

# **The Beginner's Guide To Mod Making**

**for**

## **Diablo II Lord of Destruction**

**by**

**Phrozen Heart  
Mordini  
Fusman  
Myhruginoc**

### **Version History**

v1.4	9/8/2003	Updated to include v1.10 and v1.10s beta information. Expanded and included some explanations.
v1.3	6/22/2002	Complete rewrite to provide examples and detailed explanations. Incorporates several exercises which were previously stand-alone tutorials.
v1.2x beta	4/18/2002	Pretty much a total rewrite to hopefully help iron out any previous confusion that might have occurred. These versions were all closed betas only.
v1.1	12/8/2001	Added a link on the 'Step 1' page to the latest datafile for use with MPQ Viewer as some people have had problems extracting string files.
v1.0	10/4/2001	First public release.

### **Additional Credits**

Mike Meneldil  
Rich Grenfell  
Paul Siramy  
Drackin

Heynooffense  
Soul Slayer  
Alkalund  
ChaoticDeath

Peaceatlast20  
Rage AA  
Joel Falcou  
chippydip

# Getting Started

*The journey of a thousand miles  
begins with a single step*

Welcome to Diablo II, the way *you* want it! You have played the standard game enough, the Realms lag so much, and maybe you downloaded a modified version and wondered how they could change things so much. Now you want to try yourself, but don't know where to begin. This guide will show you where to find what you need, and how to make simple changes. There are often several ways to accomplish the same things; any method written in this document is but one way to get there.

Previous versions of the Guide were written for version 1.09. With the recent releases of the 1.10 betas, some files have changed extensively while others are relatively untouched. Some explanations herein will be general to both versions while others will be version specific. If you decide to mod 1.10 beta, keep in mind there are quite a few bugs. The exercises in this Guide are unlikely to be affected when the official 1.10 version is released, but more advanced techniques you learn later will certainly be affected.

It is possible to mod for earlier versions of the game, but that is outside of the scope of this Guide. We will concern ourselves with modding 1.09 and beta 1.10. A new version of the Guide will be issued after 1.10 has been officially released and Battle.net is updated.

## Source Files

Look in your Diablo II directory (usually C:\Program Files\Diablo II). You will find a lot of files, most of which we won't concern ourselves with. The ones of interest are Game.exe and the files with an .mpq extension. Game.exe actually runs the game. You will also find Diablo II.exe, but that is only a loader for Game.exe. The mpq files are archive files that contain thousands of data files within them, everything from item definitions to the movies between acts. These data files can be grouped by their extensions.

d2char.mpq	data files for Diablo II – character animations
d2data.mpq	data files for Diablo II – everything else
d2exp.mpq	data files, sound effects and animations for Expansion features
d2music.mpq	Blizzard logo videos and music files for Diablo II
d2speech.mpq	character and NPC speech for Diablo II
d2xtalk.mpq	character and NPC speech for Expansion
d2sfx.mpq	sound effects for Diablo II
d2video.mpq	movies for Diablo II
d2xmusic.mpq	music files for Expansion
d2xvideo.mpq	movies for Expansion
patch_D2.mpq	all changes introduced by any patch, not included in release versions

**Table 1 - Archive Files for Diablo II Lord of Destruction**

txt	1: database files for static game information 2: descriptive files which are not used in game
bin	compiled versions of the txt database files used when the game runs
cof	control files for animations
dc6	graphics for inventory, background panels and certain animations
dcc	character, monster, object and effects animations
ds1	map files used for area definition and static monster/object placement
dt1	floor, wall and roof tiles used by maps for surface appearances
tbl	1 : string tables for text seen in the game 2 : font kerning tables
wav	sound files for effects, speech and music
dat	1 : Diablo II palettes, used for converting dc6 & dcc 2 : Colormaps, to see the same monster in different colors

**Table 2 - Diablo II File Types – Inside the Archives**

### Tools of the Trade

Before you can even start, you will need to get your hands on the basic tools used to access and modify all of these files. All of the basic tools are available from the Phrozen Keep File Center. A package of commonly used tools and this Guide are packaged together as the Beginner's Toolkit.

- MPQ Viewer – for viewing and extracting files from the .mpq archives. There is a separate datafile for all known contents of the archives, without it MPQ Viewer won't know what to look for. The datafile gets changed with almost every game release.
- Tab Delimited Text Editor – for editing the database txt files. D2Excel is designed specifically for these files, and is available for download. Microsoft Excel or other spreadsheet programs will work, if you import and export as tab delimited text files with nothing for string delimiters. Also, MS Excel will corrupt the data table if the rightmost fields are empty. Although the files have .txt extensions, they are *not* ordinary text files, so do not use a word processor or even Notepad as they will corrupt the databases.
- Baron Darkstorm's Table Editor – for editing string table files. This is an easy tool to use, but it has trouble with PatchString.tbl file (more on this later). Other tbl editors are D2Tbl and Peer TBL Editor (v3.38).
- DC6Con and DC6Maker – for viewing and converting dc6 type graphics files.
- CV5 (v5.2) – for extracting, viewing and converting several types of graphics files. One component, CVDCC.DLL (now at v3.0), is required for dcc files but may be a separate download. Editing animation files is outside of the scope of this guide.
- MPQ2K and MPQ Stormless Editor – for packing your files into an mpq after you have changed them.

Another file that may need to be downloaded separately is patch\_D2.mpq. This file is not included on the CD-ROM releases for either Diablo II or Lord of Destruction. You get them normally by logging on to Battle.net and getting the latest patch pushed to your machine. You can also download patches and install them separately. Patch\_D2.mpq stores updated versions of any file that is located in *any* of the release mpq's. When you make your changes to a file from *any* mpq, you need to store it in patch\_d2.mpq, so you should not try to modify a CD-only installation of the game. Run at least one patch to update the game and obtain patch\_D2.mpq. Information in patch\_D2.mpq will always override earlier versions of the same data files stored in the release archive files.

1.07	6/19/2001	Release version of Lord of Destruction
1.08	6/27/2001	Maintenance and features upgrade
1.09	8/21/2001	Major change in game features
1.09b	10/5/2001	Maintenance update
1.09c	11/29/2001	Maintenance update
1.09d	12/5/2001	Maintenance update
<i>1.10 beta</i>	<i>7/3/2003</i>	<i>Major feature changes</i>
<i>1.10s beta</i>	<i>8/4/2003</i>	<i>Bug fixes and changes to 1.10 beta</i>

**Table 3 - Lord of Destruction Update History**

Many mod makers downgraded to version 1.09b because of a serious bug introduced with 1.09c and still present in 1.09d. Blizzard had provided certain skills on magical equipment that would be activated by combat events, these skills are known as “chance-cast skills.” But with version 1.09c and 1.09d, the chance-cast skills are displayed but no damage is caused by them. You can downgrade to version 1.09b by replacing three program files with earlier versions: they are D2Client.dll, D2Common.dll and D2Game.dll. All three dlls are available at the Phrozen Keep in various versions. Save your 1.09d dlls for use on Battle.net. Do not mix files between 1.09 (b or d) with 1.10, the changes between 1.09x and 1.10 are much greater and you will crash.

Two versions of the 1.10 beta have been released, and more versions are possible before the official release is made available. Since the version is not yet stable, if you choose to mod 1.10 then be sure your files are specific to the beta version. Changes to txt and tbl files brought from an earlier beta to a later beta (or the official release) must be tested again in the new update. The image formats are stable and not likely to change.

You can identify what version you are running by comparing the properties of your files to the table below. Right-click on the file in Windows Explorer to see these file sizes (Size, not Size on Disk) and dates (Modified, not Created or Accessed).

1.09b	D2Client.dll	1.08 MB (1,138,745 bytes)	8/16/2001
1.09b	D2Common.dll	676 KB (692,281 bytes)	8/16/2001
1.09b	D2Game.dll	0.98 MB (1,028,151 bytes)	8/16/2001
1.09d (Battle.net)	D2Client.dll	1.08 MB (1,134,644 bytes)	11/29/2001
1.09d (Battle.net)	D2Common.dll	676 KB (692,276 bytes)	11/29/2001
1.09d (Battle.net)	D2Game.dll	0.98 MB (1,028,146 bytes)	11/29/2001
1.10 beta	D2Client.dll	1.02 MB (1,073,204 bytes)	7/3/2003
1.10 beta	D2Common.dll	692 KB (708,660 bytes)	7/3/2003
1.10 beta	D2Game.dll	1.10 MB (1,159,218 bytes)	7/3/2003
1.10s beta	D2Client.dll	1.02 MB (1,073,204 bytes)	7/27/2003
1.10s beta	D2Common.dll	696 KB (712,756 bytes)	7/27/2003
1.10s beta	D2Game.dll	1.10 MB (1,163,314 bytes)	7/27/2003

**Table 4 – Distinguishing Between Versions**

Now you need a way to keep track of all of the files you will be generating within your Diablo II directory. One method is to make subdirectories within your Diablo II directory, with one mod or test setup in each subdirectory. You can make a shortcut to Game.exe with a mod directory as your starting directory. There is also a package of Visual Basic scripts available at the Phrozen Keep which will run your mods in their own subdirectory and automate changing DLLs for those mods that provide custom DLLs. You can use the DLL management feature to run mods with 1.09b DLLs while keeping unmodded Diablo II at the current version for Battle.net play.

The very first thing you should do after downloading the tools is make yourself a workspace separate from the game files. Make a copy of patch\_D2.mpq and save it in a safe directory, that way you always have a clean version to work from. Make another copy and move it to your workspace. Install the tools in the workspace so they always look for files there. This is especially important for MPQ2K, which uses a file with the same name as one of the game components (storm.dll); you must not get these versions of this file mixed up!

## Exercise #1 – Increasing Quantities

We will start with a very simple modification, changing the number of keys that can stack together. The standard limit is 12, which is a very low number for chest-studded dungeons like the Durance of Hate. We will make the upper limit 50 so we don't keep running back to town for more.

To begin, you will need to run MPQ Viewer. Start the program and click on the File menu. Note there are two Open options. The first time you run MPQ Viewer you need to load a datafile, which tells MPQView where to find the various internal files within the archive files. This datafile is separate from the viewer and changes between versions; be sure you download the latest datafile. To load a datafile, select the first Open option as shown below. MPQ Viewer will remember the last datafile used.

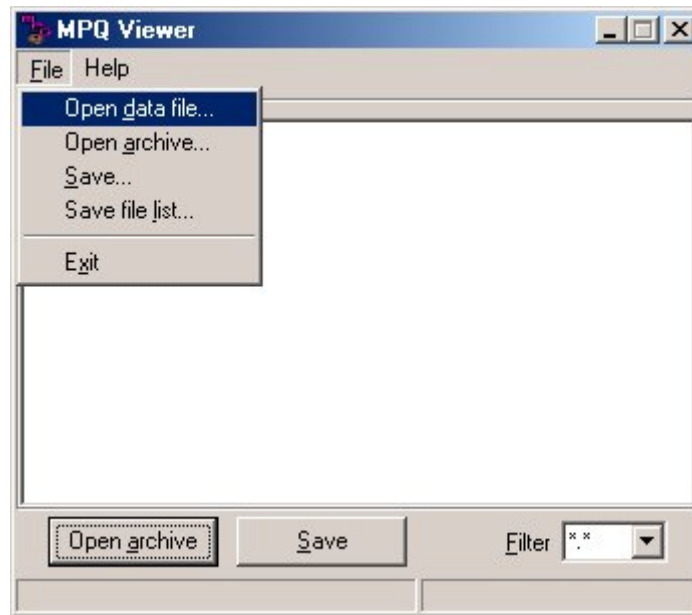


Figure 1 - MPQ Viewer Open Datafile

Next, click the Open Archive button, or select from the File menu. Select patch\_D2.mpq in the File Open dialog, and the entire contents of the mpq will appear in the main window.

