NewThread->ThreadData = TD;

                NewThread->Item = SelItem;
                NewThread->RingAmulet = false;
                NewThread->Image = s;

                DWORD ThreadID;
                NewThread->Thread = CreateThread(NULL,0,&RingAmuletAttack,NewThread,0,&ThreadID);

                EndDialog(hWnd,IDOK);
            }
        case IDCANCEL:
            EndDialog(hWnd,0);
            return true;
        }
        break;
    }
    return false;
}
LRESULT CALLBACK AmuletImageDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
    switch(uMsg)
    {
    case WM_INITDIALOG:
        {
            // Fill in Images
            for (int z=0;z<3;z++)
                SendDlgItemMessage(hWnd,IDC_TAB2AmuletImage_Image1+z,BM_SETIMAGE,IMAGE_BITMAP,(LPARAM) AmuletImages[z].h
Bmp);

            int x = SelItem->DWARandomOffset(1) % 3;

            CheckRadioButton(hWnd,IDC_TAB2AmuletImage_Image1,IDC_TAB2AmuletImage_Image3,
                IDC_TAB2AmuletImage_Image1+x);
        }
        return true;
    case WM_COMMAND:
        switch (LOWORD(wParam))
        {
        case IDOK: {
                int s = -1;
                for(int z=0;z<3;z++) {
                    if (IsDlgButtonChecked(hWnd,IDC_TAB2AmuletImage_Image1+z) == BST_CHECKED)
                        s = z;
                }

                if (s < 0)
                    EndDialog(hWnd,IDOK);

                SelItem->DWBHistory.StepAdd(SelItem);

                HGLOBAL TD = GlobalAlloc(GMEM_ZEROINIT,sizeof RingAmuletSearchThread);
                struct RingAmuletSearchThread* NewThread = (RingAmuletSearchThread*)GlobalLock(TD);;
                NewThread->ThreadData = TD;

                NewThread->Item = SelItem;
                NewThread->RingAmulet = true;
                NewThread->Image = s;

                DWORD ThreadID;
                NewThread->Thread = CreateThread(NULL,0,&RingAmuletAttack,NewThread,0,&ThreadID);

                EndDialog(hWnd,IDOK);
            }
        case IDCANCEL:
            EndDialog(hWnd,0);
            return true;
        }
        break;
```

```
    }
    return false;
}

bool SaveToClipboard(Item *I)
{
    if (!OpenClipboard(hMainDialog))
        return false;

    if (!EmptyClipboard())
        return false;

    Item *G = 0;

    // Jamella Clipboard Format
    {
        if (ClipBoardData)
            GlobalFree(ClipBoardData);

        int ClipSize = I->ItemRecordLength();

        for(Item *G = I->Gems;G != 0;G = G->Next())
            ClipSize += G->ItemRecordLength();

        ClipSize += sizeof (JamClipboardHead);

        ClipBoardData = GlobalAlloc(GMEM_MOVEABLE | GMEM_ZEROINIT,ClipSize);
        BYTE *Clip = (BYTE*)GlobalLock(ClipBoardData);

        struct JamClipboardHead *JamHead = (JamClipboardHead*) Clip;
        JamHead->IID = I->ItemRecordID();
        JamHead->xSize = I->xSize();
        JamHead->ySize = I->ySize();
        JamHead->BodyPlace = I->BodyPlace();

        int o = sizeof (JamClipboardHead);

        memcpy(Clip+o,I->GetItemRecord(),I->ItemRecordLength());
        o += I->ItemRecordLength();

        for(G = I->Gems;G != 0;G = G->Next())
        {
            memcpy(Clip+o,G->GetItemRecord(),G->ItemRecordLength());
            o += G->ItemRecordLength();
        }

        GlobalUnlock(ClipBoardData);

        SetClipboardData(ClipBoardFormat,ClipBoardData);
    }

    if (I->ItemRecordID() == IT_103)
    {
        // John Doe Clipboard Format
        if (ClipBoardDataJohnDoe)
            GlobalFree(ClipBoardDataJohnDoe);

        int ClipSize = I->ItemRecordLength();

        for(G = I->Gems;G != 0;G = G->Next())
            ClipSize += G->ItemRecordLength();

        ClipSize += 4;

        ClipBoardDataJohnDoe = GlobalAlloc(GMEM_MOVEABLE | GMEM_ZEROINIT,ClipSize);
        BYTE *Clip = (BYTE*)GlobalLock(ClipBoardDataJohnDoe);

        *(DWORD*)Clip = I->Gems->Count();

        int o = 4;
        memcpy(Clip+o,I->GetItemRecord(),I->ItemRecordLength());
        o += I->ItemRecordLength();

        for(G = I->Gems;G != 0;G = G->Next())
        {
            memcpy(Clip+o,G->GetItemRecord(),G->ItemRecordLength());
            o += G->ItemRecordLength();
        }
```

```
        GlobalUnlock(ClipBoardDataJohnDoe);

        SetClipboardData(ClipBoardFormatJohnDoe,ClipBoardDataJohnDoe);
    }

    CloseClipboard();
    return true;
}


static struct
{
    int     code;
    char*   text;
    int     index;
}
Qualities[] =
{
    { CRUDEITEM,    "Crude" },
    { USUALITEM,    "Regular" },
    { SUPERIORITEM, "Superior" },
    { MAGICITEM,    "Magical" },
    { SETITEM,      "Set Item" },
    { RAREITEM,     "Rare Item" },
    { UNIQUEITEM,   "Unique Item" }
};

bool FindCube();

LRESULT CALLBACK ProcessItemMessage(HWND hWnd,WPARAM wParam,LPARAM lParam)
{
    switch(LOWORD(wParam))
    {
    // Item Context Menu
    case IDR_TAB2_Delete:
        {
            if (MouseButtonDown) break;
            if (!SelItem) break;
            if (SelItem->Busy) break;

            if (SelItemDeleteable())
            {
                if (SelItem == CopyBuffer)
                {
                    delete CopyBuffer;
                    CopyBuffer = 0;
                }
                else
                {
                    SelItem->Delete();
                }
                MakeSelection(0);
            }
        }
        break;
    case IDR_TAB2_Copy:
        {
            if (!SelItem || SelItem->Busy) break;
            SaveToClipboard(SelItem);
        }
        break;
    case IDR_TAB2_Cut:
        {
            if (!SelItem || SelItem->Busy) break;
            SaveToClipboard(SelItem);

            if (SelItemDeleteable()) {
                if (SelItem == CopyBuffer) {
                    delete CopyBuffer;
                    CopyBuffer = 0;
                }
                else {
                    SelItem->Delete();
                }
                MakeSelection(0);
            }
        }
        break;
    case IDR_TAB2_Socketed:
```

```
    if (SelItem->Socketed() || SelItem->Socketable() || RegOptions.AllItemsSocketable)
    {
        if (SelItem->Socketed()) {
            SelItem->SetSocketed(false);
            SelItem->SetGemNum(0);
            if (SelItem->Gems) {
                delete SelItem->Gems;
                SelItem->Gems = 0;
            }
        }
        else
            SelItem->SetSocketed(true);
        UpdateTab2();
    }
    break;
case IDR_TAB2_Identified:
    {
        if (SelItem->Identified())
            SelItem->SetIdentified(false);
        else
            SelItem->SetIdentified(true);
        UpdateTab2();
    }
    break;
case IDR_TAB2_Starter:
    {
        if (SelItem->Starter())
            SelItem->SetStarter(false);
        else
            SelItem->SetStarter(true);
        UpdateTab2();
    }
    break;
case IDR_TAB2_SetQuantity:
    {
        DialogBox(hInstance,MAKEINTRESOURCE(IDD_TAB2Quantity),hWnd,(DLGPROC) QuantityDialogProc);
        UpdateTab2();
    }
    break;
case IDR_TAB2_SetDurability:
    {
        DialogBox(hInstance,MAKEINTRESOURCE(IDD_TAB2Durability),hWnd,(DLGPROC) DurabilityDialogProc);
        UpdateTab2();
    }
    break;
case IDR_TAB2_SetDefense:
    {
        DialogBox(hInstance,MAKEINTRESOURCE(IDD_TAB2Defense),hWnd,(DLGPROC) DefenseDialogProc);
        UpdateTab2();
    }
    break;

case IDR_TAB2_SetQualityCrude:
    SelItem->SetQuality(CRUDEITEM);
    UpdateTab2();
    InvalidateRect(hTabDialog,NULL,FALSE);
    break;
case IDR_TAB2_SetQualityRegular:
    SelItem->SetQuality(USUALITEM);
    UpdateTab2();
    InvalidateRect(hTabDialog,NULL,FALSE);
    break;
case IDR_TAB2_SetQualitySuperior:
    SelItem->SetQuality(SUPERIORITEM);
    UpdateTab2();
    InvalidateRect(hTabDialog,NULL,FALSE);
    break;
case IDR_TAB2_SetQualityMagical:
    SelItem->SetQuality(MAGICITEM);
    UpdateTab2();
    InvalidateRect(hTabDialog,NULL,FALSE);
    break;
case IDR_TAB2_SetQualityRare:
    SelItem->SetQuality(RAREITEM);
    UpdateTab2();
    InvalidateRect(hTabDialog,NULL,FALSE);
    break;
case IDR_TAB2_SetQualitySet:
    SelItem->SetQuality(SETITEM);
```

```
        UpdateTab2();
        InvalidateRect(hTabDialog,NULL,FALSE);
        break;
    case IDR_TAB2_SetQualityUnique:
        SelItem->SetQuality(UNIQUEITEM);
        UpdateTab2();
        InvalidateRect(hTabDialog,NULL,FALSE);
        break;

    case IDR_TAB2_AttributeSelect:
        if (SelItem->Quality() == MAGICITEM)
        {
            DialogBox(hInstance,MAKEINTRESOURCE(IDD_TAB2Magic),hTabDialog,(DLGPROC) Tab2MagicDialogProc);
            SelItem->Decoded = 0;
            UpdateTab2();
        }
        else if (SelItem->Quality() == RAREITEM)
        {
            DialogBox(hInstance,MAKEINTRESOURCE(IDD_TAB2Rare),hTabDialog,(DLGPROC) Tab2RareDialogProc);
            SelItem->Decoded = 0;
            UpdateTab2();
        }
        break;
    case IDR_TAB2_GemsEdit:
        {
            if (!SelItem->Socketed()) break;

            if (SelItem->Quality() == MAGICITEM || SelItem->Quality() == RAREITEM && RegOptions.NoAnnoyingMsgs)
            {
                MessageBox(hWnd,"WARNING:\nSocketed magical or rare items cannot be created in the game, they are truly
hacked!\nBlizzard may bring out a blocking patch one day!",PROGRAMNAME,MB_OK | MB_ICONHAND);
            }

            if (SelItem->Info->Sockets == 0 || RegOptions.A7Gems) {
                DialogBox(hInstance,MAKEINTRESOURCE(IDD_TAB2Gems7),hTabDialog,(DLGPROC) Tab2GemsDialogProc);
            }
            else if (SelItem->Info->Sockets == 1) {
                DialogBox(hInstance,MAKEINTRESOURCE(IDD_TAB2Gems1),hTabDialog,(DLGPROC) Tab2GemsDialogProc);
            }
            else if (SelItem->Info->Sockets == 2) {
                DialogBox(hInstance,MAKEINTRESOURCE(IDD_TAB2Gems2),hTabDialog,(DLGPROC) Tab2GemsDialogProc);
            }
            else if (SelItem->Info->Sockets == 3) {
                DialogBox(hInstance,MAKEINTRESOURCE(IDD_TAB2Gems3),hTabDialog,(DLGPROC) Tab2GemsDialogProc);
            }

            SelItem->Decoded = 0;
            UpdateTab2();
        }
        break;
    case IDR_TAB2_SetEarProperties:
        {
            DialogBox(hInstance,MAKEINTRESOURCE(IDD_TAB2Ear),hTabDialog,(DLGPROC) EarDialogProc);
            SelItem->Decoded = 0;
            UpdateTab2();
        }
        break;
    case IDR_TAB2_ChangeRingImage:
        {
            DialogBox(hInstance,MAKEINTRESOURCE(IDD_TAB2RingImage),hTabDialog,(DLGPROC) RingImageDialogProc);
            SelItem->Info = 0;
            UpdateTab2();
            InvalidateRect(hTabDialog,NULL,FALSE);
        }
        break;
    case IDR_TAB2_ChangeAmuletImage:
        {
            DialogBox(hInstance,MAKEINTRESOURCE(IDD_TAB2AmuletImage),hTabDialog,(DLGPROC) AmuletImageDialogProc);
            SelItem->Info = 0;
            UpdateTab2();
            InvalidateRect(hTabDialog,NULL,FALSE);
        }
        break;
    }
    return true;
}

LRESULT CALLBACK Tab2DialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
```

```
    switch(uMsg)
    {
    case WM_INITDIALOG:
        {
            // Set up ItemTree
            hTV = GetDlgItem(hWnd,IDC_TAB2_ItemTree);
            LoadItemTree();

            hBody   = GetDlgItem(hWnd,IDC_TAB2_Body);
            hInv    = GetDlgItem(hWnd,IDC_TAB2_Inv);
#if INVGRIDS == 0
            hStash  = GetDlgItem(hWnd,IDC_TAB2_Stash);
            hBelt   = GetDlgItem(hWnd,IDC_TAB2_Belt);
            hCube   = GetDlgItem(hWnd,IDC_TAB2_Cube);
#endif
            hCopyBuffer = GetDlgItem(hWnd,IDC_TAB2_CopyBuffer);
            hLoad   = GetDlgItem(hWnd,IDC_TAB2_Load);
            hSave   = GetDlgItem(hWnd,IDC_TAB2_Save);

            SetWindowLong(hBody,    GWL_WNDPROC,(LONG) Tab2InventoryProc);
            SetWindowLong(hInv,     GWL_WNDPROC,(LONG) Tab2InventoryProc);
#if INVGRIDS == 0
            SetWindowLong(hStash,   GWL_WNDPROC,(LONG) Tab2InventoryProc);
            SetWindowLong(hBelt,    GWL_WNDPROC,(LONG) Tab2InventoryProc);
            SetWindowLong(hCube,    GWL_WNDPROC,(LONG) Tab2InventoryProc);
#endif
            SetWindowLong(hCopyBuffer,GWL_WNDPROC,(LONG) Tab2InventoryProc);

            EnableWindow(GetDlgItem(hWnd,IDC_TAB2_AttrRandom),FALSE);

            SelItem = 0;

            xDragTol = GetSystemMetrics(SM_CXDRAG);
            yDragTol = GetSystemMetrics(SM_CYDRAG);

            hExpertBox = 0;
            CheckDlgButton(hWnd,IDC_TAB2_ExpertMode,BST_UNCHECKED);

            SendDlgItemMessage(hWnd,IDC_TAB2_RichText,EM_SETBKGNDCOLOR,FALSE,GetSysColor(COLOR_BTNFACE));

            hToolTip = CreateToolTipCtrl(hWnd,IDD_TAB2,true);

            ClipBoardFormat = RegisterClipboardFormat(CLIPBOARDFORMAT);
            ClipBoardFormatJohnDoe = RegisterClipboardFormat("Diablo2Item");
        }
        return true;
    case WM_NOTIFY:
        switch(((LPNMHDR)lParam)->idFrom)
        {
        case IDC_TAB2_ItemTree:
            {
                NM_TREEVIEW* TreeView = (NM_TREEVIEW FAR *) lParam;
                switch(TreeView->hdr.code)
                {
                case TVN_SELCHANGING:
                    if (SelItem)
                    {
                        MakeSelection(0);
                    }
                    return false;
                case TVN_SELCHANGED:
                    if (TreeView->itemNew.lParam >= 0 && TreeView->itemNew.lParam < nItemInfos)
                    {
                        SelInfo = &ItemInfos[TreeView->itemNew.lParam];
                    }
                    return false;
                case TVN_BEGINDRAG:
                    if (TreeView->itemNew.lParam >= 0 && TreeView->itemNew.lParam < nItemInfos)
                    {
                        MakeSelection(0);
                        SelInfo = &ItemInfos[TreeView->itemNew.lParam];
                        SetCapture(hInv);
                        MouseButtonDown = true;
                        xStartPos = TreeView->ptDrag.x;
                        yStartPos = TreeView->ptDrag.y;
                        FindCube();
                    }
                    return false;
                }
```

```
        }
        break;
    }
    return true;
case WM_COMMAND:
    {
        switch(LOWORD(wParam))
        {
        default:
            ProcessItemMessage(hWnd,wParam,lParam);
            break;
        // Batch Action Context Menu
        case IDR_TAB2_BeltEmpty:
            {
                MakeEmptyBelt();
                UpdateTab2();
                InvalidateRect(hTabDialog,NULL,FALSE);
            }
            break;
        case IDR_TAB2_BeltMinorHealing:
        case IDR_TAB2_BeltLightHealing:
        case IDR_TAB2_BeltHealing:
        case IDR_TAB2_BeltGreaterHealing:
        case IDR_TAB2_BeltSuperHealing:
        case IDR_TAB2_BeltMinorMana:
        case IDR_TAB2_BeltMana:
        case IDR_TAB2_BeltLightMana:
        case IDR_TAB2_BeltGreaterMana:
        case IDR_TAB2_BeltSuperMana:
        case IDR_TAB2_BeltRejuv:
        case IDR_TAB2_BeltFullRejuv:

        case IDR_TAB2_Slot1MinorHealing:
        case IDR_TAB2_Slot1LightHealing:
        case IDR_TAB2_Slot1Healing:
        case IDR_TAB2_Slot1GreaterHealing:
        case IDR_TAB2_Slot1SuperHealing:
        case IDR_TAB2_Slot1MinorMana:
        case IDR_TAB2_Slot1Mana:
        case IDR_TAB2_Slot1LightMana:
        case IDR_TAB2_Slot1GreaterMana:
        case IDR_TAB2_Slot1SuperMana:
        case IDR_TAB2_Slot1Rejuv:
        case IDR_TAB2_Slot1FullRejuv:

        case IDR_TAB2_Slot2MinorHealing:
        case IDR_TAB2_Slot2LightHealing:
        case IDR_TAB2_Slot2Healing:
        case IDR_TAB2_Slot2GreaterHealing:
        case IDR_TAB2_Slot2SuperHealing:
        case IDR_TAB2_Slot2MinorMana:
        case IDR_TAB2_Slot2Mana:
        case IDR_TAB2_Slot2LightMana:
        case IDR_TAB2_Slot2GreaterMana:
        case IDR_TAB2_Slot2SuperMana:
        case IDR_TAB2_Slot2Rejuv:
        case IDR_TAB2_Slot2FullRejuv:

        case IDR_TAB2_Slot3MinorHealing:
        case IDR_TAB2_Slot3LightHealing:
        case IDR_TAB2_Slot3Healing:
        case IDR_TAB2_Slot3GreaterHealing:
        case IDR_TAB2_Slot3SuperHealing:
        case IDR_TAB2_Slot3MinorMana:
        case IDR_TAB2_Slot3Mana:
        case IDR_TAB2_Slot3LightMana:
        case IDR_TAB2_Slot3GreaterMana:
        case IDR_TAB2_Slot3SuperMana:
        case IDR_TAB2_Slot3Rejuv:
        case IDR_TAB2_Slot3FullRejuv:

        case IDR_TAB2_Slot4MinorHealing:
        case IDR_TAB2_Slot4LightHealing:
        case IDR_TAB2_Slot4Healing:
        case IDR_TAB2_Slot4GreaterHealing:
        case IDR_TAB2_Slot4SuperHealing:
        case IDR_TAB2_Slot4MinorMana:
        case IDR_TAB2_Slot4Mana:
        case IDR_TAB2_Slot4LightMana:
```

```
case IDR_TAB2_Slot4GreaterMana:
case IDR_TAB2_Slot4SuperMana:
case IDR_TAB2_Slot4Rejuv:
case IDR_TAB2_Slot4FullRejuv:
    {
        int IC;
        int Col;

        switch(LOWORD(wParam))
        {
        case IDR_TAB2_BeltMinorHealing:     IC = 0x15A0; Col = -1; break;
        case IDR_TAB2_BeltLightHealing:     IC = 0x15B0; Col = -1; break;
        case IDR_TAB2_BeltHealing:          IC = 0x15C0; Col = -1; break;
        case IDR_TAB2_BeltGreaterHealing:   IC = 0x15D0; Col = -1; break;
        case IDR_TAB2_BeltSuperHealing:     IC = 0x15E0; Col = -1; break;
        case IDR_TAB2_BeltMinorMana:        IC = 0x15F0; Col = -1; break;
        case IDR_TAB2_BeltLightMana:        IC = 0x1600; Col = -1; break;
        case IDR_TAB2_BeltMana:             IC = 0x1610; Col = -1; break;
        case IDR_TAB2_BeltGreaterMana:      IC = 0x1620; Col = -1; break;
        case IDR_TAB2_BeltSuperMana:        IC = 0x1630; Col = -1; break;
        case IDR_TAB2_BeltRejuv:            IC = 0x1120; Col = -1; break;
        case IDR_TAB2_BeltFullRejuv:        IC = 0x1130; Col = -1; break;

        case IDR_TAB2_Slot1MinorHealing:    IC = 0x15A0; Col = 0; break;
        case IDR_TAB2_Slot1LightHealing:    IC = 0x15B0; Col = 0; break;
        case IDR_TAB2_Slot1Healing:         IC = 0x15C0; Col = 0; break;
        case IDR_TAB2_Slot1GreaterHealing:  IC = 0x15D0; Col = 0; break;
        case IDR_TAB2_Slot1SuperHealing:    IC = 0x15E0; Col = 0; break;
        case IDR_TAB2_Slot1MinorMana:       IC = 0x15F0; Col = 0; break;
        case IDR_TAB2_Slot1LightMana:       IC = 0x1600; Col = 0; break;
        case IDR_TAB2_Slot1Mana:            IC = 0x1610; Col = 0; break;
        case IDR_TAB2_Slot1GreaterMana:     IC = 0x1620; Col = 0; break;
        case IDR_TAB2_Slot1SuperMana:       IC = 0x1630; Col = 0; break;
        case IDR_TAB2_Slot1Rejuv:           IC = 0x1120; Col = 0; break;
        case IDR_TAB2_Slot1FullRejuv:       IC = 0x1130; Col = 0; break;

        case IDR_TAB2_Slot2MinorHealing:    IC = 0x15A0; Col = 1; break;
        case IDR_TAB2_Slot2LightHealing:    IC = 0x15B0; Col = 1; break;
        case IDR_TAB2_Slot2Healing:         IC = 0x15C0; Col = 1; break;
        case IDR_TAB2_Slot2GreaterHealing:  IC = 0x15D0; Col = 1; break;
        case IDR_TAB2_Slot2SuperHealing:    IC = 0x15E0; Col = 1; break;
        case IDR_TAB2_Slot2MinorMana:       IC = 0x15F0; Col = 1; break;
        case IDR_TAB2_Slot2LightMana:       IC = 0x1600; Col = 1; break;
        case IDR_TAB2_Slot2Mana:            IC = 0x1610; Col = 1; break;
        case IDR_TAB2_Slot2GreaterMana:     IC = 0x1620; Col = 1; break;
        case IDR_TAB2_Slot2SuperMana:       IC = 0x1630; Col = 1; break;
        case IDR_TAB2_Slot2Rejuv:           IC = 0x1120; Col = 1; break;
        case IDR_TAB2_Slot2FullRejuv:       IC = 0x1130; Col = 1; break;

        case IDR_TAB2_Slot3MinorHealing:    IC = 0x15A0; Col = 2; break;
        case IDR_TAB2_Slot3LightHealing:    IC = 0x15B0; Col = 2; break;
        case IDR_TAB2_Slot3Healing:         IC = 0x15C0; Col = 2; break;
        case IDR_TAB2_Slot3GreaterHealing:  IC = 0x15D0; Col = 2; break;
        case IDR_TAB2_Slot3SuperHealing:    IC = 0x15E0; Col = 2; break;
        case IDR_TAB2_Slot3MinorMana:       IC = 0x15F0; Col = 2; break;
        case IDR_TAB2_Slot3LightMana:       IC = 0x1600; Col = 2; break;
        case IDR_TAB2_Slot3Mana:            IC = 0x1610; Col = 2; break;
        case IDR_TAB2_Slot3GreaterMana:     IC = 0x1620; Col = 2; break;
        case IDR_TAB2_Slot3SuperMana:       IC = 0x1630; Col = 2; break;
        case IDR_TAB2_Slot3Rejuv:           IC = 0x1120; Col = 2; break;
        case IDR_TAB2_Slot3FullRejuv:       IC = 0x1130; Col = 2; break;

        case IDR_TAB2_Slot4MinorHealing:    IC = 0x15A0; Col = 3; break;
        case IDR_TAB2_Slot4LightHealing:    IC = 0x15B0; Col = 3; break;
        case IDR_TAB2_Slot4Healing:         IC = 0x15C0; Col = 3; break;
        case IDR_TAB2_Slot4GreaterHealing:  IC = 0x15D0; Col = 3; break;
        case IDR_TAB2_Slot4SuperHealing:    IC = 0x15E0; Col = 3; break;
        case IDR_TAB2_Slot4MinorMana:       IC = 0x15F0; Col = 3; break;
        case IDR_TAB2_Slot4LightMana:       IC = 0x1600; Col = 3; break;
        case IDR_TAB2_Slot4Mana:            IC = 0x1610; Col = 3; break;
        case IDR_TAB2_Slot4GreaterMana:     IC = 0x1620; Col = 3; break;
        case IDR_TAB2_Slot4SuperMana:       IC = 0x1630; Col = 3; break;
        case IDR_TAB2_Slot4Rejuv:           IC = 0x1120; Col = 3; break;
        case IDR_TAB2_Slot4FullRejuv:       IC = 0x1130; Col = 3; break;

        default:                            IC = 0x0000; Col = -1; break;
        }

        IC = FindItemInfoByItemCode(IC);
```

```
        MakeEmptyBelt(Col);

        if (Col < 0)
        {
            for(int x=0;x<4;x++)
            {
                for(int y=0;y<4;y++)
                {
                    Item *I = CreateItem(&Items,&ItemInfos[IC]);
                    I->SetCoordinates(CNT_BELT,x,y);
                }
            }
        }
        else
        {
            for(int y=0;y<4;y++)
            {
                Item *I = CreateItem(&Items,&ItemInfos[IC]);
                I->SetCoordinates(CNT_BELT,Col,y);
            }
        }

        UpdateTab2();
        InvalidateRect(hTabDialog,NULL,FALSE);
    }
    break;
case IDR_TAB2_RepairAll:
    {
        Item *I;
        // Repair all Items (Set Durability to MaxDurability)
        for(I = Items;I != 0;I = I->Next())
        {
            I->SetDurability(I->DurabilityMax());
        }
        UpdateTab2();
    }
    break;

// Save & Load Items into Load Buffer
case IDC_TAB2_Save:
    SaveItemFile(hWnd);
    InvalidateRect(hTabDialog,NULL,FALSE);
    break;
case IDC_TAB2_Load:
    LoadItemFile(hWnd);
    InvalidateRect(hTabDialog,NULL,FALSE);
    break;
case IDC_TAB2_AttrRandom:
    switch(HIWORD(wParam))
    {
    case BN_CLICKED:
        {
            if (SelItem)
            {
                SelItem->DWBHistory.StepAdd(SelItem);

                SelItem->SetMagicLevel((rand() % 100) + 1);
                SelItem->SetDWB(rand() + (rand() << 16));
                SelItem->Decoded = 0;
                UpdateTab2();
            }
        }
        break;
    }
break;
case IDC_TAB2_HistoryBack:
    if (HIWORD(wParam) == BN_CLICKED)
    {
        if (SelItem) {
            SelItem->DWBHistory.StepBack(SelItem);
            SelItem->Decoded = 0;
            UpdateTab2();
            InvalidateRect(hTabDialog,NULL,FALSE);
        }
    }
    break;
case IDC_TAB2_HistoryNext:
    if (HIWORD(wParam) == BN_CLICKED)
```

```
                {
                    if (SelItem) {
                        SelItem->DWBHistory.StepNext(SelItem);
                        SelItem->Decoded = 0;
                        UpdateTab2();
                        InvalidateRect(hTabDialog,NULL,FALSE);
                    }
                }
                break;
            case IDC_TAB2_Batch:
                switch(HIWORD(wParam))
                {
                case BN_CLICKED:
                    {
                        HMENU hMenu = GetSubMenu(hBatchMenu,1);

                        POINT Pos;
                        GetCursorPos(&Pos);

                        TrackPopupMenu(hMenu, TPM_LEFTALIGN | TPM_LEFTBUTTON,
                            Pos.x, Pos.y, 0, hWnd, NULL);
                    }
                    break;
                }
            break;
            case IDC_TAB2_ExpertMode:
                switch(HIWORD(wParam))
                {
                case BN_CLICKED:
                    {
                        if (IsDlgButtonChecked(hTabDialog,IDC_TAB2_ExpertMode))
                        {
#if BUGGYMESSAGES == 1
                            MessageBox(hWnd,"You really shouldn't do this! One wrong number kills your character!",PROGR
AMNAME,MB_OK | MB_ICONSTOP | MB_APPLMODAL);
#endif
                            if (hExpertBox) {
                                DestroyWindow(hExpertBox);
                            }

                            hExpertBox = CreateDialog(hInstance,MAKEINTRESOURCE(IDD_TAB2E),hTabDialog,(DLGPROC)&Tab2EDia
logProc);
                            UpdateTab2();
                        }
                        else
                        {
                            SendMessage(hExpertBox,WM_CLOSE,0,0);
                        }
                    }
                    break;
                }
                break;
#if INVGRIDS > 0
            case IDC_TAB2_OpenCube:
                if (HIWORD(wParam) == BN_CLICKED)
                {
                    if (hCube) {
                        DestroyWindow(hCube);
                    }
                    else {
                        CreateDialogParam(hInstance,MAKEINTRESOURCE(IDD_TAB2Grid),hTabDialog,(DLGPROC)&Tab2ExGridDialogP
roc,CNT_CUBE);
                    }
                }
                break;
            case IDC_TAB2_OpenBelt:
                if (HIWORD(wParam) == BN_CLICKED)
                {
                    if (hBelt) {
                        DestroyWindow(hBelt);
                    }
                    else {
                        CreateDialogParam(hInstance,MAKEINTRESOURCE(IDD_TAB2Grid),hTabDialog,(DLGPROC)&Tab2ExGridDialogP
roc,CNT_BELT);
                    }
                }
                break;
            case IDC_TAB2_OpenStash:
                if (HIWORD(wParam) == BN_CLICKED)
```

```
                {
                    if (hStash) {
                        DestroyWindow(hStash);
                    }
                    else {
                        CreateDialogParam(hInstance,MAKEINTRESOURCE(IDD_TAB2Grid),hTabDialog,(DLGPROC)&Tab2ExGridDialogP
roc,CNT_STASH);
                    }
                }
                break;
#endif
            }
        }
        return false;
    case WM_VALIDATE:
        {
            for(Item *i = Items;i != 0;i = i->Next())
            {
                if (i->Busy)
                {
                    ErrorBox("Search Treads still running!\nClose them first");
                    return true;
                }
            }
            break;
        }
    case WM_DESTROY:
        {
            if (hExpertBox) {
                SendMessage(hExpertBox,WM_CLOSE,0,0);
            }
        }
        return false;
    }
    return false;
}

#if INVGRIDS > 0

LRESULT CALLBACK Tab2ExGridDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
    switch(uMsg)
    {
    case WM_INITDIALOG:
        {
            RECT *r;

            if (lParam == CNT_CUBE)
            {
                r = &rCube;
                hCube = hWnd;

                CheckDlgButton(hTabDialog,IDC_TAB2_OpenCube,BST_CHECKED);
                SetWindowText(hWnd,"Cube Grid");



                r->right  = InvGrids.xCube * 28 + 1;
                r->bottom = InvGrids.yCube * 28 + 1;
            }
            else if (lParam == CNT_BELT)
            {
                r = &rBelt;
                hBelt = hWnd;

                CheckDlgButton(hTabDialog,IDC_TAB2_OpenBelt,BST_CHECKED);
                SetWindowText(hWnd,"Belt Grid");

                r->right  = InvGrids.xBelt * 28 + 1;
                r->bottom = InvGrids.yBelt * 28 + 1;
            }
            else if (lParam == CNT_STASH)
            {
                r = &rStash;
                hStash = hWnd;

                CheckDlgButton(hTabDialog,IDC_TAB2_OpenStash,BST_CHECKED);
                SetWindowText(hWnd,"Stash Grid");
```

```
                r->right  = InvGrids.xStash * 28 + 1;
                r->bottom = InvGrids.yStash * 28 + 1;
            }

            if (r->top)
            {
                SetWindowPos(hWnd,HWND_TOP,
                    r->left,r->top,
                    r->right,r->bottom,SWP_NOACTIVATE);
            }
            else
            {
                SetWindowPos(hWnd,HWND_TOP,
                    r->left,r->top,
                    r->right,r->bottom,SWP_NOMOVE | SWP_NOACTIVATE);
            }

            SetWindowLong(hWnd,GWL_USERDATA,lParam);

            ShowWindow(hWnd,SW_SHOWNOACTIVATE);
        }
        return true;
    case WM_COMMAND:
        {
            ProcessItemMessage(hWnd,wParam,lParam);
        }
        return false;
    case WM_CLOSE:
        DestroyWindow(hWnd);
        return false;
    case WM_DESTROY:
        {
            switch(GetWindowLong(hWnd,GWL_USERDATA))
            {
            case CNT_CUBE:
                CheckDlgButton(hTabDialog,IDC_TAB2_OpenCube,BST_UNCHECKED);
                hCube = 0;
                GetWindowRect(hWnd,&rCube);
                break;
            case CNT_BELT:
                CheckDlgButton(hTabDialog,IDC_TAB2_OpenBelt,BST_UNCHECKED);
                hBelt = 0;
                GetWindowRect(hWnd,&rBelt);
                break;
            case CNT_STASH:
                CheckDlgButton(hTabDialog,IDC_TAB2_OpenStash,BST_UNCHECKED);
                hStash = 0;
                GetWindowRect(hWnd,&rStash);
                break;
            }
        }
        return false;
    }
    return false;
}

#endif

bool FoundCube = false;
bool FindCube()
{
    for(class Item *i = Items;i != 0;i = i->Next())
    {
        if (i->ItemCode() == 0x1340 || i->ItemCode() == ' xob')
        {
            FoundCube = true;
            return true;
        }
    }
    FoundCube = false;
    return false;
}

void DrawGrid(HDC hdc,int Container)
{
    int boxesX,boxesY;

    switch(Container)
    {
```

```
    case CNT_INVENTORY:
        boxesX = InvGrids.xInventory;
        boxesY = InvGrids.yInventory;
        break;
    case CNT_STASH:
        boxesX = InvGrids.xStash;
        boxesY = InvGrids.yStash;
        break;
    case CNT_CUBE:
        boxesX = InvGrids.xCube;
        boxesY = FindCube() ? InvGrids.yCube : 0;
        break;
    case CNT_BELT:
        boxesX = InvGrids.xBelt;
        boxesY = InvGrids.xBelt;
        break;
    case CNT_COPYBUFFER:
        boxesX = 2;
        boxesY = 4;
    }

    SelectObject(hdc,hBrushBlack);
    SelectObject(hdc,hPenWhite);

    HDC hdcbitmap = CreateCompatibleDC(hdc);
    SelectObject(hdcbitmap,hBmpNotPlaceable);

    for(int y=3;y>=0;y--)
    {
        for(int x=0;x<boxesX;x++)
        {
            Rectangle(hdc,x*29,y*29,x*29+30,y*29+30);
            if (y >= boxesY)
            {
                BitBlt(hdc,x*29+1,y*29+1,28,28,hdcbitmap,0,0,SRCCOPY);
            }
        }
    }

    DeleteDC(hdcbitmap);
}
inline void DrawItemFrames(HDC hdc,Item *i,int x,int y,int sx,int sy)
{
    if (!i->Info)
        i->FindInfo();
    if (!i->Decoded)
        i->Decode();

    if (SelItem == i)
    {
        // Draw Selection Box
        SelectObject(hdc,hBrushNull);
        SelectObject(hdc,SelItem->Busy ? hPenBusy : hPenGreen);
        Rectangle(hdc,x,y,x+sx+1,y+sy+1);
    }
    if (i && !i->isWearable())
    {
        // Draw Red Frame
        SelectObject(hdc,hBrushNull);
        SelectObject(hdc,hPenUnwearable);
        Rectangle(hdc,x+1,y+1,x+sx-1,y+sy-1);
    }
}
void DrawItem(HDC hdc,class Item *i)
{
    int x = i->xPixelPos() +2;
    int y = i->yPixelPos() +2;
    int sx = i->xPixelSize();
    int sy = i->yPixelSize();
    HBITMAP bmp = i->GetBitmap();

    HDC hdcbitmap = CreateCompatibleDC(hdc);
    SelectObject(hdcbitmap,bmp);
    BitBlt(hdc,x,y,sx,sy,hdcbitmap,0,0,SRCCOPY);
    DeleteDC(hdcbitmap);

    DrawItemFrames(hdc,i,x,y,sx,sy);
}
void DrawItemSimple(HDC hdc,class Item *i)
```

```
{
    int sx = i->xPixelSize();
    int sy = i->yPixelSize();
    HBITMAP bmp = i->GetBitmap();

    HDC hdcbitmap = CreateCompatibleDC(hdc);
    SelectObject(hdcbitmap,bmp);
    BitBlt(hdc,1,1,sx,sy,hdcbitmap,0,0,SRCCOPY);
    DeleteDC(hdcbitmap);

    DrawItemFrames(hdc,i,1,1,sx,sy);
}

#define BODY_POTION     0
#define BODY_HELM       1
#define BODY_AMULET     2
#define BODY_ARMOR      3
#define BODY_HANDR      4
#define BODY_HANDL      5
#define BODY_RINGR      6
#define BODY_RINGL      7
#define BODY_BELT       8
#define BODY_BOOTS      9
#define BODY_GLOVES     10

struct
{
    int     xOrig,yOrig;
    int     xSize,ySize;
    int     equalto;
}
BodyParts[] =
{
    {   0,  0,      0,   0, 0 },                // Nothing: Belt
    { 131,  2,      60, 60, 0 },                // Helm
    { 204, 31,      30, 30, 0 },                // Amulet
    { 131, 74,      60, 88, 0 },                // Armor
    {  16, 46,      60,115, 0 },                // Weapon R
    { 247, 46,      60,115, 0 },                // Weapon L
    {  90,176,      30, 30, 7 },                // Ring R
    { 204,176,      30, 30, 6 },                // Ring L
    { 132,176,      60, 30, 0 },                // Belt
    { 247,176,      60, 60, 0 },                // Boots
    {  16,176,      60, 60 ,0 }                 // Gloves
};
int BodyPartsNum = 10;

void DrawItemBody(HDC hdc,class Item *i)
{
    if (i->Container() != CNT_BODY) return;

    int c = i->BodyCode();
    if (c <= 0 || c > BodyPartsNum) return;

    int x = BodyParts[c].xOrig + (BodyParts[c].xSize - i->xPixelSize()) / 2;
    int y = BodyParts[c].yOrig + (BodyParts[c].ySize - i->yPixelSize()) / 2;
    int sx = i->xPixelSize();
    int sy = i->yPixelSize();
    HBITMAP bmp = i->GetBitmap();

    HDC hdcbitmap = CreateCompatibleDC(hdc);
    SelectObject(hdcbitmap,bmp);
    BitBlt(hdc,x,y,sx,sy,hdcbitmap,0,0,SRCCOPY);
    DeleteDC(hdcbitmap);

    DrawItemFrames(hdc,i,x,y,sx,sy);
}

void UpdateTab2()
{
    if (!SelItem)
    {
        RTFStreamSend(hTabDialog,IDC_TAB2_RichText,"");

        EnableWindow(GetDlgItem(hTabDialog,IDC_TAB2_AttrRandom),FALSE);

        if (hExpertBox)
            UpdateTab2E();
        return;
```

```
    }

    if (!SelItem->Info) SelItem->FindInfo();
    SelItem->Decode();

    RTFStreamSend(hTabDialog,IDC_TAB2_RichText,SelItem->RichText());

    if (SelItem->Quality() == MAGICITEM ||
        SelItem->Quality() == RAREITEM ||
        SelItem->Quality() == SETITEM ||
        SelItem->Quality() == UNIQUEITEM)
    {
        EnableWindow(GetDlgItem(hTabDialog,IDC_TAB2_AttrRandom),TRUE);
        EnableWindow(GetDlgItem(hTabDialog,IDC_TAB2_HistoryBack),SelItem->DWBHistory.isBack());
        EnableWindow(GetDlgItem(hTabDialog,IDC_TAB2_HistoryNext),SelItem->DWBHistory.isNext());
    }
    else
    {
        EnableWindow(GetDlgItem(hTabDialog,IDC_TAB2_AttrRandom),FALSE);
        EnableWindow(GetDlgItem(hTabDialog,IDC_TAB2_HistoryBack),FALSE);
        EnableWindow(GetDlgItem(hTabDialog,IDC_TAB2_HistoryNext),FALSE);
    }

    if (hExpertBox)
        UpdateTab2E();
}

void MakeSelection(class Item *i)
{
    SelItem = i;

    UpdateTab2();

    InvalidateRect(hTabDialog,NULL,FALSE);
}

Item *GetItemAtBodyPlace(int BodyPlace)
{
    for(Item *I = Items;I != 0;I = I->Next())
    {
        if (I->Container() != CNT_BODY)
            continue;

        if (I->BodyCode() != BodyPlace) continue;

        return I;
    }
    return 0;
}

bool ItemPlaceable(int Container,int xPos,int yPos,int xSize,int ySize,int BodyPlace)
{
    // First Sanity Check
    switch(Container)
    {
    case CNT_CUBE:
        // Is there a cube in the Inventory
        if (!FoundCube) return false;

        // Check if it's a Horadric Cube in a Horadric Cube
        if (SelItem)
            if (SelItem->ItemCode() == 0x1340 || SelItem->ItemCode() == ' xob') return false;
        else if (SelInfo)
            if (SelInfo->ItemCode == 0x1340) return false;

        break;
    case CNT_BELT:
        // Check if it's a different Item than a Potion
        if (BodyPlace != 0) return false;

        break;
    }

    // Coordinates Checker
    switch(Container)
    {
    default:
        return false;
    case CNT_BODY:
```

```
    {
        for(int z=1;z<=BodyPartsNum;z++)
        {
            if (xPos >= BodyParts[z].xOrig && xPos <= BodyParts[z].xOrig + BodyParts[z].xSize &&
                yPos >= BodyParts[z].yOrig && yPos <= BodyParts[z].yOrig + BodyParts[z].ySize)
            {
                // Check if occupied
                Item *O = GetItemAtBodyPlace(z);
                if (O != 0 && O != SelItem) return false;

                if (BodyPlace == BODY_RINGR && (z == BODY_RINGL || z == BODY_RINGR)) return true;
                if ((BodyPlace == BODY_HANDR || BodyPlace == BODY_HANDL) && z == BODY_HANDL) {
                    O = GetItemAtBodyPlace(BODY_HANDR);
                    if (O && O->Info && O->Info->Hands > 1) return false;
                    else return true;
                }
                if ((BodyPlace == BODY_HANDR || BodyPlace == BODY_HANDL) && z == BODY_HANDR) {
                    O = GetItemAtBodyPlace(BODY_HANDL);
                    if (O && O->Info && O->Info->Hands > 1) return false;
                    else return true;
                }
                if (BodyPlace == z) return true;
                return false;
            }
        }
        return false;
    }
    return false;
case CNT_COPYBUFFER:
    {
        if (CopyBuffer)
            return false;
        else
            return true;
    }
case CNT_INVENTORY:
case CNT_STASH:
case CNT_CUBE:
case CNT_BELT:
    {
        xPos = (xPos-2) / 29;
        yPos = (yPos-2) / 29;

        if (Container == CNT_INVENTORY) {
            if (xPos + xSize > InvGrids.xInventory) return false;
            if (yPos + ySize > InvGrids.yInventory) return false;
        }
        else if (Container == CNT_STASH) {
            if (xPos + xSize > InvGrids.xStash) return false;
            if (yPos + ySize > InvGrids.yStash) return false;
        }
        else if (Container == CNT_CUBE) {
            if (xPos + xSize > InvGrids.xCube) return false;
            if (yPos + ySize > InvGrids.yCube) return false;
        }
        else if (Container == CNT_BELT) {
            if (xPos + xSize > InvGrids.xBelt) return false;
            if (yPos + ySize > InvGrids.yBelt) return false;
        }

        for(Item *I = Items;I != 0;I = I->Next())
        {
            if (!I->Info)
                I->FindInfo();

            if (I->Container() != Container)
                continue;

            if (SelItem && SelItem == I) continue;

            if (xPos >= I->xPos() + I->xSize()) continue;
            if (xPos + xSize <= I->xPos()) continue;

            if (yPos >= I->yPos() + I->ySize()) continue;
            if (yPos + ySize <= I->yPos()) continue;

            return false;
        }
        return true;
```

```
        }
    }
}

bool SelItemPlaceable(int IntoContainer,int xPos,int yPos,POINT Pos,HWND hPos)
{
    bool retbool = false;

    int xSize,ySize,BodyPlace;

    if (SelItem) {
        xSize = SelItem->xSize();
        ySize = SelItem->ySize();
        BodyPlace = SelItem->BodyPlace();
    }
    else {
        xSize = SelInfo->SizeX;
        ySize = SelInfo->SizeY;
        BodyPlace = SelInfo->BodyPlace;
    }

    return ItemPlaceable(IntoContainer,xPos,yPos,xSize,ySize,BodyPlace);
}

bool SelItemDeleteable()
{
    if (!SelItem) return false;
    if (SelItem == CopyBuffer) return true;

    // Check if we're deleting the Horadric Cube
    if (SelItem->ItemCode() == 0x1340 || SelItem->ItemCode() == ' xob')
    {
        // Count Cubes :-)
        int cn = 0;
        for(Item *I = Items;I != 0;I = I->Next())
            if (I->ItemCode() == 0x1340 || I->ItemCode() == ' xob')
                cn++;

        // Last Cube ?
        if (cn == 1)
            for(Item *i = Items;i != 0;i = i->Next())
                if (i->Container() == CNT_CUBE)
                {
                    MessageBox(hTabDialog,"Horadric Cube is not empty!\nPlease delete all contained Items first.",PROGRA
MNAME,MB_OK | MB_ICONSTOP | MB_APPLMODAL);
                    return false;
                }
    }
    return true;
}

void ShowContextMenu(HWND hWnd,Item *I,int xPos,int yPos)
{
    POINT Pos;
    GetCursorPos(&Pos);

    if (I)
    {
        if (I->Busy) return;
        if (!I->Info) I->FindInfo();

        HMENU hMenu = CreatePopupMenu();

        // Selection Popup
        if (I->Info->QualityMask)
        {
            HMENU hSubMenu = CreatePopupMenu();
            if (I->Info->QualityMask & 1)
            {
                AppendMenu(hSubMenu,MF_STRING | (I->Quality() == CRUDEITEM ? MF_CHECKED : MF_UNCHECKED),
                    IDR_TAB2_SetQualityCrude,"Crude");
                AppendMenu(hSubMenu,MF_STRING | (I->Quality() == USUALITEM ? MF_CHECKED : MF_UNCHECKED),
                    IDR_TAB2_SetQualityRegular,"Regular");
                AppendMenu(hSubMenu,MF_STRING | (I->Quality() == SUPERIORITEM ? MF_CHECKED : MF_UNCHECKED),
                    IDR_TAB2_SetQualitySuperior,"Superior");
            }
            if (I->Info->QualityMask & 2)
            {
                AppendMenu(hSubMenu,MF_STRING | (I->Quality() == MAGICITEM ? MF_CHECKED : MF_UNCHECKED),
```

```
            IDR_TAB2_SetQualityMagical,"Magical");
        AppendMenu(hSubMenu,MF_STRING | (I->Quality() == RAREITEM ? MF_CHECKED : MF_UNCHECKED),
            IDR_TAB2_SetQualityRare,"Rare");
    }
    if (I->Info->QualityMask & 4 && I->Quality() == SETITEM)
    {
        AppendMenu(hSubMenu,MF_STRING | (I->Quality() == SETITEM ? MF_CHECKED : MF_UNCHECKED),
            IDR_TAB2_SetQualitySet,"Set Item");
    }
    if (I->Info->QualityMask & 8 && I->Quality() == UNIQUEITEM)
    {
        AppendMenu(hSubMenu,MF_STRING | (I->Quality() == UNIQUEITEM ? MF_CHECKED : MF_UNCHECKED),
            IDR_TAB2_SetQualityUnique,"Unique Item");
    }
    AppendMenu(hMenu,MF_POPUP,(UINT) hSubMenu,"Item Quality");
}

if (I->Quality() == MAGICITEM) {
    AppendMenu(hMenu,MF_STRING,IDR_TAB2_AttributeSelect,"Select Magical Attributes");
}
if (I->Quality() == RAREITEM) {
    AppendMenu(hMenu,MF_STRING,IDR_TAB2_AttributeSelect,"Select Rare Attributes");
}

if (I->Socketed()) {
    AppendMenu(hMenu,MF_STRING,IDR_TAB2_GemsEdit,"Change Inserted Gems");
}

if (I->ItemRecordID() == IT_103EAR || I->ItemRecordID() == IT_104EAR ) {
    AppendMenu(hMenu,MF_STRING,IDR_TAB2_SetEarProperties,"Change Ear Properties");
    AppendMenu(hMenu,MF_SEPARATOR,0,0);
}

if (I->ItemCode() == 0x1190 || I->ItemCode() == ' nir') {
    AppendMenu(hMenu,MF_STRING,IDR_TAB2_ChangeRingImage,"Change Ring Image");
}
if (I->ItemCode() == 0x1170 || I->ItemCode() == ' uma') {
    AppendMenu(hMenu,MF_STRING,IDR_TAB2_ChangeAmuletImage,"Change Amulet Image");
}

// Quantity Selection Entry
if (I->Info->Quantity)
{
    AppendMenu(hMenu,MF_STRING,IDR_TAB2_SetQuantity,"Set Quantity");
}

// Durability Selection Entry
if (I->Info->Durability)
{
    AppendMenu(hMenu,MF_STRING,IDR_TAB2_SetDurability,"Set Durability");
}

// Defense Selection Entry
if (I->Info->ACMin > 0)
{
    AppendMenu(hMenu,MF_STRING,IDR_TAB2_SetDefense,"Select Defense");
}

// Seperator
if (I->Info->Quantity || I->Info->Durability || I->Info->QualityMask)
    AppendMenu(hMenu,MF_SEPARATOR,0,0);

// Add Item Flags Selections
if (I->Socketed() || I->Socketable() || RegOptions.AllItemsSocketable)
{
    AppendMenu(hMenu,MF_STRING | (I->Socketed() ? MF_CHECKED : MF_UNCHECKED),
        IDR_TAB2_Socketed,"Socketed");
}

AppendMenu(hMenu,MF_STRING | (I->Identified() ? MF_CHECKED : MF_UNCHECKED),
    IDR_TAB2_Identified,"Identified");

AppendMenu(hMenu,MF_STRING | (I->Starter() ? MF_CHECKED : MF_UNCHECKED),
    IDR_TAB2_Starter,"Starter Item");

AppendMenu(hMenu,MF_SEPARATOR,0,0);

AppendMenu(hMenu,MF_STRING,IDR_TAB2_Cut,"Cut to Clipboard");
AppendMenu(hMenu,MF_STRING,IDR_TAB2_Copy,"Copy to Clipboard");
```

```
        AppendMenu(hMenu,MF_STRING,IDR_TAB2_Delete,"Delete");

        // Show Context Menu
        TrackPopupMenu(hMenu, TPM_LEFTALIGN | TPM_LEFTBUTTON,
            Pos.x, Pos.y, 0, hTabDialog, NULL);

        DestroyMenu(hMenu);
    }
    else
    {
        if (!IsClipboardFormatAvailable(ClipBoardFormat) && !IsClipboardFormatAvailable(ClipBoardFormatJohnDoe)) {

            HMENU hMenu = CreatePopupMenu();

            AppendMenu(hMenu,MF_STRING | MF_GRAYED,0,"Paste: No Item in Clipboard");

            // Show Context Menu
            TrackPopupMenu(hMenu, TPM_LEFTALIGN | TPM_LEFTBUTTON,
                Pos.x, Pos.y, 0, hWnd, NULL);

            DestroyMenu(hMenu);
            return;
        }

        if (!OpenClipboard(hMainDialog))
            return;

        bool ok = false;

        if (IsClipboardFormatAvailable(ClipBoardFormat))
        {
            HGLOBAL Clip = GetClipboardData(ClipBoardFormat);

            struct JamClipboardHead *JamHead = (JamClipboardHead*) GlobalLock(Clip);

            ok = ItemPlaceable(ContainerFromHWND(hWnd),xPos,yPos,JamHead->xSize,JamHead->ySize,JamHead->BodyPlace);

            GlobalUnlock(Clip);
        }
        else if (IsClipboardFormatAvailable(ClipBoardFormatJohnDoe))
        {
            if (hWnd != hCopyBuffer) {
                HMENU hMenu = CreatePopupMenu();

                AppendMenu(hMenu,MF_STRING | MF_GRAYED,0,"Paste: John Doe's Item only in CopyBuffer");

                // Show Context Menu
                TrackPopupMenu(hMenu, TPM_LEFTALIGN | TPM_LEFTBUTTON,
                    Pos.x, Pos.y, 0, hWnd, NULL);

                DestroyMenu(hMenu);
                return;
            }
            ok = true;
        }

        CloseClipboard();

        HMENU hMenu = CreatePopupMenu();

        if (!ok) {
            AppendMenu(hMenu,MF_STRING | MF_GRAYED,0,"Paste: Not Placeable here!");
        }
        else {
            AppendMenu(hMenu,MF_STRING,IDR_TAB2_Paste,"Paste from Clipboard");
        }

        // Show Context Menu
        TrackPopupMenu(hMenu, TPM_LEFTALIGN | TPM_LEFTBUTTON,
            Pos.x, Pos.y, 0, hWnd, NULL);

        xStartPos = xPos;
        yStartPos = yPos;

        DestroyMenu(hMenu);
    }
}

LRESULT CALLBACK Tab2InventoryProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
```

```
{
    switch(uMsg)
    {
    case WM_PAINT:
        {
            PAINTSTRUCT ps;
            HDC hdc = BeginPaint(hWnd,&ps);

            if (hWnd == hInv)
            {
                DrawGrid(hdc,CNT_INVENTORY);
                for(Item *i = Items;i != 0;i = i->Next())
                {
                    if (i->Container() == CNT_INVENTORY)
                    {
                        DrawItem(hdc,i);
                    }
                }
            }
            else if (hWnd == hBelt)
            {
                DrawGrid(hdc,CNT_BELT);
                for(Item *i = Items;i != 0;i = i->Next())
                {
                    if (i->Container() == CNT_BELT)
                    {
                        DrawItem(hdc,i);
                    }
                }
            }
            else if (hWnd == hCube)
            {
                DrawGrid(hdc,CNT_CUBE);
                for(Item *i = Items;i != 0;i = i->Next())
                {
                    if (i->Container() == CNT_CUBE)
                    {
                        DrawItem(hdc,i);
                    }
                }
            }
            else if (hWnd == hStash)
            {
                DrawGrid(hdc,CNT_STASH);
                for(Item *i = Items;i != 0;i = i->Next())
                {
                    if (i->Container() == CNT_STASH)
                    {
                        DrawItem(hdc,i);
                    }
                }
            }
            else if (hWnd == hCopyBuffer)
            {
                DrawGrid(hdc,CNT_COPYBUFFER);
                if (CopyBuffer)
                {
                    DrawItemSimple(hdc,CopyBuffer);
                }
            }
            else if (hWnd == hBody)
            {
                // Put bg image
                {
                    HDC hdcbitmap = CreateCompatibleDC(hdc);
                    SelectObject(hdcbitmap,hBmpBodyWhole);
                    BITMAP bmpinfo;
                    GetObject(hBmpBodyWhole,sizeof bmpinfo,&bmpinfo);
                    BitBlt(hdc,0,0,bmpinfo.bmWidth,bmpinfo.bmHeight,hdcbitmap,0,0,SRCCOPY);
                    DeleteDC(hdcbitmap);
                }

                for(Item *i = Items;i != 0;i = i->Next())
                {
                    if (i->Container() == CNT_BODY)
                    {
                        DrawItemBody(hdc,i);
                    }
                }
```

```
        }

        EndPaint(hWnd, &ps);
    }
    return true;
case WM_LBUTTONDOWN:
    {
        SetFocus(hWnd);

        int xPos = LOWORD(lParam);
        int yPos = HIWORD(lParam);

        if (hWnd == hInv || hWnd == hBelt || hWnd == hCube || hWnd == hStash)
        {
            int cd = ContainerFromHWND(hWnd);

            for(Item *i = Items;i != 0;i = i->Next())
            {
                if (i->Container() == cd)
                {
                    if (i->isInRegion(xPos -2,yPos -2))
                    {
                        MakeSelection(i);
                        break;
                    }
                }
            }
        }
        else if (hWnd == hBody)
        {
            for(int z=1;z<=BodyPartsNum;z++)
            {
                if (xPos >= BodyParts[z].xOrig && xPos <= BodyParts[z].xOrig + BodyParts[z].xSize &&
                    yPos >= BodyParts[z].yOrig && yPos <= BodyParts[z].yOrig + BodyParts[z].ySize)
                {
                    for(Item *i = Items;i != 0;i = i->Next())
                    {
                        if (i->Container() == CNT_BODY)
                        {
                            if (z != i->BodyCode()) continue;
                            MakeSelection(i);
                            break;
                        }
                    }
                }
            }
        }
        else if (hWnd == hCopyBuffer)
        {
            if (CopyBuffer)
            {
                MakeSelection(CopyBuffer);
            }
        }

        if (SelItem && !SelItem->Busy)
        {
            MouseButtonDown = true;
            xStartPos = xPos;
            yStartPos = yPos;
            FindCube();
            SetCapture(hWnd);
        }
    }
    return true;
case WM_RBUTTONDOWN:
    {
        if (MouseButtonDown) return false;
        SetFocus(hWnd);

        int xPos = LOWORD(lParam);
        int yPos = HIWORD(lParam);

        POINT pt = { xPos, yPos };
        ClientToScreen(hWnd,&pt);

        if (hWnd == hInv || hWnd == hBelt || hWnd == hCube || hWnd == hStash)
        {
            int cd = ContainerFromHWND(hWnd);
```

```
            for(Item *i = Items;i != 0;i = i->Next())
            {
                if (i->Container() == cd)
                {
                    if (i->isInRegion(xPos -2,yPos -2))
                    {
                        MakeSelection(i);
                        break;
                    }
                }
            }
            if (i == 0)
                MakeSelection(0);
        }
        else if (hWnd == hBody)
        {
            MakeSelection(0);
            for(int z=1;z<=BodyPartsNum;z++)
            {
                if (xPos >= BodyParts[z].xOrig && xPos <= BodyParts[z].xOrig + BodyParts[z].xSize &&
                    yPos >= BodyParts[z].yOrig && yPos <= BodyParts[z].yOrig + BodyParts[z].ySize)
                {
                    for(Item *i = Items;i != 0;i = i->Next())
                    {
                        if (i->Container() == CNT_BODY)
                        {
                            if (z != i->BodyCode()) continue;
                            MakeSelection(i);
                            break;
                        }
                    }
                    if (i == 0)
                        MakeSelection(0);
                }
            }
        }
        else if (hWnd == hCopyBuffer)
        {
            if (CopyBuffer)
                MakeSelection(CopyBuffer);
            else
                MakeSelection(0);
        }

        ShowContextMenu(hWnd,SelItem,xPos,yPos);
    }
    return true;
case WM_CAPTURECHANGED:
    {
        MouseButtonDown = false;
    }
    return true;
case WM_MOUSEMOVE:
    {
        if (!MouseButtonDown) return false;
        if (! (abs( xStartPos - LOWORD(lParam) ) > xDragTol ||
               abs( yStartPos - HIWORD(lParam) ) > yDragTol) ) return false;

        MouseButtonMoved = true;

        // Mouse Position
        POINT Pos;
        GetCursorPos(&Pos);

        HWND hPos = WindowFromPoint(Pos);

        RECT rect;
        GetWindowRect(hPos,&rect);
        int xPos = Pos.x - rect.left;
        int yPos = Pos.y - rect.top;

        HCURSOR hSetCur = hCurNo;

        int CNT = ContainerFromHWND(hPos);

        if (CNT == CNT_COPYBUFFER)
        {
            if (SelItemPlaceable(CNT,xPos,yPos,Pos,hPos))
```

```
                hSetCur = hCurMoveCopy;
            else
                hSetCur = hCurCross;
        }
        else if (CNT)
        {
            if (SelItemPlaceable(CNT,xPos,yPos,Pos,hPos))
                hSetCur = hCurMove;
            else
                hSetCur = hCurCross;
        }

        SetCursor(hSetCur);
    }
    return true;
case WM_LBUTTONUP:
    {
        MouseButtonDown = false;
        ReleaseCapture();

        if (!MouseButtonMoved) break;
        MouseButtonMoved = false;

        // Mouse Position
        POINT Pos;
        GetCursorPos(&Pos);

        HWND hPos = WindowFromPoint(Pos);

        RECT rect;
        GetWindowRect(hPos,&rect);
        int xPos = Pos.x - rect.left;
        int yPos = Pos.y - rect.top;

        int CNT = ContainerFromHWND(hPos);

        if (SelItemPlaceable(CNT,xPos,yPos,Pos,hPos))
        {
            if (!SelItem && CNT != CNT_COPYBUFFER)
            {
                // Create New Item
                SelItem = CreateItem(&Items,SelInfo);
            }
            else if (!SelItem && CNT == CNT_COPYBUFFER)
            {
                // Create New Item in Copy Buffer
                if (CopyBuffer) {
                    delete CopyBuffer;
                    CopyBuffer = 0;
                }

                SelItem = CreateItem(&CopyBuffer,SelInfo);
            }
            else if (SelItem == CopyBuffer)
            {
                SelItem = ItemDuplicate(CopyBuffer);
                SelItem->LoadItemRecord(CopyBuffer->GetItemRecord());

                if (CopyBuffer->Gems)
                {
                    for(Item *G = CopyBuffer->Gems;G != 0;G = G->Next())
                    {
                        Item *GN = ItemDuplicate(G);
                        GN->LoadItemRecord(G->GetItemRecord());

                        if (SelItem->Gems)
                            SelItem->Gems->Attach(GN);
                        else
                            SelItem->Gems = GN;
                    }
                }

                Items->Attach(SelItem);
            }

            switch(CNT)
            {
            case CNT_INVENTORY:
            case CNT_STASH:
```

```
            case CNT_CUBE:
            case CNT_BELT:
                {
                    xPos = (xPos-2) / 29;
                    yPos = (yPos-2) / 29;

                    SelItem->SetCoordinates(CNT,xPos,yPos);
                }
                break;
            case CNT_BODY:
                {
                    for(int z=1;z<=BodyPartsNum;z++)
                    {
                        if (xPos >= BodyParts[z].xOrig && xPos <= BodyParts[z].xOrig + BodyParts[z].xSize &&
                            yPos >= BodyParts[z].yOrig && yPos <= BodyParts[z].yOrig + BodyParts[z].ySize)
                        {
                            SelItem->SetCoordinates(CNT,z,0);
                        }
                    }
                }
                break;
            case CNT_COPYBUFFER:
                {
                    if (SelItem != CopyBuffer) {

