



## An Illustrated Guide

Version 1.2  
by John W. Anderson

Doom Builder: An Illustrated Guide  
Copyright © 2004 by John W. Anderson

DOOM™ and DOOM II™ are trademarks of id Software, Inc.

Updated July 26, 2004

## Table of Contents

<b>Introduction: What Doom Builder Does and What You Need .....</b>	<b>iv</b>
<b>1.0 Getting Around in Doom Builder .....</b>	<b>2</b>
<b>1.1 Configuration Settings .....</b>	<b>2</b>
1.1.1 Files Tab .....	2
1.1.2 Defaults Tab.....	3
1.1.3 Interface Tab.....	4
1.1.4 Nodebuilder Tab .....	5
1.1.5 3D Mode Tab .....	7
1.1.6 Shortcut Keys Tab .....	8
1.1.7 Testing Tab .....	10
1.1.8 Editing Tab.....	12
1.1.9 Colors Tab.....	13
1.1.10 Prefabs Tab.....	14
<b>1.2 Loading a Map .....</b>	<b>15</b>
<b>1.3 The Doom Builder Interface .....</b>	<b>17</b>
1.3.1 Doom Builder's Editing Modes.....	17
1.3.1.1 DOOM Editing Nomenclature Explained .....	19
1.3.1.2 More Terminology.....	21
1.3.2 The Toolbar.....	22
1.3.2.1 Permanent Toolbar Buttons.....	22
1.3.2.2 Mode-Sensitive Toolbar Buttons .....	25
1.3.3 The Menu Bar .....	29
1.3.3.1 The File Menu.....	29
1.3.3.2 The Edit Menu .....	32
1.3.3.3 The Prefab Menu.....	34
1.3.3.4 The Tools Menu.....	34
1.3.3.5 The Scripts Menu.....	35
1.3.4 The Details Bar .....	37
1.3.5 The Status Bar .....	39
<b>2.0 Editing Basics: Making a Level with Doom Builder .....</b>	<b>41</b>
<b>2.1 Creating Sectors .....</b>	<b>41</b>
2.1.1 Line-Draw Mode.....	42
<b>2.2 Adding Sectors.....</b>	<b>44</b>
2.2.1 The Insert Sector Function.....	44
2.2.2 Joining Sectors.....	45
2.2.3 Splitting LineDefs and Sectors.....	47
2.2.4 Parent and Child Sectors .....	49
<b>2.3 Modifying Sector, LineDef, and Thing Properties .....</b>	<b>53</b>
2.3.1 The Edit Sector Selection Dialog Box.....	53
2.3.1.1 Sector Effects and Actions.....	55
2.3.2 The Edit LineDef Selection Dialog Box.....	57
2.3.2.1 SideDefs and Texture Management.....	60
2.3.2.2 The <i>Unpegged</i> Attribute.....	61
2.3.2.3 The AutoAlign Textures Function .....	63
2.3.2.4 The Visual Offset Tool.....	65
2.3.2.5 See-Through Textures.....	66
2.3.3 The Edit Thing Selection Dialog Box .....	69
2.3.3.1 Thing Sizes .....	70

<b>2.4</b>	<b>3D Edit Mode .....</b>	<b>73</b>
2.4.1	The 3D Edit Mode Environment.....	74
2.4.2	Shortcut Keys.....	74
2.4.3	Modifying Textures and Flats.....	75
2.4.4	Modifying Sector Heights and Lighting .....	78
<b>2.5</b>	<b>Making a Door .....</b>	<b>81</b>
2.5.1	Manual Doors.....	81
2.5.2	Remote Doors.....	84
2.5.3	LineDef Action Types.....	88
2.5.3.1	Local and Remote Actions.....	89
<b>2.6</b>	<b>Making a Teleporter .....</b>	<b>90</b>
2.6.1	Making a Teleport Pad in 3D Edit Mode .....	93
2.6.2	Final Adjustments in 3D Edit Mode.....	95
<b>2.7</b>	<b>Making a Lift .....</b>	<b>96</b>
<b>2.8</b>	<b>Making Rising Stairs .....</b>	<b>99</b>
2.8.1	The Mechanics of Rising Stairs .....	99
2.8.2	Creating the Space .....	101
2.8.3	Creating the Stairs .....	103
<b>2.9</b>	<b>Managing Your Map .....</b>	<b>108</b>
2.9.1	Export Node Build .....	108
2.9.2	Changing a Map's Level Number (Lumpname) .....	110
<b>3.0</b>	<b><i>Advanced Functions in Doom Builder .....</i></b>	<b>112</b>
<b>3.1</b>	<b>Making and Using Prefabs .....</b>	<b>112</b>
<b>3.2</b>	<b>Creating Custom Profiles .....</b>	<b>114</b>
3.2.1	Node Builder Profiles .....	114
3.2.2	Map Testing Profiles .....	116
<b>3.3</b>	<b>Using Alternate Texture PWADs.....</b>	<b>118</b>
	<b><i>Acknowledgements .....</i></b>	<b>120</b>
	<b>Revision History.....</b>	<b>121</b>
	<b><i>Appendix A:.....</i></b>	<b>123</b>
	<b><i>Appendix B:.....</i></b>	<b>126</b>
	<b><i>Appendix C:.....</i></b>	<b>131</b>

## Introduction: What Doom Builder Does and What You Need

**Doom Builder**, written by Pascal "CodeImp" vd Heiden, is one of the new breed of fast, full-featured DOOM editors for Windows. It is freeware, which means it doesn't cost a penny. You can download it at the Doom Builder website: <http://www.doombuilder.com>. Doom Builder will edit DOOM, Shareware DOOM, Ultimate DOOM, and DOOM II, plus HERETIC, HEXEN, STRIFE, and DOOM ports like BOOM, zDOOM, DOOM Legacy, Skull Tag, and JDOOM.

### Features

**3D EDIT MODE** • Doom Builder has a feature quite unique among DOOM editors: a real-time, fully textured and lit **3D EDITING PREVIEW MODE**. For years, level designers have had to rely on their imagination and an unholy grasp of spatial relationships in order to picture what their DOOM level would look like while editing them. The only way to know for sure was to run the level in DOOM. Doom Builder's full-screen 3D EDIT MODE (available by pressing **W**) now puts the level designer right in the middle of his level just as though he were playing the game in DOOM. This mode is perfect for aligning textures (just point the mouse cursor at a wall and press the arrow keys), changing light levels, raising or lowering floors and ceilings, modifying textures or flats, and other editing features (discussed in [Chapter 2.4](#)).

**ALTERNATE TEXTURES** • Doom Builder supports loading of texture PWADs for those who like to create or work with custom graphics.

**HIGH-RESOLUTION TEXTURES** • Doom Builder now supports 16, 24, and 32-bit PNG (Portable Network Graphics) textures and flats for games that support them, like JDOOM.

**SCRIPT EDITOR** • Available for editing games that support polyobjects like HEXEN, zDOOM, and DOOM Legacy, and others that use embedded scripts for new action types such as JDOOM.

In addition, you can choose from between three node builders – **BSP**, **ZenNode**, or **ZDBSP** – or you can plug in your own.

Doom Builder is highly configurable. In addition to customizing the interface, the advanced level designer can create his own custom profiles for node building and map testing ([Chapter 3.2](#)).

### System Requirements

#### *Minimum:*

- a P166 MHz computer with *Windows 98* or better
- a copy of the main IWAD from the game you want to edit

#### *Recommended:*

- 500 MHz CPU (1000+ MHz even better)
- Windows 2000 or XP
- 3-button wheel mouse (3D EDIT MODE)
- a video card that supports 3D acceleration (3D EDIT MODE)
- DirectX 8.1 or later (3D EDIT MODE)

The minimums suggested are my own, as Doom Builder runs fine on an old Toshiba laptop P-166MMX with only 32MB RAM. If you have problems on a system with less than the recommended requirements, however, support cannot be guaranteed. A 3-button wheel mouse is recommended because many shortcut keys for 3D EDIT MODE require a third mouse button. Also, the traditional + and - keys on the Numeric Keypad that used to be used for *Zooming* and other functions has recently been changed to the scroll function of the mouse

wheel. These shortcuts can easily be changed, but a 3-button mouse will make life easier for you.

## What This Guide Offers

None of us are born knowing how to make DOOM levels. We have to learn. This manual will teach you step-by-step, assuming only that you have some knowledge of Windows. It will also provide not only the basics for making levels, but in-depth information on Doom Builder, the shortcut keys it uses to make certain functions quicker, the jargon associated with DOOM mapping, and other information and tips you'll need to make your levels the best.

Note that this manual makes use of *hyperlinks*. Links are highlighted in [light blue](#). Click on any link to jump to a chapter, appendix, or other area of interest. Links to websites and downloadable files are highlighted and [underlined](#). The [Table of Contents](#) is hyperlinked, as well.

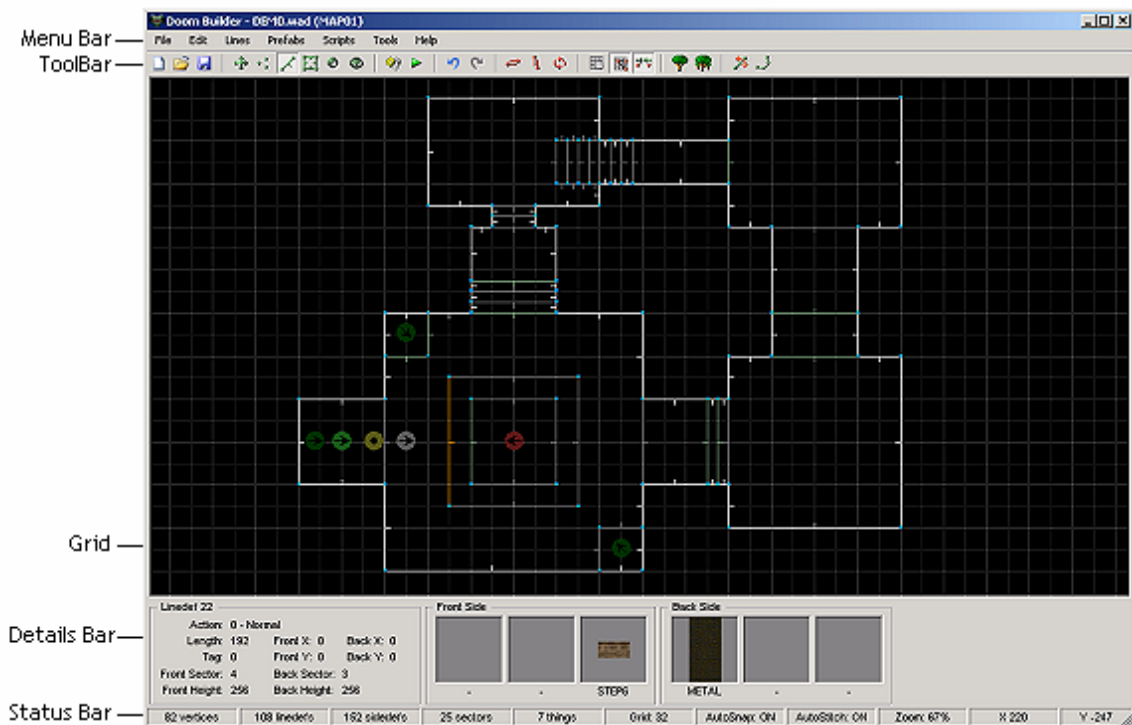
Included with this manual is DEMO.WAD, a small level that should be used as a guide for [Section 1.0](#), and which will be recreated in [Section 2.0](#). It provides the basics for learning how to use the editor – common moving structures such as doors and lifts, as well as teleporters and rising stairs.

John W. Anderson  
[drsleeper@newdoom.com](mailto:drsleeper@newdoom.com)



Double-click the Doom Builder desktop icon.

Desktop  
Shortcut



**Figure 0.1.**  
The Doom Builder interface.

SECTION

1

## Getting Around in Doom Builder

- *Configuration Settings*
- *Loading A Map*
- *The Doom Builder Interface:*

## 1.0 Getting Around in Doom Builder

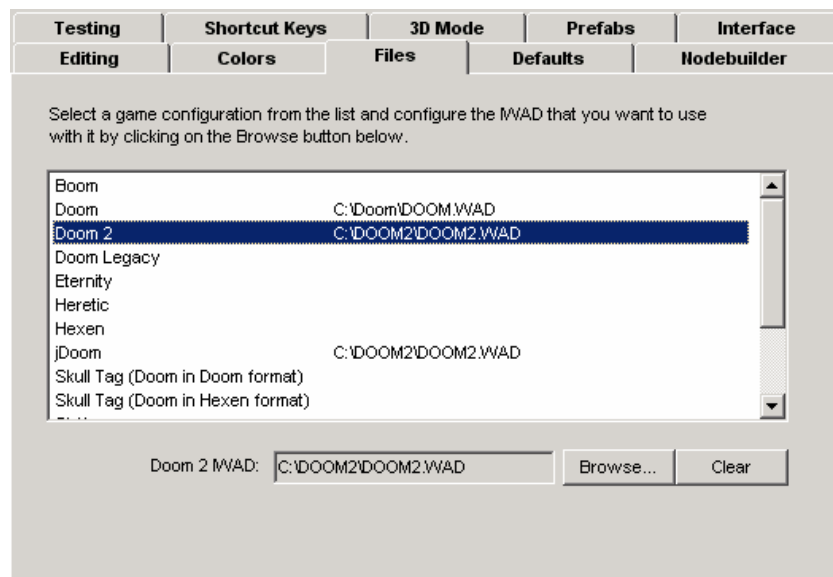
Since few people actually read introductions, I'll take the risk of repeating myself here. Section 1.0 deals in exhausted detail with Doom Builder's interface: how it looks, how it works, and how to set it up. [Section 2.0](#) actually shows you how to build a level, what buttons to click, how to move the mouse, etc. And [Section 3.0](#) covers functions that are more advanced. In order to make the actual editing lessons lively and enjoyable, [Section 2.0](#) repeats very little that is in this Section. This guide is long enough as it is. That said, you *should* read this section in full. You'll learn what the icon buttons on the toolbar do, how to save and merge maps, and all of the technical aspects of the editor's functions. But if you're anxious to start learning how to make a level with Doom Builder, you can skip to [Section 2.0](#) and jump right in. Links are provided to refer you back to the appropriate chapters in Section 1.0 for information on many of the commands and buttons you'll be using.

### 1.1 Configuration Settings

Doom Builder (DB) does most of the work for you on install, but since it is able to edit many different versions of DOOM, it wouldn't hurt to know how to point DB to where those games are located. On startup, DB may prompt you for the location of your DOOM or DOOM2 WAD, in which case it will take you directly to the **FILES** tab in the **CONFIGURATION** dialog box.

#### 1.1.1 Files Tab

Click on the **FILES** tab to see where DB looks for the main IWADs of the games it supports. The entry **Doom 2** in the list in Figure 1.0 should point to the directory where you have DOOM2 installed. If it doesn't, click on **BROWSE** and then find the directory where DOOM2 is installed and click on **doom2.exe**. Then click **OK**. You can set all of the directories for the games you plan to edit in this manner. (Normally, DB will detect the game type when you load a map. Say you load a zDOOM map: if DB has not been directed to your zDOOM directory yet, it will prompt you to do so when you load the map. This is true for any map type you load. If DB has been properly configured, it will load the map and go straight into **EDITING MODE**. So either way, you needn't worry.)



**Figure 1.0.**

Hit F5 to bring up Doom Builder's **CONFIGURATION** dialog box. The **FILES** tab is where you define your game locations.

## 1.1.2 Defaults Tab

Another area of note in the CONFIGURATION dialog box: click on the **DEFAULTS** tab. This is where you can set the **STARTUP MAPPING DEFAULTS** that apply to your level as you build (see Figure 1.1). Anytime you make a new Sector or LineDef, the default textures listed here will be used for UPPER, MIDDLE, and LOWER TEXTURES, FLOOR and CEILING TEXTURES (FLATS), as well as CEILING HEIGHT, FLOOR HEIGHT, and SECTOR BRIGHTNESS (light level). If your map is set to a particular theme where a texture is used more often than others are, you may want to change it here. Naturally, you can adjust these in your map later as you go along, but having some common defaults may save you a lot of time later.

Be sure you know the correct spelling for the texture or flat you plan to enter. If you misspell the name, DOOM will crash with the GETNUMFORNAME error when it loads your level. (That means it can't find the numeric equivalent for the texture you spelled wrong.)

**Figure 1.1.**

Set the START-UP MAPPING DEFAULTS and MAP OPEN DEFAULTS for grid properties here.

The screenshot shows the 'Defaults' tab of the Doom Builder configuration dialog. The dialog has a tabbed interface with 'Testing', 'Shortcut Keys', '3D Mode', 'Prefabs', and 'Interface' at the top. Below these are 'Editing', 'Colors', 'Files', 'Defaults', and 'Nodebuilder'. The 'Defaults' tab is active, showing two sections: 'Map Open Defaults' and 'Startup Mapping Defaults'. 'Map Open Defaults' includes a 'Grid Size' spinner set to 32, and two checked checkboxes: 'Snap to grid' and 'Stitch vertices'. 'Startup Mapping Defaults' includes several text input fields and spinners: 'Upper Texture' (BRICK6), 'Middle Texture' (BRICK6), 'Lower Texture' (BRICK6), 'Insert Thing' (1), 'Ceiling Texture' (CEIL5\_2), 'Floor Texture' (RROCK12), 'Ceiling Height' (128), 'Floor Height' (0), and 'Sector Brightness' (128).

Under the **MAP OPEN DEFAULTS**, you can set your grid properties. The default **GRID SIZE** is 32 pixels or units (valid sizes are 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024). If you like to operate at a smaller or larger grid size, change the setting and DB will always open to that size. **SNAP TO GRID** means that the Vertices on either end of your LineDefs will "snap to" the nearest grid point when you're creating or moving them around. This is desirable in order to avoid sloppy structures, so select this check box. **STITCH VERTICES** means that if you lay a Vertex on top (or within two pixels) of another, DB will merge the two. Select this check box.

The **INSERT THING** area is for selecting the default Thing to be inserted in THINGS MODE.

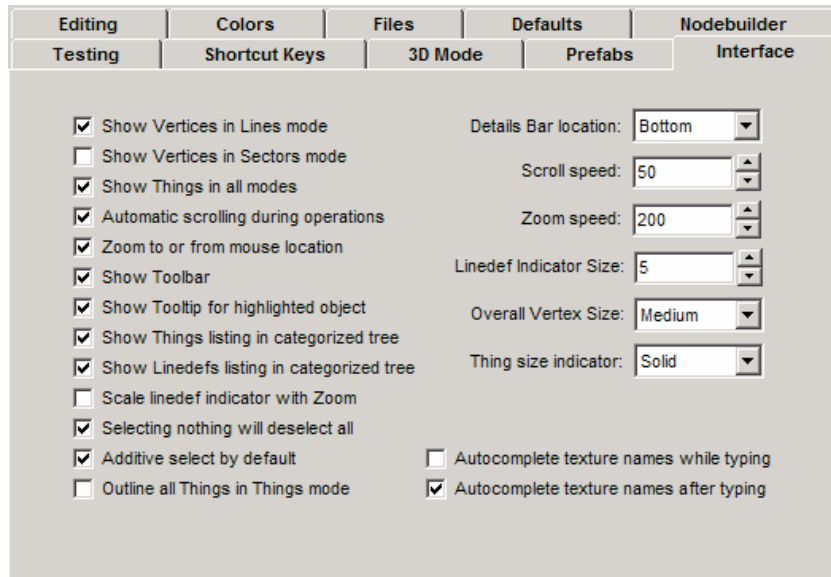
**NOTE:** Be careful you don't place a Vertex too close to another that you don't actually want to have merged. You can always click the UNDO button (or use CTRL+Z) if that happens, but decreasing your grid size and Zooming In for detail work is recommended. You can change the 2-pixel default in the editing tab, though you shouldn't increase the range.



### 1.1.3 Interface Tab

The **INTERFACE** settings allow you to show or hide certain features and sections of DB's graphical interface: map structure components, control speed settings, size and placement. Figure 1.2 shows the default settings.

**Figure 1.2.**  
Change DB's INTERFACE defaults here.



One setting you may want to change is the **DETAILS BAR LOCATION** setting. When you pass the mouse cursor over an object like a Thing, Sector, Vertex, or LineDef, information about that object's class, textures or attributes (length, tag numbers, Sector numbers, light levels, etc.) will appear in the **DETAILS BAR**, which is shown as an empty gray space beneath the grid. You can change the location so that this bar appears either on the left, right, top, or bottom. Depending on the video settings for your desktop, if you change the bar to appear on the right or left, it may not have enough room to display all of the image information. (This is the case on my laptop, which has a screen setting of 800x640; so I leave the bar on the bottom.)

Another setting to consider is **AUTOMATIC SCROLLING DURING OPERATIONS**. When this is selected, DB scrolls the grid if you move the mouse near the edges of the screen while drawing. I've found this a little hard to control, as DB will scroll quite a ways before stopping. It's a matter of preference, but I uncheck this and simply make sure I have room on the grid to build the object. The **SCROLL SPEED** can be altered in this section if you prefer to leave this option selected.

The check boxes for **SHOW THINGS LISTING** and for **SHOW LINEDEFS LISTING IN CATEGORIZED TREE** should be checked by default. This will cause the **THING TYPES** and **LINEDEF ACTION TYPES** in their respective **EDIT SELECTION** dialog boxes to be categorized by player, monster, decoration, etc., and doors, ceilings, floors, etc. respectively. Unselecting these will give you an alphabetic list that some people find tedious to sort through.

The area **SHOW VERTICES IN SECTORS MODE** is unselected by default. Please select this check box for purposes of demonstration in the tutorial section. Editing in **SECTORS MODE** doesn't require this, but it's easier to demonstrate certain operations when you can see the Vertices. You can always unselect this option at the end of the tutorial (or you may decide you like it that way).

### 1.1.4 Nodebuilder Tab

Click on the **NODEBUILDER** tab. DB comes with three node builders: *BSP v5.1* by Collin Reed, Lee Killough, and Colin Phipps; *ZDBSP v1.5* by Randy Heit; and *ZenNode v1.1.0* by Marc Rousseau. There are two main areas in this section to consider:

**QUICK NODEBUILD** is performed every time you **SAVE** your map or click the **BUILD NODES** button (or by pressing **CTRL+F8**). DB will also attempt to build the nodes when you enter **3D MODE**, and when you choose to **TEST MAP**. You may change the **WHEN TO REBUILD** area by clicking on the drop-down arrow and choosing from:

- Always rebuild nodes
- Ask me to rebuild nodes
- Never rebuild nodes

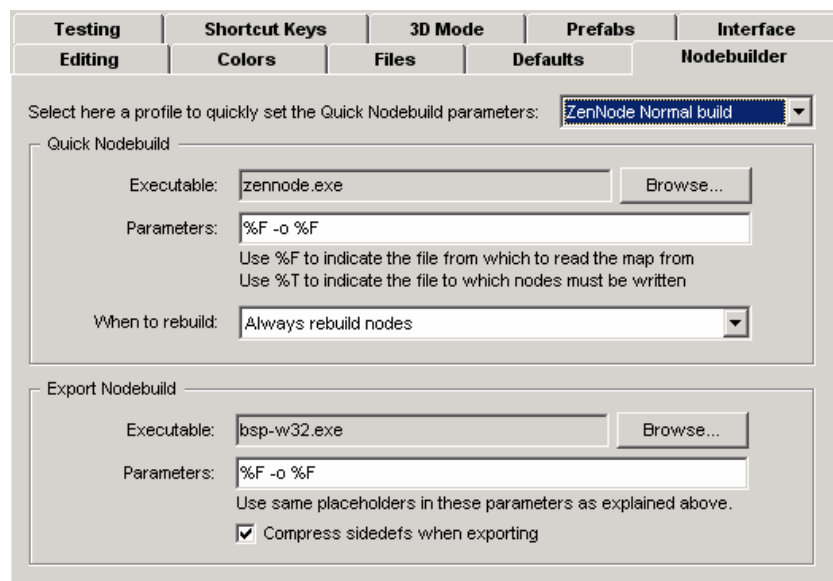
**QUICK NODEBUILD** has several **PROFILES** with preset parameters at the top of the menu. You can choose from:

- BSP Normal build
- ZDBSP Fast build (no reject)
- ZDBSP Normal build
- ZenNode Fast build (no reject)
- ZenNode Normal build

**EXPORT NODEBUILD** is used when you choose the option to **EXPORT MAP**. **EXPORT MAP** is intended for use when your level is finished and you want to do a final, quality build. The **COMPRESS SIDEDefs WHEN EXPORTING** check box is selected by default. This helps to decrease the size of your PWAD.

You may also want to perform some other actions on your map for a final save, such as compressing the **BLOCKMAP** and building the **REJECT** table; an optimized **REJECT** table greatly reduces line-of-sight calculations performed by the **DOOM** engine during game play of your level. You'll need to know which node builder does what, and which parameters to use. (See [Appendix B: Node Builder Parameters](#) for more information.)

**Figure 1.3.**  
Settings for building the nodes can be changed under the **NODEBUILDER** tab.



The current parameters and placeholders for ZenNode and BSP are **%F -o %F**.

- PLACEHOLDER: **%F** stands for the **filename** of the current map
- PARAMETER: **-o** means **output**

This tells the node builder to build the nodes for DEMO.WAD and write the *output* to the same name, DEMO.WAD. (Placeholders are used to a much greater extent in the [TESTING TAB](#) of the CONFIGURATION settings and are explained there in more detail.)

## Notes on Rebuilding the Nodes

**What are nodes?** DOOM uses a rendering algorithm based on a binary space partition, otherwise known as a **BSP TREE**. This is stored in a data lump called NODES in the WAD file. Nodes are branches in the **BSP TREE** that the DOOM engine uses to determine which walls are in front of others and can be viewed at any given time from any given viewpoint. So tools to build the **BSP TREE** are known as **node builders**. Before you can play a level that you have created, you must use a node builder to create the data that DOOM will use to render the level.

**What are the placeholders for?** When building nodes, Doom Builder writes a file with the map structures in it (THINGS, VERTICES, LINEDEFS, SIDEDEFS, and SECTORS) and calls the node builder to build the **BSP TREE** structures. When it's done, Doom Builder examines the results of the node builder to see if it actually wrote those structures. This procedure works better if you let the node builder write to a different (new) file than the one Doom Builder wrote. The PLACEHOLDER **%T** represents a different file to build to. For node builders that don't support building to a new file, you must use the PLACEHOLDER **%F**.

**Which node builder is best?** The three node builders that come with Doom Builder are the most popular and the fastest. **BSP** was the first DOOM node builder, originally released in April 1994 that – along with DEU – made DOOM level editing possible. It is a thoroughly reliable utility and continues to have updates. **ZDBSP** was written specifically for zDOOM and other DOOM ports that feature polyobject support, but works just as well on regular DOOM maps. **ZenNode** is possibly the fastest node builder of the three, can compress the BLOCKMAP, and also builds the REJECT table (quickly) by default. ZenNode is updated regularly. [Appendix B: Node Builder Parameters](#) contains a complete list of all the options and features available in BSP, ZDBSP, and ZenNode.

Another popular node builder made specifically for OpenGL DOOM ports (such as **JDoom**, **VAVOOM**, and **PRBOOM**) is **GLBSP**, written by Andrew Apted. It's based on BSP by Colin Reed, and adheres to the *GL-Friendly Nodes* specification. This means it adds some new special nodes to a WAD file that makes it easy for an OpenGL DOOM engine to compute the polygons needed for drawing the levels. If you use this node builder and would like to use it in DB, it's been tested with DB and runs fine.

Copy the Win32 version of **glbsp.exe** to the Doom Builder directory, click BROWSE to add the EXECUTABLE, and enter the following parameters:

```
-noreject %F -o %F
```

This option keeps GLBSP from clobbering the REJECT data. If you never optimize your REJECT table (shame on you if you don't), you can leave the **-noreject** option out.

You can make your own custom profiles for node building and for testing. I offer some alternate profiles for GLBSP, ZenNode and testing parameters in [Chapter 3.2: Creating Custom Profiles](#).

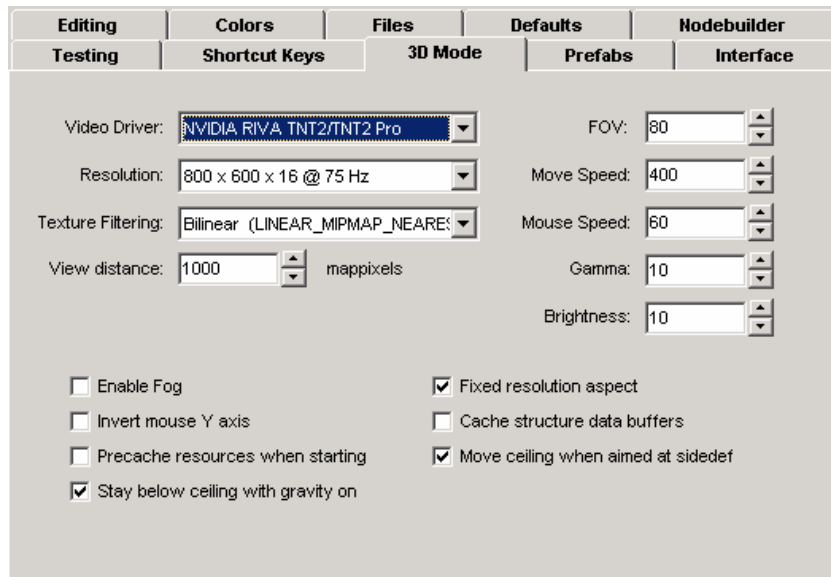
### 1.1.5 3D Mode Tab

If you click on the **3D MODE** tab, you can see the settings Doom Builder has chosen based on your video card. If you do not have a compliant video card that supports 3D acceleration, you will get a blank tab with the warning that your card does not support 3D acceleration.

In Figure 1.4, you'll see that there are a few properties you may want to change later (after you've actually gone into 3D MODE, which we'll do a bit later). FOV stands for FIELD OF VISION (or VIEW), which is set at 80 degrees by default. This controls how much of your map appears in any view. If you changed this to 120 degrees, much more of the map would be visible, but you'd be somewhat outside of DOOM's FOV, so you wouldn't be getting a realistic look at how your map actually looks in the game. I'd leave it be. Some safe things to change are your MOVE and MOUSE SPEED, plus the GAMMA and BRIGHTNESS settings. ENABLING FOG may affect your frame rate when moving about. If you feel you're already suffering a hit, you may want to lower the MAP VIEW DISTANCE, which is set by default at 3000 pixels (or grid units). If each Sector were, say, 128 units square, you'd be able to see about 25 Sectors away.

**Figure 1.4.**

Settings can be changed to alter the environment and movement in 3D EDIT MODE.

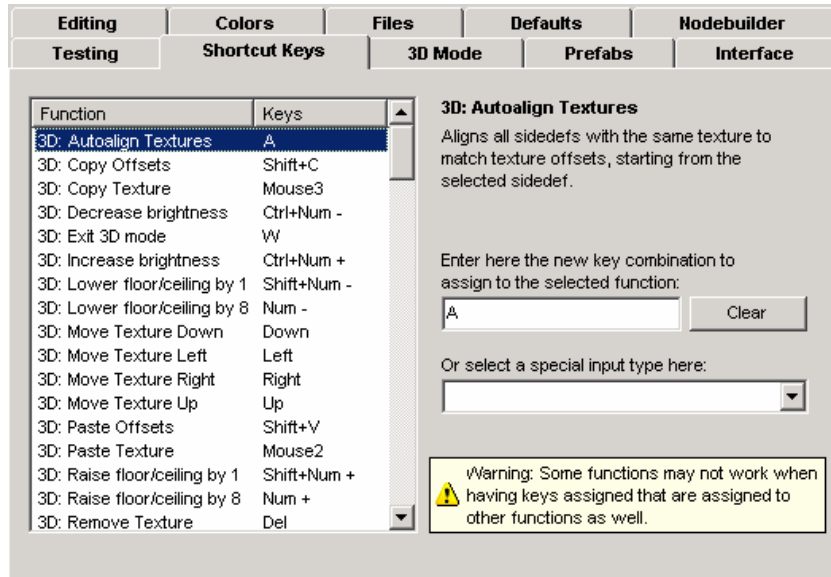


The option **MOVE CEILING WHEN AIMED AT SIDEDEF** is something you may want to select. To raise or lower a floor or ceiling height in 3D EDIT MODE, you must point and center the cross hairs on the floor or ceiling and then use the shortcut keys to change the heights. When this option is selected, you can point at a SideDef to do the same thing – so long as the SideDef belongs to the Sector you're changing. This is a handy feature because sometimes the ceiling or floor may be hard to keep in view the higher or lower it gets, especially when it's a very small Sector.

## 1.1.6 Shortcut Keys Tab

DB has many **SHORTCUT KEYS** (also called Hot Keys) to make editing actions faster. The entire list is provided in [Appendix A: Doom Builder Shortcut Keys](#) at the end of this manual.

**Figure 1.5.**  
SHORTCUT KEYS are listed here and can be modified to suit your tastes.



DB uses many of the standard Windows hot keys. Here are some of the **SHORTCUT KEYS** you'll probably use most often in 2D EDIT MODE:

Doom Builder Shortcut Keys	
<b>T</b>	Things Mode
<b>L</b>	Lines Mode
<b>S</b>	Sector Mode
<b>V</b>	Vertices Mode
<b>W</b>	3D Mode
<b>TAB</b>	Switch Modes
<b>[</b>	Decrease Grid Size
<b>]</b>	Increase Grid Size
<b>Scroll Up</b>	Zoom Out
<b>Scroll Dn</b>	Zoom In
<b>C</b>	Clear All Selected
<b>INS</b>	Insert Object
<b>ESC</b>	Cancel Current Operation
<b>DEL</b>	Delete Object
<b>A</b>	Align Textures
<b>CTRL+C</b>	Copy
<b>CTRL+V</b>	Paste
<b>CTRL+Z</b>	Undo
<b>CTRL+S</b>	Save

You may want to change the movement keys in 3D EDIT MODE, in particular. You can set the keys to move you through the map with the same configuration you use in DOOM. The *arrow keys* are reserved for aligning textures, so you should not change these. The *mouse buttons* are best used for selecting, copying, and pasting textures, so I'd leave those settings as they are, as well. You can use the mouse to move around, look up and down (mouse movement is the same as MOUSELOOK in QUAKE and zDOOM or JDOOM), but you'll probably want to set a key to move forward, since MOUSE1 will not be available for this. I personally use the following configuration:

- / Move Forward
- < Strafe Left
- > Strafe Right
- **M** Move Backward

I also use the **NUM -** and **NUM +** for *Zoom In* and *Zoom Out*, which used to be the Doom Builder default. If you're used to this instead of the new mouse wheel default, it's easy enough to change.

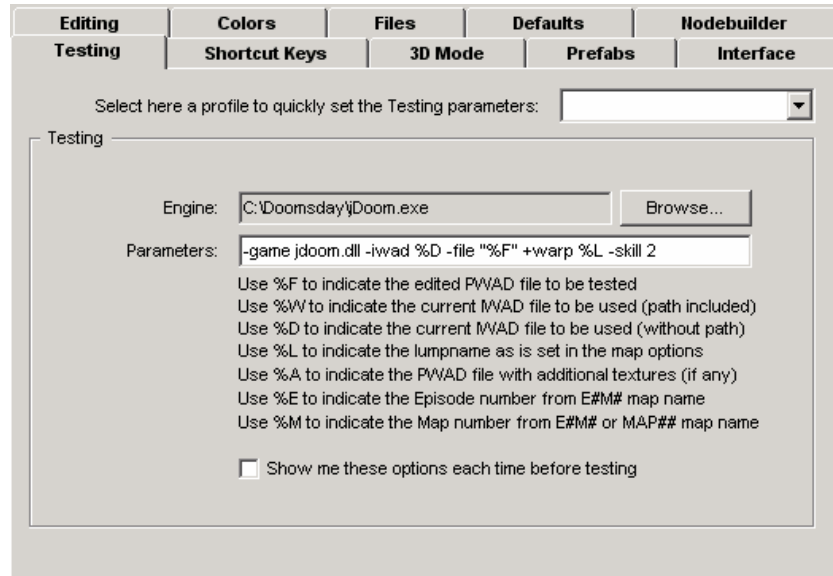
To change a shortcut key, click on the function you want to change in the list box, then in the NEW KEY COMBINATION text box merely press the new shortcut key you want to use. You don't have to type "Mouse1" – just click the left mouse button. Likewise, press the **+** key on the Numeric Keypad – don't type "NUM +".

## 1.1.7 Testing Tab

The **TESTING** tab allows you to set the parameters for launching your map in DOOM directly from Doom Builder. DB has six preset **PROFILES** to choose from which cover various skill levels in DOOM Legacy and zDOOM. DB is configured for testing in zDOOM by default; if you plan to use this feature to test your map in DOOM2 or another game, you'll need to change a few things.

**Figure 1.6.**

The **TESTING** tab is where you set the parameters for play-testing your level directly from DB.



DB uses **PLACEHOLDERS**, which are a kind of shorthand for the various map and game configurations that have already been set. In other words, DB knows which IWAD you're using, the *lumpname* (level or map number) for the map you're editing, the name of your PWAD, and the name of the texture WAD (if you should load one). So instead of typing DEMO.WAD, DB recognizes **%F** in place of the entire name.

Select the **SHOW ME THESE OPTIONS** check box if you'd like DB to prompt you for further parameters or changes before testing.

### Understanding the Placeholder Settings

- %F** represents the name of the current PWAD you're editing (DEMO.WAD).
- %W** represents the name of the current IWAD (C:\Doom2\doom2.wad) and path.
- %D** represents the name of the main IWAD (doom2.wad) without path.
- %L** represents the entry (or lump) name of the current level or map (MAP01).
- %A** represents the name of the texture PWAD you loaded (none).
- %E** represents the Episode number from an **EnMn** map (eg, DOOM1).
- %M** represents the Map number from an **EnMn** or **MAPnn** (eg, DOOM1 or zDOOM).

In order to play-test your map, you'll need to know the *command line parameters* used by the game you wish to use. Generally, they're the same for DOOM, DOOM2, HERETIC, and HEXEN

(with minor differences); but DOOM ports like zDOOM and (especially) jDOOM are a bit more involved, so I recommend you check the documentation for each game (the settings are usually defined in a FAQ) for a complete list.

Here are the settings for DOOM II and jDOOM:

**DOOM II** • Select the ENGINE executable by clicking on BROWSE and finding **doom2.exe** in your DOOM2 directory. In the PARAMETERS text box enter:

```
-iwad %D -file %F -warp %L
```

More parameters for DOOM2 can be found in the DMFAQ66B.TXT file in your DOOM2 directory.

**jDOOM** • For those of you who prefer to use jDOOM and the *Doomsday Engine*, click on BROWSE and find the **doomsday.exe** in the C:\Doomsday\bin directory. In the PARAMETERS area enter:

```
-game jdoom.dll -iwad %D -file "%F" -warp %L
```

You can find more parameters for jDOOM in the CmdLine.TXT file located in the C:\Doomsday\docs directory.

You can create custom PROFILES by editing the **PARAMETERS.CFG** file in the Doom Builder directory. Instructions for doing this can be found in [Chapter 3.2.2](#). Unless you really know what you are doing, **DON'T ATTEMPT TO EDIT THIS FILE WITHOUT READING THIS SECTION!**



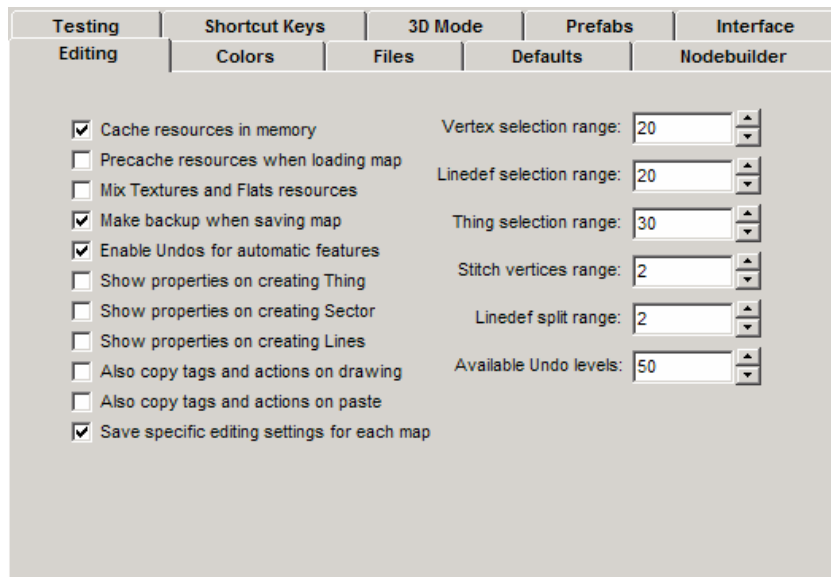
## 1.1.8 Editing Tab

The **EDITING** tab controls such settings as ENABLING UNDO and the number of UNDO LEVELS, MAKING BACKUP copies of your map while editing, the STITCH VERTICES RANGE OR LINEDEF SPLIT RANGE, and the like. The default settings are preferred (see Figure 1.7).

The check boxes for SHOW PROPERTIES ON CREATING THING should be selected by default. What this means is that every time you insert a new Thing (monster, weapon, etc.) the THING EDITING dialog box will pop up so that you can change the Thing you want to insert. By default, DB will insert a PLAYER 1 START when you've started a new map. (Afterwards, it inserts the previous Thing used.) This is something most designers want to change right away, so leave this checked.

**Figure 1.7.**

The default settings in the EDITING tab are preferred.



If you check the other two boxes, SHOW PROPERTIES ON CREATING NEW SECTOR or LINES, their editing dialog boxes will pop up every time you create a Sector or LineDef. Most designers like to create LineDefs and Sectors on the fly and go back and change their properties later, so I recommend leaving this option unchecked.

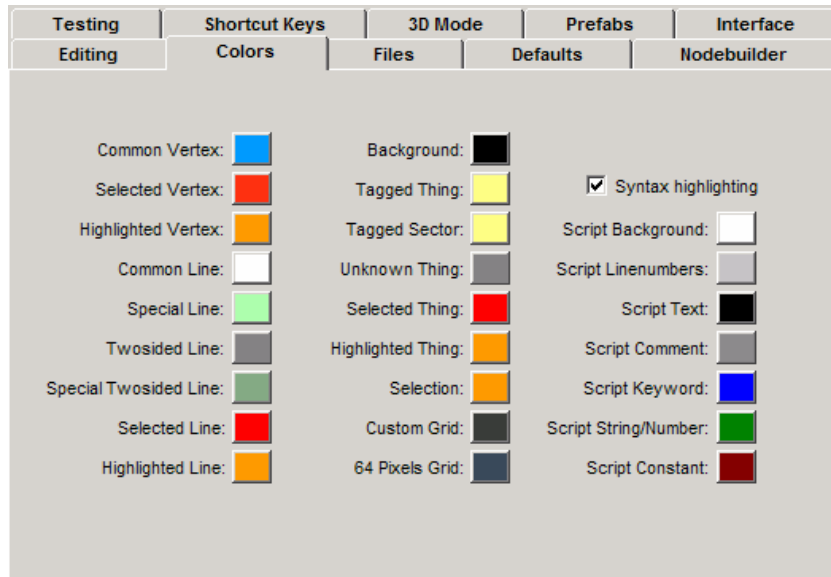
The areas VERTEX, LINEDEF, and THING SELECTION RANGE refer to the number of pixels within which your mouse cursor comes before the object is highlighted. This range is easier to control at larger zoom scales when working with structures, so the default ranges are reasonable.

## 1.1.9 Colors Tab

The **COLORS** tab allows you to change the various color coding Doom Builder uses to show various map structures such as Vertices, LineDefs, Sectors, Things, and the grid lines. Actually, I find the CUSTOM GRID lines to be a little too dark (there are two types of grid lines: what DB calls the 64 PIXEL GRID, which are the main light-gray lines you see when DB opens a map, and the CUSTOM GRID, which are the smaller lines that change when you scale down your grid below 64). I change these to a lighter gray (but don't make them too light, or they can become indistinguishable from the Lines of your map).

**Figure 1.8.**

The **COLORS** tab allows you to change grid and map structure colors.



### 1.1.10 Prefabs Tab

The **PREFABS** tab allows you to define the default folder in which to look for prefabs, plus five **QUICK PREFABS** you can set for prefabs that you may use often. (Some people create prefabs for chairs, desks, couches, etc.)

**Figure 1.9.**


The **PREFABS** tab is for defining locations of **QUICK PREFABS** THAT can then be accessed through the **LOAD PREFAB** buttons on the Doom Builder toolbar.

Editing	Colors	Files	Defaults	Nodebuilder
Testing	Shortcut Keys	3D Mode	Prefabs	Interface
Quick Prefab 1:	<input type="text" value="C:\Program Files\Doom Builder\Toilet.dbp"/>	<input type="button" value="Browse..."/>		
Quick Prefab 2:	<input type="text" value="C:\Program Files\Doom Builder\Desktop.dbp"/>	<input type="button" value="Browse..."/>		
Quick Prefab 3:	<input type="text"/>	<input type="button" value="Browse..."/>		
Quick Prefab 4:	<input type="text"/>	<input type="button" value="Browse..."/>		
Quick Prefab 5:	<input type="text"/>	<input type="button" value="Browse..."/>		
Default folder to look for Prefabs when inserting (empty for last used folder):				
	<input type="text" value="\"/>	<input type="button" value="Browse..."/>		

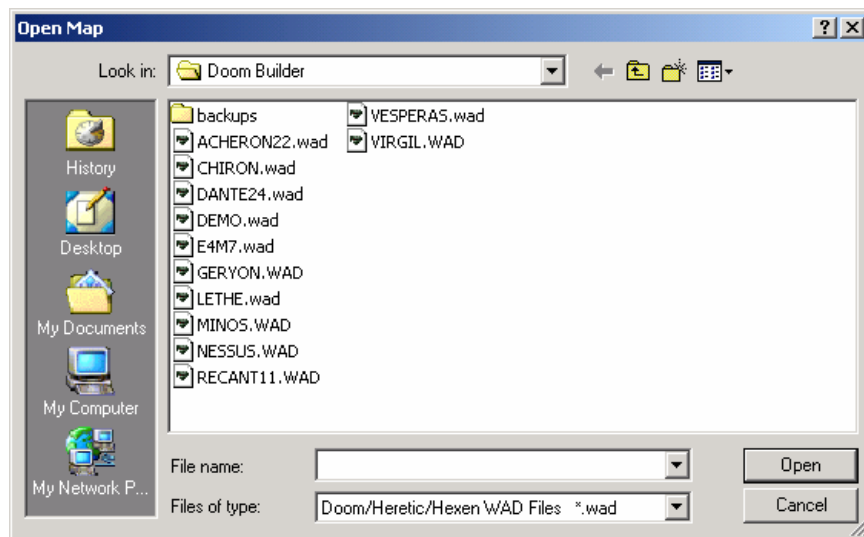
Instructions on loading and creating your own prefabs can be found in [Chapter 1.3.2.1](#) and [Chapter 1.3.3.3](#). Plus, an advanced section on creating prefabs can be found in [Chapter 3.1: Making and Using Prefabs](#).

## 1.2 Loading a Map

Let's open the DEMO.WAD map that came with this manual. It's a small level that we're going to use for display purposes as we go along – so you can get a sneak peek at what you'll be making later in [Section 2.0](#).

Click on the OPEN button  in the TOOLBAR menu or click on FILE and choose OPEN in the MENU BAR. The OPEN MAP dialog box comes up. DB will look in its own folder by default, which is where you should put DEMO.WAD. You can use the Doom Builder directory, of course, for saving levels; but it's a good idea to keep a separate directory where you plan to keep the PWADs you edit. (I have a separate Edit directory on Drive D, which is where I also keep the DOOM and DOOM2 IWADs.)

**Figure 1.10.**  
The OPEN MAP dialog box should look familiar.



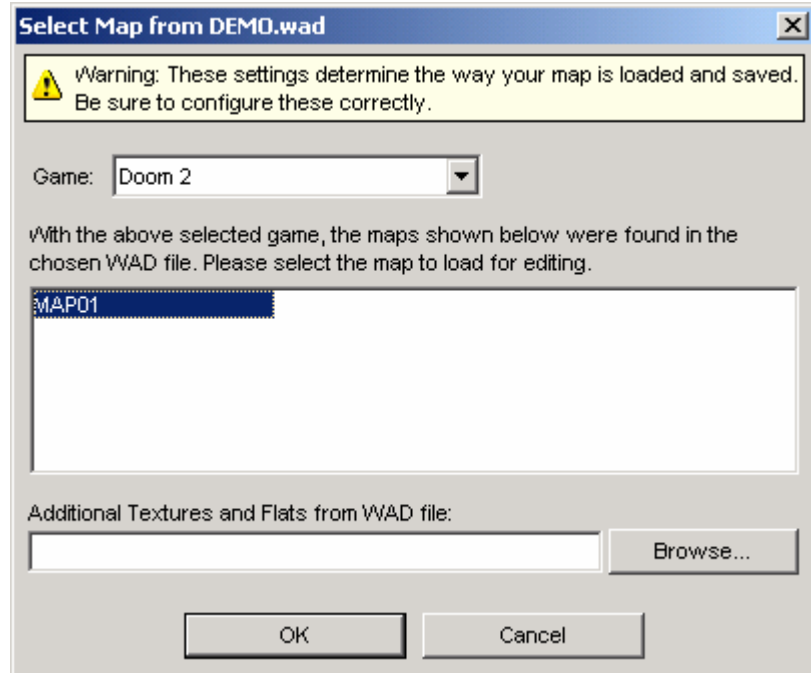
Click on DEMO.WAD when you find it, and then click OPEN.

DB will try to determine what game configuration your map is made for and will now show you the SELECT MAP dialog box (see Figure 1.11). DB is capable of loading multilevel PWADs, so this is where you would choose which map number (level or *lumpname*) to edit. DEMO.WAD has only one level. Select MAP01, then click OK.