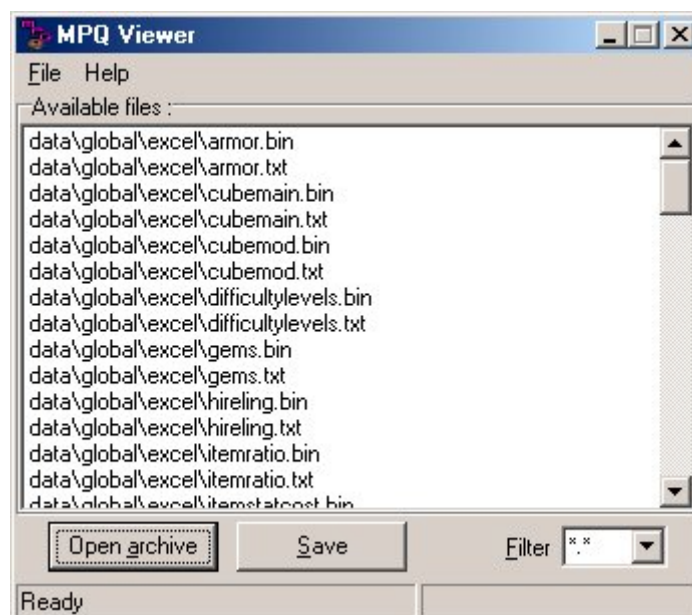
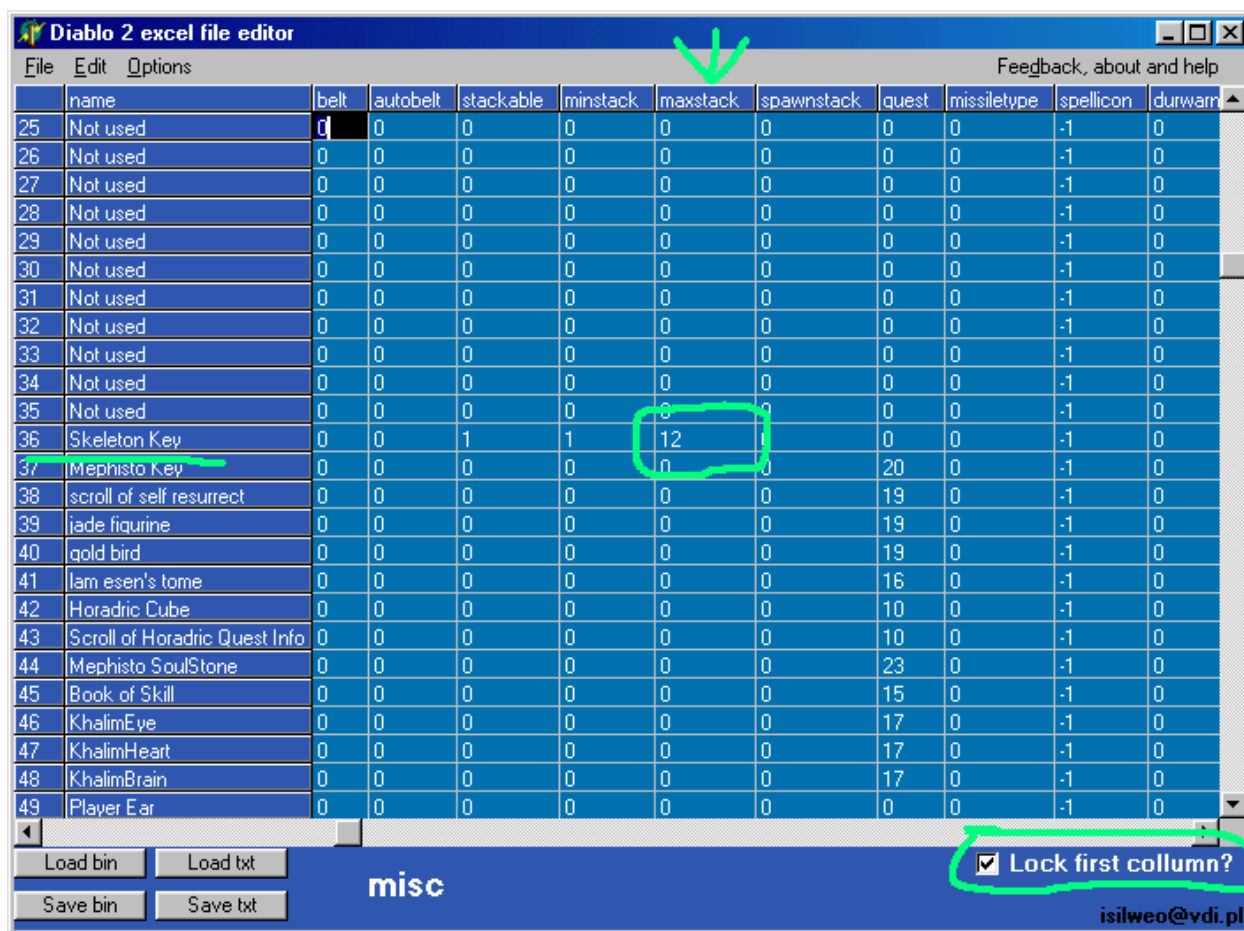


Figure 2 - MPQ Viewer Archive Listing

The paths you see in front of each filename are their position in the mpq archive, similar to how files and directories work in Windows. Table 2 shows there are two types of txt files, database files and descriptive files. We are interested in the database files only, and they are all conveniently grouped together in a single internal mpq directory, data\global\excel. Scroll down until you see the files listed in this mpq directory. The file we want is called Misc.txt, so highlight that file and click on the Save button. Save this file to your workspace. Shut down MPQ Viewer, we are done with it for this exercise.

Open up D2Excel next. Then open Misc.txt using the Load Txt button or file menu option. You will see something like this screen below. Here is where you see how these text files are actually databases. Each row is a record, and each column is a field. We want the record for keys so we can adjust the value of the maximum stack field.



Diablo 2 excel file editor

File Edit Options Feedback, about and help

	name	belt	autobelt	stackable	minstack	maxstack	spawnstack	quest	missiletype	spellicon	durwarn
25	Not used	0	0	0	0	0	0	0	0	-1	0
26	Not used	0	0	0	0	0	0	0	0	-1	0
27	Not used	0	0	0	0	0	0	0	0	-1	0
28	Not used	0	0	0	0	0	0	0	0	-1	0
29	Not used	0	0	0	0	0	0	0	0	-1	0
30	Not used	0	0	0	0	0	0	0	0	-1	0
31	Not used	0	0	0	0	0	0	0	0	-1	0
32	Not used	0	0	0	0	0	0	0	0	-1	0
33	Not used	0	0	0	0	0	0	0	0	-1	0
34	Not used	0	0	0	0	0	0	0	0	-1	0
35	Not used	0	0	0	0	0	0	0	0	-1	0
36	Skeleton Key	0	0	1	1	12	0	0	0	-1	0
37	Mephisto Key	0	0	0	0	0	0	20	0	-1	0
38	scroll of self resurrect	0	0	0	0	0	0	19	0	-1	0
39	jade figurine	0	0	0	0	0	0	19	0	-1	0
40	gold bird	0	0	0	0	0	0	19	0	-1	0
41	Iam esen's tome	0	0	0	0	0	0	16	0	-1	0
42	Horadric Cube	0	0	0	0	0	0	10	0	-1	0
43	Scroll of Horadric Quest Info	0	0	0	0	0	0	10	0	-1	0
44	Mephisto SoulStone	0	0	0	0	0	0	23	0	-1	0
45	Book of Skill	0	0	0	0	0	0	15	0	-1	0
46	KhalimEye	0	0	0	0	0	0	17	0	-1	0
47	KhalimHeart	0	0	0	0	0	0	17	0	-1	0
48	KhalimBrain	0	0	0	0	0	0	17	0	-1	0
49	Player Ear	0	0	0	0	0	0	0	0	-1	0

misc

Load bin Load txt Save bin Save txt

☒ Lock first column?

isilweo@vdi.pl

Figure 3 - Using D2Excel on Misc.txt

Now click on the Lock First Column box in the lower right corner. This will allow us to see the names of the records as we pan the field display. Scroll down until you see Skeleton Key, then scroll right until you see MaxStack in the header row. Where the row and column intersect is the value we want to change. You see it is already 12, that is what Blizzard assigned. Now click on that cell and change the number to 50. Save the file with the Save txt button. The Load and Save bin functions are not enabled.

How did we know Skeleton Key was what we wanted to change, when in game all we see is Key? That is one of the hard parts of txt files, the name shown here is not necessarily what you would see in play. The names here are string keys, which are used to *point to* the display name in the tbl files. We will go into greater detail during the fourth exercise.

You can see quickly how this change could be applied to *any* stackable item. Do you want more arrows in your quivers? Do you want 40-scroll spell books? Just edit the same field in their records. But watch your numbers! The game will not accept any value over 511.

## Making It Work

Now we have made a change, how do we get it into the game? You have made changes to a txt file, but the game doesn't read from them normally. Instead, it reads data from precompiled bin files, which contain the same information in a compressed, machine-ready format. We already discussed that you cannot save in bin format with D2Excel. We need to use a special game mode to create a new bin file from the one we just edited. This special mode is called `-direct -txt`, after the command line switches used to enable it. Every time this mode is used, the game will read from all txt files, whether external or in the mpq files, and create a complete set of new bin files. You will end up with around 70 bin files (1.09x) or 90 bin files (1.10).

In MPQ Viewer you saw how the txt and bin files are stored in the mpq in a `data\global\excel` directory. We have to make our own copy of that directory structure in our game directory. In Windows Explorer, go to your game directory, and create a new directory called **data**. Inside that directory make one called **global**. Inside the global directory make one more called **excel**. Now go back to your game directory and highlight `game.exe`. Copy the file, switch to your desktop, and right click to paste a shortcut on the desktop. Next, right click on the new shortcut and choose Properties at the bottom of the popup menu.

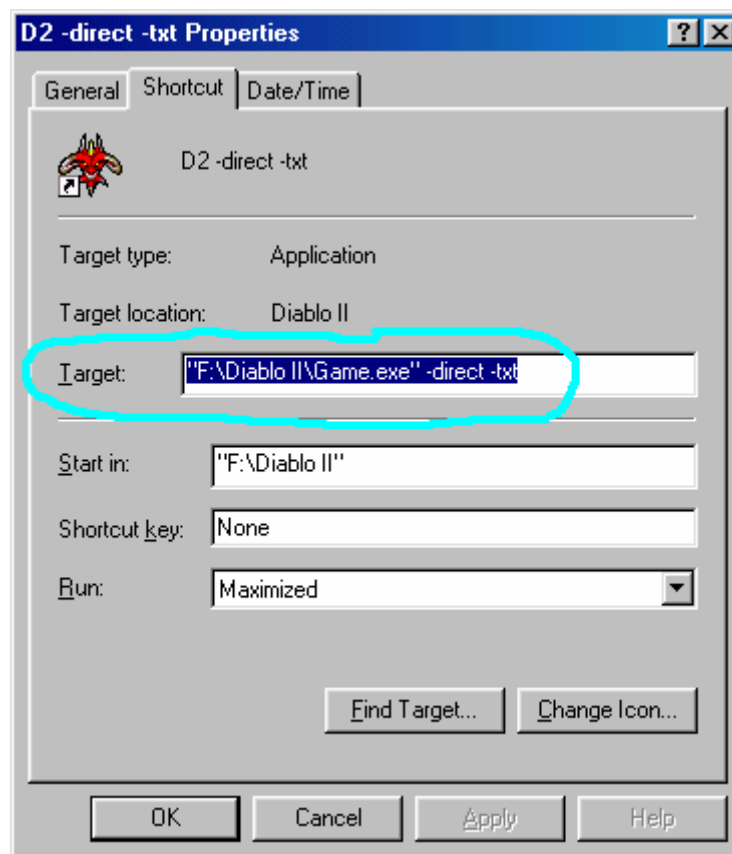


Figure 4 - Making a `-direct -txt` shortcut

Here is where you enter the information that makes Diablo II run in `-direct -txt` mode. In the Target field, make sure the full path name (use your system's path) is in quotes as above, if the path name contains any spaces. Then after the path name (outside the quotes if present) add `-direct -txt` *spaced as above*. Save the shortcut.

Now copy the newly-changed `Misc.txt` to the `data\global\excel` directory you made in your game directory. Start Diablo II with the new shortcut and send a test character into a game. You should always use test characters in any new mod so if you crash and somehow scramble your character file, you won't have lost something important. Quit this game, unless you want to go buy some keys and see if you get more than twelve.



If later you change txt files after generating bin files, it is a good idea to delete all of the bin files in data\global\excel before running Diablo II in -direct -txt mode. This ensures you will start with bin files that correspond with the latest txt files. Some information in certain files is used by other files when their bin files are generated.

At this point you will normally decide if your mod will use mpq files or run in -direct -txt mode. In the first mode, you pack your changes into patch\_D2.mpq, move patch\_D2.mpq into your game directory, and start the game as usual. Also, if you are planning to share your 1.09b or 1.09d mod with a Mac user, this is the only way that works on a Mac (and unfortunately this doesn't work for 1.10 beta). The second mode doesn't require packing into the mpq, but all players have to set up their system for it.

We will now take the changed txt file and the newly compiled bin file and insert them into the patch mpq. Go into the data\global\excel directory you made. Find misc.bin and copy it to your workspace.

### **MPQ2K is a DOS program**

Before we can run MPQ2K we have to make a couple of supporting files. MPQ2K doesn't run under Windows, so if you were to click on the file in Windows Explorer, you would see a DOS window flash on your screen and disappear. What you need to do is make a script file with MPQ2K commands and a batch file to run the program. Fire up Windows Notepad; both of these new files are ordinary text files.

We will start with the script file. The important commands in MPQ2K are Open an Archive, Add to an Archive, and Close an Archive. In Notepad, make a four-line file as below and save it as MPQLoad.txt.

```
O patch_D2.mpq
a misc.txt data\global\excel\misc.txt
a misc.bin data\global\excel\misc.bin
C patch_D2.mpq
```

Be sure you refer to directories using the backslash as shown. Using a forward slash is a very common typo, but in DOS programs the backslash represents a directory level.

The Open and Close archive commands must have capital letters. If you use a lower case 'o' in the first line, you will open the archive read-only, and you cannot add files. For the add commands, there is an upper case form and a lower case form. The only difference between the two is the lower case form also compresses the file it inserts, a useful way to minimize mpq size as you pack more and more files into it.

Now for the batch file. Type the single line in Notepad, and save this file as RunMPQ2K.bat. Be sure to replace "Text Document" with "All Files" in the Save As Type dropdown or your files will be named RunMPQ2K.bat.txt.

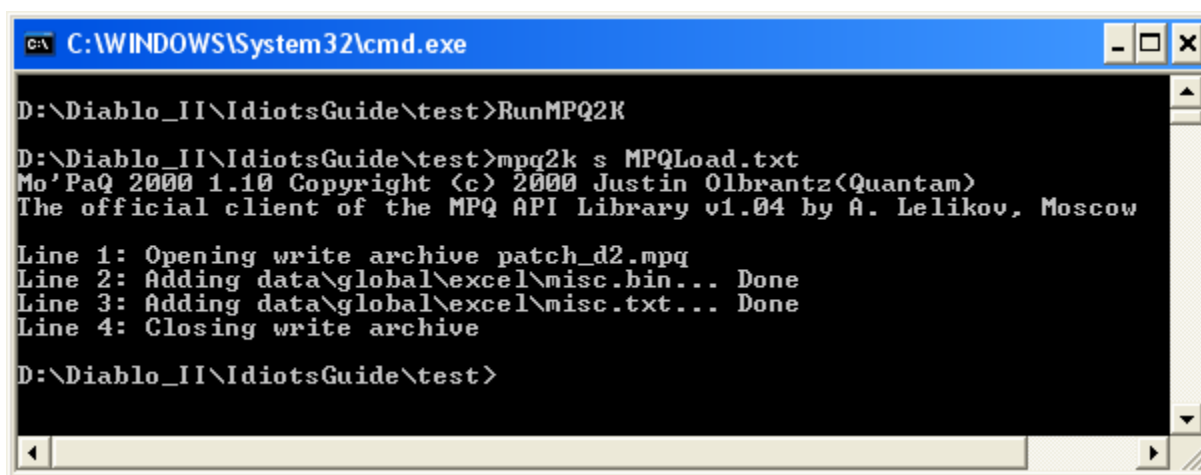
```
mpq2k s MPQLoad.txt
```

You are ready to pack the mpq at last. Make sure you have your working copy of patch\_D2.mpq, your new bin and txt files, your script and batch file, and MPQ2K's four files (mpq2k.exe, staredit.exe, storm.dll, lmpqapi.dll) in your workspace directory. Double-click on the batch file in Windows Explorer. You will still see a DOS window flash and disappear, but at least it is doing something.