                        if (CopyBuffer) {
                            delete CopyBuffer;
                            CopyBuffer = 0;
                        }

                        CopyBuffer = ItemDuplicate(SelItem);
                        CopyBuffer->LoadItemRecord(SelItem->GetItemRecord());

                        if (SelItem->Gems)
                        {
                            for(Item *G = SelItem->Gems;G != 0;G = G->Next())
                            {
                                Item *GN = ItemDuplicate(G);
                                GN->LoadItemRecord(G->GetItemRecord());

                                if (CopyBuffer->Gems)
                                    CopyBuffer->Gems->Attach(GN);
                                else
                                    CopyBuffer->Gems = GN;
                            }
                        }

                        SelItem = CopyBuffer;
                    }
                }
                break;
            }
            MakeSelection(SelItem);
            UpdateTab2();

            InvalidateRect(hTabDialog,NULL,FALSE);
            UpdateWindow(hTabDialog);
        }
        else
        {
            MessageBeep(MB_ICONHAND);
        }
    }
    return true;
case WM_KEYDOWN:
    {
        switch((int)wParam)
        {
        case VK_DELETE:
            if (MouseButtonDown) break;
            if (!SelItem) break;
            if (SelItem->Busy) break;

            if (SelItemDeleteable()) {
                if (SelItem == CopyBuffer) {
                    delete CopyBuffer;
                    CopyBuffer = 0;
                }
                else {
```

```
                    SelItem->Delete();
                }
                MakeSelection(0);
            }
            break;
        }
    }
    break;
case WM_COMMAND:
    if (LOWORD(wParam) == IDR_TAB2_Paste)
    {
        if (!IsClipboardFormatAvailable(ClipBoardFormat) && !IsClipboardFormatAvailable(ClipBoardFormatJohnDoe))
            return false;

        if (!OpenClipboard(hMainDialog))
            return false;

        int xPos = xStartPos;
        int yPos = yStartPos;

        int CNT = ContainerFromHWND(hWnd);

        Item *N = 0;

        if (IsClipboardFormatAvailable(ClipBoardFormat))
        {
            HGLOBAL Clip = GetClipboardData(ClipBoardFormat);

            BYTE *ClipData = (BYTE*)GlobalLock(Clip);

            struct JamClipboardHead *JamHead = (JamClipboardHead*) ClipData;

            if (!ItemPlaceable(CNT,xPos,yPos,JamHead->xSize,JamHead->ySize,JamHead->BodyPlace)) {
                GlobalUnlock(Clip);
                CloseClipboard();
                return false;
            }

            BYTE *o = ClipData + sizeof (JamClipboardHead);
            int s = GlobalSize(Clip);

            switch(JamHead->IID)
            {
            default:
                GlobalUnlock(Clip);
                CloseClipboard();
                return false;
            case IT_103:
                N = new Item103;

                N->LoadItemRecord(o);
                o += N->ItemRecordLength();

                while(s - (o - ClipData) >= 27)
                {
                    Item *G = new Item103;
                    G->LoadItemRecord(o);
                    o += G->ItemRecordLength();

                    if (N->Gems)
                        N->Gems->Attach(G);
                    else
                        N->Gems = G;
                }
                break;
            case IT_103EAR:
                N = new Item103Ear;

                N->LoadItemRecord(o);
                o += N->ItemRecordLength();
                break;
            case IT_104EX:
                N = new Item104Ex;

                N->LoadItemRecord(o);
                o += N->ItemRecordLength();

                while(s - (o - ClipData) >= 15)
                {
```

```
            Item *G = new Item104Sm;
            G->LoadItemRecord(o);
            o += G->ItemRecordLength();

            if (N->Gems)
                N->Gems->Attach(G);
            else
                N->Gems = G;
        }
        break;
    case IT_104SM:
        N = new Item104Sm;

        N->LoadItemRecord(o);
        o += N->ItemRecordLength();
        break;
    case IT_104EAR:
        N = new Item104Ear;

        N->LoadItemRecord(o);
        o += N->ItemRecordLength();
        break;
    }

    GlobalUnlock(Clip);
}
else if (IsClipboardFormatAvailable(ClipBoardFormatJohnDoe))
{
    HGLOBAL Clip = GetClipboardData(ClipBoardFormatJohnDoe);

    BYTE *ClipData = (BYTE*)GlobalLock(Clip);

    DWORD nGems = *(DWORD*)ClipData;

    BYTE *o = ClipData + 4;
    int s = GlobalSize(Clip);

    N = new Item103;
    N->LoadItemRecord(o);
    o += N->ItemRecordLength();

    for(DWORD z=0;z<nGems && s - (o-ClipData) >= 27;z++)
    {
        Item *G = new Item103;
        G->LoadItemRecord(o);
        o += G->ItemRecordLength();

        if (N->Gems)
            N->Gems->Attach(G);
        else
            N->Gems = G;
    }

    GlobalUnlock(Clip);
}

CloseClipboard();

if (N)
{
    switch(CNT)
    {
    case CNT_INVENTORY:
    case CNT_STASH:
    case CNT_CUBE:
    case CNT_BELT:
        {
            xPos = (xPos-2) / 29;
            yPos = (yPos-2) / 29;

            N->SetCoordinates(CNT,xPos,yPos);

            if (Items)
                Items->Attach(N);
            else
                Items = N;
        }
        break;
    case CNT_BODY:
```

```
                {
                    for(int z=1;z<=BodyPartsNum;z++)
                    {
                        if (xPos >= BodyParts[z].xOrig && xPos <= BodyParts[z].xOrig + BodyParts[z].xSize &&
                            yPos >= BodyParts[z].yOrig && yPos <= BodyParts[z].yOrig + BodyParts[z].ySize)
                        {
                            N->SetCoordinates(CNT,z,0);

                            if (Items)
                                Items->Attach(N);
                            else
                                Items = N;
                        }
                    }
                }
                break;
            case CNT_COPYBUFFER:
                {
                    if (CopyBuffer) {
                        delete CopyBuffer;
                        CopyBuffer = 0;
                    }
                    CopyBuffer = N;
                }
                break;
            }

            MakeSelection(N);
            UpdateTab2();

            InvalidateRect(hTabDialog,NULL,FALSE);
            UpdateWindow(hTabDialog);

        }
    }
    break;
  }
  return DefWindowProc(hWnd,uMsg,wParam,lParam);
}
```

```
// TAB2E.cpp from D2E

#include "JamellaD2E.h"

#define IDT_TIMER       12345

// TAB2E ToolBox
HWND hExpertBox;

const int nRawData = 32;

struct
{
    int     DialogID;
    HWND    hWindow;
}
Controls[] =
{
    { IDC_TAB2E_Raw00 },
    { IDC_TAB2E_Raw01 },
    { IDC_TAB2E_Raw02 },
    { IDC_TAB2E_Raw03 },
    { IDC_TAB2E_Raw04 },
    { IDC_TAB2E_Raw05 },
    { IDC_TAB2E_Raw06 },
    { IDC_TAB2E_Raw07 },
    { IDC_TAB2E_Raw08 },
    { IDC_TAB2E_Raw09 },
    { IDC_TAB2E_Raw0A },
    { IDC_TAB2E_Raw0B },
    { IDC_TAB2E_Raw0C },
    { IDC_TAB2E_Raw0D },
    { IDC_TAB2E_Raw0E },
    { IDC_TAB2E_Raw0F },
    { IDC_TAB2E_Raw10 },
    { IDC_TAB2E_Raw11 },
    { IDC_TAB2E_Raw12 },
    { IDC_TAB2E_Raw13 },
    { IDC_TAB2E_Raw14 },
    { IDC_TAB2E_Raw15 },
    { IDC_TAB2E_Raw16 },
    { IDC_TAB2E_Raw17 },
    { IDC_TAB2E_Raw18 },
    { IDC_TAB2E_Raw19 },
    { IDC_TAB2E_Raw1A },
    { IDC_TAB2E_Raw1B },
    { IDC_TAB2E_Raw1C },
    { IDC_TAB2E_Raw1D },
    { IDC_TAB2E_Raw1E },
    { IDC_TAB2E_Raw1F },

    { IDC_TAB2E_ItemCode },
    { IDC_TAB2E_UniqueCode },
    { IDC_TAB2E_DWA },
    { IDC_TAB2E_DWB },
    { IDC_TAB2E_MagicLevel },
    { IDC_TAB2E_GemNum },
};

inline char* MakeHex(int v,int digits)
{
    sprintf(buffer,"%0*X",digits,v);
    return buffer;
}
inline char* MakeDez(int v)
{
    sprintf(buffer,"%d",v);
    return buffer;
}
inline void ConvHex(int DlgID,BYTE *dest)
{
    GetDlgItemText(hExpertBox,DlgID,buffer,256);
    if(strlen(buffer) > 2) return;
    char *test;
    int x = strtoul(buffer,&test,16);
    if (*test == 0) *dest = x;
}
inline void ConvHex(int DlgID,WORD *dest)
{
```

```
    GetDlgItemText(hExpertBox,DlgID,buffer,256);
    if(strlen(buffer) > 4) return;
    char *test;
    int x = strtoul(buffer,&test,16);
    if (*test == 0) *dest = x;
}
inline void ConvHex(int DlgID,DWORD *dest)
{
    GetDlgItemText(hExpertBox,DlgID,buffer,256);
    if(strlen(buffer) > 8) return;
    char *test;
    int x = strtoul(buffer,&test,16);
    if (*test == 0) *dest = x;
}
inline bool ConvDezProp(int DlgID,int *dest)
{
    GetDlgItemText(hExpertBox,DlgID,buffer,256);
    char *test;
    int x = strtoul(buffer,&test,10);
    if (*test == 0) {
        *dest = x;
        return true;
    }
    return false;
}
inline bool ConvHexProp(int DlgID,int *dest)
{
    GetDlgItemText(hExpertBox,DlgID,buffer,256);
    char *test;
    int x = strtoul(buffer,&test,16);
    if (*test == 0) {
        *dest = x;
        return true;
    }
    return false;
}
void UpdateTab2E()
{
    if (!hExpertBox) return;

    if (SelItem)
    {
        for(int z=0;z< sizeof Controls / sizeof Controls[0];z++)
        {
            EnableWindow(Controls[z].hWindow,TRUE);
        }

        memset(buffer,0,sizeof buffer);

        SetDlgItemText(hExpertBox,IDC_TAB2E_ItemRecordID,SelItem->ItemRecordName());

        BYTE *RawData = SelItem->GetItemRecord();
        int RawDataLen = SelItem->ItemRecordLength();

        for(int n=0;n<nRawData;n++)
        {
            if (n < RawDataLen)
                SetDlgItemText(hExpertBox,Controls[n].DialogID,MakeHex(RawData[n],2));
            else
                SetDlgItemText(hExpertBox,Controls[n].DialogID,"");
        }

        SetDlgItemText(hExpertBox,IDC_TAB2E_ItemCode,MakeHex(SelItem->ItemCode(),8));
        SetDlgItemText(hExpertBox,IDC_TAB2E_ItemCodeChar,CodeString(SelItem->ItemCode()));
        SetDlgItemText(hExpertBox,IDC_TAB2E_UniqueCode,MakeHex(SelItem->UniqueCode(),2));

        SetDlgItemText(hExpertBox,IDC_TAB2E_Container,MakeDez(SelItem->Container()));
        SetDlgItemText(hExpertBox,IDC_TAB2E_Xoord,MakeDez(SelItem->xPos()));
        SetDlgItemText(hExpertBox,IDC_TAB2E_Yoord,MakeDez(SelItem->yPos()));
        SetDlgItemText(hExpertBox,IDC_TAB2E_BodyCode,MakeDez(SelItem->BodyCode()));

        SetDlgItemText(hExpertBox,IDC_TAB2E_DWA,MakeHex(SelItem->DWA(),8));
        SetDlgItemText(hExpertBox,IDC_TAB2E_DWB,MakeHex(SelItem->DWB(),8));
        SetDlgItemText(hExpertBox,IDC_TAB2E_MagicLevel,MakeHex(SelItem->MagicLevel(),2));

        SetDlgItemText(hExpertBox,IDC_TAB2E_GemNum,MakeDez(SelItem->GemNum()));
/*
        if (SelItem->MagicPrefix)
        {
```

```
            SetDlgItemText(hExpertBox,IDC_TAB2E_Val8,MakeDez(SelItem->MagicPrefix->N));
            SetDlgItemText(hExpertBox,IDC_TAB2E_Val9,MakeDez(SelItem->MagicPrefix->nMod));
            SetDlgItemText(hExpertBox,IDC_TAB2E_Val10,MakeDez(SelItem->modMagicPrefix));
            SetDlgItemText(hExpertBox,IDC_TAB2E_Val11,MakeDez(SelItem->modpickMagicPrefix));
        }
        else
        {
            SetDlgItemText(hExpertBox,IDC_TAB2E_Val8,"");
            SetDlgItemText(hExpertBox,IDC_TAB2E_Val9,"");
            SetDlgItemText(hExpertBox,IDC_TAB2E_Val10,"");
            SetDlgItemText(hExpertBox,IDC_TAB2E_Val11,"");
        }
        if (SelItem->MagicSuffix)
        {
            SetDlgItemText(hExpertBox,IDC_TAB2E_Val12,MakeDez(SelItem->MagicSuffix->N & 0xFF));
            SetDlgItemText(hExpertBox,IDC_TAB2E_Val13,MakeDez(SelItem->MagicSuffix->nMod));
            SetDlgItemText(hExpertBox,IDC_TAB2E_Val14,MakeDez(SelItem->modMagicSuffix));
            SetDlgItemText(hExpertBox,IDC_TAB2E_Val15,MakeDez(SelItem->modpickMagicSuffix));
        }
        else
        {
            SetDlgItemText(hExpertBox,IDC_TAB2E_Val12,"");
            SetDlgItemText(hExpertBox,IDC_TAB2E_Val13,"");
            SetDlgItemText(hExpertBox,IDC_TAB2E_Val14,"");
            SetDlgItemText(hExpertBox,IDC_TAB2E_Val15,"");
        }
*/
    }
    else
    {
        SetDlgItemText(hExpertBox,IDC_TAB2E_ItemRecordID,"");
        for(int z=0;z< sizeof Controls / sizeof Controls[0];z++)
        {
            EnableWindow(Controls[z].hWindow,FALSE);
            SetWindowText(Controls[z].hWindow,"");
        }
    }

    if (hRandomBox)
        UpdateTab2Rnd();
    if (hItemListBox)
        UpdateTab2ItemList();
}

LRESULT CALLBACK Tab2EDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
    switch(uMsg)
    {
    case WM_INITDIALOG:
        {
            {
                RECT DialogPos;

                GetWindowRect(hMainDialog,&DialogPos);
                DialogPos.left = DialogPos.right;

                SetWindowPos(hWnd,HWND_TOP,
                    DialogPos.left,DialogPos.top,
                    0,0,SWP_NOSIZE | SWP_NOACTIVATE);
            }

            for(int z=0;z< sizeof Controls / sizeof Controls[0];z++)
            {
                Controls[z].hWindow = GetDlgItem(hWnd,Controls[z].DialogID);
            }

            SetTimer(hWnd,IDT_TIMER,EXPERTBOXPOLLER,NULL);

            hExpertBox = hWnd;
            UpdateTab2E();

            ShowWindow(hWnd,SW_SHOWNOACTIVATE);
        }
        return true;
    case WM_TIMER:
        switch(wParam)
        {
        case IDT_TIMER:
            {
```

```
                if (SelItem && SelItem->Busy) {
                    UpdateTab2E();
                    InvalidateRect(hExpertBox,NULL,FALSE);
                }
            }
            break;
        }
        return false;
    case WM_COMMAND:
        {
            if (HIWORD(wParam) == EN_KILLFOCUS)
            {
                switch(LOWORD(wParam))
                {
                case IDC_TAB2E_Raw00:   case IDC_TAB2E_Raw10:
                case IDC_TAB2E_Raw01:   case IDC_TAB2E_Raw11:
                case IDC_TAB2E_Raw02:   case IDC_TAB2E_Raw12:
                case IDC_TAB2E_Raw03:   case IDC_TAB2E_Raw13:
                case IDC_TAB2E_Raw04:   case IDC_TAB2E_Raw14:
                case IDC_TAB2E_Raw05:   case IDC_TAB2E_Raw15:
                case IDC_TAB2E_Raw06:   case IDC_TAB2E_Raw16:
                case IDC_TAB2E_Raw07:   case IDC_TAB2E_Raw17:
                case IDC_TAB2E_Raw08:   case IDC_TAB2E_Raw18:
                case IDC_TAB2E_Raw09:   case IDC_TAB2E_Raw19:
                case IDC_TAB2E_Raw0A:   case IDC_TAB2E_Raw1A:
                case IDC_TAB2E_Raw0B:   case IDC_TAB2E_Raw1B:
                case IDC_TAB2E_Raw0C:   case IDC_TAB2E_Raw1C:
                case IDC_TAB2E_Raw0D:   case IDC_TAB2E_Raw1D:
                case IDC_TAB2E_Raw0E:   case IDC_TAB2E_Raw1E:
                case IDC_TAB2E_Raw0F:   case IDC_TAB2E_Raw1F:
                    {
                        if (SelItem)
                        {
                            BYTE *RawData = SelItem->GetItemRecord();
                            int RawDataLen = SelItem->ItemRecordLength();

                            for(int n=0;n<nRawData;n++)
                            {
                                if (n < RawDataLen) {
                                    ConvHex(Controls[n].DialogID,&RawData[n]);
                                }
                            }

                            SelItem->Info = 0;
                            UpdateTab2();
                            InvalidateRect(hTabDialog,NULL,FALSE);
                        }
                    }
                    break;
                case IDC_TAB2E_ItemCode:
                case IDC_TAB2E_UniqueCode:
                case IDC_TAB2E_DWA:
                case IDC_TAB2E_DWB:
                case IDC_TAB2E_MagicLevel:
                case IDC_TAB2E_GemNum:
                    {
                        if (SelItem)
                        {
                            int z;

                            if (ConvHexProp(IDC_TAB2E_ItemCode,&z)) SelItem->SetItemCode(z);
                            if (ConvHexProp(IDC_TAB2E_UniqueCode,&z)) SelItem->SetUniqueCode(z);

                            if (ConvHexProp(IDC_TAB2E_DWA,&z)) SelItem->SetDWA(z);
                            if (ConvHexProp(IDC_TAB2E_DWB,&z)) SelItem->SetDWB(z);
                            if (ConvHexProp(IDC_TAB2E_MagicLevel,&z)) SelItem->SetMagicLevel(z);

                            if (ConvDezProp(IDC_TAB2E_GemNum,&z)) SelItem->SetGemNum(z);

//                          if (SelItem->MagicPrefix)
                            {
//                              if (ConvDezProp(IDC_TAB2E_Val8,&z)) MagicPrefixTable[SelItem->iMagicPrefix].ModLevel = z
;
                                //if (ConvDezProp(IDC_TAB2E_Val9,&z)) SelItem->MagicPrefix->nMod = z;
                            }
//                          if (SelItem->MagicSuffix)
                            {
//                              if (ConvDezProp(IDC_TAB2E_Val12,&z)) MagicSuffixTable[SelItem->iMagicSuffix].ModLevel =
z;
```

```
                            //if (ConvDezProp(IDC_TAB2E_Val13,&z)) SelItem->MagicSuffix->nMod = z;
                        }

                        SelItem->Info = 0;
                        SelItem->Decoded = false;
                        UpdateTab2();
                        InvalidateRect(hTabDialog,NULL,FALSE);
                    }
                }
                break;
            }
        }
        else if (LOWORD(wParam) == IDC_TAB2E_RandA || LOWORD(wParam) == IDC_TAB2E_RandB)
        {
            if (HIWORD(wParam) == BN_CLICKED)
            {
                if (IsDlgButtonChecked(hExpertBox,LOWORD(wParam)))
                {
                    if (hRandomBox) {
                        DestroyWindow(hRandomBox);
                    }

                    CheckDlgButton(hWnd,LOWORD(wParam),BST_UNCHECKED);
                }
                else
                {
                    if (LOWORD(wParam) == IDC_TAB2E_RandA)
                    {
                        CheckDlgButton(hWnd,IDC_TAB2E_RandA,BST_CHECKED);
                        CheckDlgButton(hWnd,IDC_TAB2E_RandB,BST_UNCHECKED);
                        RandomBoxDW = 0;
                    }
                    else if (LOWORD(wParam) == IDC_TAB2E_RandB)
                    {
                        CheckDlgButton(hWnd,IDC_TAB2E_RandA,BST_UNCHECKED);
                        CheckDlgButton(hWnd,IDC_TAB2E_RandB,BST_CHECKED);
                        RandomBoxDW = 1;
                    }

                    if (!hRandomBox) {
                        hRandomBox = CreateDialog(hInstance,MAKEINTRESOURCE(IDD_TAB2Rnd),hTabDialog,(DLGPROC)&Tab2Rn
dDialogProc);
                    }
                    UpdateTab2();
                }
            }
        }
        else if (LOWORD(wParam) == IDC_TAB2E_FindInfo && HIWORD(wParam) == BN_CLICKED)
        {
            if (!SelItem) return false;
            SelItem->Info = 0;
            SelItem->FindInfo();
            UpdateTab2();
            InvalidateRect(hTabDialog,NULL,FALSE);
        }
        else if (LOWORD(wParam) == IDC_TAB2E_Decode && HIWORD(wParam) == BN_CLICKED)
        {
            if (!SelItem) return false;
            SelItem->Decoded = 0;
            SelItem->Decode();
            UpdateTab2();
            InvalidateRect(hTabDialog,NULL,FALSE);
        }
        else if (LOWORD(wParam) == IDC_TAB2E_ItemList && HIWORD(wParam) == BN_CLICKED)
        {
            if (hItemListBox) {
                DestroyWindow(hItemListBox);
            }
            else {
                hItemListBox = CreateDialog(hInstance,MAKEINTRESOURCE(IDD_TAB2ItemList),hTabDialog,(DLGPROC)&Tab2Ite
mListDialogProc);
            }
        }
    }
    return false;
case WM_CLOSE:
    {
        if (hExpertBox)
            DestroyWindow(hExpertBox);
```

```
            if (hItemListBox)
                DestroyWindow(hItemListBox);
        }
        return false;
    case WM_DESTROY:
        {
            if (hRandomBox)
                DestroyWindow(hRandomBox);

            CheckDlgButton(hTabDialog,IDC_TAB2_ExpertMode,BST_UNCHECKED);

            hExpertBox = 0;
        }
        return false;
    }
    return false;
}
```

```
// Tab2Gems.cpp from D2E

#include "JamellaD2E.h"

struct
{
    int     Frame,Bitmap,ListBox,InfoEdit;

    bool    Active;
    int     ItemCode;
}
GemID[] =
{
    { IDC_TAB2Gems_Frame1, IDC_TAB2Gems_Bmp1, IDC_TAB2Gems_Sel1, IDC_TAB2Gems_Info1 },
    { IDC_TAB2Gems_Frame2, IDC_TAB2Gems_Bmp2, IDC_TAB2Gems_Sel2, IDC_TAB2Gems_Info2 },
    { IDC_TAB2Gems_Frame3, IDC_TAB2Gems_Bmp3, IDC_TAB2Gems_Sel3, IDC_TAB2Gems_Info3 },
    { IDC_TAB2Gems_Frame4, IDC_TAB2Gems_Bmp4, IDC_TAB2Gems_Sel4, IDC_TAB2Gems_Info4 },
    { IDC_TAB2Gems_Frame5, IDC_TAB2Gems_Bmp5, IDC_TAB2Gems_Sel5, IDC_TAB2Gems_Info5 },
    { IDC_TAB2Gems_Frame6, IDC_TAB2Gems_Bmp6, IDC_TAB2Gems_Sel6, IDC_TAB2Gems_Info6 },
    { IDC_TAB2Gems_Frame7, IDC_TAB2Gems_Bmp7, IDC_TAB2Gems_Sel7, IDC_TAB2Gems_Info7 }
};

int FindnGemInfo(DWORD ICode)
{
    for(int n=0;n<nGemInfos;n++) {
        if (GemInfos[n].ItemCode == ICode || GemInfos[n].IC == ICode)
            return n;
    }
    return -1;
}
ItemInfo *FindItemInfo(DWORD ICode)
{
    for(int n=0;n<nItemInfos;n++) {
        if (ItemInfos[n].ItemCode == ICode || ItemInfos[n].IC == ICode)
            return &ItemInfos[n];
    }
    return 0;
}
inline int FindListBoxID(int DlgID)
{
    for(int z=0;z<sizeof GemID / sizeof GemID[0];z++) {
        if (GemID[z].ListBox == DlgID)
            return z;
    }
    return -1;
}

char MergeGemTextsBuffer[256];
const char *MergeGemTexts(const GemInfo* Info)
{
    char *s = MergeGemTextsBuffer;
    *s = 0;

    switch(SelItem->Info->GemClass)
    {
    case 'W':
        {
            for(int z=0;z<3;z++) {
                if (!Info->WeaponMod[z].Code) continue;

                const char *e = GetEffect(Info->WeaponMod[z].Code);
                sprintf(buffer,e,Info->WeaponMod[z].Min);

                strcat(s,buffer);
                strcat(s,"\r\n");
            }
            break;
        }
    case 'H':
        {
            for(int z=0;z<3;z++) {
                if (!Info->HelmMod[z].Code) continue;

                const char *e = GetEffect(Info->HelmMod[z].Code);
                sprintf(buffer,e,Info->HelmMod[z].Min);

                strcat(s,buffer);
                strcat(s,"\r\n");
            }
```

```
            break;
        }
    case 'S':
        {
            for(int z=0;z<3;z++) {
                if (!Info->ShieldMod[z].Code) continue;

                const char *e = GetEffect(Info->ShieldMod[z].Code);
                sprintf(buffer,e,Info->ShieldMod[z].Min);

                strcat(s,buffer);
                strcat(s,"\r\n");
            }
            break;
        }
    }

    return s;
}

void UpdateTab2Gems(HWND hWnd)
{
    for(int z=0;z<sizeof GemID / sizeof GemID[0];z++)
    {
        if (GemID[z].Active)
        {
            EnableWindow(GetDlgItem(hWnd,GemID[z].Frame),TRUE);
            EnableWindow(GetDlgItem(hWnd,GemID[z].Bitmap),TRUE);
            EnableWindow(GetDlgItem(hWnd,GemID[z].ListBox),TRUE);
            EnableWindow(GetDlgItem(hWnd,GemID[z].InfoEdit),TRUE);

            if (GemID[z].ItemCode)
            {
                int nGI = FindnGemInfo(GemID[z].ItemCode);
                if (nGI < 0) {
                    SendDlgItemMessage(hWnd,GemID[z].Bitmap,STM_SETIMAGE,IMAGE_BITMAP,(LPARAM) 0);
                    SendDlgItemMessage(hWnd,GemID[z].ListBox,CB_SETCURSEL,0,0);
                    SetDlgItemText(hWnd,GemID[z].InfoEdit,"???");
                }
                else {
                    const GemInfo *GI = &GemInfos[nGI];
                    ItemInfo *II = FindItemInfo(GemID[z].ItemCode);

                    SendDlgItemMessage(hWnd,GemID[z].Bitmap,STM_SETIMAGE,IMAGE_BITMAP,(LPARAM) ItemInfoGetBitmap(II));
                    SendDlgItemMessage(hWnd,GemID[z].ListBox,CB_SETCURSEL,nGI,0);
                    SetDlgItemText(hWnd,GemID[z].InfoEdit,MergeGemTexts(GI));
                }
            }
            else
            {
                SendDlgItemMessage(hWnd,GemID[z].Bitmap,STM_SETIMAGE,IMAGE_BITMAP,(LPARAM) 0);
                SendDlgItemMessage(hWnd,GemID[z].ListBox,CB_SETCURSEL,0,0);
                SetDlgItemText(hWnd,GemID[z].InfoEdit,"");
            }
        }
        else
        {
            EnableWindow(GetDlgItem(hWnd,GemID[z].Frame),FALSE);
            SendDlgItemMessage(hWnd,GemID[z].Bitmap,STM_SETIMAGE,IMAGE_BITMAP,0);
            EnableWindow(GetDlgItem(hWnd,GemID[z].Bitmap),FALSE);
            EnableWindow(GetDlgItem(hWnd,GemID[z].ListBox),FALSE);
            EnableWindow(GetDlgItem(hWnd,GemID[z].InfoEdit),FALSE);

            SetDlgItemText(hWnd,GemID[z].InfoEdit,"");
        }
    }
}

void LoadGems()
{
    Item *G = SelItem->Gems;

    for(int z=0;z<sizeof GemID / sizeof GemID[0];z++)
    {
        GemID[z].Active = true;

        if (SelItem->Info->Sockets > z || RegOptions.A7Gems)
        {
            if (G) {
```

```
                GemID[z].ItemCode = G->ItemCode();
            }
            else {
                GemID[z].ItemCode = 0x0000;
            }

            if (G) G = G->Next();
        }
        else {
            GemID[z].Active = false;
        }
    }
}
void SaveGems()
{
    if (SelItem->Gems) {
        delete SelItem->Gems;
        SelItem->Gems = 0;
    }

    for(int z=0;z<sizeof GemID / sizeof GemID[0];z++)
    {
        if (!GemID[z].Active) continue;
        if (!GemID[z].ItemCode) continue;

        ItemInfo *II = FindItemInfo(GemID[z].ItemCode);

        Item *G = CreateItem(&SelItem->Gems,II);

        G->SetCoordinates(CNT_SOCKET,z,0);
        G->SetQuality(USUALITEM+1);
    }

    SelItem->SetGemNum(SelItem->Gems->Count());
}

LRESULT CALLBACK Tab2GemsDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
    switch(uMsg)
    {
    case WM_INITDIALOG:
        {
            LoadGems();

            for(int z=0;z<sizeof GemID / sizeof GemID[0];z++)
                SendDlgItemMessage(hWnd,GemID[z].ListBox,CB_RESETCONTENT,0,0);

            for(int n=0;n<nGemInfos;n++)
            {
                for(int z=0;z<sizeof GemID / sizeof GemID[0];z++)
                {
                    SendDlgItemMessage(hWnd,GemID[z].ListBox,CB_ADDSTRING,0,(LPARAM)GemInfos[n].Name);
                    SendDlgItemMessage(hWnd,GemID[z].ListBox,CB_SETITEMDATA,n,(LPARAM) GemInfos[n].ItemCode);
                }
            }

            for(z=0;z<sizeof GemID / sizeof GemID[0];z++)
                SendDlgItemMessage(hWnd,GemID[z].ListBox,CB_SETCURSEL,0,0);

            UpdateTab2Gems(hWnd);
        }
        return true;
    case WM_COMMAND:
        {
            switch(LOWORD(wParam))
            {
            case IDOK:
                SaveGems();
                EndDialog(hWnd,IDOK);
                break;
            case IDCANCEL:
                EndDialog(hWnd,IDCANCEL);
                break;
            case IDC_TAB2Gems_Sel1:
            case IDC_TAB2Gems_Sel2:
            case IDC_TAB2Gems_Sel3:
            case IDC_TAB2Gems_Sel4:
            case IDC_TAB2Gems_Sel5:
            case IDC_TAB2Gems_Sel6:
```

```
            case IDC_TAB2Gems_Sel7:
                {
                    if (HIWORD(wParam) == CBN_SELCHANGE)
                    {
                        int ID = FindListBoxID(LOWORD(wParam));
                        if (!GemID[ID].Active) break;

                        HWND hCtl = (HWND) lParam;

                        int i = SendMessage(hCtl,CB_GETCURSEL,0,0);
                        GemID[ID].ItemCode = SendMessage(hCtl,CB_GETITEMDATA,i,0);

                        UpdateTab2Gems(hWnd);
                    }
                }
                break;
            }
        }
        return false;
    case WM_CLOSE:
        EndDialog(hWnd,IDCANCEL);
        return false;
    case WM_DESTROY:
        {
        }
        return false;
    }
    return false;
}
```

```
// TAB2ItemFileLoadSave.cpp from D2E

#include "JamellaD2E.h"

static const char *ReadItemErrorString = 0;

Item *MakeItemFromData(BYTE* data, int size)
{
    struct
    {
        char    JM[2];
        WORD    unimportant;
        WORD    type;
    } ItemHead;

    memcpy(&ItemHead,data,sizeof ItemHead);
    BYTE *d = data;

    Item *I = 0;

    if (ItemHead.JM[0] != 'J' || ItemHead.JM[1] != 'M')
    {
        ReadItemErrorString = "Invalid Item File!\r\nJM Header missing.";
        return 0;
    }

    if ((ItemHead.type & 0x0039) == 0x0000) // 1.03 Item Data Type
    {
        if ((size % 27) == 0) {
            I = new Item103;
            I->LoadItemRecord(d);
            size -= I->ItemRecordLength();
            d += I->ItemRecordLength();

            while(size >= 27)
            {
                Item *G = new Item103;
                if (I->Gems)
                    I->Gems->Attach(G);
                else
                    I->Gems = G;
                G->LoadItemRecord(d);
                size -= G->ItemRecordLength();
                d += G->ItemRecordLength();
            }
        }
        else if (size == 39) { // 39 = 0x27 != 27 Doe you Idiot!
            I = new Item103;
            I->LoadItemRecord(d);
        }
        else if (size == 36) { // 36 another editor made these
            I = new Item103;
            I->LoadItemRecord(d);
        }
        else {
            // this is crazy. I'll load any item length now, because
            // of a variable length item record from doe's 3.41
            I = new Item103;
            if (!I->LoadItemRecord(d)) {
                ReadItemErrorString = "Invalid Item File!\r\nItem structure indicates a 1.03 item, but JM signature is m
issing.\nFile size checks are ignored here.";
                delete I;
                return 0;
            }
        }
    }
    else if ((ItemHead.type & 0x0039) == 0x0001) // 1.03 Ear Item Data Type
    {
        if (size == 27) {
            I = new Item103Ear;
            I->LoadItemRecord(d);
        }
        else {
            ReadItemErrorString = "Invalid Item File!\r\nItem structure indicates a 1.03 ear item, but size is not 27.";
        }
    }
    else if ((ItemHead.type & 0x0039) == 0x0018) // 1.04 Extended Struct
    {
        if (size == 31 || ((size-31) % 15) == 0) {
```

```
            I = new Item104Ex;
            I->LoadItemRecord(d);
            size -= I->ItemRecordLength();
            d += I->ItemRecordLength();

            while(size >= 15)
            {
                Item *G = new Item104Sm;
                if (I->Gems)
                    I->Gems->Attach(G);
                else
                    I->Gems = G;
                G->LoadItemRecord(d);
                size -= G->ItemRecordLength();
                d += G->ItemRecordLength();
            }
        }
        else {
            ReadItemErrorString = "Invalid Item File!\r\nItem structure indicates a 1.04 extended item, but size is not
31.";
        }
    }
    else if ((ItemHead.type & 0x0039) == 0x0038) // 1.04 Simple Struct
    {

        if (size == 15) {
            I = new Item104Sm;
            I->LoadItemRecord(d);
        }
        else {
            ReadItemErrorString = "Invalid Item File!\r\nItem structure indicates a 1.04 simple item, but size is not 15
.";
        }
    }
    else if ((ItemHead.type & 0x0039) == 0x0039) // 1.04 Ear Struct
    {
        if (size == 26) {
            I = new Item104Ear;
            I->LoadItemRecord(d);
        }
        else {
            ReadItemErrorString = "Invalid Item File!\r\nItem structure indicates a 1.04 ear item, but size is not 26.";
        }
    }
    else {
        ReadItemErrorString = "Invalid Item File!\r\nUnknown item structure identifier.";
    }
    return I;
}

Item *MakeItemFromFile(HANDLE hFile)
{
    DWORD fsize = GetFileSize(hFile,NULL);

    if (fsize < 6)
    {
        ReadItemErrorString = "Invalid Item File!\r\nFile size smaller than 6.";
        return 0;
    }
    if (fsize > 512)
    {
        ReadItemErrorString = "Invalid Item File!\r\nFile much too large (> 512 bytes).";
        return 0;
    }

    unsigned long read;
    BYTE    buff[512];

    ReadFile(hFile,buff,fsize,&read,0);

    if (read != fsize)
    {
        ReadItemErrorString = "Could not read from file!";
        return 0;
    }

    return MakeItemFromData(buff,read);
}
```

```
#if defined(JAMELLAEDITOR)

// LoadItemFile OPENFILENAME Explorer Hook
static UINT_PTR CALLBACK OFNHookProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
    switch(uMsg)
    {
    case WM_INITDIALOG:
        SendDlgItemMessage(hWnd,IDC_IBWSR_RichText,EM_SETBKGNDCOLOR,FALSE,GetSysColor(COLOR_BTNFACE));
        return true;

    case WM_NOTIFY:
        switch(((NMHDR*)lParam)->code)
        {
        case CDN_SELCHANGE:
            {
                char QueryFilename[128];
                int x = CommDlg_OpenSave_GetFilePath(((NMHDR*)lParam)->hwndFrom,QueryFilename,sizeof QueryFilename);

                // Open File
                HANDLE hFile = CreateFile(QueryFilename,
                    GENERIC_READ,FILE_SHARE_WRITE,NULL,
                    OPEN_EXISTING,FILE_ATTRIBUTE_NORMAL,NULL);

                if (hFile == INVALID_HANDLE_VALUE) return true;

                ReadItemErrorString = 0;
                Item *I = MakeItemFromFile(hFile);

                if (I)
                {
                    I->FindInfo();
                    I->Decode();

                    RTFStreamSend(hWnd,IDC_IBWSR_RichText,I->RichText());

                    SendDlgItemMessage(hWnd,IDC_IBWSR_Bitmap,STM_SETIMAGE,IMAGE_BITMAP,(LPARAM)I->GetBitmap());

                    delete I;
                }
                else
                {
                    char RTF[2048];
                    char RTFBuffer[260];

                    ASCIItoRTF(RTFBuffer,ReadItemErrorString);
                    sprintf(RTF,"{\\rtf1\\ansi\\ansicpg1252\\deff0\\deflang1031 {\\fonttbl{\\f0\\fswiss\\fcharset0 MS Sa
ns Serif;}} \\uc1\\pard\\qc\\b\\f0\\fs16 %s\\par }",RTFBuffer);
                    RTFStreamSend(hWnd,IDC_IBWSR_RichText,RTF);

                    SendDlgItemMessage(hWnd,IDC_IBWSR_Bitmap,STM_SETIMAGE,IMAGE_BITMAP,(LPARAM)0);
                }

                CloseHandle(hFile);
            }
            return true;
        }
        return false;

    case WM_DESTROY:
        return true;
    }
    return false;
}

bool LoadItemFile(HWND hWnd)
{
    if (CopyBuffer && CopyBuffer->Busy) return false;

    // common dialog box structure
    OPENFILENAME ofn;
    char QueryFilename[260];

    {   // Initialize OPENFILENAME
        ZeroMemory(&ofn,sizeof(OPENFILENAME));
        ofn.lStructSize = sizeof(OPENFILENAME);
        ofn.hwndOwner = hWnd;
        ofn.lpstrFilter = "D2 Saved Item (*.d2i;*.item;*.ite;*.itm)\0*.d2i;*.item;*.ite;*.itm\0Other Files (*.*)\0*.*\0"
;
```

```
        ofn.nFilterIndex = 0;
        ofn.lpstrFile = QueryFilename;
        ofn.nMaxFile = sizeof(QueryFilename);
        ofn.lpstrFileTitle = NULL;
        ofn.lpstrDefExt = "d2i";
        ofn.nMaxFileTitle = 0;
        ofn.lpstrInitialDir = RegOptions.ItemPath;
        ofn.hInstance = hInstance;
        ofn.lpTemplateName = MAKEINTRESOURCE(IDD_IBWSR);
        ofn.lpfnHook = &OFNHookProc;

        ofn.Flags = OFN_PATHMUSTEXIST | OFN_FILEMUSTEXIST | OFN_NOREADONLYRETURN | OFN_HIDEREADONLY | OFN_ENABLETEMPLATE
 | OFN_EXPLORER | OFN_ENABLEHOOK;

        ZeroMemory(&QueryFilename,sizeof(QueryFilename));
    }

    if (GetOpenFileName(&ofn))
    {
        // Open File
        HANDLE hFile = CreateFile(QueryFilename,
            GENERIC_READ,FILE_SHARE_WRITE,NULL,
            OPEN_EXISTING,FILE_ATTRIBUTE_NORMAL,NULL);

        if (hFile == INVALID_HANDLE_VALUE)
        {
            MessageBox(hWnd,"Could not open file!",PROGRAMNAME,
                MB_OK | MB_ICONSTOP | MB_APPLMODAL);
            return false;
        }

        ReadItemErrorString = 0;
        Item *N = MakeItemFromFile(hFile);

        if (N) {

            if (CopyBuffer)
                delete CopyBuffer;

            CopyBuffer = N;

            strcpy(RegOptions.ItemPath,QueryFilename);
            char *sl = RegOptions.ItemPath;
            while(strstr(sl,"\\") != 0)
                sl = strstr(sl,"\\")+1;
            *sl = 0;

            InvalidateRect(hTabDialog,NULL,FALSE);
            UpdateWindow(hTabDialog);
        }
        else
        {
            ErrorBox(ReadItemErrorString);
        }

        CloseHandle(hFile);
    }
    return true;
}

bool SaveItemFile(HWND hWnd)
{
    if (!CopyBuffer)
    {
        ErrorBox("You must put an item into the Copy Buffer!",hWnd);
        return false;
    }
    if (CopyBuffer->Busy) return false;

    // common dialog box structure
    char QueryFilename[260];
    OPENFILENAME ofn;

    {   // Initialize OPENFILENAME
        ZeroMemory(&ofn,sizeof(OPENFILENAME));
        ofn.lStructSize = sizeof(OPENFILENAME);

        ofn.hwndOwner = hWnd;
        ofn.lpstrFilter = "D2 Saved Item (*.d2i)\0*.d2i\0";
```

```
        ofn.nFilterIndex = 0;
        ofn.lpstrFile = QueryFilename;
        ofn.nMaxFile = sizeof(QueryFilename);
        ofn.lpstrInitialDir = RegOptions.ItemPath;
        ofn.lpstrFileTitle = NULL;
        ofn.lpstrDefExt = "d2i";
        ofn.nMaxFileTitle = 0;

        ofn.Flags = OFN_PATHMUSTEXIST | OFN_OVERWRITEPROMPT | OFN_NOREADONLYRETURN | OFN_HIDEREADONLY;

        ZeroMemory(&QueryFilename,sizeof(QueryFilename));
        strcpy(QueryFilename,CopyBuffer->Name());
    }

    // Display the Open Dialog Box
    if (GetSaveFileName(&ofn))
    {
        // Open File
        HANDLE hFile = CreateFile(QueryFilename,
            GENERIC_WRITE,FILE_SHARE_WRITE,NULL,
            CREATE_ALWAYS,FILE_ATTRIBUTE_NORMAL,NULL);

        if (hFile == INVALID_HANDLE_VALUE)
        {
            MessageBox(hWnd,"Could not open file!",PROGRAMNAME,
                MB_OK | MB_ICONSTOP | MB_APPLMODAL);
            return true;
        }

        unsigned long written;
        WriteFile(hFile,CopyBuffer->GetItemRecord(),CopyBuffer->ItemRecordLength(),&written,0);

        if (written != (unsigned long)CopyBuffer->ItemRecordLength())
        {
            MessageBox(hWnd,"Could not write to file!",PROGRAMNAME,
                MB_OK | MB_ICONSTOP | MB_APPLMODAL);
        }

        if (CopyBuffer->Socketed())
        {
            for(Item *G = CopyBuffer->Gems;G != 0;G = G->Next())
            {
                unsigned long written;
                WriteFile(hFile,G->GetItemRecord(),G->ItemRecordLength(),&written,0);

                if (written != (unsigned long)G->ItemRecordLength())
                {
                    MessageBox(hWnd,"Could not write to file!",PROGRAMNAME,
                        MB_OK | MB_ICONSTOP | MB_APPLMODAL);
                }
            }
        }

        CloseHandle(hFile);

        strcpy(RegOptions.ItemPath,QueryFilename);
        char *sl = RegOptions.ItemPath;
        while(strstr(sl,"\\") != 0)
            sl = strstr(sl,"\\")+1;
        *sl = 0;
    }
    return true;
}

#endif
```

```
// Tab2M.cpp from D2E

#include "JamellaD2E.h"

/*
static DWORD WINAPI BruteForceAttack(LPVOID ThreadParameters)
{
    MagicSearchThread *P = (MagicSearchThread *)ThreadParameters;
    P->Advanced = true;

    P->Dialog = CreateDialogParam(hInstance,MAKEINTRESOURCE(IDD_TAB2S),hMainDialog,(DLGPROC) Tab2SearchDialogProc,(LPARA
M) P);
    if (!P->Dialog) return -1;

    Item* I = P->Item;
    I->Busy = true;

    int MagicLevel = 0;
    if (P->PrefixMatch && P->Prefix)
        MagicLevel = max(MagicLevel,P->Prefix->ModLevel);
    if (P->SuffixMatch && P->Suffix)
        MagicLevel = max(MagicLevel,P->Suffix->ModLevel);

    I->SetMagicLevel(MagicLevel - 2);

    while(!CheckPollMessages())
    {
        I->SetDWB(I->DWB() + 1);
        I->MD->QuickDecode();
        P->Counter++;

        if (!I->Decoded) continue;

        if (P->PrefixMatch)
        {
            if (P->Prefix != I->MD->MagicPrefix) continue;

            if (P->ForcePrefixMatch[0])
                if (P->ForcePrefixValue[0] != I->MD->MagicPrefixMag[0]) continue;
            if (P->ForcePrefixMatch[1])
                if (P->ForcePrefixValue[1] != I->MD->MagicPrefixMag[1]) continue;
            if (P->ForcePrefixMatch[2])
                if (P->ForcePrefixValue[2] != I->MD->MagicPrefixMag[2]) continue;
            if (P->ForcePrefixMatch[3])
                if (P->ForcePrefixValue[3] != I->MD->MagicPrefixMag[3]) continue;
        }
        if (P->SuffixMatch)
        {
            if (P->Suffix != I->MD->MagicSuffix) continue;

            if (P->ForceSuffixMatch[0])
                if (P->ForceSuffixValue[0] != I->MD->MagicSuffixMag[0]) continue;
        }

        break;
    }

    MessageBeep(MB_ICONASTERISK);
    I->Busy = false;
    EndDialog(P->Dialog,0);
    UpdateTab2();

    CloseHandle(P->Thread);
    GlobalFree(P->ThreadData);
    return 0;
}
*/

static DWORD WINAPI BruteForceAttack(LPVOID ThreadParameters)
{
    MagicSearchThread *P = (MagicSearchThread *)ThreadParameters;
    P->Advanced = true;

    P->Dialog = CreateDialogParam(hInstance,MAKEINTRESOURCE(IDD_TAB2S),hMainDialog,(DLGPROC) Tab2SearchDialogProc, (LPAR
AM) P);
    if (!P->Dialog) return -1;

    Item* I = P->Item;
    I->Busy = true;
```

```
int MagicLevel = 0;
if (P->PrefixMatch && P->Prefix)
    MagicLevel = max(MagicLevel,P->Prefix->ModLevel);
if (P->SuffixMatch && P->Suffix)
    MagicLevel = max(MagicLevel,P->Suffix->ModLevel);

I->SetMagicLevel(MagicLevel - 2);

bool QuitMessage = false;

DWORD SDWB = I->DWB();

for(int MG = MagicLevel;MG == MagicLevel || (P->TraverseMagicLevels && MG <= MAXMODLEVEL);MG++)
{
    I->SetMagicLevel(MG - 2);

    DWORD RDWB = SDWB;
    P->Counter = 0;

    while(! (QuitMessage = CheckPollMessages()) )
    {
        if (!P->Running)
            continue;

        I->SetDWB(RDWB++);

        P->Counter++;
        if (P->Counter == 0)
            break;

        if (!I->MD->QuickDecode()) continue;

        if (P->PrefixMatch)
        {
            if (P->Prefix != I->MD->MagicPrefix) continue;

            if (P->ForcePrefixMatch[0])
                if (P->ForcePrefixValue[0] != I->MD->MagicPrefixMag[0]) continue;
                if (P->ForcePrefixMatch[1])
                    if (P->ForcePrefixValue[1] != I->MD->MagicPrefixMag[1]) continue;
                    if (P->ForcePrefixMatch[2])
                        if (P->ForcePrefixValue[2] != I->MD->MagicPrefixMag[2]) continue;
                        if (P->ForcePrefixMatch[3])
                            if (P->ForcePrefixValue[3] != I->MD->MagicPrefixMag[3]) continue;
        }
        if (P->SuffixMatch)
        {
            if (P->Suffix != I->MD->MagicSuffix) continue;

            if (P->ForceSuffixMatch[0])
                if (P->ForceSuffixValue[0] != I->MD->MagicSuffixMag[0]) continue;
        }

        if (P->HitsSelection) {

            SearchHit *H = new SearchHit;
            H->MagicLevel = I->MagicLevel();
            H->DWA = I->DWA();
            H->DWB = I->DWB();
            H->List = 0;

            if (P->Hits) {

                SearchHit *S = P->Hits;
                while(S->List)
                    S = S->List;
                S->List = H;
            }
            else
                P->Hits = H;

            PostMessage(P->Dialog,WM_USER,0,0);
            MessageBeep(MB_ICONASTERISK);
        }
        else
            break;
    }
```

```
        if (QuitMessage) break;
    }

    I->Busy = false;
    I->Decoded = false;

    EndDialog(P->Dialog,IDOK);

    if (MG >= MAXMODLEVEL || P->TraverseMagicLevels && !QuitMessage)
        MessageBox(NULL,"Searched all combinations in the scope of this item. Your desired combination is not available
on the item. Select similar attributes and try again.",PROGRAMNAME,MB_OK | MB_ICONASTERISK);
    else
        MessageBeep(MB_ICONASTERISK);

    UpdateTab2();
    InvalidateRect(hTabDialog,NULL,FALSE);

    CloseHandle(P->Thread);
    for(SearchHit *H = P->Hits;H != 0;) {
        SearchHit *T = H->List;
        delete H;
        H = T;
    }

    GlobalFree(P->ThreadData);
    return 0;
}

static bool RestrictELevel;
static int  RestrictELevelValue;

inline void LoadPreSuffixTree(HWND hWnd)
{
    HWND hTV = GetDlgItem(hWnd,IDC_TAB2Magic_PrefixTree);

    TreeView_DeleteAllItems(hTV);

    TV_ITEM TVItem;
    TV_INSERTSTRUCT TVInsert;
    HTREEITEM hBranch;

    // None Entry
    {
        ZeroMemory(&TVItem,sizeof TVItem);
        TVItem.mask = TVIF_TEXT | TVIF_PARAM;
        TVItem.pszText = "None";
        TVItem.lParam = (LPARAM)-1;

        TVInsert.hParent = TVI_ROOT;
        TVInsert.hInsertAfter = TVI_LAST;
        TVInsert.item = TVItem;

        TreeView_InsertItem(hTV,&TVInsert);
    }

    for(int z=0;z<nMagicPreSuffixTree;z++)
    {
        MagicPreSuffixTree[z].hTree = 0;

        if (MagicPreSuffixTree[z].Depth > 1) continue;

        ZeroMemory(&TVItem,sizeof TVItem);
        TVItem.mask = TVIF_TEXT | TVIF_PARAM;
        TVItem.pszText = MagicPreSuffixTree[z].Text;
        TVItem.lParam = (LPARAM)-1;

        TVInsert.hParent = TVI_ROOT;
        TVInsert.hInsertAfter = TVI_LAST;
        TVInsert.item = TVItem;

        for(z++;z<nMagicPreSuffixTree;z++)
        {
            if (MagicPreSuffixTree[z].Depth < 2)
            {
                z--;
                break;
            }

            // Load only Prefixes
```

```
            if (MagicPreSuffixTree[z].ModID & 256) continue;

            MagicPreSuffixTree[z].hTree = 0;

            // Check if attribute is applyable
            if ((MagicPrefixTable[MagicPreSuffixTree[z].ModID].MagicMask & SelItem->Info->MagicMask) == 0) continue;

            // Check if ELevel exceeded
            if (RestrictELevel && RestrictELevelValue < MagicPrefixTable[MagicPreSuffixTree[z].ModID].ELevel) continue;

            if (TVInsert.hParent == TVI_ROOT)
            {
                hBranch = MagicPreSuffixTree[z].hTree =
                    TreeView_InsertItem(hTV,&TVInsert);
            }

            ZeroMemory(&TVItem,sizeof TVItem);
            TVItem.mask = TVIF_TEXT | TVIF_PARAM;
            TVItem.pszText = MagicPreSuffixTree[z].Text;
            TVItem.lParam = MagicPreSuffixTree[z].ModID;

            TVInsert.hParent = hBranch;
            TVInsert.hInsertAfter = TVI_LAST;
            TVInsert.item = TVItem;

            MagicPreSuffixTree[z].hTree = TreeView_InsertItem(hTV,&TVInsert);
        }
    }
}

inline void LoadSuffixTree(HWND hWnd)
{
    HWND hTV = GetDlgItem(hWnd,IDC_TAB2Magic_SuffixTree);

    TreeView_DeleteAllItems(hTV);

    TV_ITEM TVItem;
    TV_INSERTSTRUCT TVInsert;
    HTREEITEM hBranch;

    // None Entry
    {
        ZeroMemory(&TVItem,sizeof TVItem);
        TVItem.mask = TVIF_TEXT | TVIF_PARAM;
        TVItem.pszText = "None";
        TVItem.lParam = (LPARAM)-1;

        TVInsert.hParent = TVI_ROOT;
        TVInsert.hInsertAfter = TVI_LAST;
        TVInsert.item = TVItem;

        TreeView_InsertItem(hTV,&TVInsert);
    }

    for(int z=0;z<nMagicPreSuffixTree;z++)
    {
        MagicPreSuffixTree[z].hTree = 0;

        if (MagicPreSuffixTree[z].Depth > 1) continue;

        ZeroMemory(&TVItem,sizeof TVItem);
        TVItem.mask = TVIF_TEXT | TVIF_PARAM;
        TVItem.pszText = MagicPreSuffixTree[z].Text;
        TVItem.lParam = (LPARAM)-1;

        TVInsert.hParent = TVI_ROOT;
        TVInsert.hInsertAfter = TVI_LAST;
        TVInsert.item = TVItem;

        for(z++;z<nMagicPreSuffixTree;z++)
        {
            if (MagicPreSuffixTree[z].Depth < 2)
            {
                z--;
                break;
            }

            // Load only Suffixes
            if (!(MagicPreSuffixTree[z].ModID & 256)) continue;
```

```
            MagicPreSuffixTree[z].hTree = 0;

            // Check if attributes is applyable
            if ((MagicSuffixTable[MagicPreSuffixTree[z].ModID & 0xFF].MagicMask & SelItem->Info->MagicMask) == 0)
                continue;

            // Check if ELevel exceeded
            if (RestrictELevel && RestrictELevelValue < MagicSuffixTable[MagicPreSuffixTree[z].ModID & 0xFF].ELevel) con
tinue;

            if (TVInsert.hParent == TVI_ROOT)
            {
                hBranch = MagicPreSuffixTree[z].hTree =
                    TreeView_InsertItem(hTV,&TVInsert);
            }

            ZeroMemory(&TVItem,sizeof TVItem);
            TVItem.mask = TVIF_TEXT | TVIF_PARAM;
            TVItem.pszText = MagicPreSuffixTree[z].Text;
            TVItem.lParam = MagicPreSuffixTree[z].ModID;

            TVInsert.hParent = hBranch;
            TVInsert.hInsertAfter = TVI_LAST;
            TVInsert.item = TVItem;

            MagicPreSuffixTree[z].hTree = TreeView_InsertItem(hTV,&TVInsert);
        }
    }
}

inline void EnablePrefixControls(HWND hWnd,int enable)
{
    EnableWindow(GetDlgItem(hWnd,IDC_TAB2Magic_PrefixTree),enable);
}
inline void EnableSuffixControls(HWND hWnd,int enable)
{
    EnableWindow(GetDlgItem(hWnd,IDC_TAB2Magic_SuffixTree),enable);
}

struct
{
    int MatchBox;
    int Slider;
    int Min;
    int Max;
    bool ForceMatch;
    int ForceValue;
}
PrefixControls[] =
{
{ IDC_TAB2Magic_Prefix1Match, IDC_TAB2Magic_Prefix1Value, IDC_TAB2Magic_Prefix1ValueMin, IDC_TAB2Magic_Prefix1ValueMax }
,
{ IDC_TAB2Magic_Prefix2Match, IDC_TAB2Magic_Prefix2Value, IDC_TAB2Magic_Prefix2ValueMin, IDC_TAB2Magic_Prefix2ValueMax }
,
{ IDC_TAB2Magic_Prefix3Match, IDC_TAB2Magic_Prefix3Value, IDC_TAB2Magic_Prefix3ValueMin, IDC_TAB2Magic_Prefix3ValueMax }
,
{ IDC_TAB2Magic_Prefix4Match, IDC_TAB2Magic_Prefix4Value, IDC_TAB2Magic_Prefix4ValueMin, IDC_TAB2Magic_Prefix4ValueMax }
,
};

struct
{
    int MatchBox;
    int Slider;
    int Min;
    int Max;
    bool ForceMatch;
    int ForceValue;
}
SuffixControls[] =
{
{ IDC_TAB2Magic_Suffix1Match, IDC_TAB2Magic_Suffix1Value, IDC_TAB2Magic_Suffix1ValueMin, IDC_TAB2Magic_Suffix1ValueMax }
,
};

void EnablePrefixControls(HWND hWnd,int Controls,int enable)
{
    if (Controls < 0)
```

```
    {
        for(int z=0;z<4;z++)
        {
            EnableWindow(GetDlgItem(hWnd,PrefixControls[z].Slider),enable);
            EnableWindow(GetDlgItem(hWnd,PrefixControls[z].Min),enable);
            EnableWindow(GetDlgItem(hWnd,PrefixControls[z].Max),enable);
        }
    }
    else
    {
        EnableWindow(GetDlgItem(hWnd,PrefixControls[Controls].Slider),enable);
        EnableWindow(GetDlgItem(hWnd,PrefixControls[Controls].Min),enable);
        EnableWindow(GetDlgItem(hWnd,PrefixControls[Controls].Max),enable);
    }

}
void EnableSuffixControls(HWND hWnd,int Controls,int enable)
{
    if (Controls < 0)
    {
        for(int z=0;z<4;z++)
        {
            EnableWindow(GetDlgItem(hWnd,SuffixControls[z].Slider),enable);
            EnableWindow(GetDlgItem(hWnd,SuffixControls[z].Min),enable);
            EnableWindow(GetDlgItem(hWnd,SuffixControls[z].Max),enable);
        }
    }
    else
    {
        EnableWindow(GetDlgItem(hWnd,SuffixControls[Controls].Slider),enable);
        EnableWindow(GetDlgItem(hWnd,SuffixControls[Controls].Min),enable);
        EnableWindow(GetDlgItem(hWnd,SuffixControls[Controls].Max),enable);
    }

}

static bool PrefixMatch = false;
static int  PrefixTree;
static _MagicPreSuffix *Prefix;

static bool SuffixMatch = false;
static int  SuffixTree;
static _MagicPreSuffix *Suffix;

static int  OrigModLevel;

static DWORD SearchAverage()
{
    int MagicLevel = 0;
    if (PrefixMatch && Prefix)
        MagicLevel = max(MagicLevel,Prefix->ModLevel);
    if (SuffixMatch && Suffix)
        MagicLevel = max(MagicLevel,Suffix->ModLevel);

    SelItem->SetMagicLevel(MagicLevel - 2);

    SelItem->MD->BuildMagicBuffers();

    DWORDLONG Average = 1;

    if (PrefixMatch)
    {
        Average *= 2 * SelItem->MD->nPrefixBuffer;

        if (Prefix)
        for(int z=0;z<4;z++)
        {
            if (!PrefixControls[z].ForceMatch) continue;
            if (Prefix->Mod[z].Code == 0) continue;

            Average *= Prefix->Mod[z].Max - Prefix->Mod[z].Min;
        }
    }

    if (SuffixMatch)
    {
        Average *= 2 * SelItem->MD->nSuffixBuffer;

        if (Suffix)
```

```
        for(int z=0;z<1;z++)
        {
            if (!SuffixControls[z].ForceMatch) continue;
            if (Suffix->Mod[z].Code == 0) continue;

            Average *= Suffix->Mod[z].Max - Suffix->Mod[z].Min;
        }
    }
    return DWORD(Average/2 +1);
}

static void UpdateTab2Magic(HWND hWnd)
{
    for(int n=0;n<4;n++)
    {
        char *nth = n == 0 ? "1st" :
                    n == 1 ? "2nd" :
                    n == 2 ? "3rd" : "4th";

        if (!PrefixMatch || !Prefix || !Prefix->Mod[n].Code)
        {
            EnableWindow(GetDlgItem(hWnd,PrefixControls[n].MatchBox),FALSE);
            CheckDlgButton(hWnd,PrefixControls[n].MatchBox,BST_UNCHECKED);

            EnablePrefixControls(hWnd,n,FALSE);

            sprintf(buffer,"No %s Prefix Effect",nth);
            SetDlgItemText(hWnd,PrefixControls[n].MatchBox,buffer);

            SendDlgItemMessage(hWnd,PrefixControls[n].Slider,TBM_SETPOS,TRUE,0);
            SetDlgItemText(hWnd,PrefixControls[n].Min,"");
            SetDlgItemText(hWnd,PrefixControls[n].Max,"");

            PrefixControls[n].ForceMatch = 0;
            PrefixControls[n].ForceValue = 0;
        }
        else if (Prefix->Mod[n].Max - Prefix->Mod[n].Min <= 0)
        {
            EnableWindow(GetDlgItem(hWnd,PrefixControls[n].MatchBox),FALSE);
            CheckDlgButton(hWnd,PrefixControls[n].MatchBox,BST_UNCHECKED);
            EnablePrefixControls(hWnd,n,FALSE);

            sprintf(buffer,"%s Prefix Effect fixed to %i",nth,Prefix->Mod[n].Min);
            SetDlgItemText(hWnd,PrefixControls[n].MatchBox,buffer);

            SendDlgItemMessage(hWnd,PrefixControls[n].Slider,TBM_SETPOS,TRUE,0);
            SetDlgItemInt(hWnd,PrefixControls[n].Min,Prefix->Mod[n].Min,0);
            SetDlgItemInt(hWnd,PrefixControls[n].Max,Prefix->Mod[n].Max-1,0);

            PrefixControls[n].ForceMatch = 0;
            PrefixControls[n].ForceValue = 0;
        }
        else
        {
            EnableWindow(GetDlgItem(hWnd,PrefixControls[n].MatchBox),TRUE);

            if (PrefixControls[n].ForceMatch)
            {
                EnablePrefixControls(hWnd,n,TRUE);

                sprintf(buffer,"Force Value of %s Prefix Effect to %i",nth,Prefix->Mod[n].Min + PrefixControls[n].ForceV
alue);
                SetDlgItemText(hWnd,PrefixControls[n].MatchBox,buffer);

                SetDlgItemInt(hWnd,PrefixControls[n].Min,Prefix->Mod[n].Min,0);
                SetDlgItemInt(hWnd,PrefixControls[n].Max,Prefix->Mod[n].Max-1,0);