Doom Builder will now load your map.

**NOTE:** The SELECT MAP dialog box has an area for loading an ADDITIONAL TEXTURES AND FLATS PWAD. This is for those who like to add custom graphics to their levels.

**Figure 1.11.**  
Doom Builder automatically tries to detect the game type and then displays the map(s) contained within the PWAD.

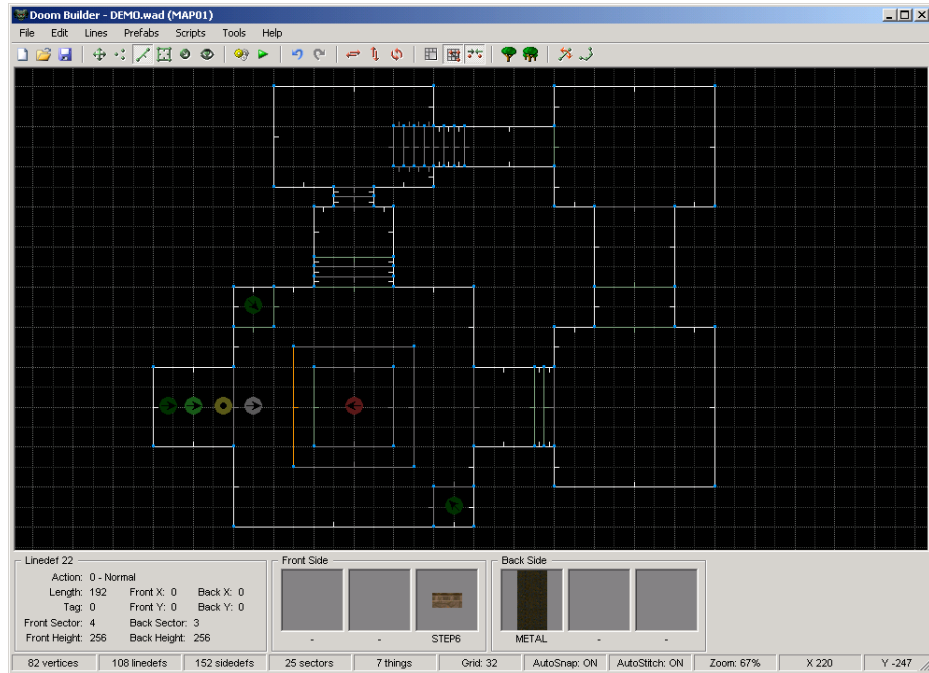


The map will be displayed against the grid backdrop with the default settings we've chosen in the CONFIGURATION dialog box. We've loaded this map to give you an idea of what to expect later when you need to reload the PWAD you've created with this manual. So we'll be looking at DEMO.WAD as we get familiar with the various functions and features of the Doom Builder interface.

## 1.3 The Doom Builder Interface

This chapter explains DB's EDITING MODES, basic terminology associated with editing and this manual, plus the various commands and functions found under the four main areas of the DB interface: the MENU BAR, TOOLBAR, DETAILS BAR, and STATUS BAR. Figure 1.12 gives you a view of the entire Doom Builder interface.

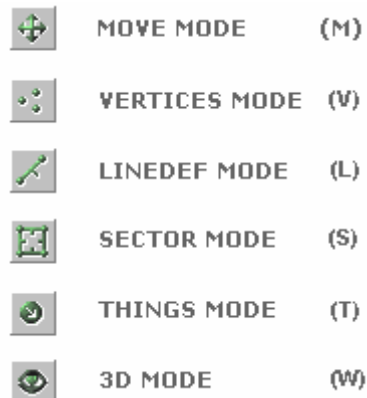
**Figure 1.12.**  
Doom Builder Interface.



### 1.3.1 Doom Builder's Editing Modes

Now that you have your grid and a map in front of you, let's talk about Doom Builder's different **EDITING MODES**.

Doom Builder has five modes of editing: **VERTICES**, **LINES** (OR **LINEDEF**), **SECTORS**, **THINGS**, and **3D MODE**. To switch between editing modes, you can hit the **TAB** key, or press the letter key for the mode you want to enter: **M** for Move, **V** for Vertices, **L** for Lines, **S** for Sectors, **T** for Things, and **W** for 3D. Or you can click on one of the editing mode buttons on the **TOOLBAR**:



**Figure 1.13.**  
EDITING MODE toolbar buttons.

None of the fundamental map structures – Vertices, LineDefs, and Sectors – can exist by themselves. No solitary Vertex exists in a DOOM map without a LineDef attached to it. And LineDefs could not exist without Vertices, as they define the *start* and *end* points of the LineDef. Likewise, Sectors are constructed of LineDefs (and SideDefs), which themselves are defined by two Vertices. And Things would have no place in which to reside if there were no Sectors. There would only be Void Space.

The various editing modes in Doom Builder are therefore rather flexible, meaning that various functions can be performed in each mode. Sectors can be created in LINEDEF MODE and SECTORS MODE. Vertices can be inserted not only in VERTICES MODE, but also in LINEDEF and SECTORS MODE. And LineDefs can be created just as easily in SECTORS MODE as in LINEDEF MODE. It's important to know this in order to appreciate the full flexibility of the editor. But don't memorize any of the above; you'll learn what can be done where in [Section 2.0](#) when you finally sit down to build a level.

You should regard each mode, however, as having fundamental uses for the object type they describe. Since these are explained in detail in [Section 2.0](#) through direct use, a brief explanation of the five editing modes should suffice to help you understand the editing terms and functions described in this section.

An object can only be edited, moved, or modified in its own editing mode.



**MOVE MODE** allows you to scroll the grid with your mouse cursor. You can also press **M**.



**VERTICES MODE** allows one to insert and move Vertices, and split LineDefs. A Vertex is inserted by pressing the right mouse button (right-click) or the INS key. To move a Vertex, right-click and hold the mouse button down and drag the object to its new position.



**LINEDEF (LINES) MODE** allows one to insert a Vertex, create a single LineDef, or an entire Sector, or to split a Sector. Doom Builder creates LineDefs and Sectors with a LINE-DRAW MODE. This works by first inserting a Vertex to which a LineDef is attached. One can draw either a single LineDef or an entire Sector. To close a Sector, you must end your drawing on the same Vertex that you started with.

LineDefs can be assigned an ACTION TYPE that dictates a function to be performed by a Sector. Doors, lifts, crushing ceilings, and rising stairs are controlled by LINEDEF ACTION TYPES, and these are assigned in LINEDEF MODE through the EDIT LINEDEF SELECTION dialog box. Textures are also assigned in this dialog in the SIDEDEFS tab, which defines the attributes of the FIRST and SECOND SIDEDEFS. The texture viewer is accessed here, and the texture X and Y offsets set here.



**SECTORS MODE** is for creating, moving, or modifying Sectors properties. LINE-DRAW MODE can also be used here to create new Sectors or divide existing ones. In the EDIT SECTOR SELECTION dialog box, Sectors can be assigned environment effects such as blinking lights and health damage. Floor and ceiling heights are modified here, as well as light levels and the floor and ceiling textures, more properly called *flats*.



**THINGS MODE** is where all Things – monsters, weapons, ammo, decorations, etc. – are placed and their attributes modified.



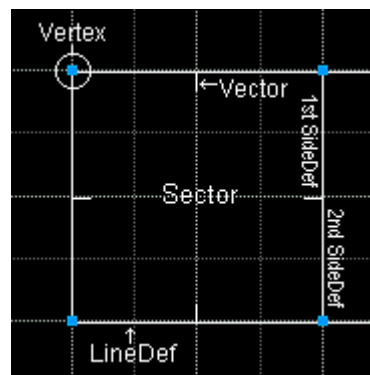
**3D EDIT MODE** is Doom Builder's most attractive feature. Nearly all cosmetic changes can be made here: texture placement and alignment, Sector heights, and brightness levels. It has value in existing simply as a means of quickly seeing what your level will look like. You can move through your level just as though you were playing it in DOOM. The 3D EDIT MODE environment is visually indistinguishable from DOOM, except that you cannot see Things, and there are no moving Sectors. You can walk through the level or "fly" in no-clip mode.

### 1.3.1.1 DOOM Editing Nomenclature Explained

For newcomers, DOOM jargon can be a bit baffling. "I know what Things are. But aren't Sectors the same thing as *rooms*? Isn't a LineDef a *wall*?" The answer is "Sort of, but not really." Even experienced level designers will speak off-handedly about rooms and walls when talking about their level: "The first room in my new level has a cool new texture on the wall at the end of the hall." But when talking about the map during the editing process, they'll revert to proper editing terminology because there are important differences between a Sector and a room, and a LineDef and a wall. ("And what the hell are SideDefs and Vectors?")

**THINGS** are enemies, weapons, keys, bonuses (such as armor and health packs), decorations (barrels, pillars, and trees), hanging bodies, dead bodies, and light sources. Things can have an **ANGLE** which determines the direction they're facing, a **FLAG** that defines whether enemies are **DEAF** or not, plus a flag for assigning which **SKILL LEVELS** the object appears in. Things properties are edited in the **EDIT THING SELECTION** dialog box ([Chapter 2.3.3](#)). They do not appear in **3D EDIT MODE** and can't be modified there.

**VERTICES** (plural of **VERTEX**) are the reference points defining the **START** and **END** of a LineDef, having an **X, Y** coordinate on the map. (Since DOOM is actually 2-dimensional, there is no **Z** coordinate.) Vertices are not drawn in DOOM, but they serve a very real purpose. As you can see in Figure 1.14, they're the hinges between LineDefs and define the coordinates for Sectors on the **X** and **Y** planes. The DOOM grid unit is based on pixels for texture-mapping purposes.



**Figure 1.14.**  
DOOM nomenclature diagram.

**LINEDefs** – or *Line Definitions* – represent lines from one Vertex to another. LineDefs may have an **ACTION TYPE** – sometimes called a *trigger* – that defines a function to be performed by a Sector (such as doors, lifts, switches, crushers, etc.) These LineDefs are activated by the player facing that LineDef and *using* the spacebar (or *use* key), or sometimes they may be walked over. LineDefs may also be assigned a **TAG** number that will cause all Sectors with the *same* TAG number to undergo the effects that the **LINEDef ACTION TYPE** dictates. TAG numbers are not Sector numbers, nor LineDef numbers: they're an arbitrary identification number assigned by the level designer. LineDefs are either one-sided and impassable, or two-sided and (usually) passable.

**SIDEDEFS** – or *Side Definitions*. Each LineDef has at least one side, usually visible, called the **FIRST SIDEDEF** – which is always the **RIGHT SIDE** – indicated by a **VECTOR**. A **VECTOR** is the little stick that juts out perpendicular to the LineDef. **RIGHT** and **LEFT** are based on the direction of the LineDef as indicated by the **START** and **END** Vertices. It can also have a second (or **LEFT**) side, called the **SECOND SIDEDEF**, if it adjoins two Sectors. (In Doom Builder, the SideDefs are also called the **FRONT SIDE** and the **BACK SIDE**.) **SIDEDEFS** actually define the boundaries in the map, and these boundaries define the borders of a Sector. An enclosed set of SideDefs is what comprises a Sector. In Figure 1.14 the LineDef on the east side of the main Sector has been labeled **1<sup>st</sup> SideDef** and **2<sup>nd</sup> SideDef** because it is shared between two Sectors. The **VECTOR** on

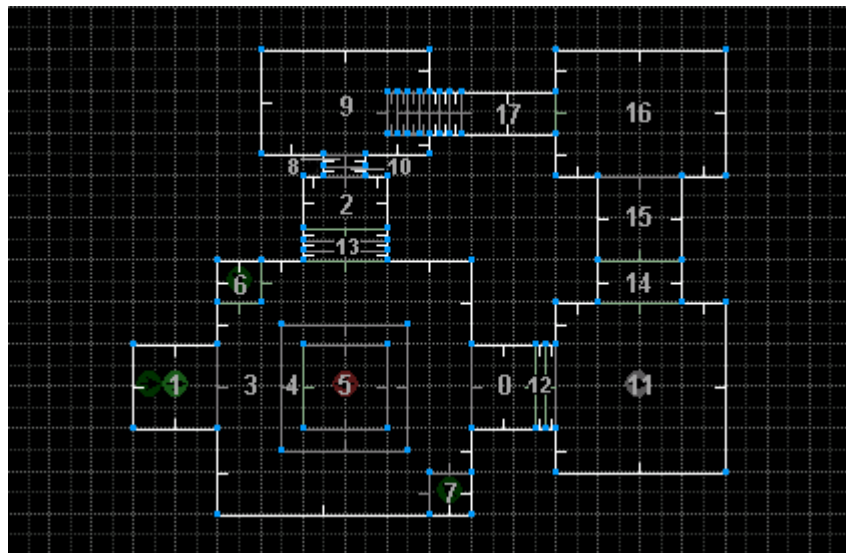


the **FIRST SIDEDEF** is facing into its Sector of origin, and the **SECOND SIDEDEF** into an adjoined, shared Sector. Textures are assigned to the SideDefs – not the LineDef.

**SECTORS** are a horizontal (east-west, north-south) area of the map where ceiling and floor heights are defined. A Sector can be thought of as a room or an area inside a room. A "room" in DOOM may have many different Sectors. A set of stairs, for instance, is composed of many adjacent Sectors of varying floor and ceiling heights. Any difference in height, texture, or light intensity requires a new Sector. Your main Sector is called a **PARENT SECTOR**, and any Sector within it a **CHILD SECTOR**. The map in Figure 1.15 shows the main Sector as being Sector 0, and **CHILD SECTORS** within it are Sectors 2, 7, 8, and 9. Sectors 1, 3, 4, 5, 6, and 10 are **PARENT SECTORS** also. Sectors are the only structures that move in DOOM. They can also have a **SPECIAL SECTOR TYPE** that describes an area-effect such as negative health (for a Sector containing toxic waste, for instance), a secret area, and certain lighting effects. Using separate Sectors for special lighting effects makes a dramatic difference to a level. Sector properties can be modified in the **EDIT SECTOR SELECTION** dialog box (see [Chapter 2.3.1](#)) or in **3D EDIT MODE**.

**Figure 1.15.**

**PARENT SECTOR** is 0. **CHILD SECTORS** are 2, 7, 8, and 9. The **CHILD SECTORS** have varying heights and light levels.



**TEXTURES** are the graphics that DOOM applies to all visible vertical surfaces. Normally, they're applied to the **FIRST SIDEDEF** of a one-sided LineDef – and thus are often called **NORMAL TEXTURES**. *But not always!* LineDefs can be two-sided, and if separated by varying Sector heights, will then have an **UPPER TEXTURE** and a **LOWER TEXTURE**, as in the case of some stair or window Sectors. This leads to the **NORMAL TEXTURE** also being called the **MIDDLE TEXTURE**, since it is placed between **UPPER** and **LOWER TEXTURES**. Textures are mapped from the *top down*, starting with the upper left corner and tiling horizontally to the right. SideDefs have **X** and **Y OFFSETS** to accommodate pasting, tiling, and alignment of the textures. Textures can be assigned or modified in the **EDIT LINEDEF SELECTION** dialog box ([Chapter 2.3.2](#)) or in **3D EDIT MODE**.

**Figure 1.16.**

Some common textures of different sizes: 16, 32, 64, and 128 wide by 128 high respectively.



**FLATS** is the name used for floor and ceiling textures. Unlike "wall" textures, FLATS cannot be manually aligned. The DOOM engine presupposes a fixed 64-pixel grid along which FLATS are automatically arranged. This means, for instance, that if you create a teleporter, you must build it on a 64-unit grid in DB. If you don't, the teleport FLATS you apply to the floor will be misaligned and look goofy. So set your grid in DB to 64 when constructing a teleport Sector. (FLATS are assigned in the EDIT SECTOR SELECTION dialog box, and in [3D EDIT MODE](#).)

**Figure 1.17.**

Some common floor and ceiling flats.



**IWAD** – or *Internal WAD* – is the main DOOM or DOOM2 data base file. It contains all the information about graphics, sound, map/level data, etc. that is necessary to play the game.

**PWAD** – or *Patch WAD* – is an external file that has the same structure as the IWAD, but far fewer entries in the directory. The data in a PWAD is substituted for the original data in the IWAD. When a PWAD is loaded, only those resources listed in the PWAD's directory are changed: everything else is loaded from the IWAD. A typical PWAD usually contains new data for a single level, in which case it would contain the 10 lumps and 11 directory entries necessary to define the level. PWADs may contain many levels, however, and may even contain new textures, graphics, and sounds. Doom Builder is able to load texture PWADs that may consist of alternate, high-resolution textures.

**OBJECT** is the generic, collective term used in this manual when referring to the specific map structures VERTEX, SECTOR, LINEDEF, SIDEDEF, and THING.

**LUMPNAME** refers to the *level number* of your map. This is the directory entry in DOOM2 PWADs that says **MAPNN** (or **ENMN** in DOOM1). This term comes up often, particularly when saving or creating a new map in the MAP OPTIONS dialog box. Don't confuse *lumpname* (MAP01) with the filename (DEMO.WAD) for your map. Lumpname is defined thus only in context with Doom Builder. A **LUMP** is just data in one of several different formats. Some contain graphics data; some contain level structure data. There are actually 10 different types of map **LUMP TYPES**, which typically appear in a PWAD. MAP01 is technically a marker under which a set of related lump formats appear: THINGS, LINEDEFS, SIDEDEFS, VERTEXES, SEGS, SSECTORS, NODES, SECTORS, REJECT, and BLOCKMAP. There are 13 other **LUMP TYPES** in the DOOM IWAD, some of which may be placed in a PWAD (such as the lumps for textures and sounds), but this requires different utilities and a deeper knowledge of DOOM than this manual covers.

### 1.3.1.2 More Terminology

In DB, when you pass the mouse cursor over an object – Thing, LineDef, Sector, or Vertex – the object changes color to **ORANGE** and its attributes appear in the DETAILS BAR at the bottom of the screen. The object is **HIGHLIGHTED**. To **SELECT** an object, **left-click** on it. The object turns **RED**. To change the attributes of an object, simply **right-click** on the object. It will turn **RED** and the EDIT OBJECT SELECTION dialog box will pop up.

To select multiple objects for mass editing, simply **left-click** on the objects. They each turn **RED** until you **right-click** to invoke the EDIT OBJECT SELECTION dialog box. You may also select entire sections of your map by holding down the left mouse button and drawing a box around the objects you wish to choose. To CLEAR selected objects, press **C** or **left-click** on the grid.

- **HIGHLIGHTED** objects are **ORANGE**
- **SELECTED** objects are **RED**

## 1.3.2 The Toolbar

The **TOOLBAR** contains icon buttons for various modes and functions in Doom Builder. As in most Windows programs, if you hold your mouse cursor over an icon, a tool tip balloon will pop up explaining what the button does. The **TOOLBAR** has eight divisions. The first seven always stay the same, while the last division changes according to the editing mode.






















**Figure 1.18.**  
The default icon **TOOLBAR**.

### 1.3.2.1 Permanent Toolbar Buttons

Figure 1.19 shows the set of icons always displayed in the **TOOLBAR**. The functions of each icon are outlined in detail below.

**Figure 1.19.**  
The **TOOLBAR** buttons and their functions.

	<b>NEW MAP</b>	<b>(CTRL-N)</b>		<b>UNDO</b>	<b>(CTRL-Z)</b>
	<b>OPEN MAP</b>	<b>(CTRL-O)</b>		<b>REDO</b>	<b>(CTRL-Y)</b>
	<b>SAVE MAP</b>	<b>(CTRL-S)</b>		<b>FLIP HORIZONTAL</b>	
	<b>MOVE MODE</b>	<b>(M)</b>		<b>FLIP VERTICAL</b>	
	<b>VERTICES MODE</b>	<b>(V)</b>		<b>ROTATE SELECTION</b>	
	<b>LINEDEF MODE</b>	<b>(L)</b>		<b>SCALE SELECTION</b>	
	<b>SECTOR MODE</b>	<b>(S)</b>		<b>GRID SETTINGS</b>	
	<b>THINGS MODE</b>	<b>(T)</b>		<b>SNAP TO GRID</b>	
	<b>3D MODE</b>	<b>(W)</b>		<b>STITCH VERTICES</b>	
	<b>BUILD NODES</b>	<b>(CTRL-F8)</b>		<b>INSERT PREFAB</b>	
	<b>TEST MAP</b>	<b>(F8)</b>		<b>INSERT PREVIOUS</b>	

Windows adepts will be familiar with the **MAP** and **UNDO** buttons, and we've already discussed the **EDIT MODE** buttons in detail in [Chapter 1.3.1](#). A few of the other icons bear further explanation:



**FLIP SELECTION HORIZONTAL** flips the selected object horizontally (just as it says) along its axis. This option is apt for Sectors and Things that stand apart from other structures. This is not the same thing as the **FLIP LINEDEF** function (explained below).



**FLIP SELECTION VERTICALLY** flips the selected object vertically along its axis. Again, good for stand-alone objects.



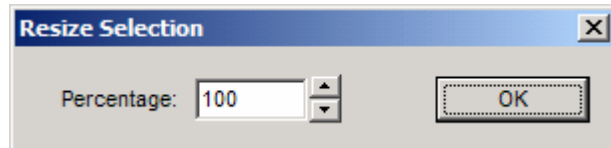
**ROTATE SELECTION** invokes the ROTATE SELECTION dialog box. Enter the number of degrees you wish to rotate the object in the text box, or click the up-down arrows for 45° increments.

**Figure 1.20.**  
ROTATE SELECTION dialog box.



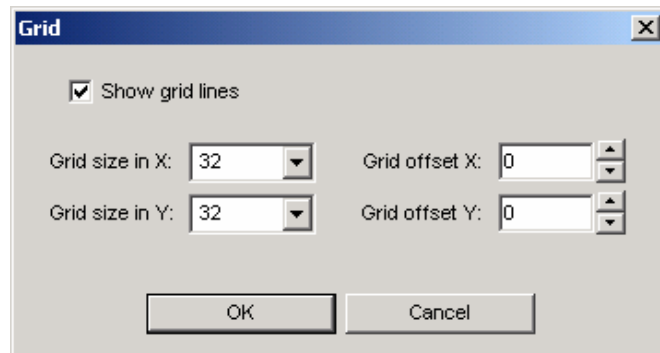
**SCALE SELECTION** invokes the RESIZE SELECTION dialogue box, which allows you to increase or decrease the size of the selected object by a percentage. You can make Sectors larger by increasing the number, or make them smaller by decrease the number. Enter the number in the text box, or click the up-down arrows for 1% increments.


**Figure 1.21.**  
Resize selection dialog box.




**GRID SETTINGS** brings up the GRID dialog box where you can adjust the default GRID SIZE. With DB, you can adjust the **X-** and **Y-AXES** to different sizes. You can also apply a **GRID OFFSET X** and **Y** by clicking the up-down arrows or entering a value in the text box. (Recall that GRID SETTINGS can also be adjusted in the [DEFAULTS](#) tab of the CONFIGURATION settings.)

**Figure 1.22.**  
The GRID dialog box allows separate adjustments to both the X and Y planes, as well as GRID OFFSETS.



**SNAP TO GRID** is on by default and the button appears depressed  in the toolbar. You can click the button to turn it off, or you can change the startup settings in the [DEFAULTS](#) tab of the CONFIGURATION dialog box.



**STITCH VERTICES** is on by default and the button appears depressed  in the toolbar. You can click the button to turn it off, or you can change the startup settings in the **DEFAULTS** tab of the **CONFIGURATION** dialog box.



**INSERT PREFAB FROM FILE** allows you to add a predefined structure to your map. Many designers often make prefab toilets, sinks, furniture, and myriad other items that they can simply call up for duty instead of having to reconstruct the same item over and over. A number of DOOM editing-specific web sites carry prefabs you can download. DB comes with two sample prefabs: a toilet and a desk. Prefabs are discussed in [Chapter 3.1](#).



**INSERT PREVIOUS PREFAB** inserts the last prefab you loaded without having to go through the selection process again. Your prefab is there at the touch of a button.

### 1.3.2.2 Mode-Sensitive Toolbar Buttons

The last section of the TOOLBAR contains buttons that appear only in certain edit modes. These buttons have a counterpart function in the MENU BAR, and may be invoked by SHORTCUT KEYS.

#### • LineDef Mode



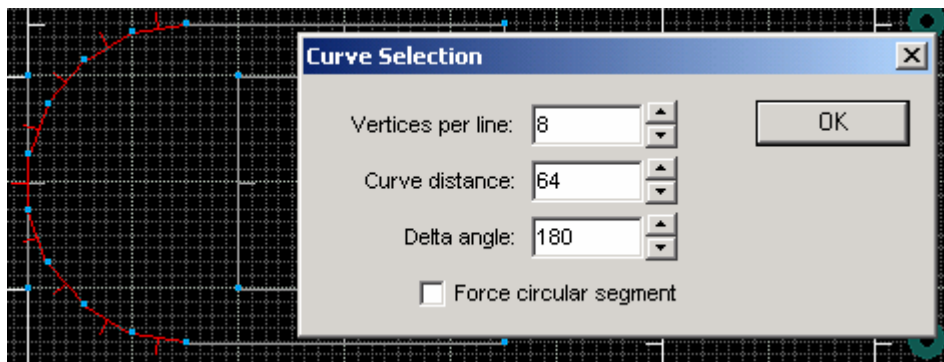
**FLIP LINEDEF** reverses the SideDefs. You will see the vector shift to the opposite side. You'll find this an extremely useful option, because many LINEDEF ACTION TYPES called *local actions* require that certain manual doors, lifts, and switches be activated from the RIGHT SIDE, or FIRST SIDEDEF. Teleporters must be entered from the FIRST SIDEDEF as well. So you'll find yourself flipping a lot of LineDefs in your level construction.




**CURVE LINEDEF** can create a nicely curved arc. Select the LineDef first (left-click), then click the button. The CURVE SELECTION dialog box comes up (Figure 1.22). Enter VERTICES PER LINE, a CURVE DISTANCE, and a DELTA ANGLE to form your curve. The curve is smoother if you add more Vertices. By default, the curve follows the FIRST SIDEDEF (RIGHT SIDE) of the LineDef. To make the LineDef curve in the opposite direction (the SECOND SIDEDEF), enter a negative number in the CURVE DISTANCE.

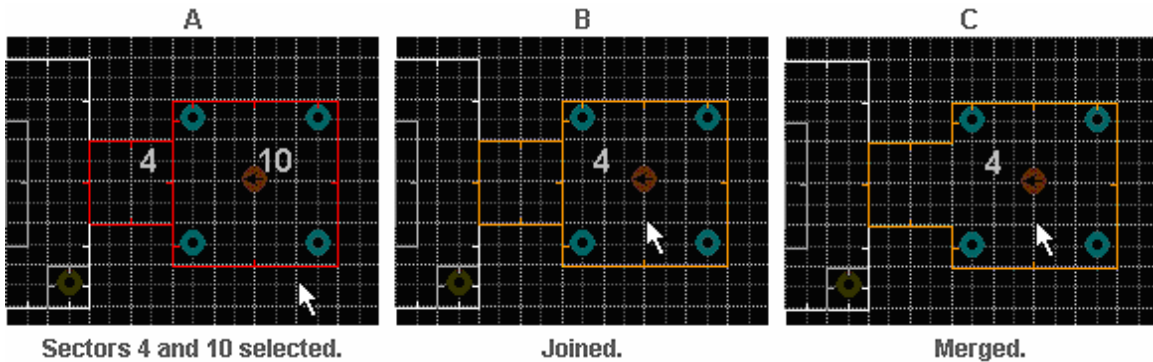
**Figure 1.23.**

The CURVE SELECTION dialog box allows adjustment of size, distance, and angle. Add more Vertices for a smoother arc.



## • Sectors Mode

 **JOIN SECTORS** will make two Sectors into one. It doesn't remove the shared LineDefs – it assigns a single Sector number to both Sectors. Say you want to join Sector 4 and Sector 10 (Figure 1.23); select first Sector 4, and then Sector 10. Click the JOIN SECTORS button. The two Sectors become one and are given Sector number 4. The **first** selected Sector (4) is retained, while the second (10) is deleted. When you highlight what was once Sector 10, both it and Sector 4 turn orange, indicating that they're a single Sector now. The DETAILS BAR will indicate your new Sector number. Sectors don't have to be adjoined or adjacent to be joined.



**Figure 1.24.**

Sectors 4 and 10 in A are JOINED in B, and MERGED in C. Note deleted LineDef in C.


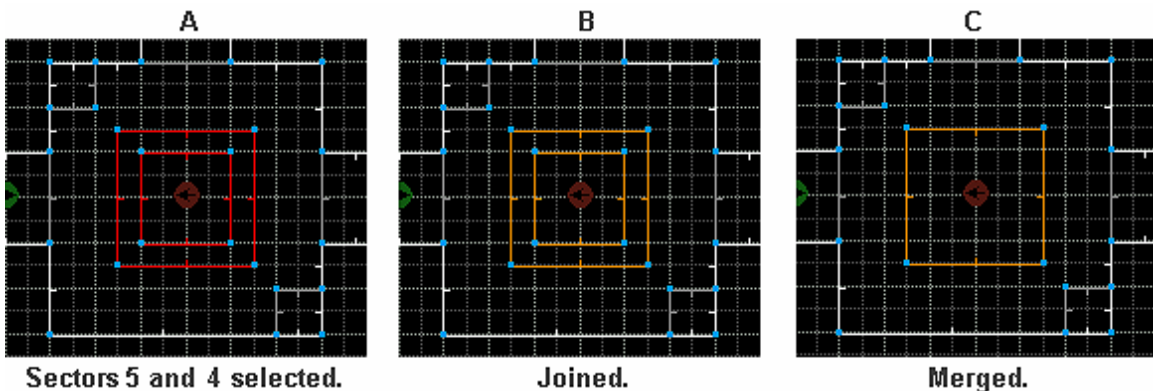
 **MERGE SECTORS** is similar to JOIN SECTORS: it deletes the shared LineDefs and makes the two adjoining Sectors a single Sector (again, retaining the **first** selected Sector).

Figure 1.24 demonstrates these two functions with PARENT and CHILD SECTORS. You would first select Sector 5, and then select Sector 4. Click the JOIN SECTORS button. The shared LineDefs (see Figure 1.24b) are not deleted, and both Sectors are now a single Sector 5.



**Figure 1.25.**

PARENT and CHILD SECTORS joined and merged.

If you clicked the MERGE SECTORS button, the shared LineDefs (indeed, the entire Sector) would be deleted (Figure 1.24c), and the new Sector would be Sector 5.

The **GRADIENT** feature evenly distributes a smooth series or grades of the brightness level, or floor or ceiling heights along a series of Sectors. You must select at least three Sectors. Only the Sectors between the first and last selection will be adjusted. For instance, say you've created a set of 4 stairs going from Floor Height 0 to Floor Height 64. Instead of selecting each stair Sector and changing the floor and/or ceiling height, you can select the lowest Sector, then the stair Sectors, and then the highest Sector, and DB will distribute the heights evenly. The order in which you select Sectors is important. The first and last selected Sectors will not be adjusted: these are used only as markers.



**GRADIENT BRIGHTNESS** will evenly grade and distribute the brightness levels of the Sectors selected between the first and last Sectors.




**GRADIENT FLOORS** will evenly grade and distribute the floor heights of the Sectors selected between the first and last Sectors.



**GRADIENT CEILINGS** will evenly grade and distribute the ceiling heights of the Sectors selected between the first and last Sectors.

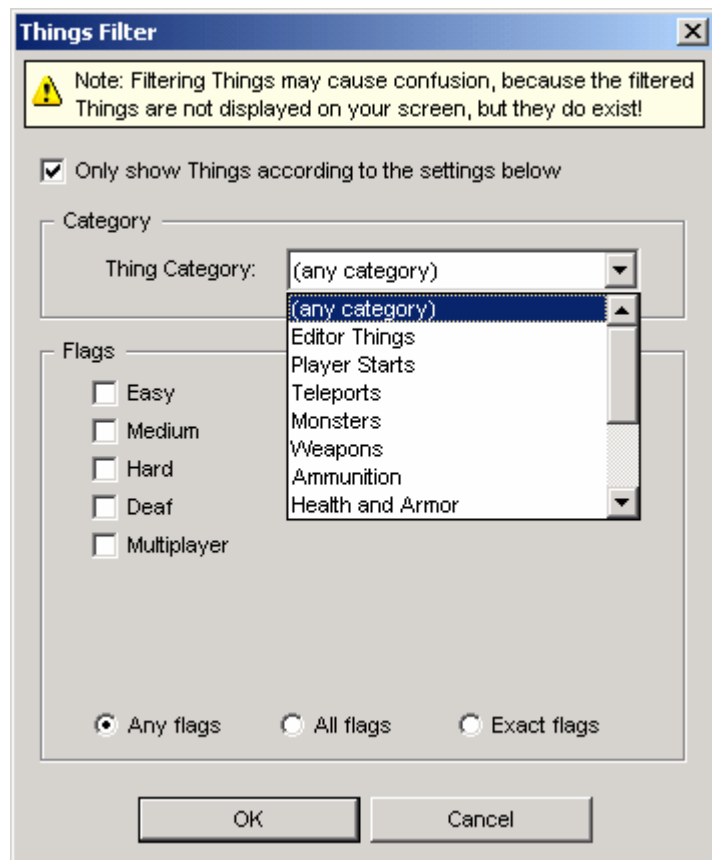


## • Things Mode

 **THINGS FILTER** is used to show only certain categories of Things in your map. Click the button to access the THINGS FILTER dialog box (Figure 1.25). Select the check box next to ONLY SHOW THINGS ACCORDING TO THE SETTINGS BELOW and then in the CATEGORY area click on the THING CATEGORY drop-down list box and choose MONSTERS. You must also select one or more FLAGS to sort the monsters by. The ANY FLAGS button will show monsters having any of the flags whose check boxes you've selected. The ALL FLAGS button will only show monsters having all of the flags selected. And EXACT FLAGS will show only those having the exact same flags as selected.

**Figure 1.26.**

The THINGS FILTER can be set to filter out all but a specified thing category by their flags.



The THINGS FILTER is an extremely handy feature for making sure your map has a good balance of health and ammo. If you often implement skill settings – as well as multiplayer settings – the THINGS FILTER is indispensable for keeping track of monsters and items in various levels.

### 1.3.3 The Menu Bar

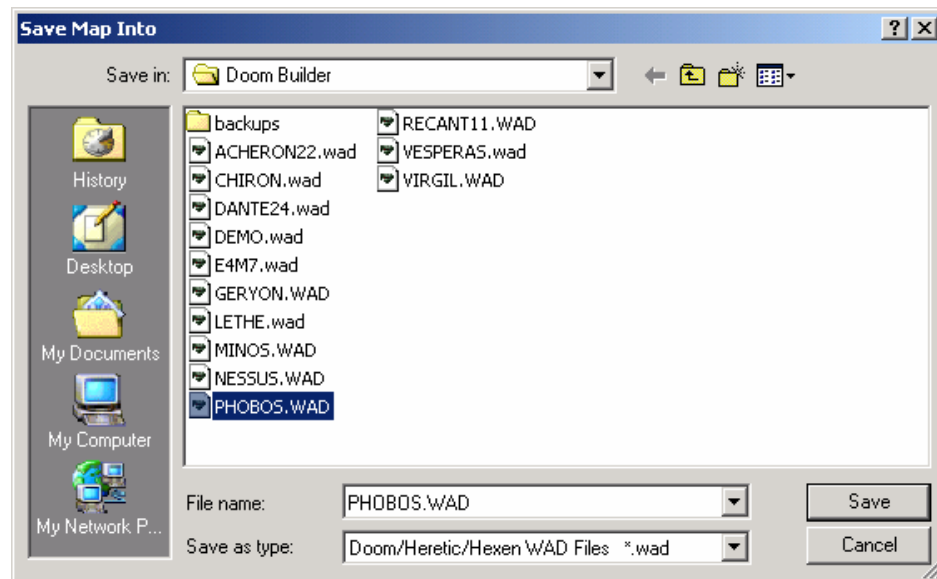
Nearly all of the functions on the MENU BAR are available either on the TOOLBAR or by using a SHORTCUT KEY. But since you won't have memorized all the shortcuts yet, we should go through the menus just so you have an idea where to find certain tasks. There are seven menus, the third of which changes according to the edit mode you're in. The SCRIPTS MENU only appears for certain game types such as HEXEN, zDOOM, Skull Tag, and jDOOM. The other five – FILE, EDIT, PREFABS, TOOLS, and HELP – retain the same commands, though some are not available in certain modes.

#### 1.3.3.1 The File Menu

Among the standard file management commands such as OPEN MAP, NEW MAP, CLOSE MAP, etc., the **FILE MENU** also lists the most recent eight maps you've opened previously, so you can click on any one for a quick load. Some important functions on this menu bear explanation.

**SAVE MAP INTO** will merge your current map into an existing PWAD that may contain one or more maps or levels. If the new PWAD already has a map with the same *lumpname* (map number) as your current map, it will be overwritten, so be careful. Your map should be in finalized form before you merge it with another, so you should do an EXPORT BUILD first. Then choose SAVE MAP INTO from the file menu, which will bring up the SAVE MAP INTO dialog box. Choose the PWAD you want to save your map into, and click OK.

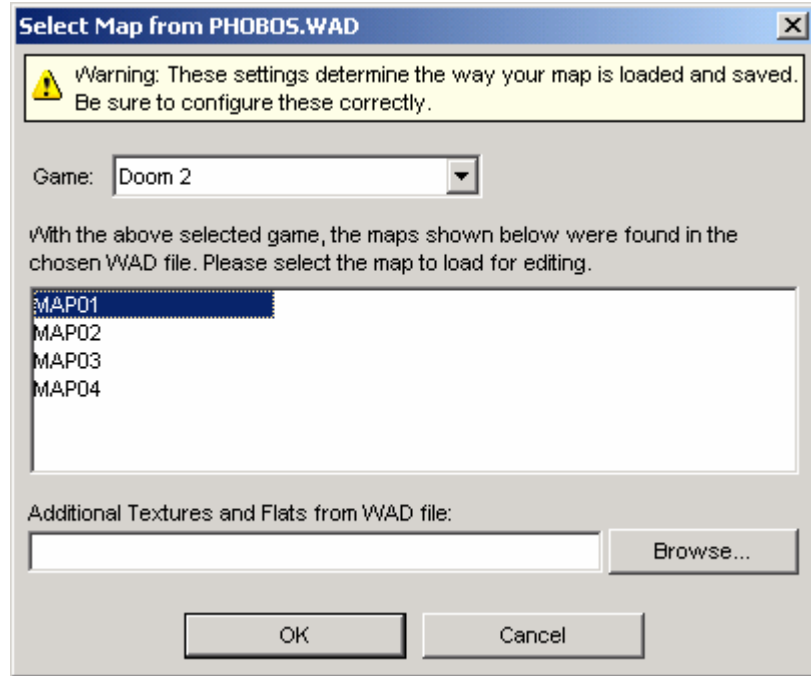
**Figure 1.27.**  
The SAVE MAP INTO command merges your current map with an alternate PWAD.



**NOTE:** If the PWAD you're saving into contains a map with the same *lumpname* as the current map (MAP01 in the case of DEMO.WAD), it will be over-written.

To test your map to see if it worked, save and close your current map, then load the PWAD you just saved into. It should then bring up the SELECT MAP FROM dialog box with a list of lumpnames (map numbers) to choose from. For instance, if your level was MAP01 and you saved into a PWAD containing MAP02, MAP03 and MAP04, all four maps should be listed.

**Figure 1.28.**  
Choose the map you wish to edit from a multilevel PWAD.

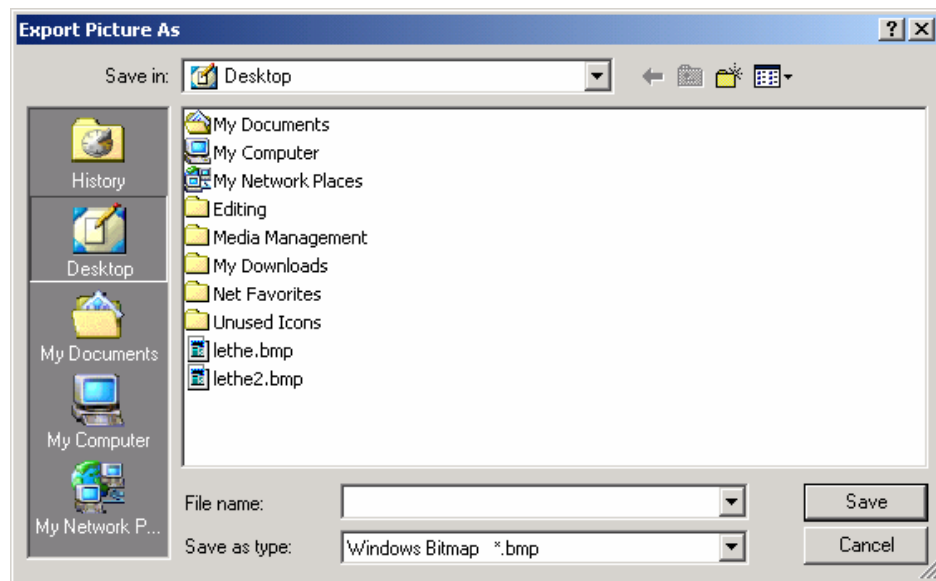


Naturally, you just click on the map you want to edit. Remember that when you're editing a map from a multilevel PWAD, only the map you loaded is updated when you save. The other maps are left as they were and the node builder function doesn't affect them.

**EXPORT MAP** is used for saving a final version of your map. When you **EXPORT MAP**, DB will also do a final node build, using the settings for **EXPORT NODEBUILD** defined in the **NODEBUILDER** tab of the **CONFIGURATION** settings. You can read more about this in [Chapter 2.9: Managing Your Map](#).

**EXPORT PICTURE** allows you to create and save a BMP, JPG, or PNG graphic image screenshot of your map structure. If you choose this option, the **EXPORT PICTURE AS** dialog box comes up prompting you for a name to save your file to.

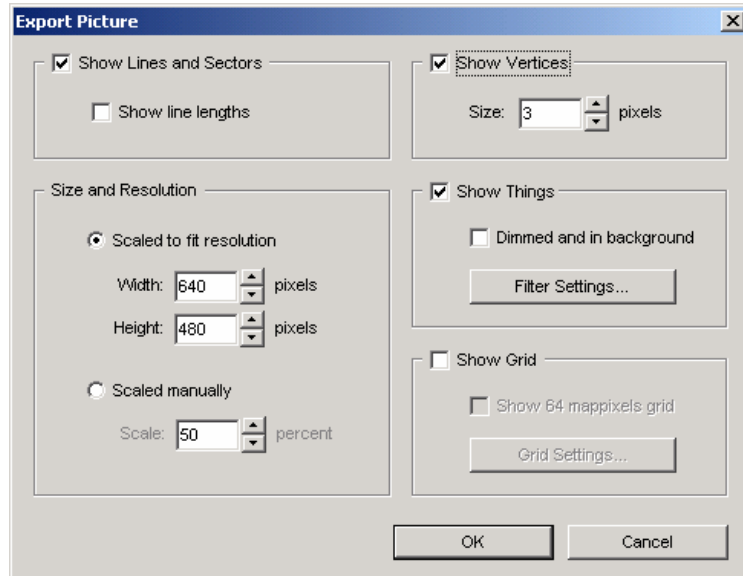
**Figure 1.29.**  
Choose a name for your exported image file.



Once you enter the filename and click OK (after first choosing a directory or folder where you would like to save the image file), the EXPORT PICTURE dialog box comes up with a host of options and features you can choose from to manipulate the way your map image looks.

**Figure 1.30.**

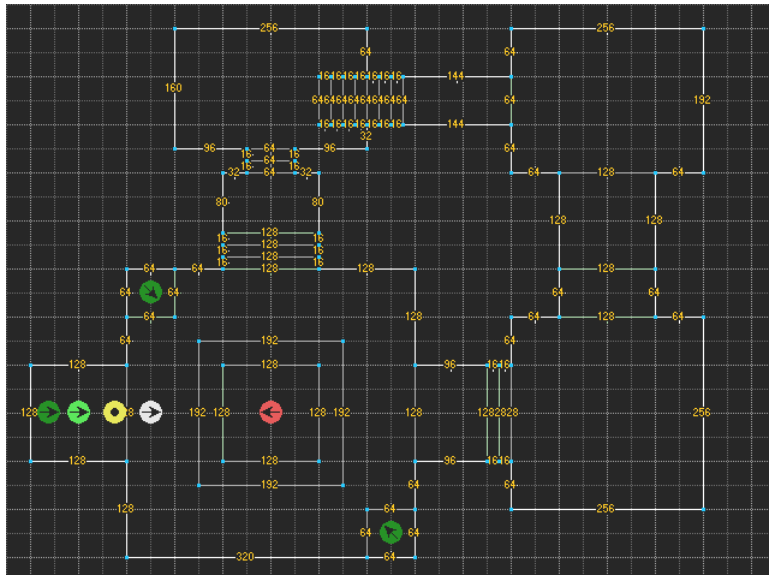
The EXPORT PICTURE dialog box has a host of ways to manipulate your final image.



**SHOW LINES AND SECTORS** area will do just that. Your image will include all of the LineDefs that comprise the Sectors of your map. If **SHOW THE LENGTHS** is checked, each LineDef will have a small orange number over the LineDef, but these are all jumbled together and unreadable if you have a large map with some very short LineDefs.

**Figure 1.31.**

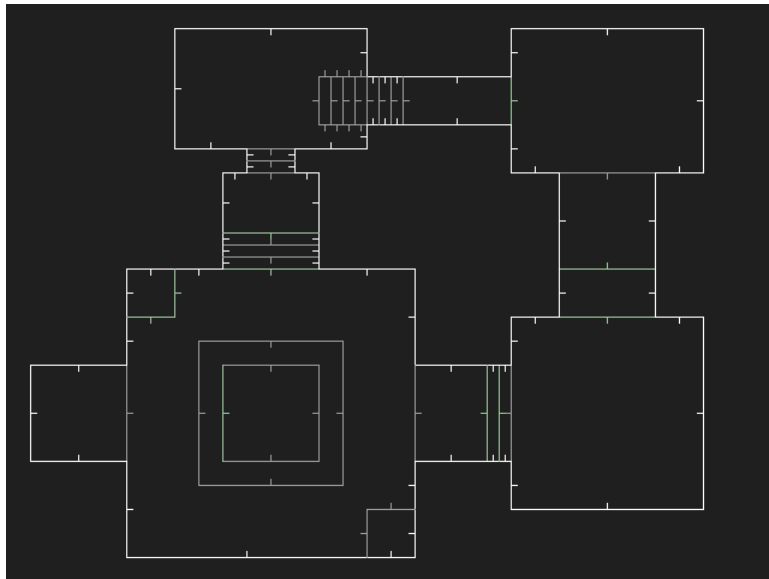
This is an exported image (scaled by half) with LineDef lengths shown.



**SHOW VERTICES** does just that, showing them as light blue. Clicking the up-down arrow buttons in the SIZE area increases the PIXEL SIZE by two-increments. The default seems to work best.

**SIZE AND RESOLUTION** currently creates an image that is **SCALED TO FIT THE RESOLUTION** to show below: **WIDTH 640 PIXELS** and **HEIGHT 480 PIXELS**. These can be changed if you want larger images. **SCALE MANUALLY** allows you to adjust the scale of the image by a percentage.

**Figure 1.32.**  
Exported picture without grid lines or vertices or LineDef lengths.



**SHOW THINGS** can show all the items in your map with the usual bright markers, or with markers **DIMMED AND IN THE BACKGROUND**. You can also use the **FILTER SETTINGS** to filter out certain items.

**SHOW GRID** will do just that, and you can adjust the grid size you'd like to have shown.

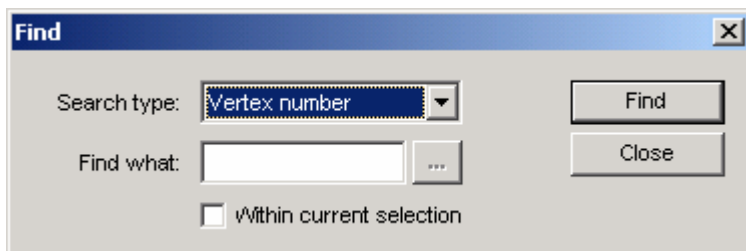
When you're happy with your settings, click OK. Experiment with different settings to get the image you like best.

### 1.3.3.2 The Edit Menu

The **EDIT MENU** contains the basic commands for undo, edit modes, copy and paste, flip and rotate, and some grid settings that are available on the **TOOLBAR** and through **SHORTCUT KEYS**. But there are a few commands here that are exclusive features of Doom Builder and extremely handy.

**FIND** allows you to search for a wide variety of different components of your map. DB's **FIND** feature is extremely sophisticated compared to other editors, which usually only allow searches for objects by their number. With DB's **FIND**, you can search for object numbers, textures, flats, actions, effects, types, and tag numbers. Let's try it.

**Figure 1.33.**  
The **FIND** command is broad and thorough. Use it to search for multiple object attributes.



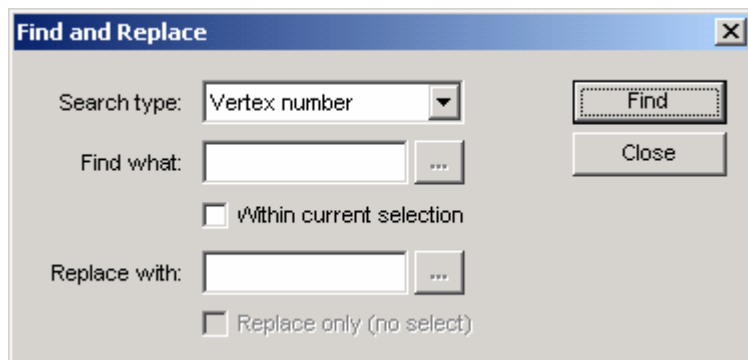
Choose **FIND** from the **EDIT MENU**. The **FIND** dialog box comes up. Click on the **SEARCH TYPE** drop-down arrow and choose **LINEDEF TEXTURE** from the drop-down list. The **ELLIPSES** button in the **FIND WHAT** area becomes active. Click it. The **SELECT TEXTURE** dialog box comes up showing all of the available textures. Scroll down the images until you find **BRICK6**, click on it, and then click **SELECT**.