You can see the results in a log file, if you change the batch file to this:

```
@echo off
mpq2k s MPQLoad.txt > MPQLoad_Log.txt
type MPQLoad_Log.txt | more
pause
```

This generates an additional file that can be viewed in Notepad, and allows you to view the results as they happen in a DOS window as shown on the next page.



```
C:\WINDOWS\System32\cmd.exe

D:\Diablo_II\IdiotsGuide\test>RunMPQ2K

D:\Diablo_II\IdiotsGuide\test>mpq2k s MPQLoad.txt
Mo'PaQ 2000 1.10 Copyright (c) 2000 Justin Olbrantz(Quantam)
The official client of the MPQ API Library v1.04 by A. Lelikov, Moscow

Line 1: Opening write archive patch_d2.mpq
Line 2: Adding data\global\excel\misc.bin... Done
Line 3: Adding data\global\excel\misc.txt... Done
Line 4: Closing write archive

D:\Diablo_II\IdiotsGuide\test>
```

**Figure 5 - MPQ2K in a DOS Window**

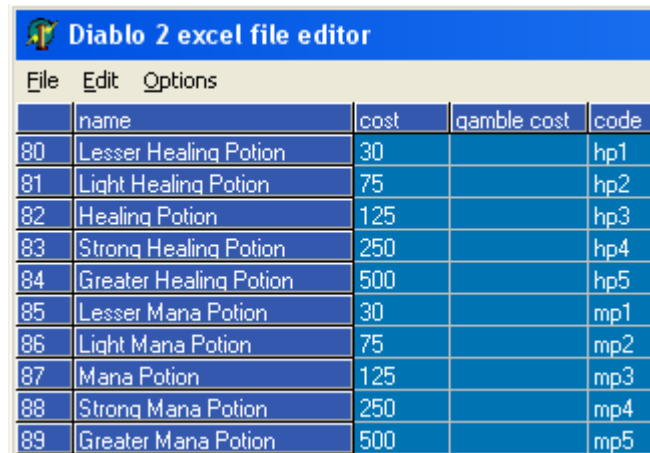
Check one more time to be sure you copied the original patch\_D2.mpq to a safe place. Now copy the modified patch\_D2.mpq to your game directory. Run the game normally, don't use the -direct -txt shortcut you made earlier. Play for a while and see that you can stack up more than twelve keys in one slot. Congratulations on making your first D2 mod!

## Exercise #2 – Stocking the Vendors

This exercise was written originally for versions 1.09b and 1.09d. In 1.10 mana potions are already buyable. But you can see how it is done in 1.10 by following the steps in this exercise. The same technique is applicable to other potions, gems, runes, miscellaneous items, and even weapons and armor. The vendor fields work the same way in all versions.

In the first exercise you learned how to extract files, edit txt databases, run in -direct -txt mode and pack the mpq with your changes. For this exercise we will return to Misc.txt and learn how to make vendors sell additional goods. The techniques are the same; we are just changing different records and fields.

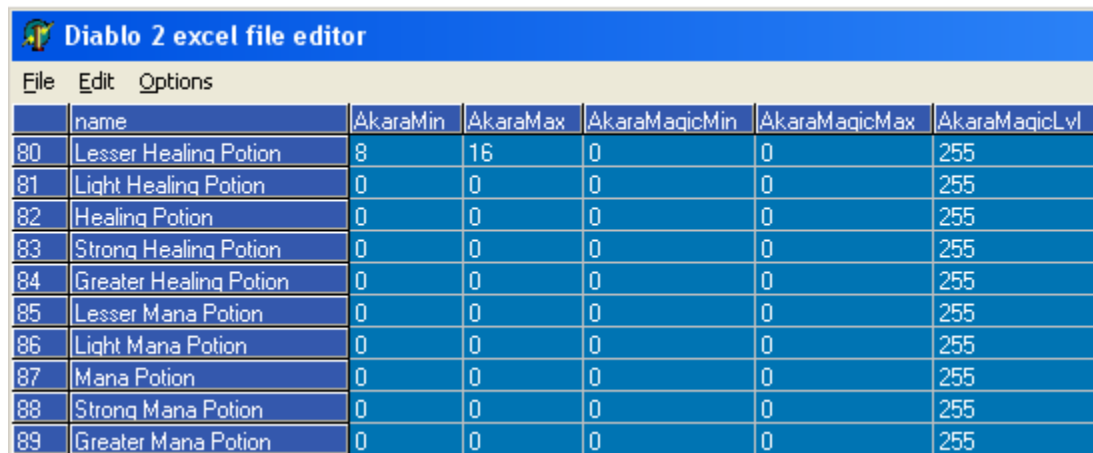
Open Misc.txt and find the lines for healing and mana potions. Scroll right until you see the Code field. You can see how the names in the Name column don't agree with the display names for these potions. For example, Strong Healing Potion here shows in game as Greater Healing Potion, while Greater Healing Potion here is the Super Healing Potion in game.



	name	cost	qamble cost	code
80	Lesser Healing Potion	30		hp1
81	Light Healing Potion	75		hp2
82	Healing Potion	125		hp3
83	Strong Healing Potion	250		hp4
84	Greater Healing Potion	500		hp5
85	Lesser Mana Potion	30		mp1
86	Light Mana Potion	75		mp2
87	Mana Potion	125		mp3
88	Strong Mana Potion	250		mp4
89	Greater Mana Potion	500		mp5

Figure 6 - Potions in Misc.txt

Now scroll right to the many fields where vendors are given the ability to sell any items. Each vendor has five fields. We will start with Akara, since she already sells healing potions. Getting her to sell mana potions also is as simple as using mana potion codes in the same way healing potion codes are used. In this case, we would enter 8 in AkaraMin and 16 in AkaraMax for the Lesser Mana Potion record. The other three fields refer to magical items, which is outside of the scope of this exercise.



	name	AkaraMin	AkaraMax	AkaraMagicMin	AkaraMagicMax	AkaraMagicLvl
80	Lesser Healing Potion	8	16	0	0	255
81	Light Healing Potion	0	0	0	0	255
82	Healing Potion	0	0	0	0	255
83	Strong Healing Potion	0	0	0	0	255
84	Greater Healing Potion	0	0	0	0	255
85	Lesser Mana Potion	0	0	0	0	255
86	Light Mana Potion	0	0	0	0	255
87	Mana Potion	0	0	0	0	255
88	Strong Mana Potion	0	0	0	0	255
89	Greater Mana Potion	0	0	0	0	255

Figure 7 - Potion Selling by Akara

You can go from act to act making the same changes for Lysander, Alkor, Jamella and Malah, giving each vendor the ability to sell mana potions just like they already sell healing potions.

The last part of this exercise is making vendors sell better potions in higher difficulties, and never running out of potions. To do this, we will scroll to the far right of the database.

NightmareUpgrade	HellUpgrade	mindam	maxdam	PermStoreItem
hp4	hp5	0	0	1
hp4	hp5	0	0	1
hp4	hp5	0	0	1
xxx	hp5	0	0	1
xxx	xxx	0	0	1
xxx	xxx	0	0	0
xxx	xxx	0	0	0
xxx	xxx	0	0	0
xxx	xxx	0	0	0
xxx	xxx	0	0	0

**Figure 8 - Better Potions, Always Potions**

Observe that the better grades of healing potions are listed in the upgrade fields, and use the same pattern in the mana records. Code “mp4” is Greater Mana Potion and code “mp5” is Super Mana Potion. Code “xxx” means not applicable, so we want to change the corresponding xxx values to mp4 and mp5. Last of all, change the PermStoreItem field to 1 for the mana potion records to make them always available. (A bug in 1.09x makes this work in Normal only, for higher difficulties you have to leave town and come back to replenish the vendor’s stock. In 1.10 use the Multibuy field to avoid this.)

As in the first exercise, save the edited file to your workspace directory. Copy the new Misc.txt to your data\global\excel directory, use your -direct -txt shortcut to start a game and generate bin files. Then copy misc.bin back to your workspace and pack patch\_D2.mpq. You can use the same MPQ2K script as last time, since these changes were in the same database file.

## Exercise #3 – Enchanted Paladin Shields

In this exercise we will look at the automatic modifiers some item types have. The automatic mods are usually bonuses to skills, such as sorceress skills on staves. Because sorceresses always have had skills available on staves, these automatic mods are often called Staff Mods. You can see this in the column header too. Except as detailed below, all steps are the same as in the first exercise. For this exercise we will need ItemTypes.txt.

	A	X	Y	Z	AA
1	ItemType	TreasureCl	Rarity	StaffMods	CostFormu
73	Auric Shields	0	1	pal	0
74	Primal Helm	0	1	bar	0
75	Pelt	0	1	dru	0

Figure 9 - Paladin Shields

This view shows Microsoft Excel using the Freeze Panes display mode, which you set by positioning the cursor at Cell B2 and then selecting menu option Window | Freeze Panes—this has the same effect as locking the first column in D2Excel. Scroll down until you see the Auric Shields record, then scroll right until you see the StaffMods field. Enter the class code for Paladin in the field as shown above. This will make all Auric Shields spawn with Paladin skills, similar to scepters, in addition to their inherent resistances. Save this file to your workspace, test and insert to mpq as usual.

This example shows applying Paladin skills to a Paladin shield (remember “Auric Shield” is not what gets displayed). In our case, you could have put other Class skills (like Sorc skills) than the Paladins ones, but only Paladins can use this item type so that would be useless. But you *can* apply any class to the StaffMods field for any item type. Remember *all* items spawned from an item type will get the bonus skills, so setting the “tors” item type with “dru” staff mods would result in every single body armor having druid skills, which is rather extreme.

This time we will use WinMPQ to pack our txt and bin files. Start the program, then go to the Options menu and select the File Lists tab. You can enter the same datafile here that you use for MPQView, using the Add List File command. The picture shows two lists are entered, as this setup is configured with one list for Diablo II and the other list for the original Diablo.

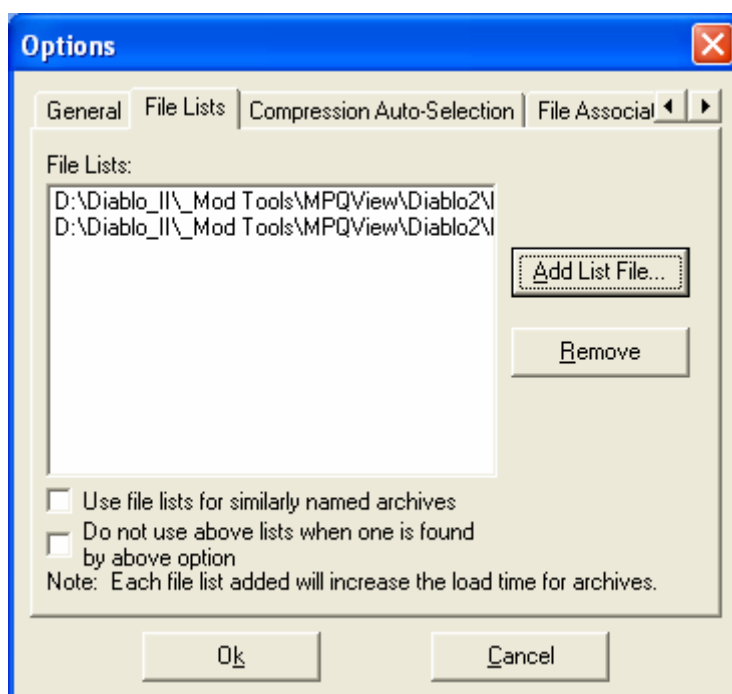
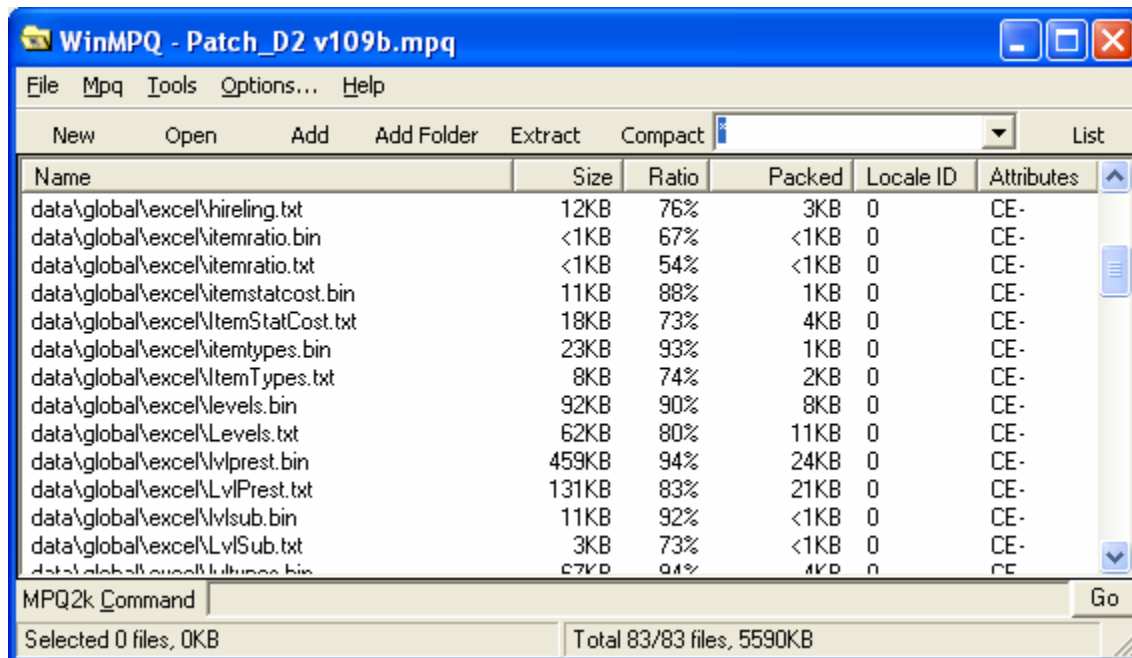


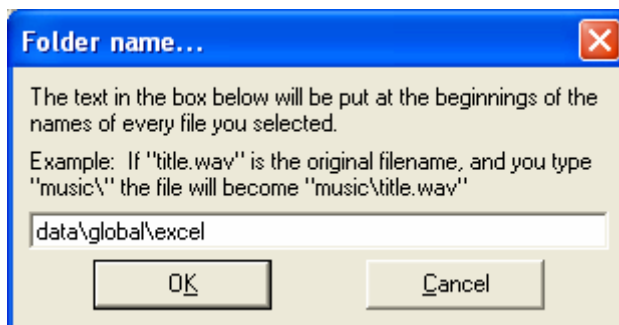
Figure 10 - WinMPQ File List

Back on the main screen, select Open and find your working copy of patch\_D2.mpq. As before, always insert your files into this mpq and leave the release archives untouched. You will now see a list of files stored within the mpq, and some information about their sizes and compression.