                SendDlgItemMessage(hWnd,PrefixControls[n].Slider,TBM_SETRANGE,TRUE,
                    MAKELONG(0,Prefix->Mod[n].Max - Prefix->Mod[n].Min -1));
                SendDlgItemMessage(hWnd,PrefixControls[n].Slider,TBM_SETPOS,TRUE,PrefixControls[n].ForceValue);
            }
            else
            {
                EnablePrefixControls(hWnd,n,FALSE);

                sprintf(buffer,"Force Value of %s Prefix Effect",nth);
                SetDlgItemText(hWnd,PrefixControls[n].MatchBox,buffer);

                SendDlgItemMessage(hWnd,PrefixControls[n].Slider,TBM_SETPOS,TRUE,0);
```

```
                SetDlgItemInt(hWnd,PrefixControls[n].Min,Prefix->Mod[n].Min,0);
                SetDlgItemInt(hWnd,PrefixControls[n].Max,Prefix->Mod[n].Max-1,0);
            }
        }
    }
    for(n=0;n<1;n++)
    {
        char *nth = n == 0 ? "1st" :
                    n == 1 ? "2nd" :
                    n == 2 ? "3rd" : "4th";

        if (!SuffixMatch || !Suffix || !Suffix->Mod[n].Code)
        {
            EnableWindow(GetDlgItem(hWnd,SuffixControls[n].MatchBox),FALSE);
            CheckDlgButton(hWnd,SuffixControls[n].MatchBox,BST_UNCHECKED);

            EnableSuffixControls(hWnd,n,FALSE);

            sprintf(buffer,"No %s Suffix Effect",nth);
            SetDlgItemText(hWnd,SuffixControls[n].MatchBox,buffer);

            SendDlgItemMessage(hWnd,SuffixControls[n].Slider,TBM_SETPOS,TRUE,0);
            SetDlgItemText(hWnd,SuffixControls[n].Min,"");
            SetDlgItemText(hWnd,SuffixControls[n].Max,"");

            SuffixControls[n].ForceMatch = 0;
            SuffixControls[n].ForceValue = 0;
        }
        else if (Suffix->Mod[n].Max - Suffix->Mod[n].Min <= 0)
        {
            EnableWindow(GetDlgItem(hWnd,SuffixControls[n].MatchBox),FALSE);
            CheckDlgButton(hWnd,SuffixControls[n].MatchBox,BST_UNCHECKED);
            EnableSuffixControls(hWnd,n,FALSE);

            sprintf(buffer,"%s Suffix Effect fixed to %i",nth,Suffix->Mod[n].Min);
            SetDlgItemText(hWnd,SuffixControls[n].MatchBox,buffer);

            SendDlgItemMessage(hWnd,SuffixControls[n].Slider,TBM_SETPOS,TRUE,0);
            SetDlgItemInt(hWnd,SuffixControls[n].Min,Suffix->Mod[n].Min,0);
            SetDlgItemInt(hWnd,SuffixControls[n].Max,Suffix->Mod[n].Max-1,0);

            SuffixControls[n].ForceMatch = 0;
            SuffixControls[n].ForceValue = 0;
        }
        else
        {
            EnableWindow(GetDlgItem(hWnd,SuffixControls[n].MatchBox),TRUE);

            if (SuffixControls[n].ForceMatch)
            {
                EnableSuffixControls(hWnd,n,TRUE);

                sprintf(buffer,"Force Value of %s Suffix Effect to %i",nth,Suffix->Mod[n].Min + SuffixControls[n].ForceV
alue);
                SetDlgItemText(hWnd,SuffixControls[n].MatchBox,buffer);

                SetDlgItemInt(hWnd,SuffixControls[n].Min,Suffix->Mod[n].Min,0);
                SetDlgItemInt(hWnd,SuffixControls[n].Max,Suffix->Mod[n].Max-1,0);

                SendDlgItemMessage(hWnd,SuffixControls[n].Slider,TBM_SETRANGE,TRUE,
                    MAKELONG(0,Suffix->Mod[n].Max - Suffix->Mod[n].Min -1));
                SendDlgItemMessage(hWnd,SuffixControls[n].Slider,TBM_SETPOS,TRUE,SuffixControls[n].ForceValue);
            }
            else
            {
                EnableSuffixControls(hWnd,n,FALSE);

                sprintf(buffer,"Force Value of %s Suffix Effect",nth);
                SetDlgItemText(hWnd,SuffixControls[n].MatchBox,buffer);

                SendDlgItemMessage(hWnd,SuffixControls[n].Slider,TBM_SETPOS,TRUE,0);

                SetDlgItemInt(hWnd,SuffixControls[n].Min,Suffix->Mod[n].Min,0);
                SetDlgItemInt(hWnd,SuffixControls[n].Max,Suffix->Mod[n].Max-1,0);
            }
        }
    }
```

```
    {
        DWORDLONG Average = SearchAverage();

        if (Average < 0)
            SetDlgItemText(hWnd,IDC_TAB2Magic_Average,"Combination can possibly never be found!");
        else
        {
            sprintf(buffer,"Average Tries: %lu",Average);
            SetDlgItemText(hWnd,IDC_TAB2Magic_Average,buffer);
        }
    }

    { // Update ELevel Requirements

        int ELevel = 0;

        if (Prefix) {
            if (Prefix->ELevel > ELevel)
                ELevel = Prefix->ELevel;
        }

        if (Suffix) {
            if (Suffix->ELevel > ELevel)
                ELevel = Suffix->ELevel;
        }

        SetDlgItemInt(hWnd,IDC_TAB2Magic_CurrentELevel,ELevel,FALSE);

    } // Update ELevel Requirements
}

void UpdateTab2MagicTrees(HWND hWnd)
{
    LoadPreSuffixTree(hWnd);
    LoadSuffixTree(hWnd);

    if (RestrictELevel)
    {
        CheckDlgButton(hWnd,IDC_TAB2Magic_LockELevel,TRUE);
        EnableWindow(GetDlgItem(hWnd,IDC_TAB2Magic_LockValue),TRUE);
        SetDlgItemInt(hWnd,IDC_TAB2Magic_LockValue,RestrictELevelValue,FALSE);
    }
    else {
        CheckDlgButton(hWnd,IDC_TAB2Magic_LockELevel,FALSE);
        EnableWindow(GetDlgItem(hWnd,IDC_TAB2Magic_LockValue),FALSE);
    }
}

LRESULT CALLBACK Tab2MagicDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
    switch(uMsg)
    {
    case WM_INITDIALOG:
        {
            SendDlgItemMessage(hWnd,IDC_CHELP,BM_SETIMAGE,IMAGE_ICON,(LPARAM) hIconHelp);

            RestrictELevel = true;
            RestrictELevelValue = fc.gf.level;
            UpdateTab2MagicTrees(hWnd);

            PrefixMatch = false;
            PrefixControls[0].ForceMatch = false;
            PrefixControls[1].ForceMatch = false;
            PrefixControls[2].ForceMatch = false;
            PrefixControls[3].ForceMatch = false;
            EnablePrefixControls(hWnd,FALSE);

            SuffixMatch = false;
            SuffixControls[0].ForceMatch = false;
            EnableSuffixControls(hWnd,FALSE);

            OrigModLevel = SelItem->MagicLevel();

            UpdateTab2Magic(hWnd);
        }
        return true;
    case WM_COMMAND:
        {
            switch(LOWORD(wParam))
```

```
            {
            case IDOK:
                {
                    // Check Combination
                    {
                        if (PrefixMatch && SuffixMatch)
                        {
                            if (!Prefix && !Suffix)
                            {
                                MessageBox(hWnd,"You must select either a Prefix or a Suffix.",PROGRAMNAME,MB_OK | MB_IC
ONSTOP);
                                break;
                            }
                        }
                    }

                    SelItem->DWBHistory.StepAdd(SelItem);

                    HGLOBAL TD = GlobalAlloc(GMEM_ZEROINIT,sizeof MagicSearchThread);
                    struct MagicSearchThread* NewThread = (MagicSearchThread*)GlobalLock(TD);;
                    NewThread->ThreadData = TD;

                    NewThread->Item = SelItem;

                    NewThread->PrefixMatch = PrefixMatch;
                    NewThread->Prefix = Prefix;
                    NewThread->SuffixMatch = SuffixMatch;
                    NewThread->Suffix = Suffix;

                    NewThread->ForcePrefixMatch[0] = PrefixControls[0].ForceMatch;
                    NewThread->ForcePrefixMatch[1] = PrefixControls[1].ForceMatch;
                    NewThread->ForcePrefixMatch[2] = PrefixControls[2].ForceMatch;
                    NewThread->ForcePrefixMatch[3] = PrefixControls[3].ForceMatch;
                    NewThread->ForcePrefixValue[0] = PrefixControls[0].ForceValue;
                    NewThread->ForcePrefixValue[1] = PrefixControls[1].ForceValue;
                    NewThread->ForcePrefixValue[2] = PrefixControls[2].ForceValue;
                    NewThread->ForcePrefixValue[3] = PrefixControls[3].ForceValue;

                    NewThread->ForceSuffixMatch[0] = SuffixControls[0].ForceMatch;
                    NewThread->ForceSuffixValue[0] = SuffixControls[0].ForceValue;

                    DWORD ThreadID;
                    NewThread->Thread = CreateThread(NULL,0,&BruteForceAttack,NewThread,0,&ThreadID);

                    EndDialog(hWnd,IDOK);
                }
                break;
            case IDCANCEL:
                SelItem->SetMagicLevel(OrigModLevel);
                EndDialog(hWnd,IDCANCEL);
                break;
            case IDC_TAB2Magic_PrefixMatch:
                if (HIWORD(wParam) == BN_CLICKED)
                {
                    if (IsDlgButtonChecked(hWnd,IDC_TAB2Magic_PrefixMatch) == BST_CHECKED)
                    {
                        EnablePrefixControls(hWnd,TRUE);
                        PrefixMatch = true;
                    }
                    else
                    {
                        EnablePrefixControls(hWnd,FALSE);
                        PrefixMatch = false;
                    }
                    UpdateTab2Magic(hWnd);
                }
                break;
            case IDC_TAB2Magic_SuffixMatch:
                if (HIWORD(wParam) == BN_CLICKED)
                {
                    if (IsDlgButtonChecked(hWnd,IDC_TAB2Magic_SuffixMatch) == BST_CHECKED)
                    {
                        EnableSuffixControls(hWnd,TRUE);
                        SuffixMatch = true;
                    }
                    else
                    {
                        EnableSuffixControls(hWnd,FALSE);
                        SuffixMatch = false;
```

```
        }
        UpdateTab2Magic(hWnd);
    }
    break;
case IDC_TAB2Magic_Prefix1Match:
case IDC_TAB2Magic_Prefix2Match:
case IDC_TAB2Magic_Prefix3Match:
case IDC_TAB2Magic_Prefix4Match:
    if (HIWORD(wParam) == BN_CLICKED)
    {
        int n = LOWORD(wParam)-IDC_TAB2Magic_Prefix1Match;

        if (IsDlgButtonChecked(hWnd,LOWORD(wParam)) == BST_CHECKED)
            PrefixControls[n].ForceMatch = true;
        else
            PrefixControls[n].ForceMatch = false;

        PrefixControls[n].ForceValue = 0;

        UpdateTab2Magic(hWnd);
    }
    break;
case IDC_TAB2Magic_Suffix1Match:
    if (HIWORD(wParam) == BN_CLICKED)
    {
        if (IsDlgButtonChecked(hWnd,LOWORD(wParam)) == BST_CHECKED)
            SuffixControls[0].ForceMatch = true;
        else
            SuffixControls[0].ForceMatch = false;

        SuffixControls[0].ForceValue = 0;

        UpdateTab2Magic(hWnd);
    }
    break;
case IDC_TAB2Magic_LockELevel:
    if (HIWORD(wParam) == BN_CLICKED)
    {
        if (IsDlgButtonChecked(hWnd,LOWORD(wParam)) == BST_UNCHECKED)
            RestrictELevel = true;
        else
            RestrictELevel = false;

        UpdateTab2MagicTrees(hWnd);
    }
    break;
case IDC_TAB2Magic_LockValue:
    if (HIWORD(wParam) == EN_KILLFOCUS)
    {
        if (IsDlgButtonChecked(hWnd,IDC_TAB2Magic_LockELevel) == BST_CHECKED) {
            int x = GetDlgItemInt(hWnd,IDC_TAB2Magic_LockValue,NULL,FALSE);
            if (x != RestrictELevelValue) {
                RestrictELevelValue = x;
                UpdateTab2MagicTrees(hWnd);
            }
        }
    }
    break;
case IDC_CHELP:
    ToggleHelpBox(hWnd,IDH_TAB2Magic);
    break;
case IDC_TAB2Magic_Clear:
    {
        PrefixMatch = false;
        PrefixControls[0].ForceMatch = false;
        PrefixControls[1].ForceMatch = false;
        PrefixControls[2].ForceMatch = false;
        PrefixControls[3].ForceMatch = false;
        EnablePrefixControls(hWnd,FALSE);

        SuffixMatch = false;
        SuffixControls[0].ForceMatch = false;
        EnableSuffixControls(hWnd,FALSE);

        CheckDlgButton(hWnd,IDC_TAB2Magic_PrefixMatch,PrefixMatch);
        CheckDlgButton(hWnd,IDC_TAB2Magic_SuffixMatch,SuffixMatch);

        UpdateTab2Magic(hWnd);
    }
```

```
            break;
        }
    }
    return false;
case WM_NOTIFY:
    switch(((NMHDR*)lParam)->idFrom)
    {
    case IDC_TAB2Magic_PrefixTree:
        switch(((NMHDR*)lParam)->code)
        {
        case TVN_SELCHANGED:
            {
                NM_TREEVIEW *NMTreeView = (NM_TREEVIEW*) lParam;

                PrefixTree = NMTreeView->itemNew.lParam;
                if (PrefixTree >= 0)
                {
                    Prefix = &MagicPrefixTable[PrefixTree];
                    PrefixControls[0].ForceValue = 0;
                    PrefixControls[1].ForceValue = 0;
                    PrefixControls[2].ForceValue = 0;
                    PrefixControls[3].ForceValue = 0;
                }
                else
                    Prefix = 0;

                UpdateTab2Magic(hWnd);
            }
            break;
        }
        break;
    case IDC_TAB2Magic_SuffixTree:
        switch(((NMHDR*)lParam)->code)
        {
        case TVN_SELCHANGED:
            {
                NM_TREEVIEW *NMTreeView = (NM_TREEVIEW*) lParam;

                SuffixTree = NMTreeView->itemNew.lParam;
                if (SuffixTree >= 0)
                {
                    Suffix = &MagicSuffixTable[SuffixTree & 0xFF];
                    SuffixControls[0].ForceValue = 0;
                }
                else
                    Suffix = 0;

                UpdateTab2Magic(hWnd);
            }
            break;
        }
        break;
    }
    return false;
case WM_HSCROLL:
    if ((HWND)lParam == GetDlgItem(hWnd,IDC_TAB2Magic_Prefix1Value))
    {
        PrefixControls[0].ForceValue = SendMessage((HWND)lParam,TBM_GETPOS,0,0);
        UpdateTab2Magic(hWnd);
    }
    if ((HWND)lParam == GetDlgItem(hWnd,IDC_TAB2Magic_Prefix2Value))
    {
        PrefixControls[1].ForceValue = SendMessage((HWND)lParam,TBM_GETPOS,0,0);
        UpdateTab2Magic(hWnd);
    }
    if ((HWND)lParam == GetDlgItem(hWnd,IDC_TAB2Magic_Prefix3Value))
    {
        PrefixControls[2].ForceValue = SendMessage((HWND)lParam,TBM_GETPOS,0,0);
        UpdateTab2Magic(hWnd);
    }
    if ((HWND)lParam == GetDlgItem(hWnd,IDC_TAB2Magic_Prefix4Value))
    {
        PrefixControls[3].ForceValue = SendMessage((HWND)lParam,TBM_GETPOS,0,0);
        UpdateTab2Magic(hWnd);
    }
    if ((HWND)lParam == GetDlgItem(hWnd,IDC_TAB2Magic_Suffix1Value))
    {
        SuffixControls[0].ForceValue = SendMessage((HWND)lParam,TBM_GETPOS,0,0);
        UpdateTab2Magic(hWnd);
```

```
        }
        break;
    case WM_CLOSE:
        {
            SelItem->SetMagicLevel(OrigModLevel);
            EndDialog(hWnd,IDCANCEL);
        }
        return false;
    case WM_DESTROY:
        {
            CloseHelpBox();
        }
        return false;
    }
    return false;
}
```

```
// Tab2ItemList.cpp from D2E

#include "JamellaD2E.h"

HWND hItemListBox;

void UpdateTab2ItemList()
{
    if (!hItemListBox) return;

    HWND hItemList = GetDlgItem(hItemListBox,IDC_TAB2ItemList);

    SendMessage(hItemList,LB_RESETCONTENT,0,0);

    for(Item *I = Items;I != 0;I = I->Next())
    {
        int ix = SendMessage(hItemList,LB_ADDSTRING,0,(LPARAM)I->Name());
        SendMessage(hItemList,LB_SETITEMDATA,ix,(LPARAM) I);

        if (I == SelItem)
            SendMessage(hItemList,LB_SETCURSEL,ix,0);
    }
}

LRESULT CALLBACK Tab2ItemListDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
    switch(uMsg)
    {
    case WM_INITDIALOG:
        {
            {
                RECT rWindow,DialogPos;
                GetWindowRect(hWnd,&rWindow);

                GetWindowRect(hMainDialog,&DialogPos);
                DialogPos.left -= rWindow.right - rWindow.left;

                SetWindowPos(hWnd,HWND_TOP,
                    DialogPos.left,DialogPos.top,
                    0,0,SWP_NOSIZE | SWP_NOACTIVATE);
            }

            CheckDlgButton(hExpertBox,IDC_TAB2E_ItemList,BST_CHECKED);

            hItemListBox = hWnd;
            UpdateTab2ItemList();

            ShowWindow(hWnd,SW_SHOWNOACTIVATE);
        }
        return true;
    case WM_COMMAND:
        if (LOWORD(wParam) == IDC_TAB2ItemList && HIWORD(wParam) == LBN_SELCHANGE) {

            int ix = SendDlgItemMessage(hWnd,IDC_TAB2ItemList,LB_GETCURSEL,0,0);
            SelItem = (Item *)SendDlgItemMessage(hWnd,IDC_TAB2ItemList,LB_GETITEMDATA,ix,0);
            UpdateTab2();
            InvalidateRect(hTabDialog,NULL,FALSE);

        }
        return false;
    case WM_CLOSE:
        {
            if (hItemListBox)
                DestroyWindow(hItemListBox);
        }
        return false;
    case WM_DESTROY:
        {
            CheckDlgButton(hExpertBox,IDC_TAB2E_ItemList,BST_UNCHECKED);
            hItemListBox = 0;
        }
        return false;
    }
    return false;
}
```

```
// Tab2R.cpp from D2E

#include "JamellaD2E.h"

// Tab2R ToolBox
HWND hRandomBox;
int RandomBoxDW;

void UpdateTab2Rnd()
{
    if (!hRandomBox) return;

    if (SelItem)
    {
        char output[2048] = "";

        RAND s,r = { 0, 666 };
        int x;

        if (RandomBoxDW == 0)
        {
            r.Seed = SelItem->DWA();
            x = -1;
        }
        else
        {
            r.Seed = SelItem->DWB();
            x = StartRandoms(SelItem,&s);
        }

        for(int z=0;z<50;z++)
        {
            sprintf(buffer,"%2d: %08X %08X\r\n",z+1,r.Seed,r.Carry);
            strcat(output,buffer);
            if (x == z)
                strcat(output,"Prefix:\r\n");

            Random(&r);
        }

        SetDlgItemText(hRandomBox,IDC_TAB2Rnd_Edit,output);
    }
    else
    {
        SetDlgItemText(hRandomBox,IDC_TAB2Rnd_Edit,"");
    }
}

LRESULT CALLBACK Tab2RndDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
    switch(uMsg)
    {
    case WM_INITDIALOG:
        {
            {
                RECT rWindow,DialogPos;
                GetWindowRect(hWnd,&rWindow);

                GetWindowRect(hMainDialog,&DialogPos);
                DialogPos.left -= rWindow.right - rWindow.left;

                SetWindowPos(hWnd,HWND_TOP,
                    DialogPos.left,DialogPos.top,
                    0,0,SWP_NOSIZE | SWP_NOACTIVATE);
            }

            ShowWindow(hWnd,SW_SHOWNOACTIVATE);
        }
        return true;
    case WM_COMMAND:
        {
        }
        return false;
    case WM_CLOSE:
        {
            if (hRandomBox)
                DestroyWindow(hRandomBox);
        }
        return false;
```

```
    case WM_DESTROY:
        {
            CheckDlgButton(hExpertBox,IDC_TAB2E_RandA,BST_UNCHECKED);
            CheckDlgButton(hExpertBox,IDC_TAB2E_RandB,BST_UNCHECKED);
            hRandomBox = 0;
        }
        return false;
    }
    return false;
}
```

```
// Tab2Rare.cpp from D2E

#include "JamellaD2E.h"

HCURSOR hCurAdd;
static HWND hTVP,hTVS,hList;
static bool userselecting = false;

static bool RestrictELevel;
static int  RestrictELevelValue;

#define ZEROATTR        -4
#define RANDOMATTR      -5

static DWORD WINAPI BruteForceAttack(LPVOID ThreadParameters)
{
    RareSearchThread *P = (RareSearchThread *)ThreadParameters;
    P->Advanced = true;

    P->Dialog = CreateDialogParam(hInstance,MAKEINTRESOURCE(IDD_TAB2S),hMainDialog,(DLGPROC) Tab2SearchDialogProc, (LPAR
AM) P);
    if (!P->Dialog) return -1;

    Item* I = P->Item;
    I->Busy = true;

    int MagicLevel = 0;
    for(int z=0;z<6;z++)
    {
        if (!P->Attribute[z]) continue;
        MagicLevel = max(MagicLevel,P->Attribute[z]->ModLevel);
    }

    int nModMin = 0,nModMax = 0;
    // Precalculate Modifier Number Range
    {
        for(int z=0;z<6;z++)
        {
            if (P->iAttribute[z] == RANDOMATTR) {
                nModMax++;
            }
            if (!P->Attribute[z]) continue;
            nModMin++;
            nModMax++;
        }
    }

    bool QuitMessage = false;

    DWORD SDWB = I->DWB();

    for(int MG = MagicLevel;MG == MagicLevel || (P->TraverseMagicLevels && MG <= MAXMODLEVEL);MG++)
    {
        I->SetMagicLevel(MG - 2);

        DWORD RDWB = SDWB;
        P->Counter = 0;

        while(! (QuitMessage = CheckPollMessages()) )
        {
            if (!P->Running)
                continue;

            I->SetDWB(RDWB++);

            P->Counter++;
            if (P->Counter == 0)
                break;

            if (!I->MD->QuickDecode()) continue;

            if (P->NamePrefix >= 0)
                if (I->MD->RarePrefix != &RarePrefixTable[P->NamePrefix]) continue;

            if (P->NameSuffix >= 0)
                if (I->MD->RareSuffix != &RareSuffixTable[P->NameSuffix]) continue;

            if (nModMin > I->MD->nRareFix) continue;
            if (nModMax < I->MD->nRareFix) continue;
```

```
            int z;
            for(z=0;z<6;z++)
            {
                if (!P->Attribute[z]) continue;

                int m;
                for(m=0;m<6;m++)
                {
                    if (!I->MD->RareFix[m]) continue;
                    if (I->MD->RareFix[m] == P->Attribute[z])
                        break;
                }
                if (m == 6) break;
            }
            if (z == 6) {

                if (P->HitsSelection) {

                    SearchHit *H = new SearchHit;
                    H->MagicLevel = I->MagicLevel();
                    H->DWA = I->DWA();
                    H->DWB = I->DWB();
                    H->List = 0;

                    if (P->Hits) {

                        SearchHit *S = P->Hits;
                        while(S->List)
                            S = S->List;
                        S->List = H;
                    }
                    else
                        P->Hits = H;

                    PostMessage(P->Dialog,WM_USER,0,0);
                    MessageBeep(MB_ICONASTERISK);
                }
                else
                    break;
            }
        }

        if (QuitMessage) break;
    }

    I->Busy = false;
    I->Decoded = false;

    EndDialog(P->Dialog,IDOK);

    if (MG >= MAXMODLEVEL || P->TraverseMagicLevels && !QuitMessage)
        MessageBox(NULL,"Searched all combinations in the scope of this item. Your desired combination is not available
on the item. Select similar attributes and try again.",PROGRAMNAME,MB_OK | MB_ICONASTERISK);
    else
        MessageBeep(MB_ICONASTERISK);

    UpdateTab2();
    InvalidateRect(hTabDialog,NULL,FALSE);

    CloseHandle(P->Thread);
    for(SearchHit *H = P->Hits;H != 0;) {
        SearchHit *T = H->List;
        delete H;
        H = T;
    }

    GlobalFree(P->ThreadData);
    return 0;
}

inline void LoadNamePrefixTree(HWND hWnd)
{
    HWND hList = GetDlgItem(hWnd,IDC_TAB2Rare_NamePrefix);

    SendMessage(hList,LB_RESETCONTENT,0,0);

    for(int z=0;z<nRarePrefixTable;z++)
    {
```

```
        if (RarePrefixTable[z].Text == 0) continue;
        if ((RarePrefixTable[z].RareMask & SelItem->Info->RareMask) == 0) continue;

        int index = SendMessage(hList,LB_ADDSTRING,0,(DWORD) RarePrefixTable[z].Text);
        SendMessage(hList,LB_SETITEMDATA,index,(DWORD) z);
    }

    SendMessage(hList,LB_INSERTSTRING,0,(DWORD) "unspecified");
    SendMessage(hList,LB_SETITEMDATA,0,(DWORD) -1);
    SendMessage(hList,LB_SETSEL,TRUE,0);
}
inline void LoadNameSuffixTree(HWND hWnd)
{
    HWND hList = GetDlgItem(hWnd,IDC_TAB2Rare_NameSuffix);

    SendMessage(hList,LB_RESETCONTENT,0,0);

    for(int z=0;z<nRareSuffixTable;z++)
    {
        if (RareSuffixTable[z].Text == 0) continue;
        if ((RareSuffixTable[z].RareMask & SelItem->Info->RareMask) == 0) continue;

        int index = SendMessage(hList,LB_ADDSTRING,0,(DWORD) RareSuffixTable[z].Text);
        SendMessage(hList,LB_SETITEMDATA,index,(DWORD) z);
    }

    SendMessage(hList,LB_INSERTSTRING,0,(DWORD) "unspecified");
    SendMessage(hList,LB_SETITEMDATA,0,(DWORD) -1);
    SendMessage(hList,LB_SETSEL,TRUE,0);
}
inline void LoadAttributesTree(HWND hWnd)
{
    hTVP = GetDlgItem(hWnd,IDC_TAB2Rare_TreePrefix);
    hTVS = GetDlgItem(hWnd,IDC_TAB2Rare_TreeSuffix);

    TreeView_DeleteAllItems(hTVP);
    TreeView_DeleteAllItems(hTVS);

    TV_ITEM TVItem;
    TV_INSERTSTRUCT TVInsert;
    HTREEITEM hBranch;

    HTREEITEM hPrefix;
    // Prefix Entry
    {
        ZeroMemory(&TVItem,sizeof TVItem);
        TVItem.mask = TVIF_TEXT | TVIF_PARAM | TVIF_STATE;
        TVItem.pszText = "Prefix";
        TVItem.lParam = (LPARAM)-1;
        TVItem.state = TVIS_EXPANDED;
        TVItem.stateMask = TVIS_EXPANDED;

        TVInsert.hParent = 0;
        TVInsert.hInsertAfter = TVI_ROOT;
        TVInsert.item = TVItem;

        hPrefix = TreeView_InsertItem(hTVP,&TVInsert);
    }

    // None Entry
    {
        ZeroMemory(&TVItem,sizeof TVItem);
        TVItem.mask = TVIF_TEXT | TVIF_PARAM;
        TVItem.pszText = "Zero Attribute";
        TVItem.lParam = (LPARAM)ZEROATTR;

        TVInsert.hParent = hPrefix;
        TVInsert.hInsertAfter = TVI_LAST;
        TVInsert.item = TVItem;

        TreeView_InsertItem(hTVP,&TVInsert);
    }

    // Random Entry
    {
        ZeroMemory(&TVItem,sizeof TVItem);
        TVItem.mask = TVIF_TEXT | TVIF_PARAM;
        TVItem.pszText = "Random Attribute";
        TVItem.lParam = (LPARAM)RANDOMATTR;
```

```
        TVInsert.hParent = hPrefix;
        TVInsert.hInsertAfter = TVI_LAST;
        TVInsert.item = TVItem;

        TreeView_InsertItem(hTVP,&TVInsert);
    }

    for(int z=0;z<nMagicPreSuffixTree;z++)
        MagicPreSuffixTree[z].hTree = 0;

    for(z=0;z<nMagicPreSuffixTree;z++)
    {
        if (MagicPreSuffixTree[z].Depth > 1) continue;

        ZeroMemory(&TVItem,sizeof TVItem);
        TVItem.mask = TVIF_TEXT | TVIF_PARAM;
        TVItem.pszText = MagicPreSuffixTree[z].Text;
        TVItem.lParam = (LPARAM)-1;

        TVInsert.hParent = hPrefix;
        TVInsert.hInsertAfter = TVI_LAST;
        TVInsert.item = TVItem;

        for(z++;z<nMagicPreSuffixTree;z++)
        {
            if (MagicPreSuffixTree[z].Depth < 2)
            {
                z--;
                break;
            }

            if (MagicPreSuffixTree[z].ModID & 256) continue;

            if ((MagicPrefixTable[MagicPreSuffixTree[z].ModID & 0xFF].MagicMask & SelItem->Info->MagicMask) == 0) contin
ue;

            // Check if ELevel exceeded
            if (RestrictELevel && RestrictELevelValue < MagicPrefixTable[MagicPreSuffixTree[z].ModID].ELevel) continue;

            if (TVInsert.hParent == hPrefix)
            {
                hBranch = MagicPreSuffixTree[z].hTree =
                    TreeView_InsertItem(hTVP,&TVInsert);
            }

            ZeroMemory(&TVItem,sizeof TVItem);
            TVItem.mask = TVIF_TEXT | TVIF_PARAM;
            TVItem.pszText = MagicPreSuffixTree[z].Text;
            TVItem.lParam = MagicPreSuffixTree[z].ModID;

            TVInsert.hParent = hBranch;
            TVInsert.hInsertAfter = TVI_LAST;
            TVInsert.item = TVItem;

            MagicPreSuffixTree[z].hTree = TreeView_InsertItem(hTVP,&TVInsert);
        }
    }

    HTREEITEM hSuffix;
    // Suffix Entry
    {
        ZeroMemory(&TVItem,sizeof TVItem);
        TVItem.mask = TVIF_TEXT | TVIF_PARAM | TVIF_STATE;
        TVItem.pszText = "Suffix";
        TVItem.lParam = (LPARAM)-1;
        TVItem.state = TVIS_EXPANDED;
        TVItem.stateMask = TVIS_EXPANDED;

        TVInsert.hParent = 0;
        TVInsert.hInsertAfter = TVI_ROOT;
        TVInsert.item = TVItem;

        hSuffix = TreeView_InsertItem(hTVS,&TVInsert);
    }

    // None Entry
    {
        ZeroMemory(&TVItem,sizeof TVItem);
```

```
        TVItem.mask = TVIF_TEXT | TVIF_PARAM;
        TVItem.pszText = "Zero Attribute";
        TVItem.lParam = (LPARAM)ZEROATTR;

        TVInsert.hParent = hSuffix;
        TVInsert.hInsertAfter = TVI_LAST;
        TVInsert.item = TVItem;

        TreeView_InsertItem(hTVS,&TVInsert);
    }

    // Random Entry
    {
        ZeroMemory(&TVItem,sizeof TVItem);
        TVItem.mask = TVIF_TEXT | TVIF_PARAM;
        TVItem.pszText = "Random Attribute";
        TVItem.lParam = (LPARAM)RANDOMATTR;

        TVInsert.hParent = hSuffix;
        TVInsert.hInsertAfter = TVI_LAST;
        TVInsert.item = TVItem;

        TreeView_InsertItem(hTVS,&TVInsert);
    }

    for(z=0;z<nMagicPreSuffixTree;z++)
    {
        if (MagicPreSuffixTree[z].Depth > 1) continue;

        ZeroMemory(&TVItem,sizeof TVItem);
        TVItem.mask = TVIF_TEXT | TVIF_PARAM;
        TVItem.pszText = MagicPreSuffixTree[z].Text;
        TVItem.lParam = (LPARAM)-1;

        TVInsert.hParent = hSuffix;
        TVInsert.hInsertAfter = TVI_LAST;
        TVInsert.item = TVItem;

        for(z++;z<nMagicPreSuffixTree;z++)
        {
            if (MagicPreSuffixTree[z].Depth < 2)
            {
                z--;
                break;
            }

            if (!(MagicPreSuffixTree[z].ModID & 256)) continue;

            if ((MagicSuffixTable[MagicPreSuffixTree[z].ModID & 0xFF].MagicMask & SelItem->Info->MagicMask) == 0) contin
ue;

            // Check if ELevel exceeded
            if (RestrictELevel && RestrictELevelValue < MagicSuffixTable[MagicPreSuffixTree[z].ModID & 0xFF].ELevel) con
tinue;

            if (TVInsert.hParent == hSuffix)
            {
                hBranch = MagicPreSuffixTree[z].hTree =
                    TreeView_InsertItem(hTVS,&TVInsert);
            }

            ZeroMemory(&TVItem,sizeof TVItem);
            TVItem.mask = TVIF_TEXT | TVIF_PARAM;
            TVItem.pszText = MagicPreSuffixTree[z].Text;
            TVItem.lParam = MagicPreSuffixTree[z].ModID;

            TVInsert.hParent = hBranch;
            TVInsert.hInsertAfter = TVI_LAST;
            TVInsert.item = TVItem;

            MagicPreSuffixTree[z].hTree = TreeView_InsertItem(hTVS,&TVInsert);
        }
    }
}

static int              SaveModLevel;
static int              SearchNamePrefix,SearchNameSuffix;

static struct
```

```
{
    int                 CheckID;
    int                 TextID;
    _MagicPreSuffix     *Selected;
    int                 iSelected;
    bool                tSelected;
}
Attribute[6] =
{
    { IDC_TAB2Rare_Check1, IDC_TAB2Rare_Text1 },
    { IDC_TAB2Rare_Check2, IDC_TAB2Rare_Text2 },
    { IDC_TAB2Rare_Check3, IDC_TAB2Rare_Text3 },
    { IDC_TAB2Rare_Check4, IDC_TAB2Rare_Text4 },
    { IDC_TAB2Rare_Check5, IDC_TAB2Rare_Text5 },
    { IDC_TAB2Rare_Check6, IDC_TAB2Rare_Text6 },
};

static int              UserSelected;

static DWORD SearchAverage()
{
    int MagicLevel = 0;
    for(int z=0;z<6;z++)
    {
        if (!Attribute[z].Selected) continue;
        MagicLevel = max(MagicLevel,Attribute[z].Selected->ModLevel);
    }
    SelItem->SetMagicLevel(MagicLevel - 2);

    SelItem->MD->BuildMagicBuffers();
    SelItem->MD->BuildRareBuffers();

    DWORDLONG Average = 1;

    if (SearchNamePrefix >= 0)
        Average *= SelItem->MD->nRarePrefixBuffer;

    if (SearchNameSuffix >= 0)
        Average *= SelItem->MD->nRareSuffixBuffer;

    for(z=0;z<6;z++)
    {
        if (!Attribute[z].Selected) continue;

        if (Attribute[z].tSelected == PREFIX)
            Average *= 2 * SelItem->MD->nPrefixBuffer;
        if (Attribute[z].tSelected == SUFFIX)
            Average *= 2 * SelItem->MD->nSuffixBuffer;
    }

    return DWORD(Average/2 +1);
}

void UpdateTab2Rare(HWND hWnd)
{
    for(int z=0;z<6;z++)
    {
        if (UserSelected == z)
            CheckDlgButton(hWnd,Attribute[z].CheckID,BST_CHECKED);
        else
            CheckDlgButton(hWnd,Attribute[z].CheckID,BST_UNCHECKED);

        if (Attribute[z].iSelected == ZEROATTR) {
            SetDlgItemText(hWnd,Attribute[z].CheckID,"Zero Attribute");
            SetDlgItemText(hWnd,Attribute[z].TextID,"");
        }
        else if (Attribute[z].iSelected == RANDOMATTR) {
            SetDlgItemText(hWnd,Attribute[z].CheckID,"Random Attribute");
            SetDlgItemText(hWnd,Attribute[z].TextID,"");
        }
        else if (Attribute[z].Selected) {
            SetDlgItemText(hWnd,Attribute[z].CheckID,Attribute[z].Selected->Text);
            SetDlgItemText(hWnd,Attribute[z].TextID,Attribute[z].Selected->Description);
        }
        else {
            SetDlgItemText(hWnd,Attribute[z].CheckID,"Error!!!");
            SetDlgItemText(hWnd,Attribute[z].TextID,"");
        }
    }
```

```
    // Display Average
    {
        DWORDLONG Average = SearchAverage();

        if (Average < 0)
            SetDlgItemText(hWnd,IDC_TAB2Rare_Average,"Combination can possibly never be found!");
        else
        {
            sprintf(buffer,"Average Tries: %lu",Average);
            SetDlgItemText(hWnd,IDC_TAB2Rare_Average,buffer);
        }
    }

    { // Update ELevel Requirements

        int ELevel = 0;

        for(int z=0;z<6;z++)
        {
            if (Attribute[z].Selected) {
                if (Attribute[z].Selected->ELevel > ELevel)
                    ELevel = Attribute[z].Selected->ELevel;
            }
        }

        SetDlgItemInt(hWnd,IDC_TAB2Rare_CurrentELevel,ELevel,FALSE);

    } // Update ELevel Requirements

    if (RestrictELevel)
    {
        CheckDlgButton(hWnd,IDC_TAB2Rare_LockELevel,TRUE);
        EnableWindow(GetDlgItem(hWnd,IDC_TAB2Rare_LockValue),TRUE);
        SetDlgItemInt(hWnd,IDC_TAB2Rare_LockValue,RestrictELevelValue,FALSE);
    }
    else {
        CheckDlgButton(hWnd,IDC_TAB2Rare_LockELevel,FALSE);
        EnableWindow(GetDlgItem(hWnd,IDC_TAB2Rare_LockValue),FALSE);
    }

}

static void TVMakeSelection(HWND hWnd,int n)
{
    for(int z=0;z<nMagicPreSuffixTree;z++)
    {
        if (MagicPreSuffixTree[z].ModID == n && MagicPreSuffixTree[z].hTree) {

            userselecting = true;

            if (MagicPreSuffixTree[z].ModID & 256) {
                TreeView_SelectItem(hTVP,0);
                TreeView_SelectItem(hTVS,MagicPreSuffixTree[z].hTree);
            }
            else {
                TreeView_SelectItem(hTVP,MagicPreSuffixTree[z].hTree);
                TreeView_SelectItem(hTVS,0);
            }

            userselecting = false;
            break;
        }
    }
}

static bool     MouseDraging = false;
static TV_ITEM  MouseDragItem;

LRESULT CALLBACK Tab2RareDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
    switch(uMsg)
    {
    case WM_INITDIALOG:
        {
            SendDlgItemMessage(hWnd,IDC_CHELP,BM_SETIMAGE,IMAGE_ICON,(LPARAM) hIconHelp);

            SaveModLevel = SelItem->MagicLevel();
```

```
            UserSelected = 0;

            LoadNamePrefixTree(hWnd);
            LoadNameSuffixTree(hWnd);
            LoadAttributesTree(hWnd);

            RestrictELevel = true;
            RestrictELevelValue = fc.gf.level;

            if (SelItem->MD->Decode())
            {
                SearchNamePrefix = FindRareNamePrefixByPointer(SelItem->MD->RarePrefix);
                SearchNameSuffix = FindRareNameSuffixByPointer(SelItem->MD->RareSuffix);

                if (SearchNamePrefix >= 0)
                    SendDlgItemMessage(hWnd,IDC_TAB2Rare_NamePrefix,LB_SELECTSTRING,-1,(LPARAM)RarePrefixTable[SearchNam
ePrefix].Text);
                if (SearchNameSuffix >= 0)
                    SendDlgItemMessage(hWnd,IDC_TAB2Rare_NameSuffix,LB_SELECTSTRING,-1,(LPARAM)RareSuffixTable[SearchNam
eSuffix].Text);

                for(int z=0;z<6;z++)
                {
                    if (SelItem->MD->RareFix[z]) {
                        Attribute[z].Selected = SelItem->MD->RareFix[z];
                        Attribute[z].tSelected = SelItem->MD->tRareFix[z];
                        Attribute[z].iSelected = FindMagicPreSuffixByPointer(Attribute[z].tSelected,Attribute[z].Selecte
d);
                    }
                    else {
                        Attribute[z].Selected = 0;
                        Attribute[z].iSelected = ZEROATTR;
                        Attribute[z].tSelected = PREFIX;
                    }
                }

            }
            else {
                SearchNamePrefix = SearchNameSuffix = -1;

                for(int z=0;z<6;z++)
                {
                    Attribute[z].Selected = 0;
                    Attribute[z].iSelected = RANDOMATTR;
                    Attribute[z].tSelected = PREFIX;
                }
            }

            UpdateTab2Rare(hWnd);
        }
        return true;
    case WM_COMMAND:
        {
            switch(LOWORD(wParam))
            {
            case IDOK:
                {
                    // Get Name Pre- and Suffixes
                    {
                        int index = SendDlgItemMessage(hWnd,IDC_TAB2Rare_NamePrefix,LB_GETCURSEL,0,0);
                        if (index < 0)
                        {
                            ErrorBox("You must select a Name Prefix!");
                            return false;
                        }
                        SearchNamePrefix = SendDlgItemMessage(hWnd,IDC_TAB2Rare_NamePrefix,LB_GETITEMDATA,index,0);

                        index = SendDlgItemMessage(hWnd,IDC_TAB2Rare_NameSuffix,LB_GETCURSEL,0,0);
                        if (index < 0)
                        {
                            ErrorBox("You must select a Name Suffix!");
                            return false;
                        }
                        SearchNameSuffix = SendDlgItemMessage(hWnd,IDC_TAB2Rare_NameSuffix,LB_GETITEMDATA,index,0);
                    }

                    // Check Prefix / Suffix Combinations
                    {
                        int n = 0;
```

```
            int nPrefix=0,nSuffix=0;
            for(int z=0;z<6;z++)
            {
                if (Attribute[z].iSelected == RANDOMATTR) {
                    n++;
                    continue;
                }
                if (!Attribute[z].Selected) continue;
                if (Attribute[z].tSelected == PREFIX) nPrefix++;
                if (Attribute[z].tSelected == SUFFIX) nSuffix++;
                n++;
            }

            if (nPrefix > 3)
            {
                ErrorBox("There can only be 3 Prefixes!");
                return false;
            }
            if (nSuffix > 3)
            {
                ErrorBox("There can only be 3 Suffixes!");
                return false;
            }
            if (n < 4)
            {
                ErrorBox("There must be at least 4 Attributes!");
                return false;
            }

            for(z=0;z<6;z++)
            {
                if (!Attribute[z].Selected) continue;
                for(int y=z-1;y>=0;y--)
                {
                    if (!Attribute[y].Selected) continue;
                    if (Attribute[z].Selected->Group == Attribute[y].Selected->Group)
                    {
                        ErrorBox("Duplicate Attribute Group found!");
                        return false;
                    }
                }
            }

        }

        SelItem->DWBHistory.StepAdd(SelItem);

        HGLOBAL TD = GlobalAlloc(GMEM_ZEROINIT,sizeof RareSearchThread);
        struct RareSearchThread* NewThread = (RareSearchThread*)GlobalLock(TD);;
        NewThread->ThreadData = TD;

        NewThread->Item = SelItem;

        NewThread->NamePrefix = SearchNamePrefix;
        NewThread->NameSuffix = SearchNameSuffix;

        for(int z=0;z<6;z++)
        {
            NewThread->Attribute[z] = Attribute[z].Selected;

            NewThread->iAttribute[z] = Attribute[z].iSelected;
            if (NewThread->iAttribute[z] >= 0)
                NewThread->iAttribute[z] &= 256;

            NewThread->tAttribute[z] = Attribute[z].tSelected;
        }

        DWORD ThreadID;
        NewThread->Thread = CreateThread(NULL,0,&BruteForceAttack,NewThread,0,&ThreadID);

        EndDialog(hWnd,IDOK);
    }
    break;
case IDCANCEL:
    SelItem->SetMagicLevel(SaveModLevel);
    EndDialog(hWnd,IDCANCEL);
    break;
case IDC_TAB2Rare_NamePrefix:
    if (HIWORD(wParam) == LBN_SELCHANGE)
```

```
    {
        int index = SendDlgItemMessage(hWnd,IDC_TAB2Rare_NamePrefix,LB_GETCURSEL,0,0);
        SearchNamePrefix = SendDlgItemMessage(hWnd,IDC_TAB2Rare_NamePrefix,LB_GETITEMDATA,index,0);
        UpdateTab2Rare(hWnd);
    }
    break;
case IDC_TAB2Rare_NameSuffix:
    if (HIWORD(wParam) == LBN_SELCHANGE)
    {
        int index = SendDlgItemMessage(hWnd,IDC_TAB2Rare_NameSuffix,LB_GETCURSEL,0,0);
        SearchNameSuffix = SendDlgItemMessage(hWnd,IDC_TAB2Rare_NameSuffix,LB_GETITEMDATA,index,0);
        UpdateTab2Rare(hWnd);
    }
    break;
case IDC_TAB2Rare_Check1:
    if (HIWORD(wParam) == BN_CLICKED) {
        UserSelected = 0;
        TVMakeSelection(hWnd,Attribute[UserSelected].iSelected);
        UpdateTab2Rare(hWnd);
    }
    break;
case IDC_TAB2Rare_Check2:
    if (HIWORD(wParam) == BN_CLICKED) {
        UserSelected = 1;
        TVMakeSelection(hWnd,Attribute[UserSelected].iSelected);
        UpdateTab2Rare(hWnd);
    }
    break;
case IDC_TAB2Rare_Check3:
    if (HIWORD(wParam) == BN_CLICKED) {
        UserSelected = 2;
        TVMakeSelection(hWnd,Attribute[UserSelected].iSelected);
        UpdateTab2Rare(hWnd);
    }
    break;
case IDC_TAB2Rare_Check4:
    if (HIWORD(wParam) == BN_CLICKED) {
        UserSelected = 3;
        TVMakeSelection(hWnd,Attribute[UserSelected].iSelected);
        UpdateTab2Rare(hWnd);
    }
    break;
case IDC_TAB2Rare_Check5:
    if (HIWORD(wParam) == BN_CLICKED) {
        UserSelected = 4;
        TVMakeSelection(hWnd,Attribute[UserSelected].iSelected);
        UpdateTab2Rare(hWnd);
    }
    break;
case IDC_TAB2Rare_Check6:
    if (HIWORD(wParam) == BN_CLICKED) {
        UserSelected = 5;
        TVMakeSelection(hWnd,Attribute[UserSelected].iSelected);
        UpdateTab2Rare(hWnd);
    }
    break;
case IDC_TAB2Rare_LockELevel:
    if (HIWORD(wParam) == BN_CLICKED)
    {
        if (IsDlgButtonChecked(hWnd,LOWORD(wParam)) == BST_UNCHECKED)
            RestrictELevel = true;
        else
            RestrictELevel = false;

        LoadAttributesTree(hWnd);
        UpdateTab2Rare(hWnd);
        TVMakeSelection(hWnd,Attribute[UserSelected].iSelected);
    }
    break;
case IDC_TAB2Rare_LockValue:
    if (HIWORD(wParam) == EN_KILLFOCUS)
    {
        if (IsDlgButtonChecked(hWnd,IDC_TAB2Rare_LockELevel) == BST_CHECKED) {
            int x = GetDlgItemInt(hWnd,IDC_TAB2Rare_LockValue,NULL,FALSE);
            if (x != RestrictELevelValue) {
                RestrictELevelValue = x;
                LoadAttributesTree(hWnd);
                UpdateTab2Rare(hWnd);
                TVMakeSelection(hWnd,Attribute[UserSelected].iSelected);
```

```
                    }
                }
            }
            break;
        case IDC_CHELP:
            ToggleHelpBox(hWnd,IDH_TAB2Rare);
            break;
        case IDC_TAB2Rare_Clear:
            {
                SendDlgItemMessage(hWnd,IDC_TAB2Rare_NamePrefix,LB_SETCURSEL,0,0);
                SendDlgItemMessage(hWnd,IDC_TAB2Rare_NameSuffix,LB_SETCURSEL,0,0);

                SearchNamePrefix = 0;
                SearchNameSuffix = 0;

                for(int z=0;z<6;z++)
                {
                    Attribute[z].Selected = 0;
                    Attribute[z].iSelected = ZEROATTR;
                    Attribute[z].tSelected = PREFIX;
                }

                UserSelected = 0;
                TVMakeSelection(hWnd,Attribute[UserSelected].iSelected);
                UpdateTab2Rare(hWnd);
            }
            break;
        }
    }
    return false;
case WM_NOTIFY:
    switch(((NMHDR*)lParam)->idFrom)
    {
    case IDC_TAB2Rare_TreePrefix:
    case IDC_TAB2Rare_TreeSuffix:
        switch(((NMHDR*)lParam)->code)
        {
        case TVN_SELCHANGED:
            {
                if (userselecting) break;
                userselecting = true;
                NM_TREEVIEW *NMTreeView = (NM_TREEVIEW *) lParam;

                if (UserSelected >= 0 && UserSelected <= 5) {
                    int n = UserSelected;

                    if (NMTreeView->itemNew.lParam >= 0)
                    {
                        Attribute[n].iSelected = NMTreeView->itemNew.lParam;

                        if ((Attribute[n].iSelected & 256) == 0) {
                            Attribute[n].tSelected = PREFIX;
                            Attribute[n].Selected = &MagicPrefixTable[Attribute[n].iSelected];
                            TreeView_SelectItem(hTVS,NULL);
                        }
                        else {
                            Attribute[n].tSelected = SUFFIX;
                            Attribute[n].Selected = &MagicSuffixTable[Attribute[n].iSelected & 255];
                            TreeView_SelectItem(hTVP,NULL);
                        }
                    }
                    else if (NMTreeView->itemNew.lParam == ZEROATTR || NMTreeView->itemNew.lParam == RANDOMATTR)
                    {
                        Attribute[n].iSelected = NMTreeView->itemNew.lParam;
                        Attribute[n].tSelected = PREFIX;
                        Attribute[n].Selected = 0;
                    }

                    UpdateTab2Rare(hWnd);
                }
                userselecting = false;
            }
            break;
        }
    }
    return false;
case WM_CLOSE:
    {
        SelItem->SetMagicLevel(SaveModLevel);
```

```
            EndDialog(hWnd,IDCANCEL);
        }
        return false;
    case WM_DESTROY:
        {
            CloseHelpBox();
        }
        return false;
    }
    return false;
}
```

```
#include "JamellaD2E.h"

const struct
{
    char    *Text;
    int     PriorityID;
    int     ClassID;
}
Priorities[] =
{
    { "Idle",           THREAD_PRIORITY_IDLE,           IDLE_PRIORITY_CLASS },
    { "Lowest",         THREAD_PRIORITY_LOWEST,         NORMAL_PRIORITY_CLASS },
    { "Below Normal",   THREAD_PRIORITY_BELOW_NORMAL,   NORMAL_PRIORITY_CLASS },
    { "Normal",         THREAD_PRIORITY_NORMAL,         NORMAL_PRIORITY_CLASS },
    { "Above Normal",   THREAD_PRIORITY_ABOVE_NORMAL,   HIGH_PRIORITY_CLASS },
    { "Highest",        THREAD_PRIORITY_HIGHEST,        HIGH_PRIORITY_CLASS },
    { "Time Critical",  THREAD_PRIORITY_TIME_CRITICAL,  REALTIME_PRIORITY_CLASS }
};
const int nPriorities = 7;

LRESULT CALLBACK Tab2SearchDialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
    switch(uMsg)
    {
    case WM_INITDIALOG:
        {
            // Save Parameter
            SetWindowLong(hWnd,GWL_USERDATA,(DWORD) lParam);
            SearchThread *P = (SearchThread *)lParam;
            P->Running = true;

            // Start Timer
            SetTimer(hWnd,IDT_TIMER,SEARCHPOLLER,NULL);
            PostMessage(hWnd,WM_TIMER,IDT_TIMER,0);

            SendDlgItemMessage(hWnd,IDC_CHELP,BM_SETIMAGE,IMAGE_ICON,(LPARAM) hIconHelp);

            // Simple Box Controls
            if (!P->Advanced)
            {
                // Setup Progress Bar
                SendDlgItemMessage(hWnd,IDC_TAB2SS_Scope,PBM_SETRANGE,0,MAKELPARAM(0,16384));
                SendDlgItemMessage(hWnd,IDC_TAB2SS_Scope,PBM_SETBARCOLOR,0,PROGRESSCOLOR);
                SendDlgItemMessage(hWnd,IDC_TAB2SS_Scope,PBM_SETPOS,0,0);
            }
            else
            {
                // Advanced Box Controls

                // Setup Progress Bar
                SendDlgItemMessage(hWnd,IDC_TAB2SA_Scope,PBM_SETRANGE,0,MAKELPARAM(0,16384));
                SendDlgItemMessage(hWnd,IDC_TAB2SA_Scope,PBM_SETBARCOLOR,0,PROGRESSCOLOR);
                SendDlgItemMessage(hWnd,IDC_TAB2SA_Scope,PBM_SETPOS,0,0);

                // Setup Priority Slider
                SendDlgItemMessage(hWnd,IDC_TAB2SA_Slider,TBM_SETRANGE,1,MAKELONG(0,nPriorities-1));
                SendDlgItemMessage(hWnd,IDC_TAB2SA_Slider,TBM_SETPOS,1,3);
            }

            // Show Windows
            ShowWindow(hWnd,SW_SHOW);

            InvalidateRect(hTabDialog,NULL,FALSE);
        }
        return true;
    case WM_TIMER:
        switch(wParam)
        {
        case IDT_TIMER:
            {
                SearchThread *P = (SearchThread *)GetWindowLong(hWnd,GWL_USERDATA);
                if (!P) break;
                if (!P->Item) break;

                char strbuffer[32];
                sprintf(strbuffer,"%lu Tries",P->Counter);

                if (!P->Advanced)
                {
```

```
                SetDlgItemText(hWnd,IDC_TAB2SS_Counter,strbuffer);

                SetDlgItemText(hWnd,IDC_TAB2SS_Current,P->Item->Name());

                SendDlgItemMessage(hWnd,IDC_TAB2SS_Scope,PBM_SETPOS,(P->Counter >> 16+2),0);
            }
            else
            {
                SetDlgItemText(hWnd,IDC_TAB2SA_Counter,strbuffer);
                SetDlgItemText(hWnd,IDC_TAB2SA_Current,P->Item->Name());
                SendDlgItemMessage(hWnd,IDC_TAB2SA_Scope,PBM_SETPOS,(P->Counter >> 16+2),0);

                SetDlgItemInt(hWnd,IDC_TAB2SA_MagicLevel,P->Item->MagicLevel(),FALSE);

                if (!P->Running) {
                    SetDlgItemText(hWnd,IDC_TAB2AS_Speed,"Search Stopped");
                }
                else if (P->Past[0] > 0) {
                    DWORD Speed = (P->Past[BFASPEEDPAST-1] - P->Past[0]) / (SEARCHPOLLER * BFASPEEDPAST / 1000);
                    sprintf(strbuffer,"%lu Tries/sec",Speed);
                    SetDlgItemText(hWnd,IDC_TAB2AS_Speed,strbuffer);
                }
                else
                    SetDlgItemText(hWnd,IDC_TAB2AS_Speed,"Measuring");
            }

            for(int z=0;z<BFASPEEDPAST-1;z++)
                P->Past[z] = P->Past[z+1];
            P->Past[BFASPEEDPAST-1] = P->Counter;
        }
        break;
    }
    return false;
case WM_COMMAND:
    {
        SearchThread *P = (SearchThread *)GetWindowLong(hWnd,GWL_USERDATA);

        switch(LOWORD(wParam))
        {
        case IDC_TAB2SS_Stop:
        case IDC_TAB2SA_Stop:

            if (P)
            {
                if (P->HitsSelection) {
                    if (P->Running) {
                        P->Running = false;
                        SetDlgItemText(hWnd,IDC_TAB2SA_Stop,"Continue Search");
                    }
                    else {
                        P->Running = true;
                        SetDlgItemText(hWnd,LOWORD(wParam),"Stop Search");
                    }
                }
                else
                    PostQuitMessage(IDOK);
            }
            else
                PostQuitMessage(IDOK);

            break;
        case IDC_TAB2SA_TraverseMagicLevels:
            {
                if (!P) break;

                if (P->TraverseMagicLevels)
                    P->TraverseMagicLevels = false;
                else
                    P->TraverseMagicLevels = true;

                CheckDlgButton(hWnd,LOWORD(wParam),P->TraverseMagicLevels);
            }
            break;
        case IDC_TAB2SA_HitSelection:
            {
                if (!P) break;

                if (P->HitsSelection) {
                    P->HitsSelection = false;
```

```
                    SetDlgItemText(hWnd,IDC_TAB2SA_Stop,"Stop Search");
                    EnableWindow(GetDlgItem(hWnd,IDC_TAB2SA_HitList),FALSE);
                    P->Running = true;
                }
                else {
                    P->HitsSelection = true;
                    SetDlgItemText(hWnd,IDC_TAB2SA_Stop,"Pause Search");
                    EnableWindow(GetDlgItem(hWnd,IDC_TAB2SA_HitList),TRUE);
                }

                CheckDlgButton(hWnd,LOWORD(wParam),P->HitsSelection);
            }
            break;
        case IDC_TAB2SA_HitList:
            if (HIWORD(wParam) == LBN_SELCHANGE && P)
            {
                P->Running = false;
                SetDlgItemText(hWnd,IDC_TAB2SA_Stop,"Continue Search");

                int ix = SendDlgItemMessage(hWnd,LOWORD(wParam),LB_GETCURSEL,0,0);
                SearchHit *H = (SearchHit*)SendDlgItemMessage(hWnd,LOWORD(wParam),LB_GETITEMDATA,ix,0);

                P->Item->SetMagicLevel(H->MagicLevel);
                P->Item->SetDWA(H->DWA);
                P->Item->SetDWB(H->DWB);
                P->Item->Decoded = false;
                UpdateTab2();
                InvalidateRect(hTabDialog,NULL,FALSE);
            }
            else if (HIWORD(wParam) == LBN_DBLCLK && P)
            {
                P->Running = false;
                SetDlgItemText(hWnd,IDC_TAB2SA_Stop,"Continue Search");

                int ix = SendDlgItemMessage(hWnd,LOWORD(wParam),LB_GETCURSEL,0,0);
                SearchHit *H = (SearchHit*)SendDlgItemMessage(hWnd,LOWORD(wParam),LB_GETITEMDATA,ix,0);

                P->Item->SetMagicLevel(H->MagicLevel);
                P->Item->SetDWA(H->DWA);
                P->Item->SetDWB(H->DWB);
                P->Item->Decoded = false;
                PostQuitMessage(IDOK);
            }
            break;
        case IDC_CHELP:
            ToggleHelpBox(hWnd,IDH_TAB2S);
            break;
        }
    }
    return false;
case WM_USER:
    {
        SearchThread *P = (SearchThread *)GetWindowLong(hWnd,GWL_USERDATA);
        if (!P) break;

        SendDlgItemMessage(hWnd,IDC_TAB2SA_HitList,LB_RESETCONTENT,0,0);
        for(SearchHit *H = P->Hits;H != 0;H = H->List)
        {
            char buffer[64];
            sprintf(buffer,"MG %i | DWB %8X",H->MagicLevel,H->DWB);
            int ix = SendDlgItemMessage(hWnd,IDC_TAB2SA_HitList,LB_ADDSTRING,0,(LPARAM)buffer);
            SendDlgItemMessage(hWnd,IDC_TAB2SA_HitList,LB_SETITEMDATA,ix,(LPARAM)H);
        }
    }
    return false;
case WM_HSCROLL:
    if ((HWND)lParam == GetDlgItem(hWnd,IDC_TAB2SA_Slider))
    {
        SearchThread *P = (SearchThread *)GetWindowLong(hWnd,GWL_USERDATA);
        int Prior = SendMessage((HWND)lParam,TBM_GETPOS,0,0);

        SetDlgItemText(hWnd,IDC_TAB2SA_Priority,Priorities[Prior].Text);
//          if (!SetPriorityClass(GetCurrentProcess(),Priorities[Prior].ClassID))
//              ErrorMessage();
            if (!SetThreadPriority(P->Thread,Priorities[Prior].PriorityID))
                ErrorMessage();
        }
        break;
    case WM_CLOSE:
```

```
        PostQuitMessage(IDOK);
        return false;
    case WM_DESTROY:
        return false;
    }
    return false;
}
```

```
// Tab3.cpp from D2E

#include "JamellaD2E.h"

char *groups[5][3] =
{
    { "Javelin and Spear Skills","Passive and Magic Skills","Box and Crossbow Skills" },
    { "Cold Spells","Lightning Spells","Fire Spells" },
    { "Summoning Spells","Posion and Bone Spells","Curses" },
    { "Defensive Auras","Offensive Auras","Combat Skills" },
    { "Warcries","Combat Masteries","Combat Skills" }
};
struct skill skills[5*30] =
{
// 0 = Amazon
    {  5,"Jab",                IDS_SKILL_A05,IDB_SKILL_A04 },
    {  9,"Power Strike",       IDS_SKILL_A09,IDB_SKILL_A09 },
    { 10,"Poison Javelin",     IDS_SKILL_A10,IDB_SKILL_A10 },
    { 14,"Impale",             IDS_SKILL_A14,IDB_SKILL_A14 },
    { 15,"Lightning Bolt",     IDS_SKILL_A15,IDB_SKILL_A15 },
    { 19,"Charged Bolt",       IDS_SKILL_A19,IDB_SKILL_A19 },
    { 20,"Plague Javelin",     IDS_SKILL_A20,IDB_SKILL_A20 },
    { 25,"Fend",               IDS_SKILL_A25,IDB_SKILL_A25 },
    { 29,"Lightning Strike",   IDS_SKILL_A29,IDB_SKILL_A29 },
    { 30,"Lightning Fury",     IDS_SKILL_A30,IDB_SKILL_A30 },

    {  3,"Inner Sight",        IDS_SKILL_A03,IDB_SKILL_A03 },
    {  4,"Critical Strike",    IDS_SKILL_A04,IDB_SKILL_A04 },
    {  8,"Dodge",              IDS_SKILL_A08,IDB_SKILL_A08 },
    { 12,"Slow Missiles",      IDS_SKILL_A12,IDB_SKILL_A12 },
    { 13,"Avoid",              IDS_SKILL_A13,IDB_SKILL_A13 },
    { 18,"Penetrate",          IDS_SKILL_A18,IDB_SKILL_A18 },
    { 23,"Decoy",              IDS_SKILL_A23,IDB_SKILL_A23 },
    { 24,"Evade",              IDS_SKILL_A24,IDB_SKILL_A24 },
    { 27,"Valkyrie",           IDS_SKILL_A27,IDB_SKILL_A27 },
    { 28,"Pierce",             IDS_SKILL_A28,IDB_SKILL_A28 },