DB will now highlight in red all of the LineDefs with **BRICK6**. The **FIND** dialog box also displays a message at the bottom of the box showing you the number of results found.

You may also type the name of a texture or flat in the **FIND WHAT** text box, or enter the number (if you know it) of the attribute you're searching for. Click on **FIND**. The **ELLIPSES** button will invoke a particular dialog box for the attribute you're searching for.

The **WITHIN CURRENT SELECTION** check box is for looking within a limited section of the map that you've selected by drawing a box around it with your mouse (position the cursor, then hold the left mouse button down and begin drawing). Select the check box to use this option.

**FIND AND REPLACE** is a feature we're familiar with in most word-processing programs, but in a level editor, it's a remarkable feature. You can quickly make texture and flat changes without having to manually edit every LineDef or Sector. Likewise, you could quickly replace Imps with Demons, Stimpacks with Medikits, and so on. As with **FIND**, simply choose the attribute in the **SEARCH TYPE** drop-down list, then enter the name or number in the **FIND WHAT** text box (or use the **ELLIPSES** button to invoke a dialog box to select the attribute). In the **REPLACE WITH** text box, enter the name or number of the attribute you want to substitute (or use the **ELLIPSES** button).



**Figure 1.34.**

The **FIND AND REPLACE** command allows mass replacement of textures, flats, and other attributes that would be slow and tedious through individual editing.

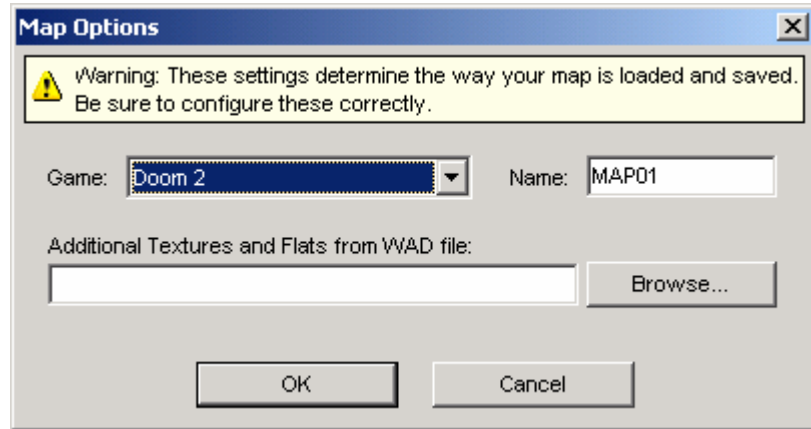
**REPLACE ONLY (NO SELECT)** means that only a replacement will take place. No results will be displayed in your map. Select this check box if you wish to implement this option.

Of the **SEARCH TYPES**, only *object* numbers cannot be replaced. For instance, Sector 10 cannot be changed to Sector 9. These are vital placement numbers. Don't confuse this with an object's type number, tag number, effect or action number. These are non-vital attributes that have nothing to do with an object's placement. It is possible to change these vital numbers in the **EDIT OBJECT SELECTION** dialog boxes (discussed in [Chapter 2.3.2](#)), but it isn't recommended unless you absolutely know what you're doing. (Sometimes a map may contain errors, such as unclosed Sectors, and these can be corrected by manually changing a SideDef's Sector reference. This is also what you would use to create a pillar.)

**MAP OPTIONS** is for changing a map's *lumpname* (level number), game type, or texture/flats PWAD. These settings determine the way DB loads and saves your map, so be careful when changing any of these. The **NAME** area refers to the map's *lumpname* – that is, the *level number* entry in the PWAD directory such as **MAP01** or **E1M1**. Don't confuse this with the *filename* of your map. If you want to change the level number of your PWAD, this is where to do it.

**Figure 1.35.**

MAP OPTIONS such as game type, map *lumpname*, and texture/flat PWADs can be changed here.



### 1.3.3.3 The Prefab Menu

The **PREFAB MENU** contains commands for loading predefined QUICK PREFABS, prefabs from file, or the previous prefab. Two exclusive commands here allow you to save your map or certain sections of your map as a Doom Builder prefab (explained in detail in [Chapter 3.1](#)).

**SAVE SELECTION AS PREFAB** will save a section inside your map as a DB prefab file, which can then be quickly loaded later as a predefined prefab. If your map contains an object that you'd like to replicate for quick insertion – instead of having to rebuild it over and over again – this is the perfect use for this option. Prefabs can be just about anything, so long as the object is itself a closed Sector. Tables, chairs, toilets, vehicles, and the like are the most common objects you find in maps, but don't let this hinder your imagination.

Select all the Sectors or LineDefs of the object you wish to save (remember that you can select mass objects by drawing a box around them), choose SAVE SELECTION AS PREFAB from the PREFABS menu, choose a directory to store your file, then enter a name for your prefab in the text box of the SAVE PREFAB FILE dialog box. DB gives the file the extension .DBP.

**SAVE MAP AS PREFAB** will save the entire contents of the screen as a prefab. No selection is necessary.

### 1.3.3.4 The Tools Menu

The **TOOLS MENU** contains commands for error checking your map and accessing the CONFIGURATION settings.

**REMOVE UNUSED TEXTURES** will strip unnecessary textures from their SideDefs if they're not visible or two-sided. Be careful you don't remove normal textures you may have placed on purpose, such as bar textures in windows. DB requires that you select an area of your map which is safe to check, and will display a reminder message if you've made no selection.

**FIX MISSING TEXTURES** will search for visible SideDefs requiring textures. Be careful not to use this function if you have purposefully left textures off SideDefs for special effects reasons.

**FIX ZERO-LENGTH LINEDEFS** will search for invalid LineDefs and remove them.

### 1.3.3.5 The Scripts Menu

Doom Builder has an extra MENU BAR category called **SCRIPTS**. The **SCRIPTS MENU** is available only when a PWAD is loaded for the game types HEXEN, zDOOM (HEXEN or DOOM in HEXEN Format), Skull Tag (DOOM in HEXEN Format), and jDOOM. The PWADs for these game types may contain special lumps in the PWAD aside from the usual ten lumps (and eleven directory entries).

There will be two or more additional lumps in a HEXEN-format PWAD:

BEHAVIOR  
SCRIPTS

If DB detects these lumps, it will examine the script and determine the game type you want to edit.

For jDOOM, the additional lumps may be any one of these three:

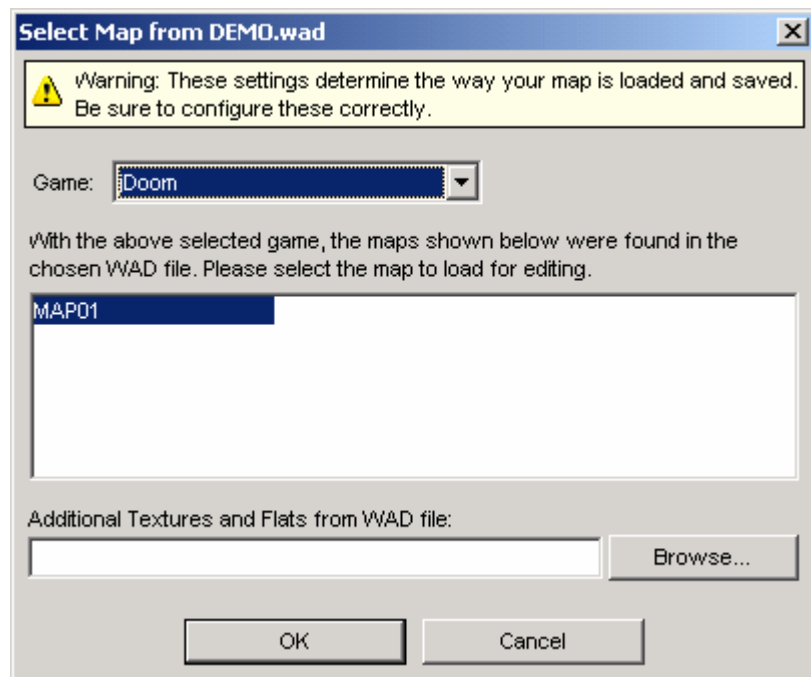
DEHACKED  
DD\_DEFNS  
DD\_DIREC

These lumps do not have to be in your PWAD in order to access the script editor. In fact, you can use Doom Builder to create one of these three directory entries for your jDOOM map if does not already exist. If you're editing a map you want to prepare for jDOOM, you would need to define that in Doom Builder when you open an existing PWAD or create a new one. For example:

- 1 Open the PWAD you want to add a jDOOM script to. The SELECT MAP FROM dialog box comes up.

**Figure 1.36.**

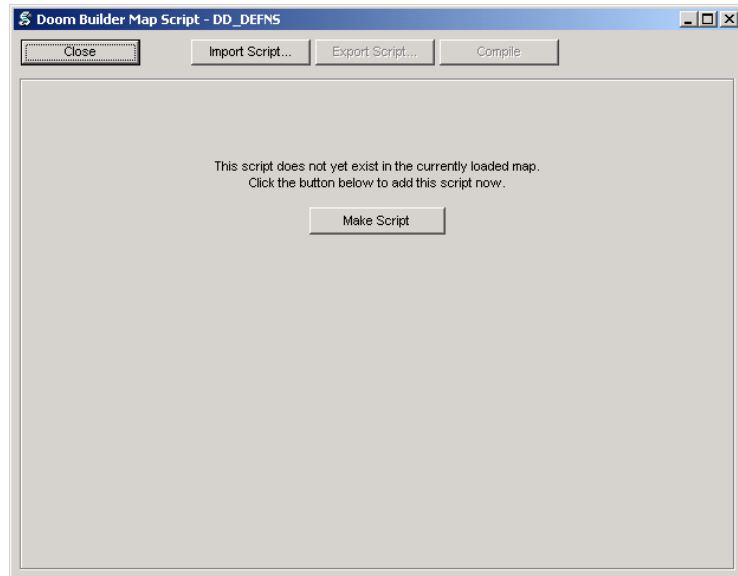
Click the GAME drop-down arrow and choose jDOOM from the drop-down list.





- 2 Select JDOOM as your GAME TYPE in the drop-down list (Figure 1.35). DB will continue to load the PWAD and the SCRIPTS MENU will now be available.

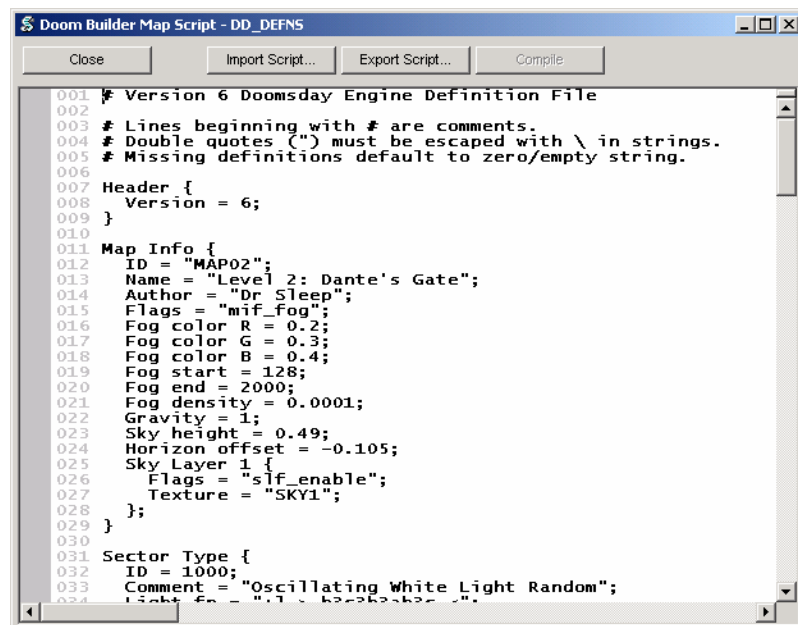
**Figure 1.37.**  
Click the MAKE SCRIPT button.



- 3 Click on the SCRIPTS MENU and select the lump you want to create, say DD\_DEFNS. The script editor will come up and inform you the lump does not exist, and asks if you want to create the lump (see Figure 1.36). Click the **MAKE SCRIPT** button.

The script editor is now ready for you to write your script, and will create a DD\_DEFNS entry in your PWAD when you save.

**Figure 1.38.**  
Script editor in JDOOM mode.



### 1.3.4 The Details Bar

For each EDITING MODE, information about the selected object appears in the bottom panel of the main window beneath the grid. This is called the **DETAILS BAR**. This area displays vital information about the object you've selected or highlighted. This is the easy way to keep track of all of the attributes and details in your map with a simple pass of your mouse. (Remember that you can change the position of the bar to the left, right, or top in the **INTERFACE** tab of the **CONFIGURATION** settings. (You can also choose not to have it displayed at all.)

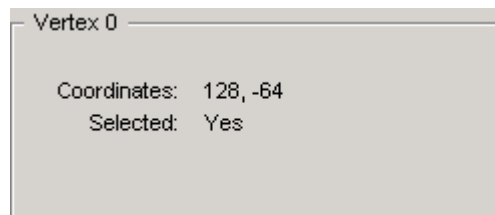


In **VERTICES MODE (V)**, the **DETAILS BAR** displays information on the selected or highlighted object as shown in Figure 1.37.

**• Vertices Details**

- **VERTEX #**
- **COORDINATES** X AND Y GRID POSITIONS
- **SELECTED** YES/NO

**Figure 1.39.**  
DETAILS BAR: Vertices mode.

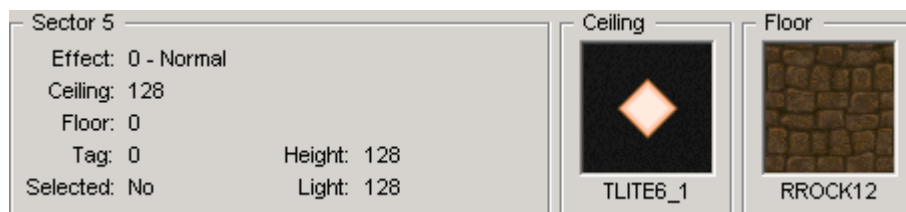


In **SECTORS MODE (press S)**, pass the mouse cursor over the center Sector (#7) of the map, highlighting it. The **DETAILS BAR** will display information about that Sector, as shown in Figure 1.38.

**• Sector Details**

- **SECTOR #**
- **EFFECT** TYPE BY NUMBER AND FUNCTION (BLINKS, -HEALTH %, SECRET..)
- **CEILING** HEIGHT
- **FLOOR** HEIGHT
- **TAG #** OF ASSIGNED LINEDEF, IF ANY
- **HEIGHT** OF THE SECTOR
- **LIGHT** LEVEL 0-255
- **SELECTED** YES/NO
- **CEILING** FLAT IMAGE AND NAME
- **FLOOR** FLAT IMAGE AND NAME

**Figure 1.40.**  
DETAILS BAR: Sectors mode.



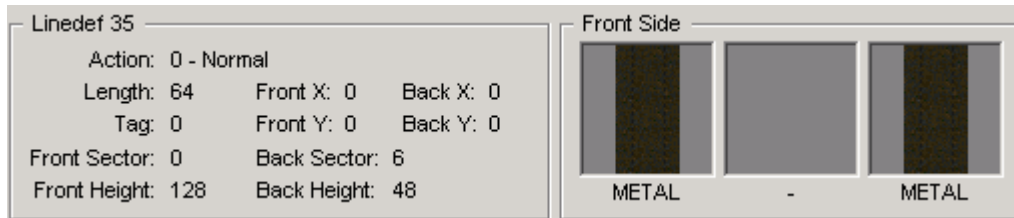


In **LINEDEF MODE (L)** the DETAILS BAR shows statistics and images, first on the LineDef itself, then on the **FRONT SIDE** [FIRST SIDEDEF] and (if it exists) the **BACK SIDE** [SECOND SIDEDEF].

### • LineDef Details

- **LINEDEF #**
- **ACTION** TYPE BY NUMBER AND FUNCTION (OPEN DOOR, RAISE FLOOR, ETC.)
- **LENGTH** IS THE LENGTH IN PIXELS OF THE LINEDEF
- **TAG #** OF ASSIGNED SECTOR, IF ANY
- **FRONT SECTOR** DISPLAYS THE SECTOR REFERENCE FOR THE FIRST SIDEDEF
- **FRONT HEIGHT** IS THE HEIGHT OF THE SECTOR IT FACES
- **FRONT AND BACK X** DISPLAYS THE X OFFSET FOR THE FIRST AND SECOND SIDEDEF
- **FRONT AND BACK Y** DISPLAYS THE Y OFFSET FOR THE FIRST AND SECOND SIDEDEF
- **BACK SECTOR** DISPLAYS THE SECTOR REFERENCE FOR THE SECOND SIDEDEF
- **BACK HEIGHT** IS THE HEIGHT OF SECTOR IT FACES
- **FRONT SIDE** TEXTURE IMAGE AND NAME FOR UPPER, MIDDLE, AND LOWER TEXTURE
- **BACK SIDE** TEXTURE IMAGE AND NAME FOR UPPER, MIDDLE, AND LOWER TEXTURE

**Figure 1.41.**  
DETAILS BAR: LineDef mode.

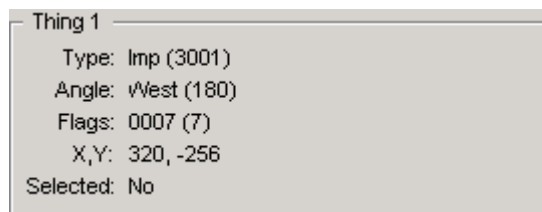


In **THINGS MODE (T)** the DETAILS BAR displays information about the selected or highlighted THING as shown in Figure 1.38.

### • Thing Details

- **THING #**
- **TYPE** BY NAME AND NUMBER
- **ANGLE** DIRECTION BY NAME AND DEGREES THING IS FACING
- **FLAGS** TYPE BY NUMBER AND FUNCTION
- **X, Y** GRID COORDINATES
- **SELECTED** YES/NO

**Figure 1.42.**  
DETAILS BAR: Things mode.



These windows will be very important, since you will always want to know the ceiling and floor heights of your Sectors, what textures are being used, the length of your LineDefs (for when you come to aligning textures later in this document), and other valuable information. Just about everything you need to know is displayed in the DETAILS BAR.

### 1.3.5 The Status Bar

The **STATUS BAR** is the small area at the very bottom of the editor, beneath the **DETAILS BAR**. This area has eleven boxes displaying statistics about your map and active settings. The **STATUS BAR** displays the following:

- Number of Vertices
- Number of LineDefs
- Number of SideDefs
- Number of Sectors
- Number of Things
- Grid Size
- AutoSnap on/off
- AutoStitch on/off
- Zoom scale percentage
- X Axis coordinate
- Y Axis coordinate

52 vertices	64 linedefs	86 sidedefs	15 sectors	7 things	
Grid: 4	AutoSnap: ON	AutoStitch: ON	Zoom: 131%	X 666	Y -666

**Figure 1.43.**

The **STATUS BAR** displays your map's statistics and active settings.


# SECTION 2

## Editing Basics: Making a Level with Doom Builder

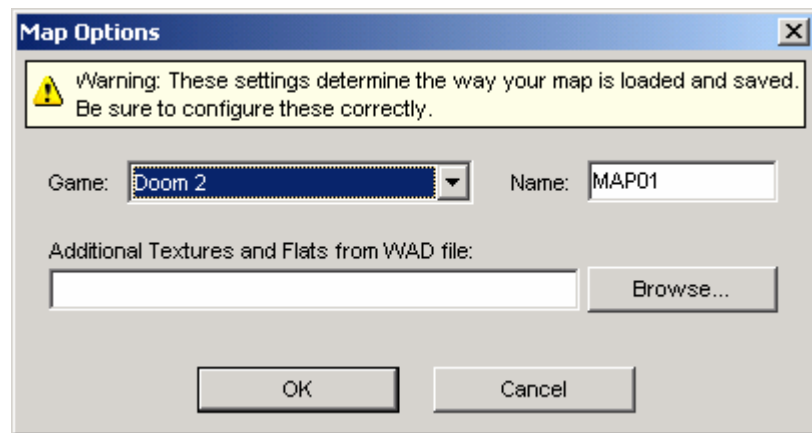
- *Creating Sectors*
- *Adding Sectors*
- *Modifying Sector, LineDef, and Thing Properties*
- *3D Edit Mode*
- *Making a Door*
- *Making a Teleporter*
- *Making a Lift*
- *Making Rising Stairs*
- *Managing Your Map*

## 2.0 Editing Basics: Making a Level with Doom Builder

Section 2.0 will guide you step-by-step through the process of building a DOOM level. Doom Builder's interface is thoroughly explained in [Section 1.0](#), but there are plenty of editing functions yet to be discovered in this section. You'll also get a healthy dose of **DDT**: DOOM Design Theory. To get started, we need to create a new map from scratch.

Click the NEW MAP button  in the toolbar. The MAP OPTIONS dialog box comes up (see Figure 2.1). Click on the drop-down arrow in the GAME area and choose **Doom 2** from the drop-down list. In the NAME area, type **MAP01** in the text box. This determines the *lumpname* (level number) of your map, **not** the *filename* (DEMO.WAD) of your map. Click OK.

**Figure 2.1.**  
The MAP OPTIONS dialog box determines the game type and *lumpname* of your new map.



DB goes through an initialization process and then displays your map grid. Your grid size should be set at 32. Check the fifth window of the DETAILS BAR at the bottom of DB to see your current grid settings. You'll want to *Zoom Out* a bit to around 40% Zoom scale. Use the mouse wheel to scroll in and out (or the NUM +/- keys if you've been using DB before version 1.53) a couple of times and keep your eye on the last window in the DETAILS BAR where your Zoom setting is displayed. We need room to draw a large Sector, and you'll now be in the same scale as the diagrams that follow. Also, if you didn't do so as advised in [Section 1.0](#), make sure for purposes of following this manual that you select the **SHOW VERTICES IN SECTORS MODE** check box in the [INTERFACE](#) tab of the CONFIGURATION dialog box (F5).

We're ready to make our first Sector in Doom Builder!

### 2.1 Creating Sectors

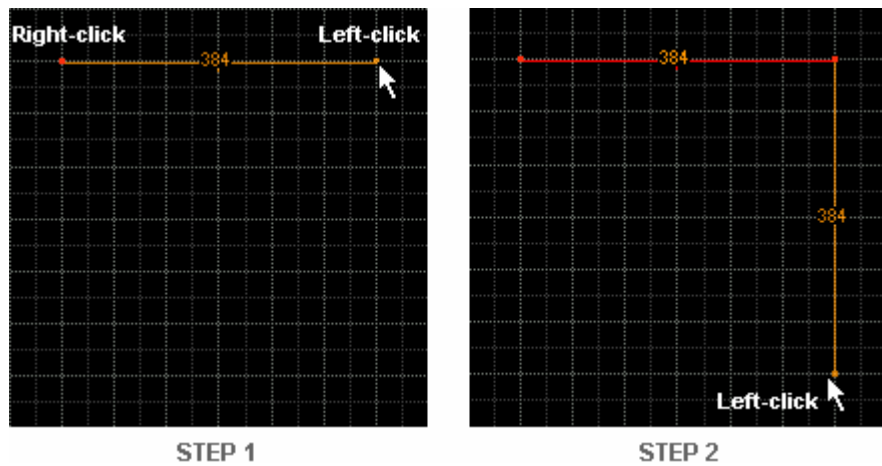
DB uses a *line-draw* method for creating LineDefs and Sectors. In SECTORS MODE, Vertices, LineDefs, SideDefs, and Sectors are created automatically as you draw. Vertices and LineDefs can be created separately in their respective modes. Note that we'll be drawing in a *clockwise* direction. This isn't absolutely necessary in DB – and sometimes we'll deliberately draw counter-clockwise – but since the *start* and *end* Vertices determine the LineDef's direction, and therefore the position of the *First SideDef* (or the RIGHT SIDE), it's not a bad idea to pick a routine and stick to it.

### 2.1.1 Line-Draw Mode

The first thing you'll need to know is how to insert Vertices and connect them with LineDefs.

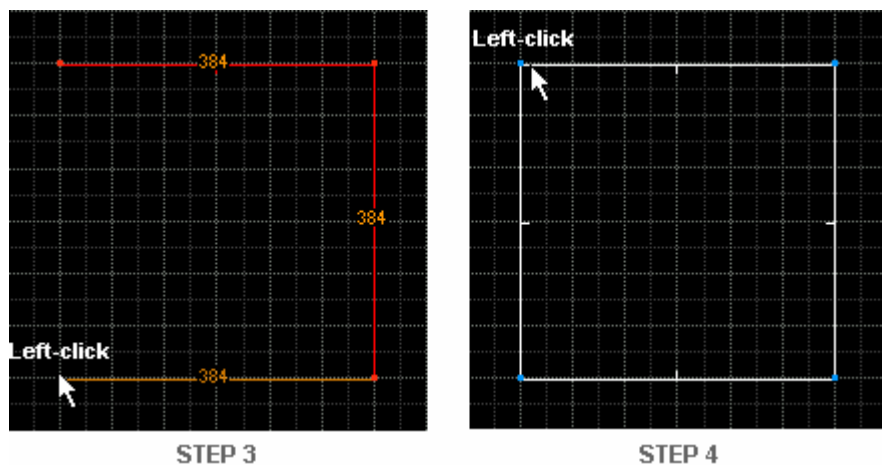
- 1 Enter SECTORS MODE (S). Position your mouse cursor around the X and Y map coordinates -192 and +192 (see the last two windows in the DETAILS BAR) and **right-click** with the mouse (see Figure 2.2a). You've inserted a Vertex! (Move the mouse to the right and you'll see that there's a LineDef attached to the Vertex. The length of the LineDef is displayed and updates dynamically as you stretch the LineDef.) Move the cursor east until the LineDef is 384 units long. Now **left-click** to *anchor* that LineDef.
- 2 Move the cursor down another 384 units and **left-click** to anchor the LineDef.

**Figure 2.2.**  
To insert the first Vertex, right-click on the grid. To anchor the LineDef and insert another Vertex, left-click.



- 3 Now move to the left another 384 units and **left-click**.

**Figure 2.3.**  
To close a Sector, you must end drawing on the first Vertex that you started with.



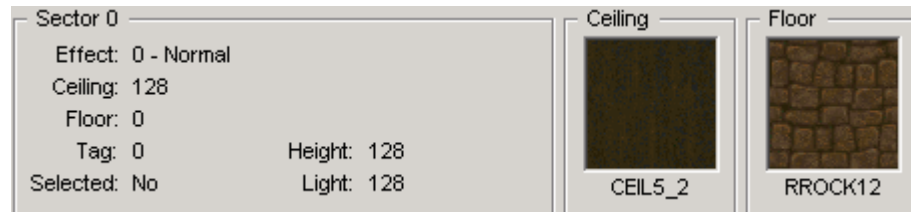
- 4 Move the mouse up and position the cursor over the first Vertex you started with and **left-click** again. DB knows you've completed a Sector and releases the mouse from LINE-DRAW MODE and creates the Sector, now shown in white. You've created your first Sector!

**NOTE:** To close a Sector, you **must** end your drawing on the same Vertex you started with.

- 5 Enter SECTORS MODE (S). Position your mouse cursor in the middle of the Sector and you'll see that it is highlighted orange, indicating a closed Sector.

All objects – Vertices, LineDefs, SideDefs, Sectors, and Things – have an identification number that is used to reference their relationship to other objects. New objects are numbered starting with zero, so this is Sector 0. The DETAILS BAR at the bottom of the screen displays information about the highlighted Sector, including images of the floor and ceiling textures (flats). All you need to do is pass the mouse cursor over an object to highlight it and bring up the detail information.

**Figure 2.4.**  
Sector information is displayed in the DETAILS BAR.



In the next chapter, you'll find out how to add and join Sectors.

### Creating Sectors: Summary of Operations

- Enter SECTORS MODE to start LINE-DRAW MODE.
- To *insert* a Vertex and start LINE-DRAW MODE, *right-click*.
- To *anchor* a LineDef and insert a new Vertex, *left-click*.
- To *close* a Sector and end LINE-DRAW MODE, *left-click* on the *start* Vertex.



## 2.2 Adding Sectors

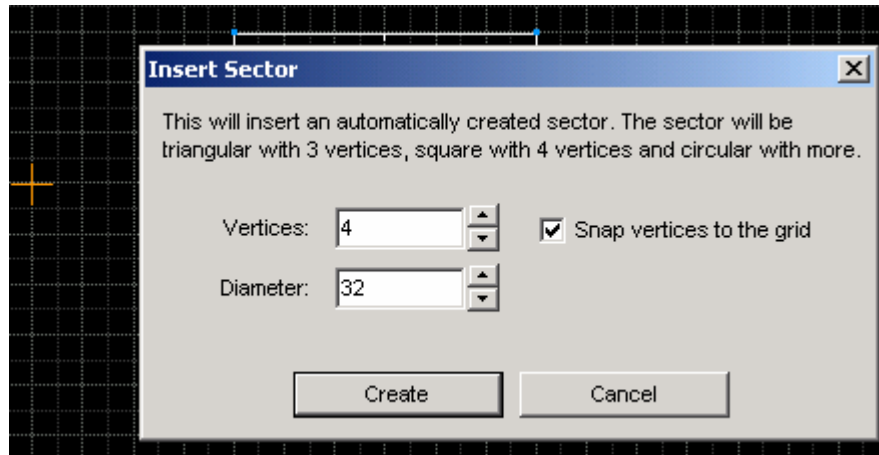
We want to expand our map and add some Sectors. This is the foundation of level creation. There are several ways to add Sectors in Doom Builder. We'll start with the easiest.

### 2.2.1 The Insert Sector Function

If you recall our DEMO map from the first chapter, we want to recreate that map and therefore need to add some rooms off our current Sector. A quick way to create a Sector is by using the **INSERT SECTOR** function.

- 1 Go into **LINEDEF MODE (L)**. Position the cursor at least 256 units to the left of the Sector (see Figure 2.5), hold down the **CTRL** key and **right-click**. An orange cross-hair appears on the grid and the **INSERT SECTOR** dialog box comes up.

**Figure 2.5.**  
The **INSERT SECTOR** dialog box will automatically create and insert a Sector.



You can define the size and shape of the Sector by entering the number of **VERTICES** and the **DIAMETER** in the text boxes (or by click the up-down arrow keys). The **SNAP VERTICES TO THE GRID** check box should be selected by default. This is an excellent way to make circular Sectors. The more Vertices you add, the smoother the circle. Nevertheless, we want to make a square 128x128.

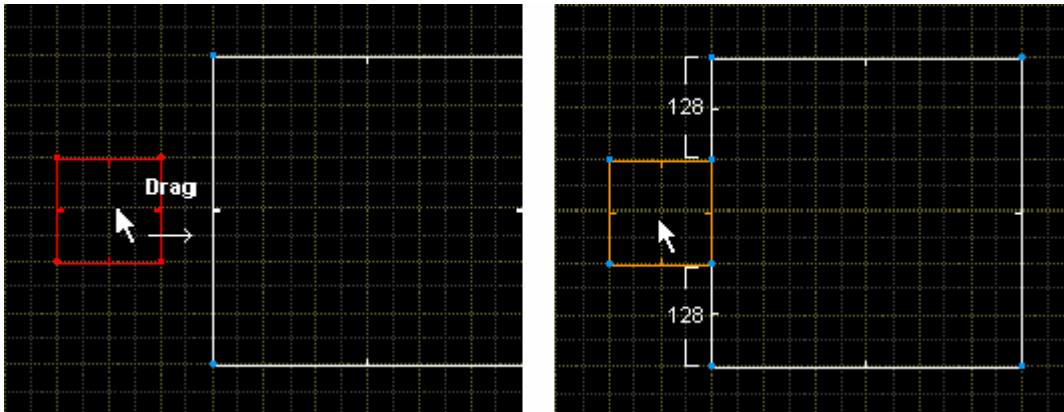
- 2 In the **DIAMETER** text box, enter 64 (or click the up-arrow, which adds 8-pixel increments). Click **CREATE**. DB inserts a 128x128 Sector. (This is Sector 1.)

There are other ways to create new Sectors, and we'll get to those in a moment: we have a couple more Sectors to add. For now, we want to join the new Sector with the first one.

## 2.2.2 Joining Sectors

Let's join Sector 1 to Sector 0. There are two different ways you could do this. The first is to simply drag Sector 1 over to Sector 0. To *drag* an object in DB, you position the cursor over the object, hold down the right mouse button, and move the mouse – *dragging* the object.

- 1 Go into SECTORS MODE (S). Position the cursor in the center of Sector 1, **right-click** and hold the button down, and drag it over to Sector 0 until they meet. You want to center Sector 1 with Sector 0 so that there are 128 units on either side of it (see Figure 2.6).



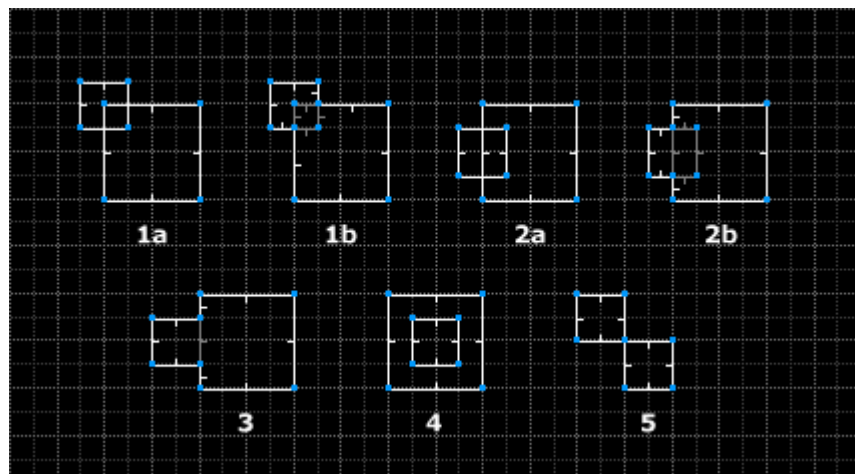
**Figure 2.6.**  
Drag Sector 1 until it adjoins Sector 0.

If you note in Figure 2.6, DB has joined the Sector, merging the Vertices of Sector 1 into Sector 0's LineDef, and splitting it into three – as indicated by the new **VECTORS**. The two Sectors now share LineDef 5, which has become two-sided and transparent.

**NOTE: NEVER DRAG OR DRAW ACROSS LINES!** When you cross LineDefs, you cross into another Sector and the resulting references may be incorrect. Always stay **WITHIN** lines or **OVERLAP** lines, but **NEVER CROSS LINES** when dragging or drawing.

### Figure 2.7.

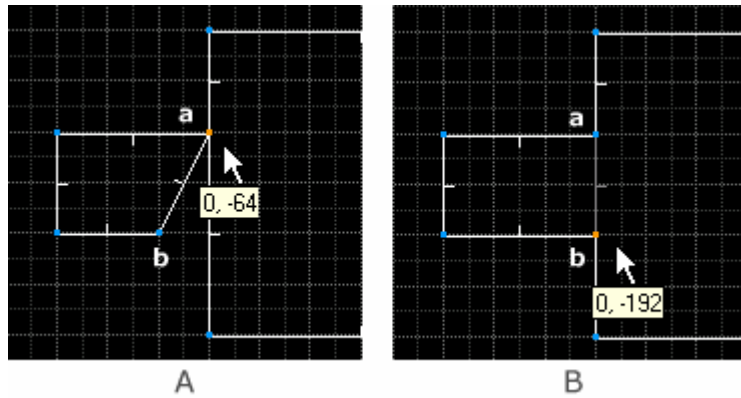
Illegal overlays have no Vertices where the LineDefs intersect. 1a and 2a are illegal; 1b and 2b are legal (note the Vertices at the intersections and the two-sided LineDefs). Examples 3, 4, and 5 are all valid.



Here's the second way to add Sector 1 to Sector 0 (click on the UNDO button  to return Sectors 0 and 1 to their initial state):

- 1 Go into VERTICES MODE (V). Just as we dragged Sector 1 over to Sector 0, this time we're going to drag the Vertices. **Right-click** on Vertex **a** and hold the button down. Drag the Vertex and position it on top of LineDef 3 as shown in Figure 2.8a.

**Figure 2.8.**  
Right-click on each Vertex and drag them over to LineDef 3.

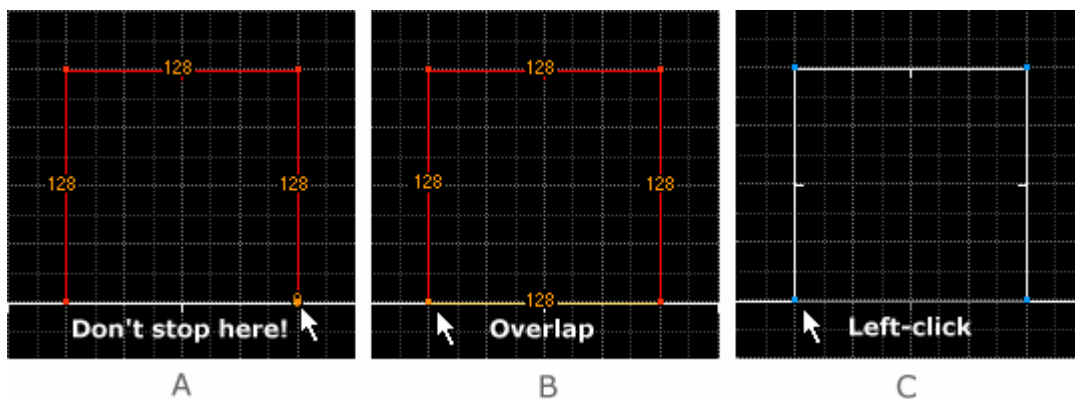


- 2 Now, **right-click** on Vertex **b** and drag it over to LineDef 3. You've joined the Sector!

**NOTE:** LineDefs that share adjoining Sectors are always two-sided. The VECTOR indicates the FIRST SIDEDEF. Doom Builder shows two-sided LineDefs as a light gray line instead of solid white. One way you can tell if your Sector has been added properly is to see that DB has converted the merged LineDef to two-sided.

Let's add another Sector using a different method. We want to add another 128x128 room to the north side of Sector 0.

- 1 Go into SECTORS MODE (S). Position the cursor on LineDef 0, 128 units from the left (see Figure 2.9). **Right-click** to insert a Vertex and begin LINE-DRAW MODE. Drag the cursor up 128 units and **left-click** to anchor the LineDef. Drag to the right 128 units and **left-click**. Drag down till you meet LineDef 0 again, and then **left-click** to insert a Vertex. **But don't stop there!**



**Figure 2.9.**  
Overlap the LineDef and end LINE-DRAW MODE on the *start* Vertex.

Even though it appears in Figure 2.9a that we've completed a new Sector, we still have to end LINE-DRAW MODE on the first Vertex we started with.

- 2 Continue dragging the LineDef and overlap LineDef 0 (Figure 2.9b) until you meet the *start* Vertex and then **left-click** on it (Figure 2.9c). DB merges the two LineDefs we overlapped into a single LineDef, now shared between Sector 0 and Sector 2. You've added the Sector!

**NOTE:** Don't be afraid to overlap LineDefs and Vertices when adding a Sector to another. Doom Builder will remove the redundant objects when it creates the new Sector. Remember to end LINE-DRAW MODE on the same Vertex you started with.

There's one more method for adding a new Sector – and this will involve splitting LineDefs.

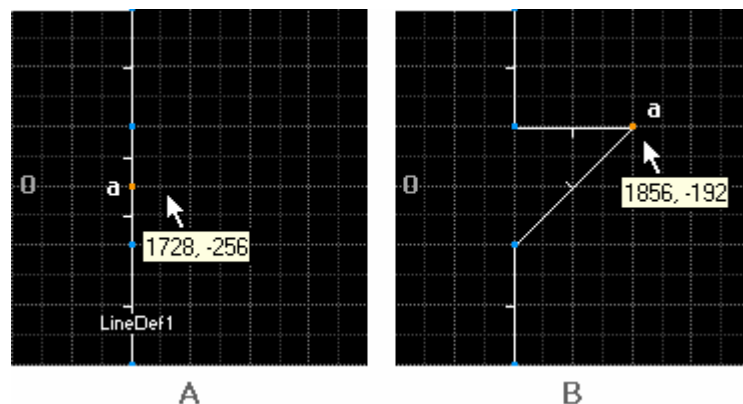
### 2.2.3 Splitting LineDefs and Sectors

Splitting LineDefs is one of the most common functions you'll perform in a map. It's done for a variety of reasons – most common among them adjusting LineDef lengths to accommodate a particular texture, adding trim to doors and corners, or splitting Sectors.

We're going to add a room like Sectors 1 and 2 to the east side of Sector 0 just by splitting LineDef 1 several times. Plus, you'll learn a different way to insert Vertices. Since we're going to be doing some detail work with Vertices, *Zoom In* to around 50% or 60% so you can see the Vertices better. Your grid should be set at 32. Use [ to decrease or ] to increase the size.

- 1 Enter VERTICES MODE. Position your cursor 128 units from the top of LineDef 1 (see Figure 2.10). Now **right-click**. You've split the LineDef by inserting a Vertex, evidenced by the new vectors.
- 2 Bring the cursor down 16 units or so and insert another Vertex with **right-click**. Do it again inside the lower LineDef. You should have three new Vertices (Figure 2.10a).
- 3 Drag Vertex **a** up 64 units and east 128 units (Figure 2.10b).

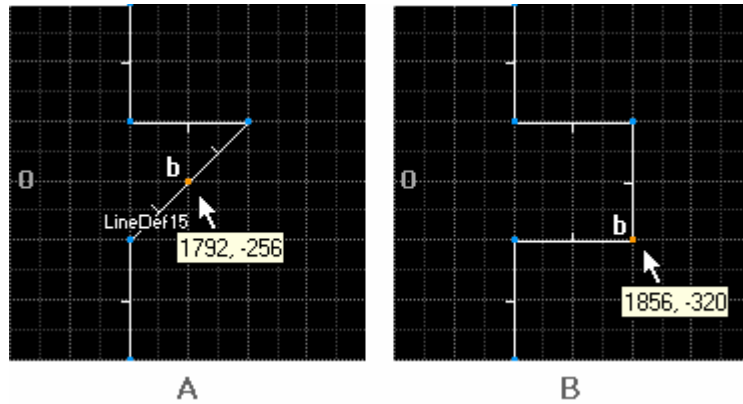
**Figure 2.10.**  
Start a new Sector by splitting a LineDef several times with right-click.



- 4 Position the cursor over LineDef 15, **right-click** to insert a Vertex and split the LineDef (Figure 2.11a).

5 Now, drag Vertex **b** down 64 and east 64 to form a square as in Figure 2.11b.

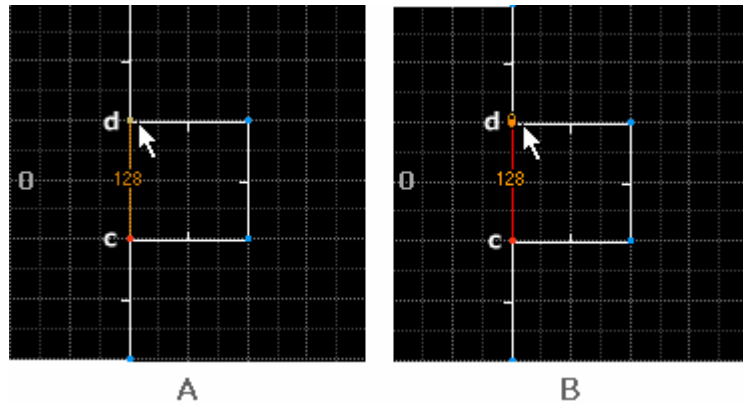
**Figure 2.11.**  
Split LineDefs and drag the Vertices to mold new areas.



This new area is still part of Sector 0. We'll be using a new SPLIT SECTOR FUNCTION to separate it.

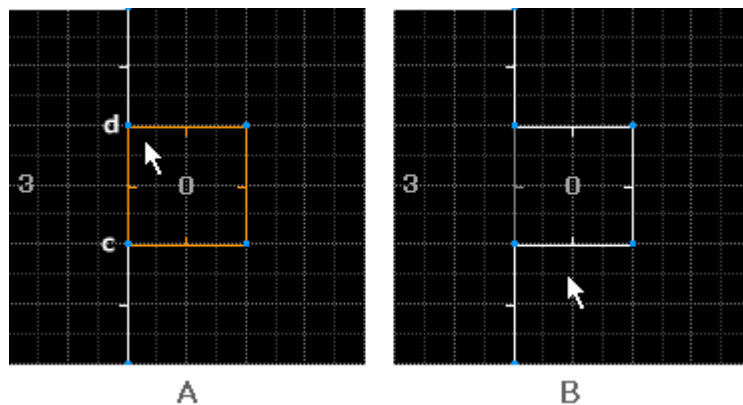
- 1 Enter SECTORS MODE. DB highlights a Sector when the cursor is over it in SECTORS MODE, but by carefully placing your cursor on Vertex **c** (Figure 2.12a), DB will turn highlight off. When it does, **right-click** to overlay a new Vertex and start LINE-DRAW MODE.
- 2 **Left-click** on Vertex **d** to anchor the new LineDef (Figure 2.12b).

**Figure 2.12.**  
Left-click to enter LINE-DRAW MODE.



3 **Right-click** on Vertex **d**. This releases LINE-DRAW MODE. You've created a new Sector!

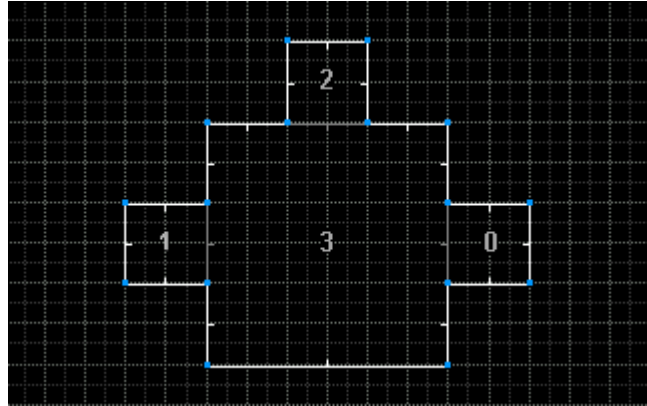
**Figure 2.13.**  
Instead of left-clicking on the *start* Vertex, right-click to invoke THE SPLIT SECTOR FUNCTION and exit.



## 2.2.4 Parent and Child Sectors

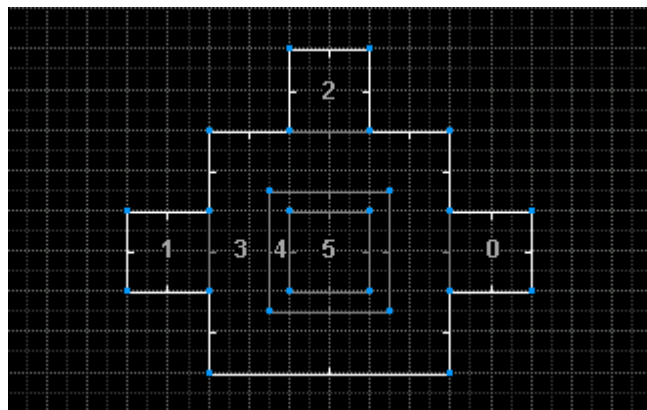
We're going to add several Sectors inside of Sector 3 using a couple different methods. Let's have a look at the map so far (Figure 2.14).

**Figure 2.14.**  
The map so far, with Sector numbers.



- 1 Enter LINEDEF MODE. Hold down the **CTRL** key and **right-click** in the center of Sector 3. The INSERT SECTOR dialog box comes up. Enter a DIAMETER of 96. Click CREATE. There should be a new Sector whose LineDefs are 192 units long. Highlight one of the LineDefs and check the DETAILS BAR for the length. Let's add another Sector in the middle of this.
- 2 Position your cursor in the very center of Sector 4. **CTRL+right-click** to bring up the INSERT SECTOR dialog box. Enter a DIAMETER of 64. Click CREATE. You should now have a 128x128 Sector in the center of Sector 4.

**Figure 2.15.**  
Sector 3 is the PARENT SECTOR. Sectors 4 and 5 are CHILD SECTORS.

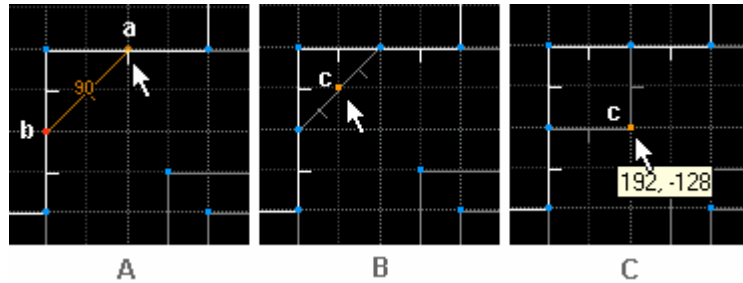


**NOTE:** Sectors 4 and 5 are called CHILD SECTORS, because they're contained within Sector 3, the PARENT SECTOR. These are merely terms of convenience for speaking intelligently about Sectors within Sectors. It's a common, but misleading practice to use the word *room* interchangeably with *Sector*. A *room* is a purely visual construct that exists only in the mind of the player as he negotiates the imaginary architecture.

Let's add CHILD SECTORS to the north-west and south-east corners of Sector 3, which we'll be making into **teleporters** in [Chapter 2.6](#). We'll be using the SPLIT SECTOR FUNCTION.

- 1 In SECTOR MODE, **right-click** 64 units across as shown in Figure 2.16a to insert a Vertex. Move down and left 64 units and **left-click** to anchor the LineDef, inserting another Vertex, and then **right-click** to end the SPLIT SECTOR FUNCTION. You've split the Sector.