**Figure 11 - MPQ File Display**

Click on the Add command to include your new files. Select the changed txt and new bin files. Then you have to assign a folder for the added files. Type in the correct folder, as shown below. Remember that txt and bin files should always be added to data\global\excel if you want to see your changes take effect!



**Figure 12 - Selecting the Destination Folder**

The added files will then be inserted and the mpq saved. This will automatically overwrite existing files in the mpq, so be sure you have your backups made ahead of time.

You can also use WinMPQ to extract files from an archive. Open an mpq as before, select the file you want and click on the Extract command above the file list. Also, the List command will refresh the file list display; you can use the adjacent dropdown list to select a file filter.

## Exercise #4 – Modifying Item Properties

In this exercise, you will make property changes and learn about the tbl files. We are going to replace the unique ring Manald Heal with a much more powerful unique, The Eye of Mordini. Except as detailed below, all steps are the same as in the first exercise.

Open MPQ Viewer and extract UniqueItems.txt from the mpq's data\global\excel directory and save to your workspace directory. Then look for the string table named patchstring.tbl. Here we make use of the filter function in MPQ Viewer, but you have to edit the default .TBL filter to lower case for it to work. (The filters are buggy and in some cases will miss files. The only way to assure yourself of seeing every file is to leave the filter as \*.\* and scroll down the entire list.)

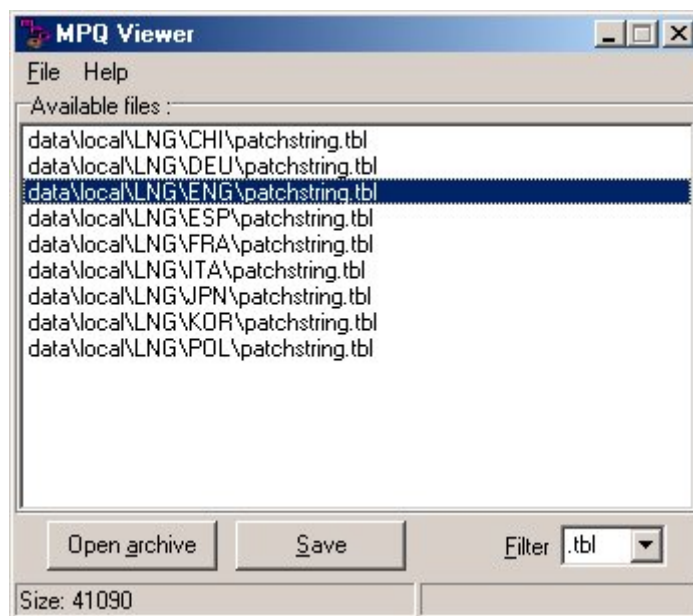


Figure 13 - Choosing a PatchString.tbl File

You will see there are a number of patchstring.tbl files. Pick the file that is in the directory corresponding to the language you run Diablo II in. For example, if you have a Spanish installation, choose the string table in the data\local\LNG\ESP directory. Extract the appropriate file to your workspace directory.

Now load UniqueItems.txt for editing as below. This example shows the database file in Microsoft Excel. Scroll down until you see the record for Manald Heal. Remember from the first exercise this name is *not* the one that displays in the game!

118	Bladebuckle	U	1
119	Nokozan Relic	0	1
120	The Eye of Etlich	0	1
121	The Mahim-Oak Curio	0	1
122	Nagelring	0	1
123	Manald Heal	0	1
124	The Stone of Jordan	0	1
125	Amulet of the Viper	0	1
126	Staff of Kings	0	1
127	Horadric Staff	0	1
128	Hell Forge Hammer	0	1
129	KhalimFlail	0	1
130	SuperKhalimFlail	0	1

Figure 14 - Finding Manald Heal in UniqueItems.txt

Scroll right through the fields until you get to the one labeled ItemMod1Code (column O in Excel). This is the first magical property that the ring possesses, “manasteal” which is obviously Steal x% Mana. Download the Magic Code Master List from the Phrozen Keep to see what all of the magic modifiers are, or you can extract Properties.txt from the mpq for yourself.

nc	ItemMod1C	ItemMod1F	ItemMod1M	ItemMod1M	ItemMod1M
19	manasteal		4	7	regen
0	mana		20	20	mana
1	mana		10	10	res-p

**Figure 15 - First Property of Manald Heal**

There are four fields associated with each property. The first field is the Magic Code, followed by three fields (ItemMod1Param, ItemMod1Min and ItemMod1Max) that define the strength of its effects according to the Master List. The first property for Manald Heal is shown above. ItemMod1Param is blank, but the other two fields are used. With ItemMod1Min less than ItemMod1Max, when the game spawns a Manald Heal the ring will appear with a random % Mana Steal between the two values.

Change the ItemMod1Code field from “manasteal” to “mana%” and change both ItemMod1Min and ItemMod1Max to 50. You have now removed the mana stealing ability from the ring and added +50% to your maximum mana. With both ItemMod1Min and ItemMod1Max at the same value, that property no longer has a range of possible values; it will always spawn the same.

nc	ItemMod1C	ItemMod1F	ItemMod1M	ItemMod1M	ItemMod1M
9	mana%		50	50	regen
0	mana		20	20	mana
1	mana		10	10	res-pi

**Figure 16 - Changing to Increased Maximum Mana**

UniqueItems.txt has fields for up to ten separate properties, each with its four-field definition; all ten work the same way. Below are two examples of other changes you can make. Once you are done, save the file. If you are using Microsoft Excel or other spreadsheet, be sure to save the file as *tab delimited text file* or it won't work in the game.

11	ItemMod2C	ItemMod2F	ItemMod2M	ItemMod2M	ItemMod2M
50	res-all-max		50	50	hp
20	mana%		25	25	ltng-r
10	res-pois		25	25	hp

**Figure 17 - Add 50% to All Maximum Resists**

11	ItemMod2C	ItemMod2F	ItemMod2M	ItemMod2M	ItemMod2M
50	allskills		1	3	hp
20	mana%		25	25	ltng-r
10	res-pois		25	25	hp

**Figure 18 - Add +1 to +3 to All Skills**

## Editing String Tables

We are not done yet. We changed the properties of the ring, but it is still called Manald Heal. So we have to change the string tables to show the new name for the item. Before we roll up our sleeves and get to work, we need to discuss what string tables are and how they function.



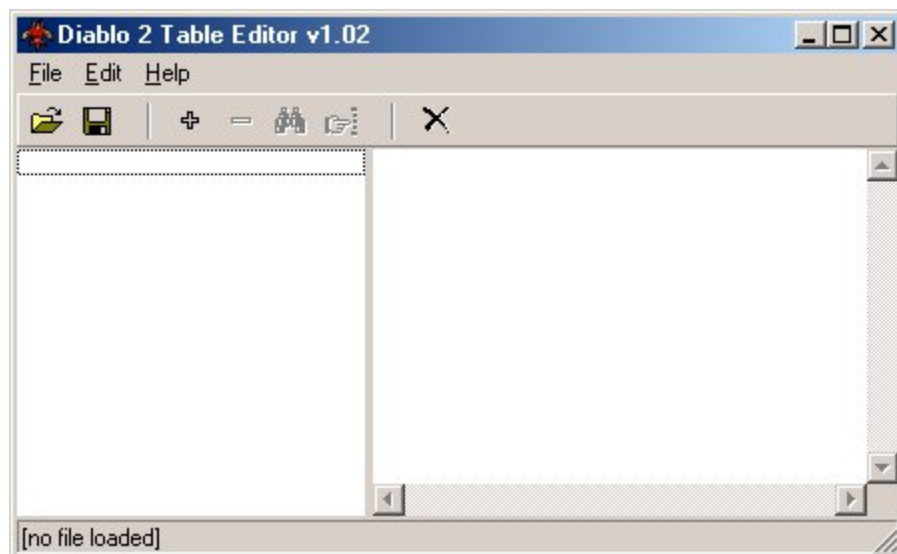
Almost all text displayed in the game is stored in .tbl files. This includes items and their properties, monsters, NPC names, Battle.net logging and error messages, and all of the scrolling text seen when you talk to a NPC or click on one of the tomes. There are three .tbl files for each language. The oldest file, String.tbl, is located in d2data.mpq; it contains all of the original game's strings. Expansion strings are stored in ExpansionString.tbl located in d2exp.mpq. And most recent changes are stored in PatchString.tbl in patch\_D2.mpq.

The string tables have two fields, regardless of which file you are using. The left field is the string key and the right field is what displays on screen. In order to properly show an item, the string key in the source database must match exactly the string key entered into a tbl file. In the case of Manald Heal, the name is used in both fields, so the string key in UniqueItems.txt was easy to decipher. But it isn't obvious that a Hand Axe is called a Hand Axe when its string key is the three-letter code "hax".

Sometimes a string is defined in more than one file. If so, any definition in ExpansionString.tbl will override any definition in String.tbl. Any definition in PatchString.tbl will override any definition in either of the other two files. You might think, why not make changes in PatchString.tbl and not worry about the other two? There are two reasons why you should not do that. The first reason is that every time Blizzard releases a patch, there will be a new PatchString.tbl. If you defined five hundred strings for a major mod, it is *not easy* to move those strings to the newer version of the file. The second reason is instability in PatchString.tbl when using the Darkstorm editor (an internal error, and unfortunate since the editor is very easy to use). For those two reasons, many mod makers choose to extract String.tbl from d2data.mpq and pack it into patch\_D2.mpq after they finish all of their changes. String.tbl is not changed at all by patches or the Expansion, so additions there are safe. But if you make a change to a record that appears in either of the other .tbl files, you have to change the record in those files. If you change String.tbl and not the others, you will never see your new string displayed.

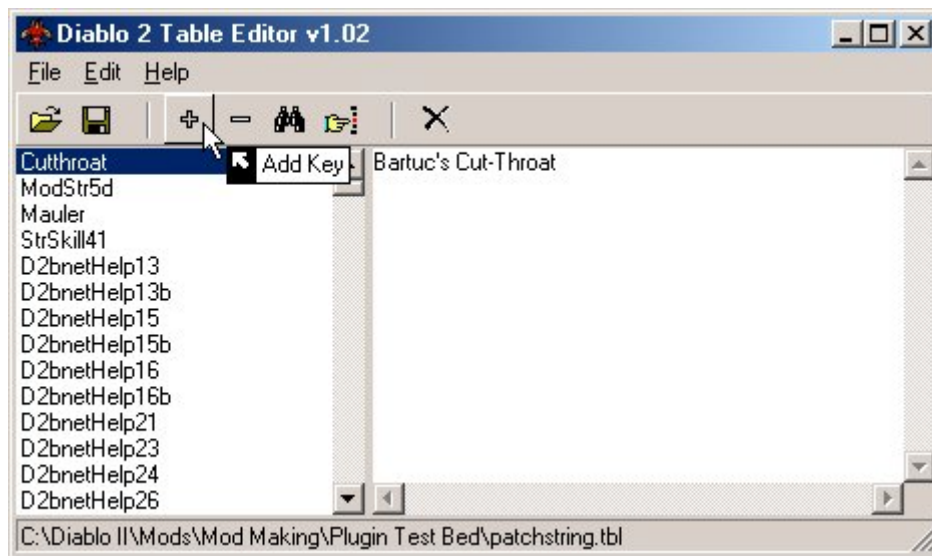
Enough theory for now, let's change Manald Heal to The Eye of Mordini. Manald Heal is named in String.tbl since it is from the original Diablo II, so we can either change the display text in String.tbl or we can add a new record to PatchString.tbl. We will look at how to do both methods, even though you only extracted PatchString.tbl earlier.