    {  1,"Magic Arrow",        IDS_SKILL_A01,IDB_SKILL_A01 },
    {  2,"Fire Arrow",         IDS_SKILL_A02,IDB_SKILL_A02 },
    {  6,"Cold Arrow",         IDS_SKILL_A06,IDB_SKILL_A06 },
    {  7,"Multiple Shots",     IDS_SKILL_A07,IDB_SKILL_A07 },
    { 11,"Exploding Arrow",    IDS_SKILL_A11,IDB_SKILL_A11 },
    { 16,"Ice Arrow",          IDS_SKILL_A16,IDB_SKILL_A16 },
    { 17,"Guided Arrow",       IDS_SKILL_A17,IDB_SKILL_A17 },
    { 21,"Strafe",             IDS_SKILL_A21,IDB_SKILL_A21 },
    { 22,"Immolation Arrow",   IDS_SKILL_A22,IDB_SKILL_A22 },
    { 26,"Freezing Arrow",     IDS_SKILL_A26,IDB_SKILL_A26 },

// 1 = Sorceress
    {  4,"Ice Bolt",           IDS_SKILL_S04,IDB_SKILL_S04 },
    {  5,"Frozen Armor",       IDS_SKILL_S05,IDB_SKILL_S05 },
    {  9,"Frost Nova",         IDS_SKILL_S09,IDB_SKILL_S09 },
    { 10,"Ice Blast",          IDS_SKILL_S10,IDB_SKILL_S10 },
    { 15,"Shiver Armor",       IDS_SKILL_S15,IDB_SKILL_S15 },
    { 20,"Glacial Spike",      IDS_SKILL_S20,IDB_SKILL_S20 },
    { 24,"Blizzard",           IDS_SKILL_S24,IDB_SKILL_S24 },
    { 25,"Chilling Armor",     IDS_SKILL_S25,IDB_SKILL_S25 },
    { 29,"Frozen Orb",         IDS_SKILL_S29,IDB_SKILL_S29 },
    { 30,"Cold Mastery",       IDS_SKILL_S30,IDB_SKILL_S30 },

    {  3,"Charged Bolt",       IDS_SKILL_S03,IDB_SKILL_S03 },
    {  7,"Static Field",       IDS_SKILL_S07,IDB_SKILL_S07 },
    {  8,"Telekinesis",        IDS_SKILL_S08,IDB_SKILL_S08 },
    { 13,"Nova",               IDS_SKILL_S13,IDB_SKILL_S13 },
    { 14,"Lightning",          IDS_SKILL_S14,IDB_SKILL_S14 },
    { 18,"Chain Lightning",    IDS_SKILL_S18,IDB_SKILL_S18 },
    { 19,"Teleport",           IDS_SKILL_S19,IDB_SKILL_S19 },
    { 22,"Thunder Storm",      IDS_SKILL_S22,IDB_SKILL_S22 },
    { 23,"Energy Shield",      IDS_SKILL_S23,IDB_SKILL_S23 },
    { 28,"Lightning Mastery",  IDS_SKILL_S28,IDB_SKILL_S28 },

    {  1,"Fire Bolt",          IDS_SKILL_S01,IDB_SKILL_S01 },
    {  2,"Warmth",             IDS_SKILL_S02,IDB_SKILL_S02 },
    {  6,"Inferno",            IDS_SKILL_S06,IDB_SKILL_S06 },
    { 11,"Blaze",              IDS_SKILL_S11,IDB_SKILL_S11 },
    { 12,"Fire Ball",          IDS_SKILL_S12,IDB_SKILL_S12 },
    { 16,"Fire Wall",          IDS_SKILL_S16,IDB_SKILL_S16 },
    { 17,"Enchant",            IDS_SKILL_S17,IDB_SKILL_S17 },
    { 21,"Meteor",             IDS_SKILL_S21,IDB_SKILL_S21 },
```

```
    { 26,"Fire Mastery",        IDS_SKILL_S26,IDB_SKILL_S26 },
    { 27,"Hydra",               IDS_SKILL_S27,IDB_SKILL_S27 },

// 2 = Necromancer
    {  4,"Skeleton Mastery",    IDS_SKILL_N04,IDB_SKILL_N04 },
    {  5,"Raise Skeleton",      IDS_SKILL_N05,IDB_SKILL_N05 },
    { 10,"Clay Golem",          IDS_SKILL_N10,IDB_SKILL_N10 },
    { 14,"Golem Mastery",       IDS_SKILL_N14,IDB_SKILL_N14 },
    { 15,"Raise Skeletal Mage", IDS_SKILL_N15,IDB_SKILL_N15 },
    { 20,"Blood Golem",         IDS_SKILL_N20,IDB_SKILL_N20 },
    { 24,"Summon Resist",       IDS_SKILL_N24,IDB_SKILL_N24 },
    { 25,"Iron Golem",          IDS_SKILL_N25,IDB_SKILL_N25 },
    { 29,"Fire Golem",          IDS_SKILL_N29,IDB_SKILL_N29 },
    { 30,"Revive",              IDS_SKILL_N30,IDB_SKILL_N30 },

    {  2,"Teeth",               IDS_SKILL_N02,IDB_SKILL_N02 },
    {  3,"Bone Armor",          IDS_SKILL_N03,IDB_SKILL_N03 },
    {  8,"Poison Dagger",       IDS_SKILL_N08,IDB_SKILL_N08 },
    {  9,"Corpse Explosion",    IDS_SKILL_N09,IDB_SKILL_N09 },
    { 13,"Bone Wall",           IDS_SKILL_N13,IDB_SKILL_N13 },
    { 18,"Poison Explosion",    IDS_SKILL_N18,IDB_SKILL_N18 },
    { 19,"Bone Spear",          IDS_SKILL_N19,IDB_SKILL_N19 },
    { 23,"Bone Prison",         IDS_SKILL_N23,IDB_SKILL_N23 },
    { 27,"Poison Nova",         IDS_SKILL_N27,IDB_SKILL_N27 },
    { 28,"Bone Spirit",         IDS_SKILL_N28,IDB_SKILL_N28 },

    {  1,"Amplify Damage",      IDS_SKILL_N01,IDB_SKILL_N01 },
    {  6,"Dim Vision",          IDS_SKILL_N06,IDB_SKILL_N06 },
    {  7,"Weaken",              IDS_SKILL_N07,IDB_SKILL_N07 },
    { 11,"Iron Maiden",         IDS_SKILL_N11,IDB_SKILL_N11 },
    { 12,"Terror",              IDS_SKILL_N12,IDB_SKILL_N12 },
    { 16,"Confuse",             IDS_SKILL_N16,IDB_SKILL_N16 },
    { 17,"Life Tap",            IDS_SKILL_N17,IDB_SKILL_N17 },
    { 21,"Attract",             IDS_SKILL_N21,IDB_SKILL_N21 },
    { 22,"Decrepify",           IDS_SKILL_N22,IDB_SKILL_N22 },
    { 26,"Lower Resist",        IDS_SKILL_N26,IDB_SKILL_N26 },

// 3 = Paladin
    {  4,"Prayer",              IDS_SKILL_P04,IDB_SKILL_P04 },
    {  5,"Resist Fire",         IDS_SKILL_P05,IDB_SKILL_P05 },
    {  9,"Defiance",            IDS_SKILL_P09,IDB_SKILL_P09 },
    { 10,"Resist Cold",         IDS_SKILL_P10,IDB_SKILL_P10 },
    { 14,"Cleansing",           IDS_SKILL_P14,IDB_SKILL_P14 },
    { 15,"Resist Lightning",    IDS_SKILL_P15,IDB_SKILL_P15 },
    { 20,"Vigor",               IDS_SKILL_P20,IDB_SKILL_P20 },
    { 25,"Meditation",          IDS_SKILL_P25,IDB_SKILL_P25 },
    { 29,"Redemption",          IDS_SKILL_P29,IDB_SKILL_P29 },
    { 30,"Salvation",           IDS_SKILL_P30,IDB_SKILL_P30 },

    {  3,"Might",               IDS_SKILL_P03,IDB_SKILL_P03 },
    {  7,"Holy Fire",           IDS_SKILL_P07,IDB_SKILL_P07 },
    {  8,"Thorns",              IDS_SKILL_P08,IDB_SKILL_P08 },
    { 13,"Blessed Armor",       IDS_SKILL_P13,IDB_SKILL_P13 },
    { 18,"Concentration",       IDS_SKILL_P18,IDB_SKILL_P18 },
    { 19,"Holy Freeze",         IDS_SKILL_P19,IDB_SKILL_P19 },
    { 23,"Holy Shock",          IDS_SKILL_P23,IDB_SKILL_P23 },
    { 24,"Sanctuary",           IDS_SKILL_P24,IDB_SKILL_P24 },
    { 27,"Fanaticism",          IDS_SKILL_P27,IDB_SKILL_P27 },
    { 28,"Conviction",          IDS_SKILL_P28,IDB_SKILL_P28 },

    {  1,"Sacrifice",           IDS_SKILL_P01,IDB_SKILL_P01 },
    {  2,"Smite",               IDS_SKILL_P02,IDB_SKILL_P02 },
    {  6,"Holy Bolt",           IDS_SKILL_P06,IDB_SKILL_P06 },
    { 11,"Zeal",                IDS_SKILL_P11,IDB_SKILL_P11 },
    { 12,"Charge",              IDS_SKILL_P12,IDB_SKILL_P12 },
    { 16,"Vengeance",           IDS_SKILL_P16,IDB_SKILL_P16 },
    { 17,"Blessed Hammer",      IDS_SKILL_P17,IDB_SKILL_P17 },
    { 21,"Conversion",          IDS_SKILL_P21,IDB_SKILL_P21 },
    { 22,"Holy Shield",         IDS_SKILL_P22,IDB_SKILL_P22 },
    { 26,"Fist of Heavens",     IDS_SKILL_P26,IDB_SKILL_P26 },

// 4 = Barbarian
    {  5,"Howl",                IDS_SKILL_B05,IDB_SKILL_B05 },
    {  6,"Find Potion",         IDS_SKILL_B06,IDB_SKILL_B06 },
    { 12,"Taunt",               IDS_SKILL_B12,IDB_SKILL_B12 },
    { 13,"Shout",               IDS_SKILL_B13,IDB_SKILL_B13 },
    { 17,"Find Item",           IDS_SKILL_B17,IDB_SKILL_B17 },
    { 21,"Battle Cry",          IDS_SKILL_B21,IDB_SKILL_B21 },
    { 24,"Battle Orders",       IDS_SKILL_B24,IDB_SKILL_B24 },
```

```
    { 25,"Grim Wand",           IDS_SKILL_B25,IDB_SKILL_B25 },
    { 29,"War Cry",             IDS_SKILL_B29,IDB_SKILL_B29 },
    { 30,"Battle Command",      IDS_SKILL_B30,IDB_SKILL_B30 },

    {  2,"Sword Mastery",       IDS_SKILL_B02,IDB_SKILL_B02 },
    {  3,"Axe Mastery",         IDS_SKILL_B03,IDB_SKILL_B03 },
    {  4,"Mace Mastery",        IDS_SKILL_B04,IDB_SKILL_B04 },
    {  9,"Pole Mastery",        IDS_SKILL_B09,IDB_SKILL_B09 },
    { 10,"Throwing Mastery",    IDS_SKILL_B10,IDB_SKILL_B10 },
    { 11,"Speare Mastery",      IDS_SKILL_B11,IDB_SKILL_B11 },
    { 16,"Increase Stamina",    IDS_SKILL_B16,IDB_SKILL_B16 },
    { 20,"Iron Skin",           IDS_SKILL_B20,IDB_SKILL_B20 },
    { 23,"Increase Speed",      IDS_SKILL_B23,IDB_SKILL_B23 },
    { 28,"Natural Resistances", IDS_SKILL_P28,IDB_SKILL_B28 },

    {  1,"Bash",                IDS_SKILL_B01,IDB_SKILL_B01 },
    {  7,"Leap",                IDS_SKILL_B07,IDB_SKILL_B07 },
    {  8,"Double Swing",        IDS_SKILL_B08,IDB_SKILL_B08 },
    { 14,"Stun",                IDS_SKILL_B14,IDB_SKILL_B14 },
    { 15,"Double Throw",        IDS_SKILL_B15,IDB_SKILL_B15 },
    { 18,"Leap Attack",         IDS_SKILL_B18,IDB_SKILL_B18 },
    { 19,"Concentrate",         IDS_SKILL_B19,IDB_SKILL_B19 },
    { 22,"Frenzy",              IDS_SKILL_B22,IDB_SKILL_B22 },
    { 26,"Whirlwind",           IDS_SKILL_B26,IDB_SKILL_B26 },
    { 27,"Berserk",             IDS_SKILL_B27,IDB_SKILL_B27 }
};
struct skillprops
{
    int     idbitmap;
    int     idedit;
};
struct skillprops props[30] =
{
    { IDC_TAB3_Bitmap_A0,IDC_TAB3_Edit_A0 },
    { IDC_TAB3_Bitmap_A1,IDC_TAB3_Edit_A1 },
    { IDC_TAB3_Bitmap_A2,IDC_TAB3_Edit_A2 },
    { IDC_TAB3_Bitmap_A3,IDC_TAB3_Edit_A3 },
    { IDC_TAB3_Bitmap_A4,IDC_TAB3_Edit_A4 },
    { IDC_TAB3_Bitmap_A5,IDC_TAB3_Edit_A5 },
    { IDC_TAB3_Bitmap_A6,IDC_TAB3_Edit_A6 },
    { IDC_TAB3_Bitmap_A7,IDC_TAB3_Edit_A7 },
    { IDC_TAB3_Bitmap_A8,IDC_TAB3_Edit_A8 },
    { IDC_TAB3_Bitmap_A9,IDC_TAB3_Edit_A9 },
    { IDC_TAB3_Bitmap_B0,IDC_TAB3_Edit_B0 },
    { IDC_TAB3_Bitmap_B1,IDC_TAB3_Edit_B1 },
    { IDC_TAB3_Bitmap_B2,IDC_TAB3_Edit_B2 },
    { IDC_TAB3_Bitmap_B3,IDC_TAB3_Edit_B3 },
    { IDC_TAB3_Bitmap_B4,IDC_TAB3_Edit_B4 },
    { IDC_TAB3_Bitmap_B5,IDC_TAB3_Edit_B5 },
    { IDC_TAB3_Bitmap_B6,IDC_TAB3_Edit_B6 },
    { IDC_TAB3_Bitmap_B7,IDC_TAB3_Edit_B7 },
    { IDC_TAB3_Bitmap_B8,IDC_TAB3_Edit_B8 },
    { IDC_TAB3_Bitmap_B9,IDC_TAB3_Edit_B9 },
    { IDC_TAB3_Bitmap_C0,IDC_TAB3_Edit_C0 },
    { IDC_TAB3_Bitmap_C1,IDC_TAB3_Edit_C1 },
    { IDC_TAB3_Bitmap_C2,IDC_TAB3_Edit_C2 },
    { IDC_TAB3_Bitmap_C3,IDC_TAB3_Edit_C3 },
    { IDC_TAB3_Bitmap_C4,IDC_TAB3_Edit_C4 },
    { IDC_TAB3_Bitmap_C5,IDC_TAB3_Edit_C5 },
    { IDC_TAB3_Bitmap_C6,IDC_TAB3_Edit_C6 },
    { IDC_TAB3_Bitmap_C7,IDC_TAB3_Edit_C7 },
    { IDC_TAB3_Bitmap_C8,IDC_TAB3_Edit_C8 },
    { IDC_TAB3_Bitmap_C9,IDC_TAB3_Edit_C9 }
};

int     selection;
skill*  classskills;
HWND    hTrack;
char    skilleffect[256];

LRESULT CALLBACK Tab3DialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
    switch(uMsg)
    {
    case WM_INITDIALOG:
        {
            SendDlgItemMessage(hWnd,IDC_TAB3_BonusPlus,STM_SETIMAGE,IMAGE_BITMAP,(LPARAM) hBmpPlus);

            // Setup Track Bar
```

```
            hTrack = GetDlgItem(hWnd,IDC_TAB3_Track);
            SendMessage(hTrack,TBM_SETRANGE,
                FALSE,(LPARAM) MAKELONG(0,20));
            SendMessage(hTrack,TBM_SETTICFREQ,(LPARAM) 1,0);
            SendMessage(hTrack,TBM_SETPAGESIZE,FALSE,(LPARAM) 1);
            SendMessage(hTrack,TBM_SETPOS,TRUE,(LPARAM) 0);

            // Insert right bitmaps and show skill levels
            for(int i=0;i<3;i++)
            {
                SetDlgItemText(hWnd,IDC_TAB3_Frame_A+i,groups[fc.Header.playerclass][i]);
            }
            classskills = &skills[fc.Header.playerclass * 30];
            for(i=0;i<30;i++)
            {
                SendDlgItemMessage(hWnd,props[i].idbitmap,STM_SETIMAGE,IMAGE_BITMAP,(LPARAM) classskills[i].hbitmap);
                SetDlgItemInt(hWnd,props[i].idedit, fc.IF.skills[classskills[i].ifoffset-1],FALSE);
            }

            selection = -1;

            // Setup user variables
            SetDlgItemInt(hWnd,IDC_TAB3_Bonus,fc.gf.skillbonus,FALSE);

            hToolTip = CreateToolTipCtrl(hWnd,IDD_TAB3,true);
        }
        return true;
    case WM_COMMAND:
        {
            switch(LOWORD(wParam))
            {
            case IDC_TAB3_Bitmap_A0:            case IDC_TAB3_Bitmap_A1:
            case IDC_TAB3_Bitmap_A2:            case IDC_TAB3_Bitmap_A3:
            case IDC_TAB3_Bitmap_A4:            case IDC_TAB3_Bitmap_A5:
            case IDC_TAB3_Bitmap_A6:            case IDC_TAB3_Bitmap_A7:
            case IDC_TAB3_Bitmap_A8:            case IDC_TAB3_Bitmap_A9:
            case IDC_TAB3_Bitmap_B0:            case IDC_TAB3_Bitmap_B1:
            case IDC_TAB3_Bitmap_B2:            case IDC_TAB3_Bitmap_B3:
            case IDC_TAB3_Bitmap_B4:            case IDC_TAB3_Bitmap_B5:
            case IDC_TAB3_Bitmap_B6:            case IDC_TAB3_Bitmap_B7:
            case IDC_TAB3_Bitmap_B8:            case IDC_TAB3_Bitmap_B9:
            case IDC_TAB3_Bitmap_C0:            case IDC_TAB3_Bitmap_C1:
            case IDC_TAB3_Bitmap_C2:            case IDC_TAB3_Bitmap_C3:
            case IDC_TAB3_Bitmap_C4:            case IDC_TAB3_Bitmap_C5:
            case IDC_TAB3_Bitmap_C6:            case IDC_TAB3_Bitmap_C7:
            case IDC_TAB3_Bitmap_C8:            case IDC_TAB3_Bitmap_C9:
                {
                    HWND hBmp = GetDlgItem(hWnd,IDC_TAB3_Prop_Bitmap);
                    HWND hEdit = GetDlgItem(hWnd,IDC_TAB3_Prop_Edit);
                    HWND hEffect = GetDlgItem(hWnd,IDC_TAB3_Prop_Effect);

                    if (selection < 0)
                    {
                        ShowWindow(hBmp,SW_SHOW);
                        ShowWindow(hEdit,SW_SHOW);
                        ShowWindow(hEffect,SW_SHOW);
                        ShowWindow(hTrack,SW_SHOW);
                    }

                    if (LOWORD(wParam) - IDC_TAB3_Bitmap_A0 == selection)
                    {
                        fc.IF.skills[classskills[selection].ifoffset-1]++;

                        if (fc.IF.skills[classskills[selection].ifoffset-1] > 20)
                            fc.IF.skills[classskills[selection].ifoffset-1] = 20;

                        SetDlgItemInt(hWnd,props[selection].idedit,fc.IF.skills[classskills[selection].ifoffset-1],FALSE
);
                        SendMessage(hTrack,TBM_SETPOS,TRUE,(LPARAM) fc.IF.skills[classskills[selection].ifoffset-1]);
                    }
                    else
                    {
                        selection = LOWORD(wParam) - IDC_TAB3_Bitmap_A0;

                        SendMessage(hBmp,STM_SETIMAGE,IMAGE_BITMAP,(LPARAM) classskills[selection].hbitmap);
                        SendMessage(hEdit,WM_SETTEXT,0,(LPARAM) classskills[selection].name);
                        SendMessage(hTrack,TBM_SETPOS,TRUE,(LPARAM) fc.IF.skills[classskills[selection].ifoffset-1]);
                        LoadString(hInstance,classskills[selection].idstring,skilleffect,256);
                        SendMessage(hEffect,WM_SETTEXT,0,(LPARAM) skilleffect);
```

```
                }
            }
            break;
        case IDC_TAB3_BonusPlus:
            {
                DWORD x = GetDlgItemInt(hWnd,IDC_TAB3_Bonus,NULL,FALSE);
                SetDlgItemInt(hWnd,IDC_TAB3_Bonus,x+1,FALSE);
            }
            break;
        case IDR_TAB3_MaximizeAll:
            {
                for(int i=0;i<30;i++)
                {
                    fc.IF.skills[i] = 20;
                    SetDlgItemInt(hWnd,props[i].idedit,20,FALSE);
                    if (selection > 0)
                    {
                        SendMessage(hTrack,TBM_SETPOS,TRUE,(LPARAM) fc.IF.skills[classskills[selection].ifoffset-1])
;
                    }
                }
            }
            break;
        case IDR_TAB3_SetAll0:
        case IDR_TAB3_SetAll4:
        case IDR_TAB3_SetAll8:
        case IDR_TAB3_SetAll12:
        case IDR_TAB3_SetAll16:
        case IDR_TAB3_SetAll20:
            {
                for(int i=0;i<30;i++)
                {
                    fc.IF.skills[i] =
                        (LOWORD(wParam) == IDR_TAB3_SetAll0) ? 0 :
                        (LOWORD(wParam) == IDR_TAB3_SetAll4) ? 4 :
                        (LOWORD(wParam) == IDR_TAB3_SetAll8) ? 8 :
                        (LOWORD(wParam) == IDR_TAB3_SetAll12) ? 12 :
                        (LOWORD(wParam) == IDR_TAB3_SetAll16) ? 16 :
                        (LOWORD(wParam) == IDR_TAB3_SetAll20) ? 20 : 0;

                    SetDlgItemInt(hWnd,props[i].idedit,fc.IF.skills[i],FALSE);
                    if (selection > 0) {
                        SendMessage(hTrack,TBM_SETPOS,TRUE,(LPARAM) fc.IF.skills[classskills[selection].ifoffset-1])
;
                    }
                }

            }
            break;
        case IDC_TAB3_Batch:
            switch(HIWORD(wParam))
            {
            case BN_CLICKED:
                {
                    HMENU hMenu = GetSubMenu(hBatchMenu,2);

                    POINT Pos;
                    GetCursorPos(&Pos);

                    TrackPopupMenu(hMenu, TPM_LEFTALIGN | TPM_LEFTBUTTON,
                        Pos.x, Pos.y, 0, hWnd, NULL);
                }
                break;
            }
        break;
        }
    }
    break;
case WM_HSCROLL:
    if ((HWND)lParam == hTrack)
    {
        fc.IF.skills[classskills[selection].ifoffset-1] = (BYTE)SendMessage(hTrack,TBM_GETPOS,0,0);
        SetDlgItemInt(hWnd,props[selection].idedit,fc.IF.skills[classskills[selection].ifoffset-1],FALSE);
        break;
    }
    break;
case WM_DESTROY:
    {
        // Retrieve user variables
```

```
            fc.gf.skillbonus = GetDlgItemInt(hWnd,IDC_TAB3_Bonus,NULL,FALSE);

            DestroyWindow(hToolTip);
        }
        return false;
    }
    return false;
}
```

```
// Tab4.cpp from D2E

#include "JamellaD2E.h"

struct queststatus actA1[] =
{
    { "Not started", 0x0000 },
    { "Look for the Den", 0x0004 },
    { "Get reward from Akara", 0x0004 },
    { "Quest just completed", 0x901D },
    { "Quest completed", 0x001D },
    { 0,0 }
};
struct queststatus actA2[] =
{
    { "Not started", 0x0000 },
    { "Look for Blood Raven", 0x0004 },
    { "Kill Blood Raven", 0x001C },
    { "Get reward from Akara", 0x001E },
    { "Quest just completed", 0x901D },
    { "Quest completed", 0x001D },
    { 0,0 }
};
struct queststatus actA3[] =
{
    { "Not started", 0x0000 },
    { "Look for the Scroll", 0x0004 },
    { "Get reward from Akara", 0x000E },
    { "Quest just completed", 0x900D },
    { "Killed Cow King!", 0x0114 },
    { "Quest completed", 0x001D },
    { 0,0 }
};
struct queststatus actA4[] =
{
    { "Not started", 0x0000 },
    { "Look for Forgotten Tower", 0x0004 },
    { "Explore the cellar dungeons", 0x0044 },
    { "Dispose of evil Countess", 0x0054 },
    { "Quest just completed", 0x9055 },
    { "Quest completed", 0x001D },
    { 0,0 }
};
struct queststatus actA5[] =
{
    { "Not started", 0x0000 },
    { "Look for ...", 0x0004 },
    { "Charsi will Imbue Item", 0x004E },
    { "Just finished", 0x001D },
    { "Long finished", 0x0001 },
    { 0,0 }
};

struct queststatus simple[] =
{
    { "Not started", 0x0000 },
    { "Look for ...", 0x0004 },
    { "Just finished", 0x001D },
    { "Long finished", 0x0001 },
    { 0,0 }
};

struct queststatus allcombo[] =
{
    { "to not started", 0x0000 },
    { "to look for ...", 0x0004 },
    { "to just finished", 0x001D },
    { "to long finished", 0x0001 },
    { 0,0 }
};

struct quest quests[21] =
{
    { IDS_QUEST11,IDB_QUEST11,actA1, 1 },
    { IDS_QUEST12,IDB_QUEST12,actA2, 2 },
    { IDS_QUEST13,IDB_QUEST13,actA3, 4 },
    { IDS_QUEST14,IDB_QUEST14,actA4, 5 },
    { IDS_QUEST15,IDB_QUEST15,actA5, 3 },
    { IDS_QUEST16,IDB_QUEST16,simple, 6 },
```

```
    { IDS_QUEST21,IDB_QUEST21,simple, 9 },
    { IDS_QUEST22,IDB_QUEST22,simple,10 },
    { IDS_QUEST23,IDB_QUEST23,simple,11 },
    { IDS_QUEST24,IDB_QUEST24,simple,12 },
    { IDS_QUEST25,IDB_QUEST25,simple,13 },
    { IDS_QUEST26,IDB_QUEST26,simple,14 },

    { IDS_QUEST31,IDB_QUEST31,simple,20 },
    { IDS_QUEST32,IDB_QUEST32,simple,19 },
    { IDS_QUEST33,IDB_QUEST33,simple,18 },
    { IDS_QUEST34,IDB_QUEST34,simple,17 },
    { IDS_QUEST35,IDB_QUEST35,simple,21 },
    { IDS_QUEST36,IDB_QUEST36,simple,22 },

    { IDS_QUEST41,IDB_QUEST41,simple,25 },
    { IDS_QUEST42,IDB_QUEST42,simple,27 },
    { IDS_QUEST43,IDB_QUEST43,simple,26 }
};
const int questsn = sizeof quests / sizeof quests[0];

int actsoffsets[4] =
{ 0, 7, 15, 23 };

struct
{
    int         text;
    int         bitmap;
    int         combo;
    HWND        hcombo;
}
interfaces[6] =
{
    { IDC_TAB4_Text1,IDC_TAB4_Bmp1,IDC_TAB4_Sel1 },
    { IDC_TAB4_Text2,IDC_TAB4_Bmp2,IDC_TAB4_Sel2 },
    { IDC_TAB4_Text3,IDC_TAB4_Bmp3,IDC_TAB4_Sel3 },
    { IDC_TAB4_Text4,IDC_TAB4_Bmp4,IDC_TAB4_Sel4 },
    { IDC_TAB4_Text5,IDC_TAB4_Bmp5,IDC_TAB4_Sel5 },
    { IDC_TAB4_Text6,IDC_TAB4_Bmp6,IDC_TAB4_Sel6 }
};

int     act;
WORD*   ptr;

void updatequests(HWND hWnd)
{
    int x = act * 6;
    for(int id=0;id < 6;id++)
    {
        if (x+id < questsn)
        {
            struct quest *quest = &quests[x+id];
            WORD qstat = ptr[quest->offset];

            SetDlgItemText(hWnd,interfaces[id].text,quest->string);
            SendDlgItemMessage(hWnd,interfaces[id].bitmap,
                STM_SETIMAGE,IMAGE_BITMAP,(LPARAM) quest->hbitmap);

            EnableWindow(interfaces[id].hcombo,TRUE);
            SendMessage(interfaces[id].hcombo,CB_RESETCONTENT,0,0);
            bool addstatus = true;
            for (int i=0;quest->stati[i].text;i++)
            {
                SendMessage(interfaces[id].hcombo,
                    CB_ADDSTRING,0,(LPARAM) quest->stati[i].text);

                if ((qstat & 0xFF) != (quest->stati[i].value & 0xFF)) continue;

                if (quest->stati[i].value & 0xFF00)
                {
                    if (qstat!= quest->stati[i].value) continue;
                }
                SendMessage(interfaces[id].hcombo,CB_SETCURSEL,i,0);
                addstatus = false;
            }
            if (addstatus)
            {
                sprintf(buffer,"Status %4X",qstat);
```

```
                SendMessage(interfaces[id].hcombo,
                    CB_ADDSTRING,0,(LPARAM) buffer);
                SendMessage(interfaces[id].hcombo,CB_SETCURSEL,i,0);
            }
        }
        else
        {
            SetDlgItemText(hWnd,interfaces[id].text,"");
            SendDlgItemMessage(hWnd,interfaces[id].bitmap,
                STM_SETIMAGE,IMAGE_BITMAP,(LPARAM) 0);

            EnableWindow(interfaces[id].hcombo,FALSE);
            SendMessage(interfaces[id].hcombo,CB_RESETCONTENT,0,0);
        }
    }
}
void updateacts(HWND hWnd)
{
    // Set Check Buttons
    for(int z=0;z<3;z++)
    {
        CheckDlgButton(hWnd,IDC_TAB4_Act2_On+z,
            ptr[actsoffsets[z+1]] == 0x0000 ? BST_UNCHECKED : BST_CHECKED);
        EnableWindow(GetDlgItem(hWnd,IDC_TAB4_Act2+z),!(ptr[actsoffsets[z+1]] == 0x0000));
    }
    updatequests(hWnd);
}

LRESULT CALLBACK Tab4DialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
    switch(uMsg)
    {
    case WM_INITDIALOG:
        {
            for(int x = 0;x < 6;x++)
            {
                interfaces[x].hcombo = GetDlgItem(hWnd,interfaces[x].combo);
            }

            // Select default levels
            CheckRadioButton(hWnd,IDC_TAB4_Diff1,IDC_TAB4_Diff3,
                IDC_TAB4_Diff1);
            CheckRadioButton(hWnd,IDC_TAB4_Act1,IDC_TAB4_Act4,
                IDC_TAB4_Act1);

            act = 0;
            ptr = fc.Woo.quests1;
            updateacts(hWnd);

            // Fill all combo
            SendDlgItemMessage(hWnd,IDC_TAB4_SelAll,CB_RESETCONTENT,0,0);
            for (int i=0;allcombo[i].text;i++)
            {
                SendDlgItemMessage(hWnd,IDC_TAB4_SelAll,
                    CB_ADDSTRING,0,(LPARAM) allcombo[i].text);
            }
            SendDlgItemMessage(hWnd,IDC_TAB4_SelAll,CB_SETCURSEL,0,0);

            hToolTip = CreateToolTipCtrl(hWnd,IDD_TAB4,true);
        }
        return true;
    case WM_COMMAND:
        {
            if (LOWORD(wParam) == IDC_TAB4_Diff1 && HIWORD(wParam) == BN_CLICKED)
            {
                if (IsDlgButtonChecked(hWnd,IDC_TAB4_Diff1))
                {
                    ptr = fc.Woo.quests1;
                    updateacts(hWnd);
                }
            }
            else if (LOWORD(wParam) == IDC_TAB4_Diff2 && HIWORD(wParam) == BN_CLICKED)
            {
                if (IsDlgButtonChecked(hWnd,IDC_TAB4_Diff2))
                {
                    ptr = fc.Woo.quests2;
                    updateacts(hWnd);
                }
            }
```

```
else if (LOWORD(wParam) == IDC_TAB4_Diff3 && HIWORD(wParam) == BN_CLICKED)
{
    if (IsDlgButtonChecked(hWnd,IDC_TAB4_Diff3))
    {
        ptr = fc.Woo.quests3;
        updateacts(hWnd);
    }
}
if (LOWORD(wParam) == IDC_TAB4_Act1 && HIWORD(wParam) == BN_CLICKED)
{
    if (IsDlgButtonChecked(hWnd,IDC_TAB4_Act1))
    {
        act = 0;
        updatequests(hWnd);
    }
}
else if (LOWORD(wParam) == IDC_TAB4_Act2 && HIWORD(wParam) == BN_CLICKED)
{
    if (IsDlgButtonChecked(hWnd,IDC_TAB4_Act2))
    {
        act = 1;
        updatequests(hWnd);
    }
}
else if (LOWORD(wParam) == IDC_TAB4_Act3 && HIWORD(wParam) == BN_CLICKED)
{
    if (IsDlgButtonChecked(hWnd,IDC_TAB4_Act3))
    {
        act = 2;
        updatequests(hWnd);
    }
}
else if (LOWORD(wParam) == IDC_TAB4_Act4 && HIWORD(wParam) == BN_CLICKED)
{
    if (IsDlgButtonChecked(hWnd,IDC_TAB4_Act4))
    {
        act = 3;
        updatequests(hWnd);
    }
}
else if (LOWORD(wParam) == IDC_TAB4_Act2_On && HIWORD(wParam) == BN_CLICKED)
{
    if (IsDlgButtonChecked(hWnd,IDC_TAB4_Act2_On))
    {
        ptr[actsoffsets[1]] = 0x0001;
        updateacts(hWnd);
    }
    else
    {
        ptr[actsoffsets[1]] = 0x0000;
        updateacts(hWnd);
    }
}
else if (LOWORD(wParam) == IDC_TAB4_Act3_On && HIWORD(wParam) == BN_CLICKED)
{
    if (IsDlgButtonChecked(hWnd,IDC_TAB4_Act3_On))
    {
        ptr[actsoffsets[2]] = 0x0001;
        updateacts(hWnd);
    }
    else
    {
        ptr[actsoffsets[2]] = 0x0000;
        updateacts(hWnd);
    }
}
else if (LOWORD(wParam) == IDC_TAB4_Act4_On && HIWORD(wParam) == BN_CLICKED)
{
    if (IsDlgButtonChecked(hWnd,IDC_TAB4_Act4_On))
    {
        ptr[actsoffsets[3]] = 0x0001;
        updateacts(hWnd);
    }
    else
    {
        ptr[actsoffsets[3]] = 0x0000;
        updateacts(hWnd);
    }
}
```

```
            else if (HIWORD(wParam) == CBN_SELCHANGE)
            {
                for (int i=0;i<6;i++)
                {
                    if (LOWORD(wParam) == interfaces[i].combo)
                    {
                        int sel = SendMessage(interfaces[i].hcombo,CB_GETCURSEL,0,0);

                        struct quest *quest = &quests[act*6 + i];

                        if (quest->stati[sel].text)
                        {
                            ptr[quest->offset] = quest->stati[sel].value;
                        }
                        updatequests(hWnd);
                    }
                }
            }
            else if (LOWORD(wParam) == IDC_TAB4_SetAll1 && HIWORD(wParam) == BN_CLICKED)
            {
                int sel = SendDlgItemMessage(hWnd,IDC_TAB4_SelAll,CB_GETCURSEL,0,0);
                for (int i=0;i<6;i++)
                {
                    if (act*6 + i < questsn)
                    {
                        struct quest *quest = &quests[act*6 + i];
                        ptr[quest->offset] = allcombo[sel].value;
                    }
                }
                updatequests(hWnd);
            }
            else if (LOWORD(wParam) == IDC_TAB4_SetAll2 && HIWORD(wParam) == BN_CLICKED)
            {
                int sel = SendDlgItemMessage(hWnd,IDC_TAB4_SelAll,CB_GETCURSEL,0,0);
                for (int i=0;i<questsn;i++)
                {
                    struct quest *quest = &quests[i];
                    ptr[quest->offset] = allcombo[sel].value;
                }
                for(i=0;i<3;i++)
                {
                    ptr[actsoffsets[i+1]] = 0x0001;
                }
                updateacts(hWnd);
            }
            else if (LOWORD(wParam) == IDC_TAB4_CowLevel && HIWORD(wParam) == BN_CLICKED)
            {
                DialogBox(hInstance,MAKEINTRESOURCE(IDD_COWLEVEL),hWnd,(DLGPROC) CowLevelDialogProc);
            }
        }
        break;
    case WM_DESTROY:
        {
            DestroyWindow(hToolTip);
        }
        return false;
    }
    return false;
}
```

```
// Tab5.cpp from D2E

#include "JamellaD2E.h"

DWORD *waypoints;
HBITMAP hBmpWaypointOn;
HBITMAP hBmpWaypointOff;

void inline updatewaypoints(HWND hWnd)
{
    for(int id=0;id <= IDC_TAB5_Way33-IDC_TAB5_Way01;id++)
    {
        SendDlgItemMessage(hWnd,IDC_TAB5_Way01+id,STM_SETIMAGE,IMAGE_BITMAP,
            (*waypoints & (1 << id)) ? (LPARAM)hBmpWaypointOn : (LPARAM)hBmpWaypointOff);
    }
}

LRESULT CALLBACK Tab5DialogProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
{
    switch(uMsg)
    {
    case WM_INITDIALOG:
        {
            // Select default difficulty level
            CheckRadioButton(hWnd,IDC_TAB5_Diff1,IDC_TAB5_Diff3,IDC_TAB5_Diff1);

            waypoints = &fc.WS.waypoints1;
            updatewaypoints(hWnd);

            hToolTip = CreateToolTipCtrl(hWnd,IDD_TAB5,true);
        }
        return true;
    case WM_COMMAND:
        {
            if (LOWORD(wParam) == IDC_TAB5_Diff1 && HIWORD(wParam) == BN_CLICKED)
            {
                if (IsDlgButtonChecked(hWnd,IDC_TAB5_Diff1))
                {
                    waypoints = &fc.WS.waypoints1;
                    updatewaypoints(hWnd);
                }
            }
            else if (LOWORD(wParam) == IDC_TAB5_Diff2 && HIWORD(wParam) == BN_CLICKED)
            {
                if (IsDlgButtonChecked(hWnd,IDC_TAB5_Diff2))
                {
                    waypoints = &fc.WS.waypoints2;
                    updatewaypoints(hWnd);
                }
            }
            else if (LOWORD(wParam) == IDC_TAB5_Diff3 && HIWORD(wParam) == BN_CLICKED)
            {
                if (IsDlgButtonChecked(hWnd,IDC_TAB5_Diff3))
                {
                    waypoints = &fc.WS.waypoints3;
                    updatewaypoints(hWnd);
                }
            }
            else if (LOWORD(wParam) >= IDC_TAB5_Way01 && LOWORD(wParam) <= IDC_TAB5_Way33 &&
                (HIWORD(wParam) == BN_CLICKED || HIWORD(wParam) == BN_DBLCLK))
            {
                DWORD o = 1 << (LOWORD(wParam) - IDC_TAB5_Way01) ;

                if (*waypoints & o)
                    *waypoints &= o ^ 0xFFFFFFFF;
                else
                    *waypoints |= o;
                updatewaypoints(hWnd);
            }
            else if (LOWORD(wParam) == IDC_TAB5_Batch && HIWORD(wParam) == BN_CLICKED)
            {
                HMENU hMenu = GetSubMenu(hBatchMenu,4);

                POINT Pos;
                GetCursorPos(&Pos);

                TrackPopupMenu(hMenu, TPM_LEFTALIGN | TPM_LEFTBUTTON,
                    Pos.x, Pos.y, 0, hWnd, NULL);
            }
```

```
            else if (LOWORD(wParam) == IDR_TAB5_ActivateHereAll)
            {
                *waypoints = 0x3FFFFFFF;
                updatewaypoints(hWnd);
            }
            else if (LOWORD(wParam) == IDR_TAB5_ActivateAllAll)
            {
                fc.WS.waypoints1 = 0x3FFFFFFF;
                fc.WS.waypoints2 = 0x3FFFFFFF;
                fc.WS.waypoints3 = 0x3FFFFFFF;
                updatewaypoints(hWnd);
            }
            else if (LOWORD(wParam) == IDR_TAB5_DeactivateHereAll)
            {
                *waypoints = 0x0;
                updatewaypoints(hWnd);
            }
            else if (LOWORD(wParam) == IDR_TAB5_DeactivateAllAll)
            {
                fc.WS.waypoints1 = 0x0;
                fc.WS.waypoints2 = 0x0;
                fc.WS.waypoints3 = 0x0;
                updatewaypoints(hWnd);
            }
        }
        break;
    case WM_DESTROY:
        {
            DestroyWindow(hToolTip);
        }
        return false;
    }
    return false;
}
```

```
#include "JamellaD2E.h"

void StripRTF(char *d, const char *s)
{
    while(*s)
    {
        if (*s == '{')
        {
            int dp = 0;
            while(*s)
            {
                if (*s == '}' && dp == 0)
                    break;
                else if (*s == '}' && dp > 0) {
                    dp--;
                    s++;
                    continue;
                }
                else if (*s == '{') {
                    dp++;
                    s++;
                    continue;
                }

                else if (*s == '\\' && dp == 1)
                {
                    if (*(s+1) == 'p' &&  *(s+2) == 'a' && *(s+3) == 'r')
                        *d++ = '\n';

                    while(*s && *s != ' ' && *s != '{')
                        s++;

                    if (*s == ' ') s++;
                }
                else if (dp == 1)
                    *d++ = *s++;
                else
                    s++;
            }
        }
    }
    *d = 0;
}



void WriteTextSummary(HWND hWnd)
{
    if (!fc.isloaded())
    {
        ErrorBox("You must load a character!",hWnd);
        return;
    }

    // common dialog box structure
    char QueryFilename[260];
    OPENFILENAME ofn;

    {   // Initialize OPENFILENAME
        ZeroMemory(&ofn,sizeof(OPENFILENAME));
        ofn.lStructSize = sizeof(OPENFILENAME);

        ofn.hwndOwner = hWnd;
        ofn.lpstrFilter = "Text File (*.txt)\0*.txt\0";
        ofn.nFilterIndex = 0;
        ofn.lpstrFile = QueryFilename;
        ofn.nMaxFile = sizeof(QueryFilename);
        ofn.lpstrFileTitle = NULL;
        ofn.lpstrDefExt = "txt";
        ofn.nMaxFileTitle = 0;

        ofn.Flags = OFN_PATHMUSTEXIST | OFN_OVERWRITEPROMPT | OFN_NOREADONLYRETURN | OFN_HIDEREADONLY;

        ZeroMemory(&QueryFilename,sizeof(QueryFilename));
        strcpy(QueryFilename,fc.Header.playername);
    }

    // Display the Open Dialog Box
    if (GetSaveFileName(&ofn))
```

```
    {
        // Open File
        FILE *f = fopen(QueryFilename,"wt");

        if (f == 0)
        {
            MessageBox(hWnd,"Could not open file!",PROGRAMNAME,
                MB_OK | MB_ICONSTOP | MB_APPLMODAL);
            return;
        }

        fprintf(f,"Diablo 2 Character Summary\nwritten by %s\n\n",PROGRAMNAME);


        fprintf(f,"Character Properties:\n");
        fprintf(f,"Name: %s\n",fc.Header.playername);
        fprintf(f,"Class: %s\n",CharClasses[fc.Header.playerclass]);
        fprintf(f,"Level: %u\n",fc.gf.level);
        fprintf(f,"Experience: %u\n",fc.gf.experience);

        int diff = 0;
        if ((fc.Header.diff & 0x0C) == 0x04) diff = 1;
        if ((fc.Header.diff & 0x0C) == 0x08) diff = 2;
        if ((fc.Header.diff & 0x0C) == 0x0C) diff = 3;

        fprintf(f,"Difficulty: %s\n",Difficulties[diff]);
        fprintf(f,"Hardcore: %s\n",fc.Header.hardcore & 0x04 ? "Yes" : "No");
        fprintf(f,"Dead: %s\n",fc.Header.hardcore & 0x08 ? "Yes" : "No");


        fprintf(f,"\nCharacter Stats:\n");
        fprintf(f,"Strength: %u\n",fc.gf.strength);
        fprintf(f,"Dexterity: %u\n",fc.gf.dexterity);
        fprintf(f,"Vitality: %u\n",fc.gf.vitality);
        fprintf(f,"Energy: %u\n",fc.gf.energy);
        fprintf(f,"Health: %u / %u\n",fc.gf.health,fc.gf.healthmax);
        fprintf(f,"Mana: %u / %u\n",fc.gf.mana,fc.gf.manamax);
        fprintf(f,"Stamina: %u / %u\n",fc.gf.stamina,fc.gf.staminamax);
        fprintf(f,"Gold on Body: %u\n",fc.gf.stamina,fc.gf.goldperson);
        fprintf(f,"Gold in Stash: %u\n",fc.gf.stamina,fc.gf.goldstash);

        fprintf(f,"\n%u Items:\n",Items->Count());

        for(Item *I = Items;I != 0;I = I->Next())
        {
            char description[2048];
            StripRTF(description,I->RichText());
            fprintf(f,"%s\n",description);
        }

        fclose(f);
    }

    return;
}
```

```
// ToolTips.cpp for Jamella's Diablo 2 Editor

#include "JamellaD2E.h"

struct
{
    int     ToolSet;
    int     CtrlID;
    char*   Text;
}
ToolTips[] =
{
{ IDD_TAB1,IDC_TAB1_Name, "" },
{ IDD_TAB1,IDC_TAB1_Rename, "" },
{ IDD_TAB1,IDC_TAB1_Class, "Changes your character's class.\n\rOptions: Amazon, Barbarian, Necromancer, Sorceress, Palad
in.\n\rNote that when changing your class, your character's skills are changed too. Please adapt these in the Skills Tab
." },
{ IDD_TAB1,IDC_TAB1_Level, "" },
{ IDD_TAB1,IDC_TAB1_Experience, "" },
{ IDD_TAB1,IDC_TAB1_StartTown, "" },
{ IDD_TAB1,IDC_TAB1_Char1, "" },
{ IDD_TAB1,IDC_TAB1_Char2, "" },
{ IDD_TAB1,IDC_TAB1_Char3, "" },
{ IDD_TAB1,IDC_TAB1_Char4, "" },
{ IDD_TAB1,IDC_TAB1_Char5, "" },
{ IDD_TAB1,IDC_TAB1_Plus1, "" },
{ IDD_TAB1,IDC_TAB1_Plus2, "" },
{ IDD_TAB1,IDC_TAB1_Plus3, "" },
{ IDD_TAB1,IDC_TAB1_Plus4, "" },
{ IDD_TAB1,IDC_TAB1_Plus5, "" },
{ IDD_TAB1,IDC_TAB1_StatsLink, "" },
{ IDD_TAB1,IDC_TAB1_Batch, "" },
{ IDD_TAB1,IDC_TAB1_Hardcore, "" },
{ IDD_TAB1,IDC_TAB1_Dead, "" },
{ IDD_TAB1,IDC_TAB1_Difficulty, "" },
{ IDD_TAB1,IDC_TAB1_Health, "" },
{ IDD_TAB1,IDC_TAB1_HealthMax, "" },
{ IDD_TAB1,IDC_TAB1_Stamina, "" },
{ IDD_TAB1,IDC_TAB1_StaminaMax, "" },
{ IDD_TAB1,IDC_TAB1_Mana, "" },
{ IDD_TAB1,IDC_TAB1_ManaMax, "" },
{ IDD_TAB1,IDC_TAB1_GoldPerson, "" },
{ IDD_TAB1,IDC_TAB1_SetMaxGoldPerson, "" },
{ IDD_TAB1,IDC_TAB1_GoldStash, "" },
{ IDD_TAB1,IDC_TAB1_SetMaxGoldStash, "" },
{ IDD_TAB2,IDC_TAB2_Body, "" },
{ IDD_TAB2,IDC_TAB2_Inv, "" },
{ IDD_TAB2,IDC_TAB2_Cube, "" },
{ IDD_TAB2,IDC_TAB2_Belt, "" },
{ IDD_TAB2,IDC_TAB2_Stash, "" },
{ IDD_TAB2,IDC_TAB2_CopyBuffer, "This buffer field acts as an item duplicator.\r\nDrop an item into this buffer and you
can copy it multiple times from it. After finished with duplicating you can delete it.\r\nYou can also load and save the
 contents of this buffer using the two buttons above." },
{ IDD_TAB2,IDC_TAB2_Save, "" },
{ IDD_TAB2,IDC_TAB2_Load, "" },
{ IDD_TAB2,IDC_TAB2_RichText, "" },
{ IDD_TAB2,IDC_TAB2_AttrRandom, "" },
{ IDD_TAB2,IDC_TAB2_Batch, "" },
{ IDD_TAB2,IDC_TAB2_ExpertMode, "" },
{ IDD_TAB2,IDC_TAB2_Save, "Clicking this button you can save the contents of the Copy Buffer into a file.\r\nThis file w
ill contain the 27 bytes of item data.\r\nNo inserted gems will be saved along!" },
{ IDD_TAB2,IDC_TAB2_Load, "With this button you can load a saved item into the Copy Buffer below." },
{ IDD_TAB2,IDC_TAB2_ItemTree, "This item tree contains all of the items in the game. You can drag-drop entries into your
 inventory to create a new item." },
{ IDD_TAB3,IDC_TAB3_Track, "" },
{ IDD_TAB3,IDC_TAB3_Bonus, "" },
{ IDD_TAB3,IDC_TAB3_BonusPlus, "" },
{ IDD_TAB3,IDC_TAB3_Batch, "" },
{ IDD_TAB3,IDC_TAB3_Prop_Bitmap, "" },
{ IDD_TAB4,IDC_TAB4_Diff1, "" },
{ IDD_TAB4,IDC_TAB4_Diff2, "" },
{ IDD_TAB4,IDC_TAB4_Diff3, "" },
{ IDD_TAB4,IDC_TAB4_Act1, "" },
{ IDD_TAB4,IDC_TAB4_Act2, "" },
{ IDD_TAB4,IDC_TAB4_Act3, "" },
{ IDD_TAB4,IDC_TAB4_Act4, "" },
{ IDD_TAB4,IDC_TAB4_Act2_On, "" },
{ IDD_TAB4,IDC_TAB4_Act3_On, "" },
{ IDD_TAB4,IDC_TAB4_Act4_On, "" },
```

```
{ IDD_TAB4,IDC_TAB4_SelAll, "" },
{ IDD_TAB4,IDC_TAB4_SetAll1, "" },
{ IDD_TAB4,IDC_TAB4_SetAll2, "" },
{ IDD_TAB4,IDC_TAB4_CowLevel, "" },
{ IDD_TAB4,IDC_TAB4_Sel1, "" },
{ IDD_TAB4,IDC_TAB4_Sel2, "" },
{ IDD_TAB4,IDC_TAB4_Sel3, "" },
{ IDD_TAB4,IDC_TAB4_Sel4, "" },
{ IDD_TAB4,IDC_TAB4_Sel5, "" },
{ IDD_TAB4,IDC_TAB4_Sel6, "" },
{ IDD_TAB4,IDC_TAB4_Bmp1, "" },
{ IDD_TAB4,IDC_TAB4_Bmp2, "" },
{ IDD_TAB4,IDC_TAB4_Bmp3, "" },
{ IDD_TAB4,IDC_TAB4_Bmp4, "" },
{ IDD_TAB4,IDC_TAB4_Bmp5, "" },
{ IDD_TAB4,IDC_TAB4_Bmp6, "" },
{ IDD_TAB5,IDC_TAB5_Diff1, "" },
{ IDD_TAB5,IDC_TAB5_Diff2, "" },
{ IDD_TAB5,IDC_TAB5_Diff3, "" },
{ IDD_TAB5,IDC_TAB5_Batch, "" },
{ IDD_TAB5,IDC_TAB5_Way01, "" },
{ IDD_TAB5,IDC_TAB5_Way02, "" },
{ IDD_TAB5,IDC_TAB5_Way03, "" },
{ IDD_TAB5,IDC_TAB5_Way04, "" },
{ IDD_TAB5,IDC_TAB5_Way05, "" },
{ IDD_TAB5,IDC_TAB5_Way06, "" },
{ IDD_TAB5,IDC_TAB5_Way11, "" },
{ IDD_TAB5,IDC_TAB5_Way12, "" },
{ IDD_TAB5,IDC_TAB5_Way13, "" },
{ IDD_TAB5,IDC_TAB5_Way14, "" },
{ IDD_TAB5,IDC_TAB5_Way15, "" },
{ IDD_TAB5,IDC_TAB5_Way16, "" },
{ IDD_TAB5,IDC_TAB5_Way21, "" },
{ IDD_TAB5,IDC_TAB5_Way22, "" },
{ IDD_TAB5,IDC_TAB5_Way23, "" },
{ IDD_TAB5,IDC_TAB5_Way24, "" },
{ IDD_TAB5,IDC_TAB5_Way25, "" },
{ IDD_TAB5,IDC_TAB5_Way26, "" },
{ IDD_TAB5,IDC_TAB5_Way31, "" },
{ IDD_TAB5,IDC_TAB5_Way32, "" },
{ IDD_TAB5,IDC_TAB5_Way33, "" },
};

HWND hToolTip;

HWND CreateToolTipCtrl(HWND hParent,int ToolSet,bool Relay)
{
    if (!RegOptions.ToolTips) return 0;

    HWND hTip = CreateWindowEx(NULL, TOOLTIPS_CLASS, NULL,
        WS_POPUP | TTS_NOPREFIX | TTS_ALWAYSTIP | TTS_BALLOON,
        CW_USEDEFAULT, CW_USEDEFAULT,
        CW_USEDEFAULT, CW_USEDEFAULT,
        hParent, NULL, hInstance,
        NULL);

    SetWindowPos(hTip, HWND_TOPMOST,0, 0, 0, 0,
        SWP_NOMOVE | SWP_NOSIZE | SWP_NOACTIVATE);

    SendMessage(hTip,TTM_ACTIVATE,TRUE,0);

    SendMessage(hTip,TTM_SETTIPTEXTCOLOR,RGB(255,50,50),0);
    SendMessage(hTip,TTM_SETTIPBKCOLOR,RGB(255,255,220),0);

    SendMessage(hTip,TTM_SETDELAYTIME,TTDT_AUTOMATIC,1000);
    SendMessage(hTip,TTM_SETMAXTIPWIDTH,0,300);

    TOOLINFO ti;
    ZeroMemory(&ti,sizeof ti);

    ti.cbSize = sizeof ti;
    ti.uFlags = TTF_IDISHWND | (Relay ? TTF_SUBCLASS : 0) | TTF_TRANSPARENT;
    ti.hwnd = hParent;
    ti.hinst = hInstance;

    for(int z=0;z<sizeof ToolTips / sizeof ToolTips[0];z++)
    {
        if (ToolTips[z].ToolSet == ToolSet)
        {
```

```
            ti.uId = (int)GetDlgItem(hParent,ToolTips[z].CtrlID);
            ti.lpszText = ToolTips[z].Text;

            if (!ToolTips[z].Text || *ToolTips[z].Text == 0) continue;

            SendMessage(hTip,TTM_ADDTOOL,0,(LPARAM) &ti);
        }
    }

    return hTip;
}
```

```
#include "JamellaD2E.h"

const struct _UniqueItem UniqueItems[] =
{
{ "The Gnasher",10,'ht00',30,30,'sp02',50,50,'sp03',20,20,0,0,0,0,0,0,0,0,0,0,0,0 },
{ "Deathspade",10,'hw01',1,1,'dm01',8,8,'ht01',15,15,'sp05',2,2,0,0,0,0,0,0,0,0,0,0 },
{ "Bladebone",10,'md01',100,100,'md03',40,40,'at02',20,20,'ar00',20,20,'df00',4,4,'df01',7,7,0,0,0 },
{ "Mindrend",10,'hw01',60,60,'hl11',20,20,'dl00',1,1,'dl01',12,12,'ht00',50,50,0,0,0,0,0,0 },
{ "Rakescar",10,'dp00',64,64,'dp01',96,96,'dp02',75,75,'ht00',50,50,'rp00',50,50,0,0,0,0,0,0 },
{ "Fechmars Axe",10,'dm02',50,50,'sp01',1,1,'rc00',30,30,'lt00',2,2,0,0,0,0,0,0,0,0,0,0 },
{ "Goreshovel",10,'at03',1,1,'st00',25,25,'sp02',60,60,0,0,0,0,0,0,0,0,0,0,0,0 },
{ "The Chieftan",10,'dm02',100,100,'ra00',10,10,'rp00',20,20,'at01',10,10,0,0,0,0,0,0,0,0,0,0 },
{ "Brainhew",10,'dm01',14,14,'mm00',25,25,'lt00',4,4,'mm03',10,10,0,0,0,0,0,0,0,0,0,0 },
{ "The Humongous",10,'st00',10,10,'dm01',8,8,'dm00',15,15,'sp03',33,33,'rq00',20,20,0,0,0,0,0,0 },
{ "Iros Torch",10,'sk02',1,1,'mh03',6,6,'df00',5,5,'df01',9,9,0,0,0,0,0,0,0,0,0,0 },
{ "Maelstromwrath",10,'dl00',1,1,'dl01',9,9,'rl00',40,40,'dc02',1,1,'ca02',20,20,'mm00',13,13,0,0,0 },
{ "Gravenspine",10,'st00',10,10,'dx00',10,10,'dc00',4,4,'dc01',8,8,'dc03',75,75,'mm03',5,5,0,0,0 },
{ "Umes Lament",10,'sk02',2,2,'mm00',40,40,'ca02',20,20,'hw00',50,50,0,0,0,0,0,0,0,0,0,0 },
{ "Felloak",10,'rl00',60,60,'rf00',20,20,'ko00',1,1,'df00',6,6,'df01',8,8,0,0,0,0,0,0 },
{ "Knell Striker",10,'sp03',25,25,'rf00',20,20,'rp00',20,20,'mm00',15,15,'ht00',35,35,0,0,0,0,0,0 },
{ "Rusthandle",10,'sk01',1,1,'dm01',3,3,'dm00',7,7,'ar03',1,1,'mh03',8,8,0,0,0,0,0,0 },
{ "Stormeye",10,'dl00',1,1,'dl01',6,6,'dc00',3,3,'dc01',5,5,'dc03',75,75,'lf00',10,10,0,0,0 },
{ "Stoutnail",10,'at00',1,10,'dm02',100,100,'mh00',15,15,'ar03',2,2,0,0,0,0,0,0,0,0,0,0 },
{ "Crushflange",10,'st00',15,15,'ko00',1,1,'lt00',2,2,'rf00',50,50,0,0,0,0,0,0,0,0,0,0 },
{ "Bloodrise",10,'dm02',100,100,'ht01',50,50,'sp06',3,5,'lt00',2,2,'at01',10,10,0,0,0,0,0,0 },
{ "The Generals Tan Do Li Ga",10,'dm01',1,1,'dm00',20,20,'sp07',50,50,'ar00',25,25,'mm03',5,5,0,0,0,0,0,0 },
{ "Ironstone",10,'ht00',80,80,'dm02',60,60,'dl00',1,1,'dl01',10,10,'dx00',-5,-5,0,0,0,0,0,0 },
{ "Bonesob",10,'dm02',200,200,'sp03',40,40,'rf00',30,30,'rc00',30,30,0,0,0,0,0,0,0,0,0,0 },
{ "Steeldriver",10,'rq00',-50,-50,'at03',30,30,'hl08',25,25,0,0,0,0,0,0,0,0,0,0,0,0 },
{ "Rixots Keen",10,'dm01',5,5,'ht01',20,20,'lt00',2,2,'sp03',25,25,'ar00',25,25,0,0,0,0,0,0 },
{ "Blood Crescent",10,'ra00',15,15,'dm02',25,25,'mh00',15,15,'lt00',4,4,0,0,0,0,0,0,0,0,0,0 },
{ "Krintizs Skewer",10,'ar04',1,1,'st00',10,10,'dx00',10,10,'mm03',3,3,'dm02',50,50,0,0,0,0,0,0 },
{ "Gleamscythe",10,'lt00',3,3,'mm00',30,30,'ar00',20,20,'at02',20,20,0,0,0,0,0,0,0,0,0,0 },
{ "Azurewrath",10,'ds00',30,30,'ib01',10,10,'dc00',3,3,'dc01',6,6,'dc03',100,100,0,0,0,0,0,0 },
{ "Griswolds Edge",10,'df00',8,10,'df01',12,16,'ht00',40,40,'at01',-10,-10,'ko00',1,1,0,0,0,0,0,0 },
{ "Hellplague",10,'mm03',5,5,'mh03',5,5,'dp00',48,48,'dp01',96,96,'dp02',150,150,0,0,0,0,0,0 },
{ "Culwens Point",10,'sk08',1,1,'pd00',50,50,'gh01',20,20,'bl02',20,20,'ht00',60,60,0,0,0,0,0,0 },
{ "Shadowfang",10,'mm03',5,5,'rc00',20,20,'lt00',-2,-2,'dc00',5,5,'dc01',10,10,'dc03',75,75,0,0,0 },
{ "Soulflay",10,'mm03',4,10,'mh03',4,4,'dm02',30,30,'ra00',5,5,0,0,0,0,0,0,0,0,0,0 },
{ "Kinemils Awl",10,'ht00',45,45,'mm00',20,20,'df00',6,6,'df01',12,12,0,0,0,0,0,0,0,0,0,0 },
{ "Blacktongue",10,'dp00',48,48,'hl10',1,1,'mh00',-10,-10,'rp00',50,50,'dp01',64,64,'dp02',75,75,0,0,0 },
{ "Ripsaw",10,'sp02',50,50,'sp06',3,5,'dm00',15,15,'mm03',6,6,0,0,0,0,0,0,0,0,0,0 },
{ "The Patriarch",10,'ar01',3,3,'ar03',3,3,'hw01',50,50,'ib00',100,100,0,0,0,0,0,0,0,0,0,0 },
{ "Gull",10,'dm01',1,1,'dm00',15,15,'ib01',50,50,'mm00',-5,-5,0,0,0,0,0,0,0,0,0,0 },
{ "The Diggler",10,'dx00',10,10,'dm02',50,50,'at03',1,1,'rc00',25,25,'rf00',25,25,0,0,0,0,0,0 },
{ "The Jade Tan Do",10,'ht00',75,75,'sp10',1,1,'dp00',24,24,'dp01',48,48,'dp02',75,75,0,0,0,0,0,0 },
{ "Irices Shard",10,'ca02',20,20,'mm00',50,50,'ht00',55,55,'ra00',10,10,0,0,0,0,0,0,0,0,0,0 },
{ "The Dragon Chang",10,'ht00',35,35,'dm01',10,10,'lt00',2,2,'md01',100,100,0,0,0,0,0,0,0,0,0,0 },
{ "Razortine",10,'sp07',25,25,'ar06',50,50,'st00',15,15,'dx00',8,8,'at03',30,30,0,0,0,0,0,0 },
{ "Bloodthief",10,'sp02',35,35,'st00',10,10,'mh03',8,8,'mh00',26,26,0,0,0,0,0,0,0,0,0,0 },
{ "Lance of Yaggai",10,'at00',8,8,'dl00',1,1,'dl01',40,40,'ra00',15,15,0,0,0,0,0,0,0,0,0,0 },
{ "The Tannr Gorerod",10,'df00',12,12,'df01',18,18,'rf01',15,15,'mh00',30,30,'ht00',60,60,'lt00',3,3,'rf00',15,15 },
{ "Dimoaks Hew",10,'dx00',15,15,'dm02',100,100,'at02',20,20,'ar00',-8,-8,0,0,0,0,0,0,0,0,0,0 },
{ "Steelgoad",10,'hw00',75,75,'ds00',30,30,'ht00',30,30,'ra00',5,5,0,0,0,0,0,0,0,0,0,0 },
{ "Soul Harvest",10,'sp02',30,30,'ht00',45,45,'ra00',20,20,'dp00',32,32,'dp01',48,48,'dp02',125,125,0,0,0 },
{ "The Battlebranch",10,'at03',30,30,'dx00',10,10,'dm02',50,50,'ht00',40,40,'mh03',7,7,0,0,0,0,0,0 },
{ "Woestave",10,'dp00',24,24,'dp01',96,96,'sp07',50,50,'rp01',15,15,'rp00',15,15,'lt00',2,2,'dp02',75,75 },
{ "The Grim Reaper",10,'ds00',50,50,'hl10',1,1,'mm03',5,5,'mh00',-20,-20,0,0,0,0,0,0,0,0,0,0 },
{ "Bane Ash",10,'df00',4,4,'df01',6,6,'rf00',50,50,'mm00',30,30,'at02',20,20,0,0,0,0,0,0 },
{ "Serpent Lord",10,'dp00',8,8,'dp01',128,128,'rp00',50,50,'lt00',-1,-1,'dp02',75,75,'mm00',10,10,0,0,0 },
{ "Lazarus Spire",10,'ra00',30,30,'ar03',3,3,'mm00',20,20,'at02',20,20,'st00',9,9,'df00',7,7,'df01',12,12 },
{ "The Salamander",10,'sk07',2,2,'df00',1,1,'df01',10,10,'rf00',20,20,'ht00',50,50,0,0,0,0,0,0 },
{ "The Iron Jang Bong",10,'dl00',1,1,'ht01',50,50,'ar00',25,25,'sk03',1,1,'ca02',20,20,'dl01',25,25,0,0,0 },
{ "Pluckeye",10,'ht00',28,28,'dm02',100,100,'mh00',10,10,'lt00',2,2,'mm03',3,3,0,0,0,0,0,0 },
{ "Witherstring",10,'at03',1,1,'dm01',1,1,'dm00',3,3,'ht00',50,50,'sp14',35,35,0,0,0,0,0,0 },
{ "Rimeraven",10,'ht01',50,50,'dx00',3,3,'sp15',30,30,'st00',3,3,0,0,0,0,0,0,0,0,0,0 },
{ "Piercerib",10,'ra00',10,10,'ds00',30,30,'ht00',60,60,'md01',100,100,0,0,0,0,0,0,0,0,0,0 },
{ "Pullspite",10,'dl00',1,1,'st00',8,8,'ht00',28,28,'sp13',10,10,'dl01',10,10,'rl00',25,25,0,0,0 },
{ "Wizendraw",10,'sp14',1,1,'mm00',30,30,'at02',20,20,'rc00',26,26,'ht00',36,36,0,0,0,0,0,0 },
{ "Hellclap",10,'at01',10,10,'df00',6,6,'df01',9,9,'ht00',20,20,'rf00',40,40,'dx00',12,12,0,0,0 },
{ "Blastbark",10,'dm02',50,50,'st00',5,5,'sk00',1,1,'mm03',3,3,0,0,0,0,0,0,0,0,0,0 },
{ "Leadcrow",10,'dx00',10,10,'mh00',10,10,'dm02',50,50,'rp00',30,30,'ds00',25,25,'ht00',40,40,0,0,0 },
{ "Ichorsting",10,'dp00',24,24,'dp01',32,32,'dx00',20,20,'sp13',50,50,'ht00',50,50,'dp02',75,75,0,0,0 },
{ "Hellcast",10,'sp15',1,1,'rf01',15,15,'rf00',15,15,'ht00',70,70,'at02',20,20,0,0,0,0,0,0 },
{ "Doomspittle",10,'sk00',1,1,'sp13',35,35,'at03',30,30,'mh00',15,15,0,0,0,0,0,0,0,0,0,0 },
{ "War Bonnet",10,'mh00',15,15,'ht00',30,30,'dm02',40,40,'mm00',15,15,0,0,0,0,0,0,0,0,0,0 },
{ "Tarnhelm",10,'ib00',75,75,'ib01',25,50,'sk08',1,1,0,0,0,0,0,0,0,0,0,0,0,0 },
{ "Coif of Glory",10,'at04',4,4,'hw01',1,1,'rl00',15,15,'ar08',25,25,0,0,0,0,0,0,0,0,0,0 },
{ "Duskdeep",10,'lt00',-2,-2,'ra00',5,5,'ar01',3,3,'dm00',8,8,0,0,0,0,0,0,0,0,0,0 },
```