**Figure 2.16.**  
Add a CHILD SECTOR by using the SPLIT SECTOR FUNCTION.



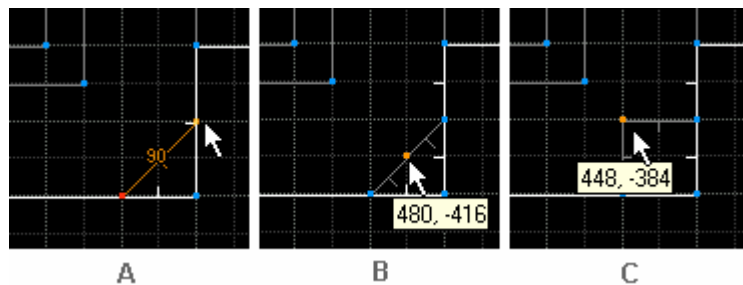
- 2 Enter VERTICES MODE. Position your cursor in the center of the new LineDef and **right-click** to add a Vertex and split the LineDef. Now drag Vertex **c** (Figure 2.16c) to its new position to form a square.

**NOTE:** You can insert a Vertex and drag it in the same operation since you're holding the right mouse button down.

Let's quickly add another Sector just like this one in the opposite corner.

- 1 Enter SECTORS MODE. **Right-click** on position **a** as shown in Figure 2.17a. **Left-click** on position **b**. Now **right-click** to end the SPLIT SECTOR FUNCTION.
- 2 Enter VERTICES MODE. **Right-click** to split the LineDef, then drag Vertex **c** to form a square.

**Figure 2.17.**  
Use the SPLIT SECTOR FUNCTION in SECTORS MODE; but split LineDefs in VERTICES MODE.



You've completed the two Sectors that will become teleporter pads a little later. You've also had a useful demonstration of several ways to add and split Sectors.

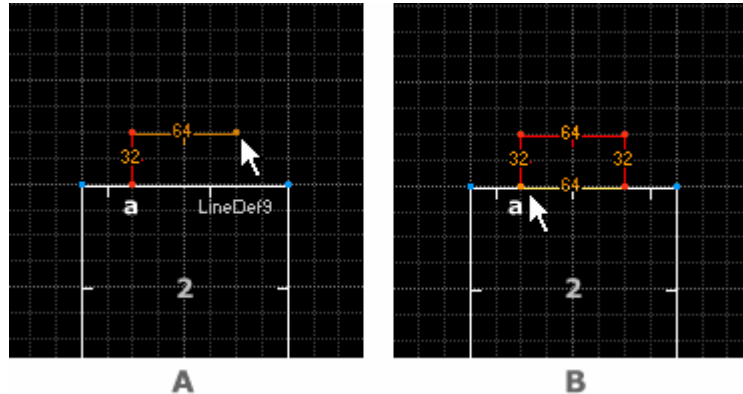
But there's one more way to insert a Vertex and split Sectors.

First, let's construct two more small Sectors. We're going to make a window and a small outside courtyard off Sector 2.

- 1 Enter SECTORS MODE. As shown in Figure 2.18a, **right-click** on position **a** to enter LINE-DRAW MODE 32 units from the start of LineDef 9 (Sector 2). Go up 32 units, **left-click** to anchor the LineDef, go east 64 units, and so on to continue the rectangle shape. Remember to end LINE-DRAW MODE on Vertex **a**, the *start* Vertex, in order to close the Sector.

**Figure 2.18.**

Right-click to enter LINE-DRAW MODE and insert Vertex **a**, complete the entire rectangle shape and left-click to end on Vertex **a** and close the Sector.

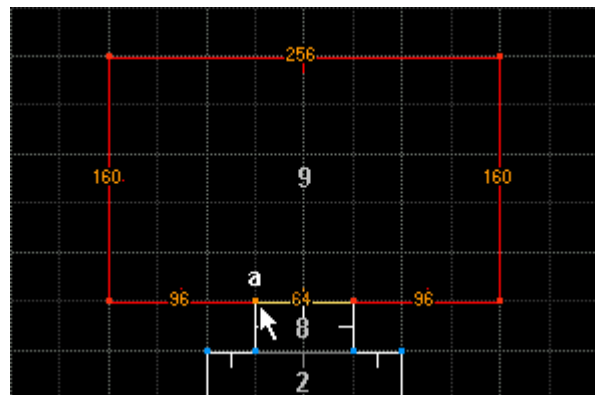


Now, for the courtyard.

- 1 **Right-click** on Vertex **a** (see Figure 2.19), go west 96 units, **left-click** to anchor the LineDef, and continue as shown. Always remember to overlap the last LineDef and **left-click** on the *start* Vertex. DB will merge the LineDefs when it constructs Sector 9.

**Figure 2.19.**

Right-click on Vertex **a** and continue the rectangle. Left-click to anchor LineDefs, and left-click again on Vertex **a** to end LINE-DRAW MODE. You've created Sector 9, which will become a courtyard.

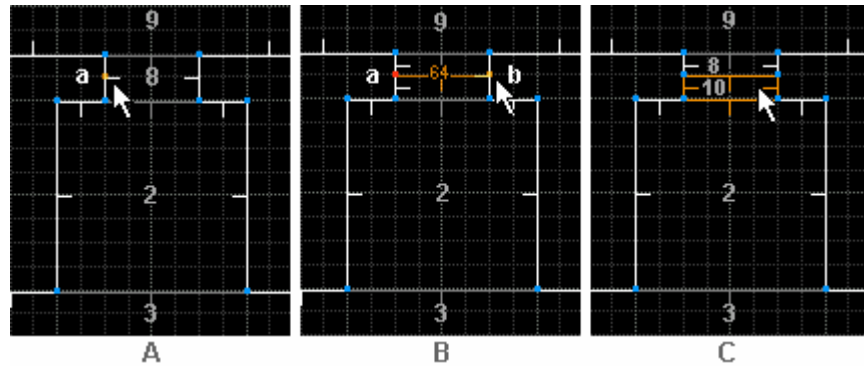


Okay. Now we're going to split Sector 8 horizontally. This Sector will be a window into Sector 9. We're going to split the Sector so that the LineDef running through the middle can later have *see-through* bar textures. (We'll add the texture in the next chapter.) Here's the new way to insert a Vertex:

- 1 Enter LINEDEF MODE. Decrease the grid scale to 16 (press **[]**). Position the mouse cursor to the west and outside of Sector 8. Press **INS**. Move the cursor a little. *You'll see that you have a Vertex dangling from the end of the cursor.* Now the lines in LINEDEF MODE won't be highlighted when you pass the cursor over them.



**Figure 2.20.**  
In LINEDEF MODE, press INS to insert a Vertex and split a LineDef.

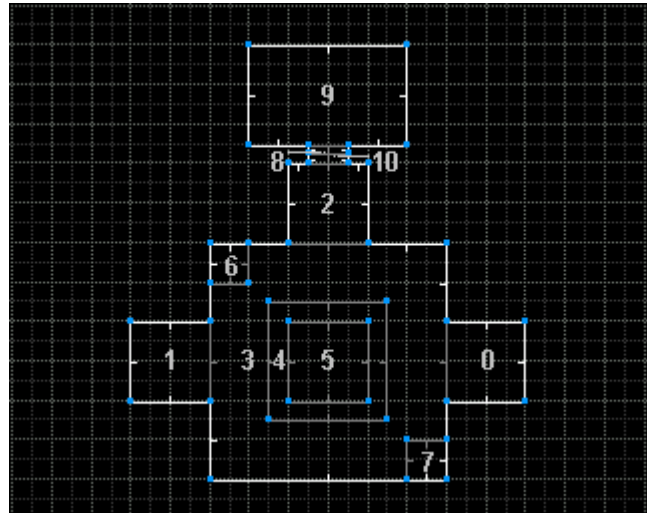


- Place the Vertex on position **a** as shown in Figure 2.20, directly in the middle of the LineDef. **Left-click**. The Vertex is inserted and has split the LineDef. Move the cursor to the opposite LineDef on position **b**. You're in LINE-DRAW MODE. **Left-click** to anchor the LineDef. We want to use the SPLIT SECTOR FUNCTION, so now **right-click** to end LINE-DRAW MODE. The Sector is split into two.

The SPLIT SECTOR FUNCTION is extremely useful, as you can see; and we'll be using it quite a bit. You can effectively split Sectors in two, draw LineDefs within a Sector to create *triggers*, or cut up a Sector into individual bits for special lighting effects.

Let's have a final look at our map (Figure 2.21). It's not particularly clever, but it serves to introduce you to the different techniques, key and mouse presses you need to know to create objects. And right now, you should know them all. Doom Builder isn't a difficult editor to use, as you can see, once you know how it works.

**Figure 2.21.**  
You've learned how to add Sectors, join Sectors, and split Sectors. You now know all the key and mouse combinations needed to make levels in Doom Builder.



### Adding Sectors: Summary of Operations

- Create a new Sector with the INSERT SECTOR tool using *CTRL-right-click*
- Drag a Sector or set of Vertices by holding down the *right mouse button*
- Split LineDefs in VERTICES MODE with *right-click* (preferred) or the *INS* key (slower)
- Split a Sector in SECTORS MODE using the SPLIT SECTOR FUNCTION: *right-click* to start, *right-click* to end

## 2.3 Modifying Sector, LineDef, and Thing Properties

Now we'll get to the real meat of level design, which is customizing your level to a particular look and feel. The first room of your map should determine the mood for the entire level. It's the first place the player sees, and you should take time to set up an atmosphere. Instead of throwing a gang of Imps and Demons at the player right off the bat, hit him with an aesthetic he's never seen before.

Note that many of the operations such as floor and ceiling height changes, texture and flat placement and alignment, and lighting can all be done in 3D EDIT MODE (covered in the next chapter). In fact, it's often preferable to do so. But many other operations must be done through the dialog boxes. I'll point out which functions can be performed in 3D EDIT MODE as we go along.

Now, let's look at how changes are made, object by object.

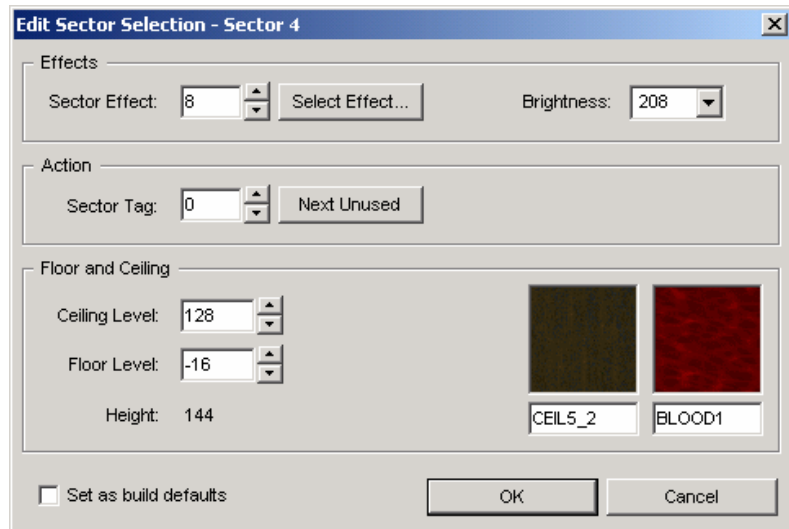
### 2.3.1 The Edit Sector Selection Dialog Box

In [Chapter 1.1.2](#), we set certain texture and flat defaults that would be used throughout our level in the DEFAULTS tab of the CONFIGURATION dialog box. We chose BRICK6 as our default texture, RROCK12 as our default floor flat, and CEIL5\_2 as our default ceiling flat. If you pass your cursor over Sector 3, you'll be able to see what flats are being used in the DETAILS BAR below the grid. Let's change the flats and some other attributes in some of our CHILD SECTORS.

- 1 Enter SECTORS MODE. To bring up the EDIT SECTOR SELECTION dialog box, right-click on Sector 4. We want to add a little blood trench here with Sector 5 as a platform in the middle. Lower the floor height 16 pixels by entering **-16** in the FLOOR LEVEL text box, or click the down arrow for -8 pixel increments. (Note that the HEIGHT setting changes from 128 to 144. This represents the overall Sector height.)

**Figure 2.22.**

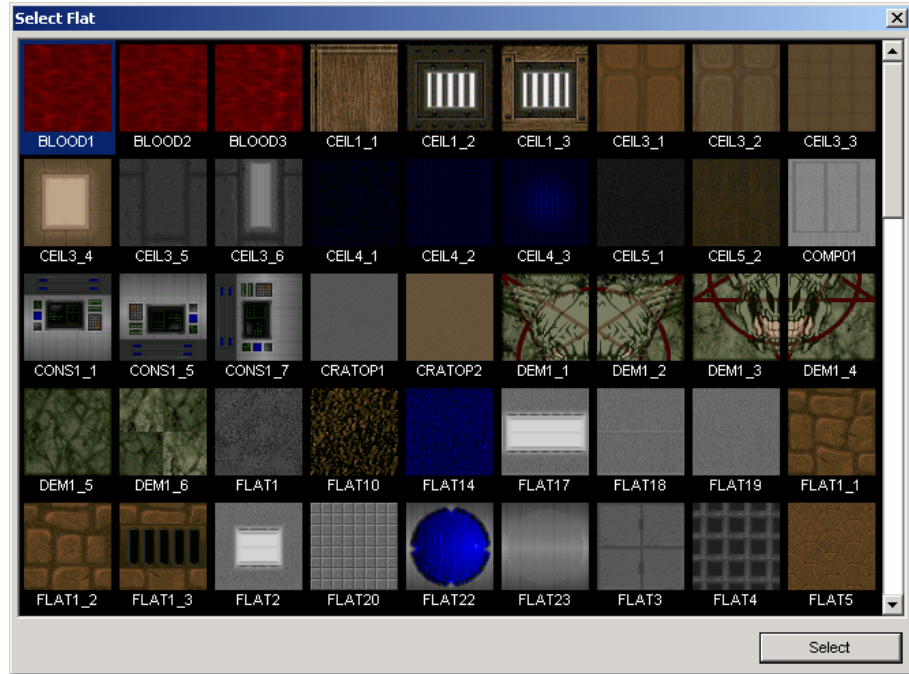
Right-click to invoke the EDIT SECTOR SELECTION dialog box.



- 2 We want this area to be brighter than the rest of the room. Click on the drop-down arrow in the BRIGHTNESS area, scroll up and choose **208**. Click on the FLOOR TEXTURE image to bring up the SELECT FLAT dialog box.

**NOTE:** Animated textures and flats use a series of frames beginning with the first-numbered flat or texture. When using animated graphics, always choose the first graphic in the series, such as BLOOD1.

**Figure 2.23.**  
Use the scrollbar to view all the images in the SELECT FLAT dialog box.



- 3 Click on **BLOOD1**, then click SELECT to return to the EDIT SECTOR SELECTION dialog box. Now click OK.

Let's make a couple more changes.

- 1 Right-click on Sector 5. Change the CEILING HEIGHT to **112**. Click OK.
- 2 Select Sectors 8 and 10 (left-click on them both). Right-click to bring up the dialog box and change the FLOOR HEIGHT to **32**, and then the CEILING HEIGHT to **96**. Click OK to return to the map. Press **C** to clear all selections.

Because we've altered the floor and ceiling heights of Sectors 4, 5, 8 and 10, all of the LineDefs shared by Sectors 3, 4, and 5 plus 8, 9, and 10 now have visible SideDefs. These need textures. We're going to change some of them in **3D EDIT MODE** a little later.

I should point out now that all of the changes we've made so far in this chapter can be done in 3D EDIT MODE, which we'll get to in [Chapter 2.4](#). You can point at a floor or ceiling, left-click, and bring up a flat viewer similar to the one above. You can change floor and ceiling heights, light levels, and make texture changes. 3D EDIT MODE is one of Doom Builder's most attractive features, but it's also important to go through the steps in regular 2D EDIT MODE, since it's what you'll be most familiar with if you've created levels before. Throughout this manual, I'll be pointing out what actions can be done in 3D EDIT MODE, just so you know that you have a choice.

But now there are two other very important areas in this dialog box with functions that *can't* be done in 3D EDIT MODE: Sector EFFECTS and ACTIONS.

### 2.3.1.1 Sector Effects and Actions

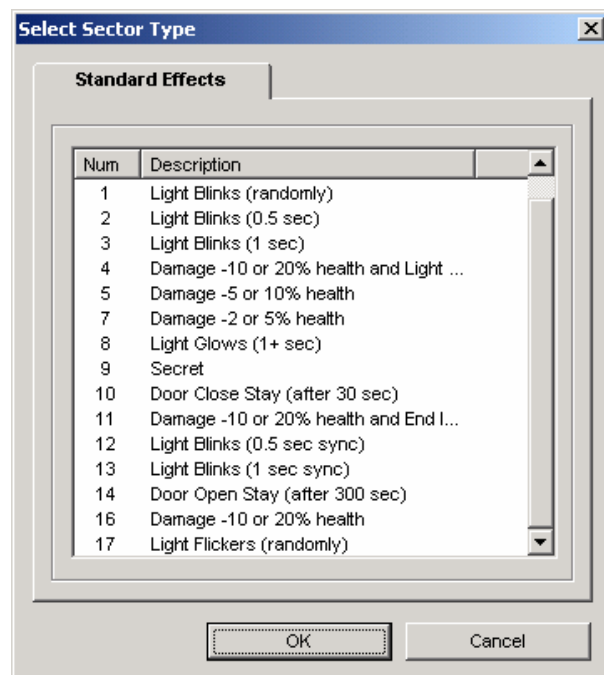
Sector **ACTIONS** are completely different and separate from Sector **EFFECTS**. Sector **ACTIONS** are dependent on **LINEDEF ACTION TYPES**, which are assigned to the LineDefs and require activation by the player in one form or another (usually by the player *using* a SWITCH or by *walking over* an invisible *trigger*). The actions defined by **LINEDEF ACTION TYPES** (open door, raise lift, lower ceiling, etc.) are performed by Sectors, and the Sectors and LineDefs are associated by shared **TAG** numbers that are assigned by the level designer. The tag numbers are totally arbitrary and are used simply to link a LineDef and its command with the Sector taking the orders. They would both be given the same tag number. So, the **SECTOR TAG** text box in the **EDIT SECTOR SELECTION** dialog box is where the tag number goes. (We won't be working with **ACTION TYPES** or **TAG** numbers until we get to editing LineDefs, which is in the next chapter.)

Sector **ACTIONS** are therefore usually temporary and dependent on an action by the player.

Sector **EFFECTS** (also called **SECTOR SPECIALS** or **SECTOR TYPES**) are permanent. Once set, they remain in their state throughout the game. They're not dependent on being triggered by the player: they're environmental effects, of a sort, and define the nature of the entire Sector. For instance, you've probably fallen into your share of toxic waste pools in **DOOM** and discovered their disastrous effect on your health and well-being if you didn't find a radiation suit very quickly. This is a Sector special effect – and there are five different types that cause damage. Randomly blinking and glowing lights are the most common Sector effect. There are seven different kinds of those. Two others control doors that open or close, and one defines the room as *secret*, giving the player special credit for discovering it. So there are 15 in all, not counting Sector type 0, which is *normal*.

Let's change the **EFFECT** in Sector 4 to give the blood pool an eerie, pulsating glow.

- 1 Right-click on Sector 4. Click the **SELECT EFFECT** button to bring up the **SELECT SECTOR TYPE** dialog box. Note that this is where all of your **SECTOR SPECIALS** are located. Choose number **8 LIGHT GLOWS (1- SEC)**. Click **OK**.



**Figure 2.24.**

Choose Sector special effects in the **STANDARD EFFECTS** tab of the **SELECT SECTOR TYPE** dialog box.

Only one Sector effect can be applied to a Sector, but it can still perform actions defined by LINEDEF ACTION TYPES. Take note that the effects applied to a PARENT SECTOR do not overlap or affect their CHILD SECTORS. (That's the whole point of CHILD SECTORS, in one sense.)

**NOTE:** Sector **EFFECTS** are permanent environment definitions of a Sector's properties. Sector **ACTIONS** are performed in conjunction with a LineDef assigned a LINEDEF ACTION TYPE and sharing a TAG number.

Sector effects and actions cannot be defined or modified in 3D EDIT MODE. So while we're in 2D EDIT MODE, let's modify some of our LineDefs in the same manner as the Sector changes we just made to familiarize you with the EDIT LINEDEF SELECTION dialog box.

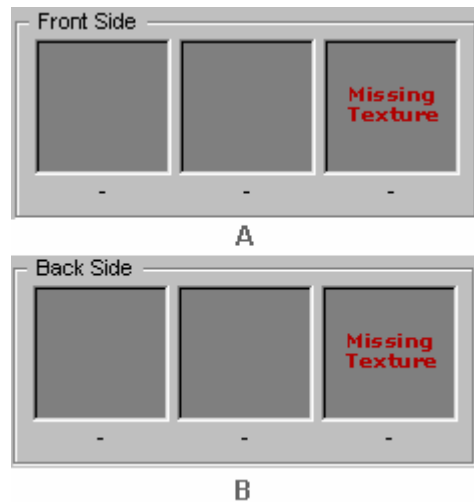
### 2.3.2 The Edit LineDef Selection Dialog Box

LineDefs are the workhorses of the DOOM engine, and the most easily misunderstood. They're more than simple lines: they're complex, composite structures that possess attributes, handle information about texture placement, and carry instructions that control Sector actions. In this chapter we'll be dealing mostly with the SideDefs.

- 1 Go into LINEDEF MODE. Highlight the LineDefs shared by Sectors 3 and 4 (19, 20, 21, and 22). The DETAILS BAR shows now that we have *missing textures*: specifically, the *lower texture* on the FRONT SIDE or FIRST SIDEDEF (Figure 2.25a). And if you highlight the inner LineDefs shared by Sectors 4 and 5 (23, 24, 25, and 26), you'll see that we're missing *lower textures* on the BACK SIDE or SECOND SIDEDEF (Figure 2.25b).

**Figure 2.25.**

FRONT SIDE [FIRST SIDEDEF] and BACK SIDE [SECOND SIDEDEF] are missing *lower textures*.

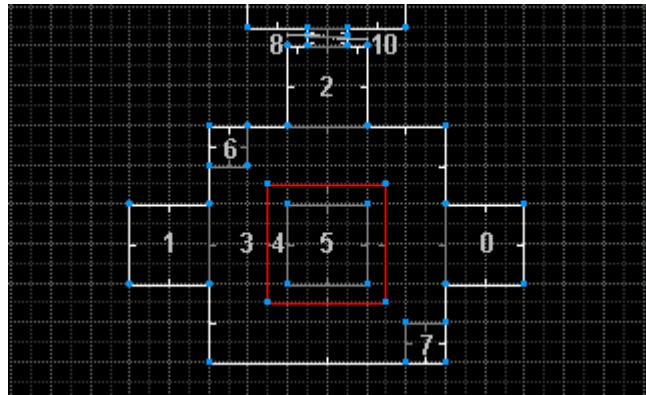


Let's do a mass selection of one set of LineDefs to edit instead of changing them one at a time.

- 2 Left-click on LineDefs 19, 20, 21, and 22 (see Figure 2.26). They all turn red. Now right-click on any one of them. The EDIT LINEDEF SELECTION dialog box comes up.

**Figure 2.26.**

Select the LineDefs shared by Sectors 3 and 4 by left-clicking on each in turn.



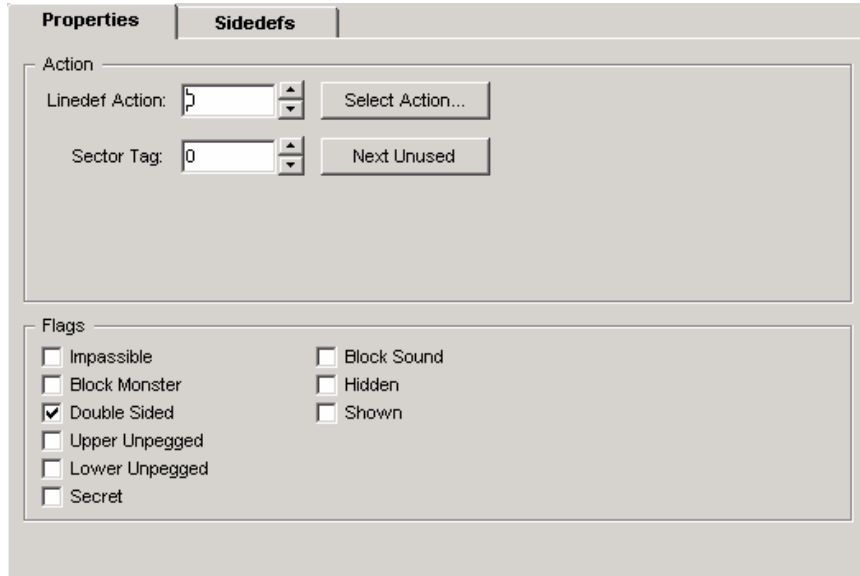
**NOTE:** You can edit several objects at once, so long as you're only changing one attribute. Don't mix texture changes for both FIRST and SECOND SIDEDFS together. Do them separately.

The EDIT LINEDEF SELECTION dialog box has two tabs: **PROPERTIES** and **SIDEDFS**.

The **PROPERTIES** tab is for setting LINEDEF ACTIONS, SECTOR TAGS, and FLAGS (see Figure 2.27). (We'll be returning to this tab a little later when we talk about the UNPEGGED attribute, and LINEDEF ACTION TYPES.)

**Figure 2.27.**

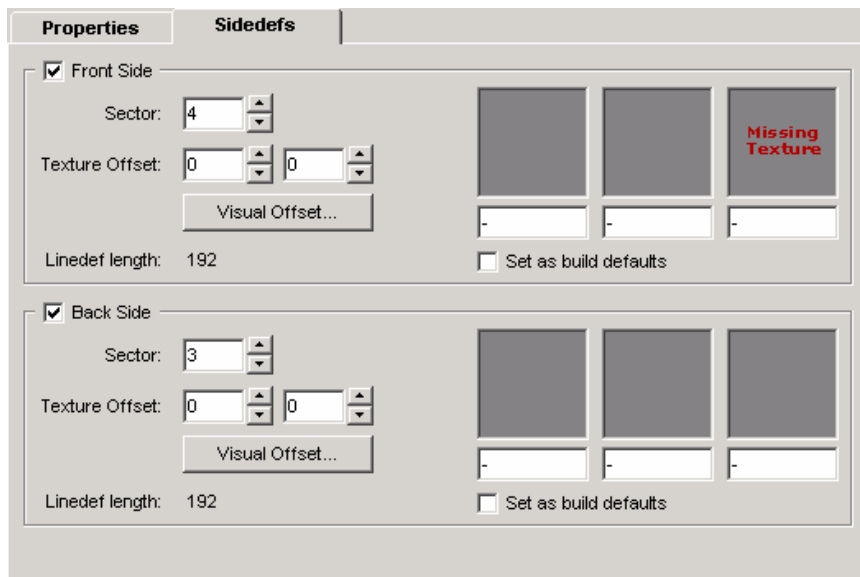
The PROPERTIES tab is for setting LINEDEF ACTIONS, which define all moving Sector properties in DOOM.



The **SIDEDFS** tab is for selecting UPPER, LOWER, and MIDDLE (NORMAL) TEXTURES, setting texture OFFSETS X and Y (for aligning textures), and modifying SECTOR references for the FIRST and SECOND SIDEDFS (see Figure 2.28).

**Figure 2.28.**

Make texture changes in the SIDEDFS tab.



To select a texture, you can enter its name in the respective text box (be sure to spell it correctly or DOOM will crash with the GETNUMFORNAME error. Also, be sure to leave the NULL INDICATOR dash "-" in text boxes with no textures, or this will cause an error, as well), or left-click on the image area. Let's do that now.

- 1 Left-click on the FRONT SIDE lower texture image with *missing texture* in red letters. This brings up the SELECT TEXTURE dialog box (Figure 2.29). Scroll down the images until you see the **METAL** texture and then left-click on it. Click SELECT.

**Figure 2.29.**  
Click on a texture to see its size in the lower left corner.



- 2 Click OK to exit THE EDIT LINEDEF SELECTION dialog box. You've added *lower textures* to the FIRST SIDEDEFS.

Now you need to take care of the SIDEDEFS made visible by your altering the floor and ceiling heights of Sectors 8 and 10 to make a window.

- 1 Highlight LineDef 35. You'll see in the DETAILS BAR that the FRONT SIDE is missing UPPER and LOWER TEXTURES. Right-click on LineDef 35 to bring up the EDIT LINEDEF SELECTION dialog box. Click on the SIDEDEFS tab. In the text boxes under the UPPER and LOWER *missing texture* images, enter **BRICK6**. Click OK.

We have more attributes to change, but we'll do that in 3D EDIT MODE in the next section. Let's talk a little more about the FIRST and SECOND SIDEDEFS, plus *lower*, *upper*, and *middle textures*: what they are, when and why you need them.



### 2.3.2.1 SideDefs and Texture Management

LineDefs, SideDefs, Textures, and Flats were discussed in [Chapter 1.3.1.1](#) to some extent, but let's reexamine them visually.

Each LineDef has at least one side, called the **FIRST SIDEDEF**, which is always the **RIGHT SIDE**, and is indicated by a *vector* – the little stick that juts out perpendicular to the LineDef. It can also have a second – or **left** – side, called the **SECOND SIDEDEF**, if it adjoins two Sectors. **SideDefs** actually define the boundaries in the map, and these boundaries define the borders of a Sector. An enclosed set of SideDefs is what comprises a Sector.

Textures are assigned to the SideDefs. Texture maps are drawn from the *top down*, starting with the upper left edge and tiled horizontally to the right. Most textures are either 64x128 or 128x128 pixels. This is why your map should have binary Sector heights to accommodate the textures you plan on using. Your walls may be higher than 128 pixels, but you should take care to choose textures that tile well vertically.

When LineDefs are shared by Sectors of varying heights, one or both SideDefs will require *upper* or *lower* textures.

- **MIDDLE** or **NORMAL TEXTURES** are on the wall space between the floor and the ceiling.
- **LOWER TEXTURES** are on the wall space between one floor height and the next as viewed from the Sector with the lower floor.
- **UPPER TEXTURES** are on the wall space between one ceiling height and the next as viewed from the Sector with the higher ceiling.

**Figure 2.30.**  
The blood trench with missing upper and lower textures.

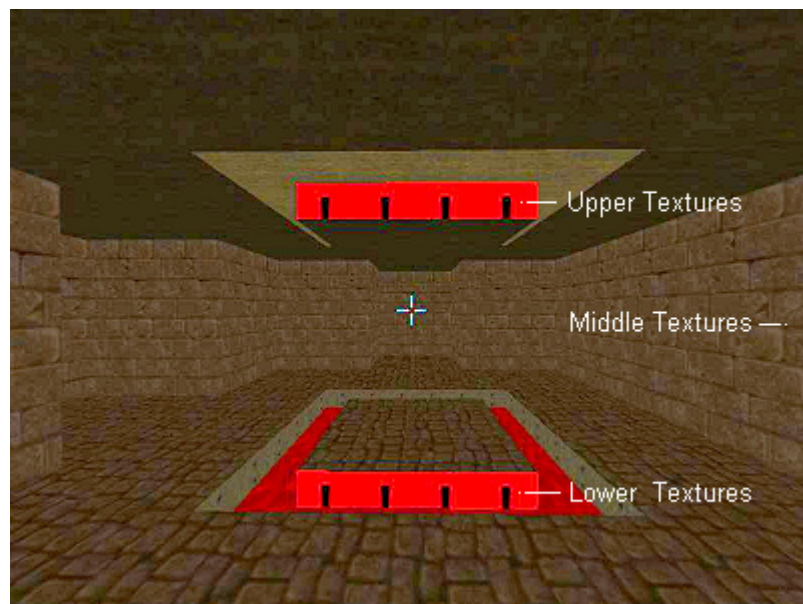


Figure 2.30 shows our map and the areas where textures are needed. In DB, missing textures are drawn as bright orange texture markers with exclamation marks. We lowered the floor height of Sector 4 to make a blood trench. This exposed the **FIRST SIDEDEFs** of the LineDefs shared by Sectors 3 and 4, and the **SECOND SIDEDEFs** of the LineDefs shared by Sectors 4 and 5.

We'll also deal a little more with *upper* and *lower* textures when we construct doors and lifts. But one more important thing about them in particular: something called the **UNPEGGED** attribute.

### 2.3.2.2 The *Unpegged* Attribute

Figure 2.31 shows our window Sector. If you look closely at the upper and lower textures, you'll see that they're misaligned with their neighbors on the left and right.

**Figure 2.31.**  
Sectors made into windows typically have upper and lower textures.



This is typical for window Sectors and isn't due to something we did wrong. The DOOM engine maps upper and lower textures in this fashion:

- **UPPER TEXTURES** are drawn from the bottom up.
- **LOWER TEXTURES** are drawn from the top down.

Figure 2.32 reconstructs our window Sector area using the BRICK6 texture. This texture is 64 pixels wide and 128 tall. DOOM textures are cleverly planned and drawn. Note that each of the eight rows of bricks is roughly 8 pixels high. Our upper texture is therefore 32 pixels high, and since it's drawn from the bottom up, that means it's drawn vertically from Y mapping coordinate 128 to 97. The lower texture is drawn from 1 to 32 down.

**Figure 2.32.**  
The BRICK6 texture split into sections and how it is mapped as upper, lower, and middle textures. Textures that are *misaligned* don't match the X or Y mapping coordinates of their neighbors.

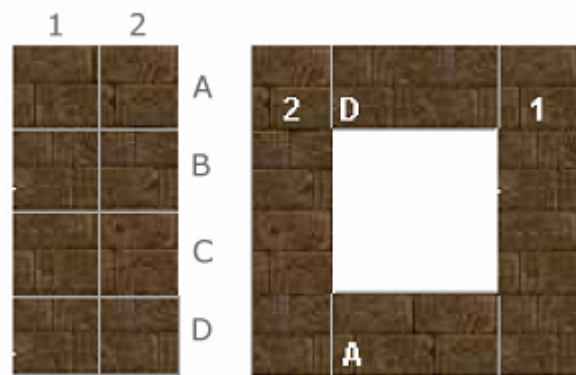


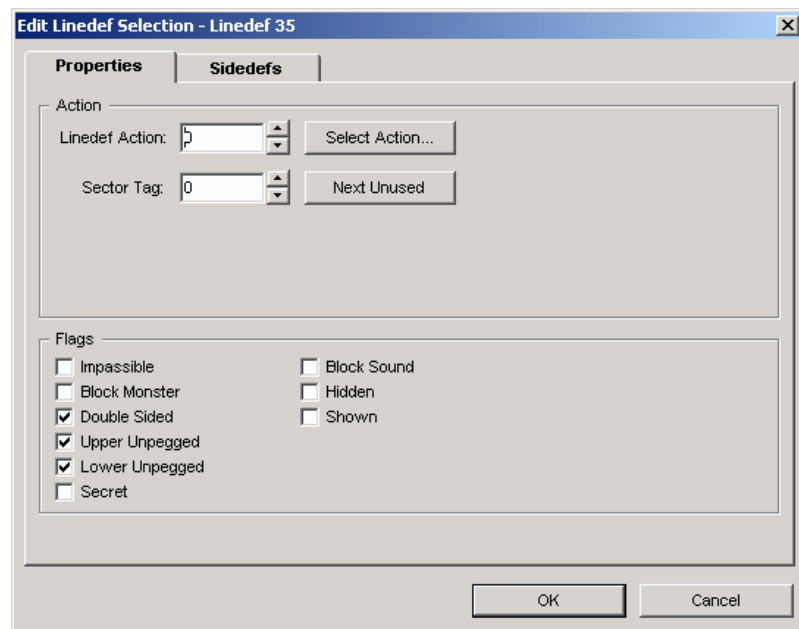
Figure 2.32 further shows the BRICK6 texture sliced into sections and the portions of the texture that are mapped as upper, lower, and middle textures. We could fix this misalignment by adjusting the Y OFFSETS (up/down) for each of the SideDefs in the SIDEDEFS tab, but it's easier to use another method that adjusts the properties of the LineDef itself.

We can correct this by selecting the **UNPEGGED** attribute flags in the PROPERTIES tab of the EDIT LINEDEF SELECTION dialog box. *What this will do is cause the **upper texture** to be drawn from the **top down**, and the **lower texture** to be drawn from the **bottom up**.* It takes care of both SideDefs at once.

This can also be corrected in 3D EDIT MODE, which we'll look at in [Chapter 2.4](#). In fact, all texture alignment problems can be corrected dynamically in 3D EDIT MODE. It's important, however, to know how to do this in the dialog box so that you understand what exactly is being changed, and why. Plus, it's sometimes easier and quicker to do it while you're editing other LineDef properties when constructing your map. On the whole, however, most texture and flat changes are easier to do when you can see the change before your eyes in 3D EDIT MODE.

- 1 Go into LINEDEF MODE. Right-click on LineDef 35 to bring up the EDIT LINEDEF SELECTION dialog box. In the PROPERTIES tab and under FLAGS, select the check boxes next to UPPER UNPEGGED and LOWER UNPEGGED. Click OK.

**Figure 2.33.**  
Select the UPPER UNPEGGED and LOWER UNPEGGED FLAGS.



This causes the upper and lower textures to be mapped in the correct order, as mentioned above.

Remember that anytime you have upper textures or lower textures, they'll be drawn as described. But it's not always necessary to apply the UNPEGGED attribute to them. Stairs, for instance, are typically 16 pixels high. The STEP textures used for stairs are 16 pixels by 32 pixels. Thus, they fit the stair risers whether they're drawn from the top down or the bottom up. In short, if your texture fits, you don't have to unpeg it or align it.

Let's look at the AUTOALIGN TEXTURES function next. (For more information on texture alignment and the UNPEGGED attribute, see my article entitled [Texture Management: The Unpegged Attribute and Texture Alignment](#).)

### 2.3.2.3 The AutoAlign Textures Function

Aligning textures in your level is probably the second most important aspect of level design and certainly one of the factors that determines whether your level is any good or not. I'll tell you why this is so at the end of this chapter.

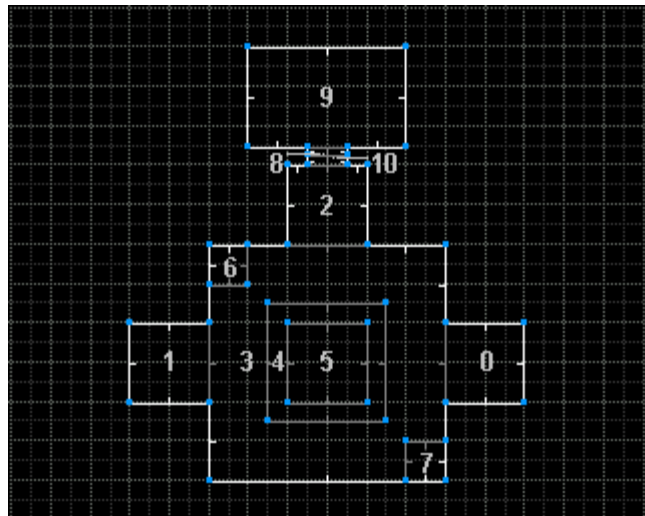
For now, let's examine what causes textures to be misaligned in the first place, and how to correct them. Using the `UNPEGGED` attribute on upper and lower textures is good for aligning textures along the Y axis. But the textures may still be misaligned on the X axis, depending on the lengths of your LineDefs, where the texture started, and a long list of other circumstances.

Textures are generally 64 or 128 pixels wide. The rooms we've made in our test level have kept to a binary scheme (or multiples of 8) with the textures very much in mind. You'll notice that most Sectors have – or started with – LineDef lengths of 384, 256, 128, and 64. As we've added other Sectors and windows, the LineDefs have been split here and there; but with only one instance that we really need to take care of, and that's our window Sector again.

Let's have a look at the map again.

**Figure 2.34.**

Textures in Sectors 2, 8, and 10 are likely misaligned around the window.

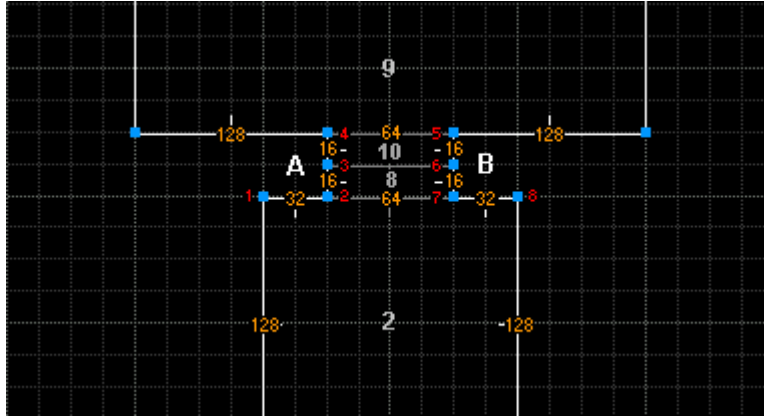


Where we're likely to run into a problem is Sector 2: our window splits the north wall of Sector 2 leaving 32-pixel LineDefs on either side of the window. And since the window Sector itself is split into Sectors 8 and 10, their short LineDefs are only 16 pixels wide.

Let's look at a diagram of how this is texture mapped (see Figure 2.35).

DOOM maps textures from the top left corner down vertically and to the right horizontally on each LineDef. When a new LineDef starts, the texture is remapped, starting again at the top left corner; it isn't "continued" from where the previous texture left off.

**Figure 2.35.**  
Textures from Vertex 2 to Vertex 8 will need to be aligned along the X axis.

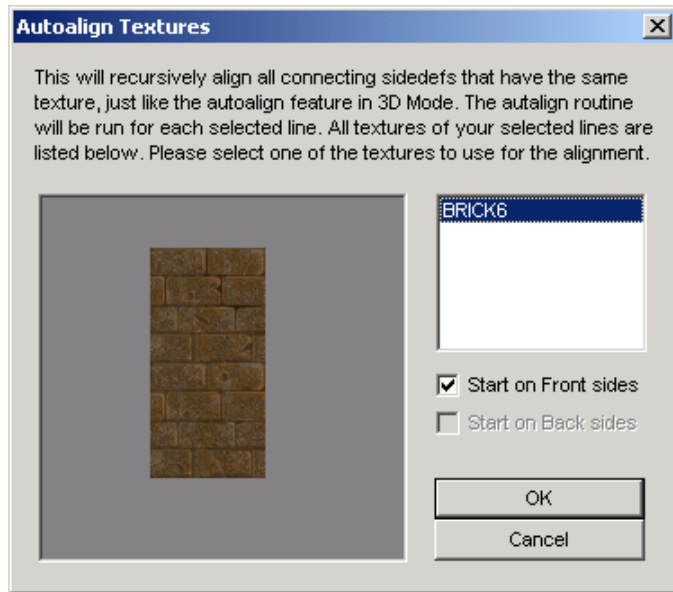


In area **A** from Vertex 1 to Vertex 2, the BRICK6 texture runs from pixel 1 to 32 and then stops. From Vertex 2 to Vertex 7, the texture starts over at 1 and ends at 64. Because the previous texture is only 32 pixels wide, the LineDef from Vertex 2 to 7 should be offset 32 on its X axis. The LineDef from Vertex 2 to 3 needs to be offset 32 pixels also. Since it is 16 pixels long, you add that to the offset of 32 and get an offset of 48 for the next LineDef from Vertex 3 to 4.

You see that this isn't exactly something requiring knowledge of Einstein's *Special Theory of Relativity*, but you would need to do these sorts of calculations to keep your textures aligned (if you had to do it manually, as in the old days). Fortunately, Doom Builder's AUTOALIGN TEXTURES function takes care of all of that for you.

- 1 Enter LINEDEF MODE. Select the LineDef from Vertex 1 to 2. Press the **A** key. This brings up the AUTOALIGN TEXTURES dialog box (Figure 2.36). It'll show you the texture you've selected – in this case **BRICK6**. The LineDef we selected only has one side, so the START ON FRONT SIDES area is selected by default. (Had we selected the LineDef from Vertex 2 to 7, both choices would have been available.) The AUTOALIGN TEXTURES function *will* correct the two-sided LineDefs of our window, however, along the X axis. Click OK.

**Figure 2.36.**  
The AUTOALIGN TEXTURES dialog box is summoned by pressing A.



- 2 Right-click on the two-sided LineDef from Vertex 2 to 7. Click the `SIDEDEFS` tab. You'll see that the `TEXTURE OFFSET` for the X axis has been offset 32. Click OK to exit.

Texture alignment is as easy as that. If you were to check the LineDefs inside the window, you'd see that they've all been given the proper offset. In fact, the `AUTOALIGN TEXTURES` function has aligned every texture in the entire level, since `BRICK6` is contiguous throughout. Of course, as you build your level, you'll need to consider changes that have offset texture alignment, so it's really best to align your textures either a room at a time, or when the level architecture is finished.

When we get to the chapter on `3D EDIT MODE`, you'll find that texture alignment is just as easy. And you'll be able to see the changes dynamically.

But Doom Builder has one more alignment tool up its sleeve: the `CHANGE TEXTURE OFFSET`.

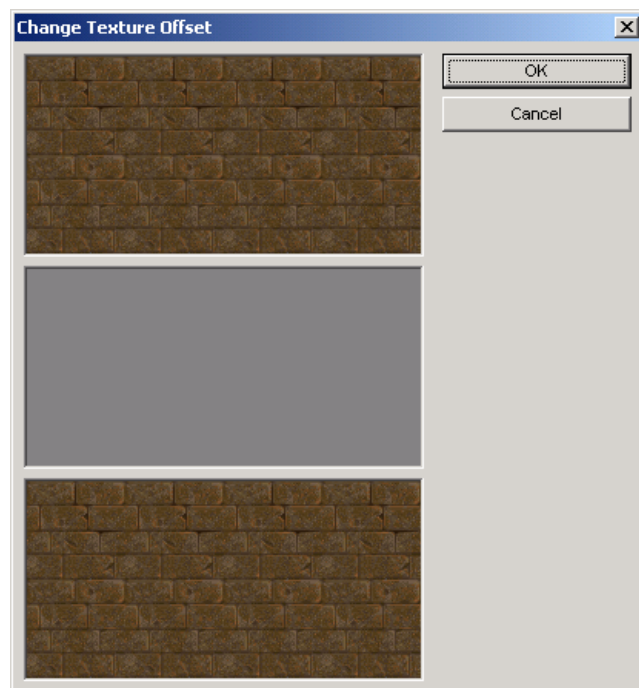
#### 2.3.2.4 The Visual Offset Tool

You really have no excuse for not having your textures aligned if you're using Doom Builder. It has just too many ways to get the job done. An easy-to-miss tool is the `CHANGE TEXTURE OFFSET` function, found in the `EDIT LINEDEF SELECTION` dialog box. Let's have a quick look.

- 1 Enter `LINEDEF MODE`. Right-click on the LineDef of our window Sector. Click on the `SIDEDEFS` tab. In the `FRONT SIDE` area underneath the X and Y `AXES OFFSET` text boxes is the `VISUAL OFFSET` button. Click on it to bring up the `CHANGE TEXTURE OFFSET` dialog box.

**Figure 2.37.**

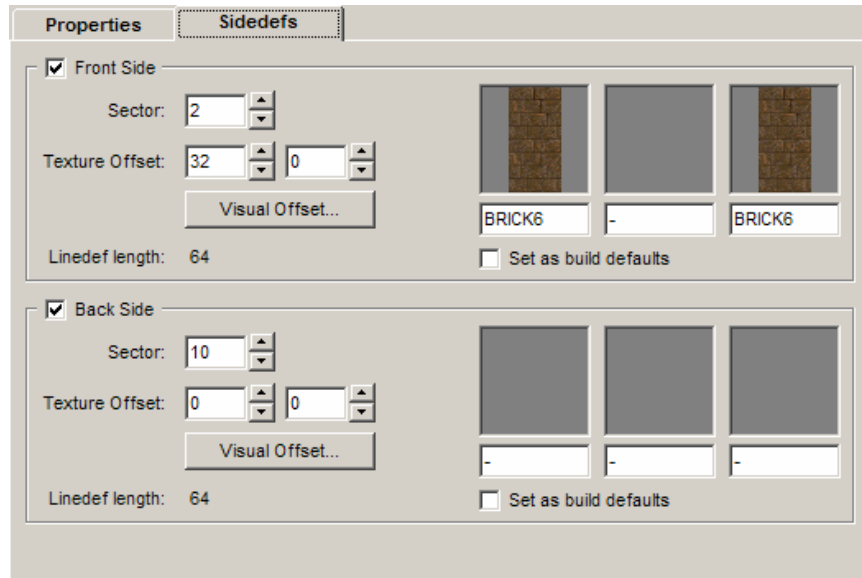
The `CHANGE TEXTURE OFFSET` tool can be accessed through the `SIDEDEFS` tab of the `EDIT LINEDEF SELECTION` dialog box.



- 2 This is a simple visual tool that allows you to manually adjust the textures with your mouse. If you place your cursor over the *upper texture*, you'll see that the cursor changes into cross-arrows. Just left-click and hold down the mouse button and drag the texture around. Don't worry about messing up the offset. Now click OK.



**Figure 2.38.**  
The offsets have been changed for the X and Y-axes.



Back in the `SIDEDEFS` tab, you'll see that the `X OFFSET` is changed. We didn't really need to do any aligning here, but you can see where this tool might come in handy for quick changes.

- 3 Click `CANCEL` instead of `OK` so that the offsets are discarded.

So far you've got the `AUTOALIGN TEXTURES` function and the `CHANGE TEXTURE OFFSET` function. (You have your brain, too. Figuring out offsets isn't that difficult.) When you get to `3D EDIT MODE`, you'll find just how quick and easy texture alignment can be in Doom Builder.