Run the Darkstorm Table Editor (file name tbled102.exe). You will get this opening screen:



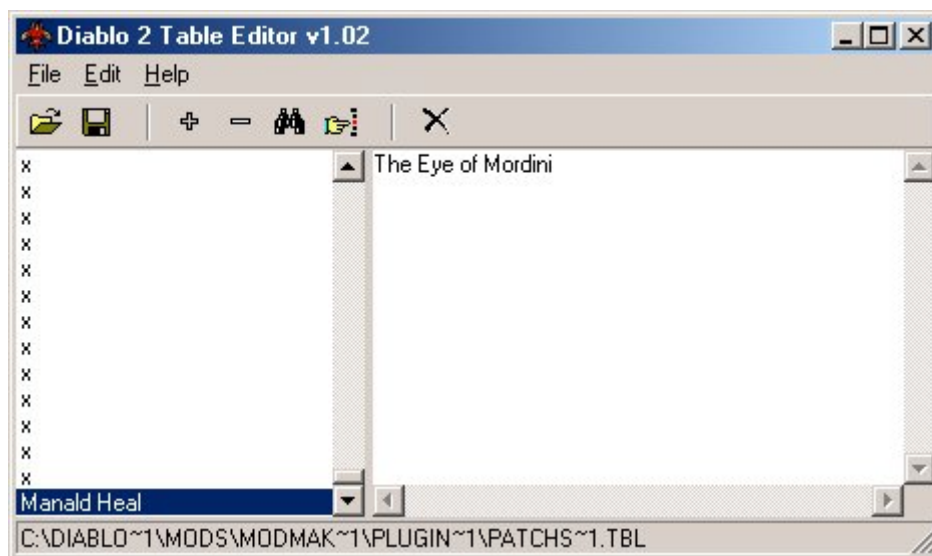
**Figure 19 - Darkstorm Table Editor**

Open PatchString.tbl and scroll down towards the bottom. You will notice the display string in the right field is only displayed when you have its string key selected. Also, some keys are nothing more than an “x” or occasionally a capital “X”. These keys do not correspond to records in any of the txt files; they are special application strings directly referenced from the executable files. You will also see strings where the display text reads “All Resistances +%d” – this is where the text includes a variable that is filled in when the game is running.



**Figure 20 - Table Editor with PatchString.tbl Loaded**

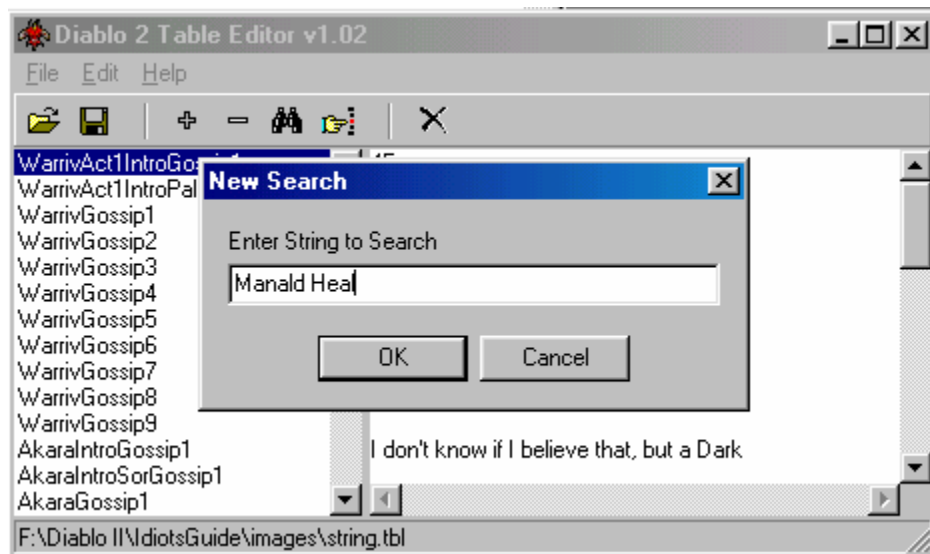
Since we are adding a new key to PatchString.tbl, click on the Add Key button as shown above. You will jump to the end of the table and your cursor will be in the left field. You have one chance now to enter the new string key; you must match exactly the key for Manald Heal in UniqueItems.txt. Fortunately, the key is the same as the display name, so enter “Manald Heal”. On the right, enter “The Eye of Mordini” as shown below. It doesn’t matter to the user that the display name doesn’t match the internal name at all. If you make a mistake, you must delete the key (button next to Add Key) and start over. Be sure you *never delete any of the existing keys*, if you won’t be using one then delete the display text for that key only!



**Figure 21 - Adding a New Key and String**

Now save the .tbl file to your workspace directory.

If you were changing String.tbl instead of adding to PatchString.tbl, the process is similar. But instead of clicking on the plus for Add String, you would click on the binoculars for Find String. This function only works on the right field; you can't search by string key. Search for "Manald Heal", and when that record is found edit the right field to read "The Eye of Mordini".



**Figure 22 - Using the Search Function**

As before, copy your modified UniqueItems.txt file to your data\global\excel directory. But now you need a place for the modified PatchString.tbl. So enter the data directory, and next to global add a new directory called **local**. Inside that directory make one for **LNG**, and inside there make one for the language code of your choice – ENG for English, DEU for German, etc. Copy your PatchString.tbl to the new directory.

If you want to force an English version for your tbl files, for all users regardless of country version, make a special file by opening Notepad, typing one and only one space, then saving the file as **Use** (without any extension). Place this file in the Data\Local directory, and pack it in your mod mpq when you make one. Your tbl files would then go to the Data\Local\Lng\Eng directory.

Using your -direct -txt shortcut, start a game and generate bin files in your data\global\excel folder. Find your new UniqueItems.bin and copy it to your workspace. Now make a new version of your MPQ2K script file so you add the three changed files to patch\_D2.mpq:

```
O patch_D2.mpq
a uniqueitems.txt data\global\excel\uniqueitems.txt
a uniqueitems.bin data\global\excel\uniqueitems.bin
a patchstring.tbl data\local\LNG\ENG\patchstring.tbl
C patch_D2.mpq
```

When you look through other txt files, you will notice many of them have fields similar to those in UniqueItems.txt. *Any time you see the four fields \*code, \*param, \*min and \*max in a group (\* being a common group name), they all work the same as in this file.*

### **Divide and Conquer!**

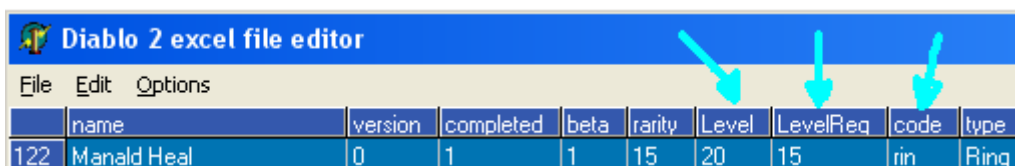
These last two exercises show some simple changes. A full-blown mod has hundreds and even thousands of changes in it. Don't change everything at once! Start out making one change at a time, test and correct, then make the next change. As you gain experience you will feel more comfortable making more than a few changes between tests. But the more changes you test at one time, the harder it is to find the change that isn't working right.

In the next exercise you will learn how to test your new items.

## Exercise #5 – Forcing an Item Drop

By now you have realized it is easy to find keys and arrows, but Paladin shields and Manald Heal rings don't fall out of the sky very often. There are several techniques available for forcing a specific drop, but going into their theory requires a certain familiarity with mod making. The method presented here is the Quill Rat Drop, named after those ubiquitous pests in the Blood Moor. It is easy enough if you don't worry too much about how it works. And here we make the distinction between a debugging version of a modified file and a release version.

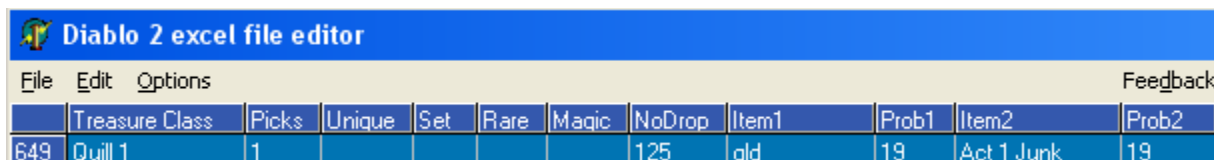
For this test we will extract and modify the file `TreasureClassEx.txt`. This is a fearsome-looking database containing most of the mechanism for selecting what gets dropped when you kill a monster or pop a chest. For now we will look at a very small part of this file, since Blizzard was kind enough to provide a set of records specifically for Quill Rats.



Diablo 2 excel file editor									
File	Edit	Options							
	name	version	completed	beta	rarity	Level	LevelReq	code	type
122	Manald Heal	0	1	1	15	20	15	rin	Ring

Figure 23 - Manald Heal from UniqueItems.txt Revisited

We will use the files we modified for the last exercise; we want to see what The Eye of Mordini actually looks like, and what it does for our characters. Make a copy of `UniqueItems.txt` so any changes we make can be reversed quickly; this edit is for debugging only. In `UniqueItems.txt`, look for the Manald Heal record again, this time viewing the first few fields. We are interested in three fields as shown above. First, change both Level and LevelReq to 1. This will allow a Quill Rat to drop the ring, and for our test character to wear it. Next, make note of the three-letter code "rin". We will need this code to tell the game we want Quill Rats to drop it. Save the file and copy it over to your `data\global\excel` directory.



Diablo 2 excel file editor											Feedback
File	Edit	Options									
	Treasure Class	Picks	Unique	Set	Rare	Magic	NoDrop	Item1	Prob1	Item2	Prob2
649	Quill 1	1					125	gld	19	Act 1 Junk	19

Figure 24 - Quill Rat Record in TreasureClassEx

Now open `TreasureClassEx` and scroll down to the first Quill Rat row. You will see these fields used, and everything to the right is blank until you get far over. This part shown is all we need though. First, delete all information from the NoDrop, Item2 and Prob2 fields. Next, replace "gld" with "rin" so the Quill Rat will drop a ring. Now go to the Unique field and enter the number 1024. Without going into a lot of detail, this forces the drop to be a unique ring. Quill Rats are very low level beasts in Normal difficulty, so the only unique ring they can drop will be our modified Manald Heal. Save this file and copy it to your `data\global\excel` directory.

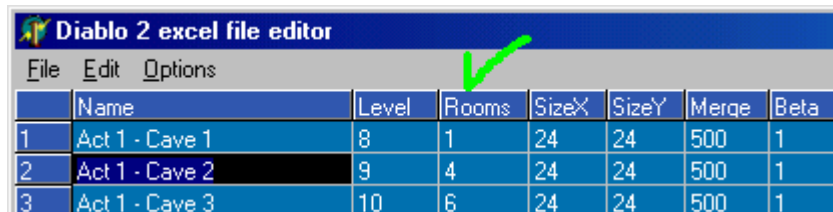
Now start a game using the `-direct -txt` shortcut. Go out into the Blood Moor and kill a few Quill Rats. Check their drops and you should find several copies of The Eye of Mordini. Identify them and they should be wearable. Put one on and look over your stats. Compare the visible properties and effects against what you intended with your changes to `UniqueItems.txt`. If everything looks fine, you have made your changes successfully and that feature is ready for release. Remember not to use your debugging files for mod release!

This method only works where the base item or magical property don't depend on the level of the character or monster, such as magic and rare items. To make the test more effective, you need to create a Level 90+ test character with access to Hell difficulty. You can use Jamella's Hero Editor or Shadowmaster to create high-level test characters. You won't need to change Level and LevelReq for most items when testing with a high-level character in Hell difficulty. Change the Quill 1 (H) record in `TreasureClassEx`, ten records past the one shown here, to force high-level drops.

## Exercise #6 – Increasing Area Size

So far, all of the mods have been changes to item definitions and availability. There are many more ways to change the game! Changing and importing monsters, altering quests, changing graphics and NPC speech are features in many mods available at the Phrozen Keep, but many of these features are beyond the scope of this introductory guide. One change which is not difficult is to modify how certain levels are generated.

There are three types of areas in Diablo II. Outdoor levels are composed of many sections which are organized somewhat randomly. Each section is a separate .ds1 file. These areas are not expandable (yet). Other areas are “hard-coded” into one or more configurations, with special monsters and objects included as part of the .ds1 file. These areas are difficult to expand.



	Name	Level	Rooms	SizeX	SizeY	Merge	Beta
1	Act 1 - Cave 1	8	1	24	24	500	1
2	Act 1 - Cave 2	9	4	24	24	500	1
3	Act 1 - Cave 3	10	6	24	24	500	1

Figure 25 - Editing Maze Size

The third area type is the Maze, which is easy to expand. All you need to do is open LvlMaze.txt and edit the Rooms field. In version 1.10, this table has additional Rooms fields so you can have different sizes for each difficulty.

Mazes are found in most Acts:

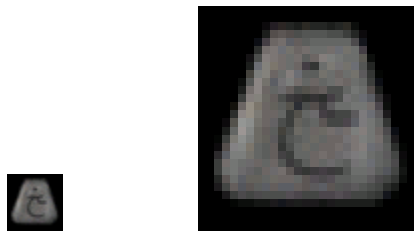
- |       |  |
|-------|--|
| Act 1 | First floor of all caves (excluding Den of Evil)<br>Crypt and Mausoleum underneath the Rogue Cemetary<br>First four levels of the Forgotten Tower<br>Jail and Catacomb levels of the Corrupted Monastery, except Andariel's Lair |
| Act 2 | Sewers under Lut Gholein<br>Stony Tomb<br>Halls of the Dead<br>Maggot's Lair<br>Ancient Tunnels<br>First floor of the Claw Viper Temple  |
| Act 3 | both Spider caves<br>Flayer Dungeon<br>Swampy Pit<br>first level of Sewers under Kurast<br>first two levels of the Durance of Hate   |
| Act 4 | {none}   |
| Act 5 | First floor of all Ice Caves   |

Other areas are listed in LvlMaze.txt. Some of them are known not to work with this method, such as the second (or “Treasure”) levels of Act 1 caves or the three mini-Hells of Act 5. Others *might work* as they haven't been tested.

## Exercise #7 – Item Images

We will touch briefly on adding and changing the pictures which are shown when you are in the inventory screens. Item images are stored in dc6 files located in the data\global\items folder of the archive files. This isn't a format that image editors can work with, so you need to use an image converter. To create a new image, you need to make a picture file, then convert it to dc6 format. To edit an existing image, you convert from dc6 format, edit the resulting picture file, then convert it back to dc6 format. For this exercise we will make a new rune image. But there are no blank runes in the mpq!