```
{ "Wormskull",10,'sk02',1,1,'mh03',5,5,'mm00',10,10,'rp00',25,25,0,0,0,0,0,0,0,0 },
{ "Howltusk",10,'ar03',2,2,'at00',3,3,'ar02',40,40,'dm03',35,35,0,0,0,0,0,0,0,0 },
{ "Undead Crown",10,'mh03',4,4,'ar00',8,8,'rp00',50,50,'dc02',1,1,0,0,0,0,0,0,0,0 },
{ "The Face of Horror",10,'hw00',1,1,'st00',20,20,'ra00',5,5,'md01',50,50,0,0,0,0,0,0,0,0 },
{ "Greyform",10,'ar03',3,3,'rc00',20,20,'rf00',20,20,'dx00',10,10,'mh03',5,5,0,0,0,0,0,0 },
{ "Blinkbats Form",10,'ar08',16,16,'ve00',1,1,'ar00',25,25,'df00',3,3,'df01',6,6,0,0,0,0,0,0 },
{ "The Centurion",10,'ar00',30,30,'ht00',25,25,'ar01',2,2,'mh00',15,15,'mm00',15,15,'hl12',15,15,'lf00',3,3 },
{ "Twitchthroe",10,'at01',1,1,'dx00',10,10,'bl00',25,25,'ar00',15,15,'st00',10,10,0,0,0,0,0,0 },
{ "Darkglow",10,'ht00',20,20,'ra01',5,5,'lt00',3,3,'ar09',25,25,'ra00',10,10,0,0,0,0,0,0 },
{ "Hawkmail",10,'ar00',12,12,'rc01',15,15,'rc00',15,15,'sp10',1,1,0,0,0,0,0,0,0,0 },
{ "Sparking Mail",10,'ar00',40,40,'dl00',1,1,'dl01',10,10,'at04',1,10,0,0,0,0,0,0,0,0 },
{ "Venomsward",10,'dp00',24,24,'dp01',32,32,'dm02',50,50,'rp01',15,15,'lt00',2,2,'dp02',75,75,'rp00',15,15 },
{ "Iceblink",10,'sp01',1,1,'rc00',30,30,'lt00',4,4,'ar03',1,1,0,0,0,0,0,0,0,0 },
{ "Boneflesh",10,'mh03',5,5,'ar00',35,35,'ht00',35,35,0,0,0,0,0,0,0,0,0,0 },
{ "Rockfleece",10,'rq00',-10,-10,'ar00',25,25,'rp00',50,50,'ar01',3,3,'st00',5,5,0,0,0,0,0,0 },
{ "Rattlecage",10,'hw00',1,1,'ht00',45,45,'sp03',25,25,'ar00',45,45,0,0,0,0,0,0,0,0 },
{ "Goldskin",10,'ar00',75,75,'ra00',30,30,'at00',1,10,'lt00',2,2,0,0,0,0,0,0,0,0 },
{ "Victors Silk",10,'mm03',5,5,'sk08',1,1,'lt00',2,2,0,0,0,0,0,0,0,0,0,0 },
{ "Heavenly Garb",10,'ar02',100,100,'ra00',10,10,'hl11',25,25,0,0,0,0,0,0,0,0,0,0 },
{ "Pelta Lunata",10,'ar00',40,40,'mh00',10,10,'st00',2,2,'mm00',10,10,0,0,0,0,0,0,0,0 },
{ "Umbral Disk",10,'hw01',1,1,'dx00',10,10,'ar00',18,18,'mh00',20,20,'lt00',-2,-2,0,0,0,0,0,0 },
{ "Stormguild",10,'ar03',1,1,'rl00',25,25,'ar00',15,15,'dl00',1,1,'dl01',6,6,0,0,0,0,0,0 },
{ "Wall of the Eyeless",10,'sp05',5,5,'mm03',3,3,'ca02',20,20,'rp00',20,20,0,0,0,0,0,0,0,0 },
{ "Swordback Hold",10,'at00',5,5,'bl00',20,20,'sp02',50,50,'sp06',3,5,0,0,0,0,0,0,0,0 },
{ "Steelclash",10,'bl00',25,25,'sk01',1,1,'ar01',3,3,'lt00',3,3,0,0,0,0,0,0,0,0 },
{ "Bverrit Keep",10,'ar00',20,20,'rf00',25,25,'st00',5,5,'ar03',4,4,0,0,0,0,0,0,0,0 },
{ "The Ward",10,'ar00',40,40,'ar03',2,2,'st00',10,10,'bl00',10,10,0,0,0,0,0,0,0,0 },
{ "The Hand of Broc",10,'mm03',3,3,'mh03',3,3,'rp00',10,10,'mm00',20,20,0,0,0,0,0,0,0,0 },
{ "Bloodfist",10,'dm01',5,5,'mh00',40,40,'gh02',30,30,0,0,0,0,0,0,0,0,0,0 },
{ "Chance Guards",10,'ib00',200,200,'ib01',24,24,'ht00',25,25,'ar00',15,15,'lt00',2,2,0,0,0,0,0,0 },
{ "Magefist",10,'ca02',20,20,'hl11',25,25,'sk07',1,1,'df00',1,1,'df01',5,5,0,0,0,0,0,0 },
{ "Frostburn",10,'ar00',30,30,'dm02',85,85,'mm01',40,40,'dc00',1,1,'dc01',6,6,'dc03',50,50,0,0,0 },
{ "Hotspur",10,'rf01',15,15,'mh00',15,15,'df00',3,3,'df01',6,6,0,0,0,'rf00',15,15,0,0,0 },
{ "Gorefoot",10,'sp06',3,5,'ve01',10,10,'mm03',2,2,'at00',2,2,0,0,0,0,0,0,0,0 },
{ "Treads of Cthon",10,'ve02',1,1,'ar08',50,50,'hl13',50,50,'mh00',10,10,0,0,0,0,0,0,0,0 },
{ "Goblin Toe",10,'sp03',25,25,'ar01',1,1,'ar03',1,1,'ar00',5,5,'lt00',-1,-1,0,0,0,0,0,0 },
{ "Tearhaunch",10,'ar00',10,10,'st00',5,5,'dx00',5,5,'ve01',1,1,'ra00',10,10,0,0,0,0,0,0 },
{ "Lenyms Cord",10,'mm00',15,15,'hl11',30,30,'ra00',5,5,'lt00',1,1,0,0,0,0,0,0,0,0 },
{ "Snakecord",10,'dp00',16,16,'dp01',24,32,'dp02',75,75,'rp00',25,25,'ar00',11,11,'lf00',5,5,0,0,0 },
{ "Nightsmoke",10,'ra00',10,10,'dm03',50,50,'mm00',20,20,'ar01',2,2,0,0,0,0,0,0,0,0 },
{ "Goldwrap",10,'ib01',30,30,'lt00',2,2,'ar00',40,40,'at01',10,10,0,0,0,0,0,0,0,0 },
{ "Bladebuckle",10,'at00',8,8,'ar00',20,20,'ar01',3,3,'st00',5,5,'dx00',10,10,0,0,0,0,0,0 },
{ "Nokozan Relic",10,'df00',3,3,'rf01',10,10,'rf00',10,10,'lt00',3,3,'df01',6,6,0,0,0,0,0,0 },
{ "The Eye of Etlich",10,'ar08',10,40,'lt00',1,5,'sk08',1,1,'mh03',3,7,'dc00',1,2,'dc01',3,5,'dc03',50,300 },
{ "The Mahim-Oak Curio",10,'dx00',5,5,'mm00',20,20,'st00',5,5,'mh00',20,20,'ar00',10,10,'ht00',40,40,0,0,0 },
{ "Nagelring",10,'ar03',2,2,'at00',3,3,'ht00',18,18,'ib01',15,15,0,0,0,0,0,0,0,0 },
{ "Manald Heal",10,'mm03',4,4,'lf00',5,5,'mh00',20,20,0,0,0,0,0,0,0,0,0,0 },
{ "The Stone of Jordan",10,'mm00',20,20,'mm01',25,25,'dl00',1,1,'sk08',1,1,'dl01',12,12,0,0,0,0,0,0 },
{ "Amulet of the Viper",3,'mm00',10,10,'rp00',25,25,'mh00',10,10,0,0,0,0,0,0,0,0,0,0 },
{ "Staff of Kings",2,'ra00',10,10,'at03',50,50,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },
{ "Horadric Staff",5,'mm00',10,10,'rp00',25,25,'mh00',10,10,'ra00',10,10,'at03',50,50,0,0,0,0,0,0 },
{ "Hell Forge Hammer",5,'df00',5,5,'df01',20,20,'rf00',40,40,'ar00',35,35,0,0,0,0,0,0,0,0 },
{ "KhalimFlail",5,'dl00',1,1,'dl01',20,20,'at03',50,50,'ht00',40,40,0,0,0,0,0,0,0,0 },
{ "SuperKhalimFlail",5,'dl00',1,1,'dl01',40,40,'at03',50,50,'ht00',40,40,'mm03',6,6,'mh03',6,6,0,0,0 },
};

int nUniqueItems = sizeof UniqueItems / sizeof UniqueItems[0];
```

```
// Main.cpp from D2E

#include "JamellaD2E.h"
#include <time.h>
#include <shlwapi.h>

HINSTANCE hInstance;

inline void DoCRC();

DWORD GetDllVersion(LPCTSTR DllName)
{
    HINSTANCE hInst;
    DWORD dwVersion = 0;

    hInst = LoadLibrary(DllName);

    if(hInst)
    {
        DLLGETVERSIONPROC pDllGetVersion;

        pDllGetVersion = (DLLGETVERSIONPROC) GetProcAddress(hInst,"DllGetVersion");

        if(pDllGetVersion)
        {
            DLLVERSIONINFO dvi;
            HRESULT hr;

            ZeroMemory(&dvi, sizeof(dvi));
            dvi.cbSize = sizeof(dvi);

            hr = (*pDllGetVersion)(&dvi);

            if (hr == NOERROR)
            {
                dwVersion = MAKEWORD(dvi.dwMinorVersion,dvi.dwMajorVersion);
            }
        }

        FreeLibrary(hInst);
    }
    return dwVersion;
}

int PASCAL WinMain(HINSTANCE hInstance,
                   HINSTANCE hPrevInstance,
                   LPSTR lpszCmdLine,
                   int nCmdShow)
{
    ::hInstance = hInstance;
    srand((unsigned)time(NULL));

    if(GetDllVersion(TEXT("comctl32.dll")) < _WIN32_IE)
    {
        ErrorBox("Your system uses a Common Control Library prior to 5.80.\nThe editor requires one >= 5.80.\nDownload i
t from my web pages http://jamella.dyns.cx or from Microsoft.");
        return -1;
    }

    INITCOMMONCONTROLSEX cci;
    cci.dwSize = sizeof cci;
    cci.dwICC = ICC_PROGRESS_CLASS | ICC_TAB_CLASSES | ICC_TREEVIEW_CLASSES;
    if (!InitCommonControlsEx(&cci))
    {
        ErrorBox("The editor could not initialize the common control library.\nThis is major operating system failure. Y
ou better watch out.");
        return -1;
    }

    HINSTANCE hRichEdit = LoadLibrary("RICHED32.DLL");
    if (!hRichEdit)
    {
        ErrorBox("The editor could not load the RichEdit library.\nDownload it from my web pages http://jamella.dyns.cx
or from Microsoft.");
        return -1;
    }

    // Parse Command Line Options
    ParseCommandLine();
```

```
//  DoCRC();

    LoadEditorRegistryValues();

    int x = MainDialog(lpszCmdLine);

    FreeLibrary(hRichEdit);

    return x;
}

int ErrorMessage()
{
    char str[260];

    FormatMessage(FORMAT_MESSAGE_FROM_SYSTEM,
        NULL,
        GetLastError(),0,
        (LPTSTR) str,260,
        NULL);

    MessageBox(NULL,str,"Diablo 2 Save Game Editor",
        MB_OK | MB_ICONINFORMATION);

    return false;
}

inline void DoCRC()
{
    // Open File
    HANDLE hFile = CreateFile(ProgramFilePath(),
        GENERIC_READ,FILE_SHARE_READ,NULL,
        OPEN_EXISTING,FILE_ATTRIBUTE_NORMAL,NULL);

    if (hFile == INVALID_HANDLE_VALUE)
    {
        MessageBox(NULL,"Could not open program file for CRC checking!",PROGRAMNAME,
            MB_OK | MB_ICONSTOP | MB_APPLMODAL);
        exit(0);
    }

    const int CRCsize = 1024*3*4;
    BYTE CRCbuffer[CRCsize];
    DWORD *CRCbufferd = (DWORD*)CRCbuffer;
    DWORD CRCread;

    DWORD   fsizeread = 0;
    DWORD   CRC = 0;

    while(ReadFile(hFile,CRCbuffer,CRCsize,&CRCread,NULL))
    {
        if (CRCread == 0) break;
        fsizeread += CRCread;

        for(int z=0;z<CRCsize/3/4;z+=3)
        {
            CRC += CRCbufferd[z+0] * 1;
            CRC += CRCbufferd[z+1] * 3;
            CRC += CRCbufferd[z+2] * 7;
        }
    }

    if (CRC != 0x00000000)
    {
        MessageBox(NULL,"CRC check of program file failed! This executable was tampered with.\nGet a new one from http:/
/jamella.dyns.cx!",PROGRAMNAME,
            MB_OK | MB_ICONSTOP | MB_APPLMODAL);
        exit(0);
    }

    CloseHandle(hFile);
}
```

```
//{{NO_DEPENDENCIES}}
// Microsoft Developer Studio generated include file.
// Used by JamellaD2E.rc
//
#define IDI_ICON                        1
#define IDI_D2SICON                     2
#define IDI_ITEMICON                    3
#define IDOK2                           3
#define IDC_CUR_NO                      106
#define IDB_WAYPOINT_OFF                201
#define IDB_WAYPOINT_ON                 202
#define IDB_QUEST11                     225
#define IDB_QUEST12                     226
#define IDB_QUEST13                     227
#define IDB_QUEST14                     228
#define IDB_QUEST15                     229
#define IDB_QUEST16                     230
#define IDB_QUEST21                     231
#define IDB_QUEST22                     232
#define IDB_QUEST23                     233
#define IDB_QUEST24                     234
#define IDB_QUEST25                     235
#define IDB_QUEST26                     236
#define IDB_QUEST31                     237
#define IDB_QUEST32                     238
#define IDB_QUEST33                     239
#define IDB_QUEST34                     240
#define IDB_QUEST35                     241
#define IDB_QUEST36                     242
#define IDB_QUEST41                     243
#define IDB_QUEST42                     244
#define IDB_QUEST43                     245
#define IDD_COWLEVEL                    246
#define IDB_COWLEVEL_Cube               247
#define IDB_COWLEVEL_Portal             248
#define IDB_COWLEVEL_Leg                250
#define IDR_MAIN                        252
#define IDD_PROGRESS                    259
#define IDB_GRID_STASH                  260
#define IDB_GRID_CUBE                   261
#define IDB_GRID_INV                    262
#define IDB_GRID_BELT                   263
#define IDB_INV_WHOLE                   264
#define IDB_ITEM_UNKNOWN                273
#define IDC_CUR_MOVE                    274
#define IDC_CUR_CROSS                   275
#define IDB_NOTPLACEABLE                276
#define IDD_INFO                        285
#define IDB_JAMELLASMALL                286
#define IDB_GRID_2x4                    287
#define IDB_WEBLINK                     288
#define IDD_TAB2E                       289
#define IDD_TAB2Rnd                     290
#define IDD_TAB2Magic                   292
#define IDD_TAB2SS                      293
#define IDD_TAB2Rare                    295
#define IDC_CUR_ADD                     296
#define IDD_SAVE                        297
#define IDD_TAB2Quantity                298
#define IDD_TAB2Durability              299
#define IDD_TAB2Defense                 300
#define IDR_WAVE_JamellaSound           302
#define IDD_TAB2Gems7                   303
#define IDR_BATCH                       305
#define IDD_TAB2Gems1                   306
#define IDD_TAB2Gems2                   307
#define IDB_PLUS16                      308
#define IDN_Sorceress                   309
#define IDN_Barbarian                   310
#define IDN_Necromancer                 311
#define IDN_Paladin                     312
#define IDN_Amazon                      313
#define IDD_TAB2Ear                     314
#define IDD_NEW                         315
#define IDD_RENAME                      316
#define IDC_CUR_MOVECOPY                317
#define IDD_TAB2RingImage               318
#define IDD_TAB2AmuletImage             319
#define IDD_IBWSR                       322
```

```
#define IDD_UOPTIONS                    323
#define IDB_GRID_2x4_Black              327
#define IDD_TAB2Gems3                   328
#define IDN_SetTyrael                   330
#define IDD_HELP                        336
#define IDH_TAB2Magic                   340
#define IDH_TAB2Rare                    342
#define IDD_EOPTIONS                    343
#define IDI_CHELP                       344
#define IDD_TAB2SearchA                 346
#define IDD_TAB2S                       346
#define IDH_TAB2S                       349
#define IDC_TAB3_Bonus                  1074
#define IDC_TAB3_BonusPlus              1075
#define IDC_TAB5_Diff1                  1076
#define IDC_TAB5_Diff2                  1077
#define IDC_TAB5_Diff3                  1078
#define IDC_TAB4_Act1                   1079
#define IDC_TAB5_Batch                  1079
#define IDC_TAB4_Act2                   1080
#define IDC_TAB4_Act3                   1081
#define IDC_TAB5_Way01                  1081
#define IDC_TAB4_Act4                   1082
#define IDC_TAB5_Way02                  1082
#define IDC_TAB5_Way03                  1083
#define IDC_TAB5_Way04                  1084
#define IDC_TAB5_Way05                  1085
#define IDC_TAB5_Way06                  1086
#define IDC_TAB5_Way07                  1087
#define IDC_TAB5_Way08                  1088
#define IDC_TAB5_Way09                  1089
#define IDC_TAB5_Way11                  1090
#define IDC_TAB5_Way12                  1091
#define IDC_TAB5_Way13                  1092
#define IDC_TAB5_Way14                  1093
#define IDC_TAB5_Way15                  1094
#define IDC_TAB5_Way16                  1095
#define IDC_TAB5_Way17                  1096
#define IDC_TAB5_Way18                  1097
#define IDC_TAB5_Way19                  1098
#define IDC_TAB5_Way21                  1099
#define IDC_TAB5_Way22                  1100
#define IDC_TAB5_Way23                  1101
#define IDC_TAB5_Way24                  1102
#define IDC_TAB5_Way25                  1103
#define IDC_TAB5_Way26                  1104
#define IDC_TAB5_Way27                  1105
#define IDC_TAB5_Way28                  1106
#define IDC_TAB5_Way29                  1107
#define IDC_TAB5_Way31                  1108
#define IDC_TAB5_Way32                  1109
#define IDC_TAB5_Way33                  1110
#define IDC_TAB1_MaxGoldStash           1111
#define IDC_TAB1_SetMaxGoldPerson       1112
#define IDC_TAB1_SetMaxGoldStash        1113
#define IDC_TAB4_SelAll                 1115
#define IDC_TAB4_SetAll1                1116
#define IDC_TAB4_Sel1                   1117
#define IDC_TAB4_Diff1                  1118
#define IDC_TAB4_Diff2                  1119
#define IDC_TAB4_Diff3                  1120
#define IDC_TAB4_Text1                  1121
#define IDC_TAB4_Text2                  1122
#define IDC_TAB4_Text3                  1123
#define IDC_TAB4_Text4                  1124
#define IDC_TAB4_Sel2                   1125
#define IDC_TAB4_Sel3                   1126
#define IDC_TAB4_Text5                  1127
#define IDC_TAB4_Sel4                   1128
#define IDC_TAB4_Sel5                   1129
#define IDC_TAB4_Sel6                   1130
#define IDC_TAB4_Text6                  1131
#define IDC_TAB4_Bmp1                   1132
#define IDC_TAB4_Bmp2                   1133
#define IDC_TAB4_Bmp3                   1134
#define IDC_TAB4_Bmp4                   1135
#define IDC_TAB4_Bmp5                   1136
#define IDC_TAB4_Bmp6                   1137
#define IDC_TAB4_SetAll2                1138
```

```
#define IDC_TAB4_Act2_On               1141
#define IDC_TAB0_Version               1141
#define IDC_TAB4_Act3_On               1142
#define IDC_TAB4_Act4_On               1143
#define IDC_TAB4_CowLevel              1146
#define IDC_COWLEVEL_TextA             1148
#define IDC_COWLEVEL_TextB             1149
#define IDC_TAB2_ItemTree              1149
#define IDC_COWLEVEL_TextB2            1150
#define IDC_PROGRESS_Bar               1151
#define IDC_PROGRESS_Text              1152
#define IDC_TAB0_BmpJamella            1153
#define IDC_TAB2_Inv                   1158
#define IDC_TAB2_Belt                  1159
#define IDC_TAB2_Cube                  1160
#define IDC_TAB2_Stash                 1161
#define IDC_TAB2_ExpertMode            1172
#define IDC_TAB2_Body                  1173
#define IDC_TAB1_Difficulty            1179
#define IDC_TAB1_StartTown             1180
#define IDC_INFO_PERMURL               1180
#define IDC_INFO_Email                 1182
#define IDC_INFO_Program               1183
#define IDC_INFO_URL                   1184
#define IDC_INFO_Date                  1185
#define IDC_TAB2_Save                  1186
#define IDC_TAB2_Load                  1187
#define IDC_TAB2_CopyBuffer            1188
#define IDC_INFO_WEBLINK1              1189
#define IDC_TAB2E_Raw00                1190
#define IDC_TAB2E_Raw02                1191
#define IDD_TAB0                       1200
#define IDR_MAINDIALOG                 1200
#define IDC_TAB2E_GemNum               1206
#define IDC_TAB2Rnd_Edit               1207
#define IDC_TAB2E_ItemCode             1207
#define IDC_TAB2E_Xoord                1208
#define IDC_TAB2E_Yoord                1209
#define IDC_TAB2E_DWA                  1211
#define IDC_TAB2E_DWB                  1212
#define IDC_TAB2Magic_PrefixTree       1215
#define IDC_TAB2Magic_SuffixTree       1216
#define IDC_TAB2Search_Progress        1219
#define IDC_TAB2SA_Counter             1219
#define IDC_TAB2Search_Current         1220
#define IDC_TAB2SA_Current             1220
#define IDC_TAB2_RichText              1222
#define IDC_TAB2E_RandA                1223
#define IDC_TAB2E_RandB                1224
#define IDC_TAB2Rare_NamePrefix        1225
#define IDC_TAB2Rare_NameSuffix        1226
#define IDC_TAB2E_Raw01                1230
#define IDC_TAB2E_Raw03                1232
#define IDC_TAB2E_Raw04                1233
#define IDC_TAB2Rare_Tree              1234
#define IDC_TAB2E_Raw05                1234
#define IDC_TAB2Rare_List              1235
#define IDC_TAB2E_Raw06                1235
#define IDC_TAB2Rare_TreePrefix        1235
#define IDC_SAVE_Backup                1236
#define IDC_TAB2E_Raw07                1236
#define IDC_TAB2Magic_PrefixMatch      1237
#define IDC_TAB2E_Raw08                1237
#define IDC_TAB2Magic_SuffixMatch      1238
#define IDC_TAB2E_Raw09                1238
#define IDC_TAB2Magic_Prefix1Match     1239
#define IDC_TAB2E_Raw0A                1239
#define IDC_TAB2Magic_Prefix2Match     1240
#define IDC_TAB2E_Raw0B                1240
#define IDC_TAB2Magic_Prefix3Match     1241
#define IDC_TAB2E_Raw0C                1241
#define IDC_TAB2Magic_Prefix4Match     1242
#define IDC_TAB2E_Raw0D                1242
#define IDC_TAB2Magic_Suffix1Match     1243
#define IDC_TAB2E_Raw0E                1243
#define IDC_TAB2E_Raw0F                1244
#define IDC_TAB2E_Raw10                1245
#define IDC_TAB2Magic_Prefix1ValueMin  1246
#define IDC_TAB2E_Raw11                1246
```

```
#define IDC_TAB2Magic_Prefix1ValueMax   1247
#define IDC_TAB2E_Raw12                 1247
#define IDC_TAB2Magic_Prefix2ValueMax   1248
#define IDC_TAB2E_Raw13                 1248
#define IDC_TAB2Magic_Prefix2ValueMin   1249
#define IDC_TAB2E_Raw14                 1249
#define IDC_TAB2Magic_Prefix3ValueMax   1250
#define IDC_TAB2E_Raw15                 1250
#define IDC_TAB2Magic_Prefix3ValueMin   1251
#define IDC_TAB2E_Raw16                 1251
#define IDC_TAB2Magic_Prefix4ValueMax   1252
#define IDC_TAB2E_Raw17                 1252
#define IDC_TAB2Magic_Prefix4ValueMin   1253
#define IDC_TAB2E_Raw18                 1253
#define IDC_TAB2Magic_Prefix1Value      1254
#define IDC_TAB2E_Raw19                 1254
#define IDC_TAB2Magic_Prefix2Value      1255
#define IDC_TAB2Search_Slider           1255
#define IDC_TAB2E_Raw1A                 1255
#define IDC_TAB2SA_Slider               1255
#define IDC_TAB2Magic_Prefix3Value      1256
#define IDC_TAB2E_Raw1B                 1256
#define IDC_TAB2Magic_Prefix4Value      1257
#define IDC_TAB2Search_Stop             1257
#define IDC_TAB2E_Raw1C                 1257
#define IDC_TAB2SA_Stop                 1257
#define IDC_TAB2Magic_Suffix1ValueMax   1258
#define IDC_TAB2Search_Priority         1258
#define IDC_TAB2E_Raw1D                 1258
#define IDC_TAB2SA_Priority             1258
#define IDC_TAB2Magic_Suffix1ValueMin   1259
#define IDC_TAB2Quantity_Set            1259
#define IDC_TAB2E_Raw1E                 1259
#define IDC_TAB2Magic_Suffix1Value      1260
#define IDC_TAB2Quantity_Max            1260
#define IDC_TAB2Durability_Max          1260
#define IDC_TAB2E_Raw1F                 1260
#define IDC_TAB2Durability_Set          1261
#define IDC_TAB2E_UniqueCode            1261
#define IDC_TAB2Defense_Value           1262
#define IDC_TAB2Defense_Range           1263
#define IDC_TAB2Rare_Average            1263
#define IDC_TAB2E_BodyCode              1263
#define IDC_TAB2Magic_Average           1264
#define IDC_TAB2E_MagicLevel            1264
#define IDC_TAB2E_ItemCodeChar          1265
#define IDC_TAB2Gems_Sel1               1266
#define IDC_TAB2E_Container             1266
#define IDC_TAB2Gems_Sel2               1267
#define IDC_TAB2Gems_Sel3               1268
#define IDC_TAB2Gems_Sel4               1269
#define IDC_TAB2Gems_Bmp1               1270
#define IDC_TAB2Gems_Bmp2               1271
#define IDC_TAB2Gems_Bmp3               1272
#define IDC_TAB2Gems_Info1              1273
#define IDC_TAB2Gems_Info2              1274
#define IDC_TAB2Gems_Info3              1275
#define IDC_TAB2Gems_Frame1             1276
#define IDC_TAB2Gems_Frame2             1277
#define IDC_TAB2Gems_Frame3             1278
#define IDC_TAB2Gems_Frame4             1279
#define IDC_TAB2Gems_Bmp4               1280
#define IDC_TAB1_Batch                  1281
#define IDC_TAB2Gems_Info4              1281
#define IDC_TAB3_Batch                  1282
#define IDC_TAB1_StatsLink              1283
#define IDC_TAB2Ear_Name                1284
#define IDC_TAB2Ear_Class               1285
#define IDC_TAB2Ear_Level               1286
#define IDC_TAB1_Rename                 1286
#define IDC_TAB2Gems_Frame5             1286
#define IDC_NEW_Class                   1287
#define IDC_TAB2Gems_Bmp5               1287
#define IDC_NEW_Name                    1288
#define IDC_TAB2Gems_Sel5               1288
#define IDC_TAB2Gems_Info5              1289
#define IDC_RENAME_Move                 1290
#define IDC_TAB2Gems_Frame6             1290
#define IDC_RENAME_Copy                 1291
```

```
#define IDC_TAB2Gems_Bmp6               1291
#define IDC_RENAME_Name                 1292
#define IDC_TAB2Gems_Sel6               1292
#define IDC_TAB2Gems_Info6              1293
#define IDC_TAB2Gems_Frame7             1294
#define IDC_TAB2AmuletImage_Image1      1295
#define IDC_TAB2Gems_Bmp7               1295
#define IDC_TAB2AmuletImage_Image2      1296
#define IDC_TAB2Gems_Sel7               1296
#define IDC_TAB2RingImage_Image1        1297
#define IDC_TAB2AmuletImage_Image3      1297
#define IDC_TAB2Gems_Info7              1297
#define IDC_TAB2RingImage_Image2        1298
#define IDC_TAB2RingImage_Image3        1299
#define IDC_TAB2RingImage_Image4        1300
#define IDD_TAB1                        1300
#define IDC_TAB1_MaxGoldPerson          1300
#define IDC_TAB2RingImage_Image5        1301
#define IDC_TAB1_Name                   1301
#define IDC_TAB1_Class                  1302
#define IDC_TAB1_Level                  1303
#define IDC_TAB1_Experience             1304
#define IDC_TAB1_Char1                  1305
#define IDC_TAB1_Char2                  1306
#define IDC_TAB1_Char3                  1307
#define IDC_TAB1_Char4                  1308
#define IDC_TAB1_Char5                  1309
#define IDC_TAB1_Plus1                  1310
#define IDC_TAB1_Plus2                  1311
#define IDC_TAB1_Plus3                  1312
#define IDC_TAB2E_ItemRecordID          1312
#define IDC_TAB1_Plus4                  1313
#define IDC_TAB1_Plus5                  1314
#define IDC_TAB1_GoldPerson             1315
#define IDC_TAB1_HealthMax              1316
#define IDC_HELP_Text                   1316
#define IDC_TAB1_Health                 1318
#define IDC_TAB1_StaminaMax             1319
#define IDC_OPTIONS_IFormat1            1319
#define IDC_TAB1_Stamina                1320
#define IDC_OPTIONS_IFormat2            1320
#define IDC_TAB2Search_Total            1320
#define IDC_TAB2SA_Scope                1320
#define IDC_TAB1_ManaMax                1321
#define IDC_OPTIONS_IFormat3            1321
#define IDC_TAB1_Mana                   1322
#define IDC_TAB2Rare_Check1             1322
#define IDC_TAB1_Dead                   1323
#define IDC_TAB2Rare_Text1              1323
#define IDC_TAB1_Hardcore               1324
#define IDC_TAB2Rare_Text2              1324
#define IDC_TAB1_GoldStash              1325
#define IDC_TAB2Rare_Text3              1325
#define IDC_TAB2Rare_Text4              1326
#define IDC_TAB2Rare_Text5              1327
#define IDC_TAB2Rare_Check2             1328
#define IDC_TAB2Rare_Check3             1329
#define IDC_TAB2Rare_Check4             1330
#define IDC_TAB2Rare_Check5             1331
#define IDC_TAB2Rare_Check6             1332
#define IDC_TAB2Rare_Text6              1333
#define IDC_UOPTIONS_AllSocktable       1334
#define IDC_EOPTIONS_7Gems              1335
#define IDC_UOPTIONS_ExceedQuantity     1336
#define IDC_UOPTIONS_Tooltips           1337
#define IDC_IBWSR_Bitmap                1338
#define IDC_UOPTIONS_AnnoyingMsgs       1338
#define IDC_IBWSR_RichText              1339
#define IDC_TAB2Magic_LockELevel        1340
#define IDC_TAB2Magic_LockValue         1341
#define IDC_TAB2Magic_CurrentELevel     1342
#define IDC_NEW_CreateNewbie            1343
#define IDC_NEW_Templates               1344
#define IDC_NEW_Description             1345
#define IDC_CHELP                       1346
#define IDC_TAB2Rare_TreeSuffix         1347
#define IDC_TAB2Rare_LockELevel         1348
#define IDC_TAB2Rare_CurrentELevel      1349
#define IDC_TAB2Rare_LockValue          1350
```

```
#define IDC_UOPTIONS_Associations       1352
#define IDC_TAB2_HistoryBack            1353
#define IDC_TAB2_HistoryNext            1354
#define IDC_TAB2E_FindInfo              1357
#define IDC_TAB2E_Decode                1358
#define IDC_TAB2Rare_Clear              1358
#define IDC_TAB2Magic_Clear             1359
#define IDC_TAB2S_Expand                1360
#define IDC_TAB2SS_Expand               1360
#define IDC_TAB2Search_Contract         1361
#define IDC_TAB2SS_Progress             1362
#define IDC_TAB2SS_Counter              1362
#define IDC_TAB2SS_Current              1363
#define IDC_TAB2SS_Scope                1364
#define IDC_TAB2SS_Stop                 1365
#define IDC_TAB2SA_TraverseMagicLevels  1366
#define IDC_TAB2SA_HitSelection         1367
#define IDC_TAB2SA_HitList              1368
#define IDC_TAB2SA_MagicLevel           1369
#define IDC_TAB2SA_Static1              1370
#define IDC_TAB2SA_Static2              1371
#define IDC_TAB2SA_Static3              1372
#define IDC_TAB2SA_Static4              1373
#define IDC_TAB2SA_Static5              1374
#define IDC_TAB2SA_Static6              1375
#define IDC_TAB2SA_Static7              1376
#define IDC_TAB2AS_Speed                1377
#define IDC_TAB2E_ItemList              1378
#define IDC_TAB2ItemList                1382
#define IDC_INFO_LINK                   1384
#define IDC_TAB2_OpenCube               1385
#define IDC_TAB2_OpenBelt               1386
#define IDC_TAB2_OpenStash              1387
#define IDC_TAB2Grid_Grid               1387
#define IDD_TAB2                        1400
#define IDD_TAB3                        1500
#define IDC_TAB3_Frame_A                1501
#define IDC_TAB3_Frame_B                1502
#define IDC_TAB3_Frame_C                1503
#define IDC_TAB3_Edit_A0                1504
#define IDC_TAB3_Edit_A1                1505
#define IDC_TAB3_Edit_A2                1506
#define IDC_TAB3_Edit_A3                1507
#define IDC_TAB3_Edit_A4                1508
#define IDC_TAB3_Edit_A5                1509
#define IDC_TAB3_Edit_A6                1510
#define IDC_TAB3_Edit_A7                1511
#define IDC_TAB3_Edit_A8                1512
#define IDC_TAB3_Edit_A9                1513
#define IDC_TAB3_Edit_B0                1514
#define IDC_TAB3_Edit_B1                1515
#define IDC_TAB3_Edit_B2                1516
#define IDC_TAB3_Edit_B3                1517
#define IDC_TAB3_Edit_B4                1518
#define IDC_TAB3_Edit_B5                1519
#define IDC_TAB3_Edit_B6                1520
#define IDC_TAB3_Edit_B7                1521
#define IDC_TAB3_Edit_B8                1522
#define IDC_TAB3_Edit_B9                1523
#define IDC_TAB3_Edit_C0                1524
#define IDC_TAB3_Edit_C1                1525
#define IDC_TAB3_Edit_C2                1526
#define IDC_TAB3_Edit_C3                1527
#define IDC_TAB3_Edit_C4                1528
#define IDC_TAB3_Edit_C5                1529
#define IDC_TAB3_Edit_C6                1530
#define IDC_TAB3_Edit_C7                1531
#define IDC_TAB3_Edit_C8                1532
#define IDC_TAB3_Edit_C9                1533
#define IDC_TAB3_Bitmap_A0              1534
#define IDC_TAB3_Bitmap_A1              1535
#define IDC_TAB3_Bitmap_A2              1536
#define IDC_TAB3_Bitmap_A3              1537
#define IDC_TAB3_Bitmap_A4              1538
#define IDC_TAB3_Bitmap_A5              1539
#define IDC_TAB3_Bitmap_A6              1540
#define IDC_TAB3_Bitmap_A7              1541
#define IDC_TAB3_Bitmap_A8              1542
#define IDC_TAB3_Bitmap_A9              1543
```

```
#define IDC_TAB3_Bitmap_B0              1544
#define IDC_TAB3_Bitmap_B1              1545
#define IDC_TAB3_Bitmap_B2              1546
#define IDC_TAB3_Bitmap_B3              1547
#define IDC_TAB3_Bitmap_B4              1548
#define IDC_TAB3_Bitmap_B5              1549
#define IDC_TAB3_Bitmap_B6              1550
#define IDC_TAB3_Bitmap_B7              1551
#define IDC_TAB3_Bitmap_B8              1552
#define IDC_TAB3_Bitmap_B9              1553
#define IDC_TAB3_Bitmap_C0              1554
#define IDC_TAB3_Bitmap_C1              1555
#define IDC_TAB3_Bitmap_C2              1556
#define IDC_TAB3_Bitmap_C3              1557
#define IDC_TAB3_Bitmap_C4              1558
#define IDC_TAB3_Bitmap_C5              1559
#define IDC_TAB3_Bitmap_C6              1560
#define IDC_TAB3_Bitmap_C7              1561
#define IDC_TAB3_Bitmap_C8              1562
#define IDC_TAB3_Bitmap_C9              1563
#define IDC_TAB3_Prop_Bitmap            1564
#define IDC_TAB3_Track                  1565
#define IDC_TAB3_Prop_Edit              1566
#define IDC_TAB3_Prop_Effect            1567
#define IDD_TAB5                        1600
#define IDD_TAB4                        1700
#define IDD_TAB2ItemList                1702
#define IDD_TAB2Grid                    1703
#define IDD_TAB2ExGrid                  1704
#define IDC_TAB2_AttrRandom             1800
#define IDC_TAB2_Batch                  1801
#define IDR_EXIT                        10002
#define IDB_ITEM_RING5                  11277
#define IDB_ITEM_AMULET2                11278
#define IDB_ITEM_AMULET3                11279
#define IDB_ITEM_RING1                  11280
#define IDB_ITEM_RING2                  11281
#define IDB_ITEM_RING3                  11282
#define IDB_ITEM_RING4                  11283
#define IDB_ITEM_AMULET1                11284
#define IDB_TAB1                        20000
#define IDB_TAB2                        20001
#define IDB_TAB3                        20002
#define IDB_TAB4                        20003
#define IDB_TAB5                        20004
#define IDB_JAMELLA                     21001
#define IDB_PLUS                        22000
#define IDB_SKILL_A01                   23100
#define IDB_SKILL_A02                   23101
#define IDB_SKILL_A03                   23102
#define IDB_SKILL_A04                   23103
#define IDB_SKILL_A05                   23104
#define IDB_SKILL_A06                   23105
#define IDB_SKILL_A07                   23106
#define IDB_SKILL_A08                   23107
#define IDB_SKILL_A09                   23108
#define IDB_SKILL_A10                   23109
#define IDB_SKILL_A11                   23110
#define IDB_SKILL_A12                   23111
#define IDB_SKILL_A13                   23112
#define IDB_SKILL_A14                   23113
#define IDB_SKILL_A15                   23114
#define IDB_SKILL_A16                   23115
#define IDB_SKILL_A17                   23116
#define IDB_SKILL_A18                   23117
#define IDB_SKILL_A19                   23118
#define IDB_SKILL_A20                   23119
#define IDB_SKILL_A21                   23120
#define IDB_SKILL_A22                   23121
#define IDB_SKILL_A23                   23122
#define IDB_SKILL_A24                   23123
#define IDB_SKILL_A25                   23124
#define IDB_SKILL_A26                   23125
#define IDB_SKILL_A27                   23126
#define IDB_SKILL_A28                   23127
#define IDB_SKILL_A29                   23128
#define IDB_SKILL_A30                   23129
#define IDB_SKILL_S01                   23200
#define IDB_SKILL_S02                   23201
```

```
#define IDB_SKILL_S03                   23202
#define IDB_SKILL_S04                   23203
#define IDB_SKILL_S05                   23204
#define IDB_SKILL_S06                   23205
#define IDB_SKILL_S07                   23206
#define IDB_SKILL_S08                   23207
#define IDB_SKILL_S09                   23208
#define IDB_SKILL_S10                   23209
#define IDB_SKILL_S11                   23210
#define IDB_SKILL_S12                   23211
#define IDB_SKILL_S13                   23212
#define IDB_SKILL_S14                   23213
#define IDB_SKILL_S15                   23214
#define IDB_SKILL_S16                   23215
#define IDB_SKILL_S17                   23216
#define IDB_SKILL_S18                   23217
#define IDB_SKILL_S19                   23218
#define IDB_SKILL_S20                   23219
#define IDB_SKILL_S21                   23220
#define IDB_SKILL_S22                   23221
#define IDB_SKILL_S23                   23222
#define IDB_SKILL_S24                   23223
#define IDB_SKILL_S25                   23224
#define IDB_SKILL_S26                   23225
#define IDB_SKILL_S27                   23226
#define IDB_SKILL_S28                   23227
#define IDB_SKILL_S29                   23228
#define IDB_SKILL_S30                   23229
#define IDB_SKILL_N01                   23300
#define IDB_SKILL_N02                   23301
#define IDB_SKILL_N03                   23302
#define IDB_SKILL_N04                   23303
#define IDB_SKILL_N05                   23304
#define IDB_SKILL_N06                   23305
#define IDB_SKILL_N07                   23306
#define IDB_SKILL_N08                   23307
#define IDB_SKILL_N09                   23308
#define IDB_SKILL_N10                   23309
#define IDB_SKILL_N11                   23310
#define IDB_SKILL_N12                   23311
#define IDB_SKILL_N13                   23312
#define IDB_SKILL_N14                   23313
#define IDB_SKILL_N15                   23314
#define IDB_SKILL_N16                   23315
#define IDB_SKILL_N17                   23316
#define IDB_SKILL_N18                   23317
#define IDB_SKILL_N19                   23318
#define IDB_SKILL_N20                   23319
#define IDB_SKILL_N21                   23320
#define IDB_SKILL_N22                   23321
#define IDB_SKILL_N23                   23322
#define IDB_SKILL_N24                   23323
#define IDB_SKILL_N25                   23324
#define IDB_SKILL_N26                   23325
#define IDB_SKILL_N27                   23326
#define IDB_SKILL_N28                   23327
#define IDB_SKILL_N29                   23328
#define IDB_SKILL_N30                   23329
#define IDB_SKILL_P01                   23400
#define IDB_SKILL_P02                   23401
#define IDB_SKILL_P03                   23402
#define IDB_SKILL_P04                   23403
#define IDB_SKILL_P05                   23404
#define IDB_SKILL_P06                   23405
#define IDB_SKILL_P07                   23406
#define IDB_SKILL_P08                   23407
#define IDB_SKILL_P09                   23408
#define IDB_SKILL_P10                   23409
#define IDB_SKILL_P11                   23410
#define IDB_SKILL_P12                   23411
#define IDB_SKILL_P13                   23412
#define IDB_SKILL_P14                   23413
#define IDB_SKILL_P15                   23414
#define IDB_SKILL_P16                   23415
#define IDB_SKILL_P17                   23416
#define IDB_SKILL_P18                   23417
#define IDB_SKILL_P19                   23418
#define IDB_SKILL_P20                   23419
#define IDB_SKILL_P21                   23420
```

```
#define IDB_SKILL_P22                   23421
#define IDB_SKILL_P23                   23422
#define IDB_SKILL_P24                   23423
#define IDB_SKILL_P25                   23424
#define IDB_SKILL_P26                   23425
#define IDB_SKILL_P27                   23426
#define IDB_SKILL_P28                   23427
#define IDB_SKILL_P29                   23428
#define IDB_SKILL_P30                   23429
#define IDB_SKILL_B01                   23500
#define IDB_SKILL_B02                   23501
#define IDB_SKILL_B03                   23502
#define IDB_SKILL_B04                   23503
#define IDB_SKILL_B05                   23504
#define IDB_SKILL_B06                   23505
#define IDB_SKILL_B07                   23506
#define IDB_SKILL_B08                   23507
#define IDB_SKILL_B09                   23508
#define IDB_SKILL_B10                   23509
#define IDB_SKILL_B11                   23510
#define IDB_SKILL_B12                   23511
#define IDB_SKILL_B13                   23512
#define IDB_SKILL_B14                   23513
#define IDB_SKILL_B15                   23514
#define IDB_SKILL_B16                   23515
#define IDB_SKILL_B17                   23516
#define IDB_SKILL_B18                   23517
#define IDB_SKILL_B19                   23518
#define IDB_SKILL_B20                   23519
#define IDB_SKILL_B21                   23520
#define IDB_SKILL_B22                   23521
#define IDB_SKILL_B23                   23522
#define IDB_SKILL_B24                   23523
#define IDB_SKILL_B25                   23524
#define IDB_SKILL_B26                   23525
#define IDB_SKILL_B27                   23526
#define IDB_SKILL_B28                   23527
#define IDB_SKILL_B29                   23528
#define IDB_SKILL_B30                   23529
#define IDB_ITEM_THEGNASHER             25003
#define IDB_ITEM_DEATHSPADE             25005
#define IDB_ITEM_BERSERKERSHATCHET      25007
#define IDB_ITEM_TANCREDSCROWBILL       25010
#define IDB_ITEM_MINDREND               25011
#define IDB_ITEM_RAKESCAR               25013
#define IDB_ITEM_FECHMARSAXE            25015
#define IDB_ITEM_GORESHOVEL             25017
#define IDB_ITEM_THECHIEFTAN            25019
#define IDB_ITEM_BRAINHEW               25021
#define IDB_ITEM_THEHUMONGOUS           25023
#define IDB_ITEM_IROSTORCH              25025
#define IDB_ITEM_MAELSTROMWRATH         25027
#define IDB_ITEM_GRAVENSPINE            25029
#define IDB_ITEM_INFERNALTORCH          25031
#define IDB_ITEM_UMESLAMENT             25032
#define IDB_ITEM_FELLOAK                25034
#define IDB_ITEM_KNELLSTRIKER           25036
#define IDB_ITEM_CIVERBSCUDGEL          25038
#define IDB_ITEM_RUSTHANDLE             25039
#define IDB_ITEM_MILABREGASROD          25041
#define IDB_ITEM_STOUTNAIL              25044
#define IDB_ITEM_CRUSHFLANGE            25046
#define IDB_ITEM_BLOODRISE              25048
#define IDB_ITEM_THEGENERALSTANDOLIGA   25050
#define IDB_ITEM_IRONSTONE              25052
#define IDB_ITEM_BONESNAP               25054
#define IDB_ITEM_STEELDRIVER            25056
#define IDB_ITEM_RIXOTSKEEN             25058
#define IDB_ITEM_BLOODCRESCENT          25060
#define IDB_ITEM_KRINTIZSSKEWER         25063
#define IDB_ITEM_GLEAMSCYTHE            25065
#define IDB_ITEM_AZUREWRATH             25067
#define IDB_ITEM_ISENHARTSLIGHTBRAND    25069
#define IDB_ITEM_GRISWOLDSEDGE          25070
#define IDB_ITEM_CLEGLAWSTOOTH          25072
#define IDB_ITEM_HELLPLAGUE             25073
#define IDB_ITEM_DEATHSTOUCH            25075
#define IDB_ITEM_SHADOWFANG             25078
#define IDB_ITEM_SOULFLAY               25080
```

```
#define IDB_ITEM_KINEMILSAWL            25082
#define IDB_ITEM_BLACKTONGUE            25084
#define IDB_ITEM_RIPSAW                 25086
#define IDB_ITEM_THEPATRIARCH           25088
#define IDB_ITEM_THEDIGGLER             25092
#define IDB_ITEM_THEJADETANDO           25094
#define IDB_ITEM_IRICESSHARD            25096
#define IDB_ITEM_THEDRAGONCHANG         25107
#define IDB_ITEM_RAZORTINE              25109
#define IDB_ITEM_BLOODTHIEF             25111
#define IDB_ITEM_LANCEOFYAGGAI          25113
#define IDB_ITEM_THETANNRGORER0D        25115
#define IDB_ITEM_DIMOAKSHEW             25117
#define IDB_ITEM_STEELGOAD              25119
#define IDB_ITEM_SOULHARVEST            25121
#define IDB_ITEM_THEBATTLEBRANCH        25123
#define IDB_ITEM_WOESTAVE               25125
#define IDB_ITEM_BANEASH                25129
#define IDB_ITEM_SERPENTLORD            25131
#define IDB_ITEM_LAZARUSSPIRE           25133
#define IDB_ITEM_CATHANSRULE            25135
#define IDB_ITEM_THESALAMANDER          25136
#define IDB_ITEM_ARCANNASDEATHWAND      25138
#define IDB_ITEM_THEIRONLANGBONG        25139
#define IDB_ITEM_PLUCKEYE               25141
#define IDB_ITEM_WITHERSTRING           25143
#define IDB_ITEM_RIMERAVEN              25145
#define IDB_ITEM_PIERCERIB              25147
#define IDB_ITEM_PULLSPITE              25149
#define IDB_ITEM_VIDALASBARB            25151
#define IDB_ITEM_WIZENDRAW              25152
#define IDB_ITEM_ARCTICHORN             25154
#define IDB_ITEM_HELLCLAP               25155
#define IDB_ITEM_BLASTBARK              25157
#define IDB_ITEM_LEADCROW               25159
#define IDB_ITEM_ICHORSTING             25161
#define IDB_ITEM_HELLCAST               25163
#define IDB_ITEM_DOOMSPITTLE            25165
#define IDB_ITEM_RANCIDGAS              25166
#define IDB_ITEM_OIL                    25167
#define IDB_ITEM_CHOKINGGAS             25168
#define IDB_ITEM_EXPLODING              25169
#define IDB_ITEM_STRANGLINGGAS          25170
#define IDB_ITEM_FULM                   25171
#define IDB_ITEM_DECOYGIDBINN           25172
#define IDB_ITEM_THEGIDBINN             25173
#define IDB_ITEM_WIRTSLEG               25174
#define IDB_ITEM_HORADRICMALUS          25175
#define IDB_ITEM_HELLFORGEHAMMER        25176
#define IDB_ITEM_HORADRICSTAFF          25177
#define IDB_ITEM_SHAFTOFHORADRICSTAFF   25178
#define IDB_ITEM_HANDAXE                25179
#define IDB_ITEM_AXE                    25180
#define IDB_ITEM_DOUBLEAXE              25181
#define IDB_ITEM_MILITARYPICK           25182
#define IDB_ITEM_WARAXE                 25183
#define IDB_ITEM_LARGEAXE               25184
#define IDB_ITEM_BROADAXE               25185
#define IDB_ITEM_BATTLEAXE              25186
#define IDB_ITEM_GREATAXE               25187
#define IDB_ITEM_GIANTAXE               25188
#define IDB_ITEM_WAND                   25189
#define IDB_ITEM_YEWWAND                25190
#define IDB_ITEM_BONEWAND               25191
#define IDB_ITEM_GRIMWAND               25192
#define IDB_ITEM_CLUB                   25193
#define IDB_ITEM_SCEPTER                25194
#define IDB_ITEM_GRANDSCEPTER           25195
#define IDB_ITEM_WARSCEPTER             25196
#define IDB_ITEM_SPIKEDCLUB             25197
#define IDB_ITEM_MACE                   25198
#define IDB_ITEM_MORNINGSTAR            25199
#define IDB_ITEM_FLAIL                  25200
#define IDB_ITEM_WARHAMMER              25201
#define IDB_ITEM_MAUL                   25202
#define IDB_ITEM_GREATMAUL              25203
#define IDB_ITEM_SHORTSWORD             25204
#define IDB_ITEM_SCIMITAR               25205
#define IDB_ITEM_SABRE                  25206
```

```
#define IDB_ITEM_FALCHION               25207
#define IDB_ITEM_CRYSTALSWORD           25208
#define IDB_ITEM_BROADSWORD             25209
#define IDB_ITEM_LONGSWORD              25210
#define IDB_ITEM_WARSWORD               25211
#define IDB_ITEM_2HSWORD                25212
#define IDB_ITEM_CLAYMORE               25213
#define IDB_ITEM_GIANTSWORD             25214
#define IDB_ITEM_BASTARDSWORD           25215
#define IDB_ITEM_FLAMBERGE              25216
#define IDB_ITEM_GREATSWORD             25217
#define IDB_ITEM_DAGGER                 25218
#define IDB_ITEM_DIRK                   25219
#define IDB_ITEM_KRIS                   25220
#define IDB_ITEM_BLADE                  25221
#define IDB_ITEM_THROWINGKNIFE          25222
#define IDB_ITEM_THROWINGAXE            25223
#define IDB_ITEM_BALANCEDKNIFE          25224
#define IDB_ITEM_BALANCEDAXE            25225
#define IDB_ITEM_JAVELIN                25226
#define IDB_ITEM_PILUM                  25227
#define IDB_ITEM_SHORTSPEAR             25228
#define IDB_ITEM_GLAIVE                 25229
#define IDB_ITEM_THROWINGSPEAR          25230
#define IDB_ITEM_SPEAR                  25231
#define IDB_ITEM_TRIDENT                25232
#define IDB_ITEM_BRANDISTOCK            25233
#define IDB_ITEM_SPETUM                 25234
#define IDB_ITEM_PIKE                   25235
#define IDB_ITEM_BARDICHE               25236
#define IDB_ITEM_VOULGE                 25237
#define IDB_ITEM_SCYTHE                 25238
#define IDB_ITEM_POLEAXE                25239
#define IDB_ITEM_HALBERD                25240
#define IDB_ITEM_WARSCYTHE              25241
#define IDB_ITEM_SHORTSTAFF             25242
#define IDB_ITEM_LONGSTAFF              25243
#define IDB_ITEM_GNARLEDSTAFF           25244
#define IDB_ITEM_BATTLESTAFF            25245
#define IDB_ITEM_WARSTAFF               25246
#define IDB_ITEM_SHORTBOW               25247
#define IDB_ITEM_HUNTERSBOW             25248
#define IDB_ITEM_LONGBOW                25249
#define IDB_ITEM_COMPOSITEBOW           25250
#define IDB_ITEM_SHORTBATTLEBOW         25251
#define IDB_ITEM_LONGBATTLEBOW          25252
#define IDB_ITEM_SHORTWARBOW            25253
#define IDB_ITEM_LONGWARBOW             25254
#define IDB_ITEM_REPEATINGCROSSBOW      25255
#define IDB_ITEM_LIGHTCROSSBOW          25256
#define IDB_ITEM_CROSSBOW               25257
#define IDB_ITEM_HEAVYCROSSBOW          25258
#define IDB_ITEM_KHALIMSFLAIL           25259
#define IDB_ITEM_KHALIMSWILL            25260
#define IDB_ITEM_INFERNALCRANIUM        25262
#define IDB_ITEM_WARBONET               25263
#define IDB_ITEM_ARCANNASHEAD           25265
#define IDB_ITEM_TARNHELM               25266
#define IDB_ITEM_BERSERKERSHEADGEAR     25268
#define IDB_ITEM_COIFOFGLORY            25269
#define IDB_ITEM_ISENHARTSHORNS         25271
#define IDB_ITEM_DUSKDEEP               25272
#define IDB_ITEM_SIGONSVISOR            25274
#define IDB_ITEM_HOWLTUSK               25275
#define IDB_ITEM_IRATHASCOIL            25277
#define IDB_ITEM_MILABREGASDIADEM       25278
#define IDB_ITEM_UNDEADCROWN            25279
#define IDB_ITEM_ARCTICFURS             25284
#define IDB_ITEM_GREYFORM               25285
#define IDB_ITEM_VIDALASAMBUSH          25287
#define IDB_ITEM_BLINKBATSFORM          25288
#define IDB_ITEM_THECENTURION           25290
#define IDB_ITEM_TWITCHTHROE            25292
#define IDB_ITEM_DARKGLOW               25295
#define IDB_ITEM_HAWKMAIL               25297
#define IDB_ITEM_CATHANSMESH            25299
#define IDB_ITEM_SPARKLINGMAIL          25300
#define IDB_ITEM_ISENHARTSCASE          25302
#define IDB_ITEM_VENOMSWARD             25303
```