### 2.3.2.5 See-Through Textures

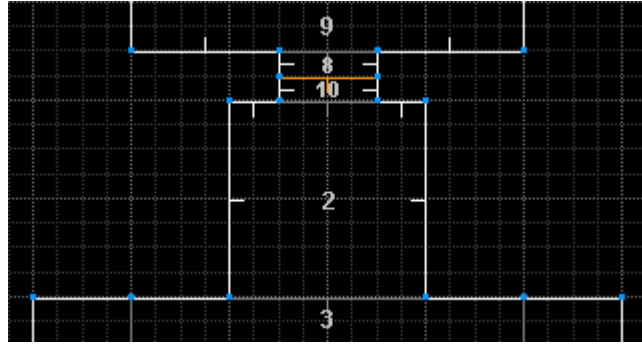
*See-through textures* are textures composed of only one patch, usually a bar, grate, or grid design with transparent gaps, which is placed on a two-sided, transparent `LineDef`. (Some people call these *transparent textures*, but this is incorrect. *Transparent textures* are **no textures** at all.) We're placing such a texture between the floor and ceiling of our window Sector, which we purposely split in the previous chapter with a two-sided `LineDef` (see Figure 2.39).

Most textures are composed of separate patches, which are just graphics positioned at specific coordinates for texture mapping purposes. You may have noticed that switch textures are a composite of two patches. For instance, `SW1LION` and `SW1SATYR` are composed of the `METAL` texture with a separate graphic of the lion or satyr face overlaid. Many other non-switch textures appear to be single textures, but are actually composed of several patches.

See-through textures are composed of only one patch. We're going to use one of the bar types called `MIDBARS3`.

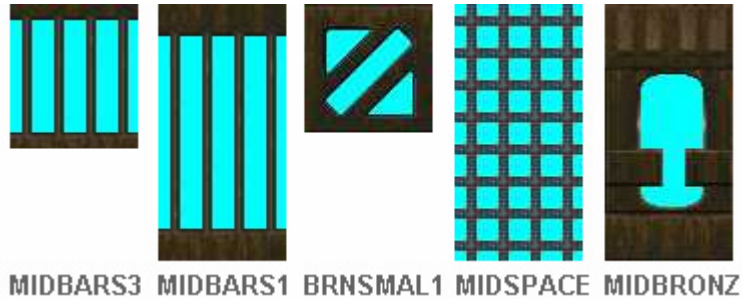
- 1 Enter `LINEDEF MODE`. Right-click on the `LineDef` shared by Sectors 8 and 10 (see Figure 2.39). Click on the `SIDEDEFS` tab of the `EDIT LINEDEF SELECTION` dialog box. Then click on the `FRONT SIDE` (or `FIRST SIDEDEF`) `MIDDLE TEXTURE` image area and select **`MIDBARS3`** in the `SELECT TEXTURE` dialog box, or enter `MIDBARS3` in the text box.

**Figure 2.39.**  
See-through textures are for two-sided LineDefs.



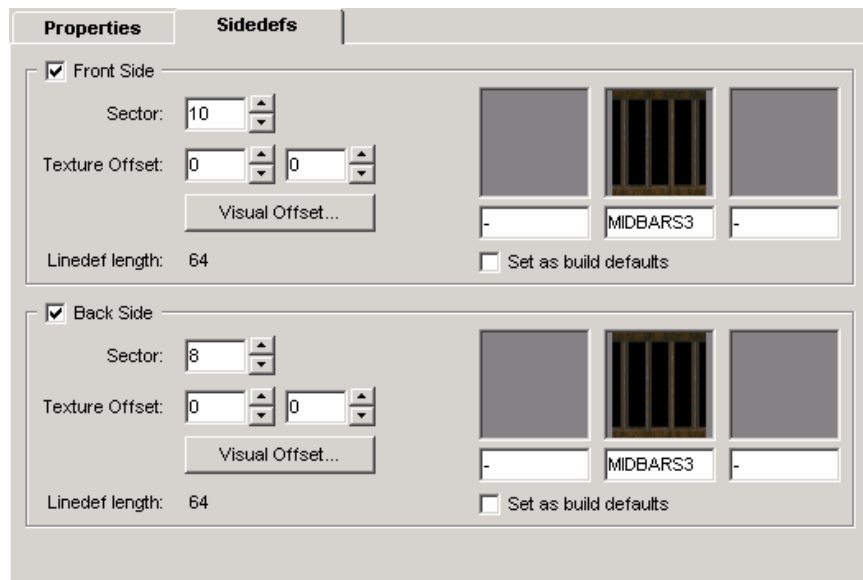
Now we must do the same with the BACK SIDE (SECOND SIDEDEF)! That's because textures can only be seen from one side of the LineDef. If we only placed the bars on the FIRST SIDEDEF, you'd only see the texture from the right side of the LineDef – that is, from Sector 10 only. Anyone in Sector 8 would see an empty window.

**Figure 2.40.**  
Typical see-through textures in DOOM.



- 2 Enter **MIDBARS3** in the MIDDLE TEXTURE text box of the BACK SIDE (SECOND SIDEDEF). Click on the PROPERTIES tab and select the check box next to IMPASSABLE. Click OK to exit.

**Figure 2.41.**  
Assign the MIDBARS3 texture to both the FIRST and SECOND SIDEDEFS.





Now, since this Sector is 32 pixels high and not low enough (normally) for a player to access, we might not have bothered to set the flag in the `PROPERTIES` tab for `IMPASSABLE`. But because DOOM ports like `JDOOM` and `zDOOM` allow jumping, we want to set the flag anyway. It doesn't take a lot of extra effort to set a `LineDef` `IMPASSABLE` that we don't want anyone crossing.

Let's have a look at our bars texture in `3D EDIT MODE` in Figure 2.42.

**Figure 2.42.**

See-through textures in `3D EDIT MODE`.



Note that the texture is slightly misaligned. In texture viewers, the `MIDBARS3` texture looks as though it has a four pixel metal border along the top and bottom. In reality, the texture is made of a single patch that has a single 8-pixel strip along the top. This has to be manually aligned. We can do it simply enough in `2D EDIT MODE` by entering a `Y OFFSET` of `-4` pixels in the `SIDEDEFS` tab of the `EDIT LINEDEF SELECTION` dialog box. Or we can fix it with the arrow keys in `3D EDIT MODE`. (The `AUTOALIGN TEXTURES` function will not work because the texture is not in relationship to another `MIDBARS3` texture.)

It's always a good idea to make texture alignment changes when you notice them, otherwise you may forget.

- 1 Right-click on `LineDef` 47. Click the `SIDEDEFS` tab. Enter `-4` in the second (`Y`) `OFFSET` text box for both the `FRONT SIDE` and the `BACK SIDE`. Click `OK`.

See-through textures add a greater sense of being in a 3D environment, so try them out in your future levels.

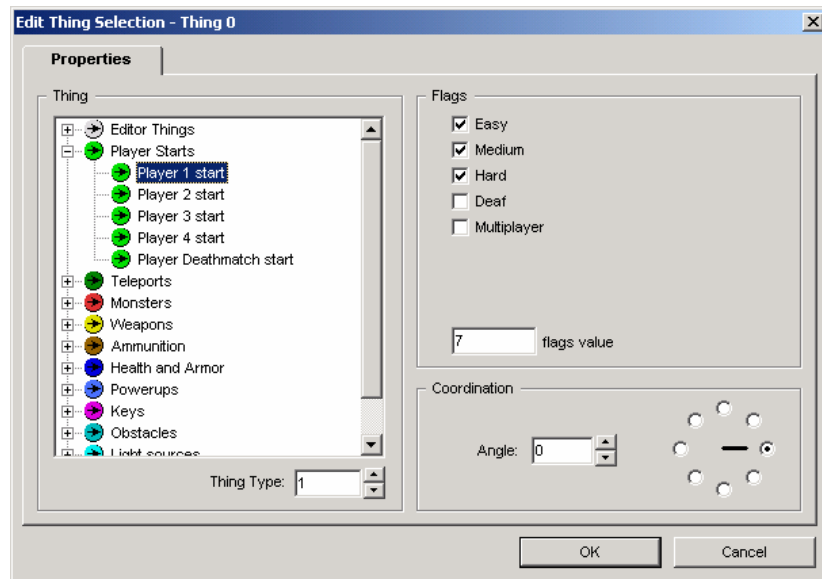
Let's complete our examination of the editing dialog boxes that allow us to change the properties and attributes of various objects with the third and last: the `EDIT THING SELECTION` dialog box.

### 2.3.3 The Edit Thing Selection Dialog Box

Things include all of the monsters, decorations, ammo, weapons, light sources, etc. Before you can even play your map, you need at least one Thing in it: a `PLAYER 1 START`. I recommend placing at least four player starts as well as four multiplayer starts in your levels: people like playing `DeathMatch` and your map will be more attractive to folks if they're able to `DeathMatch` in it. (Things are not visible in `3D EDIT MODE`; therefore, they can't be edited there.)

- 1 Enter `THINGS MODE (T)`. Position your cursor in the middle of Sector 1. Right-click. `DB` inserts a `PLAYER 1 START` by default, so we don't need to change this.
- 2 Position your cursor in the center of Sector 5. Right-click. Now right-click again on the Thing to bring up the `EDIT THING SELECTION` dialog box. Things are listed in a categorized tree; the image marker they use to represent themselves in the map is shown next to their category. Click on the `+` symbol to expand the category `MONSTERS`. Select `IMP` from the list.

**Figure 2.43.**  
The `EDIT THING SELECTION` dialog box comes up by default when you insert a Thing.



- 3 In the `COORDINATION` area, click the option button to give the `Imp` a west-facing `ANGLE` of `180°`. You could also enter this number in the `ANGLE` text box, or click the up-down arrows for `45°` increments.

A Thing has a `COORDINATION ANGLE` (east, south-east, south, south-west, etc.) which indicates what direction it is facing. This really only applies to your *player starts* and *monsters*, whose sprite frames are different for various angles. All other Things always look the same from any angle. (`JDOOM` and other `OpenGL DOOM` ports may use polygonal `3D` models for decorations, pickups, weapons, decorations, etc., so this isn't strictly true. Something to keep in mind.)

A Thing has five `FLAGS` to indicate which `SKILL LEVEL(s)` it appears in during game play, its state (sometimes called the `AMBUSH bit`), and whether it appears in `DeathMatch (MULTIPLAYER)`:

- **EASY** – `SKILL 1` and `2`: *I'm Too Young To Die* and *Hey, Not Too Rough*
- **MEDIUM** – `SKILL 3`: *Hurt Me Plenty*
- **HARD** – `SKILL 4` and `5`: *Ultra-Violence* and *Nightmare*
- **DEAF** – A monster will not react until it sees the player
- **MULTIPLAYER** – The Thing is present **only** in `DeathMatch` and/or `COOPERATIVE` play

All three skill levels are selected by default; but it's a good idea to implement the difficulty levels to make your map as diverse and challenging as possible. This usually means more monsters and less health and ammo the more difficult the skill level. (It also means that a seasoned DOOM player doesn't expect to confront a Cyberdemon if he's chosen *Hey, Not Too Rough*. You may want to tag the Cyberdemon as **HARD** only, and place a Baron of Hell in the same room tagged **MEDIUM**, and an Imp – or nothing – for **EASY**. Likewise, you may want a super-weapon such as the BFG to appear only under the most difficult skill levels and substitute a Super-Shotgun for easier skill levels. You can imagine now just how involved, complex, and thoughtful good level design can get.)



Remember that the **THING FILTER** button in the **TOOLBAR** can be used to sort what Things are visible in DB. You can use this to keep track of *which* Things have *what* FLAG.

The **FLAGS VALUE** is a number based on the combination of **FLAGS** chosen. The **THING TYPE** area is the object's **TYPE** number.

Remember that the categorized tree list can be changed to an alphabetical list in the **INTERFACE** tab of your **CONFIGURATION** settings (F5).

### 2.3.3.1 Thing Sizes

Another feature of Doom Builder that isn't readily apparent, but for which no small effort has been extended, is the matter of a Thing's **SIZE**. If you highlight the **PLAYER 1 START** and the **IMP** you might not notice the difference, but their bounding boxes correctly represent their **SIZES** according to [The Unofficial DOOM Specs](#) by Matt Fell. This is important, because you need to construct your level so that monsters (and the player) can actually negotiate the spaces you've made. That is, you need to make sure monsters can fit through the doors and hallways you've made, if you intend for them to be able to track or chase the player.

A Cyberdemon, for instance, has a radius of 40 grid or pixel units. That means the minimum width he can pass through must be at least 84 pixels (a space must be at least 4 pixels wider than the Thing's entire width for it to pass through). And although we're speaking of an object's radius, remember that a Thing's bounding box is **square**, not round.

For demonstration purposes, let's insert a couple of monsters to give you an idea of what I'm talking about. We'll also find out how to copy and paste Things. Let's look at our map again in Figure 2.44, with the current Things selected showing their bounding boxes. Decrease your grid size to 16.

**Figure 2.44.**

The **PLAYER 1 START** and **IMP** bounding boxes are drawn to represent their exact scale size.



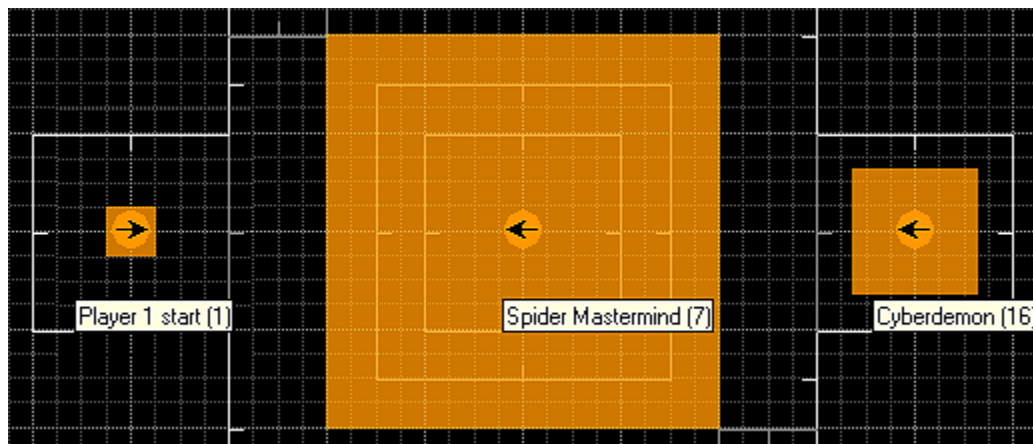
Note that the Imp box [radius 20] is a little bigger than the **PLAYER 1 START** box [radius 16]. Let's look at a more dramatic example of Thing sizes (see Figure 2.45).

- 1 Right-click on the **IMP** to bring up the **EDIT** box and select **CYBERDEMON**. Click **OK**.

- 2 Select the CYBERDEMON (left-click) and press CTRL+C to copy it. Move your cursor to the center of Sector 0. Press CTRL+V. Move your mouse a little and you'll see that the CYBERDEMON marker is dangling at the end of your cursor. This is the copy. Position it in the center of Sector 0 and left-click. You've pasted the CYBERDEMON.

Let's change the Cyberdemon in Sector 5 to the Spider Mastermind.

- 1 Right-click on the CYBERDEMON, then choose SPIDER MASTERMIND and click OK.
- 2 Highlight the marker to show the SPIDER MASTERMIND's bounding box. His radius is 128.



**Figure 2.45.**

To copy a Thing, select the object, press CTRL+C. To paste, press CTRL+V.

As mentioned, seeing the actual sizes is quite useful for determining whether a monster will fit through a particular doorway. It's also useful to make sure you don't have monsters stuck inside walls. If a monster is stuck inside a wall, you may be able to see it, but it won't be able to move. Make sure your bounding boxes are well within the LineDefs of the Sector in which you place your Thing. See the following table for the height and width of monsters in DOOM.

	Actual		Minimum Clearance	
Object	Height	Width	Height	Width
Player	56	32	56	33
Trooper/Sergeant	56	40	56	44
Heavy Weapon Dude	56	40	56	44
Wolfenstein SS	56	40	56	44
Imp	56	40	56	44
Demon/Spectre	56	60	56	64
Cacodemon	56	62	56	64
Lost Soul	56	32	56	36
Hell Knight	64	48	56	52
Baron of Hell	64	48	64	52
Arachnotron	64	128	64	132
Pain Elemental	56	62	56	66
Revenant	56	40	56	44
Mancubus	64	96	64	100
Arch-Vile	56	40	56	44
Spider Mastermind	100	256	100	264
Cyberdemon	110	80	110	84

This table is adapted from information in Matt Fell's [The Unofficial DOOM Specs v1.666](#) and Scott Amspoker's [DOOM Metrics](#).

Note that DB can't indicate the height of Things, which will make a difference as well whether monsters can fit through structural elements. It's good to know the radius, but you'll also need to know the heights of Things, which is listed in the table above.

The step distance for players and monsters is generally 24 units. That is, any Sector higher than 24 units is impassible. Note that this doesn't apply to floating monsters.

**NOTE:** There's a neat utility that checks for stuck monsters. It checks Thing heights as well as their radius. It's called `STUCK` and was written by Jim Flynn. You can find it on most DOOM FTP sites (and at [Dr Sleep's DOOM Apothecary](#) in the [Downloads](#) | [Tools](#) section) as [STUCK.ZIP](#).

- 1 Let's get your map back to the way it was. Highlight the `CYBERDEMON` and press `DEL`. Right-click on the `SPIDER MASTERMIND` and change it back to an `IMP`. Click `OK`.

**NOTE:** When you copy a Thing, you're copying all of its properties and attributes as well. Copy and paste is useful for placing multiple Things of the same type instead of having to insert and modify them one after the other.

And now we can get to the chapter you've been waiting for: 3D Edit Mode.

## 2.4 3D Edit Mode

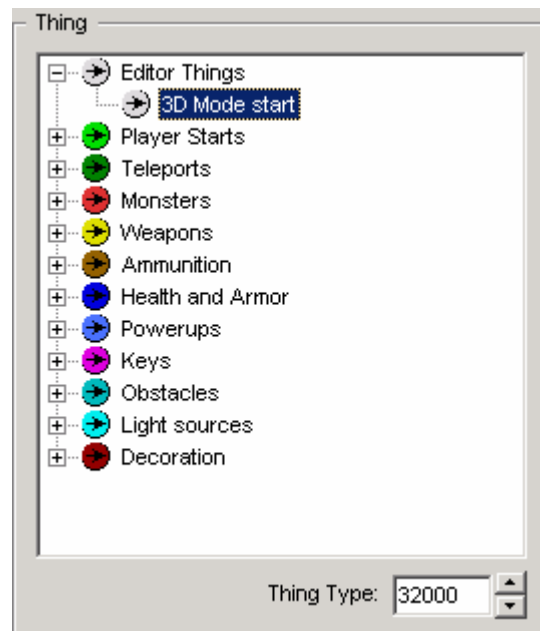
One of Doom Builder's most unique and helpful features is its **3D EDITING MODE**. We'll explore its many uses in this chapter. Not least among them is the obvious advantage of being able to walk through your level and see it as it will actually look in DOOM. And all without having to go through the time and trouble of leaving the editor, running DOOM, warping to your level, etc. 3D EDIT MODE is most useful for texture placement and alignment, adjusting the light level settings by eye instead of just guessing, and raising or lowering ceiling and floor heights.

Before entering 3D MODE, you may want to place a 3D MODE START object in your map. This is a start position similar to the PLAYER 1 START: wherever you place it, that's where your start position in 3D MODE will be instead of the default PLAYER 1 START. Once it's in your map, you can easily move it around to whatever position you want to be the next time you enter 3D MODE. By default, however, the 3D MODE START will move to the position you were when you left 3D EDIT MODE.

- 1 The easiest way is to press **CTRL+W** in THINGS MODE. This will insert a 3D MODE START or move an existing one to your mouse cursor position. To adjust the angle, however, you still have to enter the EDIT THING SELECTION dialog box.

**Figure 2.46.**

Place a 3D MODE START position in your map with CTRL+W or in the EDIT THING SELECTION dialog box.



- 2 Enter THINGS MODE. Right-click to insert an object, then right-click on it to bring up the EDIT THING SELECTION dialog box. Expand the EDITOR THINGS category and then select 3D MODE START. You can adjust the angle for this object just like any other DOOM Thing. Click OK.

## 2.4.1 The 3D Edit Mode Environment

The 3D EDIT MODE environment is similar to the game play environment in DOOM: in fact, you can't tell the difference, visually. Your level looks exactly as it would look in DOOM, right down to the lighting – except that the sky is not rendered, so you see the ceiling with the sky texture tiled across it. You can toggle **GRAVITY MODE** so that you can walk through your level and go up stairs, as normal. Since there is no BLOCKMAP processing, you are in perpetual NO CLIP MODE: that is, while you won't fall through floors, you can still walk through walls. If you go through a wall with gravity on, you'll fall to the next floor height, even if you're not actually within the map anymore. When gravity is off, you "float" through the level. You can go through floors as well as walls. There are no moving objects in 3D EDIT MODE: doors and lifts don't move, triggers don't work, etc. Also, you cannot see Things: no monsters, decorations, weapons, etc.

3D MODE is purely a visual construct for cosmetic changes. You can't alter the geometry of Vertices, LineDefs or Sectors. What you *can* do, though, is modify anything that doesn't affect the nodes or require a build of the BLOCKMAP.

## 2.4.2 Shortcut Keys

Before we enter 3D MODE, you should write or print a list of the 3D EDIT MODE shortcut keys. They are:

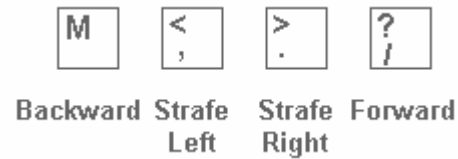
3D Edit Mode Shortcut Keys	
Select Texture	<b>Mouse1</b>
Paste Texture	<b>Mouse2</b>
Copy Texture	<b>Mouse3</b>
Move Texture Up	<b>Arrow Up</b>
Move Texture Down	<b>Arrow Down</b>
Move Texture Left	<b>Arrow Left</b>
Move Texture Right	<b>Arrow Right</b>
Remove Texture	<b>DEL</b>
Auto-Align Textures	<b>A</b>
Toggle Lower Unpegged	<b>L</b>
Toggle Middle Texture	<b>T</b>
Toggle Upper Unpegged	<b>U</b>
Increase Brightness	<b>CTRL+Scroll-Up</b>
Decrease Brightness	<b>CTRL+Scroll-Dn</b>
Lower Floor/Ceiling by 1	<b>SHIFT+Scroll-Dn</b>
Lower Floor/Ceiling by 8	<b>Scroll-Dn</b>
Raise Floor/Ceiling by 1	<b>SHIFT+Scroll-Up</b>
Raise Floor/Ceiling by 8	<b>Scroll-Up</b>
Toggle Gravity	<b>G</b>
Toggle Light	<b>B</b>
Forward	<b>D</b>
Backward	<b>E</b>
Strafe Left	<b>S</b>
Strafe Right	<b>F</b>
Exit 3D Mode	<b>W</b>



To change a shortcut key, press F5 to enter the CONFIGURATION dialog box and go to the SHORTCUT KEYS tab. You can change the movement keys to those you use in DOOM, if you like. The only difference may be that you cannot use the mouse buttons to move forward. I recommend a setup that has all four fingers settled comfortably on the BACKWARD, STRAFE LEFT, STRAFE RIGHT, and FORWARD keys. Make sure you do not choose a set that contains a key already assigned.

**Figure 2.47.**

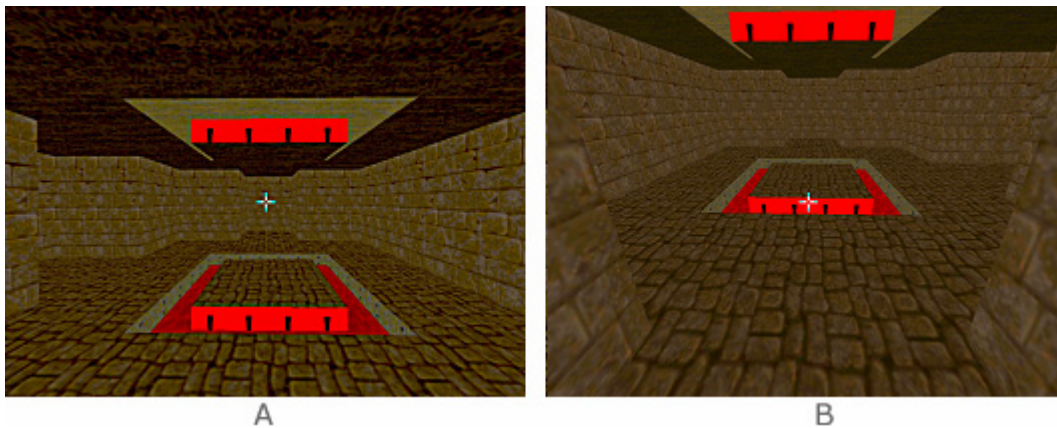
Change your 3D MODE keys to those you would use for game play in DOOM. This key combination (which other players assure me is goofy) is only a suggestion.



### 2.4.3 Modifying Textures and Flats

To modify any object, just position the crosshairs on it and make your key presses. For direction purposes, I'll tell you to **point** at an object. You do this basically by *looking* at an object, moving the mouse as you would with *mouselook* in QUAKE, UNREAL, and other 3D games.

- 1 Press **W** to enter 3D EDIT MODE. The mouse buttons control selecting, copying, and pasting of textures. Let's try copy and paste first. Point at the **METAL** texture on the lower SideDefs inside the blood trench (Figure 2.48b).

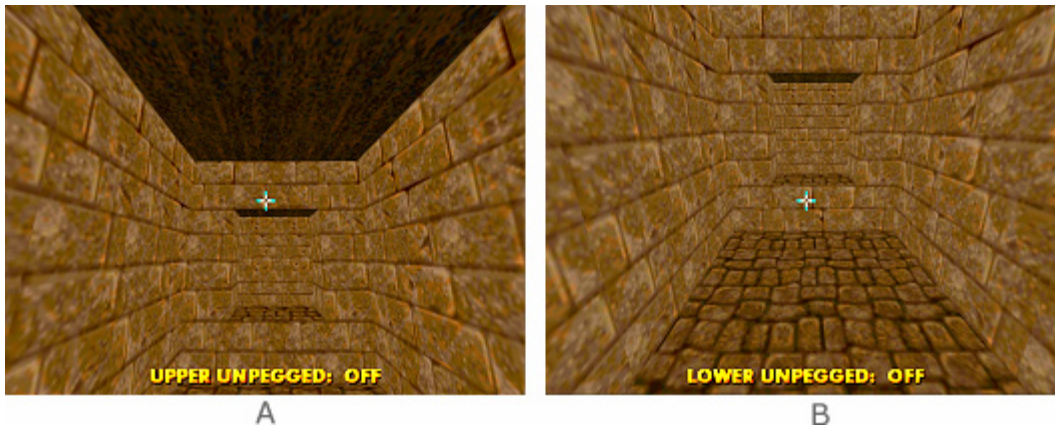


**Figure 2.48.**

Use MOUSE1 to select, MOUSE2 to copy, and MOUSE3 to paste.

- 2 Middle-click (MOUSE2) to copy the texture. Point at the upper missing texture. Press MOUSE3 to paste. Now circle around and paste the METAL texture on all of the upper and lower SideDefs that are missing textures. Note that these upper and lower textures do not need to be unpegged, because the METAL texture is such that no misalignment is noticeable.
- 3 Let's go over to our window Sector. Point to the upper texture above the window. Press **U**. A message UPPER UNPEGGED: OFF will come up, and the texture shifts back to being drawn from the bottom up. It's noticeably misaligned. Let's set it back: press U again. Now point to the lower texture. Press **L** and the texture shifts back to being drawn from the top down. The message LOWER UNPEGGED: OFF comes up. Press L again to unpeg the texture.





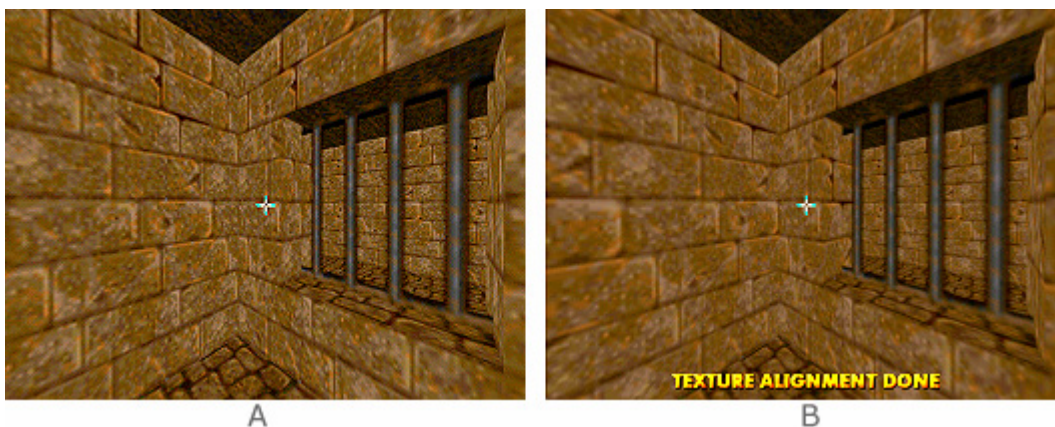
**Figure 2.49.**

Unpegging textures in 3D EDIT MODE. Press U for UPPER and L for LOWER.

That's all there is to it. This is much quicker than the several steps you'd have to go through in 2D EDIT MODE. Plus, you get to see the results of the action as you do it.

Aligning textures is a snap in 3D EDIT MODE. It's also the preferred way to align your textures in Doom Builder. We went through the steps for using the AUTOALIGN TEXTURES function in 2D EDIT MODE in [Chapter 2.3.2.3](#). We undid the changes there so you can see for yourself how it works in 3D MODE.

- 1 Move closer to the window and point to the texture on the left side (Figure 2.50a). Press the **A** key. You should see the textures inside the window, above and below it, and to the right shift as the texture offset for the X axis is changed (Figure 2.50b).



**Figure 2.50.**

The AUTOALIGN TEXTURES function affects all contiguous textures of the same name.

Aligning textures in this fashion will in fact align all like textures that are contiguous with each other throughout the level. Since our entire level is BRICK6 (for the moment), all texture offsets needed have been applied throughout the entire level. (BRICK6 is a texture that aligns very well: in fact, although it's 64 pixels wide, it actually tiles itself at 32 pixels. That is, BRICK6 from pixel x0 to 32 is exactly the same from pixel x33 to 64. I chose this texture for demonstration purposes in our level because it aligns so well. Unfortunately, it's therefore not a good choice for texture alignment purposes. But there are still subtle changes that are noticeable if the texture is misaligned, for instance, inside our window Sector.)

As mentioned earlier, there's no sense in aligning your textures if you plan to make more structural changes to your level. You may want to align textures one room at a time as you finish them, or wait until the entire level is done. (I don't recommend the latter. You always want to do a final sweep of your level to check for misaligned textures, but the job is even more tedious if you haven't done any alignment at all. I suggest aligning room by room as you go.)

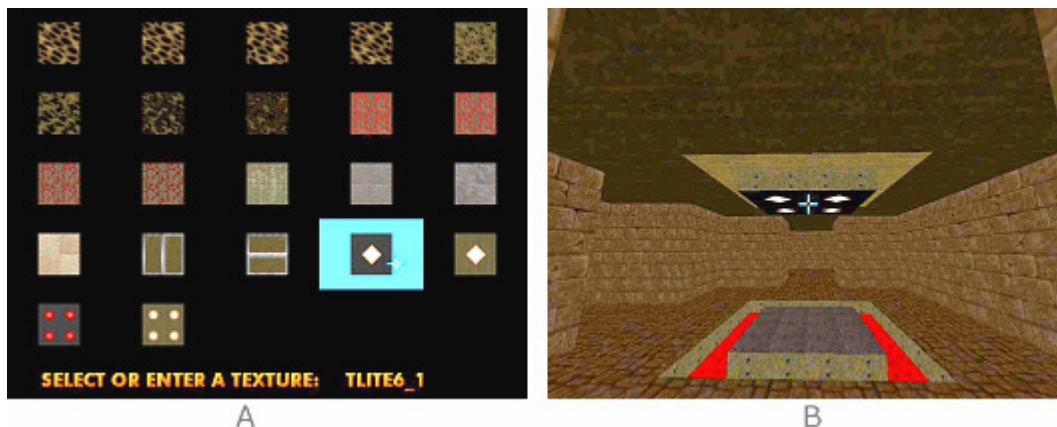
Now let's change some flats.

While we're at the window, you might note that the windowsill, as it were (it's the floor, actually), has the RROCK12 floor texture and doesn't look right. Let's change that to an innocuous flat that goes better with BRICK6. In fact, CEIL5\_2 is just fine.

- 1 Point at the ceiling of the window Sector and middle-click to copy the ceiling texture, **CEIL5\_2**.
- 2 Point at the windowsill and right-click to paste the texture. You'll note that you only have half of the windowsill since the window Sector is split into two.
- 3 Strafe or move into the courtyard to access the second Sector. (You can't point through the MIDBARS3 texture, because even though the texture is transparent, that's just an illusion. This may seem self-evident to some, but I've caught myself a dozen times trying to do this.) Right-click to paste the **CEIL5\_2** texture to the second half of the windowsill.

Let's fix some other areas. Go out into the main Sector 3.

- 1 Point at the ceiling of the platform in the trench (Figure 2.51b). Left-click (MOUSE1) on it to bring up the FLATS VIEWER. Look for **TLITE6\_1**. Note that you can type the name of the flat in the lower right hand area and the viewer will find the texture image (Figure 2.51a). Click on **TLITE6\_1** to select it.



**Figure 2.51.**

Press MOUSE1 (left-click) on any flat to bring up the FLATS VIEWER.

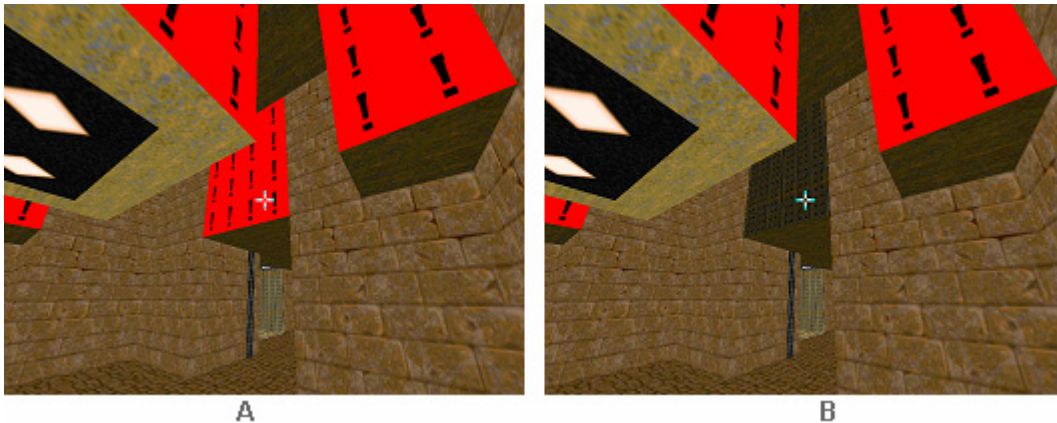
- 2 Point to the floor of Sector 5 and left-click. In the FLATS VIEWER find **SLIME12** and select it.

Don't you wonder how you ever got along in other editors without 3D EDIT MODE? Let's learn how to modify floor and ceiling heights, as well as lighting.

## 2.4.4 Modifying Sector Heights and Lighting

Let's make some simple but drastic changes to the main room of our map. We're going to give this level that unmistakable *Dr Sleep Master Level* look by changing some ceiling and floor heights, a couple flats and textures, and completely transform the current look.

- 1 Enter 3D EDIT MODE. Point to the ceiling of Sector 3, Scroll-Up with the mouse wheel, and raise it to a height of **256**. This has exposed upper textures over the entrances to Sectors 0, 1, 2, 6, and 7 (see Figure 2.52).
- 2 Point to the upper texture over Sector 0 and left-click to bring up the texture viewer. Find (or enter) **SUPPORT3** and select it. Middle-click on the **SUPPORT3** texture to copy it and turn to Sector 2. Point to the upper texture there and right-click, pasting **SUPPORT3**. Do the same with the upper texture over Sector 1, and over the CHILD SECTORS 6 and 7.



**Figure 2.52.**

Upper textures are exposed over Sectors 0, 1, 2, 6, and 7 when we raise the ceiling.

- 3 Point at the floor of Sector 5. Scroll-Down to lower the floor to **-32**. Left-click to bring up the flats viewer. Find and select **FLOOR4\_6**. Middle-click to copy the flat.
- 4 Point at the floor of Sector 4 and right-click to paste FLOOR4\_6 over BLOOD1.

**Figure 2.53.**

A level can be radically altered by changing floor heights and making a few texture changes.





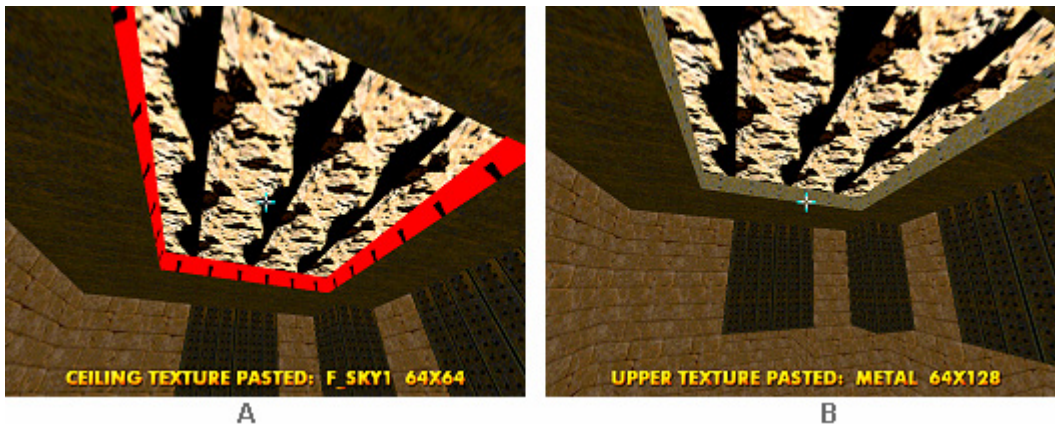
- Lower textures are exposed on the **SECOND SIDEDEFS** of the LineDefs shared by Sector 5 and 4. Point at one of them and left-click to bring up the texture viewer. Find and select **STEP6**. Copy STEP6 and paste it to the other **SECOND SIDEDEFS**.

**Figure 2.54.**

Copy and paste textures when possible, instead of invoking the texture viewer.



- Point at the ceiling of Sector 5. Use Scroll-Up and raise it to a height of **272**.
- Point at the ceiling of Sector 4. Raise it to a height of **240**.
- Point at the missing upper texture on the **SECOND SIDEDEF** of Sector 5 and left-click to bring up the texture viewer. Find **METAL** and select it. Middle-click to copy the METAL texture and then right-click to paste it on the remaining SideDefs. Do this with the other three missing upper textures (Figure 2.55).
- Point at the ceiling of Sector 5 and left-click to bring up the flats viewer. Find and select **F\_SKY1**.



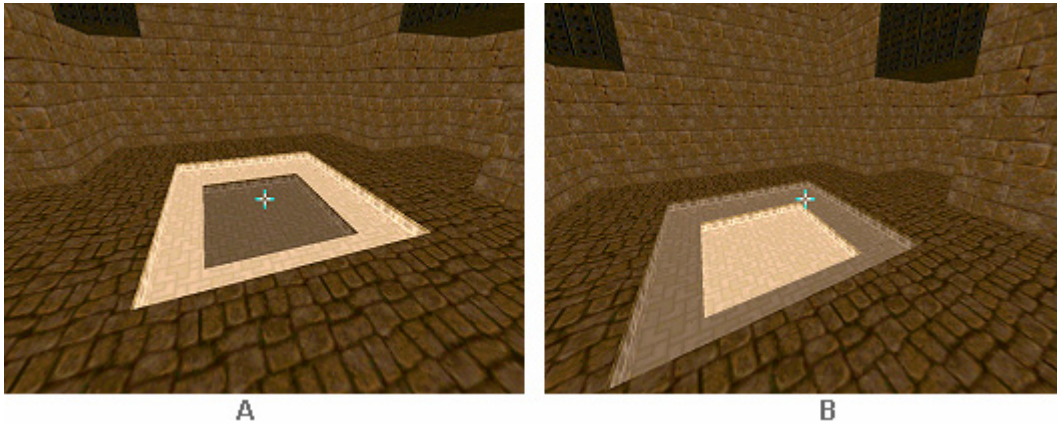
**Figure 2.55.**

When revealing a section of sky in a level, allow an inset of at least 16 pixels to give it the illusion of depth.

Since we've opened up the level to the sky, let's modify the lighting in Sectors 4 and 5.

- Point at the floor of Sector 5. Press **CTRL+Scroll-Up** to increase the brightness to **208**.

2 Point at the floor of Sector 4. Decrease the brightness to **144**.



**Figure 2.56.**

Use CTRL+Scroll-Up/Scroll-Down to increase/decrease brightness levels.

Once the teleporters are raised, textured, and lit, this room will have an entirely different look. We'll be modifying light levels in 3D EDIT MODE a bit more in [Chapter 2.6.1: Making a Teleport Pad in 3D Mode](#), so you'll get to practice some more.

Next, let's learn how to make a door.

## 2.5 Making a Door

Making doors in DB is fairly easy. We'll be using all of the functions we've already learned in creating and adding Sectors, as well as modifying Sector and LineDef properties. A **MANUAL DOOR** is a special door type in DOOM, as opposed to **REMOTE DOORS**, which require a few extra steps. MANUAL DOORS use a LINEDEF ACTION TYPE called a **LOCAL ACTION**. LOCAL ACTIONS require the player to activate an object with the *use* key (usually the spacebar), which is a *switch* activator as opposed to a *walkover* activator. We'll learn more about these terms below, and when we find out how to construct REMOTE DOORS in this section. We'll also be taking our first hard look at the various LINEDEF ACTION TYPES that make DOOM so dynamic.

Let's jump right into it.

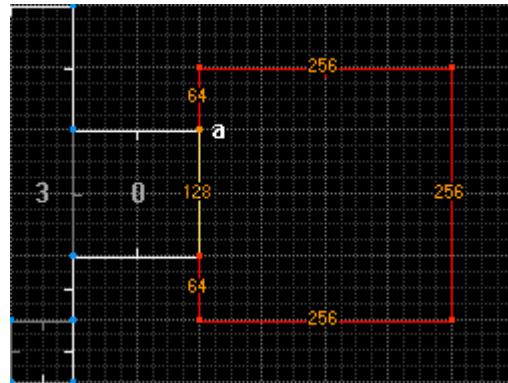
### 2.5.1 Manual Doors

We want to create a 256x256 Sector off Sector 0. Sector 0 will contain our door. (You should know how to draw Sectors by now, so systematic directions will be getting slimmer from here on out.)

- 1 Enter SECTORS MODE. Right-click on Vertex **a** (as shown in Figure 2.57) to begin LINE-DRAW MODE. Draw a 256x256 Sector, ending on Vertex **a**. This will be Sector 11.

**Figure 2.57.**

Begin LINE-DRAW MODE on Vertex **a** and construct a 256x256 Sector.



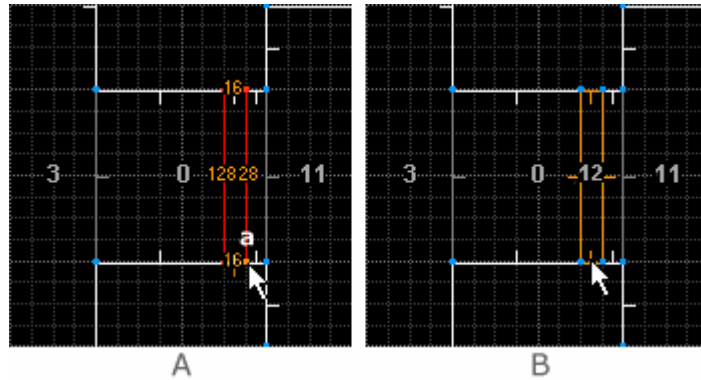
Now let's make our door in Sector 0. Doors are typically 16 units thick. I've mentioned before that you should usually draw Sectors in a *clockwise* direction. This is done so that the FIRST SIDEDEFS inside a new Sector reference that Sector. In other words, their Vectors would all be facing *into* the Sector.

MANUAL DOOR Sectors are different. Their FIRST SIDEDEFS – or RIGHT SIDES – need to be facing *out* and *into* the shared Sectors or the PARENT SECTOR, depending on where they're placed. This is because a MANUAL DOOR uses what's called a LOCAL ACTION, and must be activated from its RIGHT SIDE – or FIRST SIDEDEF. (We'll learn more about this in the chapter on LINEDEF ACTION TYPES.) In our case, the door Sector will be a CHILD SECTOR of Sector 0. Therefore, you'll be drawing *counter-clockwise*.


- 1 Right-click on position **a** as shown in Figure 2.58. Draw up – *counter-clockwise* – and complete the Sector (Figure 2.58a). Your Sector should now have its **FIRST SIDEDEFS** facing *out* (as shown in Figure 2.58b).

**Figure 2.58.**

Draw *counter-clockwise* to construct a manual door Sector so that the **FIRST SIDEDEFS** face “out”.

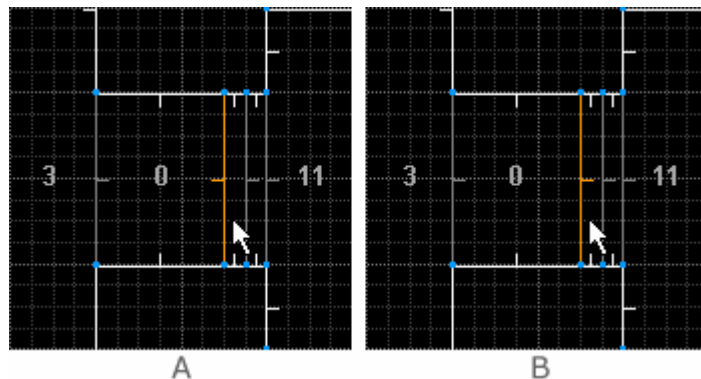


It's easy enough to correct your SideDefs, should you ever forget and draw the door Sector clockwise. You can simply use the **FLIP LINEDEF** function. Let's try it.

- 1 Enter **LINEDEF MODE**. Highlight one of the door LineDefs and press **F**. The Vector flips to the other direction. It's as simple as that. (You can also select the LineDef and click on the **LINES MENU** and choose **FLIP LINEDEF**, or click the **FLIP LINEDEF** button  in the **TOOLBAR**).

**Figure 2.59.**

Highlight a LineDef and press **F**, or select the LineDef and click the **FLIP LINEDEF** button in the **TOOLBAR**.

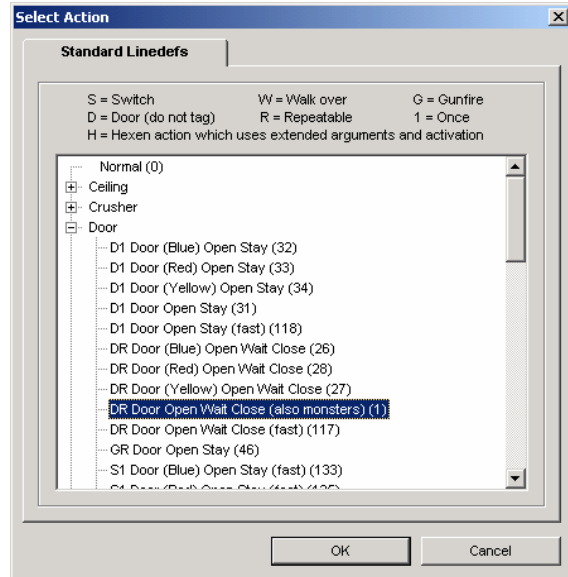


Back to our door. The reason why the **FIRST SIDEDEFS** must face outwards is that we're going to assign a **LOCAL ACTION** to the LineDefs. **LOCAL ACTIONS** are a **LINEDEF ACTION TYPE** that *must be activated by the player on the FIRST SIDEDEF*. (This is not true for LineDefs that are activated by *walking over* them, except for teleporters. **LINEDEF ACTION TYPES** are covered in detail in the next chapter.)

- 1 Select both of the door LineDefs, and then right-click on one of them to bring up the **EDIT LINEDEF SELECTION** dialog box.
- 2 In the **PROPERTIES** tab click on **SELECT ACTION**. The **SELECT ACTION** dialog box comes up. Expand the **DOORS** category and select **DR DOOR OPEN WAIT CLOSE (ALSO MONSTERS) (1)**. Click **OK**. Click **OK** again to exit the dialog box.

**Figure 2.60.**

The SELECT ACTION dialog box contains all of the LINEDEF ACTION TYPES. The categorized tree list can be changed to an alphabetical list in the INTERFACE tab of the CONFIGURATION settings (F5).



If you look at your door Sector now, you'll see that it is drawn in light green, which DB uses to indicate special two-sided lines. But we're not finished yet.

- 2 Enter SECTORS MODE. Select the door Sector and right-click. In the EDIT SECTOR SELECTION dialog box, change the CEILING HEIGHT to match the FLOOR HEIGHT, which is **0**. Click OK.

**Doors must have the same floor and ceiling height.** That height should usually match the FLOOR HEIGHT of the PARENT SECTOR or the Sector the door is being accessed from. That's because a door is a moving Sector that is in the *down* position. Our map so far has a floor height of 0 and a ceiling height of 128. The door Sector therefore needs to have the ceiling height at 0 so that the Sector is initially in the *down* position. When activated, the door will rise to a position 4 pixels short of the ceiling height. But we're still not finished.

- 3 Enter LINEDEF MODE. Since a door is a moving Sector in the *down* position, the LineDefs will have *upper textures*. We need to assign some. Select both LineDefs and right-click to bring up the EDIT LINEDEF SELECTION dialog box. In the SIDEDEFS tab, click on the FRONT SIDE upper texture. In the SELECT TEXTURE dialog box, choose **BIGDOOR4** and click SELECT. Then click OK.

We still have one more chore. Doors have unusual properties. The sides of the doors - the door tracks - will tend to move upwards with the door when it is opened. This looks goofy, and I'm sure you've seen it happen in some amateur levels. That's because DOOM renders normal textures from the top down, and as the adjacent Sector rises, the textures appear to rise with it. The way to prevent this is to set the UNPEGGED attribute we've already learned about.