The hard way is to extract an existing rune from d2exp.mpq, convert it to an editable format, and wipe out the rune symbol. This is very difficult to do and still retain the lighting effects on the rune image. Fortunately this isn't necessary, as there are item packs available at the Keep File Center, and some of them have what we need already. One such is Riparious' Gem Pack, which has two different blank runes in both bmp and dc6 format. Load the bmp version in the image editor of your choice and draw a runic symbol. The sample below is an Arabic symbol from the Riparious pack, which illustrates the difficulties in working with inventory images. The small image at left is the true size.



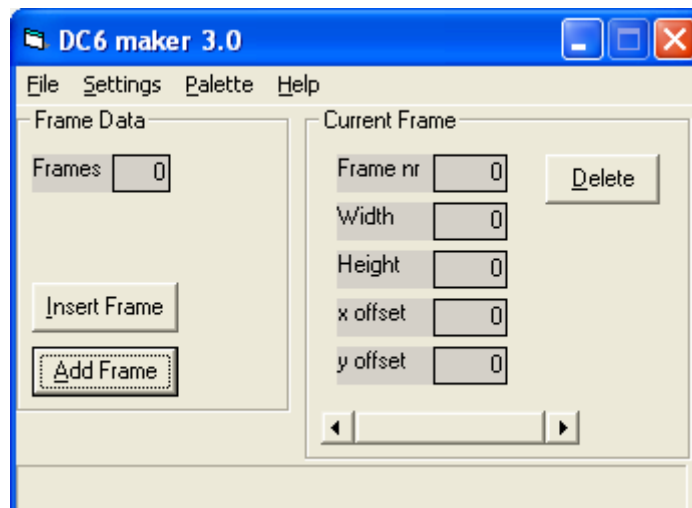
**Figure 26 - Sample rune image, actual and enlarged**

Inventory images must be particular sizes, in order to line up properly with the inventory grids. Each grid unit is 28 pixels in either direction, and images must be an integral multiple of a grid unit.

Size	Dimension	Size	Dimension	Size	Dimension	Size	Dimension
1x1	28x28	2x1	56x28	3x1	84x28	4x1	112x28
1x2	28x56	2x2	56x56	3x2	84x56	4x2	112x56
1x3	28x84	2x3	56x84	3x3	84x84	4x3	112x84
1x4	28x112	2x4	56x112	3x4	84x112	4x4	112x112

**Table 5 - Common inventory image dimensions in pixels**

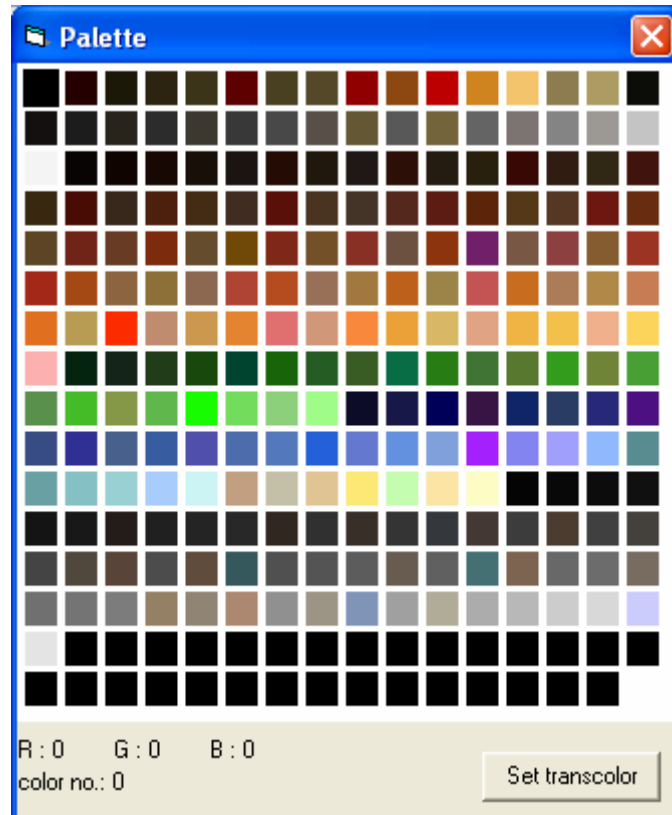
Save your image as a pcx or bmp file, in 256 colors. Diablo II does not use the higher color modes, and your images may not look right if you save in high color or true color modes. Then start DC6 Maker.



**Figure 27 - DC6 Maker opening screen**

DC6 Maker is like Notepad, you start with a blank workspace to begin with. When you are converting an image file to dc6, you start by inserting it as a frame. This is because the dc6 format is flexible and supports multi-frame images and even some animations. But inventory images are static, so we need only one frame for each file. Click the Insert Frame button and select your image file. You will see a small preview window pop up to the right of the application window.

Each act has its own color scheme, stored in a palette of 256 colors. That is fine for objects and monsters but not so great for inventory. First, you need to pick a static palette that won't change between acts. Click on Palette and select the units palette. Then go to Settings and select Use Transcolor. This sets the color we want to use for transparent backgrounds, such as the black in the rune image. Inventory uses color 0 (pure black) as the transcolor, so on the palette window pick the upper left corner color square and verify at bottom you selected 0,0,0. Then click on the Set Transcolor button.



**Figure 28 - Selecting the transparent background color**

Back on the main screen, pick Save As from the file menu, then name the file the way you will refer to it in the game. Then go to the txt file which has the item you just made an image for, and put the file name (without the .dc6) in the invfile field. If you have made a weapon or armor image for a set or unique item, use the setinvfile or uniqueinvfile field instead.



## Exercise #8 – Cube Recipes

One of the most entertaining facets of the game is the Horadric Cube. Blizzard provided a few useful recipes to get each of us started, but the Cube is capable of far more than that. In this exercise, we will cover the basic structure of a cube recipe, and correct a deficiency in the game. Start by extracting cubemain.txt from the mpq. The structure of cubemain.txt was radically altered with version 1.09, so this exercise requires you be running 1.09 or any of the later sub-releases (1.09b, 1.09d). The file is again radically changed in 1.10, but the parts we will be working with are still the same.

A good way to learn cube recipe structure is to look at the Blizzard recipes. For example, if you want to have all runes be upgradeable, not just the first nine, then complete the series which is shown in part below.

	description	enabled	min diff	version	start day	stop day	weekday	class
57	3 rune 07 -> rune 08	1		100				
58	3 rune 08 -> rune 09	1		100				
59	3 rune 09 -> rune 10	1		100				
60	3 perfect gems + 1 magic item -> magic item	1		100				
61	6 perfect skulls + 1 rare item -> 1 low level rar	1		0				
62	1 perfect skull + 1 rare item + soj -> 1 high lev	1		0				

Figure 29 - Cubemain.txt Character Qualification section (1.09 shown)

The first part of the database is the character qualification section. The first field is a description of the recipe, it is only for reference and has no game impact. A recipe must be enabled to be usable, this is a Boolean field so set it to 1. Min diff is the minimum difficulty required: 0 or blank for normal, 1 for Nightmare and 2 for Hell. Version is 0 for either Diablo II or expansion games, 100 is expansion games only. The next three fields compare these values to the day of the week and day of the month retrieved from your system clock, these appear in versions 1.09b/d only (in 1.10 beta they are Op, Param and Value and have more general application). Class is the three-letter character class code; if you want to restrict your recipe to Necromancers, put “nec” in this field.

	numinputs	input 1	input 2	input 3	input 4	input 5	input 6	input 7
58	3	"r08,qty=3"						
59	3	"r09,qty=3"						
60	4	"any,mag"	"gem4,qty=3"					
61	7	"any,rar"	"skz,qty=6"					
62	3	"any,rar"	skz	The Stone of Jordan				
63	5	"any,rar,nos"	"skz,qty=3"	The Stone of Jordan				
64	4	"fhl,mag,upg"	jew	r06	gpb			
65	4	"mbt,mag,upg"	jew	r08	gpb			

Figure 30 - Input Section of Cubemain.txt

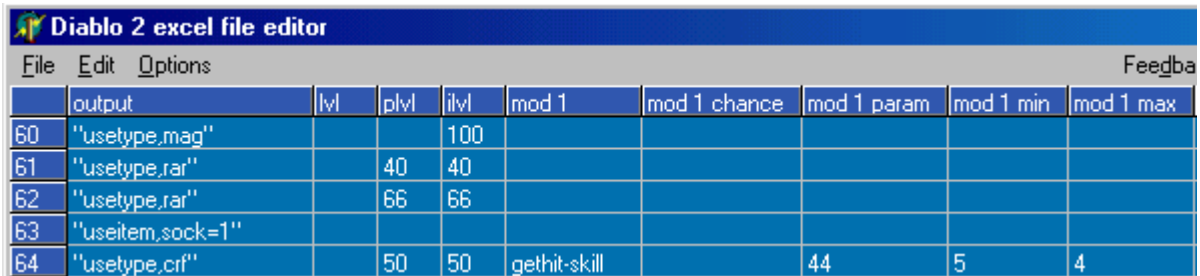
The second part of Cubemain.txt is the input section. These fields are the ingredients used by the recipe. Some Blizzard recipe ingredients are shown above. The first input is important, any recipe that transforms an object works on whatever object is in this first input field. Also, the number of ingredients must equal the sum of all inputs including the individual quantities. Quotes are used for any input with sub-fields, such as “fhl,mag,upg”. Three-letter item codes or four-letter type codes are permitted.

The table on the next page lists some of the parameters used for inputs and outputs. Most parameters work as either input or output parameters.



=	joins parameter to numeric value	upg	upgradeable recipe, also works for exceptional and elite versions	any	any item, but this parameter sometimes doesn't work
low	low-grade item	crf	crafted item	eth	ethereal item
nor	normal item	tmp	tempered item (1.10 only)	nos	no sockets on item
hiq	superior item	bas	base item (1.10 only) - input	sock	add sockets (needs =)
mag	magical item	reg	base item (1.10 only) - output	pre	magical prefix (needs =)
rar	rare item	exc	exceptional item (1.10 only)	suf	magical suffix (needs =)
set	set item	eli	elite item (1.10 only)	qty	quantity (needs =)
uni	unique item	noe	not ethereal item		

**Table 6 – Some cube recipe parameters**



	output	lvl	plvl	ilvl	mod 1	mod 1 chance	mod 1 param	mod 1 min	mod 1 max
60	"usetype,mag"			100					
61	"usetype,rar"		40	40					
62	"usetype,rar"		66	66					
63	"useitem,sock=1"								
64	"usetype,crf"		50	50	gethit-skill	44	5	4	

**Figure 31 - Cubemain.txt output section**

The third part of Cubemain.txt is the output section. These fields describe the properties of the newly-generated item. Additionally, there are three important parameters that are output parameters only: useitem, usetype, and sock = #. Useitem will take whatever item is listed in the first input field, including all properties already present, and apply whatever properties are assigned in the output section. You can use the useitem parameter to make recursive recipes, where you keep improving an item up to game limits by reapplying the recipe over and over. Usetype will take the base item from the first input field, strip all existing properties away (including ethereality!), “re-roll” the item’s characteristics and apply output properties. Finally, sock will apply the designated number of sockets to an item, up to game-defined limits.

The three level fields (lvl, plvl and ilvl) control the level requirements for random properties. If lvl is used, the other two fields are ignored and all random properties will be that level or higher. If only plvl is defined, then the maximum level for random properties (also known as ilvl and this can be confusing) is plvl% of player's level, and there is no minimum. If only ilvl is defined, the value is the percent of the level for the item in the input1 field, or the old\_ilvl. If plvl and ilvl are both defined, then plvl% of the player's level is the minimum, and ilvl% of old\_ilvl is the maximum. In any case, the maximum level for properties is capped at 99 for all practical purposes. The character level required to use the new item will be the level requirement for the highest mod on the item, *plus* 3 for each random mod.

There are five property blocks after that, the first one of which is shown above. These blocks are used the same way as the property blocks in UniqueItems.txt, with one addition. If a value is used in Mod # Chance, that property has the value percent chance of being generated. So if you want a property to appear about a third of the time the recipe is used, put 33 in this field.

In 1.10x, there are actually three output sections. The second and third output sections are structured the same way as the first, but field names are similar to b ilvl and c mod2 param. We won't be using the other outputs in this exercise, but they can be used to have cube recipes with more than one output. An example would be a catalyst recipe, where an object used as a recipe component is returned to you unchanged.

Okay, we have thrown a lot of information out about cube recipes, how do we use it? Let us correct a deficiency Blizzard coded into the game. After Act 2 normal difficulty, it is impossible to buy normal items from the vendors, everything is magical. So we will make two recipes, one to convert magical weapons to superior socketed weapons, and one to convert magical armor to superior socketed armor. This is one of those cases where the “any” parameter does not work, so we need a recipe for each. We will allow any magical, non-socketed item plus two Identify Item scrolls create the maximum number of sockets possible in the new item. See the previous figures for locating the right fields to input data.

Make two new rows at the bottom of Cubemain. When adding rows, it is important to remember never to insert a row, always add it on the bottom. Inserting rows may work in a few txt files, but causes severe problems in most. Type a brief description of the recipe function in the first field for each row. Then skip over to the input section and enter 3 in the NumInputs field. In the first input field of the weapons record, enter “weap,nos,mag” for magical, non-socketed weapon. For the armor record, “armo,nos,mag” is put in the first input field. In both records, the second field should read “isc,qty=2”. Now scroll to the output section, and put “usetype,hiq,sock=6” in the output field for both records.

Save the file, make bins, pack to mpq as in previous exercises. Go into a game and find or buy a magical weapon and a magical armor. Get some identify scrolls and test the recipes. Remember you won’t get six sockets for many items, and in the lower difficulties no more than three. You will also see the new items can get superior grade properties where those are generated. With multiple recipe pairs, you can control the number of sockets more precisely, and have any item available for runewords! A sample set of recipes is shown below, downgrading a magical item to a superior item using any charm. Perfect gems are used to specify how many sockets are added.