```
#define IDB_ITEM_BERSERKERSHAUBERK      25305
#define IDB_ITEM_ICEBLINK               25306
#define IDB_ITEM_BONEFLESH              25308
#define IDB_ITEM_ROCKFLEECE             25310
#define IDB_ITEM_SIGONSSHELTER          25312
#define IDB_ITEM_RATTLECAGE             25313
#define IDB_ITEM_TANCREDSSPINE          25315
#define IDB_ITEM_GOLDSKIN               25316
#define IDB_ITEM_MILABREGASROBE         25318
#define IDB_ITEM_VICTORSSILK            25319
#define IDB_ITEM_ARCANNASFLESH          25321
#define IDB_ITEM_HEAVENLYGARB           25322
#define IDB_ITEM_HSARUSIRONFIST         25324
#define IDB_ITEM_PELTALUNATA            25325
#define IDB_ITEM_CLEGLAWSCLAW           25327
#define IDB_ITEM_UMBRALDISK             25328
#define IDB_ITEM_STORMGUILD             25331
#define IDB_ITEM_MILABREGASORB          25333
#define IDB_ITEM_STEELCLASH             25334
#define IDB_ITEM_SIGONSGUARD            25336
#define IDB_ITEM_BVERRITKEEP            25337
#define IDB_ITEM_ISENHARTSPARRY         25339
#define IDB_ITEM_THEWARD                25340
#define IDB_ITEM_DEATHSHAND             25342
#define IDB_ITEM_THEHANDOFBROC          25343
#define IDB_ITEM_BLOODFIST              25345
#define IDB_ITEM_CLEGLAWSPINCERS        25347
#define IDB_ITEM_CHANCEGUARDS           25348
#define IDB_ITEM_IRATHASCUFF            25350
#define IDB_ITEM_ARCTICMITTS            25351
#define IDB_ITEM_MAGEFIST               25352
#define IDB_ITEM_SIGONSGAUNTLETS        25354
#define IDB_ITEM_FROSTBURN              25355
#define IDB_ITEM_TANCREDSHOBNAILS       25357
#define IDB_ITEM_HOTSPUR                25358
#define IDB_ITEM_GOREFOOT               25360
#define IDB_ITEM_HSARUSIRONHEEL         25362
#define IDB_ITEM_TREADSOFCTHON          25363
#define IDB_ITEM_VIDALASFETLOCK         25365
#define IDB_ITEM_GOBLINTOE              25366
#define IDB_ITEM_SIGONSGREAVES          25368
#define IDB_ITEM_TEARHAUNCH             25369
#define IDB_ITEM_DEATHSGUARD            25371
#define IDB_ITEM_LENYMSCORD             25372
#define IDB_ITEM_ARCTICBINDING          25374
#define IDB_ITEM_SNAKECORD              25375
#define IDB_ITEM_HSARUSIRONSTAY         25377
#define IDB_ITEM_NIGHTSMOKE             25378
#define IDB_ITEM_IRATHASCORD            25380
#define IDB_ITEM_INFERNALBUCKLE         25381
#define IDB_ITEM_GOLDWRAP               25382
#define IDB_ITEM_SIGONSWRAP             25384
#define IDB_ITEM_BLADEBUCKLE            25385
#define IDB_ITEM_TANCREDSSKULL          25387
#define IDB_ITEM_WORMSKULL              25388
#define IDB_ITEM_WALLOFTHEEYELESS       25390
#define IDB_ITEM_SWORDBACKHOLD          25392
#define IDB_ITEM_CAP                    25393
#define IDB_ITEM_SKULLCAP               25394
#define IDB_ITEM_HELM                   25395
#define IDB_ITEM_FULLHELM               25396
#define IDB_ITEM_GREATHELM              25397
#define IDB_ITEM_CROWN                  25398
#define IDB_ITEM_MASK                   25399
#define IDB_ITEM_QUILTED                25400
#define IDB_ITEM_LEATHER                25401
#define IDB_ITEM_HARDLEATHER            25402
#define IDB_ITEM_STUDDEDLEATHER         25403
#define IDB_ITEM_RINGMAIL               25404
#define IDB_ITEM_SCALEMAIL              25405
#define IDB_ITEM_CHAINMAIL              25406
#define IDB_ITEM_BREASTPLATE            25407
#define IDB_ITEM_SPLINTMAIL             25408
#define IDB_ITEM_PLATEMAIL              25409
#define IDB_ITEM_FIELDPLATE             25410
#define IDB_ITEM_GOTHICPLATE            25411
#define IDB_ITEM_FULLPLATE              25412
#define IDB_ITEM_ANCIENTARMOR           25413
#define IDB_ITEM_LIGHTPLATE             25414
```

```
#define IDB_ITEM_BUCKLER                25415
#define IDB_ITEM_SMALLSHIELD            25416
#define IDB_ITEM_LARGESHIELD            25417
#define IDB_ITEM_KITESHIELD             25418
#define IDB_ITEM_TOWERSHIELD            25419
#define IDB_ITEM_GOTHICSHIELD           25420
#define IDB_ITEM_LEATHERGLOVES          25421
#define IDB_ITEM_HEAVYGLOVES            25422
#define IDB_ITEM_CHAINGLOVES            25423
#define IDB_ITEM_LIGHTGAUNTLETS         25424
#define IDB_ITEM_GAUNTLETS              25425
#define IDB_ITEM_LEATHERBOOTS           25426
#define IDB_ITEM_HEAVYBOOTS             25427
#define IDB_ITEM_CHAINBOOTS             25428
#define IDB_ITEM_LIGHTPLATEBOOTS        25429
#define IDB_ITEM_PLATEBOOTS             25430
#define IDB_ITEM_SASH                   25431
#define IDB_ITEM_LIGHTBELT              25432
#define IDB_ITEM_BELT                   25433
#define IDB_ITEM_HEAVYBELT              25434
#define IDB_ITEM_GIRDLE                 25435
#define IDB_ITEM_BONEHELM               25436
#define IDB_ITEM_BONESHIELD             25437
#define IDB_ITEM_SPIKEDSHIELD           25438
#define IDB_ITEM_ELIXEROFVITALITY       25439
#define IDB_ITEM_HPO                    25440
#define IDB_ITEM_MPO                    25441
#define IDB_ITEM_HPF                    25442
#define IDB_ITEM_MPF                    25443
#define IDB_ITEM_STAMINA                25444
#define IDB_ITEM_ANTIDOTE               25445
#define IDB_ITEM_REJUV                  25446
#define IDB_ITEM_FULLREJUV              25447
#define IDB_ITEM_THAWING                25448
#define IDB_ITEM_TOMEBLUE               25449
#define IDB_ITEM_TOMERED                25450
#define IDB_ITEM_TOPOFHORADRICSTAFF     25462
#define IDB_ITEM_GOLD                   25470
#define IDB_ITEM_SCROLLOFINIFUSS        25472
#define IDB_ITEM_ARROWS                 25473
#define IDB_ITEM_TORCH                  25474
#define IDB_ITEM_BOLTS                  25475
#define IDB_ITEM_SCROLLBLUE             25476
#define IDB_ITEM_SCROLLRED              25477
#define IDB_ITEM_SKELETONJAW            25480
#define IDB_ITEM_SKELETONHRN            25482
#define IDB_ITEM_SKELETONTAL            25483
#define IDB_ITEM_SKELETONFLG            25484
#define IDB_ITEM_SKELETONFNG            25485
#define IDB_ITEM_SKELETONQLL            25486
#define IDB_ITEM_SKELETONSCZ            25488
#define IDB_ITEM_SKELETONSOL            25489
#define IDB_ITEM_KEY                    25490
#define IDB_ITEM_BLACKTOWERKEY          25491
#define IDB_ITEM_POTIONOFLIFE           25492
#define IDB_ITEM_JADEFIGURINE           25493
#define IDB_ITEM_GOLDENBIRD             25494
#define IDB_ITEM_LAMESEUSTOME           25495
#define IDB_ITEM_HORADRICCUBE           25496
#define IDB_ITEM_HORADRICSCROLL         25497
#define IDB_ITEM_MOPHISTOSSOULSTONE     25498
#define IDB_ITEM_BOOKOFSKILL            25499
#define IDB_ITEM_EYE                    25500
#define IDB_ITEM_HEART                  25501
#define IDB_ITEM_BRAIN                  25502
#define IDB_ITEM_EAR                    25503
#define IDB_ITEM_CHIPPEDAMETHYST        25504
#define IDB_ITEM_FLAWEDAMETHYST         25505
#define IDB_ITEM_AMETHYST               25506
#define IDB_ITEM_FLAWLESSAMETHYST       25507
#define IDB_ITEM_PERFECTAMETHYST        25508
#define IDB_ITEM_CHIPPEDTOPAZ           25509
#define IDB_ITEM_FLAWEDTOPAZ            25510
#define IDB_ITEM_TOPAZ                  25511
#define IDB_ITEM_FLAWLESSTOPAZ          25512
#define IDB_ITEM_PERFECTTOPAZ           25513
#define IDB_ITEM_CHIPPEDSAPPHIRE        25514
#define IDB_ITEM_FLAWEDSAPPHIRE         25515
#define IDB_ITEM_SAPPHIRE               25516
```

```
#define IDB_ITEM_FLAWLESSSAPPHIRE       25517
#define IDB_ITEM_PERFECTSAPPHIRE        25518
#define IDB_ITEM_CHIPPEDEMERALD         25519
#define IDB_ITEM_FLAWEDEMERALD          25520
#define IDB_ITEM_EMERALD                25521
#define IDB_ITEM_FLAWLESSEMERALD        25522
#define IDB_ITEM_PERFECTEMERALD         25523
#define IDB_ITEM_CHIPPEDRUBY            25524
#define IDB_ITEM_FLAWEDRUBY             25525
#define IDB_ITEM_RUBY                   25526
#define IDB_ITEM_FLAWLESSRUBY           25527
#define IDB_ITEM_PERFECTRUBY            25528
#define IDB_ITEM_CHIPPEDDIAMOND         25529
#define IDB_ITEM_FLAWEDDIAMOND          25530
#define IDB_ITEM_DIAMOND                25531
#define IDB_ITEM_FLAWLESSDIAMOND        25532
#define IDB_ITEM_PERFECTDIAMOND         25533
#define IDB_ITEM_MINORHEALING           25534
#define IDB_ITEM_LIGHTHEALING           25535
#define IDB_ITEM_HEALING                25536
#define IDB_ITEM_GREATERHEALING         25537
#define IDB_ITEM_SUPERHEALING           25538
#define IDB_ITEM_MINORMANA              25539
#define IDB_ITEM_LIGHTMANA              25540
#define IDB_ITEM_MANA                   25541
#define IDB_ITEM_GREATERMANA            25542
#define IDB_ITEM_SUPERMANA              25543
#define IDB_ITEM_CHIPPEDSKULL           25544
#define IDB_ITEM_FLAWEDSKULL            25545
#define IDB_ITEM_SKULL                  25546
#define IDB_ITEM_FLAWLESSSKULL          25547
#define IDB_ITEM_PERFECTSKULL           25548
#define IDR_CLOSE                       40001
#define IDR_RELOAD                      40004
#define IDR_INFO                        40005
#define IDR_OPEN                        40006
#define IDR_SAVE                        40007
#define IDR_TAB2_BeltFullRejuv          40008
#define IDR_TAB2_RepairAll              40011
#define IDR_TAB2_BeltEmpty              40012
#define IDR_TAB2_BeltRejuv              40013
#define IDR_TAB2_BeltSuperHealing       40014
#define IDR_TAB2_BeltGreaterHealing     40015
#define IDR_TAB2_BeltHealing            40016
#define IDR_TAB2_BeltLightHealing       40017
#define IDR_TAB2_BeltMinorHealing       40018
#define IDR_TAB2_BeltSuperMana          40019
#define IDR_TAB2_BeltGreaterMana        40020
#define IDR_TAB2_BeltMana               40021
#define IDR_TAB2_BeltLightMana          40022
#define IDR_TAB2_BeltMinorMana          40023
#define IDR_TAB2_Slot1FullRejuv         40024
#define IDR_TAB2_Slot1Rejuv             40025
#define IDR_TAB2_Slot1SuperHealing      40026
#define IDR_TAB2_Slot1GreaterHealing    40027
#define IDR_TAB2_Slot1Healing           40028
#define IDR_TAB2_Slot1LightHealing      40029
#define IDR_TAB2_Slot1MinorHealing      40030
#define IDR_TAB2_Slot1SuperMana         40031
#define IDR_TAB2_Slot1GreaterMana       40032
#define IDR_TAB2_Slot1Mana              40033
#define IDR_TAB2_Slot1LightMana         40034
#define IDR_TAB2_Slot1MinorMana         40035
#define IDR_TAB2_Slot2FullRejuv         40036
#define IDR_TAB2_Slot2Rejuv             40037
#define IDR_TAB2_Slot2SuperHealing      40038
#define IDR_TAB2_Slot2GreaterHealing    40039
#define IDR_TAB2_Slot2Healing           40040
#define IDR_TAB2_Slot2LightHealing      40041
#define IDR_TAB2_Slot2MinorHealing      40042
#define IDR_TAB2_Slot2SuperMana         40043
#define IDR_TAB2_Slot2GreaterMana       40044
#define IDR_TAB2_Slot2Mana              40045
#define IDR_TAB2_Slot2LightMana         40046
#define IDR_TAB2_Slot2MinorMana         40047
#define IDR_TAB2_Slot3FullRejuv         40048
#define IDR_TAB2_Slot3Rejuv             40049
#define IDR_TAB2_Slot3SuperHealing      40050
#define IDR_TAB2_Slot3GreaterHealing    40051
```

```
#define IDR_TAB2_Slot3Healing           40052
#define IDR_TAB2_Slot3LightHealing      40053
#define IDR_TAB2_Slot3MinorHealing      40054
#define IDR_TAB2_Slot3SuperMana         40055
#define IDR_TAB2_Slot3GreaterMana       40056
#define IDR_TAB2_Slot3Mana              40057
#define IDR_TAB2_Slot3LightMana         40058
#define IDR_TAB2_Slot3MinorMana         40059
#define IDR_TAB2_Slot4FullRejuv         40060
#define IDR_TAB2_Slot4Rejuv             40061
#define IDR_TAB2_Slot4SuperHealing      40062
#define IDR_TAB2_Slot4GreaterHealing    40063
#define IDR_TAB2_Slot4Healing           40064
#define IDR_TAB2_Slot4LightHealing      40065
#define IDR_TAB2_Slot4MinorHealing      40066
#define IDR_TAB2_Slot4SuperMana         40067
#define IDR_TAB2_Slot4GreaterMana       40068
#define IDR_TAB2_Slot4Mana              40069
#define IDR_TAB2_Slot4LightMana         40070
#define IDR_TAB2_Slot4MinorMana         40071
#define IDR_TAB1_SetAllStats40          40080
#define IDR_TAB1_SetAllStats60          40081
#define IDR_TAB1_SetAllStats80          40083
#define IDR_TAB1_SetAllStats100         40085
#define IDR_TAB1_SetAllStats120         40086
#define IDR_TAB1_SetAllStats140         40087
#define IDR_TAB1_SetAllStats160         40088
#define IDR_TAB1_SetAllStats180         40089
#define IDR_TAB1_SetAllStats200         40090
#define IDR_TAB1_SetAllStats250         40091
#define IDR_TAB1_SetAllStats300         40092
#define IDR_TAB1_SetAllStats350         40093
#define IDR_TAB1_SetAllStats400         40094
#define IDR_TAB1_SetAllStats450         40095
#define IDR_TAB1_SetAllStats500         40096
#define IDR_TAB1_RestoreConstitution    40097
#define IDR_TAB1_SetAllConstitution400  40098
#define IDR_TAB1_SetAllConstitution600  40099
#define IDR_TAB1_SetAllConstitution800  40100
#define IDR_TAB1_SetAllConstitution1000 40101
#define IDR_TAB1_SetAllConstitution1500 40102
#define IDR_TAB1_SetAllConstitution2000 40103
#define IDR_TAB1_SetAllConstitution3000 40104
#define IDR_TAB1_SetAllConstitution5000 40105
#define IDR_TAB3_MaximizeAll            40106
#define IDR_TAB3_SetAll0                40107
#define IDR_TAB3_SetAll4                40108
#define IDR_TAB3_SetAll8                40109
#define IDR_TAB3_SetAll12               40110
#define IDR_TAB3_SetAll16               40111
#define IDR_TAB3_SetAll20               40112
#define IDR_TAB5_ActivateHereAll        40132
#define IDR_TAB5_ActivateAllAll         40133
#define IDR_TAB5_DeactivateHereAll      40134
#define IDR_TAB5_DeactivateAllAll       40135
#define IDR_NEW                         40136
#define IDR_UOPTIONS                    40139
#define IDR_EOPTIONS                    40140
#define IDR_TEXTFILE                    40141
#define IDS_SKILL_A01                   43100
#define IDS_SKILL_A02                   43101
#define IDS_SKILL_A03                   43102
#define IDS_SKILL_A04                   43103
#define IDS_SKILL_A05                   43104
#define IDS_SKILL_A06                   43105
#define IDS_SKILL_A07                   43106
#define IDS_SKILL_A08                   43107
#define IDS_SKILL_A09                   43108
#define IDS_SKILL_A10                   43109
#define IDS_SKILL_A11                   43110
#define IDS_SKILL_A12                   43111
#define IDS_SKILL_A13                   43112
#define IDS_SKILL_A14                   43113
#define IDS_SKILL_A15                   43114
#define IDS_SKILL_A16                   43115
#define IDS_SKILL_A17                   43116
#define IDS_SKILL_A18                   43117
#define IDS_SKILL_A19                   43118
#define IDS_SKILL_A20                   43119
```

```
#define IDS_SKILL_A21                   43120
#define IDS_SKILL_A22                   43121
#define IDS_SKILL_A23                   43122
#define IDS_SKILL_A24                   43123
#define IDS_SKILL_A25                   43124
#define IDS_SKILL_A26                   43125
#define IDS_SKILL_A27                   43126
#define IDS_SKILL_A28                   43127
#define IDS_SKILL_A29                   43128
#define IDS_SKILL_A30                   43129
#define IDS_SKILL_S01                   43200
#define IDS_SKILL_S02                   43201
#define IDS_SKILL_S03                   43202
#define IDS_SKILL_S04                   43203
#define IDS_SKILL_S05                   43204
#define IDS_SKILL_S06                   43205
#define IDS_SKILL_S07                   43206
#define IDS_SKILL_S08                   43207
#define IDS_SKILL_S09                   43208
#define IDS_SKILL_S10                   43209
#define IDS_SKILL_S11                   43210
#define IDS_SKILL_S12                   43211
#define IDS_SKILL_S13                   43212
#define IDS_SKILL_S14                   43213
#define IDS_SKILL_S15                   43214
#define IDS_SKILL_S16                   43215
#define IDS_SKILL_S17                   43216
#define IDS_SKILL_S18                   43217
#define IDS_SKILL_S19                   43218
#define IDS_SKILL_S20                   43219
#define IDS_SKILL_S21                   43220
#define IDS_SKILL_S22                   43221
#define IDS_SKILL_S23                   43222
#define IDS_SKILL_S24                   43223
#define IDS_SKILL_S25                   43224
#define IDS_SKILL_S26                   43225
#define IDS_SKILL_S27                   43226
#define IDS_SKILL_S28                   43227
#define IDS_SKILL_S29                   43228
#define IDS_SKILL_S30                   43229
#define IDS_SKILL_N01                   43300
#define IDS_SKILL_N02                   43301
#define IDS_SKILL_N03                   43302
#define IDS_SKILL_N05                   43303
#define IDS_SKILL_N04                   43304
#define IDS_SKILL_N06                   43305
#define IDS_SKILL_N07                   43306
#define IDS_SKILL_N08                   43307
#define IDS_SKILL_N09                   43308
#define IDS_SKILL_N10                   43309
#define IDS_SKILL_N11                   43310
#define IDS_SKILL_N12                   43311
#define IDS_SKILL_N13                   43312
#define IDS_SKILL_N14                   43313
#define IDS_SKILL_N15                   43314
#define IDS_SKILL_N16                   43315
#define IDS_SKILL_N17                   43316
#define IDS_SKILL_N18                   43317
#define IDS_SKILL_N19                   43318
#define IDS_SKILL_N20                   43319
#define IDS_SKILL_N21                   43320
#define IDS_SKILL_N22                   43321
#define IDS_SKILL_N23                   43322
#define IDS_SKILL_N24                   43323
#define IDS_SKILL_N25                   43324
#define IDS_SKILL_N26                   43325
#define IDS_SKILL_N27                   43326
#define IDS_SKILL_N28                   43327
#define IDS_SKILL_N29                   43328
#define IDS_SKILL_N30                   43329
#define IDS_SKILL_P01                   43400
#define IDS_SKILL_P02                   43401
#define IDS_SKILL_P03                   43402
#define IDS_SKILL_P04                   43403
#define IDS_SKILL_P05                   43404
#define IDS_SKILL_P06                   43405
#define IDS_SKILL_P07                   43406
#define IDS_SKILL_P08                   43407
#define IDS_SKILL_P09                   43408
```

```
#define IDS_SKILL_P10                   43409
#define IDS_SKILL_P11                   43410
#define IDS_SKILL_P12                   43411
#define IDS_SKILL_P13                   43412
#define IDS_SKILL_P14                   43413
#define IDS_SKILL_P15                   43414
#define IDS_SKILL_P16                   43415
#define IDS_SKILL_P17                   43416
#define IDS_SKILL_P18                   43417
#define IDS_SKILL_P19                   43418
#define IDS_SKILL_P20                   43419
#define IDS_SKILL_P21                   43420
#define IDS_SKILL_P22                   43421
#define IDS_SKILL_P23                   43422
#define IDS_SKILL_P24                   43423
#define IDS_SKILL_P25                   43424
#define IDS_SKILL_P26                   43425
#define IDS_SKILL_P27                   43426
#define IDS_SKILL_P28                   43427
#define IDS_SKILL_P29                   43428
#define IDS_SKILL_P30                   43429
#define IDS_SKILL_B01                   43500
#define IDS_SKILL_B02                   43501
#define IDS_SKILL_B03                   43502
#define IDS_SKILL_B04                   43503
#define IDS_SKILL_B05                   43504
#define IDS_SKILL_B06                   43505
#define IDS_SKILL_B07                   43506
#define IDS_SKILL_B08                   43507
#define IDS_SKILL_B09                   43508
#define IDS_SKILL_B10                   43509
#define IDS_SKILL_B11                   43510
#define IDS_SKILL_B12                   43511
#define IDS_SKILL_B13                   43512
#define IDS_SKILL_B14                   43513
#define IDS_SKILL_B15                   43514
#define IDS_SKILL_B16                   43515
#define IDS_SKILL_B17                   43516
#define IDS_SKILL_B18                   43517
#define IDS_SKILL_B19                   43518
#define IDS_SKILL_B20                   43519
#define IDS_SKILL_B21                   43520
#define IDS_SKILL_B22                   43521
#define IDS_SKILL_B23                   43522
#define IDS_SKILL_B24                   43523
#define IDS_SKILL_B25                   43524
#define IDS_SKILL_B26                   43525
#define IDS_SKILL_B27                   43526
#define IDS_SKILL_B28                   43527
#define IDS_SKILL_B29                   43528
#define IDS_SKILL_B30                   43529
#define IDS_QUEST11                     44001
#define IDS_QUEST12                     44002
#define IDS_QUEST13                     44003
#define IDS_QUEST14                     44004
#define IDS_QUEST15                     44005
#define IDS_QUEST16                     44006
#define IDS_QUEST21                     44007
#define IDS_QUEST22                     44008
#define IDS_QUEST23                     44009
#define IDS_QUEST24                     44010
#define IDS_QUEST25                     44011
#define IDS_QUEST26                     44012
#define IDS_QUEST31                     44013
#define IDS_QUEST32                     44014
#define IDS_QUEST33                     44015
#define IDS_QUEST34                     44016
#define IDS_QUEST35                     44017
#define IDS_QUEST36                     44018
#define IDS_QUEST41                     44019
#define IDS_QUEST42                     44020
#define IDS_QUEST43                     44021

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NO_MFC                     1
#define _APS_3D_CONTROLS                     1
```

```
#define _APS_NEXT_RESOURCE_VALUE        351
#define _APS_NEXT_COMMAND_VALUE         40142
#define _APS_NEXT_CONTROL_VALUE         1388
#define _APS_NEXT_SYMED_VALUE           120
#endif
#endif
```

```
//Microsoft Developer Studio generated resource script.
//
#include "ResourceIDs.h"

#define APSTUDIO_READONLY_SYMBOLS
////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 2 resource.
//
#include "afxres.h"
#include <dlgs.h>

////////////////////////////////////////////////////////////////////////////
#undef APSTUDIO_READONLY_SYMBOLS

////////////////////////////////////////////////////////////////////////////
// Neutral resources

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_NEU)
#ifdef _WIN32
LANGUAGE LANG_NEUTRAL, SUBLANG_NEUTRAL
#pragma code_page(1252)
#endif //_WIN32

////////////////////////////////////////////////////////////////////////////
//
// Dialog
//

IDD_TAB3 DIALOGEX 0, 0, 310, 226
STYLE WS_CHILD
FONT 8, "MS Sans Serif", 0, 0, 0x1
BEGIN
    GROUPBOX        "A",IDC_TAB3_Frame_A,4,6,96,139,BS_CENTER,
                    WS_EX_TRANSPARENT
    GROUPBOX        "B",IDC_TAB3_Frame_B,104,6,102,139,BS_CENTER,
                    WS_EX_TRANSPARENT
    GROUPBOX        "C",IDC_TAB3_Frame_C,210,6,96,139,BS_CENTER,
                    WS_EX_TRANSPARENT
    CONTROL         "",IDC_TAB3_Bitmap_A0,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,10,18,21,20
    CTEXT           "00",IDC_TAB3_Edit_A0,33,28,13,10,SS_NOPREFIX |
                    SS_REALSIZEIMAGE | 0x2000,WS_EX_STATICEDGE
    CONTROL         "",IDC_TAB3_Bitmap_A1,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,60,18,15,13
    CTEXT           "00",IDC_TAB3_Edit_A1,83,28,13,10,SS_NOPREFIX |
                    SS_REALSIZEIMAGE | 0x2000,WS_EX_STATICEDGE
    CONTROL         "",IDC_TAB3_Bitmap_A2,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,10,43,15,13
    CTEXT           "00",IDC_TAB3_Edit_A2,33,53,13,10,SS_NOPREFIX |
                    SS_REALSIZEIMAGE | 0x2000,WS_EX_STATICEDGE
    CONTROL         "",IDC_TAB3_Bitmap_A3,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,60,43,21,20
    CTEXT           "00",IDC_TAB3_Edit_A3,83,53,13,10,SS_NOPREFIX |
                    SS_REALSIZEIMAGE | 0x2000,WS_EX_STATICEDGE
    CONTROL         "",IDC_TAB3_Bitmap_A4,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,10,68,15,13
    CTEXT           "00",IDC_TAB3_Edit_A4,33,78,13,10,SS_NOPREFIX |
                    SS_REALSIZEIMAGE | 0x2000,WS_EX_STATICEDGE
    CONTROL         "",IDC_TAB3_Bitmap_A5,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,60,68,15,13
    CTEXT           "00",IDC_TAB3_Edit_A5,83,78,13,10,SS_NOPREFIX |
                    SS_REALSIZEIMAGE | 0x2000,WS_EX_STATICEDGE
    CONTROL         "",IDC_TAB3_Bitmap_A6,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,10,93,15,13
    CTEXT           "00",IDC_TAB3_Edit_A6,33,103,13,10,SS_NOPREFIX |
                    SS_REALSIZEIMAGE | 0x2000,WS_EX_STATICEDGE
    CONTROL         "",IDC_TAB3_Bitmap_A7,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,60,93,15,13
    CTEXT           "00",IDC_TAB3_Edit_A7,83,103,13,10,SS_NOPREFIX |
                    SS_REALSIZEIMAGE | 0x2000,WS_EX_STATICEDGE
    CONTROL         "",IDC_TAB3_Bitmap_A8,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,10,118,15,13
    CTEXT           "00",IDC_TAB3_Edit_A8,33,128,13,10,SS_NOPREFIX |
                    SS_REALSIZEIMAGE | 0x2000,WS_EX_STATICEDGE
    CONTROL         "",IDC_TAB3_Bitmap_A9,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,60,118,15,13
    CTEXT           "00",IDC_TAB3_Edit_A9,83,128,13,10,SS_NOPREFIX |
                    SS_REALSIZEIMAGE | 0x2000,WS_EX_STATICEDGE
```

```
CONTROL         "",IDC_TAB3_Bitmap_B0,"Static",SS_BITMAP | SS_NOTIFY |
                WS_TABSTOP,110,17,21,20
CTEXT           "00",IDC_TAB3_Edit_B0,134,27,13,10,SS_NOPREFIX |
                SS_REALSIZEIMAGE | 0x2000,WS_EX_STATICEDGE
CONTROL         "",IDC_TAB3_Bitmap_B1,"Static",SS_BITMAP | SS_NOTIFY |
                WS_TABSTOP,165,18,15,13
CTEXT           "00",IDC_TAB3_Edit_B1,189,28,13,10,SS_NOPREFIX |
                SS_REALSIZEIMAGE | 0x2000,WS_EX_STATICEDGE
CONTROL         "",IDC_TAB3_Bitmap_B2,"Static",SS_BITMAP | SS_NOTIFY |
                WS_TABSTOP,110,43,15,13
CTEXT           "00",IDC_TAB3_Edit_B2,134,53,13,10,SS_NOPREFIX |
                SS_REALSIZEIMAGE | 0x2000,WS_EX_STATICEDGE
CONTROL         "",IDC_TAB3_Bitmap_B3,"Static",SS_BITMAP | SS_NOTIFY |
                WS_TABSTOP,165,43,15,13
CTEXT           "00",IDC_TAB3_Edit_B3,189,53,13,10,SS_NOPREFIX |
                SS_REALSIZEIMAGE | 0x2000,WS_EX_STATICEDGE
CONTROL         "",IDC_TAB3_Bitmap_B4,"Static",SS_BITMAP | SS_NOTIFY |
                WS_TABSTOP,110,68,15,13
CTEXT           "00",IDC_TAB3_Edit_B4,134,78,13,10,SS_NOPREFIX |
                SS_REALSIZEIMAGE | 0x2000,WS_EX_STATICEDGE
CONTROL         "",IDC_TAB3_Bitmap_B5,"Static",SS_BITMAP | SS_NOTIFY |
                WS_TABSTOP,165,68,15,13
CTEXT           "00",IDC_TAB3_Edit_B5,189,78,13,10,SS_NOPREFIX |
                SS_REALSIZEIMAGE | 0x2000,WS_EX_STATICEDGE
CONTROL         "",IDC_TAB3_Bitmap_B6,"Static",SS_BITMAP | SS_NOTIFY |
                WS_TABSTOP,110,93,15,13
CTEXT           "00",IDC_TAB3_Edit_B6,134,103,13,10,SS_NOPREFIX |
                SS_REALSIZEIMAGE | 0x2000,WS_EX_STATICEDGE
CONTROL         "",IDC_TAB3_Bitmap_B7,"Static",SS_BITMAP | SS_NOTIFY |
                WS_TABSTOP,165,93,15,13
CTEXT           "00",IDC_TAB3_Edit_B7,189,103,13,10,SS_NOPREFIX |
                SS_REALSIZEIMAGE | 0x2000,WS_EX_STATICEDGE
CONTROL         "",IDC_TAB3_Bitmap_B8,"Static",SS_BITMAP | SS_NOTIFY |
                WS_TABSTOP,110,118,15,13
CTEXT           "00",IDC_TAB3_Edit_B8,134,128,13,10,SS_NOPREFIX |
                SS_REALSIZEIMAGE | 0x2000,WS_EX_STATICEDGE
CONTROL         "",IDC_TAB3_Bitmap_B9,"Static",SS_BITMAP | SS_NOTIFY |
                WS_TABSTOP,165,118,15,13
CTEXT           "00",IDC_TAB3_Edit_B9,189,128,13,10,SS_NOPREFIX |
                SS_REALSIZEIMAGE | 0x2000,WS_EX_STATICEDGE
CONTROL         "",IDC_TAB3_Bitmap_C0,"Static",SS_BITMAP | SS_NOTIFY |
                WS_TABSTOP,216,18,21,20
CTEXT           "00",IDC_TAB3_Edit_C0,239,28,13,10,SS_NOPREFIX |
                SS_REALSIZEIMAGE | 0x2000,WS_EX_STATICEDGE
CONTROL         "",IDC_TAB3_Bitmap_C1,"Static",SS_BITMAP | SS_NOTIFY |
                WS_TABSTOP,265,18,15,13
CTEXT           "00",IDC_TAB3_Edit_C1,289,28,13,10,SS_NOPREFIX |
                SS_REALSIZEIMAGE | 0x2000,WS_EX_STATICEDGE
CONTROL         "",IDC_TAB3_Bitmap_C2,"Static",SS_BITMAP | SS_NOTIFY |
                WS_TABSTOP,215,43,15,13
CTEXT           "00",IDC_TAB3_Edit_C2,239,53,13,10,SS_NOPREFIX |
                SS_REALSIZEIMAGE | 0x2000,WS_EX_STATICEDGE
CONTROL         "",IDC_TAB3_Bitmap_C3,"Static",SS_BITMAP | SS_NOTIFY |
                WS_TABSTOP,265,43,15,13
CTEXT           "00",IDC_TAB3_Edit_C3,289,53,13,10,SS_NOPREFIX |
                SS_REALSIZEIMAGE | 0x2000,WS_EX_STATICEDGE
CONTROL         "",IDC_TAB3_Bitmap_C4,"Static",SS_BITMAP | SS_NOTIFY |
                WS_TABSTOP,215,68,15,13
CTEXT           "00",IDC_TAB3_Edit_C4,239,78,13,10,SS_NOPREFIX |
                SS_REALSIZEIMAGE | 0x2000,WS_EX_STATICEDGE
CONTROL         "",IDC_TAB3_Bitmap_C5,"Static",SS_BITMAP | SS_NOTIFY |
                WS_TABSTOP,265,68,15,13
CTEXT           "00",IDC_TAB3_Edit_C5,289,78,13,10,SS_NOPREFIX |
                SS_REALSIZEIMAGE | 0x2000,WS_EX_STATICEDGE
CONTROL         "",IDC_TAB3_Bitmap_C6,"Static",SS_BITMAP | SS_NOTIFY |
                WS_TABSTOP,215,93,15,13
CTEXT           "00",IDC_TAB3_Edit_C6,239,103,13,10,SS_NOPREFIX |
                SS_REALSIZEIMAGE | 0x2000,WS_EX_STATICEDGE
CONTROL         "",IDC_TAB3_Bitmap_C7,"Static",SS_BITMAP | SS_NOTIFY |
                WS_TABSTOP,265,93,15,13
CTEXT           "00",IDC_TAB3_Edit_C7,289,103,13,10,SS_NOPREFIX |
                SS_REALSIZEIMAGE | 0x2000,WS_EX_STATICEDGE
CONTROL         "",IDC_TAB3_Bitmap_C8,"Static",SS_BITMAP | SS_NOTIFY |
                WS_TABSTOP,215,118,15,13
CTEXT           "00",IDC_TAB3_Edit_C8,239,128,13,10,SS_NOPREFIX |
                SS_REALSIZEIMAGE | 0x2000,WS_EX_STATICEDGE
CONTROL         "",IDC_TAB3_Bitmap_C9,"Static",SS_BITMAP | SS_NOTIFY |
                WS_TABSTOP,265,118,15,13
CTEXT           "00",IDC_TAB3_Edit_C9,289,128,13,10,SS_NOPREFIX |
```

```
                    SS_REALSIZEIMAGE | 0x2000,WS_EX_STATICEDGE
    GROUPBOX        "Properties",IDC_STATIC,5,150,210,70,BS_CENTER,
                    WS_EX_TRANSPARENT
    CONTROL         "",IDC_TAB3_Prop_Bitmap,"Static",SS_BITMAP | NOT
                    WS_VISIBLE,25,165,21,20
    CONTROL         "Slider1",IDC_TAB3_Track,"msctls_trackbar32",
                    TBS_AUTOTICKS | TBS_TOP | NOT WS_VISIBLE | WS_TABSTOP,65,
                    165,145,20
    EDITTEXT        IDC_TAB3_Prop_Edit,10,195,50,20,ES_CENTER | ES_MULTILINE |
                    ES_READONLY | NOT WS_VISIBLE | NOT WS_BORDER | NOT
                    WS_TABSTOP
    EDITTEXT        IDC_TAB3_Prop_Effect,65,195,145,20,ES_CENTER |
                    ES_MULTILINE | ES_READONLY | NOT WS_VISIBLE | NOT
                    WS_BORDER | NOT WS_TABSTOP
    GROUPBOX        "Available Skill Points",IDC_STATIC,220,185,85,35,
                    BS_CENTER,WS_EX_TRANSPARENT
    EDITTEXT        IDC_TAB3_Bonus,230,199,45,12,ES_CENTER | ES_AUTOHSCROLL |
                    ES_NUMBER
    CONTROL         22000,IDC_TAB3_BonusPlus,"Static",SS_BITMAP | SS_NOTIFY,
                    280,197,17,16
    PUSHBUTTON      "Batch Action",IDC_TAB3_Batch,225,155,75,20
END

IDD_TAB0 DIALOG DISCARDABLE  0, 0, 310, 226
STYLE WS_CHILD
FONT 8, "MS Sans Serif"
BEGIN
    CONTROL         "",IDC_TAB0_BmpJamella,"Static",SS_BITMAP | SS_NOTIFY |
                    SS_REALSIZEIMAGE,5,20,304,185
    CTEXT           "Version",IDC_TAB0_Version,5,210,300,8,SS_NOTIFY
END

IDD_TAB1 DIALOGEX 0, 0, 310, 226
STYLE WS_CHILD
FONT 8, "MS Sans Serif", 0, 0, 0x1
BEGIN
    GROUPBOX        "Name",IDC_STATIC,4,6,181,25,BS_CENTER,WS_EX_TRANSPARENT
    CTEXT           "Name",IDC_TAB1_Name,10,18,130,10,SS_NOPREFIX |
                    SS_REALSIZEIMAGE,WS_EX_STATICEDGE
    GROUPBOX        "Class",IDC_STATIC,190,6,115,25,BS_CENTER,
                    WS_EX_TRANSPARENT
    COMBOBOX        IDC_TAB1_Class,194,16,107,69,CBS_DROPDOWNLIST | WS_GROUP |
                    WS_TABSTOP
    GROUPBOX        "Level",IDC_STATIC,4,36,50,25,BS_CENTER,
                    WS_EX_TRANSPARENT
    EDITTEXT        IDC_TAB1_Level,9,46,39,12,ES_CENTER | ES_NUMBER |
                    WS_GROUP
    GROUPBOX        "Experience",IDC_STATIC,58,36,112,25,BS_CENTER,
                    WS_EX_TRANSPARENT
    EDITTEXT        IDC_TAB1_Experience,63,46,101,12,ES_CENTER | ES_NUMBER |
                    WS_GROUP
    GROUPBOX        "Characteristics",IDC_STATIC,5,65,130,138,BS_CENTER,
                    WS_EX_TRANSPARENT
    LTEXT           "Strength",IDC_STATIC,10,83,28,8
    EDITTEXT        IDC_TAB1_Char1,64,81,40,12,ES_CENTER | ES_AUTOHSCROLL |
                    ES_NUMBER | WS_GROUP
    CONTROL         22000,IDC_TAB1_Plus1,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,110,79,17,16
    LTEXT           "Dexterity",IDC_STATIC,10,105,28,8
    EDITTEXT        IDC_TAB1_Char2,64,103,40,12,ES_CENTER | ES_AUTOHSCROLL |
                    ES_NUMBER | WS_GROUP
    CONTROL         22000,IDC_TAB1_Plus2,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,110,101,17,16
    LTEXT           "Vitality",IDC_STATIC,10,127,21,8
    EDITTEXT        IDC_TAB1_Char3,64,125,40,12,ES_CENTER | ES_AUTOHSCROLL |
                    ES_NUMBER | WS_GROUP
    CONTROL         22000,IDC_TAB1_Plus3,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,110,123,17,16
    LTEXT           "Energy",IDC_STATIC,10,149,23,8
    EDITTEXT        IDC_TAB1_Char4,64,147,40,12,ES_CENTER | ES_AUTOHSCROLL |
                    ES_NUMBER | WS_GROUP
    CONTROL         22000,IDC_TAB1_Plus4,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,110,145,17,16
    LTEXT           "Available Points",IDC_STATIC,10,171,51,8
    EDITTEXT        IDC_TAB1_Char5,64,169,40,12,ES_CENTER | ES_AUTOHSCROLL |
                    ES_NUMBER | WS_GROUP
    CONTROL         22000,IDC_TAB1_Plus5,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,110,167,17,16
    GROUPBOX        "Character Type",IDC_STATIC,140,65,95,26,BS_CENTER,
```

```
                    WS_EX_TRANSPARENT
    CONTROL         "Hardcore",IDC_TAB1_Hardcore,"Button",BS_AUTOCHECKBOX |
                    BS_NOTIFY | WS_GROUP | WS_TABSTOP,145,75,45,10
    CONTROL         "Dead",IDC_TAB1_Dead,"Button",BS_AUTOCHECKBOX |
                    BS_NOTIFY | WS_TABSTOP,195,75,33,10
    GROUPBOX        "Constitution",IDC_STATIC,140,96,165,70,BS_CENTER,
                    WS_EX_TRANSPARENT
    LTEXT           "Health",IDC_STATIC,145,110,22,8
    EDITTEXT        IDC_TAB1_Health,205,108,35,12,ES_CENTER | ES_AUTOHSCROLL |
                    ES_NUMBER | WS_GROUP
    CTEXT           "/",IDC_STATIC,248,110,8,8
    EDITTEXT        IDC_TAB1_HealthMax,264,108,35,12,ES_CENTER |
                    ES_AUTOHSCROLL | ES_NUMBER
    LTEXT           "Stamina",IDC_STATIC,145,130,26,8
    EDITTEXT        IDC_TAB1_Stamina,205,128,35,12,ES_CENTER |
                    ES_AUTOHSCROLL | ES_NUMBER | WS_GROUP
    CTEXT           "/",IDC_STATIC,248,130,8,8
    EDITTEXT        IDC_TAB1_StaminaMax,264,128,35,12,ES_CENTER |
                    ES_AUTOHSCROLL | ES_NUMBER
    LTEXT           "Mana",IDC_STATIC,145,150,19,8
    EDITTEXT        IDC_TAB1_Mana,205,148,35,12,ES_CENTER | ES_AUTOHSCROLL |
                    ES_NUMBER | WS_GROUP
    CTEXT           "/",IDC_STATIC,248,150,8,8
    EDITTEXT        IDC_TAB1_ManaMax,264,148,35,12,ES_CENTER |
                    ES_AUTOHSCROLL | ES_NUMBER
    GROUPBOX        "Gold",IDC_STATIC,140,171,165,49,BS_CENTER,
                    WS_EX_TRANSPARENT
    LTEXT           "Person",IDC_STATIC,145,184,23,8
    EDITTEXT        IDC_TAB1_GoldPerson,180,182,35,12,ES_AUTOHSCROLL |
                    ES_NUMBER | WS_GROUP
    LTEXT           "Stash",IDC_STATIC,145,204,19,8
    EDITTEXT        IDC_TAB1_GoldStash,180,202,35,12,ES_AUTOHSCROLL |
                    ES_NUMBER | WS_GROUP
    LTEXT           "/ ",IDC_TAB1_MaxGoldPerson,226,184,40,8
    LTEXT           "/ ",IDC_TAB1_MaxGoldStash,226,204,40,8
    PUSHBUTTON      "Max",IDC_TAB1_SetMaxGoldPerson,275,182,20,12
    PUSHBUTTON      "Max",IDC_TAB1_SetMaxGoldStash,275,202,20,12
    GROUPBOX        "Starting Town",IDC_STATIC,174,36,131,25,BS_CENTER,
                    WS_EX_TRANSPARENT
    COMBOBOX        IDC_TAB1_StartTown,180,46,119,119,CBS_DROPDOWNLIST |
                    WS_VSCROLL | WS_TABSTOP
    GROUPBOX        "Max Difficulty",IDC_STATIC,240,65,65,26,BS_CENTER,
                    WS_EX_TRANSPARENT
    COMBOBOX        IDC_TAB1_Difficulty,244,75,57,55,CBS_DROPDOWNLIST |
                    WS_VSCROLL | WS_TABSTOP
    PUSHBUTTON      "Batch Action",IDC_TAB1_Batch,45,208,50,12
    CONTROL         "Link Stats to Constitution",IDC_TAB1_StatsLink,"Button",
                    BS_AUTOCHECKBOX | WS_TABSTOP,23,188,94,10
    PUSHBUTTON      "Rename",IDC_TAB1_Rename,145,16,35,13
END

IDD_TAB2 DIALOGEX 0, 0, 456, 335
STYLE WS_CHILD
FONT 8, "MS Sans Serif"
BEGIN
    GROUPBOX        "Inventory",IDC_STATIC,6,3,220,238,BS_CENTER,
                    WS_EX_TRANSPARENT
    GROUPBOX        "Selected Item",IDC_STATIC,230,3,220,123,BS_CENTER,
                    WS_EX_TRANSPARENT
    GROUPBOX        "Horadric Cube",IDC_STATIC,6,243,63,86,BS_CENTER,
                    WS_EX_TRANSPARENT
    GROUPBOX        "Belt",IDC_STATIC,72,243,85,86,BS_CENTER,
                    WS_EX_TRANSPARENT
    GROUPBOX        "Stash",IDC_STATIC,160,243,122,86,BS_CENTER,
                    WS_EX_TRANSPARENT
    CONTROL         "ItemTree",IDC_TAB2_ItemTree,"SysTreeView32",
                    TVS_HASBUTTONS | TVS_HASLINES | TVS_LINESATROOT |
                    TVS_NOTOOLTIPS | TVS_FULLROWSELECT | WS_TABSTOP,288,139,
                    160,185,WS_EX_STATICEDGE
    CONTROL         264,IDC_TAB2_Body,"Static",SS_BITMAP | SS_REALSIZEIMAGE,
                    9,12,213,152
    CONTROL         262,IDC_TAB2_Inv,"Static",SS_BITMAP | SS_REALSIZEIMAGE,
                    20,166,194,72
    CONTROL         261,IDC_TAB2_Cube,"Static",SS_BITMAP | SS_REALSIZEIMAGE,
                    8,252,59,72
    CONTROL         263,IDC_TAB2_Belt,"Static",SS_BITMAP | SS_REALSIZEIMAGE,
                    76,252,78,72
    CONTROL         260,IDC_TAB2_Stash,"Static",SS_BITMAP | SS_REALSIZEIMAGE,
                    163,252,117,72
```

```
    CONTROL         "Expert Toolbox",IDC_TAB2_ExpertMode,"Button",
                    BS_AUTOCHECKBOX | WS_TABSTOP,380,112,63,10
    PUSHBUTTON      "Save Item",IDC_TAB2_Save,230,130,51,12
    PUSHBUTTON      "Load Item",IDC_TAB2_Load,230,142,51,12,BS_CENTER |
                    BS_VCENTER
    CONTROL         287,IDC_TAB2_CopyBuffer,"Static",SS_BITMAP |
                    SS_REALSIZEIMAGE,235,166,39,72
    GROUPBOX        "Built-In Item Tree (Drag into Inventory)",IDC_STATIC,
                    285,130,165,199,BS_CENTER,WS_EX_TRANSPARENT
    GROUPBOX        "Copy Buffer",IDC_STATIC,230,156,52,85,BS_CENTER,
                    WS_EX_TRANSPARENT
    PUSHBUTTON      "Randomize",IDC_TAB2_AttrRandom,235,110,50,12,
                    WS_DISABLED
    CONTROL         "",IDC_TAB2_RichText,"RICHEDIT",ES_CENTER | ES_MULTILINE |
                    ES_AUTOVSCROLL | ES_READONLY | ES_NUMBER | WS_VSCROLL |
                    WS_TABSTOP,235,12,207,96,WS_EX_STATICEDGE
    PUSHBUTTON      "Batch Action",IDC_TAB2_Batch,320,110,50,12
    PUSHBUTTON      "<",IDC_TAB2_HistoryBack,290,110,10,12,WS_DISABLED
    PUSHBUTTON      ">",IDC_TAB2_HistoryNext,300,110,10,12,WS_DISABLED
END

IDD_TAB4 DIALOGEX 0, 0, 310, 226
STYLE WS_CHILD
FONT 8, "MS Sans Serif", 0, 0, 0x1
BEGIN
    CONTROL         "Normal",IDC_TAB4_Diff1,"Button",BS_AUTORADIOBUTTON |
                    WS_GROUP | WS_TABSTOP,10,16,38,10
    CONTROL         "Nightmare",IDC_TAB4_Diff2,"Button",BS_AUTORADIOBUTTON |
                    WS_TABSTOP,10,33,48,10
    CONTROL         "Hell",IDC_TAB4_Diff3,"Button",BS_AUTORADIOBUTTON |
                    WS_TABSTOP,10,50,28,10
    GROUPBOX        "Act Activation && Selection",IDC_STATIC,70,5,110,60,
                    BS_CENTER | WS_GROUP,WS_EX_TRANSPARENT
    GROUPBOX        "Difficulty",IDC_STATIC,4,5,61,60,BS_CENTER,
                    WS_EX_TRANSPARENT
    GROUPBOX        "Change all",IDC_STATIC,184,5,121,45,BS_CENTER,
                    WS_EX_TRANSPARENT
    CONTROL         "Act I",IDC_TAB4_Act1,"Button",BS_AUTORADIOBUTTON |
                    WS_GROUP | WS_TABSTOP,78,15,31,10
    CONTROL         "Act II",IDC_TAB4_Act2,"Button",BS_AUTORADIOBUTTON |
                    WS_TABSTOP,78,27,33,10
    CONTROL         "Act III",IDC_TAB4_Act3,"Button",BS_AUTORADIOBUTTON |
                    WS_TABSTOP,78,39,35,10
    CONTROL         "Act IV",IDC_TAB4_Act4,"Button",BS_AUTORADIOBUTTON |
                    WS_TABSTOP,78,51,35,10
    COMBOBOX        IDC_TAB4_SelAll,190,23,65,60,CBS_DROPDOWNLIST |
                    WS_VSCROLL | WS_TABSTOP
    PUSHBUTTON      "in this Act",IDC_TAB4_SetAll1,260,15,40,13
    GROUPBOX        "Quests",IDC_STATIC,4,65,301,155,BS_CENTER | WS_GROUP,
                    WS_EX_TRANSPARENT
    CONTROL         "",IDC_TAB4_Bmp1,"Static",SS_BITMAP | SS_NOTIFY,40,89,31,
                    31
    CTEXT           "Quest",IDC_TAB4_Text1,10,77,90,8
    COMBOBOX        IDC_TAB4_Sel1,15,125,80,65,CBS_DROPDOWNLIST |
                    CBS_AUTOHSCROLL | CBS_DISABLENOSCROLL | WS_TABSTOP
    CONTROL         "",IDC_STATIC,"Static",SS_ETCHEDHORZ,10,144,290,1
    CONTROL         "",IDC_STATIC,"Static",SS_ETCHEDVERT,104,75,1,65
    CONTROL         "",IDC_STATIC,"Static",SS_ETCHEDVERT,205,75,1,64
    CONTROL         "",IDC_STATIC,"Static",SS_ETCHEDVERT,104,150,1,64
    CONTROL         "",IDC_STATIC,"Static",SS_ETCHEDVERT,205,150,1,63
    COMBOBOX        IDC_TAB4_Sel2,115,125,80,65,CBS_DROPDOWNLIST |
                    WS_VSCROLL | WS_TABSTOP
    COMBOBOX        IDC_TAB4_Sel3,215,125,80,65,CBS_DROPDOWNLIST |
                    WS_VSCROLL | WS_TABSTOP
    CONTROL         "",IDC_TAB4_Bmp2,"Static",SS_BITMAP | SS_NOTIFY,140,89,
                    31,31
    CTEXT           "Quest",IDC_TAB4_Text2,110,77,90,8
    CONTROL         "",IDC_TAB4_Bmp3,"Static",SS_BITMAP | SS_NOTIFY,239,89,
                    31,31
    CTEXT           "Quest",IDC_TAB4_Text3,210,77,90,8
    CONTROL         "",IDC_TAB4_Bmp4,"Static",SS_BITMAP | SS_NOTIFY,40,162,
                    15,13
    CTEXT           "Quest",IDC_TAB4_Text4,10,150,90,8
    COMBOBOX        IDC_TAB4_Sel4,15,200,80,65,CBS_DROPDOWNLIST | WS_VSCROLL |
                    WS_TABSTOP
    COMBOBOX        IDC_TAB4_Sel5,115,200,80,65,CBS_DROPDOWNLIST |
                    WS_VSCROLL | WS_TABSTOP
    COMBOBOX        IDC_TAB4_Sel6,215,200,80,65,CBS_DROPDOWNLIST |
                    WS_VSCROLL | WS_TABSTOP
```

```
    CONTROL         "",IDC_TAB4_Bmp5,"Static",SS_BITMAP | SS_NOTIFY,139,162,
                    15,13
    CTEXT           "Quest",IDC_TAB4_Text5,110,150,90,8
    CONTROL         "",IDC_TAB4_Bmp6,"Static",SS_BITMAP | SS_NOTIFY,239,162,
                    31,31
    CTEXT           "Quest",IDC_TAB4_Text6,210,150,90,8
    PUSHBUTTON      "in all Acts",IDC_TAB4_SetAll2,260,32,39,13
    CONTROL         "Enabled",IDC_TAB4_Act2_On,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,130,27,45,10
    CONTROL         "Enabled",IDC_TAB4_Act3_On,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,130,39,45,10
    CONTROL         "Finished",IDC_TAB4_Act4_On,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,130,51,45,10
    PUSHBUTTON      "Info about Secret Cow Level",IDC_TAB4_CowLevel,184,53,
                    121,13
END

IDD_TAB5 DIALOGEX 0, 0, 310, 226
STYLE WS_CHILD
FONT 8, "MS Sans Serif", 0, 0, 0x1
BEGIN
    CONTROL         "Normal",IDC_TAB5_Diff1,"Button",BS_AUTORADIOBUTTON |
                    WS_GROUP | WS_TABSTOP,10,16,38,10
    CONTROL         "Nightmare",IDC_TAB5_Diff2,"Button",BS_AUTORADIOBUTTON |
                    WS_TABSTOP,67,16,48,10
    CONTROL         "Hell",IDC_TAB5_Diff3,"Button",BS_AUTORADIOBUTTON |
                    WS_TABSTOP,134,16,28,10
    PUSHBUTTON      "Batch Action",IDC_TAB5_Batch,190,10,111,20,WS_GROUP
    GROUPBOX        "Act I",IDC_STATIC,4,36,71,184,BS_CENTER,
                    WS_EX_TRANSPARENT
    CONTROL         "",IDC_TAB5_Way01,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_GROUP | WS_TABSTOP,8,51,13,12
    LTEXT           "Rogue Encampment",IDC_STATIC,25,49,45,17,NOT WS_GROUP
    CONTROL         "",IDC_TAB5_Way02,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,8,69,13,12
    LTEXT           "Cold Plains",IDC_STATIC,25,71,45,8,NOT WS_GROUP
    CONTROL         "",IDC_TAB5_Way03,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,8,87,13,12
    LTEXT           "Stony Field",IDC_STATIC,25,89,45,8,NOT WS_GROUP
    CONTROL         "",IDC_TAB5_Way04,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,8,105,13,12
    LTEXT           "Dark Wood",IDC_STATIC,25,107,45,8,NOT WS_GROUP
    CONTROL         "",IDC_TAB5_Way05,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,8,123,13,12
    LTEXT           "Black Marsh",IDC_STATIC,25,125,45,8,NOT WS_GROUP
    CONTROL         "",IDC_TAB5_Way06,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,8,141,13,12
    LTEXT           "Outer Cloister",IDC_STATIC,25,143,45,8,NOT WS_GROUP
    CONTROL         "",IDC_TAB5_Way07,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,8,159,13,12
    LTEXT           "Jail Level 1",IDC_STATIC,25,161,45,8,NOT WS_GROUP
    CONTROL         "",IDC_TAB5_Way08,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,8,177,13,12
    LTEXT           "Inner Cloister",IDC_STATIC,25,179,45,8,NOT WS_GROUP
    CONTROL         "",IDC_TAB5_Way09,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,8,195,13,12
    LTEXT           "Catacombs Level 2",IDC_STATIC,25,194,45,16,NOT WS_GROUP
    GROUPBOX        "Act II",IDC_STATIC,79,36,72,184,BS_CENTER,
                    WS_EX_TRANSPARENT
    CONTROL         "",IDC_TAB5_Way11,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,82,51,13,12
    LTEXT           "Lut Gholein",IDC_STATIC,100,53,45,8,NOT WS_GROUP
    CONTROL         "",IDC_TAB5_Way12,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,82,68,13,12
    LTEXT           "Sewers   Level 2",IDC_STATIC,100,67,45,16,NOT WS_GROUP
    CONTROL         "",IDC_TAB5_Way13,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,82,87,13,12
    LTEXT           "Dry Hills",IDC_STATIC,100,89,45,8,NOT WS_GROUP
    CONTROL         "",IDC_TAB5_Way14,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,82,105,13,12
    LTEXT           "Halls of the Dead Level 2",IDC_STATIC,100,104,45,16,NOT
                    WS_GROUP
    CONTROL         "",IDC_TAB5_Way15,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,82,123,13,12
    LTEXT           "Far Oasis",IDC_STATIC,100,125,45,8,NOT WS_GROUP
    CONTROL         "",IDC_TAB5_Way16,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,82,140,13,12
    LTEXT           "Lost City",IDC_STATIC,100,142,45,8,NOT WS_GROUP
    CONTROL         "",IDC_TAB5_Way17,"Static",SS_BITMAP | SS_NOTIFY |
```

```
                    WS_TABSTOP,82,159,13,12
    LTEXT           "Palace Celler Level 1",IDC_STATIC,100,158,45,16,NOT
                    WS_GROUP
    CONTROL         "",IDC_TAB5_Way18,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,82,177,13,12
    LTEXT           "Arcane Sanctuary",IDC_STATIC,100,176,45,16,NOT WS_GROUP
    CONTROL         "",IDC_TAB5_Way19,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,82,195,13,12
    LTEXT           "Canyon of  the Magi",IDC_STATIC,100,194,45,16,NOT
                    WS_GROUP
    GROUPBOX        "Act III",IDC_STATIC,154,36,71,184,BS_CENTER,
                    WS_EX_TRANSPARENT
    CONTROL         "",IDC_TAB5_Way21,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,157,51,13,12
    LTEXT           "Kurast Docks",IDC_STATIC,175,54,45,8,NOT WS_GROUP
    CONTROL         "",IDC_TAB5_Way22,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,157,69,13,12
    LTEXT           "Spider Forest",IDC_STATIC,174,71,45,8,NOT WS_GROUP
    CONTROL         "",IDC_TAB5_Way23,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,157,87,13,12
    LTEXT           "Great Marsh",IDC_STATIC,174,89,45,8,NOT WS_GROUP
    CONTROL         "",IDC_TAB5_Way24,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,157,105,13,12
    LTEXT           "Flayer Jungle",IDC_STATIC,174,107,45,8,NOT WS_GROUP
    CONTROL         "",IDC_TAB5_Way25,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,157,123,13,12
    LTEXT           "Lower Kurast",IDC_STATIC,174,125,45,8,NOT WS_GROUP
    CONTROL         "",IDC_TAB5_Way26,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,157,141,13,12
    LTEXT           "Kurast Bazaar",IDC_STATIC,174,143,45,8,NOT WS_GROUP
    CONTROL         "",IDC_TAB5_Way27,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,157,159,13,12
    LTEXT           "Upper Kurast",IDC_STATIC,174,161,45,8,NOT WS_GROUP
    CONTROL         "",IDC_TAB5_Way28,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,157,177,13,12
    LTEXT           "Travincal",IDC_STATIC,174,179,45,8,NOT WS_GROUP
    CONTROL         "",IDC_TAB5_Way29,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,157,195,13,12
    LTEXT           "Durance of Hate Level 2",IDC_STATIC,174,194,45,15,NOT
                    WS_GROUP
    GROUPBOX        "Act IV",IDC_STATIC,230,36,75,184,BS_CENTER,
                    WS_EX_TRANSPARENT
    CONTROL         "",IDC_TAB5_Way31,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,232,51,13,12
    LTEXT           "Pandemonium Fortress",IDC_STATIC,249,49,51,17,NOT
                    WS_GROUP
    CONTROL         "",IDC_TAB5_Way32,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,232,69,13,12
    LTEXT           "City of the Damned",IDC_STATIC,250,68,45,16,NOT
                    WS_GROUP
    CONTROL         "",IDC_TAB5_Way33,"Static",SS_BITMAP | SS_NOTIFY |
                    WS_TABSTOP,232,87,13,12
    LTEXT           "River of Flame",IDC_STATIC,249,89,45,8,NOT WS_GROUP
    GROUPBOX        "Difficulty",IDC_STATIC,4,6,177,24,BS_CENTER,
                    WS_EX_TRANSPARENT
END

IDD_COWLEVEL DIALOG DISCARDABLE  0, 0, 442, 266
STYLE DS_MODALFRAME | DS_CENTER | WS_POPUP | WS_VISIBLE | WS_CAPTION |
    WS_SYSMENU
CAPTION "The Secret Cow Level"
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON   "OK",IDOK,189,245,65,14
    CONTROL         247,IDC_STATIC,"Static",SS_BITMAP,150,5,143,202
    CONTROL         248,IDC_STATIC,"Static",SS_BITMAP,304,5,133,110
    CTEXT           "You need to get Wirt's Leg from his corpse located in the northwest area of Tristam.",
                    IDC_COWLEVEL_TextA,5,120,134,17
    CTEXT           "Back in the Rogues' Camp place Wirt's Leg together with a Tomb of Town Portal from Akara into the H
oradric Cube and trasmute.",
                    IDC_COWLEVEL_TextB,150,213,142,27
    CONTROL         250,IDC_STATIC,"Static",SS_BITMAP,5,5,133,110
    CONTROL         "",IDC_STATIC,"Static",SS_ETCHEDVERT,144,5,1,230
    CONTROL         "",IDC_STATIC,"Static",SS_ETCHEDVERT,298,5,1,230
    CTEXT           "A secret Red Portal will appear at your location. Enter it and be prepared to face the most horribl
e evil man can think of: COWS.",
                    IDC_COWLEVEL_TextB2,304,120,134,33
    CTEXT           "If you have previously killed the Cow King in the same difficulty level, you cannot reenter the Cow
 Level.\nThis flag is stored in the Quest ""Search for Cain"". Reset this quest and you can kill the Cow King again.",
```

```
                    IDC_STATIC,305,187,135,48
END

IDD_PROGRESS DIALOG DISCARDABLE  0, 0, 310, 225
STYLE WS_CHILD
FONT 8, "MS Sans Serif"
BEGIN
    CONTROL         "Progress",IDC_PROGRESS_Bar,"msctls_progress32",
                    PBS_SMOOTH,17,121,275,12
    CTEXT           "Static",IDC_PROGRESS_Text,67,91,170,15
END

IDD_INFO DIALOGEX 0, 0, 257, 143
STYLE DS_MODALFRAME | DS_CENTER | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Jamella's Diablo 2 Hero Editor"
FONT 8, "MS Sans Serif"
BEGIN
    CONTROL         286,IDC_STATIC,"Static",SS_BITMAP,5,5,77,107
    CTEXT           "Program\nVersion",IDC_INFO_Program,90,5,160,18,0,
                    WS_EX_STATICEDGE
    CTEXT           "Compile Date\nVersion Code",IDC_INFO_Date,90,28,160,18,
                    0,WS_EX_STATICEDGE
    LTEXT           "URL",IDC_STATIC,90,53,16,8
    EDITTEXT        IDC_INFO_URL,90,61,143,12,ES_CENTER | ES_AUTOHSCROLL |
                    ES_READONLY | NOT WS_BORDER
    PUSHBUTTON      "Button1",IDC_INFO_LINK,234,59,16,15,BS_BITMAP
    LTEXT           "Email",IDC_STATIC,90,74,18,8
    EDITTEXT        IDC_INFO_Email,90,81,160,12,ES_CENTER | ES_AUTOHSCROLL |
                    ES_READONLY | NOT WS_BORDER
    LTEXT           "When sending bug reports, please specify the version code above.",
                    IDC_STATIC,90,100,160,16
    DEFPUSHBUTTON   "OK",IDOK,142,125,50,14
END