So let's go back and put textures and FLAGS where we need them.


- 4 Select both side LineDefs - the door tracks - of the door Sector and then right-click on them. In the PROPERTIES tab, select the LOWER UNPEGGED check box in the FLAGS area. In the SIDEDEFS tab, change the FRONT SIDE middle texture BRICK6 to **DOORTRAK** (practice your spelling by entering the name in the text box). Click OK.

*Voila!* You've completed a door.

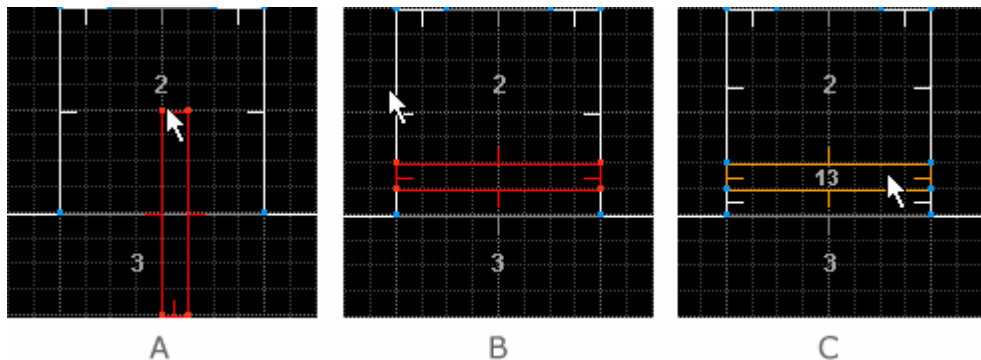


## 2.5.2 Remote Doors

A REMOTE DOOR is a Sector that is opened by a *switch* or *trigger* (repeatable or non-repeatable). You've played levels where you try to open a door and a message comes up that says "This door is opened elsewhere," and later found the switch that opens it. Or you've walked over an invisible LineDef and had a door open without your having to manually activate anything. These are REMOTE DOORS. They're easy to make, and I'll show you another neat function for making the required Sector quickly using the copy, paste, and rotate functions.

- 1 Enter SECTORS MODE. Highlight the manual door, Sector 12. Press **CTRL+C**.
- 2 Move the cursor to Sector 2 and press **CTRL+V**. The copied Sector dangles at the end of your cursor (see Figure 2.61a). We need to rotate the Sector so that it fits into Sector 12.
- 3 Click the ROTATE SELECTION button  on the TOOLBAR (don't worry: the Sector will stop at the edge of the screen and wait for your cursor to return). Enter **90** in the ROTATE SELECTION dialog box and click OK.
- 4 Position the Sector as shown in Figure 2.61b and then left-click to anchor it. Figure 2.61c shows the successfully transferred Sector.

**Figure 2.61.**  
Use copy and paste to make new Sectors.



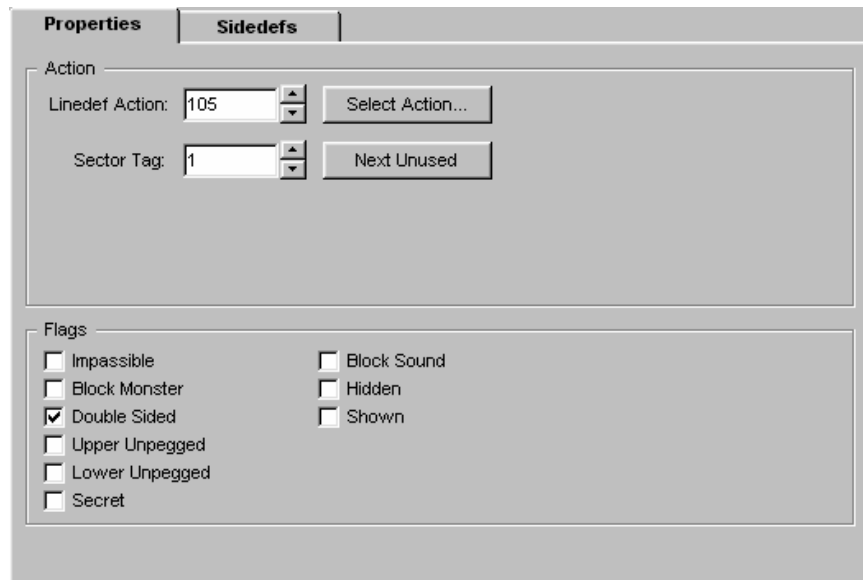
Now you need to assign a LINEDEF ACTION TYPE to the LineDef in front of the door. This will be a *walkover trigger*. When the player walks over the LineDef, the door will open. (This is a perfect example of a REMOTE ACTION.)

- 1 Enter LINEDEF MODE. Right-click on the LineDef (12) in front of the remote door to bring up the EDIT LINEDEF SELECTION dialog box.
- 2 Click the SELECT ACTION button to bring up the SELECT ACTION dialog box. In the STANDARD LINEDEFS tab, expand the DOORS category and go down the list (alphabetized by ACTIVATION TYPES: Dx, Gx, Sx, Wx).
- 3 Select **WR DOOR OPEN WAIT CLOSE (FAST) (105)**. Click OK.

- 4 In the PROPERTIES tab, click the NEXT UNUSED button (or enter 1 in the text box) to set a TAG number for the LineDef. Click OK to exit the dialog box.

**Figure 2.62.**

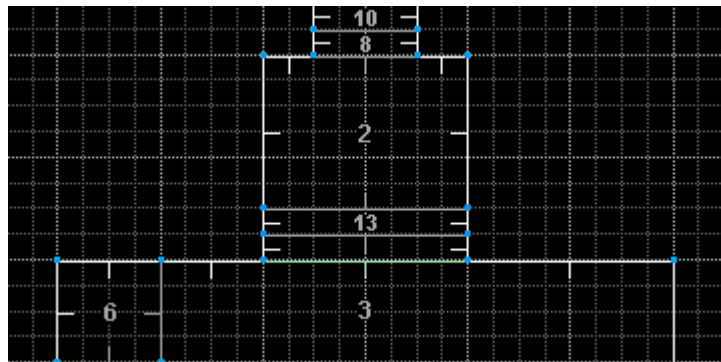
The NEXT UNUSED button will find the next available TAG number.



The LineDef now has an ACTION TYPE and a TAG. What you've done is define an action to be performed by a particular Sector. The LineDef needs to know upon which Sector it is acting. So the TAG you gave to the LineDef must now also be assigned to the Sector you want to have perform the action.

**Figure 2.63.**

The LineDef is drawn in light green to indicate a special line.



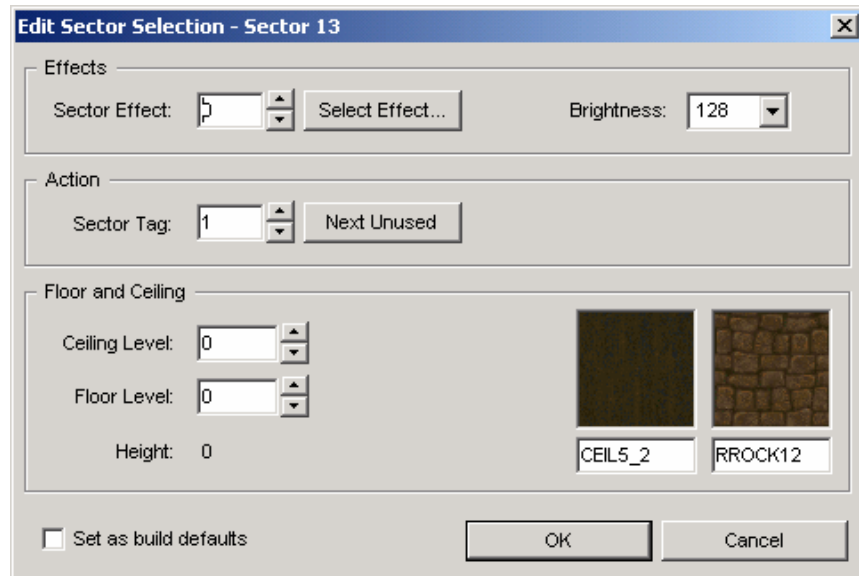
If you have a look at the map now, you'll see that the LineDef is now drawn in light green, indicating that it has a special function.

When the LineDef is activated – in this case, when you *walkover* it – the action is transferred by checking the TAG and looking for Sectors with the same TAG. So: when you walk over the LineDef in front of the door, the door will OPEN, WAIT for 4 seconds, and then CLOSE FAST (this is a *turbo* door). But before that can happen, we have to assign a TAG number to Sector 13.

- 1 Enter SECTORS MODE. Right-click on Sector 13 to bring up the EDIT SECTOR SELECTION dialog box. In the ACTION area, enter 1 in the SECTOR TAG text box. **Do not click NEXT UNUSED.**

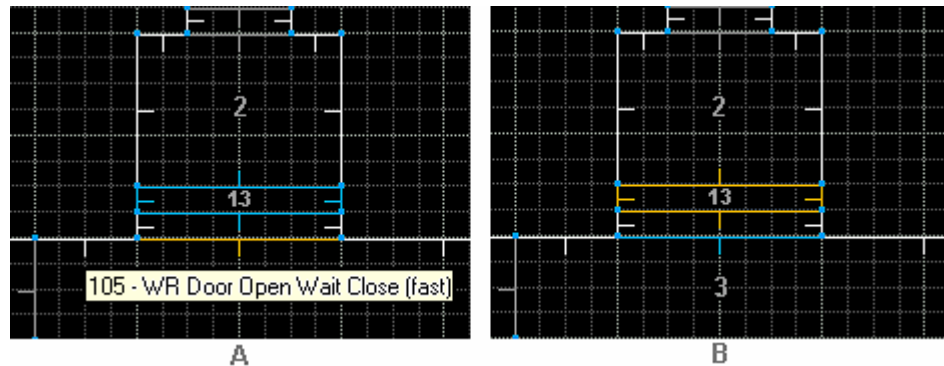
This would select the number 2, since we already selected NEXT UNUSED in the EDIT LINEDEF SELECTION dialog box. It can only be used once per TAG (it doesn't matter which edit box you use first). Click OK.

**Figure 2.64.**  
You can only use the NEXT UNUSED feature once per tag.



Let's look at the map. If you highlight the LineDef, the Sector will be now highlighted in sky blue (Figure 2.65a), indicating that they are now associated with each other by their TAG number.

**Figure 2.65.**  
The Sector and LineDef now have the same TAG number.



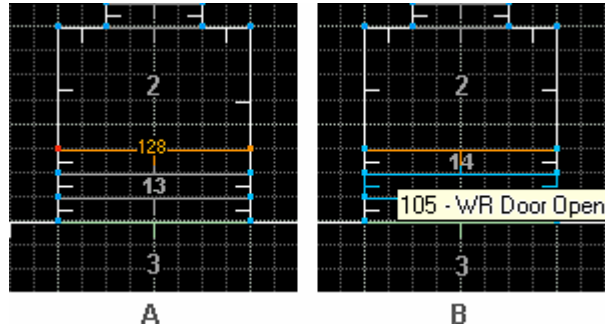
- 1 Enter SECTORS MODE and highlight the Sector: the LineDef will light up in sky blue (Figure 2.65b).

This is a good way to check to see if you have the right Sectors and LineDefs tagged correctly. The remote door is complete, but we must remember that a player going through this door will need a way to get back out. He can't activate the door itself, so we need to place a trigger on the other side of the door in Sector 2 as well.

- 2 Stay in SECTORS MODE. Decrease your grid scale to 16. Right-click on the left edge of Sector 2 (see Figure 2.66a) 16 units up from the door Sector to start LINE-DRAW MODE. Left-click to anchor the LineDef on the opposite side, then right-click to end LINE-DRAW MODE.

**Figure 2.66.**

Add a trigger to open the door from inside Sector 2.

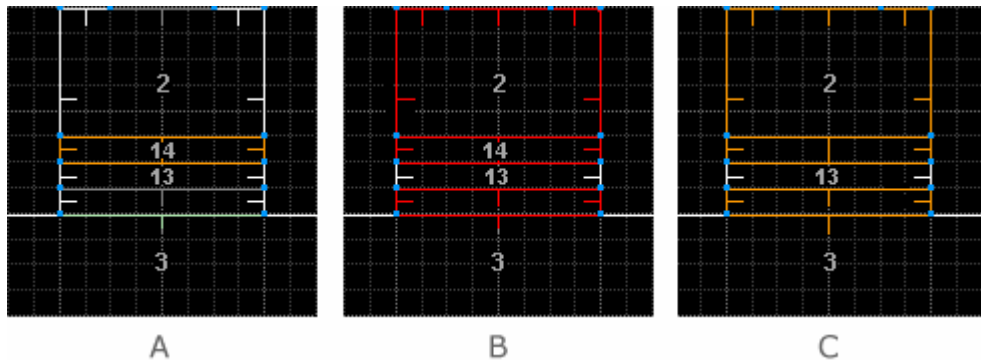



- 1 Enter LINEDEF MODE. Right-click on the new LineDef to bring up the EDIT LINEDEF SELECTION dialog box. Enter a TAG number of 1. Click SELECT ACTION, go to the DOORS category, and choose **WR DOOR OPEN WAIT CLOSE (FAST) (105)**. Click OK, and OK again.
- 2 Highlight the LineDef and DB should light up the door Sector in sky blue and display the LINEDEF ACTION TYPE tool tip (Figure 2.66b).

Adding this LineDef has split Sector 2 and created Sector 14 between the new LineDef and the door Sector (Figure 2.67a). It isn't necessary to have two separate Sectors – and it's always a good idea to have as few Sectors as possible in your map for LOS (line-of-sight) reasons. So let's get rid of Sector 14 by using one of DB's unique features: the JOIN SECTORS function.

**Figure 2.67.**

Select Sector 2 first, then Sector 14. DB will retain the first selected Sector as the main Sector.



- 1 Enter SECTORS MODE. Select first Sector 2, then select Sector 14 (Figure 2.67b).
- 2 Click the JOIN SECTORS button  in the toolbar. Highlight Sector 2 (Figure 2.67c). *Voila!* Sector 14 has been joined with Sector 2.

**NOTE:** Don't forget to assign the DOORTRAK texture to the sides of the door Sector, and remember to set the LOWER UNPEGGED flag for the door track's *normal texture*. And remember to apply an appropriate matching texture to the underside (ceiling) of the door Sector.

Now that you've successfully constructed a remote door using LINEDEF ACTION TYPES and TAGS, let's examine LINEDEF ACTION TYPES a little more closely in the next chapter.

### 2.5.3 LineDef Action Types

The terminology connected with LINEDEF ACTION TYPES is all too often misused, with certain words being interchanged indiscriminately. I'm no pedant, but words mean things, and the only way to mean what you say is to say what you mean. So throughout this manual I've adopted the terminology used by Matt Fell in [The Unofficial DOOM Specs](#), Jim Flynn's [LineDef Action Type Reference](#) (which I helped collate back in 1994), Raphaël Quinet's DEU Specs, and various newsgroup posts and emails by members of id Software themselves.

**LINEDEF ACTION TYPES** (also called **FUNCTION TYPES**) control various effects like doors opening, ceilings and lifts lowering and raising, etc. There are 143 different types (not counting the many more created for DOOM ports such as zDOOM and jDOOM), many of them performing similar functions but at different speeds or height increments, or *activated* in a different manner.

A LINEDEF ACTION TYPE is assigned to a LineDef and the LineDef is given a **TAG** number. Any Sector given the same TAG number will perform the function dictated by the ACTION TYPE. TAG numbers are not ACTION TYPES, nor are they the LineDef or Sector numbers. They're an arbitrary identification number assigned by the level designer.

LINEDEF ACTION TYPES may be *activated* by the player (or sometimes a monster). The player may activate it either by *using* a **switch** (pressing the spacebar or *use* key), by *walking* over it, or by **gunfire** – shooting at it. The *manual doors* are a special type of their own category.

A LINEDEF ACTION TYPE is either *repeatable* or *non-repeatable*. In Doom Builder, the ACTIVATION TYPES are listed in the STANDARD LINEDEFS tab of the SELECT ACTION dialog box.

Trigger	Activator	Repeatability	Category	Type
<b>S1</b>	Switch	Once	Switch	Local
<b>SR</b>	Switch	Repeatable	Button	Local
<b>W1</b>	Walkover	Once	Trigger	Remote
<b>WR</b>	Walkover	Repeatable	Trigger*	Remote
<b>G1</b>	Gunfire	Once	Impact	Local
<b>GR</b>	Gunfire	Repeatable	Impact	Local
<b>D1</b>	Door (no tag required)	Once	Manual	Local
<b>DR</b>	Door (no tag required)	Repeatable	Manual	Local

A **TRIGGER** is the condition that causes the function to be activated. It's the acronymic symbol for ACTIVATOR (**S**, **W**, **G**, **D**) and REPEATABILITY (**1** and **R**). An **ACTIVATOR** is the event that initiates the LINEDEF ACTION TYPE. **REPEATABILITY** is either *once* or *more than once*. **TYPE** indicates which of two broad functions the LINEDEF ACTION TYPE is: *local* or *remote*. And **CATEGORY** is the term generally accepted and used by all enlightened designers and coders.

*Trigger* is also widely used to indicate a two-sided transparent LineDef laid out across a Sector to act as a "trip-wire" that – when crossed – initiates the ACTION TYPE assigned to it in whatever Sector has been paired with its TAG number. *Trigger* in this context always refers to a *remote action*. *Switch* always refers to a *local action*, even though it may activate a Sector far away.

---

\* Id sometimes refers to this ACTION TYPE category as **Retrigger**.

**NOTE:** There are only two kinds of LINEDEF ACTION TYPES: *local actions* (*switch*) and *remote actions* (*trigger*). These are sometimes referred to by others as *switch-activated* actions and *pressure-activated* actions, but not in this manual.

### 2.5.3.1 Local and Remote Actions

For *local actions*, the LineDef you're activating is usually part of the Sector performing the action. That is, its SECOND SIDEDEF – or LEFT SIDE – is shared by the Sector. This is so because all *local actions* must be activated on the FIRST SIDEDEF – or RIGHT SIDE – of the LineDef. A *local action* is always activated by the spacebar (or default *use* key): therefore, it's usually a switch, button, or manual activator.

*Remote actions* – such as *triggers* (LineDefs that are *walkover*) – may be crossed from either side to activate. This applies to any mover activated by a two-sided *walkover* LineDef. Therefore, just as doors can be *local* or *remote*, lifts can be activated locally – from the FIRST SIDEDEF, or RIGHT SIDE – or remotely – walkover from either side, even if the LineDef is part of the Sector being activated. (The one notable exception to this rule is a **TELEPORTER**. A TELEPORTER **must** be entered from the FIRST SIDEDEF, or RIGHT SIDE. See [Chapter 2.6](#) for more information.)

All actions, *local* and *remote*, require a shared TAG association between the LineDef and the Sector (the *trigger* and the *mover*) – except for one more notable exception: MANUAL doors with the trigger DR or D1.

For instance, the first door we made was a MANUAL (or *local*) door which did not require a TAG. This is a special condition used only for this type of door, indicated by its trigger **DR** or **D1**. It's easy to confuse a MANUAL door with the way a lift works, because in both cases you seem to be activating the Sector itself. That's because these are *local actions*: the LineDef you're activating is part of the Sector (its SECOND SIDEDEF – or left side – is shared by the Sector performing the action). Many lifts require you to stand right up against them and press the spacebar to make them lower. These lifts still need a TAG number. It just so happens that this is a *local action* – so it appears that it operates in the same manner as a MANUAL door.

**NOTE:** All *local actions* must be activated on the FIRST SIDEDEF – or RIGHT SIDE – of the LineDef.

*Remote actions* using *triggers* are not limited to doors and lifts. One of the things that made DOOM so unforgettable (and terrifying) was walking down a hallway and crossing a transparent trigger that opened a secret door or room that suddenly gushed forth a horde of monsters you weren't expecting. Using this type of ambush is always guaranteed to get a frenzied reaction from an unwitting player.

A complete compendium of the LINEDEF ACTION TYPES is provided in [Appendix C: LineDef Action Type Reference](#). This is a must-have reference for the level designer.

Now let's learn how to use LINEDEF ACTION TYPES to make a teleporter.

## 2.6 Making a Teleporter

As you've seen in various DOOM levels, a TELEPORTER transports the player to another part of the same level. A teleporter consists of two components: a LineDef and a Thing. The LineDef is assigned a LINEDEF ACTION TYPE, which is tagged to a destination Sector. The Thing is a TELEPORT DESTINATION that is placed at whatever point in the Sector you want the player to appear. Like a PLAYER 1 START, the TELEPORT DESTINATION has an angle that determines what direction the player is facing when he appears in the Sector.

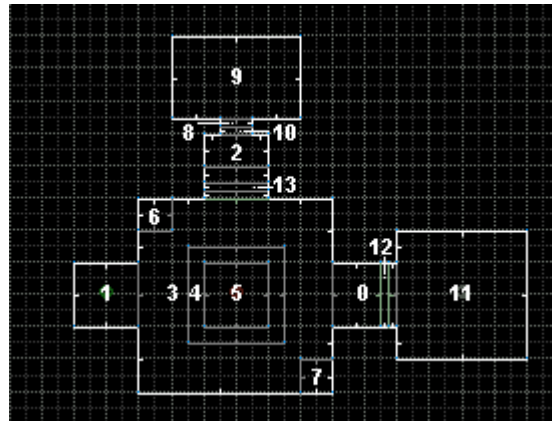
A teleporter doesn't really need a separate Sector to work. The raised square platform with glowing teleporter textures is an illusion in DOOM to give the appearance of a teleporter pad. The actual teleport mechanism is the LineDef or *trigger* that the player walks over. This is best illustrated by MONSTER TELEPORT ONLY triggers, which are just transparent LineDefs with no obvious teleporter pad or gate. We'll create one of these a little later.


Teleporters are actually tricky to make. Many new level designers get it wrong and wonder what it is that they're missing. When we learned about REMOTE ACTIONS in [Chapter 2.5.2](#), I mentioned that the action was triggered no matter which direction the player crossed the line – right or left. But I also mentioned that teleporters are different. The teleporter ACTION TYPE requires the player to enter from the **RIGHT SIDE** – the FIRST SIDEDEF – and won't work from the left. This is why – you may have noticed – that you can step off a teleporter pad in some DOOM levels without getting teleported. But, if you back up or cross the pad from the front, you're teleported. The Sectors we're going to make teleport pads of will need to have their LineDefs flipped so that the FIRST SIDEDEFS face out and *into* their PARENT SECTOR.

We've already created our teleporter Sectors, but now let's add the other components needed to make them work. We'll be working in regular 2D EDIT MODE to assign the LineDef and Thing components, and 3D EDIT MODE to apply the visual look. Let's have a look at our map so far, and then get to work on Sectors 6 and 7.

**Figure 2.68.**

Sectors 6 and 7 will be made into teleport pads.



- 1 Enter LINEDEF MODE. Select the two LineDefs shared by Sector 6 and Sector 3 (see Figure 2.69).
- 2 Press **F** to flip the LineDefs (or click the FLIP LINEDEFS button  in the toolbar). **Teleport LineDefs must be crossed from the RIGHT SIDE (FIRST SIDEDEF).**
- 3 Right-click on the selected LineDefs (we're going to modify both of them at the same time) to bring up the EDIT LINEDEF SELECTION dialog box. Click SELECT ACTION. Expand the

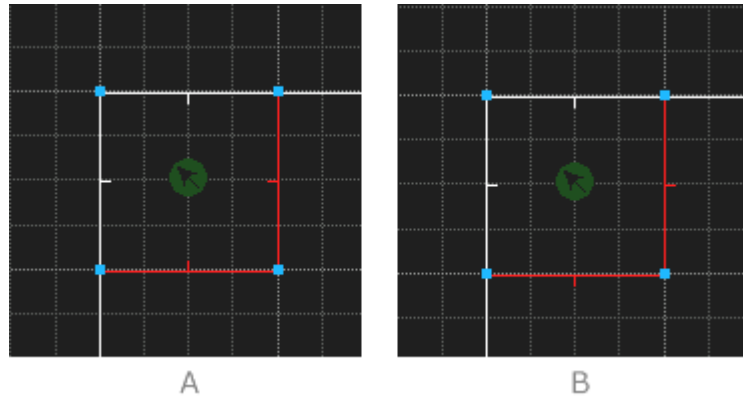


TELEPORT category and select **WR TELEPORT (97)**. Click OK. Now click NEXT UNUSED or enter **2** in the SECTOR TAG text box. Click OK.

Now we need to tag the destination Sector, which is going to be the other teleporter, Sector 7.

**Figure 2.69.**

Press F or click the FLIP LINEDEFS button in the toolbar to switch the FIRST SIDEDEFS.



- 1 Enter SECTORS MODE. Right-click on Sector 7. Enter **2** in the SECTOR TAG text box. Click on SELECT EFFECT and select **8 LIGHT GLOWS (1+SEC)** and click OK. Click OK to exit.

We've tagged the LineDef and destination Sector and given it a pulsating, glow effect. Now we need to add the DESTINATION THING.

- 2 Enter THINGS MODE. Right-click on Sector 7 to insert a Thing, then right-click on it to bring up the EDIT THING SELECTION dialog box. In the TELEPORTS category, select TELEPORT DESTINATION. Set the ANGLE by clicking on the NORTH-WEST option button or enter an angle of 135. Click OK.

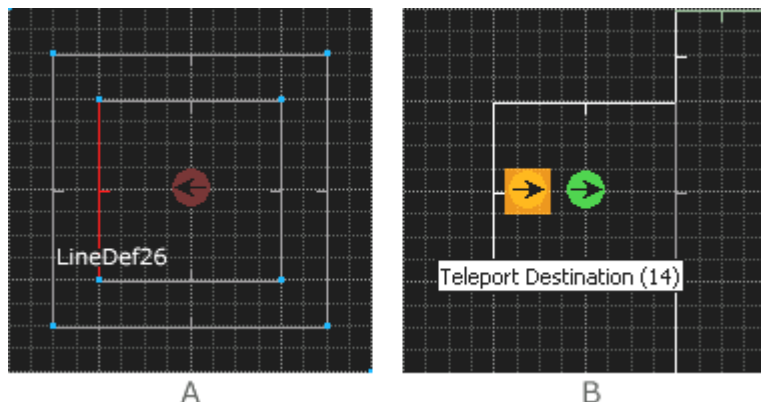
That's it! You can do the same with Sector 7 (assign an ACTION, TAG, and TELEPORT DESTINATION Thing) later. All that's left is to dress up the teleport Sectors and make them look cool. But let's do something else real quick with teleporters first.

There's an Imp in the middle of Sector 5. A neat trick (or a dirty trick, depending on your point of view) is to have the player glimpse a monster when the level first starts, and then have the monster disappear as he spots the player and advances toward him. Better yet, have the monster reappear behind the player. We're going to make a monster teleport.

- 1 Enter LINEDEF MODE. Select LineDef 26 shared between Sectors 4 and 5 (see Figure 2.70). Right-click on it to bring up the EDIT LINEDEF SELECTION dialog box. Click SELECT ACTION, choose **W1 TELEPORT (MONSTERS ONLY) (125)**, click OK. Click NEXT UNUSED or enter **3** in the SECTOR TAG text box and click OK.

**Figure 2.70.**

Set LineDef26 as the teleport trigger and place a TELEPORT DESTINATION Thing in Sector 1.

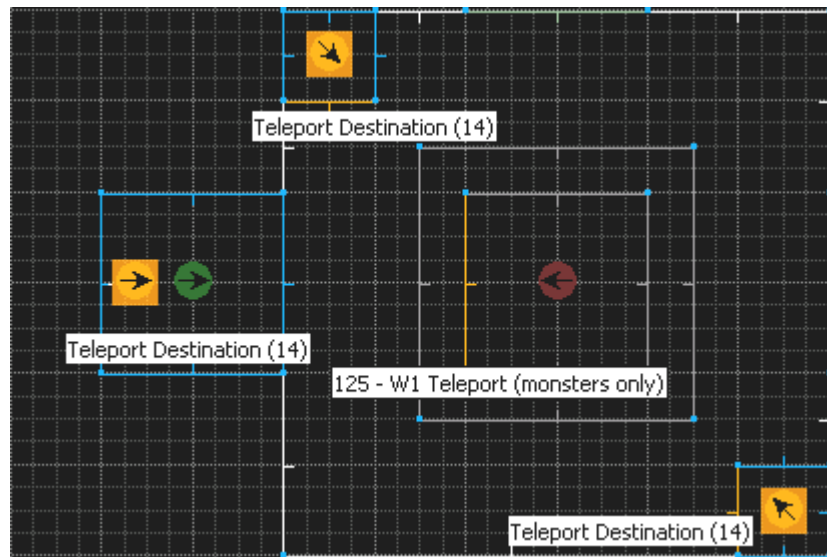




- 2 Enter SECTORS MODE. We want the Imp to teleport right behind the player. You may need to Zoom In so that the mouse doesn't select the PLAYER 1 START. Also, decrease your grid scale to 8. Right-click 40 units behind the PLAYER 1 START to insert a Thing (Imps are a little larger than players, having a radius of 20. If you're standing on or too near the TELEPORT DESTINATION when the Imp gates in, he'll be fragged). Right-click on it to bring up the edit box, and then change the ANGLE to east (or enter 0 in the ANGLE text box). Click OK. That's it!

We chose **W1**, which means this will only work once. We don't want the Imp teleporting every time he crosses that line. And we didn't set all four of the LineDefs with a teleport action, because each one is separate from the rest, and crossing one of them does not inactivate the other three. See Figure 2.71 for a view of the entire teleport setup.

**Figure 2.71.**  
All teleport Sectors and *triggers* diagramed.

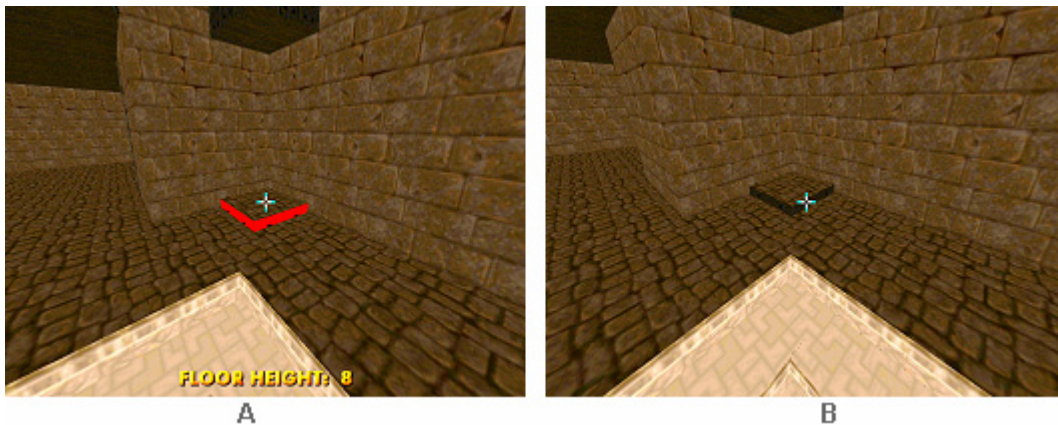


Now we want to go to 3D EDIT MODE and really make our teleporter look like a classic DOOM teleport pad.

## 2.6.1 Making a Teleport Pad in 3D Edit Mode

Teleporter pads traditionally set themselves apart with a raised platform and a slightly lowered ceiling. Add a GATE texture and a light texture above it, some special lighting effects, and you have a convincing teleporter pad. 3D EDIT MODE is ideal for giving your teleporter that authentic look. This will also give you some more practice working in 3D EDIT MODE.

- 1 Enter 3D EDIT MODE. Point to the floor of Sector 6. Use the mouse wheel to Scroll-Up and raise the floor by 8 pixels (Figure 2.72a).



**Figure 2.72.**

Raise the floor height with mouse wheel Scroll-Up. Watch out for missing lower textures.

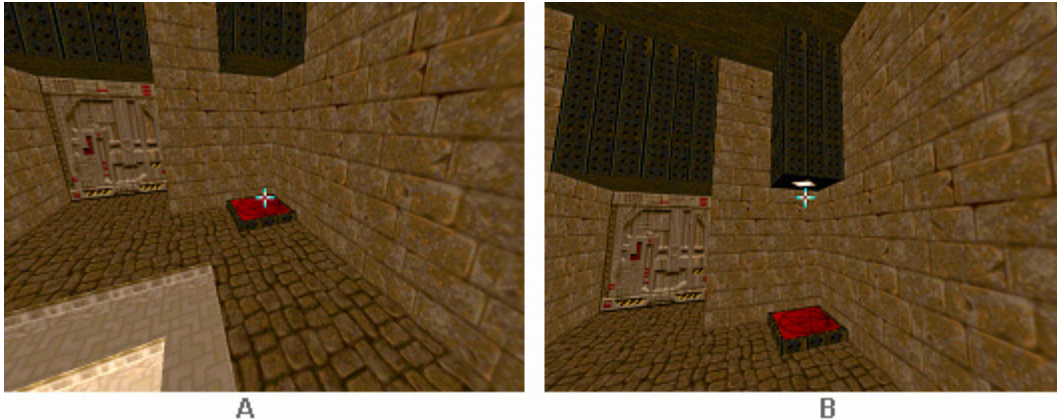
Note that we'll now need some lower textures.

- 2 Point to the lower texture on one of the SideDefs of the teleporter pad and left-click to bring up the texture viewer. Find **METAL** and select it. Copy the METAL texture and paste it to the other SideDef (Figure 2.72b).

Ordinarily, one might want to lower the ceiling 8 pixels over a teleporter pad, but we've already raised the rest of the ceiling around it in an earlier lesson, so a nice overhang already exists.

Now let's change the floor and ceiling flats.

- 1 Point to the floor of Sector 6 and left-click to bring up the flats viewer. Try typing in the name of the flat this time. You have to use **BACKSPACE** first to delete the current name. Then type **GATE3**. When the flats viewer finds the image, select it.
- 2 Now, point to the ceiling, left-click, and find the **TLITE6\_1** texture and select it.



**Figure 2.73.**

Apply one of the GATE flats to give the teleporter that DOOM II look.

It's looking pretty good! But let's explore one more feature of 3D EDIT MODE and use it to dramatize the lighting. We gave the Sector a glow effect, so we need a higher light level to enhance it.

- 1 Point to the floor or ceiling of Sector 6 and then use **CTRL+Scroll-Up**. This increases the light level by 16 increments. Continue until you reach a BRIGHTNESS of **208** (see Figure 2.74). This will really make the teleporter stand out.

**Figure 2.74.**

Press CTRL+Scroll-Up to increase BRIGHTNESS, CTRL+Scroll-Down to decrease BRIGHTNESS.



You should make the same changes to Sector 7. Your teleporters should look consistent throughout a level; make changes to this or that one only if they take the player somewhere special. Using the red GATE textures for regular teleporters and then using the white GATE4 texture on one that leads to a secret area is a nice conceit. Teleporters can also be used to end a level. That is, you might construct something that *looks* like a teleporter, but you'd actually assign the *walkover* LINEDEF ACTION TYPE **W1 EXIT LEVEL (52)**.

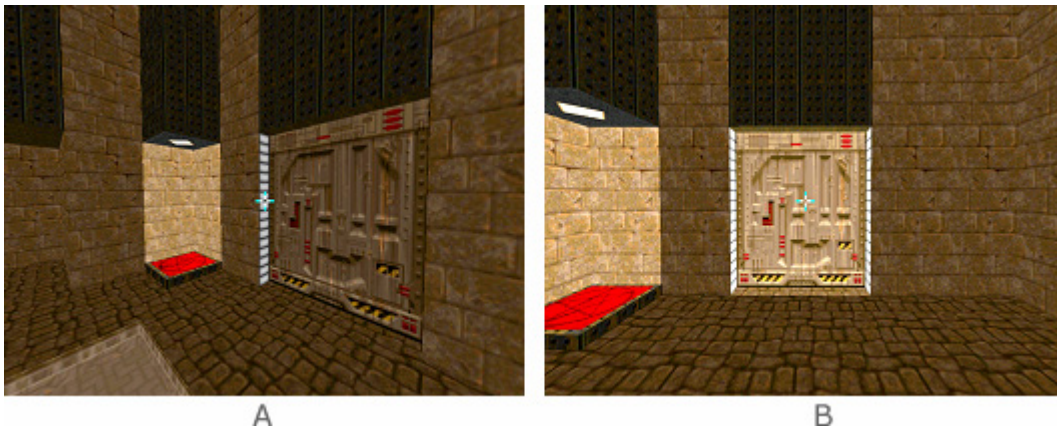
Let's make a couple final adjustments to this room while we're in 3D EDIT MODE.



## 2.6.2 Final Adjustments in 3D Edit Mode

You've seen how changing a few details can radically alter a level's look. Let's make a few lighting changes to the doors while we're in 3D EDIT MODE, just so you can see how a simple, plain level can be jazzed up. We'll do first the remote, and then the manual door.

- 1 Point at the BRICK6 texture inset next to the door (Figure 2.75). Left-click to bring up the texture viewer. Find **LITE5** and select it. Middle-click to copy the texture and paste it on the opposite side of the door.
- 2 Point at the floor of Sector 2 and use CTRL+Scroll-Up to increase the brightness to **192**.



**Figure 2.75.**  
Adding light textures around doors adds drama.

- 3 Point at the ceiling of Sector 0 and left-click to bring up the texture viewer. Find **TLITE6\_6** and select it.
- 4 Use CTRL+Scroll-Up to increase the brightness to **192**.

**Figure 2.76.**  
A look at the result of our minor changes. Pretty nice, huh?



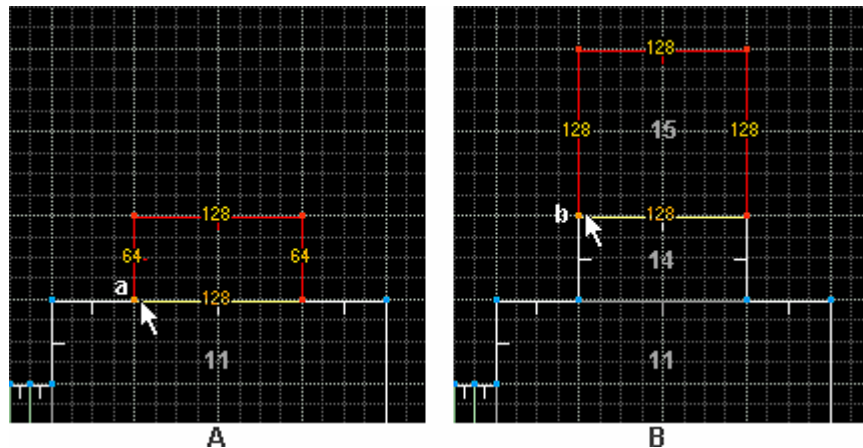
## 2.7 Making a Lift

We're going to have to add a few new Sectors to our map in order to construct a lift. Let's start with the north wall of Sector 11.

- 1 Enter SECTORS MODE. Right-click at position **a** in Figure 2.77a to start LINE-DRAW MODE and complete a 64x128 Sector as shown. This will be the lift.
- 2 Right-click at position **b** as shown in Figure 2.77b and draw a 128x128 Sector. This will be the lower room.

**Figure 2.77.**

Add two Sectors as shown. The first will be the lift connecting Sector 11 to the new lower room.



- 1 Right-click on Sector 14 to bring up the EDIT SECTORS SELECTION dialog box. Let's put a STEP texture on the floor of the lift to give it that *lift* look. Left-click on the floor image to bring up the SELECT FLAT dialog box (or enter **STEP2** in the text box). Find **STEP2** and click SELECT.

Let's give the lift area its own erratic lighting:

- 2 Left-click on the ceiling image and find **TLITE6\_1** and select it.
- 3 Click on SELECT ACTION and choose **1 LIGHT BLINKS (RANDOMLY)** and click OK.
- 4 Change the BRIGHTNESS to **192**. And finally, click NEXT UNUSED (or enter **5**) for the SECTOR TAG.
- 5 Right-click on Sector 15. Change the FLOOR HEIGHT to **-128**, and change the CEILING HEIGHT to **0**.

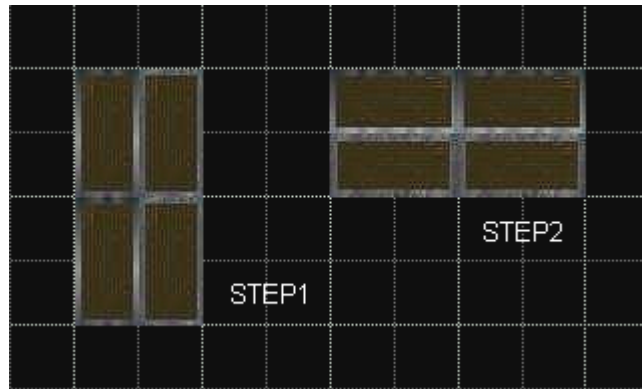
Now we have some upper and lower textures to deal with because of the height difference, as well as the most important thing: a LINEDEF ACTION TYPE.

Attention to this kind of detail is important for keeping the player from being abruptly woken from the continuous fictive dream you've constructed. A DOOM level is an alternate reality in which the player willingly suspends his disbelief to participate. Any error – even one as seemingly slight as a misaligned texture – will bring the player back to reality and make him realize he's only playing a DOOM level: and a sloppy one, at that.

**NOTE:** Draw your lift on the 64x64 grid. Remember that DOOM tiles all flats on this grid scheme. So you should choose a flat that looks like it belongs on a lift platform. STEP1 and STEP2 are made specifically for lifts positioned in one of two areas: STEP1 for east-west and STEP2 for north-south (see Figure 2.78).

**Figure 2.78.**

Use STEP1 for east-west lifts, and STEP2 for north-south.



- 6 Enter **LINEDEF** MODE. Right-click on LineDef 68. Enter **5** in the **SECTOR TAG** text box. Click on **SELECT ACTION** and expand the **LIFT** category. Select **WR LIFT LOWER WAIT RAISE (FAST) (120)**. Click OK.

Now here's something some people forget about (or just don't realize): if you highlight LineDef 68, you'll see that Doom Builder gives you no warning that textures are missing. (That's because it has no way of knowing what you're up to.) But when the lift lowers, LineDef 68 is going to have its **SECOND SIDedef** exposed, and therefore requires a *lower texture*. Let's put textures all around the lift that match and that are appropriate for a lift shaft. We'll use **SUPPORT3** since this level is themed in subdued, earth tones (metal, iron, and brick).

- 7 Click on the **SIDedefs** tab (you're still editing LineDef 68). Enter **SUPPORT3** in the **BACK SIDE lower texture** text box (or click the image and use the texture viewer). Click OK.
- 8 Select both LineDefs 69 and 72. Right-click on one of them and enter the **SIDedefs** tab. Change the **BRICK6** texture to **SUPPORT3** in the **FRONT SIDE middle texture** text box. Click OK.
- 9 Right-click on LineDef 70. Click on the **SIDedefs** tab. Note that you need not only an *upper texture* on the **FIRST SIDedef**, but also a *lower texture* on the **SECOND SIDedef**. This lower texture is the lift base that will move up and down. Enter **SUPPORT3** in both the **FRONT SIDE upper texture** and **BACK SIDE lower texture** text boxes. (Many people use the **PLAT1** texture for this, but my theme calls for **SUPPORT3**.)

That takes care of the textures.

Now: The lift is in the *up* position and will lower when the *walkover* trigger is crossed. But you need a way for the player to get back up after he leaves the lift and it rises to its original position. Therefore, the opposite side of the lift will need a *trigger* or *activator* as well. Instead of a switch, you could make the lift fully automatic by adding another *walkover* trigger in front of it in the lower room. Or you could make both. The problem with *walkover* triggers of this sort alone is that it's not immediately visible to the player. If he gets off a lift and sees something he doesn't like, he may want to beat a hasty retreat without having advanced far enough to find the transparent trigger. He'll just turn around and try to activate the lift with the *use* key. So there's no harm in doing both. Take note, however, that *the walkover trigger must be more than 16 pixels away from the LineDef switch on the lift or else neither of them will work*. This is one of DOOM's limitations (or features).

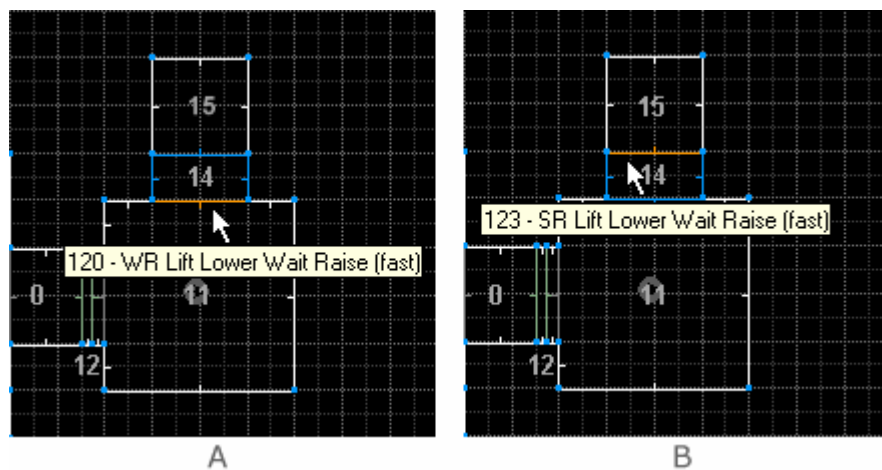
But let's keep things simple and use the opposite LineDef as a switch that the player will need to activate with the *use* key. Since switches are considered LOCAL ACTIONS, you'll recall that they must be activated from the **RIGHT SIDE** (FIRST SIDEDEF). We'll have to flip LineDef 70 so that the FIRST SIDEDEF is facing away from the lift and into Sector 15. Walkover triggers such as LineDef 68 are REMOTE ACTIONS and can be crossed from either side; therefore, we don't have to worry about its direction (except for teleporters, of course).

Let's finish assigning a LINEDEF ACTION TYPE and SECTOR TAG for LineDef 70.

**10** Click on the PROPERTIES tab. Click SELECT ACTION, expand the LIFT category, and select **SR LIFT LOWER WAIT RAISE (FAST) (123)**. Click OK. Enter **5** in the SECTOR TAG text box. Click OK.

**11** Highlight LineDef 70 and press **F** (or click the FLIP LINEDEF button) to flip the LineDef. We're done. You've created a lift!

**Figure 2.79.**  
Select the switch repeatable (SR) lift action.



This chapter turned out to run longer than I expected, but I wanted to explain every aspect of lifts I could think of and bring in all the related material discussed previously. A lift shows off all aspects of REMOTE and LOCAL ACTIONS for LINEDEF ACTION TYPES and use of SECTOR TAGS. It reveals unexpected occurrences of *upper* and *lower textures*. The 64-pixel grid scheme comes into play for choosing your flats; and theme begins to make an appearance in your choice of textures. Lifts are also one of the main features of DOOM that give it the illusion of 3D.

**NOTE:** Choose your textures for the lower sides of the lift with care. The SUPPORT textures work best; but SHAWN and some of the other silver or metallic textures work well, too. Keep in mind your color scheme for the level and use textures that match (see Figure 2.80).

**Figure 2.80.**  
Popular textures for use on lift bases and the surrounding shaft.



## 2.8 Making Rising Stairs

Rising stairs are one of the most difficult of structures in DOOM, even for folks who've been doing this a while. But they're very cool to have, if you can pull it off. The most popular approach is to have a raised area out of reach of the player that can only be accessed by the rising stairs. Whether to make the stairs obvious or not is a matter of taste and strategy.

### 2.8.1 The Mechanics of Rising Stairs

Like anything else that's cool in DOOM, rising stairs are very tricky to make. In fact, except for the Donut Function (that's where a floor with toxic waste rises to become a regular floor while a pillar in the middle of it lowers), rising stairs are the most difficult moving structures in DOOM to construct. And yet (again, like everything else), there's a simple trick to it that I'll explain with a few diagrams and a few facts.

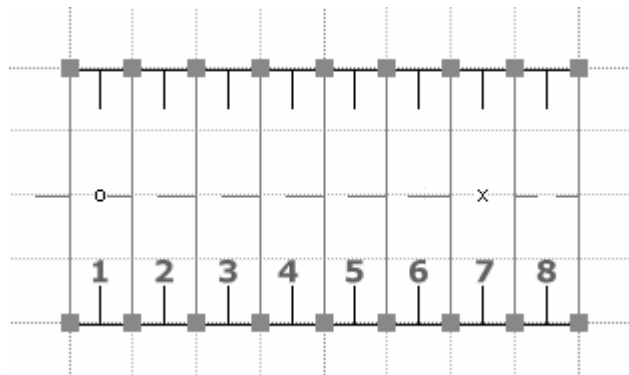
Rising stairs and the Donut Function are called *entrainers*. Rising stairs are composed of two or more adjacent Sectors. The first Sector in the series is assigned a TAG number associated with a LINEDEF ACTION TYPE assigned to either a *switch* or a *trigger* LineDef. When activated, the first Sector rises and *entrains* its action to the next adjacent Sector whose shared LineDef has its RIGHT SIDE facing *into* the adjacent Sector. This continues until the last Sector in the train has no RIGHT SIDE facing into the previous adjacent Sector.

Sound confusing? It's actually much easier than it sounds if you examine it visually.

Figure 2.81 shows a series of adjacent Sectors, some of which are part of the train of rising stairs. Sector 1 with the zero is the start Sector. As long as the RIGHT SIDE of the next Sector's shared LineDef is facing into the previous Sector, the Sectors will continue to rise. Sector 7 will be the last Sector to rise, as Sector 8's shared RIGHT SIDE is *not* facing into Sector 7.

**Figure 2.81.**

A LineDef's RIGHT SIDE displays a *Vector* (the little stick that juts out perpendicular to it). The stairs will stop rising after Sector 7.