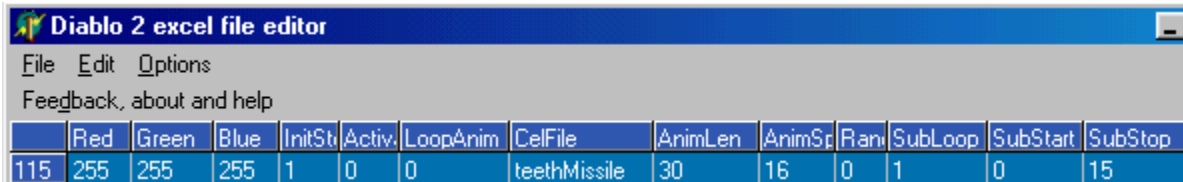
	I	J	K	L	M	N	O	P	Q
1	numinputs	input 1	input 2	input 3	input 4	input 5	input 6	input 7	output
125	2	weap,mag,nos	char						usetype,nor,hiq
126	3	weap,mag,nos	char	gpv					usetype,nor,hiq,sock=1
127	3	weap,mag,nos	char	gpr					usetype,nor,hiq,sock=2
128	3	weap,mag,nos	char	gpb					usetype,nor,hiq,sock=3
129	3	weap,mag,nos	char	gpg					usetype,nor,hiq,sock=4
130	3	weap,mag,nos	char	gpy					usetype,nor,hiq,sock=5
131	3	weap,mag,nos	char	gpw					usetype,nor,hiq,sock=6
132	2	armo,mag,nos	char						usetype,nor,hiq
133	3	armo,mag,nos	char	gpv					usetype,nor,hiq,sock=1
134	3	armo,mag,nos	char	gpr					usetype,nor,hiq,sock=2
135	3	armo,mag,nos	char	gpb					usetype,nor,hiq,sock=3
136	3	armo,mag,nos	char	gpg					usetype,nor,hiq,sock=4
137	3	armo,mag,nos	char	gpy					usetype,nor,hiq,sock=5
138	3	armo,mag,nos	char	gpw					usetype,nor,hiq,sock=6

Figure 32 - Sample set of socketing recipes

## Exercise #9 – Editing Skills

Skills editing is a feature that covers the spectrum of mod making difficulty. Some changes can be accomplished by editing the skills.txt table, whereas other changes involve editing graphic files and even the game code itself. We will limit ourselves in this exercise to txt file changes. We will change the Necromancer Teeth skill into Ice Bolts.

For this exercise, extract missiles.txt, skills.txt and string.tbl from the mpqs. In the figures below, certain fields have been squeezed to bring important fields together. Between versions 1.09b/d and version 1.10 there are many changes in both Missiles.txt and Skills.txt. Although the figures below are for 1.09b/d, the fields we will change are the same in both versions.



	Red	Green	Blue	InitSt	Activ	LoopAnim	CelFile	AnimLen	AnimSp	Rani	SubLoop	SubStart	SubStop
115	255	255	255	1	0	0	teethMissile	30	16	0	1	0	15

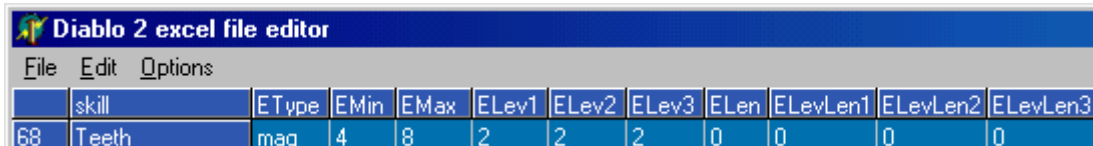
Figure 33 - Teeth in Missiles.txt

Start with missiles.txt, and scroll down to the Teeth record. Scroll right to the RGB color fields as shown above. The default for Teeth is 255/255/255, which is bright white. To give the new ice bolt a bluish cast, change these fields to 81/81/255.

CelFile is the name of the missile animation without the dcc extension. Change this field to “icebolt” (without the quotes). Each dcc file has a specific number of animation frames, and that number must match the number in the AnimLen field. Change this field to 6.

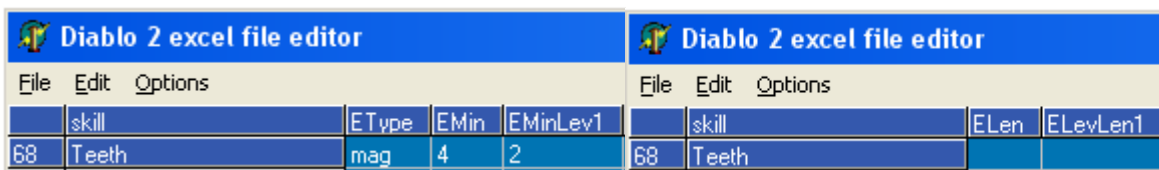
LoopAnim, SubLoop, SubStart and SubStop are fields which control how animations are repeated while the missile is in flight. Change LoopAnim to 1 and the other three fields to 0.

Check your changes against the existing record for Ice Bolt to confirm they match in the specific fields affected. Then save your work in missiles.txt.



	skill	EType	EMin	EMax	ELv1	ELv2	ELv3	ELen	ELvLen1	ELvLen2	ELvLen3
68	Teeth	mag	4	8	2	2	2	0	0	0	0

Figure 34 - Teeth in Skills.txt (versions 1.09d/d)



	skill	EType	EMin	EMinLev1
68	Teeth	mag	4	2

	skill	ELen	ELvLen1
68	Teeth		

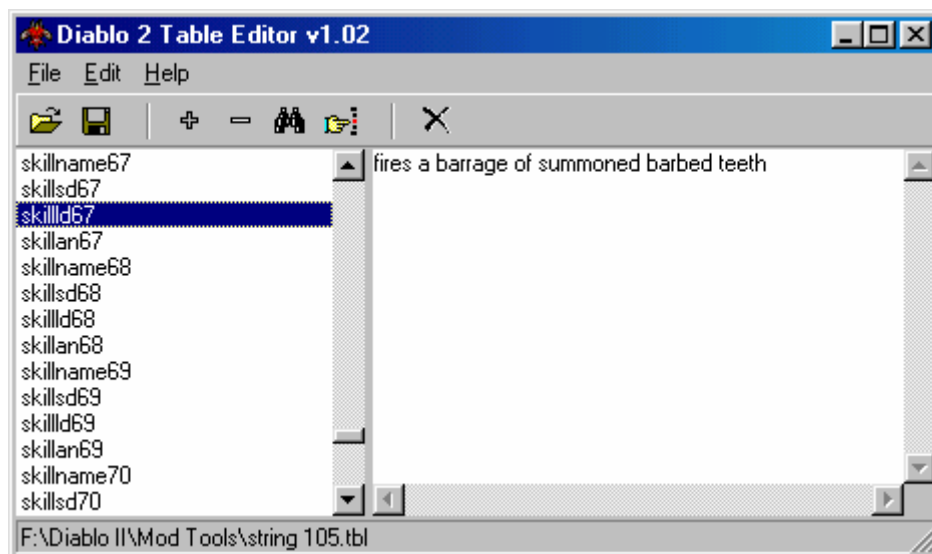
Figure 35 - Teeth in Skills.txt (version 1.10)

Open skills.txt next and find the row for the Teeth skill. Scroll right until the elemental damage fields are visible, as shown above. Teeth is damage type magic but Ice Bolt is magic type cold, so change the Etype field to cold. The other fields shown here control the damage, which we will not change this time.

The Elen fields control the duration of the skill effect. The basic unit of time in Diablo II is the frame rate. There are 25 and only 25 frames per second, this is hard-coded into the game. Cold is one of the damage types that is time-dependent, so we need some value other than 0 in the Elen field. For two seconds of cold duration, put 50 in the Elen field. The other three fields control the duration per skill level. We will leave these three fields at 0.

Save your changes in skills.txt. Now we will change the strings associated with the newly-altered skill.

In the Table Editor, open string.tbl and use the search function (binoculars icon) to find instances of “teeth” in the file. The first instance is at string key strSkill28, which is a generic listing of the missile type. Change this value to “bolts”. The next instance is at the head of the Teeth Skill key block, shown below.



**Figure 36 - Teeth Skill String Block**

The first key of this block, SkillName67, is the display name for the skill. You may want to see “Ice Bolts” or “Ice Teeth” or something completely different; this is the record to change. Below that are two records for describing the new skill. Skillsd67 is the short description, it currently reads “fires barbed teeth”. The next record is the long description, shown above. Edit these two fields to provide suitable descriptions, such as “fires ice bolts” for the short form and “fires a sleeting barrage of deadly frigid shards” for the long form. The last record of the skill block is the alternate name, skillan67. In most cases the alternate name is the same as the display name, so change this copy from “Teeth” to whatever you have in the skillname67 record.

Save these changes to string.tbl. Generate bin files for missiles.txt and skills.txt, and insert all five files into patch\_D2.mpq. Now play your Necromancer and see how the skill has changed. You should now see cold damage against your foes. When you highlight your skill, the name and description changes should read as you entered them in string.tbl.

Many skills have information which is linked through code to the values in skills.txt. A more advanced technique where you swap skills between classes requires DLL editing. But this exercise shows you can change the behavior of skills considerably without knowing anything about graphics or assembly language.

## Exercise #10 – New Auras (1.10x only)

So far we have discussed changes that can be made in either 1.09x or 1.10x versions. This exercise and the next are for version 1.10x only; they use files and capabilities that are not available in 1.09x.

The easiest way to create a new skill is to clone an existing one. Open up states.txt and find ‘meditation’ in the states field. It should be ID number 48. Copy this entire line, and paste it at the bottom of states.txt. You should now have two lines of meditation. Rename the bottom one (the one you just pasted) to ‘auraregen’, and give it any unused ID number in states.txt, in this case 181. (We use ‘auraregen’ for the state name so it isn’t confused with ‘regen’ from properties.txt.) The figure below shows only those fields we will use for this exercise.

state	id	aura	immed	overlay1	overlay2
auraregen	181	1	1	aura_resistlight	

**Figure 37 - Changes to States.txt**

We want this to take immediate affect, so leave the 1 under the immed field alone. We are now done with states.txt, but if you want to have a different overlay for this aura, put the name of an overlay animation file in the field ‘overlay1’ and the field ‘overlay2’ (or as in this case, leave the second overlay field blank). Save states.txt.

Next open skills.txt. Find Meditation, its skill ID should be 120. Copy this line, and paste it at the bottom. Change the skill's name to ‘RegenHPMana’. Change the ID to 357 so it doesnt interfere with any of the other skills. Under the skilldesc column, put ‘auraregen’ there.

Still in skills.txt, skip over to the aurastate and auratargetstate fields. Under both of them, put ‘auraregen’. Now skip over to the aurastat1 field. It should say ‘manarecoverybonus’ there. Next to it, in the aurastatcalc1 field, it should say ‘ln34’. This is fine, since it is telling the game to recover our mana faster, which is half of the regeneration we want. Now go to the aurastat2 and aurastatcalc2 fields. Right now, those fields have ‘hitpoints’ and ‘skill(‘Prayer’.edns)’, which are the codes for the synergy bonus Meditation receives. We won’t want synergies, so we will use those fields for the life recovery power. In aurastat2 we will put ‘hpregen’ and in aurastatcalc2 we put ‘ln12’. We will use the radius and radius increase per skill level as the same numbers for hit points and points per skill level. The Guide does not cover the various types of calculations, but the numbers in the formulas used here are references to Param fields in Skills.txt.

A	B	C	D	W	X	AC	AD
skill	Id	charclass	skilldesc	aurastate	auratargetstate	aurastat2	aurastatcalc2
RegenHPMana	357	pal	RegenHPMana	auraregen	auraregen	hpregen	ln12

**Figure 38 - Changes to Skills.txt**

Now, still being in skills.txt, go to the right and find the reqlevel field. For testing purposes, put ‘1’ in this field. Remove any required skills listed in the reqskill1–reqskill3 fields, those list prerequisite skills needed in order to earn this skill. We are now done with skills.txt, so save it and exit.

We are now in the last database file, skilldesc.txt. Copy the line for Meditation, and paste it again at the end of skilldesc.txt. We will be modifying this line extensively! Change the name of the skilldesc to ‘RegenHPMana’, this must exactly match what was entered in the skilldesc field of Skills.txt. Now we will position the skill icon on the Combat Skills page (primarily for distinction) and place it in an unused spot. Under the SkillPage column, put a 3 there. Under the SkillRow column, put a 1. And finally, under the SkillColumn column, put a 2.

A	B	C	D	H	I	J	K
skilldesc	SkillPage	SkillRow	SkillColumn	str name	str short	str long	str alt
RegenHPMana	1	1	2	Skillname282	skillsd282	skillld282	skillan282

**Figure 39 – Placing Skill Icon and Adding Skill Text to Skilldesc.txt**

Now we need some text in the skill description to tell us what we have. As we saw in the previous exercise with Teeth, all skills have a common format: skill name, short description, long description and alternate name. For the new skill we will need to add the string records for Skillname282, skillsd282, skillld282 and skillan282. Open up a .tbl file (string.tbl is a good choice) and add the records shown as follows. It is common for Skillname and skillan to be the same.