IDD_TAB2E DIALOGEX 0, 0, 160, 270
STYLE DS_CENTER | WS_POPUP | WS_CAPTION | WS_SYSMENU
EXSTYLE WS_EX_TOOLWINDOW
CAPTION "Expert Item Codes"
FONT 8, "MS Sans Serif"
BEGIN
    PUSHBUTTON      "Find Info",IDC_TAB2E_FindInfo,115,227,40,10
    EDITTEXT        IDC_TAB2E_Raw00,28,41,15,12,ES_CENTER | ES_AUTOHSCROLL
    CTEXT           "Raw Data",IDC_STATIC,5,28,150,8
    EDITTEXT        IDC_TAB2E_GemNum,95,212,59,12,ES_AUTOHSCROLL
    EDITTEXT        IDC_TAB2E_ItemCode,95,108,60,12,ES_AUTOHSCROLL
    EDITTEXT        IDC_TAB2E_Xoord,95,147,30,12,ES_AUTOHSCROLL |
                    ES_READONLY
    EDITTEXT        IDC_TAB2E_Yoord,125,147,29,12,ES_AUTOHSCROLL |
                    ES_READONLY
    EDITTEXT        IDC_TAB2E_DWA,95,173,59,12,ES_AUTOHSCROLL
    EDITTEXT        IDC_TAB2E_DWB,95,186,60,13,ES_AUTOHSCROLL
    RADIOBUTTON     "DWORD A",IDC_TAB2E_RandA,98,246,51,10,BS_NOTIFY |
                    WS_GROUP | WS_TABSTOP
    RADIOBUTTON     "DWORD B",IDC_TAB2E_RandB,98,257,51,10,BS_NOTIFY |
                    WS_GROUP | WS_TABSTOP
    CONTROL         "",IDC_STATIC,"Static",SS_ETCHEDHORZ,2,100,157,1
    LTEXT           "Socketed Gems",IDC_STATIC,5,214,60,8
    LTEXT           "Item Code",IDC_STATIC,5,110,60,8
    LTEXT           "X && Y Coordinates",IDC_STATIC,5,149,60,8
    LTEXT           "Set && Unique Class",IDC_STATIC,5,123,60,8
    LTEXT           "DWA",IDC_STATIC,5,175,60,8
    LTEXT           "DWB",IDC_STATIC,5,188,60,8
    LTEXT           "Display output of Diablo 2 Random Generator for",
                    IDC_STATIC,5,248,85,16
    CONTROL         "",IDC_STATIC,"Static",SS_ETCHEDHORZ,3,240,155,1
    CONTROL         "",IDC_STATIC,"Static",SS_ETCHEDHORZ,2,22,157,1
    LTEXT           "Item Record ID",IDC_STATIC,5,7,60,8
    EDITTEXT        IDC_TAB2E_Raw01,44,41,15,12,ES_CENTER | ES_AUTOHSCROLL
    EDITTEXT        IDC_TAB2E_Raw02,60,41,15,12,ES_CENTER | ES_AUTOHSCROLL
    EDITTEXT        IDC_TAB2E_Raw03,76,41,15,12,ES_CENTER | ES_AUTOHSCROLL
    EDITTEXT        IDC_TAB2E_Raw04,92,41,15,12,ES_CENTER | ES_AUTOHSCROLL
    EDITTEXT        IDC_TAB2E_Raw05,108,41,15,12,ES_CENTER | ES_AUTOHSCROLL
    EDITTEXT        IDC_TAB2E_Raw06,123,41,15,12,ES_CENTER | ES_AUTOHSCROLL
    EDITTEXT        IDC_TAB2E_Raw07,139,41,15,12,ES_CENTER | ES_AUTOHSCROLL
    EDITTEXT        IDC_TAB2E_Raw08,28,54,15,12,ES_CENTER | ES_AUTOHSCROLL
    EDITTEXT        IDC_TAB2E_Raw09,44,54,15,12,ES_CENTER | ES_AUTOHSCROLL
    EDITTEXT        IDC_TAB2E_Raw0A,60,54,15,12,ES_CENTER | ES_AUTOHSCROLL
    EDITTEXT        IDC_TAB2E_Raw0B,76,54,15,12,ES_CENTER | ES_AUTOHSCROLL
    EDITTEXT        IDC_TAB2E_Raw0C,92,54,15,12,ES_CENTER | ES_AUTOHSCROLL
```

```
    EDITTEXT        IDC_TAB2E_Raw0D,108,54,15,12,ES_CENTER | ES_AUTOHSCROLL
    EDITTEXT        IDC_TAB2E_Raw0E,123,54,15,12,ES_CENTER | ES_AUTOHSCROLL
    EDITTEXT        IDC_TAB2E_Raw0F,139,54,15,12,ES_CENTER | ES_AUTOHSCROLL
    EDITTEXT        IDC_TAB2E_Raw10,28,67,15,12,ES_CENTER | ES_AUTOHSCROLL
    EDITTEXT        IDC_TAB2E_Raw11,44,67,15,12,ES_CENTER | ES_AUTOHSCROLL
    EDITTEXT        IDC_TAB2E_Raw12,60,67,15,12,ES_CENTER | ES_AUTOHSCROLL
    EDITTEXT        IDC_TAB2E_Raw13,76,67,15,12,ES_CENTER | ES_AUTOHSCROLL
    EDITTEXT        IDC_TAB2E_Raw14,92,67,15,12,ES_CENTER | ES_AUTOHSCROLL
    EDITTEXT        IDC_TAB2E_Raw15,108,67,15,12,ES_CENTER | ES_AUTOHSCROLL
    EDITTEXT        IDC_TAB2E_Raw16,123,67,15,12,ES_CENTER | ES_AUTOHSCROLL
    EDITTEXT        IDC_TAB2E_Raw17,139,67,15,12,ES_CENTER | ES_AUTOHSCROLL
    EDITTEXT        IDC_TAB2E_Raw18,28,80,15,12,ES_CENTER | ES_AUTOHSCROLL
    EDITTEXT        IDC_TAB2E_Raw19,44,80,15,12,ES_CENTER | ES_AUTOHSCROLL
    EDITTEXT        IDC_TAB2E_Raw1A,60,80,15,12,ES_CENTER | ES_AUTOHSCROLL
    EDITTEXT        IDC_TAB2E_Raw1B,76,80,15,12,ES_CENTER | ES_AUTOHSCROLL
    EDITTEXT        IDC_TAB2E_Raw1C,92,80,15,12,ES_CENTER | ES_AUTOHSCROLL
    EDITTEXT        IDC_TAB2E_Raw1D,108,80,15,12,ES_CENTER | ES_AUTOHSCROLL
    EDITTEXT        IDC_TAB2E_Raw1E,123,80,15,12,ES_CENTER | ES_AUTOHSCROLL
    EDITTEXT        IDC_TAB2E_Raw1F,139,80,15,12,ES_CENTER | ES_AUTOHSCROLL
    CTEXT           "0x00",IDC_STATIC,5,43,20,8
    CTEXT           "0x08",IDC_STATIC,5,56,20,8
    CTEXT           "0x10",IDC_STATIC,5,69,20,8
    CTEXT           "0x18",IDC_STATIC,5,82,20,8
    EDITTEXT        IDC_TAB2E_UniqueCode,95,121,60,12,ES_AUTOHSCROLL
    EDITTEXT        IDC_TAB2E_BodyCode,95,160,59,12,ES_AUTOHSCROLL |
                    ES_READONLY
    LTEXT           "Body Code",IDC_STATIC,5,162,60,8
    EDITTEXT        IDC_TAB2E_MagicLevel,95,199,60,13,ES_AUTOHSCROLL
    LTEXT           "Magic Level",IDC_STATIC,5,201,60,8
    CTEXT           "Static",IDC_TAB2E_ItemRecordID,75,6,79,9,SS_SUNKEN
    EDITTEXT        IDC_TAB2E_ItemCodeChar,70,108,25,12,ES_AUTOHSCROLL |
                    ES_READONLY
    PUSHBUTTON      "Decode",IDC_TAB2E_Decode,75,227,40,10
    EDITTEXT        IDC_TAB2E_Container,95,134,59,12,ES_AUTOHSCROLL |
                    ES_READONLY
    LTEXT           "Container",IDC_STATIC,5,136,60,8
    CHECKBOX        "Item List",IDC_TAB2E_ItemList,35,227,40,10,BS_PUSHLIKE
END

IDD_TAB2Rnd DIALOGEX 0, 0, 114, 278
STYLE WS_POPUP | WS_CAPTION | WS_SYSMENU
EXSTYLE WS_EX_TOOLWINDOW
CAPTION "Random Generator Output"
FONT 8, "Courier New", 0, 0, 0x1
BEGIN
    EDITTEXT        IDC_TAB2Rnd_Edit,5,5,104,268,ES_MULTILINE |
                    ES_AUTOVSCROLL | ES_READONLY | NOT WS_BORDER |
                    WS_VSCROLL,WS_EX_STATICEDGE
END

IDD_TAB2Magic DIALOGEX 0, 0, 335, 350
STYLE DS_MODALFRAME | DS_SETFOREGROUND | DS_CENTER | WS_POPUP | WS_CAPTION |
    WS_SYSMENU
CAPTION "Magical Attributes"
FONT 8, "MS Sans Serif", 0, 0, 0x1
BEGIN
    CONTROL         "Force Magical Prefix Match",IDC_TAB2Magic_PrefixMatch,
                    "Button",BS_AUTOCHECKBOX | BS_NOTIFY | WS_TABSTOP,34,16,
                    102,10
    CONTROL         "Tree1",IDC_TAB2Magic_PrefixTree,"SysTreeView32",
                    TVS_HASBUTTONS | TVS_HASLINES | TVS_LINESATROOT |
                    TVS_DISABLEDRAGDROP | TVS_SHOWSELALWAYS | WS_TABSTOP,10,
                    30,150,175,WS_EX_STATICEDGE
    CONTROL         "Slider2",IDC_TAB2Magic_Prefix1Value,"msctls_trackbar32",
                    TBS_BOTH | TBS_NOTICKS | WS_TABSTOP,25,224,120,15
    CONTROL         "Slider2",IDC_TAB2Magic_Prefix2Value,"msctls_trackbar32",
                    TBS_BOTH | TBS_NOTICKS | WS_TABSTOP,25,257,120,15
    CONTROL         "Slider2",IDC_TAB2Magic_Prefix3Value,"msctls_trackbar32",
                    TBS_BOTH | TBS_NOTICKS | WS_TABSTOP,25,290,120,15
    CONTROL         "Slider2",IDC_TAB2Magic_Prefix4Value,"msctls_trackbar32",
                    TBS_BOTH | TBS_NOTICKS | WS_TABSTOP,25,325,120,15
    LTEXT           "Average Tries: 8787878",IDC_TAB2Magic_Average,170,307,
                    105,8
    PUSHBUTTON      "Search",IDOK,170,329,50,14
    CONTROL         "Force Magical Suffix Match",IDC_TAB2Magic_SuffixMatch,
                    "Button",BS_AUTOCHECKBOX | BS_NOTIFY | WS_TABSTOP,199,16,
                    102,10
    CONTROL         "Tree1",IDC_TAB2Magic_SuffixTree,"SysTreeView32",
                    TVS_HASBUTTONS | TVS_HASLINES | TVS_LINESATROOT |
```

```
                    TVS_DISABLEDRAGDROP | TVS_SHOWSELALWAYS | WS_TABSTOP,175,
                    30,150,175,WS_EX_STATICEDGE
    CONTROL         "Slider2",IDC_TAB2Magic_Suffix1Value,"msctls_trackbar32",
                    TBS_BOTH | TBS_NOTICKS | WS_TABSTOP,191,225,120,15
    DEFPUSHBUTTON   "Cancel",IDCANCEL,280,329,50,14
    GROUPBOX        "Prefix",IDC_STATIC,5,5,160,339,BS_CENTER
    CONTROL         "Force Value of 1st prefix effect to 444",
                    IDC_TAB2Magic_Prefix1Match,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,15,213,140,10
    CONTROL         "",IDC_STATIC,"Static",SS_ETCHEDHORZ,10,209,149,1
    LTEXT           "999",IDC_TAB2Magic_Prefix1ValueMin,10,228,13,8
    RTEXT           "999",IDC_TAB2Magic_Prefix1ValueMax,145,228,13,8
    GROUPBOX        "Suffix",IDC_STATIC,170,5,160,239,BS_CENTER
    CONTROL         "",IDC_STATIC,"Static",SS_ETCHEDHORZ,10,242,148,1
    CONTROL         "Force Value of 1st prefix effect to 444",
                    IDC_TAB2Magic_Prefix2Match,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,15,247,140,10
    RTEXT           "999",IDC_TAB2Magic_Prefix2ValueMax,145,260,13,8
    LTEXT           "999",IDC_TAB2Magic_Prefix2ValueMin,10,260,13,8
    CONTROL         "",IDC_STATIC,"Static",SS_ETCHEDHORZ,10,276,147,1
    CONTROL         "Force Value of 1st prefix effect to 444",
                    IDC_TAB2Magic_Prefix3Match,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,15,280,140,10
    RTEXT           "999",IDC_TAB2Magic_Prefix3ValueMax,145,293,13,8
    LTEXT           "999",IDC_TAB2Magic_Prefix3ValueMin,10,293,13,8
    CONTROL         "",IDC_STATIC,"Static",SS_ETCHEDHORZ,10,309,146,1
    CONTROL         "Force Value of 1st prefix effect to 444",
                    IDC_TAB2Magic_Prefix4Match,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,15,313,140,10
    RTEXT           "999",IDC_TAB2Magic_Prefix4ValueMax,145,328,13,8
    LTEXT           "999",IDC_TAB2Magic_Prefix4ValueMin,10,328,13,8
    CONTROL         "Force Value of 1st prefix effect to 444",
                    IDC_TAB2Magic_Suffix1Match,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,181,214,140,10
    RTEXT           "999",IDC_TAB2Magic_Suffix1ValueMax,311,228,13,8
    LTEXT           "999",IDC_TAB2Magic_Suffix1ValueMin,175,228,13,8
    CONTROL         "",IDC_STATIC,"Static",SS_ETCHEDHORZ,175,209,148,1
    GROUPBOX        "Required Experience Level",IDC_STATIC,170,247,160,45
    LTEXT           "Current Selection requires:",IDC_STATIC,175,258,84,8
    CTEXT           "Static",IDC_TAB2Magic_CurrentELevel,305,257,20,10,0,
                    WS_EX_STATICEDGE
    CHECKBOX        "Hide attributes that make the exprerience level exceed",
                    IDC_TAB2Magic_LockELevel,175,270,125,16,BS_MULTILINE
    EDITTEXT        IDC_TAB2Magic_LockValue,305,272,20,12,ES_CENTER |
                    ES_AUTOHSCROLL | ES_NUMBER
    CHECKBOX        "?",IDC_CHELP,260,329,16,14,BS_ICON | BS_CENTER |
                    BS_VCENTER | BS_PUSHLIKE
    PUSHBUTTON      "Clear",IDC_TAB2Magic_Clear,225,329,30,14
END

IDD_TAB2SS DIALOGEX 0, 0, 155, 70
STYLE DS_MODALFRAME | DS_CENTER | WS_POPUP | WS_CAPTION
CAPTION "Running Brute Force Attack..."
FONT 8, "MS Sans Serif", 0, 0, 0x1
BEGIN
    CTEXT           "0",IDC_TAB2SS_Counter,0,20,155,8
    CTEXT           "0",IDC_TAB2SS_Current,0,7,155,8
    DEFPUSHBUTTON   "Stop Search",IDC_TAB2SS_Stop,45,49,65,16
    CONTROL         "Progress1",IDC_TAB2SS_Scope,"msctls_progress32",
                    PBS_SMOOTH,5,34,145,10,WS_EX_STATICEDGE
END

IDD_TAB2Rare DIALOGEX 0, 0, 445, 309
STYLE DS_MODALFRAME | DS_CENTER | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Rare Attributes"
FONT 8, "MS Sans Serif", 0, 0, 0x1
BEGIN
    LISTBOX         IDC_TAB2Rare_NamePrefix,10,15,75,75,LBS_SORT |
                    LBS_NOINTEGRALHEIGHT | NOT WS_BORDER | WS_VSCROLL |
                    WS_TABSTOP,WS_EX_STATICEDGE
    LISTBOX         IDC_TAB2Rare_NameSuffix,85,15,75,75,LBS_SORT |
                    LBS_NOINTEGRALHEIGHT | NOT WS_BORDER | WS_VSCROLL |
                    WS_TABSTOP,WS_EX_STATICEDGE
    CONTROL         "Tree1",IDC_TAB2Rare_TreeSuffix,"SysTreeView32",
                    TVS_HASBUTTONS | TVS_HASLINES | TVS_LINESATROOT |
                    TVS_DISABLEDRAGDROP | TVS_SHOWSELALWAYS | WS_TABSTOP,305,
                    15,130,265,WS_EX_STATICEDGE
    PUSHBUTTON      "Search",IDOK,197,290,50,14
    DEFPUSHBUTTON   "Cancel",IDCANCEL,390,290,50,14
```

```
    GROUPBOX        "Magical Attributes",IDC_STATIC,5,100,160,137,BS_CENTER
    GROUPBOX        "Name Prefix && Suffix",IDC_STATIC,5,5,160,90,BS_CENTER
    GROUPBOX        "Selected Attribute",IDC_STATIC,170,5,270,280,BS_CENTER
    LTEXT           "Average Tries: 8787878",IDC_TAB2Rare_Average,6,293,127,
                    8
    RADIOBUTTON     "1st Attribute",IDC_TAB2Rare_Check1,15,115,140,8,
                    BS_NOTIFY
    RADIOBUTTON     "2nd Attribute",IDC_TAB2Rare_Check2,15,135,140,8,
                    BS_NOTIFY
    RADIOBUTTON     "3rd Attribute",IDC_TAB2Rare_Check3,15,155,140,8,
                    BS_NOTIFY
    RADIOBUTTON     "4th Attribute",IDC_TAB2Rare_Check4,15,175,140,8,
                    BS_NOTIFY
    RADIOBUTTON     "5th Attribute",IDC_TAB2Rare_Check5,15,195,140,8,
                    BS_NOTIFY
    RADIOBUTTON     "6th Attribute",IDC_TAB2Rare_Check6,15,215,140,8,
                    BS_NOTIFY
    RTEXT           "bla bla",IDC_TAB2Rare_Text1,15,123,139,8
    RTEXT           "bla bla",IDC_TAB2Rare_Text2,15,143,139,8
    RTEXT           "bla bla",IDC_TAB2Rare_Text3,15,163,139,8
    RTEXT           "bla bla",IDC_TAB2Rare_Text4,15,183,139,8
    RTEXT           "bla bla",IDC_TAB2Rare_Text5,15,203,139,8
    RTEXT           "bla bla",IDC_TAB2Rare_Text6,15,223,139,8
    GROUPBOX        "Required Experience Level",IDC_STATIC,5,240,160,45
    LTEXT           "Current Selection requires:",IDC_STATIC,11,251,84,8
    CTEXT           "Static",IDC_TAB2Rare_CurrentELevel,141,250,20,10,0,
                    WS_EX_STATICEDGE
    CHECKBOX        "Hide attributes that make the exprerience level exceed",
                    IDC_TAB2Rare_LockELevel,11,263,125,16,BS_MULTILINE
    EDITTEXT        IDC_TAB2Rare_LockValue,141,265,20,12,ES_CENTER |
                    ES_AUTOHSCROLL | ES_NUMBER
    CONTROL         "Tree1",IDC_TAB2Rare_TreePrefix,"SysTreeView32",
                    TVS_HASBUTTONS | TVS_HASLINES | TVS_LINESATROOT |
                    TVS_DISABLEDRAGDROP | TVS_SHOWSELALWAYS | WS_TABSTOP,175,
                    15,130,265,WS_EX_STATICEDGE
    CHECKBOX        "?",IDC_CHELP,370,290,16,14,BS_ICON | BS_CENTER |
                    BS_VCENTER | BS_PUSHLIKE
    PUSHBUTTON      "Clear",IDC_TAB2Rare_Clear,135,290,30,14
END

IDD_SAVE DIALOG DISCARDABLE  0, 0, 207, 124
STYLE DS_MODALFRAME | DS_CENTER | WS_POPUP | WS_CAPTION
CAPTION "Save File?"
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON   "Save",IDOK,50,103,50,15
    PUSHBUTTON      "Cancel",IDCANCEL,105,103,50,15
    CTEXT           "Are you absolutly sure you want to save your changes?",
                    IDC_STATIC,11,10,185,8
    CONTROL         "Make Backup Copy",IDC_SAVE_Backup,"Button",
                    BS_AUTOCHECKBOX | WS_TABSTOP,64,26,79,10
    LTEXT           "WARNING: Only check the box if you are certain that your character file is read correctly by the ga
me!\nIt is a good choice to check this box the first time you save and then never again.",
                    IDC_STATIC,10,44,185,32
    LTEXT           "Backup files have the extention *.jam and are located in the same directory as the original charact
er files.",
                    IDC_STATIC,10,80,185,16
END

IDD_TAB2Quantity DIALOG DISCARDABLE  0, 0, 117, 95
STYLE DS_MODALFRAME | DS_CENTER | WS_POPUP | WS_CAPTION
CAPTION "Set Quantity"
FONT 8, "MS Sans Serif"
BEGIN
    LTEXT           "Set Item Quantity to",IDC_STATIC,27,7,63,8
    EDITTEXT        IDC_TAB2Quantity_Set,18,21,34,12,ES_CENTER |
                    ES_AUTOHSCROLL | ES_NUMBER
    LTEXT           "/  4545  Max",IDC_TAB2Quantity_Max,57,23,42,8
    CTEXT           "Diablo II's Maximum can be exceeded. Usual Max  is 255. You can extend the maximum to 65535 in the
options.",
                    IDC_STATIC,5,38,105,32
    DEFPUSHBUTTON   "OK",IDOK,21,75,35,14
    PUSHBUTTON      "Cancel",IDCANCEL,61,75,35,14
END

IDD_TAB2Durability DIALOG DISCARDABLE  0, 0, 110, 79
STYLE DS_MODALFRAME | DS_CENTER | WS_POPUP | WS_CAPTION
CAPTION "Set Durability"
FONT 8, "MS Sans Serif"
```

```
BEGIN
    LTEXT           "Set Item Durability to",IDC_STATIC,21,7,66,8
    EDITTEXT        IDC_TAB2Durability_Set,21,21,25,12,ES_CENTER |
                    ES_AUTOHSCROLL | ES_NUMBER
    EDITTEXT        IDC_TAB2Durability_Max,62,21,25,12,ES_CENTER |
                    ES_AUTOHSCROLL | ES_NUMBER
    CTEXT           "/",IDC_STATIC,50,23,8,8
    CTEXT           "Maximum is 255.\nDurability must be >= Maximum.",
                    IDC_STATIC,5,38,100,17
    DEFPUSHBUTTON   "OK",IDOK,17,59,35,14
    PUSHBUTTON      "Cancel",IDCANCEL,57,59,35,14
END

IDD_TAB2Defense DIALOG DISCARDABLE  0, 0, 110, 74
STYLE DS_MODALFRAME | DS_CENTER | WS_POPUP | WS_CAPTION
CAPTION "Select Defense"
FONT 8, "MS Sans Serif"
BEGIN
    CTEXT           "Search for Item Base Defense",IDC_STATIC,5,7,98,8
    CTEXT           "Range 24 to 35",IDC_TAB2Defense_Range,5,20,98,8
    EDITTEXT        IDC_TAB2Defense_Value,42,35,25,12,ES_CENTER |
                    ES_AUTOHSCROLL | ES_NUMBER
    DEFPUSHBUTTON   "OK",IDOK,17,55,35,14
    PUSHBUTTON      "Cancel",IDCANCEL,57,55,35,14
END

IDD_TAB2Gems1 DIALOG DISCARDABLE  0, 0, 295, 74
STYLE DS_MODALFRAME | DS_SETFOREGROUND | DS_CENTER | WS_POPUP | WS_CAPTION |
    WS_SYSMENU
CAPTION "Edit Inserted Gems"
FONT 8, "MS Sans Serif"
BEGIN
    CONTROL         25547,IDC_TAB2Gems_Bmp1,"Static",SS_BITMAP,15,20,19,17
    COMBOBOX        IDC_TAB2Gems_Sel1,45,22,100,175,CBS_DROPDOWNLIST |
                    WS_VSCROLL | WS_TABSTOP
    CTEXT           "Info Gem 1",IDC_TAB2Gems_Info1,155,15,125,30
    DEFPUSHBUTTON   "OK",IDOK,90,55,50,14
    PUSHBUTTON      "Cancel",IDCANCEL,145,55,50,14
    GROUPBOX        "First Gem",IDC_TAB2Gems_Frame1,5,5,285,45
END

IDD_TAB2Gems2 DIALOG DISCARDABLE  0, 0, 295, 124
STYLE DS_MODALFRAME | DS_SETFOREGROUND | DS_CENTER | WS_POPUP | WS_CAPTION |
    WS_SYSMENU
CAPTION "Edit Inserted Gems"
FONT 8, "MS Sans Serif"
BEGIN
    GROUPBOX        "First Gem",IDC_TAB2Gems_Frame1,5,5,285,45
    CONTROL         25547,IDC_TAB2Gems_Bmp1,"Static",SS_BITMAP,15,20,19,17
    COMBOBOX        IDC_TAB2Gems_Sel1,45,22,100,175,CBS_DROPDOWNLIST |
                    WS_VSCROLL | WS_TABSTOP
    CTEXT           "Info Gem 1",IDC_TAB2Gems_Info1,155,15,125,30
    GROUPBOX        "Second Gem",IDC_TAB2Gems_Frame2,5,55,285,45
    CONTROL         25547,IDC_TAB2Gems_Bmp2,"Static",SS_BITMAP,15,70,19,17
    COMBOBOX        IDC_TAB2Gems_Sel2,45,72,100,175,CBS_DROPDOWNLIST |
                    WS_VSCROLL | WS_TABSTOP
    CTEXT           "Info Gem 2",IDC_TAB2Gems_Info2,155,65,125,30
    DEFPUSHBUTTON   "OK",IDOK,95,105,50,14
    PUSHBUTTON      "Cancel",IDCANCEL,150,105,50,14
END

IDD_TAB2Gems7 DIALOG DISCARDABLE  0, 0, 495, 223
STYLE DS_MODALFRAME | DS_SETFOREGROUND | DS_CENTER | WS_POPUP | WS_CAPTION |
    WS_SYSMENU
CAPTION "Edit Inserted Gems"
FONT 8, "MS Sans Serif"
BEGIN
    GROUPBOX        "First Gem",IDC_TAB2Gems_Frame1,5,5,240,45
    CONTROL         25547,IDC_TAB2Gems_Bmp1,"Static",SS_BITMAP,15,20,19,17
    COMBOBOX        IDC_TAB2Gems_Sel1,45,22,80,175,CBS_DROPDOWNLIST |
                    WS_VSCROLL | WS_TABSTOP
    CTEXT           "Info Gem 1",IDC_TAB2Gems_Info1,130,15,110,30
    DEFPUSHBUTTON   "OK",IDOK,195,205,50,14
    PUSHBUTTON      "Cancel",IDCANCEL,250,205,50,14
    GROUPBOX        "Second Gem",IDC_TAB2Gems_Frame2,5,55,240,45
    CONTROL         25547,IDC_TAB2Gems_Bmp2,"Static",SS_BITMAP,15,70,19,17
    COMBOBOX        IDC_TAB2Gems_Sel2,45,72,80,175,CBS_DROPDOWNLIST |
                    WS_VSCROLL | WS_TABSTOP
    CTEXT           "Info Gem 1",IDC_TAB2Gems_Info2,130,65,110,30
```

```
    GROUPBOX        "Third Gem",IDC_TAB2Gems_Frame3,5,105,240,45
    CONTROL         25547,IDC_TAB2Gems_Bmp3,"Static",SS_BITMAP,15,121,19,17
    COMBOBOX        IDC_TAB2Gems_Sel3,45,122,80,175,CBS_DROPDOWNLIST |
                    WS_VSCROLL | WS_TABSTOP
    CTEXT           "Info Gem 1",IDC_TAB2Gems_Info3,130,115,110,30
    GROUPBOX        "Fourth Gem",IDC_TAB2Gems_Frame4,250,5,240,45
    CONTROL         25547,IDC_TAB2Gems_Bmp4,"Static",SS_BITMAP,260,20,19,17
    COMBOBOX        IDC_TAB2Gems_Sel4,290,22,80,175,CBS_DROPDOWNLIST |
                    WS_VSCROLL | WS_TABSTOP
    CTEXT           "Info Gem 1",IDC_TAB2Gems_Info4,375,15,110,30
    GROUPBOX        "Fifth Gem",IDC_TAB2Gems_Frame5,250,55,240,45
    CONTROL         25547,IDC_TAB2Gems_Bmp5,"Static",SS_BITMAP,260,70,19,17
    COMBOBOX        IDC_TAB2Gems_Sel5,290,72,80,175,CBS_DROPDOWNLIST |
                    WS_VSCROLL | WS_TABSTOP
    CTEXT           "Info Gem 1",IDC_TAB2Gems_Info5,375,65,110,30
    GROUPBOX        "Sixth Gem",IDC_TAB2Gems_Frame6,250,105,240,45
    CONTROL         25547,IDC_TAB2Gems_Bmp6,"Static",SS_BITMAP,260,120,19,17
    COMBOBOX        IDC_TAB2Gems_Sel6,290,122,80,175,CBS_DROPDOWNLIST |
                    WS_VSCROLL | WS_TABSTOP
    CTEXT           "Info Gem 1",IDC_TAB2Gems_Info6,375,115,110,30
    GROUPBOX        "Seventh Gem",IDC_TAB2Gems_Frame7,128,155,240,45
    CONTROL         25547,IDC_TAB2Gems_Bmp7,"Static",SS_BITMAP,138,170,19,17
    COMBOBOX        IDC_TAB2Gems_Sel7,168,172,80,175,CBS_DROPDOWNLIST |
                    WS_VSCROLL | WS_TABSTOP
    CTEXT           "Info Gem 1",IDC_TAB2Gems_Info7,253,165,110,30
END

IDD_TAB2Ear DIALOG DISCARDABLE  0, 0, 161, 119
STYLE DS_MODALFRAME | DS_SETFOREGROUND | DS_CENTER | WS_POPUP | WS_CAPTION |
    WS_SYSMENU
CAPTION "Edit Ear Properties"
FONT 8, "MS Sans Serif"
BEGIN
    CONTROL         25503,IDC_TAB2Gems_Bmp1,"Static",SS_BITMAP,71,5,19,17
    GROUPBOX        "Killed Opponent's",IDC_STATIC,5,30,150,64
    LTEXT           "Name",IDC_STATIC,12,47,30,8
    EDITTEXT        IDC_TAB2Ear_Name,42,45,105,12,ES_AUTOHSCROLL
    LTEXT           "Class",IDC_STATIC,12,61,30,8
    COMBOBOX        IDC_TAB2Ear_Class,42,59,105,70,CBS_DROPDOWNLIST |
                    WS_VSCROLL | WS_TABSTOP
    LTEXT           "Level",IDC_STATIC,12,76,30,8
    EDITTEXT        IDC_TAB2Ear_Level,42,74,105,12,ES_AUTOHSCROLL |
                    ES_NUMBER
    DEFPUSHBUTTON   "OK",IDOK,28,100,50,14
    PUSHBUTTON      "Cancel",IDCANCEL,83,100,50,14
END

IDD_NEW DIALOG DISCARDABLE  0, 0, 162, 66
STYLE DS_MODALFRAME | DS_SETFOREGROUND | DS_CENTER | WS_POPUP | WS_CAPTION |
    WS_SYSMENU
CAPTION "Create New Character"
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON   "Create",IDC_NEW_CreateNewbie,56,45,50,14
    LTEXT           "Name",IDC_STATIC,10,29,30,8
    LTEXT           "Class",IDC_STATIC,10,12,30,8
    EDITTEXT        IDC_NEW_Name,46,27,105,12,ES_AUTOHSCROLL
    COMBOBOX        IDC_NEW_Class,46,10,105,70,CBS_DROPDOWNLIST | WS_VSCROLL |
                    WS_TABSTOP
END

IDD_RENAME DIALOG DISCARDABLE  0, 0, 162, 101
STYLE DS_MODALFRAME | DS_SETFOREGROUND | DS_CENTER | WS_POPUP | WS_CAPTION |
    WS_SYSMENU
CAPTION "Rename Character"
FONT 8, "MS Sans Serif"
BEGIN
    LTEXT           "New Name",-1,10,12,40,8
    EDITTEXT        IDC_RENAME_Name,55,10,95,12,ES_AUTOHSCROLL | WS_GROUP
    CONTROL         "Rename character and move all corresponding files.",
                    IDC_RENAME_Move,"Button",BS_AUTORADIOBUTTON |
                    BS_MULTILINE | WS_GROUP,20,30,120,16
    CONTROL         "Copy only character file, leave old character and map files untouched.",
                    IDC_RENAME_Copy,"Button",BS_AUTORADIOBUTTON |
                    BS_MULTILINE,20,50,120,23
    DEFPUSHBUTTON   "Rename",IDOK,27,80,50,14,WS_GROUP
    PUSHBUTTON      "Cancel",IDCANCEL,83,80,50,14,WS_GROUP
END
```

```
IDD_TAB2RingImage DIALOG DISCARDABLE  0, 0, 188, 66
STYLE DS_MODALFRAME | DS_SETFOREGROUND | DS_CENTER | WS_POPUP | WS_CAPTION |
    WS_SYSMENU
CAPTION "Select Ring Image"
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON   "OK",IDOK,41,45,50,14,WS_GROUP
    PUSHBUTTON      "Cancel",IDCANCEL,96,45,50,14,WS_GROUP
    CONTROL         "Image 1",IDC_TAB2RingImage_Image1,"Button",
                    BS_AUTORADIOBUTTON | BS_BITMAP | BS_PUSHLIKE | WS_GROUP,
                    10,10,30,25
    CONTROL         "Image 2",IDC_TAB2RingImage_Image2,"Button",
                    BS_AUTORADIOBUTTON | BS_BITMAP | BS_PUSHLIKE,45,10,30,25
    CONTROL         "Image 3",IDC_TAB2RingImage_Image3,"Button",
                    BS_AUTORADIOBUTTON | BS_BITMAP | BS_PUSHLIKE,80,10,30,25
    CONTROL         "Image 4",IDC_TAB2RingImage_Image4,"Button",
                    BS_AUTORADIOBUTTON | BS_BITMAP | BS_PUSHLIKE,115,10,30,
                    25
    CONTROL         "Image 5",IDC_TAB2RingImage_Image5,"Button",
                    BS_AUTORADIOBUTTON | BS_BITMAP | BS_PUSHLIKE,150,10,30,
                    25
END

IDD_TAB2AmuletImage DIALOG DISCARDABLE  0, 0, 130, 66
STYLE DS_MODALFRAME | DS_SETFOREGROUND | DS_CENTER | WS_POPUP | WS_CAPTION |
    WS_SYSMENU
CAPTION "Select Amulet Image"
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON   "OK",IDOK,12,45,50,14,WS_GROUP
    PUSHBUTTON      "Cancel",IDCANCEL,67,45,50,14,WS_GROUP
    CONTROL         "Image 1",IDC_TAB2AmuletImage_Image1,"Button",
                    BS_AUTORADIOBUTTON | BS_BITMAP | BS_PUSHLIKE | WS_GROUP,
                    10,10,30,25
    CONTROL         "Image 2",IDC_TAB2AmuletImage_Image2,"Button",
                    BS_AUTORADIOBUTTON | BS_BITMAP | BS_PUSHLIKE,50,10,30,25
    CONTROL         "Image 3",IDC_TAB2AmuletImage_Image3,"Button",
                    BS_AUTORADIOBUTTON | BS_BITMAP | BS_PUSHLIKE,90,10,30,25
END

IDD_IBWSR DIALOGEX 0, 0, 282, 130
STYLE DS_3DLOOK | DS_CONTROL | WS_CHILD | WS_CLIPSIBLINGS | WS_BORDER
FONT 8, "MS Sans Serif", 0, 0, 0x1
BEGIN
    CONTROL         "",IDC_IBWSR_RichText,"RICHEDIT",ES_CENTER |
                    ES_MULTILINE | ES_AUTOVSCROLL | ES_READONLY | ES_NUMBER |
                    WS_VSCROLL | WS_TABSTOP,70,5,207,120,WS_EX_STATICEDGE
    CONTROL         327,IDC_IBWSR_Bitmap,"Static",SS_BITMAP | SS_CENTERIMAGE |
                    SS_REALSIZEIMAGE,15,28,39,72
END

IDD_UOPTIONS DIALOG DISCARDABLE  0, 0, 197, 135
STYLE DS_MODALFRAME | DS_CENTER | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Options"
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON   "OK",IDOK,46,115,50,14
    PUSHBUTTON      "Cancel",IDCANCEL,101,115,50,14
    CONTROL         "Associate with *.d2s and *.d2i, *.item, *.itm, *.ite files for nice icons and open functions.",
                    IDC_UOPTIONS_Associations,"Button",BS_AUTOCHECKBOX |
                    BS_MULTILINE | WS_TABSTOP,10,10,175,15
    CONTROL         "Item Quantity may exceed 255\n(Extended Quantity Field)",
                    IDC_UOPTIONS_ExceedQuantity,"Button",BS_AUTOCHECKBOX |
                    BS_MULTILINE | WS_TABSTOP,10,30,175,16
    CONTROL         "Show tooltip help on the editor tabs. Disable this if it annoys you.",
                    IDC_UOPTIONS_Tooltips,"Button",BS_AUTOCHECKBOX |
                    BS_MULTILINE | WS_TABSTOP,10,55,175,16
    CONTROL         "Disable the annoying ""Are you sure?!?"" messages all around in the editor.",
                    IDC_UOPTIONS_AnnoyingMsgs,"Button",BS_AUTOCHECKBOX |
                    BS_MULTILINE | WS_TABSTOP,10,75,175,16
END

IDD_TAB2Gems3 DIALOG DISCARDABLE  0, 0, 295, 174
STYLE DS_MODALFRAME | DS_SETFOREGROUND | DS_CENTER | WS_POPUP | WS_CAPTION |
    WS_SYSMENU
CAPTION "Edit Inserted Gems"
FONT 8, "MS Sans Serif"
BEGIN
    GROUPBOX        "First Gem",IDC_TAB2Gems_Frame1,5,5,285,45
```

```
    CONTROL         25547,IDC_TAB2Gems_Bmp1,"Static",SS_BITMAP,15,20,19,17
    COMBOBOX        IDC_TAB2Gems_Sel1,45,22,100,175,CBS_DROPDOWNLIST |
                    WS_VSCROLL | WS_TABSTOP
    CTEXT           "Info Gem 1",IDC_TAB2Gems_Info1,155,15,125,30
    GROUPBOX        "Second Gem",IDC_TAB2Gems_Frame2,5,55,285,45
    CONTROL         25547,IDC_TAB2Gems_Bmp2,"Static",SS_BITMAP,15,70,19,17
    COMBOBOX        IDC_TAB2Gems_Sel2,45,72,100,175,CBS_DROPDOWNLIST |
                    WS_VSCROLL | WS_TABSTOP
    CTEXT           "Info Gem 2",IDC_TAB2Gems_Info2,155,65,125,30
    GROUPBOX        "Third Gem",IDC_TAB2Gems_Frame3,5,105,285,45
    CONTROL         25547,IDC_TAB2Gems_Bmp3,"Static",SS_BITMAP,15,121,19,17
    COMBOBOX        IDC_TAB2Gems_Sel3,45,123,100,175,CBS_DROPDOWNLIST |
                    WS_VSCROLL | WS_TABSTOP
    CTEXT           "Info Gem 3",IDC_TAB2Gems_Info3,155,115,125,30
    DEFPUSHBUTTON   "OK",IDOK,95,155,50,14
    PUSHBUTTON      "Cancel",IDCANCEL,150,155,50,14
END

IDD_HELP DIALOGEX 0, 0, 138, 274
STYLE WS_POPUP | WS_CAPTION | WS_SYSMENU | WS_THICKFRAME
EXSTYLE WS_EX_TOOLWINDOW
CAPTION "Instant Help Reference"
FONT 8, "Courier New", 0, 0, 0x1
BEGIN
    CONTROL         "",IDC_HELP_Text,"RICHEDIT",ES_CENTER | ES_MULTILINE |
                    ES_AUTOVSCROLL | ES_READONLY | ES_NUMBER | WS_VSCROLL |
                    WS_GROUP,0,0,138,274,WS_EX_STATICEDGE
END

IDD_EOPTIONS DIALOG DISCARDABLE  0, 0, 262, 191
STYLE DS_MODALFRAME | DS_CENTER | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Expert Options"
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON   "OK",IDOK,78,170,50,14
    PUSHBUTTON      "Cancel",IDCANCEL,133,170,50,14
    CONTROL         "Auto-detected",IDC_OPTIONS_IFormat1,"Button",
                    BS_AUTORADIOBUTTON,15,36,100,10
    GROUPBOX        "Created Item Record Format",IDC_STATIC,5,25,130,50,
                    BS_CENTER
    CONTROL         "for D2 <= 1.03",IDC_OPTIONS_IFormat2,"Button",
                    BS_AUTORADIOBUTTON,15,48,95,10
    CONTROL         "for D2 >= 1.04",IDC_OPTIONS_IFormat3,"Button",
                    BS_AUTORADIOBUTTON,15,59,95,10
    GROUPBOX        "Enhanced Item Hacking",IDC_STATIC,5,80,131,65,BS_CENTER
    CONTROL         "Make All Items Socketable",IDC_UOPTIONS_AllSocktable,
                    "Button",BS_AUTOCHECKBOX | WS_TABSTOP,15,95,100,10
    CONTROL         "Allow 7 Gems on socketed Item",IDC_EOPTIONS_7Gems,
                    "Button",BS_AUTOCHECKBOX | BS_MULTILINE | WS_TABSTOP,15,
                    110,115,8
    CTEXT           "! Use these options with caution !\nUnder usual circumstances none of these are activated!",
                    IDC_STATIC,5,5,250,16
END

IDD_TAB2S DIALOGEX 0, 0, 239, 173
STYLE DS_MODALFRAME | DS_CENTER | WS_MINIMIZEBOX | WS_POPUP | WS_CAPTION |
    WS_SYSMENU
CAPTION "Running Brute Force Attack..."
FONT 8, "MS Sans Serif"
BEGIN
    CTEXT           "0",IDC_TAB2SA_Counter,10,23,220,8
    CTEXT           "0",IDC_TAB2SA_Current,10,11,220,8
    DEFPUSHBUTTON   "Stop Search",IDC_TAB2SA_Stop,90,152,65,16
    LTEXT           "Thread Priority:",IDC_TAB2SA_Static4,10,94,48,8
    RTEXT           "Normal",IDC_TAB2SA_Priority,58,94,46,8
    CONTROL         "Slider1",IDC_TAB2SA_Slider,"msctls_trackbar32",
                    TBS_AUTOTICKS | TBS_TOP | TBS_NOTICKS | WS_TABSTOP,9,105,
                    96,14
    CONTROL         "Progress1",IDC_TAB2SA_Scope,"msctls_progress32",
                    PBS_SMOOTH,10,35,220,10,WS_EX_STATICEDGE
    CHECKBOX        "Search Multiple Levels",IDC_TAB2SA_TraverseMagicLevels,
                    10,71,87,10
    LTEXT           "Current Magic Level:",IDC_TAB2SA_Static2,10,59,66,8
    RTEXT           "500",IDC_TAB2SA_MagicLevel,90,59,15,8
    CONTROL         "",IDC_TAB2SA_Static1,"Static",SS_ETCHEDFRAME,5,5,230,45
    CONTROL         "",IDC_TAB2SA_Static5,"Static",SS_ETCHEDFRAME,115,55,120,
                    92
    CONTROL         "",IDC_TAB2SA_Static3,"Static",SS_ETCHEDFRAME,5,88,105,
                    36
```

```
    CHECKBOX        "Manual Hit Selection",IDC_TAB2SA_HitSelection,121,60,
                    109,10
    LISTBOX         IDC_TAB2SA_HitList,120,73,110,69,LBS_SORT |
                    LBS_NOINTEGRALHEIGHT | WS_DISABLED | WS_VSCROLL |
                    WS_TABSTOP
    CONTROL         "",IDC_TAB2SA_Static6,"Static",SS_ETCHEDFRAME,5,55,105,
                    30
    CHECKBOX        "?",IDC_CHELP,220,153,16,15,BS_ICON | BS_CENTER |
                    BS_VCENTER | BS_PUSHLIKE
    CONTROL         "",IDC_TAB2SA_Static7,"Static",SS_ETCHEDFRAME,5,129,105,
                    18
    CTEXT           "Speed",IDC_TAB2AS_Speed,10,134,95,8
END

IDD_TAB2ItemList DIALOGEX 0, 0, 67, 231
STYLE WS_POPUP | WS_CAPTION | WS_SYSMENU
EXSTYLE WS_EX_TOOLWINDOW
CAPTION "Item List"
FONT 8, "MS Sans Serif"
BEGIN
    LISTBOX         IDC_TAB2ItemList,0,0,67,230,LBS_NOINTEGRALHEIGHT | NOT
                    WS_BORDER | WS_VSCROLL | WS_TABSTOP,WS_EX_CLIENTEDGE
END

IDD_TAB2Grid DIALOGEX 0, 0, 192, 118
STYLE DS_CENTER | WS_POPUP | WS_CAPTION | WS_SYSMENU
EXSTYLE WS_EX_TOOLWINDOW
CAPTION "Inventory Grid"
FONT 8, "MS Sans Serif"
BEGIN
    CONTROL         261,IDC_TAB2Grid_Grid,"Static",SS_BITMAP |
                    SS_REALSIZEIMAGE,0,0,59,72
END

IDD_TAB2ExGrid DIALOGEX 0, 0, 456, 298
STYLE WS_CHILD
FONT 8, "MS Sans Serif"
BEGIN
    GROUPBOX        "Inventory",IDC_STATIC,6,3,220,262,BS_CENTER,
                    WS_EX_TRANSPARENT
    GROUPBOX        "Selected Item",IDC_STATIC,230,3,220,123,BS_CENTER,
                    WS_EX_TRANSPARENT
    GROUPBOX        "Horadric Cube",IDC_STATIC,5,265,63,27,BS_CENTER,
                    WS_EX_TRANSPARENT
    GROUPBOX        "Belt",IDC_STATIC,71,265,85,27,BS_CENTER,
                    WS_EX_TRANSPARENT
    GROUPBOX        "Stash",IDC_STATIC,159,264,122,28,BS_CENTER,
                    WS_EX_TRANSPARENT
    CONTROL         "ItemTree",IDC_TAB2_ItemTree,"SysTreeView32",
                    TVS_HASBUTTONS | TVS_HASLINES | TVS_LINESATROOT |
                    TVS_NOTOOLTIPS | TVS_FULLROWSELECT | WS_TABSTOP,288,139,
                    160,150,WS_EX_STATICEDGE
    CONTROL         264,IDC_TAB2_Body,"Static",SS_BITMAP | SS_REALSIZEIMAGE,
                    9,12,213,152
    CONTROL         262,IDC_TAB2_Inv,"Static",SS_BITMAP | SS_REALSIZEIMAGE,
                    20,166,194,72
    CONTROL         "Expert Toolbox",IDC_TAB2_ExpertMode,"Button",
                    BS_AUTOCHECKBOX | WS_TABSTOP,380,112,63,10
    PUSHBUTTON      "Save Item",IDC_TAB2_Save,230,130,51,12
    PUSHBUTTON      "Load Item",IDC_TAB2_Load,230,142,51,12,BS_CENTER |
                    BS_VCENTER
    CONTROL         287,IDC_TAB2_CopyBuffer,"Static",SS_BITMAP |
                    SS_REALSIZEIMAGE,235,166,39,72
    GROUPBOX        "Built-In Item Tree (Drag into Inventory)",IDC_STATIC,
                    285,130,165,163,BS_CENTER,WS_EX_TRANSPARENT
    GROUPBOX        "Copy Buffer",IDC_STATIC,230,156,52,85,BS_CENTER,
                    WS_EX_TRANSPARENT
    PUSHBUTTON      "Randomize",IDC_TAB2_AttrRandom,235,110,50,12,
                    WS_DISABLED
    CONTROL         "",IDC_TAB2_RichText,"RICHEDIT",ES_CENTER | ES_MULTILINE |
                    ES_AUTOVSCROLL | ES_READONLY | ES_NUMBER | WS_VSCROLL |
                    WS_TABSTOP,235,12,207,96,WS_EX_STATICEDGE
    PUSHBUTTON      "Batch Action",IDC_TAB2_Batch,320,110,50,12
    PUSHBUTTON      "<",IDC_TAB2_HistoryBack,290,110,10,12,WS_DISABLED
    PUSHBUTTON      ">",IDC_TAB2_HistoryNext,300,110,10,12,WS_DISABLED
    CHECKBOX        "Open",IDC_TAB2_OpenCube,9,277,55,11,BS_PUSHLIKE
    CHECKBOX        "Open",IDC_TAB2_OpenBelt,75,277,78,11,BS_PUSHLIKE
    CHECKBOX        "Open",IDC_TAB2_OpenStash,163,277,113,11,BS_PUSHLIKE
END
```

```
//////////////////////////////////////////////////////////////////////////////
//
// DESIGNINFO
//

#ifdef APSTUDIO_INVOKED
GUIDELINES DESIGNINFO DISCARDABLE
BEGIN
    IDD_TAB3, DIALOG
    BEGIN
        RIGHTMARGIN, 275
        BOTTOMMARGIN, 91
    END

    IDD_TAB0, DIALOG
    BEGIN
        RIGHTMARGIN, 309
        BOTTOMMARGIN, 195
    END

    IDD_TAB1, DIALOG
    BEGIN
        RIGHTMARGIN, 309
        TOPMARGIN, 1
        BOTTOMMARGIN, 195
    END

    IDD_TAB2, DIALOG
    BEGIN
        RIGHTMARGIN, 455
        BOTTOMMARGIN, 304
    END

    IDD_TAB4, DIALOG
    BEGIN
        RIGHTMARGIN, 309
        BOTTOMMARGIN, 195
    END

    IDD_TAB5, DIALOG
    BEGIN
        RIGHTMARGIN, 309
        BOTTOMMARGIN, 195
    END

    IDD_COWLEVEL, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 435
        TOPMARGIN, 7
        BOTTOMMARGIN, 259
    END

    IDD_PROGRESS, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 177
        TOPMARGIN, 7
        BOTTOMMARGIN, 66
    END

    IDD_INFO, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 250
        TOPMARGIN, 7
        BOTTOMMARGIN, 136
    END

    IDD_TAB2E, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 153
        TOPMARGIN, 7
        BOTTOMMARGIN, 263
    END
```

```
IDD_TAB2Rnd, DIALOG
BEGIN
    LEFTMARGIN, 7
    RIGHTMARGIN, 107
    TOPMARGIN, 7
    BOTTOMMARGIN, 271
END

IDD_TAB2Magic, DIALOG
BEGIN
    LEFTMARGIN, 7
    RIGHTMARGIN, 328
    TOPMARGIN, 7
    BOTTOMMARGIN, 343
END

IDD_TAB2SS, DIALOG
BEGIN
    LEFTMARGIN, 7
    RIGHTMARGIN, 148
    TOPMARGIN, 7
    BOTTOMMARGIN, 63
END

IDD_TAB2Rare, DIALOG
BEGIN
    LEFTMARGIN, 7
    RIGHTMARGIN, 438
    TOPMARGIN, 7
    BOTTOMMARGIN, 302
END

IDD_SAVE, DIALOG
BEGIN
    LEFTMARGIN, 7
    RIGHTMARGIN, 200
    TOPMARGIN, 7
    BOTTOMMARGIN, 117
END

IDD_TAB2Quantity, DIALOG
BEGIN
    LEFTMARGIN, 7
    RIGHTMARGIN, 110
    TOPMARGIN, 7
    BOTTOMMARGIN, 88
END

IDD_TAB2Durability, DIALOG
BEGIN
    LEFTMARGIN, 7
    RIGHTMARGIN, 103
    TOPMARGIN, 7
    BOTTOMMARGIN, 72
END

IDD_TAB2Defense, DIALOG
BEGIN
    LEFTMARGIN, 7
    RIGHTMARGIN, 103
    TOPMARGIN, 7
    BOTTOMMARGIN, 67
END

IDD_TAB2Gems1, DIALOG
BEGIN
    LEFTMARGIN, 7
    RIGHTMARGIN, 288
    TOPMARGIN, 7
    BOTTOMMARGIN, 67
END

IDD_TAB2Gems2, DIALOG
BEGIN
    LEFTMARGIN, 7
    RIGHTMARGIN, 288
    TOPMARGIN, 7
    BOTTOMMARGIN, 117
END
```

```
IDD_TAB2Gems7, DIALOG
BEGIN
    LEFTMARGIN, 7
    RIGHTMARGIN, 488
    TOPMARGIN, 7
    BOTTOMMARGIN, 216
END

IDD_TAB2Ear, DIALOG
BEGIN
    LEFTMARGIN, 7
    RIGHTMARGIN, 154
    TOPMARGIN, 7
    BOTTOMMARGIN, 112
END

IDD_NEW, DIALOG
BEGIN
    LEFTMARGIN, 7
    RIGHTMARGIN, 155
    TOPMARGIN, 7
    BOTTOMMARGIN, 59
END

IDD_RENAME, DIALOG
BEGIN
    LEFTMARGIN, 7
    RIGHTMARGIN, 155
    TOPMARGIN, 7
    BOTTOMMARGIN, 94
END

IDD_TAB2RingImage, DIALOG
BEGIN
    LEFTMARGIN, 7
    RIGHTMARGIN, 181
    TOPMARGIN, 7
    BOTTOMMARGIN, 59
END

IDD_TAB2AmuletImage, DIALOG
BEGIN
    LEFTMARGIN, 7
    RIGHTMARGIN, 123
    TOPMARGIN, 7
    BOTTOMMARGIN, 59
END

IDD_IBWSR, DIALOG
BEGIN
    RIGHTMARGIN, 281
    BOTTOMMARGIN, 99
END

IDD_UOPTIONS, DIALOG
BEGIN
    LEFTMARGIN, 7
    RIGHTMARGIN, 190
    TOPMARGIN, 7
    BOTTOMMARGIN, 128
END

IDD_TAB2Gems3, DIALOG
BEGIN
    LEFTMARGIN, 7
    RIGHTMARGIN, 288
    TOPMARGIN, 7
    BOTTOMMARGIN, 167
END

IDD_HELP, DIALOG
BEGIN
    LEFTMARGIN, 7
    RIGHTMARGIN, 131
    TOPMARGIN, 7
    BOTTOMMARGIN, 267
END
```

```
    IDD_EOPTIONS, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 255
        TOPMARGIN, 7
        BOTTOMMARGIN, 184
    END

    IDD_TAB2S, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 232
        TOPMARGIN, 7
        BOTTOMMARGIN, 166
    END

    IDD_TAB2ItemList, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 60
        TOPMARGIN, 7
        BOTTOMMARGIN, 224
    END

    IDD_TAB2Grid, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 185
        TOPMARGIN, 7
        BOTTOMMARGIN, 111
    END

    IDD_TAB2ExGrid, DIALOG
    BEGIN
        RIGHTMARGIN, 455
        BOTTOMMARGIN, 267
    END
END
#endif    // APSTUDIO_INVOKED


////////////////////////////////////////////////////////////////////////////
//
// Menu
//

IDR_MAINDIALOG MENU DISCARDABLE
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "&New\tCtrl+N",                IDR_NEW
        MENUITEM "&Open\tCtrl+O",               IDR_OPEN
        MENUITEM "&Save\tCtrl+S",               IDR_SAVE
        MENUITEM "&Reload\tCtrl+R",             IDR_RELOAD
        MENUITEM "&Close",                      IDR_CLOSE
        MENUITEM SEPARATOR
        MENUITEM "O&ptions",                    IDR_UOPTIONS
        MENUITEM "Save Text Summary",           IDR_TEXTFILE
        MENUITEM SEPARATOR
        MENUITEM "&Exit",                       IDR_EXIT
    END
    MENUITEM "&Info",                       IDR_INFO, HELP
END

IDR_BATCH MENU DISCARDABLE
BEGIN
    POPUP "TAB1"
    BEGIN
        MENUITEM "Restore Constitution",        IDR_TAB1_RestoreConstitution
        POPUP "Set All Character Stats to"
        BEGIN
            MENUITEM "40 Points",                   IDR_TAB1_SetAllStats40
            MENUITEM "60 Points",                   IDR_TAB1_SetAllStats60
            MENUITEM "80 Points",                   IDR_TAB1_SetAllStats80
            MENUITEM "100 Points",                  IDR_TAB1_SetAllStats100
            MENUITEM "120 Points",                  IDR_TAB1_SetAllStats120
            MENUITEM "140 Points",                  IDR_TAB1_SetAllStats140
            MENUITEM "160 Points",                  IDR_TAB1_SetAllStats160
            MENUITEM "180 Points",                  IDR_TAB1_SetAllStats180
```

```
        MENUITEM "200 Points",                  IDR_TAB1_SetAllStats200
        MENUITEM "250 Points",                  IDR_TAB1_SetAllStats250
        MENUITEM "300 Points",                  IDR_TAB1_SetAllStats300
        MENUITEM "350 Points",                  IDR_TAB1_SetAllStats350
        MENUITEM "400 Points",                  IDR_TAB1_SetAllStats400
        MENUITEM "450 Points",                  IDR_TAB1_SetAllStats450
        MENUITEM "500 Points",                  IDR_TAB1_SetAllStats500
    END
    POPUP "Set All Constitution Values to"
    BEGIN
        MENUITEM "400 Points",                  IDR_TAB1_SetAllConstitution400

        MENUITEM "600 Points",                  IDR_TAB1_SetAllConstitution600

        MENUITEM "800 Points",                  IDR_TAB1_SetAllConstitution800

        MENUITEM "1000 Points",                 IDR_TAB1_SetAllConstitution1000

        MENUITEM "1500 Points",                 IDR_TAB1_SetAllConstitution1500

        MENUITEM "2000 Points",                 IDR_TAB1_SetAllConstitution2000

        MENUITEM "3000 Points",                 IDR_TAB1_SetAllConstitution3000

        MENUITEM "5000 Points",                 IDR_TAB1_SetAllConstitution5000

    END
END
POPUP "TAB2"
BEGIN
    POPUP "Fill Belt Slots"
    BEGIN
        POPUP "All Columns with"
        BEGIN
            MENUITEM "Minor Healing Potions",       IDR_TAB2_BeltMinorHealing

            MENUITEM "Light Healing Potions",       IDR_TAB2_BeltLightHealing

            MENUITEM "Healing Potions",             IDR_TAB2_BeltHealing
            MENUITEM "Greater Healing Potions",     IDR_TAB2_BeltGreaterHealing

            MENUITEM "Super Healing Potions",       IDR_TAB2_BeltSuperHealing

            MENUITEM SEPARATOR
            MENUITEM "Minor Mana Potions",          IDR_TAB2_BeltMinorMana

            MENUITEM "Light Mana Potions",          IDR_TAB2_BeltLightMana

            MENUITEM "Mana Potions",                IDR_TAB2_BeltMana
            MENUITEM "Greater Mana Potions",        IDR_TAB2_BeltGreaterMana

            MENUITEM "Super Mana Potions",          IDR_TAB2_BeltSuperMana

            MENUITEM SEPARATOR
            MENUITEM "Rejuvenation Potions",        IDR_TAB2_BeltRejuv
            MENUITEM "Full Rejuventation Potions",  IDR_TAB2_BeltFullRejuv

        END
        POPUP "Column 1 with"
        BEGIN
            MENUITEM "Minor Healing Potions",       IDR_TAB2_Slot1MinorHealing

            MENUITEM "Light Healing Potions",       IDR_TAB2_Slot1LightHealing

            MENUITEM "Healing Potions",             IDR_TAB2_Slot1Healing

            MENUITEM "Greater Healing Potions",     IDR_TAB2_Slot1GreaterHealing

            MENUITEM "Super Healing Potions",       IDR_TAB2_Slot1SuperHealing

            MENUITEM SEPARATOR
            MENUITEM "Minor Mana Potions",          IDR_TAB2_Slot1MinorMana

            MENUITEM "Light Mana Potions",          IDR_TAB2_Slot1LightMana

            MENUITEM "Mana Potions",                IDR_TAB2_Slot1Mana
            MENUITEM "Greater Mana Potions",        IDR_TAB2_Slot1GreaterMana

            MENUITEM "Super Mana Potions",          IDR_TAB2_Slot1SuperMana
```

```
    MENUITEM SEPARATOR
    MENUITEM "Rejuvenation Potions",        IDR_TAB2_Slot1Rejuv
    MENUITEM "Full Rejuventation Potions",  IDR_TAB2_Slot1FullRejuv

END
POPUP "Column 2 with"
BEGIN
    MENUITEM "Minor Healing Potions",       IDR_TAB2_Slot2MinorHealing

    MENUITEM "Light Healing Potions",       IDR_TAB2_Slot2LightHealing

    MENUITEM "Healing Potions",             IDR_TAB2_Slot2Healing

    MENUITEM "Super Healing Potions",       IDR_TAB2_Slot2SuperHealing

    MENUITEM "Greater Healing Potions",     IDR_TAB2_Slot2GreaterHealing

    MENUITEM SEPARATOR
    MENUITEM "Minor Mana Potions",          IDR_TAB2_Slot2MinorMana

    MENUITEM "Light Mana Potions",          IDR_TAB2_Slot2LightMana

    MENUITEM "Mana Potions",                IDR_TAB2_Slot2Mana
    MENUITEM "Greater Mana Potions",        IDR_TAB2_Slot2GreaterMana

    MENUITEM "Super Mana Potions",          IDR_TAB2_Slot2SuperMana

    MENUITEM SEPARATOR
    MENUITEM "Rejuvenation Potions",        IDR_TAB2_Slot2Rejuv
    MENUITEM "Full Rejuventation Potions",  IDR_TAB2_Slot2FullRejuv

END
POPUP "Column 3 with"
BEGIN
    MENUITEM "Minor Healing Potions",       IDR_TAB2_Slot3MinorHealing

    MENUITEM "Light Healing Potions",       IDR_TAB2_Slot3LightHealing

    MENUITEM "Healing Potions",             IDR_TAB2_Slot3Healing

    MENUITEM "Greater Healing Potions",     IDR_TAB2_Slot3GreaterHealing

    MENUITEM "Super Healing Potions",       IDR_TAB2_Slot3SuperHealing

    MENUITEM SEPARATOR
    MENUITEM "Minor Mana Potions",          IDR_TAB2_Slot3MinorMana

    MENUITEM "Light Mana Potions",          IDR_TAB2_Slot3LightMana

    MENUITEM "Mana Potions",                IDR_TAB2_Slot3Mana
    MENUITEM "Greater Mana Potions",        IDR_TAB2_Slot3GreaterMana

    MENUITEM "Super Mana Potions",          IDR_TAB2_Slot3SuperMana

    MENUITEM SEPARATOR
    MENUITEM "Rejuvenation Potions",        IDR_TAB2_Slot3Rejuv
    MENUITEM "Full Rejuventation Potions",  IDR_TAB2_Slot3FullRejuv

END
POPUP "Column 4 with"
BEGIN
    MENUITEM "Minor Healing Potions",       IDR_TAB2_Slot4MinorHealing

    MENUITEM "Light Healing Potions",       IDR_TAB2_Slot4LightHealing

    MENUITEM "Healing Potions",             IDR_TAB2_Slot4Healing

    MENUITEM "Greater Healing Potions",     IDR_TAB2_Slot4GreaterHealing

    MENUITEM "Super Healing Potions",       IDR_TAB2_Slot4SuperHealing

    MENUITEM SEPARATOR
    MENUITEM "Minor Mana Potions",          IDR_TAB2_Slot4MinorMana