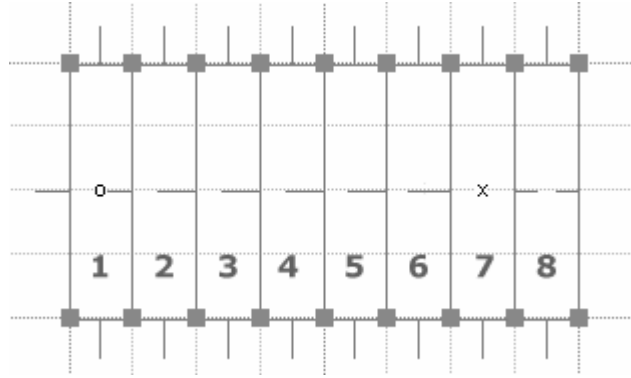


The Sectors in this set of stairs have outer LineDefs that are one-sided. Therefore, their First SideDefs – or RIGHT SIDES – face into the Sector and do not have an adjacent Sector. If this set of stairs existed in the middle of a larger Sector – that is, if they were CHILD SECTORS of a large PARENT SECTOR – the side LineDefs would be two-sided. And if their First SideDefs faced into the stair Sectors, they would *entrain* the outside Sector in which they sit. So, the First SideDefs must be flipped, as in Figure 2.82.



**Figure 2.82.**

Rising stairs within a larger Sector have LineDefs that share the PARENT SECTOR. The FIRST SIDEDEFS of those LineDefs must be flipped so they do not point into the stair Sectors.



We're going to be building a set of rising stairs that share both of these properties so that you can get an idea of how to construct rising stairs both ways.

Stairs may rise either 8 or 16 pixels. They may be fast (turbo) or slow. And they may be activated by either a *switch* (S1) or a *walkover* trigger (W1). They are not repeatable, of course. Therefore, there are four LINEDEF ACTION TYPES for rising stairs:

1. S1 Stairs Raise by 16 (fast) (127)
2. S1 Stairs Raise by 8 (7)
3. W1 Stairs Raise by 16 (fast) (100)
4. W1 Stairs Raise by 8 (8)

Doom Builder is missing number 3 from its list of ACTION TYPES, but we can plug this in ourselves (which will be another useful lesson) by simply entering the ACTION TYPE number 100 in the LINEDEF ACTION text box when we come to that step.

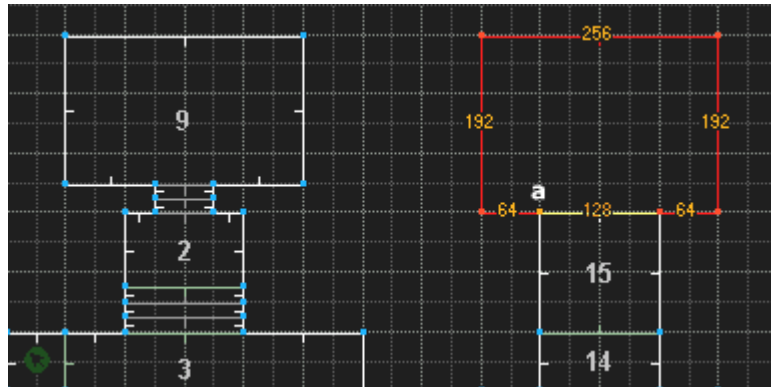
Besides having the same floor height, rising stairs must also have the same floor texture.

## 2.8.2 Creating the Space

To start, we'll need to add a couple more Sectors to the map. We'll add these right off the left area to make a room to the north, and then a hallway that runs to the previously inaccessible outer courtyard. We'll also need a small sunken CHILD SECTOR in the courtyard that is connected to the hallway (see Figures 2.83 to 2.85). Our diagrams will get a little bigger here, since we're dealing with some large structures. You may want to Zoom Out to 40% or 50% scale.

- 1 Enter SECTORS MODE. Right-click on position **a** and draw a Sector 192 high by 256 wide as shown in Figure 2.83.

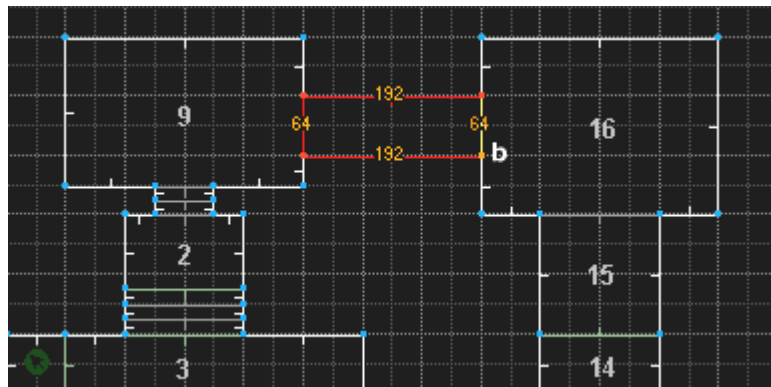
**Figure 2.83.**  
Draw a Sector 192x256. Don't forget to end drawing on the *start* Vertex!



The ceiling and floor heights for this Sector will be inherited from Sector 15: floor height **-128**, ceiling **0**.

- 2 Right-click on position **b** (Figure 2.84) and draw a 64x192 Sector between Sectors 9 and 16.

**Figure 2.84.**  
Connect Sectors 9 and 16 with a hallway, which will contain the stairs.



The ceiling and floor heights for this Sector will *not* be inherited. Therefore, we need to change them. But let's do one more thing first:

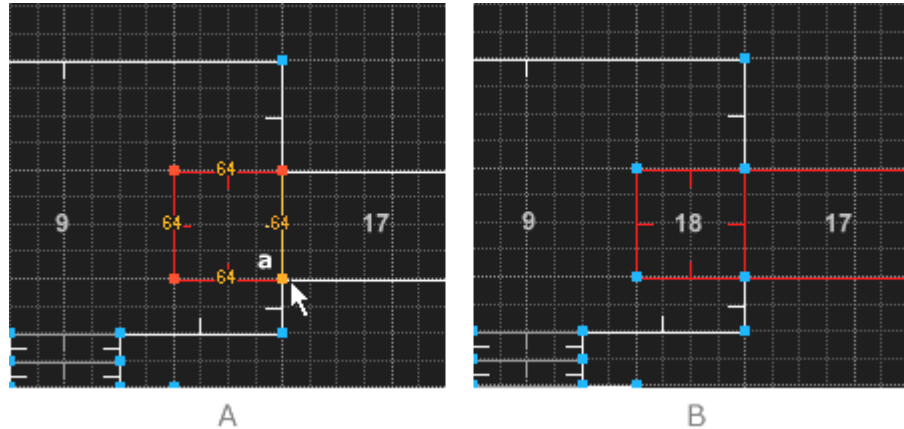
We need to add one more Sector that will be sunken into the courtyard floor of Sector 9. It looks better for the stairs to rise into a section of the courtyard, rather than rising to the edge of it. We're also going to adjust the *floor* and *ceiling heights* of these new Sectors without

entering the EDIT SECTORS SELECTION dialog box. There are some **shortcut keys** that will do this in 2D EDIT MODE that we haven't taken advantage of yet: **PgDn**, **PgUp**, **HOME**, and **END**.

- 1 Stay in SECTORS MODE. Right-click on position **a** (as shown in Figure 2.85a) and draw a 64x64 Sector.
- 2 Select this new Sector 18 and the previous Sector 17 (Figure 2.85b).

We want to lower the floor height from 0 to **-128**. Keep your eye on the *floor* and *ceiling height* numbers in the DETAILS BAR.

**Figure 2.85.**  
Add a CHILD SECTOR to Sector 9. Use the PgDn, PgUp, HOME, and END keys to raise or lower Sector heights.



- 3 Press the **END** key. You'll see that the floor has been lowered by **8** pixels. Continue pressing this key in rapid succession until you reach a height of **-128**.
- 4 Press **C** to clear the selections. Now select Sector 17 only.
- 5 Press the **PgDn** key. The ceiling has been lowered by **8** pixels. Continue until you reach a height of **0**.

Isn't that much quicker than entering the dialog box? Make sure you print out the shortcut key list and study it. It will come in very handy.

The floor height of Sector 9 is **0**, so it will be **128** units above the new Sector 18. Our stairs will have to rise this distance. You've also created some instances of exposed upper and lower textures that will need to be taken care of later.

Let's construct a series of stairs in Sectors 17 and 18. They'll raise 16 pixels and be 16 pixels deep, so we'll only need half as many to reach the floor of Sector 9. We'll also make a *walkover* trigger to activate the stairs using LINEDEF ACTION TYPE 100.

**NOTE:** Use the shortcut keys to lower and raise ceiling heights in 2D EDIT MODE instead of entering the EDIT SECTOR SELECTION dialog box.

- **PgUp** raises the ceiling by 8 pixels.
- **PgDn** lowers the ceiling by 8 pixels.
- **HOME** raises the floor by 8 pixels.
- **END** lowers the floor by 8 pixels.

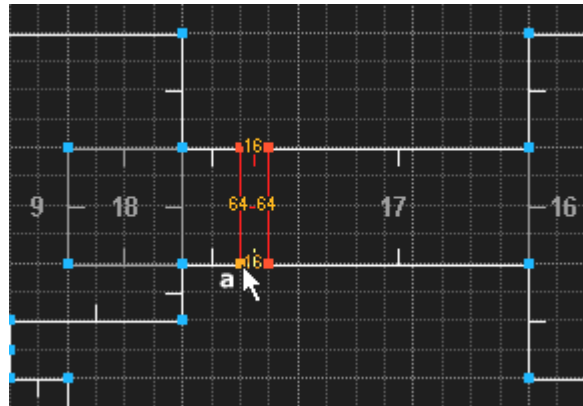
### 2.8.3 Creating the Stairs

It's important when drawing the stair Sectors that you **draw clockwise!** You want the RIGHT SIDES (with the Vector) to face *into* the Sector you're drawing. You'll still need to flip some LineDefs, but this will get most of them right.

- 1 Enter SECTORS MODE. At position **a** (as shown in Figure 2.86) right-click and start LINE-DRAW MODE. Draw up 64, 16 to the east, 64 down, and 16 west. Left-click on the *start* Vertex **a** to close the Sector.

**Figure 2.86.**

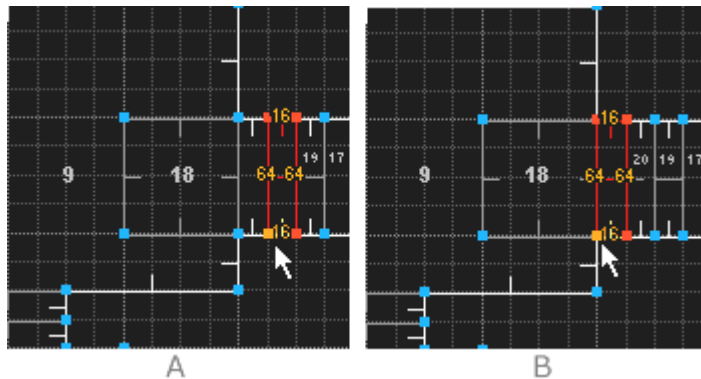
Draw your first stair Sector, being sure to overlap LineDefs and close the Sector.



- 2 Draw two more identical Sectors, overlapping the LineDefs and closing the Sectors as you go (Figure 2.87). Doom Builder will merge all superfluous Vertices and LineDefs.

**Figure 2.87.**

Draw two more identical Sectors in Sector 17.

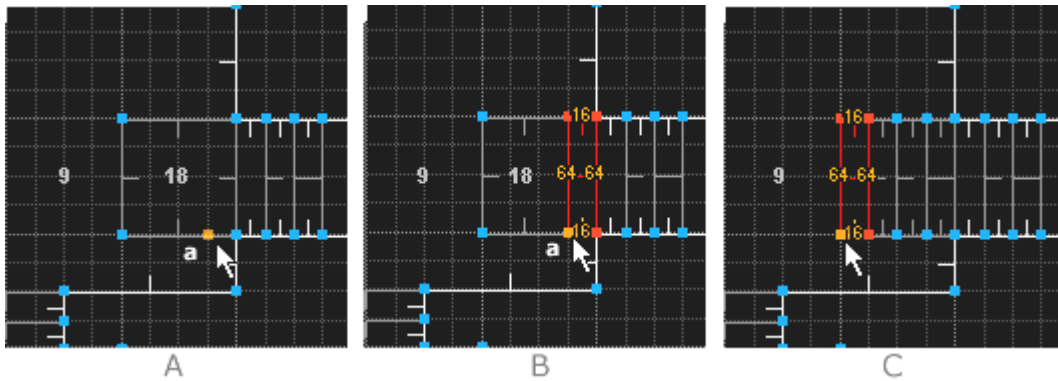


Even though it looks like you already have a 16x64 Sector between Sectors 18 and 20 (see Figure 2.87b), that space is actually still Sector 17. If you highlight it before drawing the third stair Sector, you'll see that it highlights all of Sector 17. So be sure to draw that third Sector (which will become Sector 21).

When you get to Sector 18, you'll have to switch to LINEDEF MODE. In SECTORS MODE, you'll be inside Sector 9 (since Sector 18 is a CHILD SECTOR of Sector 9) and won't be able to right-click without the Sector being highlighted and the EDIT SECTOR SELECTION dialog box being invoked.

In LINEDEF MODE, you also can't right-click to start LINE-DRAW MODE. But you *can* use the **INS** key to add a Vertex, which will dangle at the end of your cursor. Position the Vertex at the point you want, then **left-click** to insert the Vertex, split the LineDef, and start LINE-DRAW MODE and continue drawing the Sector as usual.

- 3 Enter LINEDEF MODE. Place your cursor over position **a** (as shown in Figure 2.88) and press **INS**. **Left-click** to insert the Vertex and draw the Sector *clockwise* as shown.
- 4 Use the method in Step 3 to draw three more identical Sectors (Figure 2.88).



**Figure 2.88.**

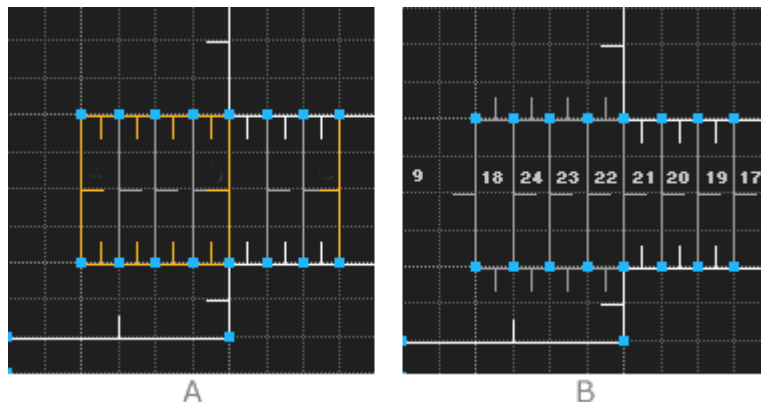
Enter LINEDEF MODE and use the INS key to add a Vertex, then left-click to start LINE-DRAW MODE.

As with the previous set of Sectors in Sector 17, the last Sector in Sector 18 looks like it already exists. But all of the Sectors you just drew are CHILD SECTORS of Sector 18. You have to overdraw that last Sector (Figure 2.88c) to get rid of the PARENT SECTOR attribute and make it a proper, separate Sector. It will still be Sector 18.

Now we'll need to flip some LineDefs so that the RIGHT SIDES are facing into the rising Sectors as needed. Figure 2.89a shows our map with those LineDefs highlighted that need flipped. Remember also that Sector 18 was a CHILD SECTOR of Sector 9. The Sectors you constructed in Sector 18 must have their outer FIRST SIDEDEFS flipped. If you don't, they will *entrain* Sector 9 and cause it to rise. Rising stairs follow according to the next First SideDef (or RIGHT SIDE) of the next adjacent Sector. Sector 9 is adjacent to all four of them, so we must flip those shared LineDefs, as shown in Figure 2.89b.

**Figure 2.89.**

Flip the LineDefs highlighted in orange.



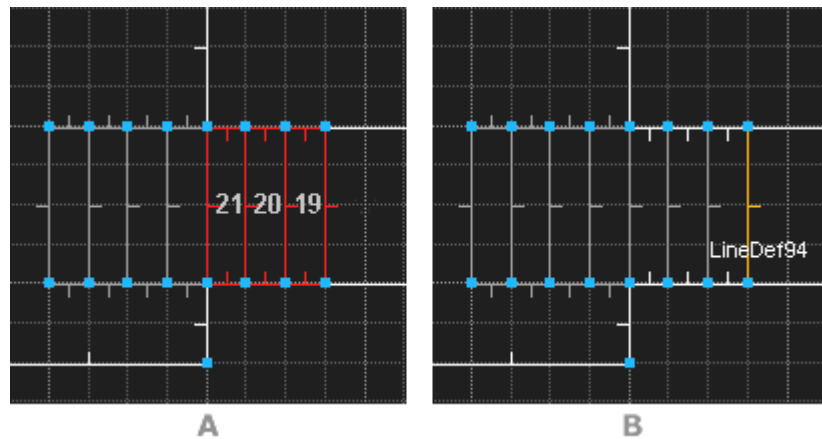
- 1 Enter LINEDEF MODE. Select the LineDefs highlighted in orange (Figure 2.89a) and press **F** to flip their RIGHT SIDES.

Figure 2.89b shows the map with the Sectors as they should look. We have one more task to perform with some of the Sectors. As the stairs rise, they will become too high for the player to get into the courtyard. The Sector ceiling height for 19, 20, and 21 needs to be raised 32 pixels to give the proper clearance and a more appealing aesthetic look rather than just raising the height of Sector 21 (which is the problem Sector). We'll also have an *upper texture* to fix.

- 1 Enter SECTORS MODE. Select Sectors 19, 20, and 21. Press **PgUp** several times to raise the CEILING HEIGHT from 0 to **32**.
- 2 Enter LINEDEF MODE. Select LineDef 94. Right-click to bring up the EDIT LINEDEF SELECTION dialog box. In the PROPERTIES tab, select the UPPER UNPEGGED check box in the FLAGS area.
- 3 Click the SIDEDEFS tab and enter **BRICK6** in the BACK SIDE *upper texture* text box. Click OK.

**Figure 2.90.**

Change the Sector heights so the player has clearance into the courtyard. LineDef 94 will have *upper textures* that need fixed.

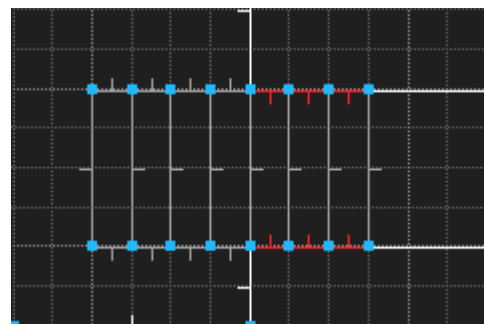


Raising the ceiling heights of Sectors 19, 20, and 21 has caused the textures to shift up 32 pixels as well. We can fix them by setting the UNPEGGED attribute for *lower textures*.

- 1 Enter LINEDEF MODE. Select LineDefs 85, 97, 91 and 95, 90, and 96 as shown in Figure 2.91. Right-click to bring up the EDIT LINEDEF SELECTION dialog box. Select the check box next to LOWER UNPEGGED in the FLAGS area of the PROPERTIES tab. Click OK.

**Figure 2.91.**

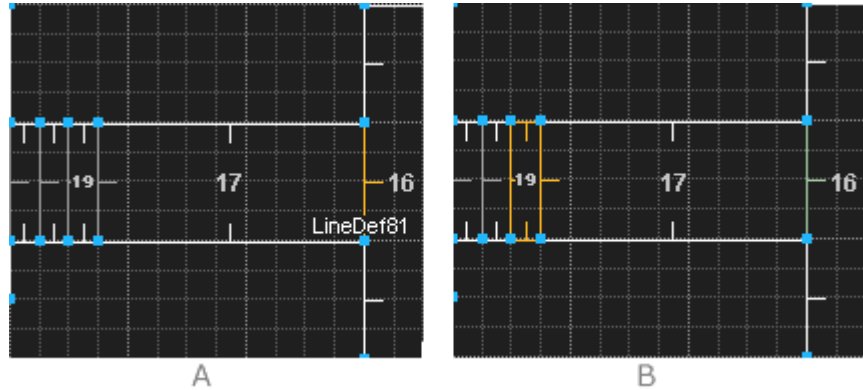
Set the LOWER UNPEGGED attribute.



We're now ready to assign an ACTION TYPE and create a *trigger* to activate the stairs. We'll also need to assign TAG numbers to the LineDef and first Sector.

- 2 Right-click on LineDef 81 (see Figure 2.92a) to invoke the EDIT LINEDEF SELECTION dialog box. Click SELECTION ACTION to bring up the STANDARD LINEDEFS tab of the dialog box. Expand the STAIRS category. Note that **w1 STAIRS RAISE 16 (FAST) (100)** is missing. Click OK.
- 3 Enter **100** in the LINEDEF ACTION text box. Click NEXT UNUSED to find a TAG number (or enter **6** in the text box). Click OK.

**Figure 2.92.**  
Assign a LINEDEF ACTION TYPE and TAG numbers to the highlighted LineDef and Sector.



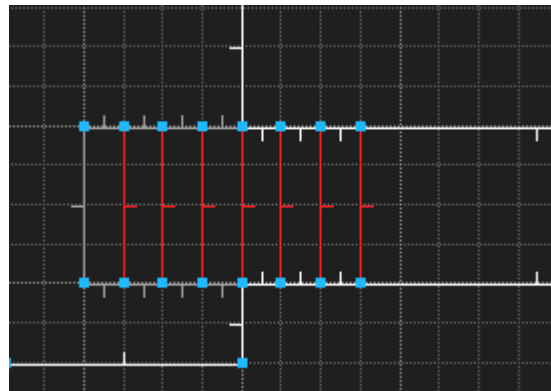
- 4 Enter SECTORS MODE. Right-click on Sector 19 to bring up the EDIT SECTORS SELECTION dialog box. Enter a TAG number of **6** in the SECTOR TAG text box. Click OK.

You've created a set of rising stairs!

But there is one more important step that is easy to forget: when the stairs rise, they will expose *lower textures* on their FIRST SIDEDEF (the RIGHT SIDE) risers. DB – nor any other editor – will not warn you that you have missing *lower textures*, because – as with lifts – that's not something that can be predicted. Let's add a STEP texture to the FIRST SIDEDEFS that need them.

- 5 Enter LINEDEF MODE. Select LineDefs 94, 92, 98, 84, 101, 104, and 107 (see Figure 2.93). Right-click on them and click the SIDEDEFS tab. Click on the FRONT SIDE *lower texture* image to bring up the SELECT TEXTURE dialog box. Find **STEP6**. Select it, then click OK to exit.

**Figure 2.93.**  
When the stairs rise, their *lower textures* will be exposed on the FIRST SIDEDEFS.

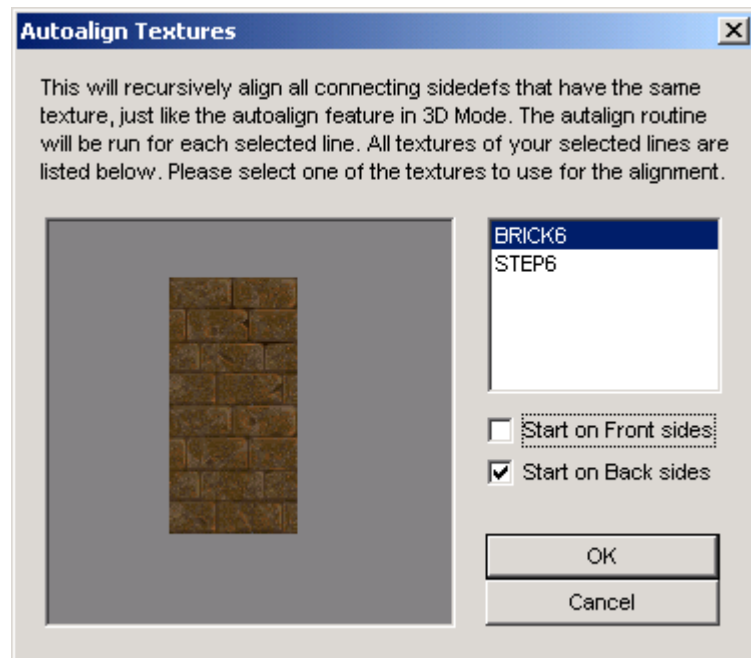


There are a still just a few cosmetic changes to make. The Sectors created in what was Sector 18 have inherited the ceiling and floor heights of the previous Sectors. We need to change the ceiling height of the last four Sectors to 128 to match the courtyard ceiling height. This will expose *upper textures* to fix, too. We'll also adjust the brightness of these Sectors and the courtyard, as they're outside and should have a higher light level. We can do this without entering the EDIT SECTORS SELECTION dialog box by using our shortcut keys.

- 1 Select Sectors 22, 23, 24, and 18. Use the **PgUp** key to raise the CEILING LEVELS to **128**. Press **CTRL+PgUp** to adjust the BRIGHTNESS to **208**. Press **C** to clear all selections.
- 2 Select Sector 9 and adjust the brightness to **208** with the **CTRL+PgUp** keys.
- 3 Enter LINEDEF MODE. Right-click on LineDef 84. Select the UPPER UNPEGGED check box in the FLAGS area in the PROPERTIES tab. Click the SIDEDEFS tab. Enter **BRICK6** in the BACK SIDE *upper texture* text box. Click OK.
- 4 Highlight LineDef 84 and press the **A** key. The AUTOALIGN TEXTURES dialog box comes up. Unselect the check box next to START ON FRONT SIDES. Click OK.

**Figure 2.94.**

Align textures starting on the BACK SIDES.



This last step will AUTOALIGN the textures not only around the courtyard, but along the stair LineDefs as well. If you like, you can enter 3D EDIT MODE to have a look at how everything looks.

And that concludes the lesson on building rising stairs. We've constructed a fairly difficult set, as rising stairs go. For further information on rising stairs, I recommend **Tricks of the DOOM Gurus** (the second edition is called **3D Game Alchemy**) by SAMS Publishing, and Hank Leukart's **The DOOM Hacker's Guide**. Practical information is around for free in Raphaël Quinet's *DEU5: 5.21 DOOM Editor Utilities*, available as DEU.TXT on most FTP game sites (/Docs directories). It also comes with most versions of the DEU editor, DEU52x.ZIP. (Check the [Editors](#) page at [Dr Sleep's DOOM Apothecary](#).)



## 2.9 Managing Your Map



**Save your map often.** In fact, you should get in the habit of clicking the SAVE MAP button in the toolbar every few minutes – or at least after every major structural change in your map. Doom Builder has never crashed on me, but there *are* such things as brown-outs and electrical storms, and cats and other critters that crawl behind your computer and step on the reset button of your surge protector (yes, I've actually had that happen).

DB has a variety of save and [merge](#) options, a feature for [changing the map/level number](#) (or *lumpname*) of your PWAD, and various means for [testing](#) your level straight from the editor. These are all explained in detail in [Section 1.0](#) (click the hyperlinks to go to the appropriate chapter) and are accessible from the FILE and EDIT MENUS.

### 2.9.1 Export Node Build

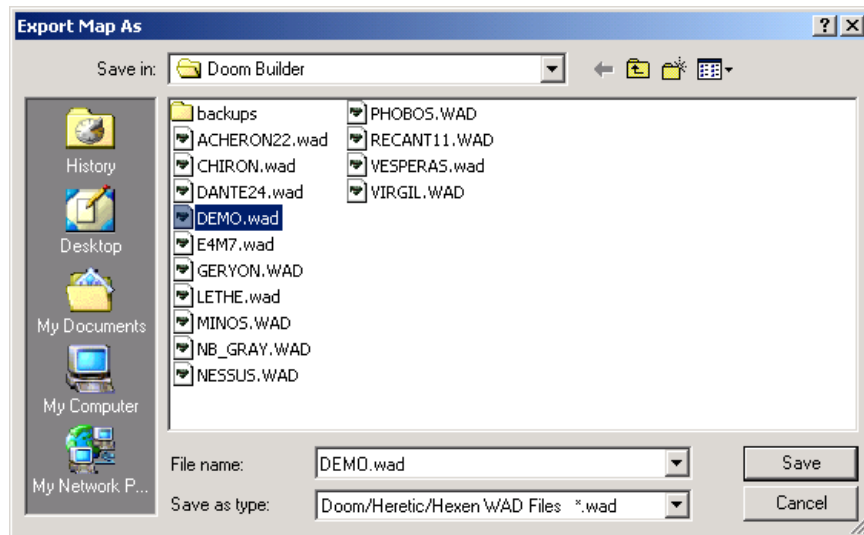
After you've aligned the last texture, tweaked the lighting in half a dozen Sectors, and double-checked the ratio of monsters to health and ammo, you're finally ready to release your latest masterpiece to the general public. But you're not quite finished. You'll do one last node build, one more optimization of the REJECT<sup>†</sup>, and one last save. DB provides a means for doing all of these in one go with the EXPORT MAP function.

**EXPORT MAP** is intended to be used when your map is complete and you want to do a final save and node build on it.

- 1 Click on the FILE MENU and select EXPORT MAP. You'll be prompted to enter a filename in the EXPORT MAP AS dialog box. DB will then use the export node builder and parameters that you have defined in the [NODEBUILDER](#) tab of the CONFIGURATION settings.

**Figure 2.95.**

You can enter the same or a different filename for your map. DB will then do a final build and save.

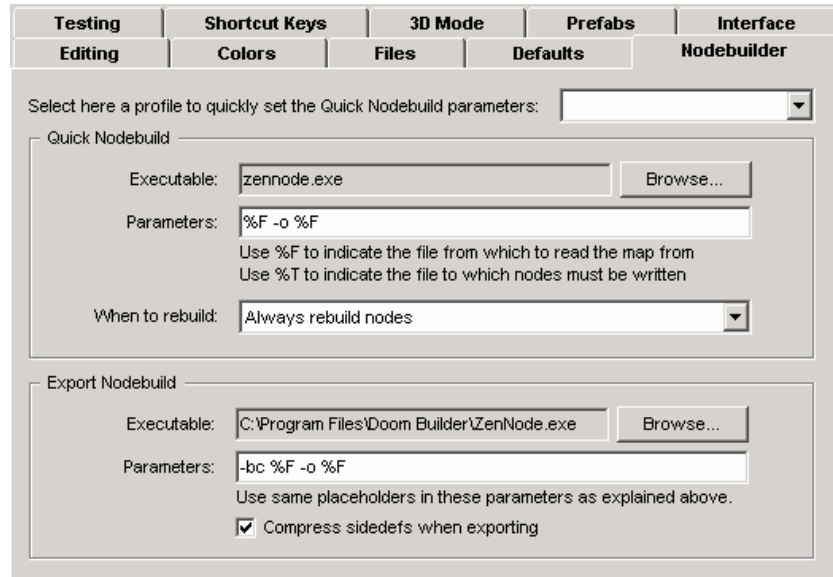


<sup>†</sup> The DOOM engine uses the REJECT resource to make line-of-sight (LOS) calculations during game play to determine which Sectors can be seen by other Sectors. This in turn determines whether or when monsters can see the player. An optimized REJECT table speeds a level up by 80%. Some people will argue that the REJECT resource no longer needs to be optimized because of today's fast machines. That may be true, but an optimized REJECT table nevertheless makes a PWAD more efficient, and it only takes a few seconds to do it with ZenNode. The REJECT resource has other uses, which can make for some very interesting special effects such as blind monsters and Sectors that are safe zones. See my article [Optimizing the REJECT Resource](#).

2 Press **F5** to bring up the CONFIGURATION dialog box. Click on the NODEBUILDER tab.

You'll see that there is a separate field for EXPORT NODEBUILDER, which is by default set to ZDBSP. This is where you choose the node builder you want to use for your final build. The QUICK NODEBUILD field is for choosing a node builder that will do a fairly quick node build for general editing and testing purposes. With a final node build, you may want to choose a different node builder, or assign different parameters for the node builder to perform (such as a REJECT build and compression of the BLOCKMAP). There is also the option to COMPRESS SIDEDEFS WHEN EXPORTING. This will significantly reduce the size of your PWAD, and I recommend that you use it.

**Figure 2.96.**  
Choose a node builder for doing a final build of your map.

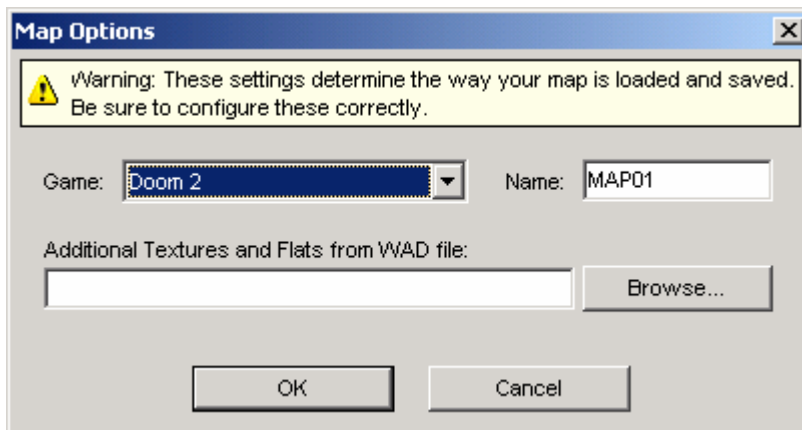


For more information on how to change the node builder options, see [Chapter 1.1.4: Nodebuilders Tab](#). If you want to use a different node builder than one of the three included with Doom Builder, see [Chapter 3.2.1: Node Builder Profiles](#) for instructions on how to create custom profiles for different node builders. Also, see [Appendix B: Node Builder Parameters](#) for all of the options available with BSP, ZDBSP, and ZenNode.

## 2.9.2 Changing a Map's Level Number (Lumpname)

Should you ever want to make your map something other than MAP01, Doom Builder makes it easy to change the level number, or *lumpname*, as it's more properly called. As I've said a number of times, the *lumpname* is the directory entry in a PWAD that tells DOOM which episode and level number (**EnMn** - DOOM1) or map number (**MAPnn** - DOOM2) the map replaces. This will determine which music soundtrack is played during game play, and also which sky is used. There are other actions that are level-specific, such as secret levels and certain Boss-death functions that end the level or lower a floor – but many designers like a certain music soundtrack and pick their level numbers accordingly. It's always easier to just choose MAP01 (or E1M1), but it's also a little boring.

- 1 To change the lumpname for your level, press **F2** (or select MAP OPTIONS from the FILE MENU). This brings up the MAP OPTIONS dialog box.
- 2 In the NAME area, enter the level number you want to change your map to. For DOOM2 it's **MAPnn** and for DOOM1 it's **EnMn**. Click OK.



**Figure 2.97.**

Change your PWAD's lumpname (or level number) in the MAP OPTIONS dialog box.

It's that simple.

Be careful if you're editing a map in a multilevel PWAD. DB only saves the map currently loaded in the editor; if you change the lumpname to a level that already exists in the PWAD directory, it will be overwritten.

That concludes [Section 2.0](#). You can learn about Doom Builder's more advanced functions in [Section 3.0](#).

# SECTION 3

## Advanced Functions in Doom Builder

- *Making and Using Prefabs*
- *Creating Custom Profiles*
- *Using Alternate Texture PWADs*

## 3.0 Advanced Functions in Doom Builder

### 3.1 Making and Using Prefabs

Doom Builder has a very handy set of features for making and loading **PREFABS**. Prefabs are pre-designed or "prefabricated" objects that one may use often in a map, but find inconvenient to rebuild over and over again. Objects like chairs, tables, couches, computers, toilets, sinks – just about anything you can think of – are good candidates for prefabs. And a prefab doesn't have to be a small object like those listed. It can be an entire area of a map – say, a specially designed set of stairs that one uses often; or a special teleporter pad that you like to use throughout your maps; or a certain design motif that you want to have appear in all of your levels. Just about any structure in a DOOM map you can imagine can be a prefab.

Doom Builder does not save any special Sector or LINEDEF ACTION TYPE information with the prefab. So saving a teleporter pad as a prefab would not preserve the tag number or teleport ACTION TYPE. (This tag information would not be relevant in a new level anyway.) But all else – texture, flat, height, and lighting – is preserved. For complex objects, the prefab feature is still a godsend.

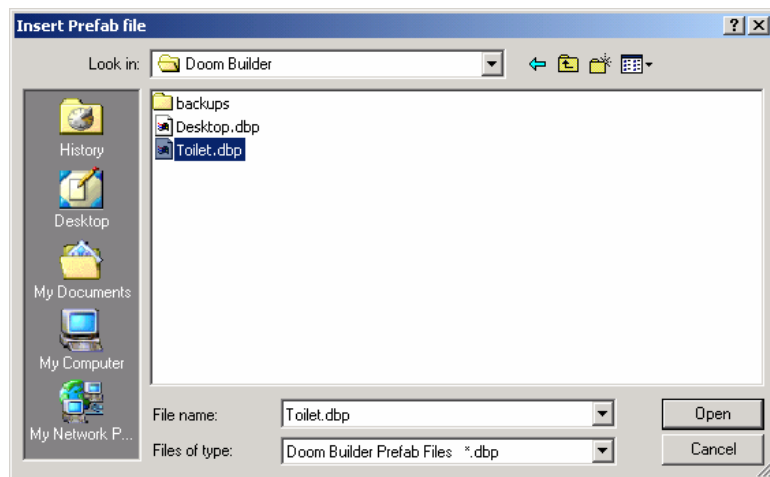
Prefabs are designed by you in Doom Builder and then saved out as a special prefab file. Prefabs must have the extension **.DBP** (Doom Builder Prefab) and the structure should consist of closed Sectors. Prefabs are wee maps, in other words. We'll go through the steps of defining, saving, and loading prefabs.

Doom Builder comes with two prefabs. To get a better picture of what we're dealing with, let's load and insert a prefab structure first. There are two prefab buttons on the toolbar:



**INSERT PREFAB FROM FILE** summons the INSERT PREFAB FILE dialog box. LOOK IN the Doom Builder directory and then select **TOILET.DBP**. Click OPEN.

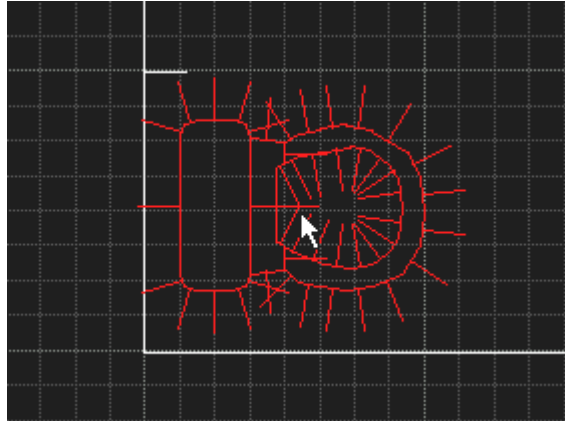
**Figure 3.1.**  
Prefabs must have an extension of .DBP.



DB loads the prefab and leaves it dangling at the end of your mouse cursor. Just move the cursor to wherever you want to insert the prefab and left-click to drop it. You can then select the object and rotate it or do whatever you need to do to get it in the position you want. Position the toilet prefab in the lower south-west corner of the map (as shown in Figure 3.2).

### Figure 3.2.

The prefab stays at the end of the mouse cursor. Left-click to "drop" it.



**INSERT PREVIOUS PREFAB** does just that: it inserts the last prefab you loaded. If you click this button in the toolbar, the TOILET.DBP will be loaded again.

Commands for saving out prefabs are in the PREFAB MENU.

**SAVE SELECTION AS PREFAB** saves all selected Sectors and LineDefs as a separate prefab file. Let's save a portion of TEST.WAD for demonstration purposes.

- 1 Select Sector 2 and Sector 13.
- 2 Click the PREFAB MENU and then select SAVE SELECTION AS PREFAB. In the SAVE PREFAB FILE dialog box, type **TEST** in the FILE NAME text box.

DB will assign the .DBP extension when it saves the file. This item is now available for recall as TEST.DBP.

If your selection contains a LineDef that is shared with another Sector that wasn't selected, Doom Builder will make the LineDef one-sided and impassable and close the Sector properly for you.

**NOTE:** Remember that you can select large sections of a map by holding down the left mouse button and drawing a box around the objects.

**SAVE ENTIRE MAP AS PREFAB** requires no selecting. The entire contents of the currently opened map grid are saved out as a prefab. This option is handy for creating prefabs from scratch in DB. With no other level loaded, you can just start building your prefab and save the entire contents of your grid as a prefab.

**NOTE:** DB does not save any TAG or ACTION TYPE references in the selection. So all doors and lifts will no longer work if you're saving such portions of a map. It *will* save all other properties of the structures, such as textures and flats, and light levels.

## 3.2 Creating Custom Profiles

You can create your own custom profiles for node building and map testing by editing the file **parameters.cfg**. You can use Windows *Notepad* or your favorite text editor. Make sure you read this section carefully before fooling around with this file, because you may render your other profiles useless if you screw something up. It's not at all difficult to edit the profiles, but even something as seemingly insignificant as a misplaced space, backslash (\), or bracket ( { } ) can have a deleterious effect. **Be sure to make a backup copy of parameters.cfg** (rename it *parameters.cfg.org* or just copy it to another directory).

DB's configuration files follow a C standard structure. Let's look at the structure of the configuration file for the various profiles that are separated into **NODEBUILDERS** and **ENGINES**.

### 3.2.1 Node Builder Profiles

The parameters for the NODE BUILDERS section of the file are expressed:

```
nodebuilders
{
    zdbbsp1
    {
        title = "ZDBSP Fast build (no reject)";
        executable = "zdbbsp.exe";
        parameters = "-R \"%F\" -o \"%T\"";
    }
}
```

**Nodebuilders** is the parameter type, and **zdbbsp1** is the profile name. The profile must be within brackets (**{}**). All definitions must be within sub-brackets (**{ }**) of the profile, and the string values must be within quotation marks ("" ). A semi-colon (;) indicates the end of the definition.

- **title** is the profile title that will show up in the drop-down box on the NODEBUILDER tab of the CONFIGURATION dialog box.
- **executable** is the node builder executable, which should be in the C:\Program Files\Doom Builder directory.
- **parameters** is the string of options or parameters needed to complete a specific build type by the node builder combined with Doom Builder's placeholders **%F** and **%T**.

The placeholder **%F** stands for the filename of your PWAD. The placeholder **%T** stands for an alternate filename that the node builder will write the results (or *output*) to instead of overwriting your current PWAD. (Note that not all node builders support the **%T** option.)

In English, this means you should just copy and paste an existing profile and substitute the values to suit your needs.

**NOTE:** Only copy (or create) the information between the first bracket following **nodebuilders**, and the last bracket in the same column before **engines**. This information is **bold-faced** in the first example above.

For instance, here are a few profiles for GLBSP:

```
glbsp1
{
    title = "glBSP OpenGL Build";
    executable = "glbsp.exe";
    parameters = "%F -o %F";
}
```

The parameters above set GLBSP to build the GL and normal nodes and store them within your PWAD. If you are making a level intended only for an OpenGL DOOM port such as `JDoom`, `EDGE`, or `VAVoom`, this is fine. But if you would like your level to be playable in regular DOOM or `ZDoom`, then the GL nodes build within your PWAD may cause these games to crash. You can use the **-nogl** option to build only the normal nodes and set this up as a separate profile:

```
glbsp2
{
    title = "glBSP Normal build (no OpenGL)";
    executable = "glbsp.exe";
    parameters = "-nogl %F -o %F";
}
```

Remember, though, that this option gives you no GL node build. Since OpenGL ports require a GL node build, there is another option for building the GL nodes and saving them separately as a **.GWA** file that these games will detect. Unfortunately, DB does not support running GLBSP in this mode, so you will have to run this kind of build outside of DB.

Here are some more parameters you may want to consider for GLBSP:

- **-noreject** keeps GLBSP from clobbering any REJECT build you may have.
- **-packsides** will pack the SideDefs, creating a smaller PWAD.

If you decide to use any of these, simply plug them into the parameters field – or create a new profile – allowing a space between options and making sure the entire string is within quotes and followed by a semicolon.

```
glbsp3
{
    title = "glBSP Normal build (packsides)";
    executable = "glbsp.exe";
    parameters = "-nogl -noreject -packsides %F -o %F";
}
```

While you're free to create as many profiles as you like, you should really only create new profiles for node build options that you use on a regular basis. There's no sense cluttering up your *parameters.cfg* file or creating a confusing set of drop-down choices for profiles you may use only rarely. It may be just as easy to simply press F5, go to the NODEBUILDER tab of the CONFIGURATION dialog box and type in the parameters you only plan on using for that particular editing session.



You can just as easily create additional profiles for ZenNode, BSP, or ZDBSP if you're familiar with them and have other options you'd like to use with them on a regular basis. One more example is the ZenNode profile I use exclusively:

```
zennode3
{
    title = "ZenNode Normal build: Compress BLOCKMAP";
    executable = "zennode.exe";
    parameters = "-bc %F -o %F";
}
```

This profile tells ZenNode to compress the BLOCKMAP. When used with the COMPRESS SIDEDEFS WHEN EXPORTING option, this feature will further reduce the size of your PWAD.

All node builders' options and parameters are listed in [Appendix B: Node Builder Parameters](#). Again, just be sure to backup your *parameters.cfg* file if you decide to edit or add a profile.

### 3.2.2 Map Testing Profiles

The parameters for the ENGINES (map testing) profiles are similar to those for node builders. Note that the backslash (\) is needed as an *escape character* before quotation marks ("). Since quotation marks are used to indicate the start and end of the string, quotes *within* your parameters require the escape character to denote that they do not represent the end of the string. This allows you to insert characters in the string in a way that they don't interrupt with the format. Note that the backslash (\) must precede quote marks around placeholders, since they themselves are within the value quote marks.

```
engines
{
    zdoom2
    {
        title = "ZDoom Singleplayer Skill 2";
        executable = "zdoom.exe";
        parameters = "-iwad %D -file \"%A\" \"%F\" +map %L -skill 2";
    }
}
```

Doom Builder offers three testing profiles for zDOOM and three for DOOM LEGACY. I'll show you how to create profiles for DOOM and jDOOM so that you can jump right into either one of these games to play-test your map.

Here's a profile for testing your map in **DOOM II**:

```
doomi1
{
    title = "DOOM II Singleplayer (nomonsters)";
    executable = "doom2.exe";
    parameters = "-iwad %D -file \"%F\" -warp %L -nomonsters";
}
```

You can vary the above parameters to reflect different skill levels with monsters, though most designers prefer to test their levels without having to mess about with the level denizens getting in their way. Note that if your map is MAP01, you won't need the **-warp** parameter.

For **DOOM1**, the parameter for warping to another level is a little different:

```
doom1
{
    title = "DOOM Singleplayer (nomonsters)";
    executable = "doom2.exe";
    parameters = "-iwad %D -file \"%F\" -warp %E %M -nomonsters";
}
```

More parameters for DOOM and DOOM2 can be found in *The Official DOOM and DOOM II FAQ v6.666* by Hank Leukart, which comes with DOOM2. You should find it in your DOOM2 directory as DMFAQ66B.TXT.

**JDOOM** is a little trickier. Here's what works for me:

```
jdoom1
{
    title = "jDOOM Singleplayer";
    executable = "doomsday.exe";
    parameters = "-game jdoom.dll -iwad %D -file \"%F\" -warp %L";
}
```

Some of you may need the **-gl drD3D.dll** parameter with your screen resolution **-width nnn** and **-height nnn** specified. JDOOM uses the **drOpenGL.dll** renderer by default, so if this is already set in your console, then you don't need the parameter. You can find more command line parameters in the CmdLine.TXT located in the jDoom \doc directory.

Once you've edited your *parameters.cfg* and saved it, fire up Doom Builder and press F5 to go to the CONFIGURATION dialog box. Click on the NODEBUILDERS tab and click the drop-down arrow for the PROFILES to see if your new ones show up. Select them one at a time to make sure they load the correct parameters. Then click the TESTING tab and examine the profiles there to see if they're correct. (If for some reason they're not, go back into your *parameters.cfg* file and make sure you haven't deleted a bracket, forgotten quotes or a semi-colon. Pull up your original *parameters.cfg* that you backed up and just do a line compare. If you can't find your mistake, stick the original configuration file back in the Doom Builder directory and go back to the drawing board. The most damaging mistakes are usually the most subtle.)


**NOTE:** Be sure to make a backup copy of your *parameters.cfg* file before changing it!

More than likely you'll have edited the file correctly and have a brand new set of profiles. All of the configuration files can be edited, in fact, but I certainly don't recommend it. The various Doom Builder GAME CONFIGURATION files, however, show potential for adding LINEDEF ACTION TYPES or changing their descriptions – as well as SECTOR SPECIALS, THING TYPES, etc. There's not much point to it, since DB presents them in a clear, rational manner. But fanatics who want their LINEDEF ACTION TYPES to match those exactly of, say, DEU (or perhaps to match the descriptions given in [The Unofficial DOOM Specs](#)) could knock themselves out changing them with little harm (but a lot of effort. The zDOOM configuration file has several hundred LINEDEF ACTION TYPES while DOOM2 only has 143).

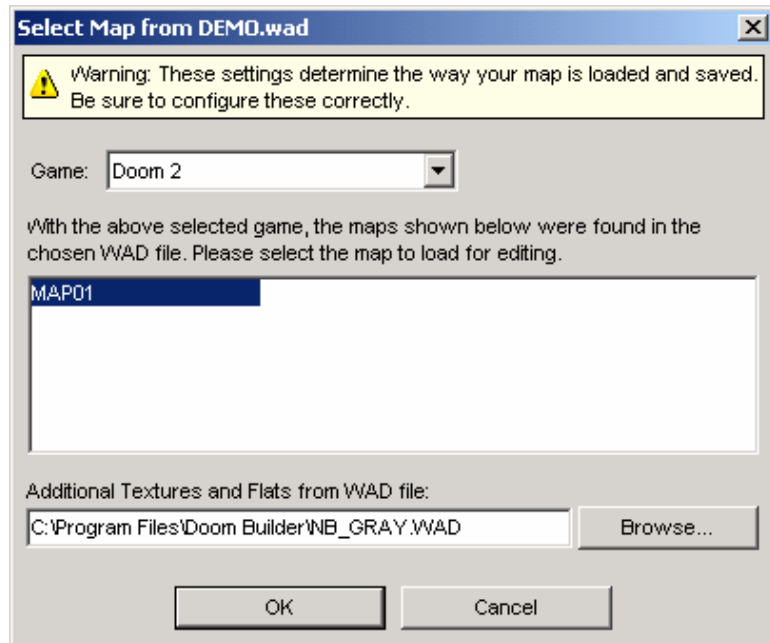
But there you are. Now that you know how to create new profiles, what are you waiting for?