Skillname282	Regenerate All
skillsd282	Regenerates HP and Mana
skillld282	gives life regeneration for you and your party when active, aura increases mana recovery and
skillan282	Regenerate All

**Table 7 – New Skill Descriptions**

This takes care of the general skill text, but what about current level, next level, and the leftover synergy description from when we copied Meditation? It isn't easy to work on these, but here is the general format. There are three different description sections (desc, dsc2, and dsc3) which control the following parts of the skill description:

desc\* -> level dependent description (bottom to top, six lines maximum)

dsc2\* -> level independent description (bottom to top, four lines maximum)

dsc3\* -> synergy descriptions (top to bottom, seven lines maximum)

Within each section, there are parameters for up to seven lines of description. The level dependent and level independent descriptions appear in reverse order (bottom to top), while synergy descriptions appear from top to bottom. Each line consists of a type (descline, dsc2line, dsc3line), one or two text references, and then one or two calculations. The type is a format string, from a list of predefined formats (refer to the file guide for the list, it isn't covered here). The text references are string keys for new and existing strings in the tbl files. The calculations are the same codes we used in Skills.txt to calculate skill effects based on the contents of Param fields.

For our new skill, we find the following descriptions already in place, because we copied Meditation:

	type	text a	text b	calc a	calc b
descline1	4	StrSkill50		skill('Prayer'.edmn)	
descline2	2	StrSkill88	StrSkill23	ln34	
descline3	19	StrSkill18		ln12	
dsc2line1	67	skillname99	Healplev2	skill('Prayer'.edmn)	
dsc3line1	40	Sksyn	skillname120	2	
dsc3line2	18	skillname99			

**Table 8 - More Skill Descriptions for Meditation**

In summary, descline1 is the effects per skill level of a synergy with Prayer, descline2 is the mana regeneration property of the skill, descline3 is the radius of effect for the aura. Then we have dsc2line1 for the invariant effect of the synergy. Finally dsc3line1 and dsc3line2 give us the listing at the bottom of the skill display that shows what the synergy is, although in this case no quantitative information appears.

Now we get to figure out what we want the skill description to say. Let's start by eliminating all synergy information, which we will do by clearing the contents of the fields associated with descline1 (except text a), dsc2line1, dsc3line1 and dsc3line2:

	type	text a	text b	calc a	calc b
descline1		StrSkill50			
descline2	2	StrSkill88	StrSkill23	ln34	
descline3	19	StrSkill18		ln12	
dsc2line1					
dsc3line1					
dsc3line2					

**Table 9 - Skill Description Without Synergy References**

What do we have left? In descline1 we kept StrSkill50, which is "Heals: ". In descline2, we have a Type 2 description, which means "form a string of [text a] + (calc a) + [text b]". Looking up the two string keys in String.tbl, we see StrSkill88 = "Mana Recovery Rate: " and StrSkill23 = " percent". Then in descline3 we have Type 19, which means "form a string of [text a] + [calc a] + " yards". And finally StrSkill18 = "Radius: ".

Now we want a string for hit points and a reference to the appropriate Param fields in Skills.txt. We will use the same format as already exists for mana, and for the second string we will reference StrSkill108 from String.tbl, which is " points". We won't use the remaining rows in our worksheet.

	type	text a	text b	calc a	calc b
descline1	2	StrSkill50	StrSkill108	ln12	
descline2	2	StrSkill88	StrSkill23	ln34	
descline3	19	StrSkill18		ln12	

**Table 10 – Completed Skill Description for the New Skill**

A	AC	AD	AE	AF	AG
skilldesc	descline1	desctexta1	desctextb1	desccalca1	desccalcb1
RegenHPMana	2	StrSkill50	StrSkill108	ln12	

**Figure 40 – Changing the Level Variant Description to show Healing Effect**

Enter these new values into the fields of Skilldesc.txt as shown above. Be sure you clear everything in fields shown in the last two figures for dsc2line1, dsc3line1 and dsc3line2. Save Skilldesc.txt, open a game and check your results!

A	BG	BH	BI	BJ	BK
skilldesc	dsc2line1	dsc2texta1	dsc2textb1	dsc2calca1	dsc2calcb1
RegenHPMana					

**Figure 41 - Removing All References to Synergy in the Level Invariant Description**

CA	CB	CC	CD	CE	CF	CG	CH
dsc3line1	dsc3texta1	dsc3textb1	dsc3calca1	dsc3calcb1	dsc3line2	dsc3texta2	dsc3textb2

**Figure 42 - Removing All References to Synergy in the Synergies Description**



## Exercise #11 – Stat-Altering Potions (1.10x only)

In this last exercise we will look at a technique for making potions that behave somewhat like shrines; the character will have a temporary change in a stat after consuming one. As in the previous exercise, this change uses features that are new with Patch 1.10. However, the technique of adding new items to treasure classes discussed at the end of the exercise works for 1.09x as well, and you should review and experiment with it often.

First, open misc.txt. Copy the thawing potion (row #10) to a new record at the bottom of the table. Rename it to Fire Resist Potion (or similar) in both the Name and \*Name fields. Change the code and namestr fields to 'frp'. Then scroll over to the State field and change it to 'potion\_fire\_res.' Remove the contents of cstate1 and cstate2 (do not leave spaces). In the Len field specify a length in frames that the effect will last, remembering there are 25 frames per second.

A	B	N	O	P	BA	BB	BC	BD
name	*name	code	alternategfx	namestr	state	cstate1	cstate2	len
Fire Resist Potion	Fire Resist Potion	frp	yps	frp	potion_fire_res			2500

Figure 43 - A new Potion of Fire Resistance

Change the Stat1 field to 'fireresist' and the calc1 field to 30 for +30% additional resistance. We'll add a bit more to this potion: in Stat2, change to 'maxfireresist' and put 5 in the calc2 field for \*5% to maximum fire resistance. Finally, in the spelldesc field put 1 and in spelldescstr the string you'll use in the table file: PotionFireRes. Save and close misc.txt.

	BE	BF	BG	BH	BI	BJ	BK	BL
1	stat1	calc1	stat2	calc2	stat3	calc3	spelldesc	spelldescstr
142	fireresist	30	maxfireresist	5			1	PotionFireRes

Figure 44 – Effects of the new potion

Now open states.txt and add a new record. In the state field, put 'potion\_fire\_res' and change the ID to the next number in the list. Move to the rfblue field and enter 1, this will change value in the resist fire field in your character stat panel to blue while the potion effect lasts. Now, scroll over to the Stat column, and input "fireresist". and finally in the eol column, input 0. It is important to have a visual effect for any stat boost, so the player knows when the boost is working. This is especially true of stats that do not show up on the character sheet, such as gold find and resistance to magic.

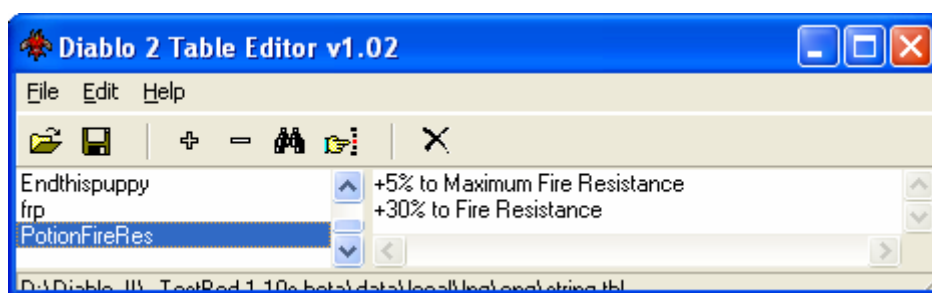
Now you have a potion that will add however much fire resistance you want, and increase your maximum fire resistance. But it has no visual representation, other than the change in the stat on the character sheet. We will give this potion the same aura as the Paladin's Resist Fire skill. Go to the overlay1 field and enter 'aura\_resistfire', then to the Castoverlay field and enter 'cast\_resistfire', and finally go to the onsound field and enter 'paladin\_aura\_resistfire'. Save and close States.txt.

A	B	R	AS	AX	AZ	BL	BT
state	id	rfblue	overlay1	castoverlay	stat	onsound	eol
potion_fire_res	182	1	aura_resistfire	cast_resistfire	fireresist	paladin_aura_resistfire	0

Figure 45 – Visual Effects While the Potion Is Working

But we still need some display strings. Using Darkstorm's String Table Editor on string.tbl, we add a string record using the '+' key as before, enter 'frp' for the key and 'Fire Resistance Potion' for the description. Then add another record with key 'PotionFireRes' and enter a two-line description in this order, '+X% to Maximum Fire Resistance', '+Y% to Fire Resistance' (where X is the number you entered for maximum fire resistance, and Y is the number you entered for the addition to fire resistance).





**Figure 46 - Two-Line Description for Fire Resistance Potion**

If you run a game in -direct -txt right now, you will find these potions will be sold by vendors that can sell thawing potions, and for 25 gold only. You must go back to Misc.txt and edit the price and availability to your preference. We will change the cost field to 500 and remove all references from the vendor fields. Akara is shown here for example (simply clear AkaraMin and AkaraMax), but you must clear all vendors that can sell the new potions if you don't want them available in town.

A	L	CC	CD	CE	CF	CG
name	cost	AkaraMin	AkaraMax	AkaraMagic	AkaraMagic	AkaraMagic
Fire Resist Potion	500					255

**Figure 47 - Adjusting Price and Availability**

Now that we have removed the potions from vendors, we need to enable them to drop from monsters and objects. This is similar to the Quill Rat Test, except we will choose a more permanent location. Load up TreasureClassEx.txt and find the records near the top labeled Potion1 through Potion6 in the Treasure Class field, those are generic potion drops. We will add our new potion to each of the six drops, using the next available slot as shown below.

	A	D	T	U	V	W	X	Y	Z	AA	AB	AC
1	Treasure Class	Picks	Item6	Prob6	Item7	Prob7	Item8	Prob8	Item9	Prob9	Item10	Prob10
21	Potion 1	1	yps	4	frp	1						
22	Potion 2	1	rws	1	yps	2	yps	3	wms	1	frp	1
23	Potion 3	1	rws	1	yps	2	yps	2	wms	1	frp	1
24	Potion 4	1	rws	1	yps	2	yps	2	wms	1	frp	1
25	Potion 5	1	yps	2	yps	2	wms	1	frp	1		
26	Potion 6	1	yps	2	yps	2	wms	1	frp	1		

**Figure 48 - Adding a New Item to an Existing Treasure Class**

This last technique is very important, as you will use it frequently to get new items dropped by monsters and objects. As said before, TreasureClassEx.txt is a very complex file, and the way it works is beyond the scope of this Guide. But the basic idea is each record represents a possible treasure drop, and the fields are what can be included in the drop. Up to ten items may be listed, each one with a probability of being selected. The Picks field is how many attempts are made from the drop record, if more than 1 then you may see up to six items in the drop. Each probability is the chance out of the total probabilities of the ten item records and the No Drops field (not shown and not counted if blank). For example, Potion1 after we edit it has 12+4+5+4+1+4+1, the last being our new item. So the total is 31 and we have a 1 in 31 chance of dropping our new potion if the Potion1 record is chosen for a treasure drop.

**Note:**

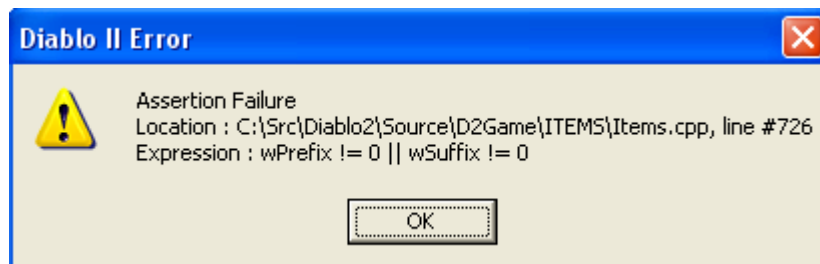
You can create *any* type of potion you want using modifiers from itemstatcosts.txt. But if you drink a second potion of the same type before the effect of the first potion wears off, the second potion will cancel the first and not add its effect.

## Conclusion

The staff at the Phrozen Keep hope you find this guide a useful introduction to the world of Diablo II mod making. There are many changes to the game possible through txt file editing alone, the exercises included here are merely the tip of a big iceberg. Use the other tutorials and file resources to learn more about the files. And remember the Phrozen Keep Forums are always open for business, should you have questions that aren't answered in these sources.

Here is a recap of some important tips for making effective and functional mods.

1. Always make incremental changes. The fewer alterations between tests, the faster you will find out what went wrong if something isn't working correctly.
2. If you add new lines to a txt file database or a string table, always add lines at the end of the file. Never insert a line, and never remove a line that you didn't add yourself. If you don't want something to display in a string table, just blank out the field on the right. There are some txt files which cannot be expanded, but of the files mentioned in this guide only LvlMaze is restricted in this way.
3. Use the Quill Rat Test to see if items spawn the way they should. If you are testing high-level items, and reducing the level and levelreq fields for testing purposes is not enough, use the first Quill Rat (H) line in TreasureClassEx.txt and test in Hell difficulty with a (generated) high-level test character.
4. Use existing similar records to see how a new record in any txt file should be organized.
5. If you crash in the game, there are two ways you can get information about what went wrong. The first way is the Assertion popup, as shown below. This message is usually very cryptic, but if you ask about it on the forum, somebody can help you better if you can describe the error using the information in the popup. The messages are easier to retrieve if you run the game windowed. Similar to -direct -txt, run the game with the -w switch for windowed mode.



Another location for finding error messages is the debug file produced each time you run the game. This file is located in the default directory, and has a name like D2020508.txt, where the name is D2 followed by two-digit year, two-digit month and two-digit day, it is the day you ran that session. Open this file, and scroll up from the end of the file until you see lines similar to

```
18:38:26.480 ***** UNHANDLED EXCEPTION: ACCESS_VIOLATION (c0000005)
18:38:26.480 Fault address: 6FD83438 01:00042438 F:\DIABLO II\D2COMMON.DLL
```

If you get an Assertion popup, the information will also appear in the debug file. And this source is the only way to find out more about the Unhandled Exception, which is a generic Windows error usually caused by Diablo II trying to access restricted memory locations.

6. Check the Phrozen Keep for the latest file guides and tutorials as you expand your knowledge. This Guide only touches the surface of what is possible.
7. *Above all else, have fun!*