    MENUITEM "Light Mana Potions",          IDR_TAB2_Slot4LightMana

    MENUITEM "Mana Potions",                IDR_TAB2_Slot4Mana
    MENUITEM "Greater Mana Potions",        IDR_TAB2_Slot4GreaterMana
```

```
                MENUITEM "Super Mana Potions",          IDR_TAB2_Slot4SuperMana

                MENUITEM SEPARATOR
                MENUITEM "Rejuvenation Potions",        IDR_TAB2_Slot4Rejuv
                MENUITEM "Full Rejuventation Potions",  IDR_TAB2_Slot4FullRejuv

            END
            MENUITEM "Empty Belt",                  IDR_TAB2_BeltEmpty
        END
        MENUITEM "Repair All Items",            IDR_TAB2_RepairAll
    END
    POPUP "TAB3"
    BEGIN
        MENUITEM "Maximize All",                IDR_TAB3_MaximizeAll
        MENUITEM SEPARATOR
        MENUITEM "Set All Skills to 0",         IDR_TAB3_SetAll0
        MENUITEM "Set All Skills to 4",         IDR_TAB3_SetAll4
        MENUITEM "Set All Skills to 8",         IDR_TAB3_SetAll8
        MENUITEM "Set All Skills to 12",        IDR_TAB3_SetAll12
        MENUITEM "Set All Skills to 16",        IDR_TAB3_SetAll16
        MENUITEM "Set All Skills to 20",        IDR_TAB3_SetAll20
    END
    MENUITEM "TAB4",                        65535
    POPUP "TAB5"
    BEGIN
        MENUITEM "Activate Waypoints in this Difficulty",
                                                IDR_TAB5_ActivateHereAll
        MENUITEM "Deactivate Waypoints in this Difficulty",
                                                IDR_TAB5_DeactivateHereAll
        MENUITEM SEPARATOR
        MENUITEM "Activate Waypoints in All Difficulties",
                                                IDR_TAB5_ActivateAllAll
        MENUITEM "Deactivate Waypoints in All Difficulties",
                                                IDR_TAB5_DeactivateAllAll
    END
END


//////////////////////////////////////////////////////////////////////////
//
// Bitmap
//

IDB_PLUS                BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Other\\Plus.bmp"
IDB_TAB1                BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Other\\Tab1.bmp"
IDB_TAB2                BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Other\\Tab2.bmp"
IDB_TAB3                BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Other\\Tab3.bmp"
IDB_TAB4                BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Other\\Tab4.bmp"
IDB_TAB5                BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Other\\Tab5.bmp"
IDB_JAMELLA             BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Other\\Jamella.bmp"
IDB_SKILL_S30           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\S30.bmp"
IDB_SKILL_S29           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\S29.bmp"
IDB_SKILL_S28           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\S28.bmp"
IDB_SKILL_S27           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\S27.bmp"
IDB_SKILL_S26           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\S26.bmp"
IDB_SKILL_S25           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\S25.bmp"
IDB_SKILL_S24           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\S24.bmp"
IDB_SKILL_S23           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\S23.bmp"
IDB_SKILL_S22           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\S22.bmp"
IDB_SKILL_S21           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\S21.bmp"
IDB_SKILL_S20           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\S20.bmp"
IDB_SKILL_S19           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\S19.bmp"
IDB_SKILL_S18           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\S18.bmp"
IDB_SKILL_S17           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\S17.bmp"
IDB_SKILL_S16           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\S16.bmp"
IDB_SKILL_S15           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\S15.bmp"
IDB_SKILL_S14           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\S14.bmp"
IDB_SKILL_S13           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\S13.bmp"
IDB_SKILL_S12           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\S12.bmp"
IDB_SKILL_S11           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\S11.bmp"
IDB_SKILL_S10           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\S10.bmp"
IDB_SKILL_S09           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\S09.bmp"
IDB_SKILL_S08           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\S08.bmp"
IDB_SKILL_S07           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\S07.bmp"
IDB_SKILL_S06           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\S06.bmp"
IDB_SKILL_S05           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\S05.bmp"
IDB_SKILL_S04           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\S04.bmp"
IDB_SKILL_S03           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\S03.bmp"
```

```
IDB_SKILL_S02           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\S02.bmp"
IDB_SKILL_S01           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\S01.bmp"
IDB_SKILL_N30           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\N30.bmp"
IDB_SKILL_N29           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\N29.bmp"
IDB_SKILL_N28           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\N28.bmp"
IDB_SKILL_N27           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\N27.bmp"
IDB_SKILL_N26           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\N26.bmp"
IDB_SKILL_N25           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\N25.bmp"
IDB_SKILL_N24           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\N24.bmp"
IDB_SKILL_N23           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\N23.bmp"
IDB_SKILL_N22           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\N22.bmp"
IDB_SKILL_N21           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\N21.bmp"
IDB_SKILL_N20           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\N20.bmp"
IDB_SKILL_N19           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\N19.bmp"
IDB_SKILL_N18           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\N18.bmp"
IDB_SKILL_N17           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\N17.bmp"
IDB_SKILL_N16           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\N16.bmp"
IDB_SKILL_N15           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\N15.bmp"
IDB_SKILL_N14           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\N14.bmp"
IDB_SKILL_N13           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\N13.bmp"
IDB_SKILL_N12           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\N12.bmp"
IDB_SKILL_N11           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\N11.bmp"
IDB_SKILL_N10           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\N10.bmp"
IDB_SKILL_N09           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\N09.bmp"
IDB_SKILL_N08           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\N08.bmp"
IDB_SKILL_N07           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\N07.bmp"
IDB_SKILL_N06           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\N06.bmp"
IDB_SKILL_N05           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\N05.bmp"
IDB_SKILL_N04           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\N04.bmp"
IDB_SKILL_N03           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\N03.bmp"
IDB_SKILL_N02           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\N02.bmp"
IDB_SKILL_N01           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\N01.bmp"
IDB_SKILL_A30           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\A30.bmp"
IDB_SKILL_A29           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\A29.bmp"
IDB_SKILL_A28           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\A28.bmp"
IDB_SKILL_A27           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\A27.bmp"
IDB_SKILL_A26           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\A26.bmp"
IDB_SKILL_A25           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\A25.bmp"
IDB_SKILL_A24           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\A24.bmp"
IDB_SKILL_A23           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\A23.bmp"
IDB_SKILL_A22           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\A22.bmp"
IDB_SKILL_A21           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\A21.bmp"
IDB_SKILL_A20           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\A20.bmp"
IDB_SKILL_A19           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\A19.bmp"
IDB_SKILL_A18           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\A18.bmp"
IDB_SKILL_A17           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\A17.bmp"
IDB_SKILL_A16           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\A16.bmp"
IDB_SKILL_A15           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\A15.bmp"
IDB_SKILL_A14           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\A14.bmp"
IDB_SKILL_A13           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\A13.bmp"
IDB_SKILL_A12           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\A12.bmp"
IDB_SKILL_A11           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\A11.bmp"
IDB_SKILL_A10           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\A10.bmp"
IDB_SKILL_A09           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\A09.bmp"
IDB_SKILL_A08           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\A08.bmp"
IDB_SKILL_A07           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\A07.bmp"
IDB_SKILL_A06           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\A06.bmp"
IDB_SKILL_A05           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\A05.bmp"
IDB_SKILL_A04           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\A04.bmp"
IDB_SKILL_A03           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\A03.bmp"
IDB_SKILL_A02           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\A02.bmp"
IDB_SKILL_A01           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\A01.bmp"
IDB_SKILL_P30           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\P30.bmp"
IDB_SKILL_P29           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\P29.bmp"
IDB_SKILL_P28           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\P28.bmp"
IDB_SKILL_P27           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\P27.bmp"
IDB_SKILL_P26           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\P26.bmp"
IDB_SKILL_P25           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\P25.bmp"
IDB_SKILL_P24           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\P24.bmp"
IDB_SKILL_P23           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\P23.bmp"
IDB_SKILL_P22           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\P22.bmp"
IDB_SKILL_P21           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\P21.bmp"
IDB_SKILL_P20           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\P20.bmp"
IDB_SKILL_P19           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\P19.bmp"
IDB_SKILL_P18           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\P18.bmp"
IDB_SKILL_P17           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\P17.bmp"
IDB_SKILL_P16           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\P16.bmp"
IDB_SKILL_P15           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\P15.bmp"
IDB_SKILL_P14           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\P14.bmp"
```

```
IDB_SKILL_P13           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\P13.bmp"
IDB_SKILL_P12           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\P12.bmp"
IDB_SKILL_P11           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\P11.bmp"
IDB_SKILL_P10           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\P10.bmp"
IDB_SKILL_P09           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\P09.bmp"
IDB_SKILL_P08           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\P08.bmp"
IDB_SKILL_P07           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\P07.bmp"
IDB_SKILL_P06           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\P06.bmp"
IDB_SKILL_P05           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\P05.bmp"
IDB_SKILL_P04           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\P04.bmp"
IDB_SKILL_P03           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\P03.bmp"
IDB_SKILL_P02           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\P02.bmp"
IDB_SKILL_P01           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\P01.bmp"
IDB_SKILL_B30           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\B30.bmp"
IDB_SKILL_B29           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\B29.bmp"
IDB_SKILL_B28           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\B28.bmp"
IDB_SKILL_B27           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\B27.bmp"
IDB_SKILL_B26           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\B26.bmp"
IDB_SKILL_B25           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\B25.bmp"
IDB_SKILL_B24           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\B24.bmp"
IDB_SKILL_B23           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\B23.bmp"
IDB_SKILL_B22           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\B22.bmp"
IDB_SKILL_B21           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\B21.bmp"
IDB_SKILL_B20           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\B20.bmp"
IDB_SKILL_B19           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\B19.bmp"
IDB_SKILL_B18           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\B18.bmp"
IDB_SKILL_B17           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\B17.bmp"
IDB_SKILL_B16           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\B16.bmp"
IDB_SKILL_B15           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\B15.bmp"
IDB_SKILL_B14           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\B14.bmp"
IDB_SKILL_B13           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\B13.bmp"
IDB_SKILL_B12           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\B12.bmp"
IDB_SKILL_B11           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\B11.bmp"
IDB_SKILL_B10           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\B10.bmp"
IDB_SKILL_B09           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\B09.bmp"
IDB_SKILL_B08           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\B08.bmp"
IDB_SKILL_B07           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\B07.bmp"
IDB_SKILL_B06           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\B06.bmp"
IDB_SKILL_B05           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\B05.bmp"
IDB_SKILL_B04           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\B04.bmp"
IDB_SKILL_B03           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\B03.bmp"
IDB_SKILL_B02           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\B02.bmp"
IDB_SKILL_B01           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Skills\\B01.bmp"
IDB_WAYPOINT_OFF        BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Other\\WaypointOff.bmp"
IDB_WAYPOINT_ON         BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Other\\WaypointOn.bmp"
IDB_QUEST11             BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Quests\\1-1.bmp"
IDB_QUEST12             BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Quests\\1-2.bmp"
IDB_QUEST13             BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Quests\\1-3.bmp"
IDB_QUEST14             BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Quests\\1-4.bmp"
IDB_QUEST15             BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Quests\\1-5.bmp"
IDB_QUEST16             BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Quests\\1-6.bmp"
IDB_QUEST21             BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Quests\\2-1.bmp"
IDB_QUEST22             BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Quests\\2-2.bmp"
IDB_QUEST23             BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Quests\\2-3.bmp"
IDB_QUEST24             BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Quests\\2-4.bmp"
IDB_QUEST25             BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Quests\\2-5.bmp"
IDB_QUEST26             BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Quests\\2-6.bmp"
IDB_QUEST31             BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Quests\\3-1.bmp"
IDB_QUEST32             BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Quests\\3-2.bmp"
IDB_QUEST33             BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Quests\\3-3.bmp"
IDB_QUEST34             BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Quests\\3-4.bmp"
IDB_QUEST35             BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Quests\\3-5.bmp"
IDB_QUEST36             BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Quests\\3-6.bmp"
IDB_QUEST41             BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Quests\\4-1.bmp"
IDB_QUEST42             BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Quests\\4-2.bmp"
IDB_QUEST43             BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Quests\\4-3.bmp"
IDB_COWLEVEL_Cube       BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Other\\CowLevelCube.bmp"
IDB_COWLEVEL_Portal     BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Other\\CowLevelPortal.bmp"
IDB_COWLEVEL_Leg        BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Other\\CowLevelLeg.bmp"
IDB_GRID_STASH          BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Other\\GridStash.bmp"
IDB_GRID_CUBE           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Other\\GridCube.bmp"
IDB_GRID_INV            BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Other\\GridInventory.bmp"
IDB_GRID_BELT           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Other\\GridBelt.bmp"
IDB_INV_WHOLE           BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Other\\InvWhole.bmp"
IDB_ITEM_UNKNOWN        BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Other\\ItemUnknown.bmp"
IDB_ITEM_THEGNASHER     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\thegnasher.bmp"
IDB_ITEM_DEATHSPADE     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\deathspade.bmp"
IDB_ITEM_BERSERKERSHATCHET BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\berserkershatchet.bmp"
IDB_ITEM_TANCREDSCROWBILL BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\tancredscrowbill.bmp"
```

```
IDB_ITEM_MINDREND       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\mindrend.bmp"
IDB_ITEM_RAKESCAR       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\rakescar.bmp"
IDB_ITEM_FECHMARSAXE    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\fechmarsaxe.bmp"
IDB_ITEM_GORESHOVEL     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\goreshovel.bmp"
IDB_ITEM_THECHIEFTAN    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\thechieftan.bmp"
IDB_ITEM_BRAINHEW       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\brainhew.bmp"
IDB_ITEM_THEHUMONGOUS   BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\thehumongous.bmp"
IDB_ITEM_IROSTORCH      BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\irostorch.bmp"
IDB_ITEM_MAELSTROMWRATH BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\maelstromwrath.bmp"
IDB_ITEM_GRAVENSPINE    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\gravenspine.bmp"
IDB_ITEM_INFERNALTORCH  BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\infernaltorch.bmp"
IDB_ITEM_UMESLAMENT     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\umeslament.bmp"
IDB_ITEM_FELLOAK        BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\felloak.bmp"
IDB_ITEM_KNELLSTRIKER   BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\knellstriker.bmp"
IDB_ITEM_CIVERBSCUDGEL  BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\civerbscudgel.bmp"
IDB_ITEM_RUSTHANDLE     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\rusthandle.bmp"
IDB_ITEM_MILABREGASROD  BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\milabregasrod.bmp"
IDB_ITEM_STOUTNAIL      BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\stoutnail.bmp"
IDB_ITEM_CRUSHFLANGE    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\crushflange.bmp"
IDB_ITEM_BLOODRISE      BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\bloodrise.bmp"
IDB_ITEM_THEGENERALSTANDOLIGA BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\thegeneralstandoliga.bmp"
IDB_ITEM_IRONSTONE      BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\ironstone.bmp"
IDB_ITEM_BONESNAP       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\bonesnap.bmp"
IDB_ITEM_STEELDRIVER    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\steeldriver.bmp"
IDB_ITEM_RIXOTSKEEN     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\rixotskeen.bmp"
IDB_ITEM_BLOODCRESCENT  BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\bloodcrescent.bmp"
IDB_ITEM_KRINTIZSSKEWER BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\krintizsskewer.bmp"
IDB_ITEM_GLEAMSCYTHE    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\gleamscythe.bmp"
IDB_ITEM_AZUREWRATH     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\azurewrath.bmp"
IDB_ITEM_ISENHARTSLIGHTBRAND BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\isenhartslightbrand.bmp"
IDB_ITEM_GRISWOLDSEDGE  BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\griswoldsedge.bmp"
IDB_ITEM_CLEGLAWSTOOTH  BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\cleglawstooth.bmp"
IDB_ITEM_HELLPLAGUE     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\hellplague.bmp"
IDB_ITEM_DEATHSTOUCH    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\deathstouch.bmp"
IDB_ITEM_SHADOWFANG     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\shadowfang.bmp"
IDB_ITEM_SOULFLAY       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\soulflay.bmp"
IDB_ITEM_KINEMILSAWL    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\kinemilsawl.bmp"
IDB_ITEM_BLACKTONGUE    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\blacktongue.bmp"
IDB_ITEM_RIPSAW         BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\ripsaw.bmp"
IDB_ITEM_THEPATRIARCH   BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\thepatriarch.bmp"
IDB_ITEM_THEDIGGLER     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\thediggler.bmp"
IDB_ITEM_THEJADETANDO   BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\thejadetando.bmp"
IDB_ITEM_IRICESSHARD    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\iricesshard.bmp"
IDB_ITEM_THEDRAGONCHANG BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\thedragonchang.bmp"
IDB_ITEM_RAZORTINE      BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\razortine.bmp"
IDB_ITEM_BLOODTHIEF     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\bloodthief.bmp"
IDB_ITEM_LANCEOFYAGGAI  BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\lanceofyaggai.bmp"
IDB_ITEM_THETANNRGOREROD BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\thetannrgorerod.bmp"
IDB_ITEM_DIMOAKSHEW     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\dimoakshew.bmp"
IDB_ITEM_STEELGOAD      BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\steelgoad.bmp"
IDB_ITEM_SOULHARVEST    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\soulharvest.bmp"
IDB_ITEM_THEBATTLEBRANCH BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\thebattlebranch.bmp"
IDB_ITEM_WOESTAVE       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\woestave.bmp"
IDB_ITEM_BANEASH        BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\baneash.bmp"
IDB_ITEM_SERPENTLORD    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\serpentlord.bmp"
IDB_ITEM_LAZARUSSPIRE   BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\lazarusspire.bmp"
IDB_ITEM_CATHANSRULE    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\cathansrule.bmp"
IDB_ITEM_THESALAMANDER  BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\thesalamander.bmp"
IDB_ITEM_ARCANNASDEATHWAND BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\arcannasdeathwand.bmp"
IDB_ITEM_THEIRONLANGBONG BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\theironlangbong.bmp"
IDB_ITEM_PLUCKEYE       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\pluckeye.bmp"
IDB_ITEM_WITHERSTRING   BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\witherstring.bmp"
IDB_ITEM_RIMERAVEN      BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\rimeraven.bmp"
IDB_ITEM_PIERCERIB      BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\piercerib.bmp"
IDB_ITEM_PULLSPITE      BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\pullspite.bmp"
IDB_ITEM_VIDALASBARB    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\vidalasbarb.bmp"
IDB_ITEM_WIZENDRAW      BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\wizendraw.bmp"
IDB_ITEM_ARCTICHORN     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\arctichorn.bmp"
IDB_ITEM_HELLCLAP       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\hellclap.bmp"
IDB_ITEM_BLASTBARK      BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\blastbark.bmp"
IDB_ITEM_LEADCROW       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\leadcrow.bmp"
IDB_ITEM_ICHORSTING     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\ichorsting.bmp"
IDB_ITEM_HELLCAST       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\hellcast.bmp"
IDB_ITEM_DOOMSPITTLE    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\doomspittle.bmp"
IDB_ITEM_RANCIDGAS      BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\rancidgas.bmp"
IDB_ITEM_OIL            BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\oil.bmp"
IDB_ITEM_CHOKINGGAS     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\chokinggas.bmp"
IDB_ITEM_EXPLODING      BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\exploding.bmp"
IDB_ITEM_STRANGLINGGAS  BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\stranglinggas.bmp"
```

```
IDB_ITEM_FULM           BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\fulm.bmp"
IDB_ITEM_DECOYGIDBINN   BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\decoygidbinn.bmp"
IDB_ITEM_THEGIDBINN     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\thegidbinn.bmp"
IDB_ITEM_WIRTSLEG       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\wirtsleg.bmp"
IDB_ITEM_HORADRICMALUS  BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\horadricmalus.bmp"
IDB_ITEM_HELLFORGEHAMMER BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\hellforgehammer.bmp"
IDB_ITEM_HORADRICSTAFF  BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\horadricstaff.bmp"
IDB_ITEM_SHAFTOFHORADRICSTAFF BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\shaftofhoradricstaff.bmp"
IDB_ITEM_HANDAXE        BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\handaxe.bmp"
IDB_ITEM_AXE            BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\axe.bmp"
IDB_ITEM_DOUBLEAXE      BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\doubleaxe.bmp"
IDB_ITEM_MILITARYPICK   BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\militarypick.bmp"
IDB_ITEM_WARAXE         BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\waraxe.bmp"
IDB_ITEM_LARGEAXE       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\largeaxe.bmp"
IDB_ITEM_BROADAXE       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\broadaxe.bmp"
IDB_ITEM_BATTLEAXE      BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\battleaxe.bmp"
IDB_ITEM_GREATAXE       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\greataxe.bmp"
IDB_ITEM_GIANTAXE       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\giantaxe.bmp"
IDB_ITEM_WAND           BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\wand.bmp"
IDB_ITEM_YEWWAND        BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\yewwand.bmp"
IDB_ITEM_BONEWAND       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\bonewand.bmp"
IDB_ITEM_GRIMWAND       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\grimwand.bmp"
IDB_ITEM_CLUB           BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\club.bmp"
IDB_ITEM_SCEPTER        BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\scepter.bmp"
IDB_ITEM_GRANDSCEPTER   BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\grandscepter.bmp"
IDB_ITEM_WARSCEPTER     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\warscepter.bmp"
IDB_ITEM_SPIKEDCLUB     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\spikedclub.bmp"
IDB_ITEM_MACE           BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\mace.bmp"
IDB_ITEM_MORNINGSTAR    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\morningstar.bmp"
IDB_ITEM_FLAIL          BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\flail.bmp"
IDB_ITEM_WARHAMMER      BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\warhammer.bmp"
IDB_ITEM_MAUL           BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\maul.bmp"
IDB_ITEM_GREATMAUL      BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\greatmaul.bmp"
IDB_ITEM_SHORTSWORD     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\shortsword.bmp"
IDB_ITEM_SCIMITAR       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\scimitar.bmp"
IDB_ITEM_SABRE          BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\sabre.bmp"
IDB_ITEM_FALCHION       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\falchion.bmp"
IDB_ITEM_CRYSTALSWORD   BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\crystalsword.bmp"
IDB_ITEM_BROADSWORD     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\broadsword.bmp"
IDB_ITEM_LONGSWORD      BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\longsword.bmp"
IDB_ITEM_WARSWORD       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\warsword.bmp"
IDB_ITEM_2HSWORD        BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\2hsword.bmp"
IDB_ITEM_CLAYMORE       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\claymore.bmp"
IDB_ITEM_GIANTSWORD     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\giantsword.bmp"
IDB_ITEM_BASTARDSWORD   BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\bastardsword.bmp"
IDB_ITEM_FLAMBERGE      BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\flamberge.bmp"
IDB_ITEM_GREATSWORD     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\greatsword.bmp"
IDB_ITEM_DAGGER         BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\dagger.bmp"
IDB_ITEM_DIRK           BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\dirk.bmp"
IDB_ITEM_KRIS           BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\kris.bmp"
IDB_ITEM_BLADE          BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\blade.bmp"
IDB_ITEM_THROWINGKNIFE  BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\throwingknife.bmp"
IDB_ITEM_THROWINGAXE    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\throwingaxe.bmp"
IDB_ITEM_BALANCEDKNIFE  BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\balancedknife.bmp"
IDB_ITEM_BALANCEDAXE    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\balancedaxe.bmp"
IDB_ITEM_JAVELIN        BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\javelin.bmp"
IDB_ITEM_PILUM          BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\pilum.bmp"
IDB_ITEM_SHORTSPEAR     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\shortspear.bmp"
IDB_ITEM_GLAIVE         BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\glaive.bmp"
IDB_ITEM_THROWINGSPEAR  BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\throwingspear.bmp"
IDB_ITEM_SPEAR          BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\spear.bmp"
IDB_ITEM_TRIDENT        BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\trident.bmp"
IDB_ITEM_BRANDISTOCK    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\brandistock.bmp"
IDB_ITEM_SPETUM         BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\spetum.bmp"
IDB_ITEM_PIKE           BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\pike.bmp"
IDB_ITEM_BARDICHE       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\bardiche.bmp"
IDB_ITEM_VOULGE         BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\voulge.bmp"
IDB_ITEM_SCYTHE         BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\scythe.bmp"
IDB_ITEM_POLEAXE        BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\poleaxe.bmp"
IDB_ITEM_HALBERD        BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\halberd.bmp"
IDB_ITEM_WARSCYTHE      BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\warscythe.bmp"
IDB_ITEM_SHORTSTAFF     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\shortstaff.bmp"
IDB_ITEM_LONGSTAFF      BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\longstaff.bmp"
IDB_ITEM_GNARLEDSTAFF   BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\gnarledstaff.bmp"
IDB_ITEM_BATTLESTAFF    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\battlestaff.bmp"
IDB_ITEM_WARSTAFF       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\warstaff.bmp"
IDB_ITEM_SHORTBOW       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\shortbow.bmp"
IDB_ITEM_HUNTERSBOW     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\huntersbow.bmp"
IDB_ITEM_LONGBOW        BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\longbow.bmp"
```

```
IDB_ITEM_COMPOSITEBOW   BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\compositebow.bmp"
IDB_ITEM_SHORTBATTLEBOW BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\shortbattlebow.bmp"
IDB_ITEM_LONGBATTLEBOW  BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\longbattlebow.bmp"
IDB_ITEM_SHORTWARBOW    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\shortwarbow.bmp"
IDB_ITEM_LONGWARBOW     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\longwarbow.bmp"
IDB_ITEM_LIGHTCROSSBOW  BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\lightcrossbow.bmp"
IDB_ITEM_CROSSBOW       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\crossbow.bmp"
IDB_ITEM_HEAVYCROSSBOW  BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\heavycrossbow.bmp"
IDB_ITEM_REPEATINGCROSSBOW BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\repeatingcrossbow.bmp"
IDB_ITEM_KHALIMSFLAIL   BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\khalimsflail.bmp"
IDB_ITEM_KHALIMSWILL    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\khalimswill.bmp"
IDB_ITEM_INFERNALCRANIUM BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\infernalcranium.bmp"
IDB_ITEM_WARBONET       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\warbonet.bmp"
IDB_ITEM_ARCANNASHEAD   BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\arcannashead.bmp"
IDB_ITEM_TARNHELM       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\tarnhelm.bmp"
IDB_ITEM_BERSERKERSHEADGEAR BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\berserkersheadgear.bmp"
IDB_ITEM_COIFOFGLORY    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\coifofglory.bmp"
IDB_ITEM_ISENHARTSHORNS BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\isenhartshorns.bmp"
IDB_ITEM_DUSKDEEP       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\duskdeep.bmp"
IDB_ITEM_SIGONSVISOR    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\sigonsvisor.bmp"
IDB_ITEM_HOWLTUSK       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\howltusk.bmp"
IDB_ITEM_IRATHASCOIL    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\irathascoil.bmp"
IDB_ITEM_MILABREGASDIADEM BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\milabregasdiadem.bmp"
IDB_ITEM_UNDEADCROWN    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\undeadcrown.bmp"
IDB_ITEM_ARCTICFURS     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\arcticfurs.bmp"
IDB_ITEM_GREYFORM       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\greyform.bmp"
IDB_ITEM_VIDALASAMBUSH  BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\vidalasambush.bmp"
IDB_ITEM_BLINKBATSFORM  BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\blinkbatsform.bmp"
IDB_ITEM_THECENTURION   BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\thecenturion.bmp"
IDB_ITEM_TWITCHTHROE    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\twitchthroe.bmp"
IDB_ITEM_DARKGLOW       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\darkglow.bmp"
IDB_ITEM_HAWKMAIL       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\hawkmail.bmp"
IDB_ITEM_CATHANSMESH    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\cathansmesh.bmp"
IDB_ITEM_SPARKLINGMAIL  BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\sparklingmail.bmp"
IDB_ITEM_ISENHARTSCASE  BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\isenhartscase.bmp"
IDB_ITEM_VENOMSWARD     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\venomsward.bmp"
IDB_ITEM_BERSERKERSHAUBERK BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\berserkershauberk.bmp"
IDB_ITEM_ICEBLINK       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\iceblink.bmp"
IDB_ITEM_BONEFLESH      BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\boneflesh.bmp"
IDB_ITEM_ROCKFLEECE     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\rockfleece.bmp"
IDB_ITEM_SIGONSSHELTER  BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\sigonsshelter.bmp"
IDB_ITEM_RATTLECAGE     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\rattlecage.bmp"
IDB_ITEM_TANCREDSSPINE  BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\tancredsspine.bmp"
IDB_ITEM_GOLDSKIN       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\goldskin.bmp"
IDB_ITEM_MILABREGASROBE BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\milabregasrobe.bmp"
IDB_ITEM_VICTORSSILK    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\victorssilk.bmp"
IDB_ITEM_ARCANNASFLESH  BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\arcannasflesh.bmp"
IDB_ITEM_HEAVENLYGARB   BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\heavenlygarb.bmp"
IDB_ITEM_HSARUSIRONFIST BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\hsarusironfist.bmp"
IDB_ITEM_PELTALUNATA    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\peltalunata.bmp"
IDB_ITEM_CLEGLAWSCLAW   BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\cleglawsclaw.bmp"
IDB_ITEM_UMBRALDISK     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\umbraldisk.bmp"
IDB_ITEM_STORMGUILD     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\stormguild.bmp"
IDB_ITEM_MILABREGASORB  BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\milabregasorb.bmp"
IDB_ITEM_STEELCLASH     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\steelclash.bmp"
IDB_ITEM_SIGONSGUARD    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\sigonsguard.bmp"
IDB_ITEM_BVERRITKEEP    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\bverritkeep.bmp"
IDB_ITEM_ISENHARTSPARRY BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\isenhartsparry.bmp"
IDB_ITEM_THEWARD        BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\theward.bmp"
IDB_ITEM_DEATHSHAND     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\deathshand.bmp"
IDB_ITEM_THEHANDOFBROC  BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\thehandofbroc.bmp"
IDB_ITEM_BLOODFIST      BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\bloodfist.bmp"
IDB_ITEM_CLEGLAWSPINCERS BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\cleglawspincers.bmp"
IDB_ITEM_CHANCEGUARDS   BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\chanceguards.bmp"
IDB_ITEM_IRATHASCUFF    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\irathascuff.bmp"
IDB_ITEM_ARCTICMITTS    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\arcticmitts.bmp"
IDB_ITEM_MAGEFIST       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\magefist.bmp"
IDB_ITEM_SIGONSGAUNTLETS BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\sigonsgauntlets.bmp"
IDB_ITEM_FROSTBURN      BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\frostburn.bmp"
IDB_ITEM_TANCREDSHOBNAILS BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\tancredshobnails.bmp"
IDB_ITEM_HOTSPUR        BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\hotspur.bmp"
IDB_ITEM_GOREFOOT       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\gorefoot.bmp"
IDB_ITEM_HSARUSIRONHEEL BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\hsarusironheel.bmp"
IDB_ITEM_TREADSOFCTHON  BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\treadsofcthon.bmp"
IDB_ITEM_VIDALASFETLOCK BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\vidalasfetlock.bmp"
IDB_ITEM_GOBLINTOE      BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\goblintoe.bmp"
IDB_ITEM_SIGONSGREAVES  BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\sigonsgreaves.bmp"
IDB_ITEM_TEARHAUNCH     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\tearhaunch.bmp"
IDB_ITEM_DEATHSGUARD    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\deathsguard.bmp"
```

```
IDB_ITEM_LENYMSCORD     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\lenymscord.bmp"
IDB_ITEM_ARCTICBINDING  BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\arcticbinding.bmp"
IDB_ITEM_SNAKECORD      BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\snakecord.bmp"
IDB_ITEM_HSARUSIRONSTAY BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\hsarusironstay.bmp"
IDB_ITEM_NIGHTSMOKE     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\nightsmoke.bmp"
IDB_ITEM_IRATHASCORD    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\irathascord.bmp"
IDB_ITEM_INFERNALBUCKLE BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\infernalbuckle.bmp"
IDB_ITEM_GOLDWRAP       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\goldwrap.bmp"
IDB_ITEM_SIGONSWRAP     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\sigonswrap.bmp"
IDB_ITEM_BLADEBUCKLE    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\bladebuckle.bmp"
IDB_ITEM_TANCREDSSKULL  BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\tancredsskull.bmp"
IDB_ITEM_WORMSKULL      BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\wormskull.bmp"
IDB_ITEM_WALLOFTHEEYELESS BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\walloftheeyeless.bmp"
IDB_ITEM_SWORDBACKHOLD  BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\swordbackhold.bmp"
IDB_ITEM_CAP            BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\cap.bmp"
IDB_ITEM_SKULLCAP       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\skullcap.bmp"
IDB_ITEM_HELM           BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\helm.bmp"
IDB_ITEM_FULLHELM       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\fullhelm.bmp"
IDB_ITEM_GREATHELM      BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\greathelm.bmp"
IDB_ITEM_CROWN          BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\crown.bmp"
IDB_ITEM_MASK           BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\mask.bmp"
IDB_ITEM_QUILTED        BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\quilted.bmp"
IDB_ITEM_LEATHER        BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\leather.bmp"
IDB_ITEM_HARDLEATHER    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\hardleather.bmp"
IDB_ITEM_STUDDEDLEATHER BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\studdedleather.bmp"
IDB_ITEM_RINGMAIL       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\ringmail.bmp"
IDB_ITEM_SCALEMAIL      BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\scalemail.bmp"
IDB_ITEM_CHAINMAIL      BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\chainmail.bmp"
IDB_ITEM_BREASTPLATE    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\breastplate.bmp"
IDB_ITEM_SPLINTMAIL     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\splintmail.bmp"
IDB_ITEM_PLATEMAIL      BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\platemail.bmp"
IDB_ITEM_FIELDPLATE     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\fieldplate.bmp"
IDB_ITEM_GOTHICPLATE    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\gothicplate.bmp"
IDB_ITEM_FULLPLATE      BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\fullplate.bmp"
IDB_ITEM_ANCIENTARMOR   BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\ancientarmor.bmp"
IDB_ITEM_LIGHTPLATE     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\lightplate.bmp"
IDB_ITEM_BUCKLER        BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\buckler.bmp"
IDB_ITEM_SMALLSHIELD    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\smallshield.bmp"
IDB_ITEM_LARGESHIELD    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\largeshield.bmp"
IDB_ITEM_KITESHIELD     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\kiteshield.bmp"
IDB_ITEM_TOWERSHIELD    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\towershield.bmp"
IDB_ITEM_GOTHICSHIELD   BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\gothicshield.bmp"
IDB_ITEM_LEATHERGLOVES  BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\leathergloves.bmp"
IDB_ITEM_HEAVYGLOVES    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\heavygloves.bmp"
IDB_ITEM_CHAINGLOVES    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\chaingloves.bmp"
IDB_ITEM_LIGHTGAUNTLETS BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\lightgauntlets.bmp"
IDB_ITEM_GAUNTLETS      BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\gauntlets.bmp"
IDB_ITEM_LEATHERBOOTS   BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\leatherboots.bmp"
IDB_ITEM_HEAVYBOOTS     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\heavyboots.bmp"
IDB_ITEM_CHAINBOOTS     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\chainboots.bmp"
IDB_ITEM_LIGHTPLATEBOOTS BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\lightplateboots.bmp"
IDB_ITEM_PLATEBOOTS     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\plateboots.bmp"
IDB_ITEM_SASH           BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\sash.bmp"
IDB_ITEM_LIGHTBELT      BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\lightbelt.bmp"
IDB_ITEM_BELT           BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\belt.bmp"
IDB_ITEM_HEAVYBELT      BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\heavybelt.bmp"
IDB_ITEM_GIRDLE         BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\girdle.bmp"
IDB_ITEM_BONEHELM       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\bonehelm.bmp"
IDB_ITEM_BONESHIELD     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\boneshield.bmp"
IDB_ITEM_SPIKEDSHIELD   BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\spikedshield.bmp"
IDB_ITEM_ELIXEROFVITALITY BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\elixerofvitality.bmp"
IDB_ITEM_HPO            BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\hpo.bmp"
IDB_ITEM_MPO            BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\mpo.bmp"
IDB_ITEM_HPF            BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\hpf.bmp"
IDB_ITEM_MPF            BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\mpf.bmp"
IDB_ITEM_STAMINA        BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\stamina.bmp"
IDB_ITEM_ANTIDOTE       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\antidote.bmp"
IDB_ITEM_REJUV          BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\rejuv.bmp"
IDB_ITEM_FULLREJUV      BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\fullrejuv.bmp"
IDB_ITEM_THAWING        BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\thawing.bmp"
IDB_ITEM_TOMEBLUE       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\tomeblue.bmp"
IDB_ITEM_TOMERED        BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\tomered.bmp"
IDB_ITEM_TOPOFHORADRICSTAFF BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\topofhoradricstaff.bmp"
IDB_ITEM_GOLD           BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\gold.bmp"
IDB_ITEM_SCROLLOFINIFUSS BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\scrollofinifuss.bmp"
IDB_ITEM_ARROWS         BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\arrows.bmp"
IDB_ITEM_TORCH          BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\torch.bmp"
IDB_ITEM_BOLTS          BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\bolts.bmp"
IDB_ITEM_SCROLLBLUE     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\scrollblue.bmp"
```

```
IDB_ITEM_SCROLLRED      BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\scrollred.bmp"
IDB_ITEM_SKELETONJAW    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\skeletonjaw.bmp"
IDB_ITEM_SKELETONHRN    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\skeletonhrn.bmp"
IDB_ITEM_SKELETONTAL    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\skeletontal.bmp"
IDB_ITEM_SKELETONFLG    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\skeletonflg.bmp"
IDB_ITEM_SKELETONFNG    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\skeletonfng.bmp"
IDB_ITEM_SKELETONQLL    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\skeletonqll.bmp"
IDB_ITEM_SKELETONSCZ    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\skeletonscz.bmp"
IDB_ITEM_SKELETONSOL    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\skeletonsol.bmp"
IDB_ITEM_KEY            BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\key.bmp"
IDB_ITEM_BLACKTOWERKEY  BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\blacktowerkey.bmp"
IDB_ITEM_POTIONOFLIFE   BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\potionoflife.bmp"
IDB_ITEM_JADEFIGURINE   BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\jadefigurine.bmp"
IDB_ITEM_GOLDENBIRD     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\goldenbird.bmp"
IDB_ITEM_LAMESEUSTOME   BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\lameseustome.bmp"
IDB_ITEM_HORADRICCUBE   BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\horadriccube.bmp"
IDB_ITEM_HORADRICSCROLL BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\horadricscroll.bmp"
IDB_ITEM_MOPHISTOSSOULSTONE BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\mophistossoulstone.bmp"
IDB_ITEM_BOOKOFSKILL    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\bookofskill.bmp"
IDB_ITEM_EYE            BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\eye.bmp"
IDB_ITEM_HEART          BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\heart.bmp"
IDB_ITEM_BRAIN          BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\brain.bmp"
IDB_ITEM_EAR            BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\ear.bmp"
IDB_ITEM_CHIPPEDAMETHYST BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\chippedamethyst.bmp"
IDB_ITEM_FLAWEDAMETHYST BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\flawedamethyst.bmp"
IDB_ITEM_AMETHYST       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\amethyst.bmp"
IDB_ITEM_FLAWLESSAMETHYST BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\flawlessamethyst.bmp"
IDB_ITEM_PERFECTAMETHYST BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\perfectamethyst.bmp"
IDB_ITEM_CHIPPEDTOPAZ   BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\chippedtopaz.bmp"
IDB_ITEM_FLAWEDTOPAZ    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\flawedtopaz.bmp"
IDB_ITEM_TOPAZ          BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\topaz.bmp"
IDB_ITEM_FLAWLESSTOPAZ  BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\flawlesstopaz.bmp"
IDB_ITEM_PERFECTTOPAZ   BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\perfecttopaz.bmp"
IDB_ITEM_CHIPPEDSAPPHIRE BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\chippedsapphire.bmp"
IDB_ITEM_FLAWEDSAPPHIRE BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\flawedsapphire.bmp"
IDB_ITEM_SAPPHIRE       BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\sapphire.bmp"
IDB_ITEM_FLAWLESSSAPPHIRE BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\flawlesssapphire.bmp"
IDB_ITEM_PERFECTSAPPHIRE BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\perfectsapphire.bmp"
IDB_ITEM_CHIPPEDEMERALD BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\chippedemerald.bmp"
IDB_ITEM_FLAWEDEMERALD  BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\flawedemerald.bmp"
IDB_ITEM_EMERALD        BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\emerald.bmp"
IDB_ITEM_FLAWLESSEMERALD BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\flawlessemerald.bmp"
IDB_ITEM_PERFECTEMERALD BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\perfectemerald.bmp"
IDB_ITEM_CHIPPEDRUBY    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\chippedruby.bmp"
IDB_ITEM_FLAWEDRUBY     BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\flawedruby.bmp"
IDB_ITEM_RUBY           BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\ruby.bmp"
IDB_ITEM_FLAWLESSRUBY   BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\flawlessruby.bmp"
IDB_ITEM_PERFECTRUBY    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\perfectruby.bmp"
IDB_ITEM_CHIPPEDDIAMOND BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\chippeddiamond.bmp"
IDB_ITEM_FLAWEDDIAMOND  BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\flaweddiamond.bmp"
IDB_ITEM_DIAMOND        BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\diamond.bmp"
IDB_ITEM_FLAWLESSDIAMOND BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\flawlessdiamond.bmp"
IDB_ITEM_PERFECTDIAMOND BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\perfectdiamond.bmp"
IDB_ITEM_MINORHEALING   BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\minorhealing.bmp"
IDB_ITEM_LIGHTHEALING   BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\lighthealing.bmp"
IDB_ITEM_HEALING        BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\healing.bmp"
IDB_ITEM_GREATERHEALING BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\greaterhealing.bmp"
IDB_ITEM_SUPERHEALING   BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\superhealing.bmp"
IDB_ITEM_MINORMANA      BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\minormana.bmp"
IDB_ITEM_LIGHTMANA      BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\lightmana.bmp"
IDB_ITEM_MANA           BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\mana.bmp"
IDB_ITEM_GREATERMANA    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\greatermana.bmp"
IDB_ITEM_SUPERMANA      BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\supermana.bmp"
IDB_ITEM_CHIPPEDSKULL   BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\chippedskull.bmp"
IDB_ITEM_FLAWEDSKULL    BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\flawedskull.bmp"
IDB_ITEM_SKULL          BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\skull.bmp"
IDB_ITEM_FLAWLESSSKULL  BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\flawlessskull.bmp"
IDB_ITEM_PERFECTSKULL   BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\perfectskull.bmp"
IDB_NOTPLACEABLE        BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Other\\NotPlaceable.bmp"
IDB_ITEM_RING5          BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\ring5.bmp"
IDB_ITEM_AMULET2        BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\amu2.bmp"
IDB_ITEM_AMULET3        BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\amu3.bmp"
IDB_ITEM_RING1          BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\ring1.bmp"
IDB_ITEM_RING2          BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\ring2.bmp"
IDB_ITEM_RING3          BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\ring3.bmp"
IDB_ITEM_RING4          BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\ring4.bmp"
IDB_ITEM_AMULET1        BITMAP  DISCARDABLE     "Resources\\ItemBitmaps\\amu1.bmp"
IDB_JAMELLASMALL        BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Other\\JamellaSmall.bmp"
IDB_WEBLINK             BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Other\\WebLink.bmp"
```

```
IDB_PLUS16              BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Other\\Plus16.bmp"
IDB_GRID_2x4            BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Other\\grid_2x4.bmp"
IDB_GRID_2x4_Black      BITMAP  DISCARDABLE     "Resources\\Bitmaps\\Other\\grid_2x4_black.bmp"

/////////////////////////////////////////////////////////////////////////////
//
// Icon
//

// Icon with lowest ID value placed first to ensure application icon
// remains consistent on all systems.
IDI_ICON                ICON    DISCARDABLE     "Resources\\Icons\\Icon.ico"
IDI_ITEMICON            ICON    DISCARDABLE     "Resources\\Icons\\ItemIcon2.ico"
IDI_CHELP               ICON    DISCARDABLE     "Resources\\Icons\\Chelp.ico"
IDI_D2SICON             ICON    DISCARDABLE     "resources\\icons\\d2sicon.ico"


/////////////////////////////////////////////////////////////////////////////
//
// Accelerator
//

IDR_MAIN ACCELERATORS DISCARDABLE
BEGIN
    "J",            IDR_EOPTIONS,           VIRTKEY, CONTROL, ALT, NOINVERT
    "N",            IDR_NEW,                VIRTKEY, CONTROL, NOINVERT
    "O",            IDR_OPEN,               VIRTKEY, CONTROL, NOINVERT
    "R",            IDR_RELOAD,             VIRTKEY, CONTROL, NOINVERT
    "S",            IDR_SAVE,               VIRTKEY, CONTROL, NOINVERT
END


/////////////////////////////////////////////////////////////////////////////
//
// Cursor
//

IDC_CUR_MOVE            CURSOR  DISCARDABLE     "Resources\\Cursors\\Move.cur"
IDC_CUR_CROSS           CURSOR  DISCARDABLE     "Resources\\Cursors\\cross.cur"
IDC_CUR_ADD             CURSOR  DISCARDABLE     "Resources\\Cursors\\cur_add.cur"
IDC_CUR_MOVECOPY        CURSOR  DISCARDABLE     "Resources\\Cursors\\CurMoveCopy.cur"
IDC_CUR_NO              CURSOR  DISCARDABLE     "resources\\cursors\\cur_no.cur"

/////////////////////////////////////////////////////////////////////////////
//
// WAVE
//

IDR_WAVE_JamellaSound   WAVE    DISCARDABLE     "Resources\\Waves\\JamellaSound.wav"

/////////////////////////////////////////////////////////////////////////////
//
// D2S
//

IDN_Amazon              D2S     DISCARDABLE     "Resources\\Chars\\Novices\\Amazon.d2s"
IDN_Barbarian           D2S     DISCARDABLE     "Resources\\Chars\\Novices\\Barbarian.d2s"
IDN_Necromancer         D2S     DISCARDABLE     "Resources\\Chars\\Novices\\Necromancer.d2s"
IDN_Paladin             D2S     DISCARDABLE     "Resources\\Chars\\Novices\\Paladin.d2s"
IDN_Sorceress           D2S     DISCARDABLE     "Resources\\Chars\\Novices\\Sorceress.d2s"

/////////////////////////////////////////////////////////////////////////////
//
// CHELP
//

IDH_TAB2Magic           CHELP   DISCARDABLE     "Resources\\CHelp\\TAB2Magic.rtf"
IDH_TAB2Rare            CHELP   DISCARDABLE     "Resources\\CHelp\\TAB2Rare.rtf"
IDH_TAB2S               CHELP   DISCARDABLE     "Resources\\CHelp\\SearchBox.rtf"

/////////////////////////////////////////////////////////////////////////////
//
// String Table
//

STRINGTABLE DISCARDABLE
BEGIN
    IDS_SKILL_A01           "Creates an arrow composed entirely from Mana. Damage starts at the same level as a normal a
rrow and increases with higher levels of training."
```

```
    IDS_SKILL_A02           "Enchants an arrow with the additional damage of fire."
    IDS_SKILL_A03           "Illuminates monsters and decreases their ability to defend themselves."
    IDS_SKILL_A04           "Grants a chance to do double physical damage with ranged and thrust attacks."
END

STRINGTABLE DISCARDABLE
BEGIN
    IDS_SKILL_A05           "Multiple attacks within the time span of a normal attack, each jab a bit less powerful than
 the last."
    IDS_SKILL_A06           "Enchants an arrow, adding cold damage and slowing your enemy."
    IDS_SKILL_A07           "Enchants an arrow, adding cold damage and slowing your enemy."
    IDS_SKILL_A08           "Grants a chance to move out of the way of a melee attack while standing still."
    IDS_SKILL_A09           "Adds lightning damage and increases normal damage to thrusting attacks."
    IDS_SKILL_A10           "Thrown javelin causes poison damage and leaves a trail of poison clouds."
    IDS_SKILL_A11           "Adds fire damage to normal arrows and explodes on impact."
    IDS_SKILL_A12           "Slows the missiles of nearby enemies."
    IDS_SKILL_A13           "Grants a chance to move out of the way of a missile attack while standing still."
    IDS_SKILL_A14           "A more powerful attack with an increased chance the weapon will lose durability."
    IDS_SKILL_A15           "Leaves a trail of lightning and does lightning damage."
    IDS_SKILL_A16           "Arrows have additional cold damage and momentarily freeze the target."
    IDS_SKILL_A17           "Imbues an arrow with the ability to seek its nearest target."
    IDS_SKILL_A18           "Additional chance to hit for ranged attacks."
    IDS_SKILL_A19           "A lightning attack that releases charged bolts."
    IDS_SKILL_A20           "Similar to Poison Javelin with an additional cloud of expanding poison at the point of impa
ct."
END

STRINGTABLE DISCARDABLE
BEGIN
    IDS_SKILL_A21           "Enchants an arrow that does fire damage, and explodes into a patch of fire on the ground. C
reatures passing through the flames take additional damage."
    IDS_SKILL_A22           "Fires a volley of arrows at multiple nearby targets."
    IDS_SKILL_A23           "Creates a duplicate image to distract enemies."
    IDS_SKILL_A24           "Grants a chance to escape any attack while moving."
    IDS_SKILL_A25           "Rapidly strikes several close targets."
    IDS_SKILL_A26           "Enchants an arrow to deliver cold damage that freezes any monsters near the point of impact
."
    IDS_SKILL_A27           "Summons a powerful Valkyrie warrior to fight by your side."
    IDS_SKILL_A28           "A chance that your missile will continue through its victim."
    IDS_SKILL_A29           "Does lightning damage and releases chain lightning from target."
    IDS_SKILL_A30           "Creates a powerful lightning bolt that releases multiple lightning bolts from target."
END

STRINGTABLE DISCARDABLE
BEGIN
    IDS_SKILL_S01           "Creates a bolt of fire."
    IDS_SKILL_S02           "Increases your Mana recovery rate."
    IDS_SKILL_S03           "Fires multiple, jumping bolts of electricity that seek their targets."
    IDS_SKILL_S04           "Shoots a bolt of ice that damages and slows your victim."
    IDS_SKILL_S05           "Gives a defense bonus and freezes any melee attacker that hits you."
    IDS_SKILL_S06           "A spout of flame that burn your enemies."
    IDS_SKILL_S07           "Every enemy in a radius around you lose 25% of their current health."
    IDS_SKILL_S08           "Allows you to pick up items, trigger objects, and attack others at a distance."
    IDS_SKILL_S09           "Creates an expanding ring of ice and frost that damages and slows enemies."
    IDS_SKILL_S10           "Creates a bolt of ice that completely freezes a target."
    IDS_SKILL_S11           "Leave a wall of fire in your footsteps."
    IDS_SKILL_S12           "Creates a ball of fire that explodes on impact."
    IDS_SKILL_S13           "Creates an expanding ring of electricity that does massive damage."
    IDS_SKILL_S14           "Casts a bolt of lightning."
    IDS_SKILL_S15           "Defense bonus, plus any melee attacker that hits you takes damage and is slowed."
    IDS_SKILL_S16           "Creates a wall of fire."
END

STRINGTABLE DISCARDABLE
BEGIN
    IDS_SKILL_S17           "Temporarily adds Fire damage to a weapon."
    IDS_SKILL_S18           "Casts a lightning bolt that jumps through multiple targets."
    IDS_SKILL_S19           "Instantly transports you between two points."
    IDS_SKILL_S20           "A shard of ice that inflicts massive cold damage and explodes to freeze nearby enemies."
    IDS_SKILL_S21           "Draws down a meteor from the heavens to smash your enemies."
    IDS_SKILL_S22           "Summons a thunderstorm that periodically blasts a nearby enemy with a bolt of lightning."
    IDS_SKILL_S23           "Absorbs magical and some physical damage to Mana instead of Life."
    IDS_SKILL_S24           "Summons an ice storm to rain cold death onto your enemies."
    IDS_SKILL_S25           "Confers a defense bonus and launches an ice bolt against ranged attackers."
    IDS_SKILL_S26           "Increases the damage done by your fire spells."
    IDS_SKILL_S27           "Creates a multi-headed beast that attacks your enemies with bolts of fire."
    IDS_SKILL_S28           "Reduces the Mana cost of lightning spells."
    IDS_SKILL_S29           "A pulsating orb that shreds an area with ice bolts."
    IDS_SKILL_S30           "Pierces the cold resistance of your enemies."
```

```
END

STRINGTABLE DISCARDABLE
BEGIN
    IDS_SKILL_N01           "Increases the amount of damage received."
    IDS_SKILL_N02           "Summons multiple projectiles that damage enemies."
    IDS_SKILL_N03           "A protective shield that absorbs damage."
    IDS_SKILL_N05           "Raises one skeleton per skill level to fight for you."
    IDS_SKILL_N04           "Improves the quality of your raised skeletons, magi, and revived."
    IDS_SKILL_N06           "Decreases radius of awareness."
    IDS_SKILL_N07           "Decreases the damage the target can do."
    IDS_SKILL_N08           "Adds poison damage to a dagger."
    IDS_SKILL_N09           "The targeted corpse explodes, damaging all nearby enemies."
    IDS_SKILL_N10           "Raises a Golem from the earth to fight for you."
    IDS_SKILL_N11           "Damage dealt is damage received."
    IDS_SKILL_N12           "Cursed monsters run in fear."
END

STRINGTABLE DISCARDABLE
BEGIN
    IDS_SKILL_N13           "Creates a barrier of bone."
    IDS_SKILL_N14           "Enhances speed and life of Golems."
    IDS_SKILL_N15           "Raises a Skeletal Mage that fights for you with an elemental attack."
    IDS_SKILL_N16           "Cursed monsters attack randomly."
    IDS_SKILL_N17           "Attacking a cursed soul gives you health."
    IDS_SKILL_N18           "Releases a cloud of poisonous gas from a corpse."
    IDS_SKILL_N19           "Summons a magical missile of bone."
    IDS_SKILL_N20           "Summons a Golem that is linked to the caster's health."
    IDS_SKILL_N21           "Causes other monsters to target your enemy."
    IDS_SKILL_N22           "Slows speed of the cursed."
    IDS_SKILL_N23           "Summons a ring of bone to surround a target."
    IDS_SKILL_N24           "Raises elemental resistances of your Minions."
    IDS_SKILL_N25           "Summon a Golem from a metal item. The golem gains properties of the item."
    IDS_SKILL_N26           "Elemental attacks do more damage to the cursed monster."
    IDS_SKILL_N27           "A ring of poison explodes from the Necromancer."
    IDS_SKILL_N28           "Spirit tracks down a target, or finds one of its own."
END

STRINGTABLE DISCARDABLE
BEGIN
    IDS_SKILL_N29           "A Golem of fire that uses fire damage to heal itself."
    IDS_SKILL_N30           "Resurrects a monster to fight for you."
END

STRINGTABLE DISCARDABLE
BEGIN
    IDS_SKILL_P01           "Increased damage at the cost of health."
    IDS_SKILL_P02           "Bolt of energy that damages undead, or heals friendly units."
    IDS_SKILL_P03           "Increases damage dealt by party members."
    IDS_SKILL_P04           "Heals all party members."
    IDS_SKILL_P05           "Increases the Fire resistance of all party members."
    IDS_SKILL_P06           "Shield bash that does damage and knock back."
    IDS_SKILL_P07           "Periodically does Fire damage to nearby enemies."
    IDS_SKILL_P08           "Enemies take damage when they cause melee damage to party members."
END

STRINGTABLE DISCARDABLE
BEGIN
    IDS_SKILL_P09           "Boosts the Defense of all party members."
    IDS_SKILL_P10           "Increases the Cold resistance of all party members"
    IDS_SKILL_P11           "Quickly attacks multiple adjacent enemies."
    IDS_SKILL_P12           "Closes the distance with an enemy, delivering a bash on contact."
    IDS_SKILL_P13           "Increases Attack Rating."
    IDS_SKILL_P14           "Reduces Poison duration for all party members."
    IDS_SKILL_P15           "Increases the Lighting resistance of all party members."
    IDS_SKILL_P16           "Adds Elemental (Fire, Lightning and Cold) damage to all melee attacks."
    IDS_SKILL_P17           "A magical Hammer spirals outward, damaging enemies. The Undead take additional damage."
    IDS_SKILL_P18           "Reduces the chance that your attacks will be interrupted."
    IDS_SKILL_P19           "Periodically does Cold damage to enemies nearby."
    IDS_SKILL_P20           "Increases Speed, Stamina, and Stamina recovery for all party members."
    IDS_SKILL_P21           "A successful attack has a chance to convert the target to fight evil."
    IDS_SKILL_P22           "Magically enhances shield to give defense bonuses."
    IDS_SKILL_P23           "Periodically does Lightning damage to enemies within a radius."
    IDS_SKILL_P24           "Damages and does knockback to the Undead."
END

STRINGTABLE DISCARDABLE
BEGIN
    IDS_SKILL_P25           "Boosts Mana recovery for all party members."
```

```
    IDS_SKILL_P26           "Lightning attack from the sky that releases Holy Bolts."
    IDS_SKILL_P27           "Increases the Attack Rate and Attack Rating for all party members."
    IDS_SKILL_P28           "Reduces the Defense and Resistances of all enemies."
    IDS_SKILL_P29           "Periodically attempts to redeem corpses for Health and Mana."
    IDS_SKILL_P30           "Increases all Elemental resistances for all party members."
END

STRINGTABLE DISCARDABLE
BEGIN
    IDS_SKILL_B01           "A powerful smashing blow that knocks the target back."
    IDS_SKILL_B02           "Increased damage and Attack Rating when using swords."
    IDS_SKILL_B03           "Increased damage and Attack Rating when using axes."
    IDS_SKILL_B04           "Increased damage and Attack Rating when using maces."
END

STRINGTABLE DISCARDABLE
BEGIN
    IDS_SKILL_B05           "Frightens monsters into retreating."
    IDS_SKILL_B06           "Grants a chance that a Health, Mana or Rejuvenation potion can be derived from a corpse."
    IDS_SKILL_B07           "Barbarian jumps and knocks back enemies where he lands."
    IDS_SKILL_B08           "A quick double attack that can damage two nearby enemies."
    IDS_SKILL_B09           "Increased damage and Attack Rating when using pole arms."
    IDS_SKILL_B10           "Increased damage and Attack Rating when throwing a weapon."
    IDS_SKILL_B11           "Increased damage and Attack Rating when using spears."
    IDS_SKILL_B12           "Taunts a monster to fight you."
    IDS_SKILL_B13           "Increases the defense of friendly units."
    IDS_SKILL_B14           "A successful attack briefly stuns the enemy."
    IDS_SKILL_B15           "Throws two weapons."
    IDS_SKILL_B16           "Increases stamina, and stamina recovery rate."
    IDS_SKILL_B17           "Grants a chance that an item can be found on a corpse."
    IDS_SKILL_B18           "Leaps onto a target and attacks when landing."
    IDS_SKILL_B19           "An uninterruptible attack that also temporarily improves your defense."
    IDS_SKILL_B20           "Improves overall defense."
END

STRINGTABLE DISCARDABLE
BEGIN
    IDS_SKILL_B21           "Frightens nearby enemies and lowers their defense."
    IDS_SKILL_B22           "Every successful hit increases your velocity and attack speed."
    IDS_SKILL_B23           "Increases your walk and run speeds."
    IDS_SKILL_B24           "Increases Mana, Life, and Stamina of party members. "
    IDS_SKILL_B25           "Turns a corpse into a fetish that will frighten monsters away."
    IDS_SKILL_B26           "A fierce spinning attack."
    IDS_SKILL_B27           "A powerful attack that leaves the Barbarian more vulnerable."
    IDS_SKILL_B28           "Increases resistances to elemental damage."
    IDS_SKILL_B29           "Damages and stuns your enemies."
    IDS_SKILL_B30           "Temporarily increases the skills of party members by 1."
END

STRINGTABLE DISCARDABLE
BEGIN
    IDS_QUEST11             "Den of Evil"
    IDS_QUEST12             "Sister's Burial Grounds"
    IDS_QUEST13             "Search for Cain / Cow Level"
    IDS_QUEST14             "The Forgotten Tower"
    IDS_QUEST15             "Tools of the Trade"
    IDS_QUEST16             "Sisters to the Slaughter"
    IDS_QUEST21             "Radament's Lair"
    IDS_QUEST22             "The Horadric Staff"
    IDS_QUEST23             "Tainted Sun"
    IDS_QUEST24             "Arcane Sanctuary"
    IDS_QUEST25             "The Summoner"
    IDS_QUEST26             "The Seven Tombs"
    IDS_QUEST31             "The Golden Bird"
    IDS_QUEST32             "Blade of the Old Religion"
    IDS_QUEST33             "Kahlim's Will"
END

STRINGTABLE DISCARDABLE
BEGIN
    IDS_QUEST34             "Lam Esen's Tomb"
    IDS_QUEST35             "The Blackend Temple"
    IDS_QUEST36             "The Guardian"
    IDS_QUEST41             "The Fallen Angel"
    IDS_QUEST42             "Hell Forge"
    IDS_QUEST43             "Terror's End"
END

#endif    // Neutral resources
```

```
//////////////////////////////////////////////////////////////////////////////


//////////////////////////////////////////////////////////////////////////////
// Deutsch (Deutschland) resources

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_DEU)
#ifdef _WIN32
LANGUAGE LANG_GERMAN, SUBLANG_GERMAN
#pragma code_page(1252)
#endif //_WIN32

#ifdef APSTUDIO_INVOKED
//////////////////////////////////////////////////////////////////////////////
//
// TEXTINCLUDE
//

1 TEXTINCLUDE DISCARDABLE
BEGIN
    "ResourceIDs.h\0"
END

2 TEXTINCLUDE DISCARDABLE
BEGIN
    "#include ""afxres.h""\r\n"
    "#include <dlgs.h>\r\n"
    "\0"
END

3 TEXTINCLUDE DISCARDABLE
BEGIN
    "\0"
END

#endif    // APSTUDIO_INVOKED


#ifndef _MAC
//////////////////////////////////////////////////////////////////////////////
//
// Version
//

VS_VERSION_INFO VERSIONINFO
 FILEVERSION 3,0,0,0
 PRODUCTVERSION 3,0,0,0
 FILEFLAGSMASK 0x3fL
#ifdef _DEBUG
 FILEFLAGS 0x1L
#else
 FILEFLAGS 0x0L
#endif
 FILEOS 0x40004L
 FILETYPE 0x1L
 FILESUBTYPE 0x0L
BEGIN
    BLOCK "StringFileInfo"
    BEGIN
        BLOCK "000004b0"
        BEGIN
            VALUE "Comments", "\0"
            VALUE "CompanyName", "A\0"
            VALUE "FileDescription", "Jamella's Diablo 2 Editor\0"
            VALUE "FileVersion", "3, 0, 0, 0\0"
            VALUE "InternalName", "JamellaD2Editor\0"
            VALUE "LegalCopyright", "Copyright © 2001 by Jamella\0"
            VALUE "LegalTrademarks", "\0"
            VALUE "OriginalFilename", "JamellaD2Editor.exe\0"
            VALUE "PrivateBuild", "\0"
            VALUE "ProductName", "Jamella's Diablo 2 Editor\0"
            VALUE "ProductVersion", "3, 0, 0, 0\0"
            VALUE "SpecialBuild", "\0"
        END
    END
    BLOCK "VarFileInfo"
    BEGIN
        VALUE "Translation", 0x0, 1200
    END
```

```
END

#endif    // !_MAC

#endif    // Deutsch (Deutschland) resources
/////////////////////////////////////////////////////////////////////////////
```