### 3.3 Using Alternate Texture PWADs

Doom Builder supports the use of alternate textures and flats in your level. Alternate texture PWADs can easily be loaded along with your map. If you haven't made your own textures, there are many DOOM websites on the Internet that feature texture and flat collections in PWAD form. Many are grouped according to theme (gothic, tech, medieval, etc.) and can really spice up a level, relieving the tedium of using the same old (but great!) DOOM textures.

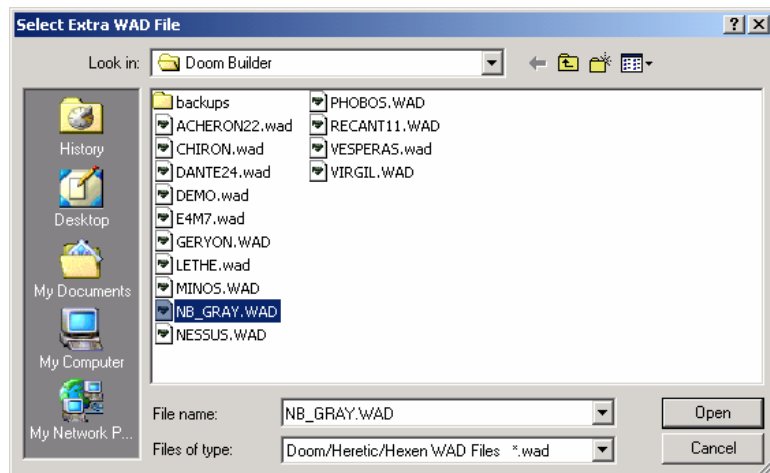
- 1 Click on the OPEN MAP button  to bring up the OPEN MAP dialog box and select your PWAD. The SELECT MAP FROM dialog box will then come up. Choose your map number, then click the BROWSE button in the ADDITIONAL TEXTURES AND FLATS FROM WAD FILE area.

**Figure 3.3.**  
Click BROWSE to select your texture PWAD.



- 2 The SELECT EXTRA WAD FILE dialog box pops up. Choose your texture PWAD and click OK.

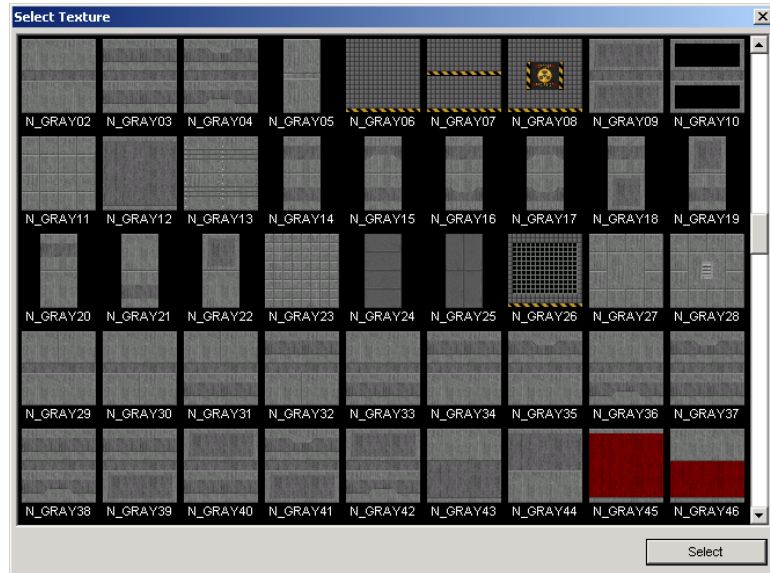
**Figure 3.4.**  
Select the texture PWAD you want to load.



- 3 Once DB loads your map and texture PWAD, simply go into LINEDEF MODE, right-click on any LineDef to bring up the SELECT TEXTURE dialog box, and your new textures should be there.

The example in Figure 3.5 shows textures from the popular **NB\_GRAY** series by Nick Baker, available at [The Afterglow](#) website.

**Figure 3.5.**  
Textures from the alternate texture PWAD NB\_GRAY.WAD.



As you can see, you select new textures just as you did old ones. And all the regular DOOM textures are still available. The NB\_GRAY series is themed to go with the GRAY and MODWALL textures from DOOM.

The ability to use alternate textures and flats makes Doom Builder the perfect editor for that MegaWAD conversion project you've been planning. If you want to construct your own texture or flat PWADs, Doom Builder supports 16-, 24-, and 32-bit high-resolution PNG graphics. These find their best use in OpenGL DOOM ports like JDOOM, which support high-resolution graphics (though JDOOM will not read these from a PWAD: you have to construct a PK3). To make your map in Doom Builder using high-resolution textures, you'll need to supply or make your own PWAD. I recommend the excellent **XWE**, *eXtensible WAD Editor* by Csabo, which is perfect for making texture and flat PWADs. (Visit the [XWE](#) website to download.)

## Acknowledgements

This guide wouldn't exist if it weren't for the excellent editor it was written for, and therefore I'm deeply indebted to its author, Pascal "CodeImp" vd Heiden, for taking the time to answer technical questions regarding Doom Builder.

David "Tolwyn" Shaw provided the most far-reaching help. His advice on how to manage the screenshots is why they look so great. (They print out even better.) He argued with me over technical terms, and reminded me of DB functions I'd either forgotten or didn't know about. He also guided me step-by-step through getting the most out of Adobe *Acrobat*, again helping to give this manual the best look without sacrificing image quality in its PDF format. I bugged Tolwyn on a near-daily basis for weeks, and this manual is the better for it (though he doesn't email me as much as he used to). Tolwyn's article [How to Use Hi-Res Textures in jDOOM](#) was a big help to me, too.

Thanks to Darren "Doom\_Dude" Finch and Owen "Sarge Baldy" Lloyd for giving the manual a once-over and pointing out errors and offering suggestions. Doom\_Dude especially deserves special thanks because he was the one who pointed me to Doom Builder in the first place. Thanks as well to Dan Stefan "Alpha" Oprean for pointing out some typos.

Last, but by no means least, thanks to Jaakko "SkyJake" Keränen for his excellent *Doomsday Engine* and jDOOM – without which I'd probably never have rediscovered the joys of level making for DOOM. Visit [Doomsday HQ](#) on the web.

Any errors in this guide should be attributed to me. If you find a technical disparity between this manual and the editor, don't write to Pascal about it. Contact me if you'd like to point anything out or offer suggestions. I'll be revising this manual on a regular basis. I would like to make mention of several resources that were a benefit to me when I found myself stuck for a definition or technical expression: Matt Fell's [The Unofficial Doom Specs v1.666](#) (rightly considered the DOOM Bible), *Tricks of the DOOM Gurus* by SAMS Publishing, and Hank Leukart's *The DOOM Hacker's Guide*. I've tried to absorb these works over the years, and give credit in this manual where it was due.

The [LineDef Action Type Reference](#) by Jim Flynn is included with this manual as [Appendix C](#), but comes from a larger work, which also cites all Sector Specials by Function and Number, as well as Things by Class and Number: [The WAD Author's DOOM Reference](#).

The latest version of Doom Builder is always available at CodeImp's *Doom Builder* home page, <http://www.doombuilder.com>, and at my web site, *Dr Sleep's DOOM Apothecary*, <http://drsleepp.newdoom.com>. Use the following links to download self-extracting ZIP archives of the manual and the tutorial DEMO.WAD:

Adobe *Acrobat* PDF: <ftp://drsleepp@server1.thefourwinds.net/data/dbguidep.exe>

Microsoft *Word*: <ftp://drsleepp@server1.thefourwinds.net/data/dbguide.exe>

This manual was written with Microsoft *Office Professional Word 2003 11.0*. Screenshots and illustrations were made with JASC Software *Paint Shop Pro 8.0*. Other image manipulation and management was accomplished with ACD System *ACDSee Powerpack 6.02*. The PDF version of this document was created with Adobe *Acrobat Professional 6.01*. The HTML version was created with Macromedia *Dreamweaver MX 2004 7.0*.

John W. Anderson

(*Dr Sleep*)

03.24.2004

[drsleepp@newdoom.com](mailto:drsleepp@newdoom.com)

## Revision History

This page documents the various major revisions to **Doom Builder: An Illustrated Guide**. Minor revisions to this manual are always going on in the background, such as corrections of typos, errors of fact, spelling, and format consistency. At least ten of these minor revisions have already been done without any announcement.

Major revisions will always accompany any new version of the Doom Builder editor. I work closely with Pascal in testing all beta versions of his editor (I also constantly provoke him to add new features), so I will always endeavor to make sure this manual accurately reflects all of the features of and any new additions to the editor.

The Doom Builder manual project was started on February 04, 2004 and finished on March 24, 2004.

**Version 1.0 published March 28, 2004.** 131 pages.

- For Doom Builder version 1.53, Build 223.

**Version 1.1 published May 07, 2004.** 144 pages.

- For Doom Builder version 1.60, Build 262.
- New images of the INTERFACE, 3D MODE, and SHORTCUT KEYS Configuration tabs.
- All new images added to reflect 256-color change to the icon toolbar.
- MOVE MODE added and documented in [Chapter 1.3.1: Doom Builder's Editing Modes](#).
- New EXPORT PICTURE feature description added to [Chapter 1.3.3.1: The File Menu](#).
- [Chapter 1.3.3.5: The Script Menu](#) added.
- Addition of X and Y OFFSETS and SECTOR REFERENCES to the LineDef DETAILS BAR described in [Chapter 1.3.4: The Details Bar](#).
- Shortcut keys SHIFT+C (Copy Offsets) and SHIFT+V (Paste Offsets) added to [Appendix A: Doom Builder Shortcut Keys](#).
- *LineDef Action Types by Number* table appended to [Appendix C: LineDef Action Type Reference](#).

**Version 1.2 published July 26, 2004.** 146 pages.

- For Doom Builder version 1.63, Build 310.
- New GRADIENT feature documented.
- New SCALING feature documented and toolbar and icon images updated.

A P P E N D I C E S

# A-C

- *Appendix A: Doom Builder Shortcut Keys*
- *Appendix B: Node Builder Parameters*
- *Appendix C: LineDef Action Type Reference*

## Appendix A: Doom Builder Shortcut Keys

<b>Edit Mode</b>	
Cancel Current Operation	<b>ESC</b>
Copy Properties	<b>CTRL+SHIFT</b>
Create Sector	<b>CTRL+INS</b>
Deselect All (Clear)	<b>C</b>
Enter 3D Mode	<b>W</b>
Draw Sector (Line/Sector Mode)	<b>INS</b>
Draw Sector	<b>CTRL+D</b>
Copy	<b>CTRL+C</b>
Paste	<b>CTRL+V</b>
Cut	<b>CTRL+X</b>
Undo	<b>CTRL+Z</b>
Redo	<b>CTRL+Y</b>
Zoom In	<b>Scroll-Up</b>
Zoom Out	<b>Scroll-Down</b>
Scroll Up	<b>Arrow Up</b>
Scroll Down	<b>Arrow Down</b>
Scroll Left	<b>Arrow Left</b>
Scroll Right	<b>Arrow Right</b>
Map Options	<b>F2</b>
Configuration	<b>F5</b>
Test Map	<b>F8</b>
Build Nodes	<b>CTRL+F8</b>
Delete	<b>DEL</b>
Vertex Mode	<b>V</b>
Lines Mode	<b>L</b>
Sector Mode	<b>S</b>
Things Mode	<b>T</b>
Switch Mode	<b>TAB</b>
Rotate	<b>R</b>
Grid Decrease	<b>[</b>
Grid Increase	<b>]</b>
Snap to Grid	<b>CTRL+ENTER</b>
Toggle Snap	<b>-</b>
Toggle Stitch	<b>-</b>
Insert Prefab from File	<b>CTRL+P</b>
Insert Previous Prefab	<b>P</b>
Insert Prefab 1	<b>CTRL+F1</b>
Insert Prefab 2	<b>CTRL+2</b>
Insert Prefab 3	<b>CTRL+F3</b>
Insert Prefab 4	<b>CTRL+F4</b>
Insert Prefab 5	<b>CTRL+F5</b>
Find	<b>-</b>
Find and Replace	<b>-</b>
Flip Horizontal	<b>-</b>
Flip Vertical	<b>-</b>
Copy Properties	<b>CTRL+SHIFT+C</b>
Paste Properties	<b>CTRL+SHIFT</b>
Remove Unused Textures	<b>-</b>
Create Sector	<b>CTRL+INS</b>



<b>Vertex Mode</b>	
Clear Unused Vertices	<b>CTRL+SHIFT+C</b>
Insert Vertex	<b>INS</b>

<b>Lines Mode</b>	
Curve Lines	<b>CTRL+C</b>
Flip LineDefs	<b>F</b>
Flip SideDefs	<b>CTRL+F</b>
Place 3D Start	<b>CTRL+W</b>
Draw Sector	<b>INS</b>
Auto-Align Textures	<b>A</b>
Select One-Sided LineDefs	<b>CTRL+1</b>
Select Two-Sided LineDefs	<b>CTRL+2</b>

<b>Sectors Mode</b>	
Draw Sector	<b>INS</b>
Increase Brightness	<b>CTRL+PgUP</b>
Decrease Brightness	<b>CTRL+PgDN</b>
Join Sectors	<b>J</b>
Merge Sectors	<b>CTRL+J</b>
Lower Ceiling by 8 Pixels	<b>PgDN</b>
Raise Ceiling by 8 Pixels	<b>PgUP</b>
Lower Floor by 8 Pixels	<b>END</b>
Raise Floor by 8 Pixels	<b>HOME</b>

<b>Things Mode</b>	
Insert Thing	<b>INS</b>
Filter	<b>-</b>

<b>3D Mode</b>	
Select Texture	<b>Mouse1</b>
Paste Texture	<b>Mouse2</b>
Copy texture	<b>Mouse3</b>
Move Texture Up	<b>Arrow Up</b>
Move Texture Down	<b>Arrow Down</b>
Move Texture left	<b>Arrow Left</b>
Move Texture Right	<b>Arrow Right</b>
Remove Texture	<b>DEL</b>
Increase Brightness	<b>CTRL+Scroll-Up</b>
Decrease Brightness	<b>CTRL+Scroll-Down</b>
Toggle Gravity	<b>G</b>
Toggle Light	<b>B</b>
Toggle Info Panel	<b>-</b>
Toggle Lower Unpegged	<b>L</b>
Toggle Middle Texture	<b>T</b>
Toggle Upper Unpegged	<b>U</b>
Forward	<b>D</b>
Backward	<b>E</b>
Strafe Left	<b>S</b>
Strafe Right	<b>F</b>
Exit 3D Mode	<b>W</b>
Auto-Align Textures	<b>A</b>
Lower Floor/Ceiling by 1	<b>SHIFT+Scroll-Down</b>
Lower Floor/Ceiling by 8	<b>Scroll-Down</b>
Raise Floor/Ceiling by 1	<b>SHIFT+Scroll-Up</b>
Raise Floor/Ceiling by 8	<b>Scroll-Up</b>

Copy Offset	<b>SHIFT+C</b>
Paste Offset	<b>SHIFT+V</b>

<b>File and Help Menu</b>	
New Map	<b>CTRL+N</b>
Open Map	<b>CTRL+O</b>
Save Map	<b>CTRL+S</b>
Export Map	<b>CTRL+SHIFT</b>
Save Map Into	<b>CTRL+F12</b>
Save Map As	<b>F12</b>
Close Map	-
Test Map	<b>F8</b>
Build Nodes	<b>CTRL+F8</b>
Help: About	-
Help: FAQ	<b>F1</b>
Help: Website	-

These are the default **SHORTCUT KEYS**. Press F5 to bring up the **CONFIGURATION** dialog box and then click the **SHORTCUTS** tab. To change a key, select the function and current shortcut then enter the key or key combination in the text box by pressing the actual keys you want to use. For instance, to change **ZOOM IN** from **Scroll-Up** to **NUM +**, don't type "NUM +" – just hit the + key in the Numeric Keypad. Be careful not to assign the same key combination to more than one function.

## Appendix B:

### Node Builder Parameters: ZenNode, ZDBSP, and BSP

#### ZenNode Options

By Marc Rousseau

##### Parameters

###### Expression:

```
ZenNode {-options} filename[.wad] [level{+level}] {-o|x output[.wad]}
```

**-o** Specifies output.wad. If not specified, ZenNode writes a file with the same name as input (and overwrites original if present).

```
zennode doom.wad -o zendoom.wad
```

**-x** Extract only the levels that have been processed from a multilevel WAD to output WAD.

```
zennode doom.wad e2m3+e3m4 -x zendoom.wad
```

**file1+** Merge individual WAD files into a larger one.

```
zennode this.wad+that.wad -o zendoom.wad
```

**-t** Don't write output file. Test mode.

##### Nodes Options

**-n-** Turn off nodes builder.

```
zennode -n- doom.wad
```

**-n1** Fast algorithm for quicker nodes build.

**-n2** Faster algorithm.

**-n3** Fastest algorithm.

**-nq** Stop progress indications from being displayed.

**-nu** Forces all SSECTORS to contain SEGS from only one SECTORS.

**-ni** If a line can't be seen in the game, don't include it in the BSP tree.

## BLOCKMAP Options

- b-** Turn off BLOCKMAP build.  

```
zennode -b- doom.wad
```
- bc** Compress the BLOCKMAP.

## REJECT Options

- r-** Turn off REJECT build.  

```
zennode -r- doom.wad
```
- rz** Insert a 0-filled REJECT.
- rf** Force rebuild, even if effects are detected.
- rc** Use child sectors to reduce LOS calculations.
- rg** Use graphs to locate articulation points in the map to reduce LOS calculations.

*For ZenNode v1.2.1*

## ZDBSP Options

By Randy Heit

ZDBSP supports several command line options to affect its behavior. If you forget what they are, you can view a quick summary of them by running ZDBSP without any options. They are also listed below in more detail than the listing ZDBSP provides. Note that these options are case-sensitive, so **-r** is not the same thing as **-R**. You can use either the long form of an option or the short form depending on your preference.

### **--help**

Displays a summary of all ZDBSP's command line options, as if you had run ZDBSP without any options.

### **--version** or **-V**

Displays ZDBSP's version number.

### **--map=MAP** or **-m**

When you use this option, ZDBSP will only build the nodes for one map in a wad instead of rebuilding the nodes for every map contained in the wad. *MAP* should be the map's full name (e.g. MAP01 or E1M1).

### **--output=FILE** or **-o**

Normally, ZDBSP creates a new wad named *tmp.wad*. You can use this option to make it write to a different file instead.

### **--no-prune** or **-q**

When you use this option, ZDBSP will not remove unused SideDefs or Sectors from a map. ZDBSP will always remove 0-length LineDefs from a map even when you use this option because it is possible for them to make the game crash under certain

circumstances. Moreover, ZDoom will itself remove 0-length LineDefs and rebuild the nodes if it finds any.

**--no-nodes or -N**

This option causes ZDBSP to take the node information from the old wad file and write it to the new wad file. I can't think of any reason why you would want to do this, but it is provided as an option nonetheless.

**--gl or -g**

This option causes ZDBSP to build two sets of nodes for a map: One set of regular nodes and one set of GL nodes. Because ZDBSP is doing twice the work, it will take twice as long to finish.

**--gl-matching or -G**

Like the previous option, this one will also make ZDBSP generate GL nodes. However, it will only build one set of nodes and then strip the extra GL information from them to create the normal nodes. Because of this, it is faster than the previous option when you want to create GL nodes, but the normal nodes will generally be less efficient because they were created from the GL nodes.

**--empty-blockmap or -b**

This option writes a zero-length BLOCKMAP to the wad. As of this writing, ZDoom is the only port that will detect this and build the BLOCKMAP itself.

**--empty-reject or -r**

When this option is used, ZDBSP will write a zero-length REJECT to the wad. As of this writing, ZDoom is the only port that supports this. A zero-length REJECT is the same thing as a REJECT filled with zeros. Since ZDBSP does not generate a REJECT table, you should always use this option if you intend for the map to be played solely with ZDoom.

**--zero-reject or -R**

This option is similar to the previous one, except ZDBSP will actually write out a full-sized REJECT lump filled with zeros. If you play with ZDoom, this is just wasted space, but other ports and Doom itself require a full-sized REJECT lump to work.

**--no-reject or -E**

This option makes ZDBSP copy the REJECT lump from the old wad to the new wad unchanged. However, if it detects that the old REJECT is the wrong size for the number of sectors in the map, it will be removed as if you had used the --empty-reject option.

**--no-polyobjs or -P**

This option disables ZDBSP's polyobject detection code. If you are building nodes for a map without polyobjects, you might be able to save a fraction of a second from the build time by using this option, but there is generally no reason to use it.

**--no-timing or -t**

If you don't care how long it takes to build nodes, use this option and ZDBSP won't tell you.

**--warn or -w**

Displays extra warning messages that ZDBSP might generate while building GL nodes. The nodes will still be usable if warnings occur, and warnings are not unlikely. If you have strange display problems with GL nodes, turning on the warning messages may help you locate the problem.

**--partition=NNN or -p**

This option controls the maximum number of SEGS that will be considered for splitters at each node. The default is 64. By increasing it, you might be able to generate "better" nodes, but you get diminishing returns the higher you make it. High values are also slower than low ones.

**--split-cost=NNN or -s**

This option adjusts how hard ZDBSP tries to avoid splitting SEGS. The default value is 8. By increasing this, ZDBSP will try harder to avoid splitting SEGS. If you decrease it, ZDBSP will split SEGS more frequently. More splits mean the BSP tree will usually be more balanced, but it will also take up more room.

**--diagonal-cost=NNN or -d**

This option controls how hard ZDBSP tries to avoid using diagonal splitters. The default value is 16. A higher value means that diagonal splitters will be more likely to be used. This can sometimes help to reduce the number of SEGS in a map. The reason avoiding diagonal splitters is important is because normal nodes do not store any fractional information for the vertices that SEGS use, so you are more likely to see slime trails with diagonal splitters than with horizontal or vertical splitters.

**--view or -v**

Under Windows, this displays a viewer that will let you inspect a map's subsectors, nodes, and REJECT. To scroll the map around, drag it with your right mouse button. The viewer was created to assist with debugging with the GL nodes. It is not very user-friendly nor is it bug-free. In fact, if you try to build nodes for more than one map with this option, there is a good chance ZDBSP will crash.

*For ZDBSP v1.5*

## BSP v5.1 Options

By Colin Reed, Lee Killough, and Colin Phipps

### Parameters

**Expression:**

```
bsp [ -noreject ] [-factor nn ] [ -q ] [ -picknode { traditional |  
  visplane } ] [ -blockmap { old | comp } ] inwad [ [ -o ] outwad ]
```

Where:

**-noreject**

Causes any existing REJECT lump in the WAD file not to be replaced.

**-factor nn**

Used for tuning the node builder. The number supplied is the weighting applied when a choice of nodeline requires other lines to be split. Increasing this value from the Default of 17 will reduce the number of extra line splits, but this will generally cause a less balanced node tree. The default is usually fine.

**-q**

Causes BSP to run quietly, only printing output if there are errors or warnings.

**-picknode**

Determines the nodeline selection algorithm. The "traditional" option is best for most Doom levels. For levels which are intended for the original doom2.exe and suffer from some marginal visplane overflows, the "visplane" algorithm is designed to minimize these and may help in some cases.

**-blockmap**

Selects the blockmap generation algorithm. The default "old" algorithm generates a simple and correct blockmap. The newer "comp" version produces a compressed blockmap, by reusing identical blocks which should be equivalent in actual use. The "comp" version is therefore better but it is relatively untested so is not yet enabled by default.

**inwad** is the *input WAD file*.

This may contain any number of levels and other lumps. The nodes and associated data resources will be built for every level in this WAD. Any other data present in the WAD will be copied to the output WAD unchanged.

**outwad** is the *output WAD file*.

If the output file already exists, BSP will write its output to a temporary file while it is working, and will only overwrite the output file once it is finished. In particular, it is safe for **outwad** to be the same as **inwad**, although this is not recommended unless you keep other backups.

Either **inwad** or **outwad** can be pipes or special files. On most UNIX systems, you can have BSP read from STDIN and write to STDOUT by using it as follows:

```
bsp -q /dev/stdin /dev/stdout
```

*For BSP v5.1*

# Appendix C:

## LineDef Action Type Reference

By Jim F. Flynn

## LineDef Action Types by [Function](#) and [Number](#)

[DOORS](#) | [CRUSHERS](#) | [CEILINGS](#) | [LIFTS](#) | [STAIRS](#) | [FLOORS](#) | [LIGHTING](#) | [TELEPORT](#) | [END LEVEL](#) | [MISC](#) | [TAGS](#)

### Credits

Special thanks to Matt Fell whose [The Unofficial Doom Specs](#) provided much new information for this revision, and [which] has been invaluable to WAD writers and DOOMers all along.

Thanks to Dr Sleep (aka John W. Anderson) for providing early information on the 1.666 LineDefs and for advice and encouragement all along. If you're looking for HERETIC information you want his [htlndf11.txt](#) file.

Thanks to Neil Bonner for pointing out the 667 Sector tag I was missing and thereby motivating this current revision.

### Legend

#### Abbreviations

**N.** = neighbor  
**MIN** = minimum  
**MAX** = maximum  
**INC** = inclusive  
**EXC** = exclusive

A \* appears on the left of any description line that only works for **DOOM 1.666** engine and above.

### LineDef Description Headers

#### Example:

#	u1	u2	Trigger	Brief Description	Qualifier	Speed	Sound
59	5	7	W1&	Open/wait 4/close	T/SEC/DMG	slow	(mover)

**#** is the LINEDEF FUNCTION TYPE number 0-143

**u1** and **u2** are the number of occurrences of a TRIGGER in DOOM1 and DOOM2.

**TRIGGER** represents the conditions that cause the function to be activated.

- The TRIGGER symbol may start with **n** if the function does not require a Sector TAG to operate.



### The basic trigger symbol letters are as follows:

- **S** = SWITCH/door, FIRST SIDEDEF must be used with spacebar to activate.
- **W** = WALKOVER only teleport LineDefs require approach from FIRST SIDEDEF.
- **M** = MONSTER WALKOVER, activated only by monster walking over line.
- **G** = IMPACT TRIGGER, activated on hit: fists, chain, bullet, shell.
- **-** = no TRIGGER required (like animated wall).

The next letter in the TRIGGER symbol is the repeatability of the function: **1** for ONCE only (per LineDef) and **R** for REPEATABLE. This may be "-" if the repeatability does not apply, as for *End Level*.

The final letter in the TRIGGER symbol can be **m** or **&**. The **m** indicates that a monster can activate the function. The **&** indicates that once activated, all other functions on the Sector are locked out even after the **&** function is completed.

**BRIEF DESCRIPTION** attempts to state in English what the function does.

**QUALIFIER** shows which key is required, crusher attribute, texture changes.

**SPEED** is the rough relative up/down velocity involved in the operation.

**SOUND** is the name of the sound associated with the action.

### Texture Change Descriptions

Texture changes involve copying attributes from another Sector to the one that is changing. If the Sector being copied is based on the line triggering the change, the description is prefaced with **T**: to indicate a trigger model change. This means that the Sector on the FIRST SIDEDEF of the triggering LineDef is the one copied in the change. If the Sector being copied is based on the Sector being changed, then it is the Sector that is on the other side of the lowest numbered two-sided LineDef in the changing Sector. Such a change is prefaced with **N**, for a numeric model change.

The remainder of the texture change description lists the attributes copied. **0** means that the changing Sector type is set to 0 and only the floor texture is copied. **SEC** means that the secret attribute of the model Sector is copied. **DMG** means that the damage attribute of the model Sector is copied. Floor textures are **ALWAYS** copied. Ceiling textures, lighting attributes and heights are **NEVER** copied.

### Floor Motion Directions

Floor LineDefs described as **DOWN TO** functions will move the floor at the speed indicated if the target height is lower. The height change is instantaneous if motion is in the other direction. Floor LineDefs described as **UP TO** act in a similar fashion but in the opposite direction.

### Door Functions

A door function described using **COMMAS** only works when the door is stable in the opposite state to the function. A door function described using **SLASHES** will work anytime to toggle the opening/closing state of the door.

## LineDef Types by Function

<b>CEILING MOVERS</b>			<b>DOORS, CRUSHERS, CEILINGS</b>				
<b>DOORS</b>			<b>MANUAL DOORS (no Sector Tag required)</b>				
#	u1	u2	Trigger	Brief Description	Qualifier	Speed	Sound
1	281	220	nSRm	Open/wait 4/close		med	(door)
26	22	14	nSR	Open/wait 4/close	BLUE KEY	med	(door)
27	26	12	nSR	Open/wait 4/close	YELLOW KEY	med	(door)
28	10	9	nSR	Open/wait 4/close	RED KEY	med	(door)
117	*0	47	nSR	Open/wait 4/close		turbo	(blaze)
31	76	45	nS1	Open		med	(door)
32	15	40	nS1	Open	BLUE KEY	med	(door)
34	19	27	nS1	Open	YELLOW KEY	med	(door)
33	14	24	nS1	Open	RED KEY	med	(door)
118	*0	8	nS1	Fast open		turbo	(blaze)

<b>CEILING MOVERS</b>			<b>DOORS, CRUSHERS, CEILINGS</b>				
<b>DOORS</b>			<b>REMOTE DOORS (Sector Tag required)</b>				
#	u1	u2	Trigger	Brief Description	Qualifier	Speed	Sound
29	1	0	S1	Open/wait 4/close		med	(door)
63	38	15	SR	Open/wait 4/close		med	(door)
4	0	1	W1	Open/wait 4/close		med	(door)
90	21	17	WR	Open/wait 4/close		med	(door)
103	41	32	S1	Open		med	(door)
61	9	36	SR	Open		med	(door)
2	114	64	W1	Open		med	(door)
86	9	3	WR	Open		med	(door)
46	13	22	GR	Open		med	(door)
111	*0	0	S1	Fast open/wait 4/close		turbo	(blaze)
114	*0	51	SR	Fast open/wait 4/close		turbo	(blaze)
108	*0	0	W1	Fast open/wait 4/close		turbo	(blaze)
105	*0	23	WR	Fast open/wait 4/close		turbo	(blaze)
112	*0	4	S1	Fast open		turbo	(blaze)
115	*0	11	SR	Fast open		turbo	(blaze)
109	*0	99	W1	Fast open		turbo	(blaze)
106	*0	6	WR	Fast open		turbo	(blaze)
133	*0	0	S1	Fast open	BLUE KEY	turbo	(blaze)
99	*0	2	SR	Fast open	BLUE KEY	turbo	(blaze)
135	*0	16	S1	Fast open	RED KEY	turbo	(blaze)
134	*0	4	SR	Fast open	RED KEY	turbo	(blaze)
137	*0	6	S1	Fast open	YELLOW KEY	turbo	(blaze)
136	*0	4	SR	Fast open	YELLOW KEY	turbo	(blaze)
50	*0	0	S1	Close		med	(door)
42	6	1	SR	Close		med	(door)
3	2	9	W1	Close		med	(door)
75	6	0	WR	Close		med	(door)

113	*0	0	S1	Fast close		turbo	(blaze)
116	*0	1	SR	Fast close		turbo	(blaze)
110	*0	1	W1	Fast close		turbo	(blaze)
107	*0	0	WR	Fast close		turbo	(blaze)
16	3	2	W1	Close/wait 30/open		med	(door)
76	2	2	WR	Close/wait 30/open		med	(door)

**CEILING MOVERS**      **DOORS, CRUSHERS, CEILINGS**  
**CRUSHERS**

#	u1	u2	Trigger	Brief Description	Qualifier	Speed	Sound
6	0	0	W1&	Start fast (non-fatal)		med	(crush)
77	6	3	WR&	Start fast (non-fatal)		med	(crush)
49	0	1	S1&	Start slow (fatal)		slow	(crush)
25	0	0	W1&	Start slow (fatal)		med	(crush)
73	17	6	WR&	Start slow (fatal)		med	(crush)
141	*0	1	W1&	Start slow silent (fatal)		slow	(quiet)
57	0	0	W1&	Stop crusher			
74	24	13	WR&	Stop crusher			

**CEILING MOVERS**      **DOORS, CRUSHERS, CEILINGS**  
**CEILINGS**

#	u1	u2	Trigger	Brief Description	Qualifier	Speed	Sound
40	4	0	W1	Up to max ceil exc		slow	(mover)
41	0	0	S1	Down to floor		slow	(mover)
43	0	0	SR	Down to floor		slow	(mover)
44	1	0	W1	Down to floor +8		slow	(mover)
72	0	0	WR	Down to floor +8		slow	(mover)

**FLOOR MOVERS**      **LIFTS, STAIRS, FLOORS**  
**LIFTS**

#	u1	u2	Trigger	Brief Description	Qualifier	Speed	Sound
21	1	0	S1	Lower/wait 3/raise		fast	(lift)
62	51	143	SR	Lower/wait 3/raise		fast	(lift)
10	1	0	W1	Lower/wait 3/raise		fast	(lift)
88	65	51	WR	Lower/wait 3/raise		fast	(lift)
122	0	0	S1	Fast lower/wait 3/raise		turbo	(lift)
123	0	162	SR	Fast lower/wait 3/raise		turbo	(lift)
121	0	0	W1	Fast lower/wait 3/raise		turbo	(lift)
120	0	58	WR	Fast lower/wait 3/raise		turbo	(lift)

**FLOOR MOVERS**      **LIFTS, STAIRS, FLOORS**  
**STAIRS**

#	u1	u2	Trigger	Brief Description	Qualifier	Speed	Sound
7	11	6	S1	Raise 8		slow	(mover)
8	2	1	W1	Raise 8		slow	(mover)
127	0	6	S1	Fast raise 16	CRUSH	turbo	(mover)
100	0	1	W1	Fast raise 16	CRUSH	turbo	(mover)

FLOOR MOVERS			LIFTS, STAIRS, FLOORS				
FLOORS							
#	u1	u2	Trigger	Brief Description	Qualifier	Speed	Sound
58	5	0	W1	Absolute rise 24		slow	(mover)
92	0	0	WR	Absolute rise 24		slow	(mover)
15	0	1	S1&	Absolute rise 24	T:0	slow	(mover)
66	0	0	SR&	Absolute rise 24	T:0	slow	(mover)
59	5	7	W1&	Absolute rise 24	T:SEC/DMG	slow	(mover)
93	0	0	WR&	Absolute rise 24	T:SEC/DMG	slow	(mover)
14	1	0	S1&	Absolute rise 32	T:0	slow	(mover)
67	0	9	SR&	Absolute rise 32	T:0	slow	(mover)
140	*0	1	S1	Absolute rise 512		med	(mover)
102	14	22	S1	Down to max floor exc		slow	(mover)
45	0	2	SR	Down to max floor exc		slow	(mover)
19	11	22	W1	Down to max floor exc		slow	(mover)
83	0	7	WR	Down to max floor exc		slow	(mover)
71	0	24	S1	Down to max floor exc +8		fast	(mover)
70	3	2	SR	Down to max floor exc +8		fast	(mover)
36	8	27	W1	Down to max floor exc +8		fast	(mover)
98	4	0	WR	Down to max floor exc +8		fast	(mover)
23	16	12	S1	Down to min floor exc		slow	(mover)
60	0	4	SR	Down to min floor exc		slow	(mover)
38	23	37	W1	Down to min floor exc		slow	(mover)
82	6	0	WR	Down to min floor exc		slow	(mover)
37	31	11	W1	Down to min floor exc	N:SEC/DMG	slow	(mover2)
84	0	0	WR	Down to min floor exc	N:SEC/DMG	slow	(mover2)
20	13	12	S1&	Up to next floor exc	T:0	slow	(mover)
68	0	2	SR&	Up to next floor exc	T:0	slow	(mover)
22	3	18	W1&	Up to next floor exc	T:0	slow	(mover)
95	0	0	WR&	Up to next floor exc	T:0	slow	(mover)
47	0	2	G1&	Up to next floor exc	T:0	slow	(mover)
18	10	8	S1	Up to next floor exc		slow	(mover)
69	0	0	SR	Up to next floor exc		slow	(mover)
119	*0	13	W1	Up to next floor exc		slow	(mover)
128	*0	0	WR	Up to next floor exc		slow	(mover)
131	*0	2	S1	Up to next floor exc		turbo	(mover)
132	*0	0	SR	Up to next floor exc		turbo	(mover)
130	*0	0	W1	Up to next floor exc		turbo	(mover)
129	*0	0	WR	Up to next floor exc		turbo	(mover)
101	0	0	S1	Up to min ceil inc		slow	(mover)
64	0	0	SR	Up to min ceil inc		slow	(mover)
5	1	3	W1	Up to min ceil inc		slow	(mover)
91	18	1	WR	Up to min ceil inc		slow	(mover)
24	1	0	G1	Up to min ceil inc		slow	(mover)
55	0	0	S1	Up to min ceil inc -8	CRUSH	slow	(mover)

65	0	0	SR	Up to min ceil inc -8	CRUSH	slow	(mover)
56	5	0	W1&	Up to min ceil inc -8	CRUSH	slow	(mover)
94	0	1	WR&	Up to min ceil inc -8	CRUSH	slow	(mover)
53	0	0	W1&	Move min<->max floor inc		slow	(lift)
87	6	4	WR&	Move min<->max floor inc		slow	(lift)
54	0	0	W1&	Stop moving floor			
89	10	5	WR&	Stop moving floor			
30	2	2	W1	Rise by shortest outer lower		slow	(mover)
96	0	0	WR	Rise by shortest outer lower		slow	(mover)
9	2	0	S1	Donut function	N:SEC/DMG	slow	(mover5)

**OTHER EFFECTS LIGHTING, TELEPORTERS, END LEVEL, MISC**  
**LIGHTING**

#	u1	u2	Trigger	Brief Description	Qualifier	Speed	Sound
139	*0	2	SR	Light level to 0			(clunk)
35	3	14	W1	Light level to 0			
79	0	0	WR	Light level to 0			
138	*0	2	SR	Light level to 255			(clunk)
13	4	1	W1	Light level to 255			
81	0	0	WR	Light level to 255			
17	0	0	W1	Start 1 sec blinking (type 3)			
12	0	0	W1	Light to max n. light exc			
80	0	0	WR	Light to max n. light exc			
104	2	0	W1	Light to min n. light exc			

**OTHER EFFECTS LIGHTING, TELEPORTERS, END LEVEL, MISC**  
**TELEPORTERS**

#	u1	u2	Trigger	Brief Description	Qualifier	Speed	Sound
39	3	3	W1m	Teleport			(tport)
97	125	411	WRm	Teleport			(tport)
125	*0	4	W1m	Monster only teleport			(tport)
126	*0	20	WRm	Monster only teleport			(tport)

**OTHER EFFECTS LIGHTING, TELEPORTERS, END LEVEL, MISC**  
**END LEVEL**

#	u1	u2	Trigger	Brief Description	Qualifier	Speed	Sound
11	15	19	nS-	End level. Go to next level			(clunk)
52	19	55	nW-	End level. Go to next level			(clunk)
51	3	1	nS-	End level. Go to secret level			(clunk)
124	*0	4	nW-	End level. Go to secret level			(clunk)

**OTHER EFFECTS LIGHTING, TELEPORTERS, END LEVEL, MISC**  
**MISC**

#	u1	u2	Trigger	Brief Description	Qualifier	Speed	Sound
48	99	103	n--	Animated wall			
0	0	0	n--	Null tag indicating no function is assigned			

## Special Sector Tags

**0** TAG used to specify no tag assigned. However, a LineDef assigned to tag 0 (none) will attempt to carry its function out on all 0 tagged Sectors unless it also has 0 LineDef type. This usually causes the game to crash and is always weird - don't do it.

**99** Artifact created by bug/feature of Id's BSP that collapses Sectors that touch and have identical Sector attributes. Used to prevent this. Has no effect on rising stairs other than this. Not necessary at all with Colin Reed's BSP.

**666** In a transition level, when last Boss dies - Lower floor to min floor.

- DOOM I Boss = BARON, CYBERDEMON, SPIDER MASTERMIND
- DOOM II Boss = MANCUBUS, KEEN
- If **EnM8** and no 666 tag exists level ends on Boss death. The level cannot be ended on boss death in DOOM II, only Romero death.
- A transition level is **EnM8** in DOOM I, **MAP07**, and an unknown list of others in DOOM II.

**667** In a transition level, when last ARACHNOTRON dies - Floor rises 64. It is unknown if this tag has any effect or use in DOOM I.

**999** cf 99.

## LineDef Types by Number

#	u1	u2	Trigger	Brief Description	Qualifier	Speed	Sound
0	0	0	n--	Null tag indicating no function			
1	281	220	nSRm	Open/wait 4/close		med	(door)
2	114	64	W1	Open		med	(door)
3	2	9	W1	Close		med	(door)
4	0	1	W1	Open/wait 4/close		med	(door)
5	1	3	W1	Up to min ceil inc		slow	(mover)
6	0	0	W1&	Start fast (non-fatal)		med	(crush)
7	11	6	S1	Raise 8		slow	(mover)
8	2	1	W1	Raise 8		slow	(mover)
9	2	0	S1	Donut function	N:SEC/DMG	slow	(mover)
10	1	0	W1	Lower/wait 3/raise		fast	(lift)
11	15	19	nS-	End level. Go to next level			(clunk)
12	0	0	W1	Light to max n. light exc			
13	4	1	W1	Light level to 255			
14	1	0	S1&	Absolute rise 32	T:0	slow	(mover)
15	0	1	S1&	Absolute rise 24	T:0	slow	(mover)
16	3	2	W1	Close/wait 30/open		med	(door)
17	0	0	W1	Start 1 sec blinking (type 3)			
18	10	8	S1	Up to next floor exc		slow	(mover)
19	11	22	W1	Down to max floor exc		slow	(mover)
20	13	12	S1&	Up to next floor exc	T:0	slow	(mover)
21	1	0	S1	Lower/wait 3/raise		fast	(lift)
22	3	18	W1&	Up to next floor exc	T:0	slow	(mover)
23	16	12	S1	Down to min floor exc		slow	(mover)
24	1	0	G1	Up to min ceil inc		slow	(mover)
25	0	0	W1&	Start slow (fatal)		med	(crush)
26	22	14	nSR	Open/wait 4/close	BLUE KEY	med	(door)
27	26	12	nSR	Open/wait 4/close	YELLOW KEY	med	(door)
28	10	9	nSR	Open/wait 4/close	RED KEY	med	(door)
29	1	0	S1	Open/wait 4/close		med	(door)
30	2	2	W1	Rise by shortest outer lower		slow	(mover)
31	76	45	nS1	Open		med	(door)
32	15	40	nS1	Open	BLUE KEY	med	(door)
33	14	24	nS1	Open	RED KEY	med	(door)
34	19	27	nS1	Open	YELLOW KEY	med	(door)
35	3	14	W1	Light level to 0			
36	8	27	W1	Down to max floor exc +8		fast	(mover)
37	31	11	W1	Down to min floor exc	N:SEC/DMG	slow	(mover)
38	23	37	W1	Down to min floor exc		slow	(mover)
39	3	3	W1m	Teleport			(tport)
40	4	0	W1	Up to max ceil exc		slow	(mover)

41	0	0	S1	Down to floor		slow	(mover)
42	6	1	SR	Close		med	(door)
43	0	0	SR	Down to floor		slow	(mover)
44	1	0	W1	Down to floor +8		slow	(mover)
45	0	2	SR	Down to max floor exc		slow	(mover)
46	13	22	GR	Open		med	(door)
47	0	2	G1&	Up to next floor exc	T:0	slow	(mover)
48	99	103	n--	Animated wall			
49	0	1	S1&	Start slow (fatal)		slow	(crush)
50	*0	0	S1	Close		med	(door)
51	3	1	nS-	End level. Go to secret level			(clunk)
52	19	55	nW-	End level. Go to next level			(clunk)
53	0	0	W1&	Move min<->max floor inc		slow	(lift)
54	0	0	W1&	Stop moving floor			
55	0	0	S1	Up to min ceil inc -8	CRUSH	slow	(mover)
56	5	0	W1&	Up to min ceil inc -8	CRUSH	slow	(mover)
57	0	0	W1&	Stop crusher			
58	5	0	W1	Absolute rise 24		slow	(mover)
59	5	7	W1&	Absolute rise 24	T:SEC/DMG	slow	(mover)
60	0	4	SR	Down to min floor exc		slow	(mover)
61	9	36	SR	Open		med	(door)
62	51	143	SR	Lower/wait 3/raise		fast	(lift)
63	38	15	SR	Open/wait 4/close		med	(door)
64	0	0	SR	Up to min ceil inc		slow	(mover)
65	0	0	SR	Up to min ceil inc -8	CRUSH	slow	(mover)
66	0	0	SR&	Absolute rise 24	T:0	slow	(mover)
67	0	9	SR&	Absolute rise 32	T:0	slow	(mover)
68	0	2	SR&	Up to next floor exc	T:0	slow	(mover)
69	0	0	SR	Up to next floor exc		slow	(mover)
70	3	2	SR	Down to max floor exc +8		fast	(mover)
71	0	24	S1	Down to max floor exc +8		fast	(mover)
72	0	0	WR	Down to floor +8		slow	(mover)
73	17	6	WR&	Start slow (fatal)		med	(crush)
74	24	13	WR&	Stop crusher			
75	6	0	WR	Close		med	(door)
76	2	2	WR	Close/wait 30/open		med	(door)
77*	6	3	WR&	Start fast (non-fatal)		med	(crush)
79	0	0	WR	Light level to 0			
80	0	0	WR	Light to max n. light exc			
81	0	0	WR	Light level to 255			
82	6	0	WR	Down to min floor exc		slow	(mover)
83	0	7	WR	Down to max floor exc		slow	(mover)
84**	0	0	WR	Down to min floor exc	N:SEC/DMG	slow	(mover)
86	9	3	WR	Open		med	(door)
87	6	4	WR&	Move min<->max floor inc		slow	(lift)



88	65	51	WR	Lower/wait 3/raise		fast	(lift)
89	10	5	WR&	Stop moving floor			
90	21	17	WR	Open/wait 4/close		med	(door)
91	18	1	WR	Up to min ceil inc		slow	(mover)
92	0	0	WR	Absolute rise 24		slow	(mover)
93	0	0	WR&	Absolute rise 24	T:SEC/DMG	slow	(mover)
94	0	1	WR&	Up to min ceil inc -8	CRUSH	slow	(mover)
95	0	0	WR&	Up to next floor exc	T:0	slow	(mover)
96	0	0	WR	Rise by shortest outer lower		slow	(mover)
97	125	411	WRm	Teleport			(tport)
98	4	0	WR	Down to max floor exc +8		fast	(mover)
99	*0	2	SR	Fast open	BLUE KEY	turbo	(blaze)
100	0	1	W1	Fast raise 16	CRUSH	turbo	(mover)
101	0	0	S1	Up to min ceil inc		slow	(mover)
102	14	22	S1	Down to max floor exc		slow	(mover)
103	41	32	S1	Open		med	(door)
104	2	0	W1	Light to min n. light exc			
105	*0	23	WR	Fast open/wait 4/close		turbo	(blaze)
106	*0	6	WR	Fast open		turbo	(blaze)
107	*0	0	WR	Fast close		turbo	(blaze)
108	*0	0	W1	Fast open/wait 4/close		turbo	(blaze)
109	*0	99	W1	Fast open		turbo	(blaze)
110	*0	1	W1	Fast close		turbo	(blaze)
111	*0	0	S1	Fast open/wait 4/close		turbo	(blaze)
112	*0	4	S1	Fast open		turbo	(blaze)
113	*0	0	S1	Fast close		turbo	(blaze)
114	*0	51	SR	Fast open/wait 4/close		turbo	(blaze)
115	*0	11	SR	Fast open		turbo	(blaze)
116	*0	1	SR	Fast close		turbo	(blaze)
117	*0	47	nSR	Open/wait 4/close		turbo	(blaze)
118	*0	8	nS1	Fast open		turbo	(blaze)
119	*0	13	W1	Up to next floor exc		slow	(mover)
120	0	58	WR	Fast lower/wait 3/raise		turbo	(lift)
121	0	0	W1	Fast lower/wait 3/raise		turbo	(lift)
122	0	0	S1	Fast lower/wait 3/raise		turbo	(lift)
123	0	162	SR	Fast lower/wait 3/raise		turbo	(lift)
124	*0	4	nW-	End level. Go to secret level			(clunk)
125	*0	4	W1m	Monster only teleport			(tport)
126	*0	20	WRm	Monster only teleport			(tport)
127	0	6	S1	Fast raise 16	CRUSH	turbo	(mover)
128	*0	0	WR	Up to next floor exc		slow	(mover)
129	*0	0	WR	Up to next floor exc		turbo	(mover)
130	*0	0	W1	Up to next floor exc		turbo	(mover)
131	*0	2	S1	Up to next floor exc		turbo	(mover)
132	*0	0	SR	Up to next floor exc		turbo	(mover)

133	*0	0	S1	Fast open	BLUE KEY	turbo	(blaze)
134	*0	4	SR	Fast open	RED KEY	turbo	(blaze)
135	*0	16	S1	Fast open	RED KEY	turbo	(blaze)
136	*0	4	SR	Fast open	YELLOW KEY	turbo	(blaze)
137	*0	6	S1	Fast open	YELLOW KEY	turbo	(blaze)
138	*0	2	SR	Light level to 255			(clunk)
139	*0	2	SR	Light level to 0			(clunk)
140	*0	1	S1	Absolute rise 512		med	(mover)
141	*0	1	W1&	Start slow silent (fatal)		slow	(quiet)

\*There is no 78 and \*\*no 85.