Introduction

Doom Builder is an advanced, revolutionary map editor for Doom and games based on the Doom engine, such as Heretic, Hexen and Strife. This editor is highly extendible for the different game engines of the Doom community. Doom Builder introduced the 3D editing mode in the Doom community and is still the leading editor for Doom levels today.

Reference Manual

This is the Reference Manual for Doom Builder 2. You can use this manual to look up how the editor works and what the specific modes and actions do. This manual is not a beginners tutorial that teaches you how to make maps.

Terminology

Resource

A file or directory from which textures, flats, sprites and other information is read that is used during map editing. Doom Builder supports WAD files, PK3 files and directories as resources.

WAD

A WAD file is a collection of data, which can include textures, sprites, sounds but also maps. This is the most common resource type used by Doom. A WAD file can be an **IWAD** (Internal WAD) which contains all data needed to run a game without any additional WAD files, or a **PWAD** (Patch WAD) which only contains the data you wish to change (for example; a new map and some textures only, but no change in sounds or sprites). Doom always requires a single IWAD and optionally one or more PWADs.

PK3

The PK3 file is actually a ZIP file and contains a directory with a specific structure. It can be used as a replacement for WAD files in some Doom sourceports and is smaller than the WAD file because it is compressed. Doom Builder can read from PK3 files as resources, but for better performance it is recommended that you unzip your PK3 file (keeping the directory structure) and add the directory as resource. For more information, see Using ZIPs as WAD replacement for more information.

Element

A map consists of vertices, linedefs, sidedefs, sectors and things. These are all elements in a map and each has their own set of properties. Some elements are connected to other elements: A linedef is always connected to two vertices and has one or two sidedefs. A sidedef is always connected to one sector and a sector has one or more sidedefs.

Vertex

This is the most simple element in a map. A vertex is a point on the map which has X and Y coordinates.

Linedef

This is a line in your map geometry which connects two vertices. Every wall and sector border must have a linedef. Linedefs can have an action that is triggered when player pushes the wall or walks over the line. The linedef has a front (right) and a back (left) side where a sidedef can be attached. The front side should always have a sidedef attached, but when the linedef is a wall with nothing behind it (void), it should not have a back side.

Sidedef

A sidedef defines how one side of a linedef looks like and which sector it forms. A sidedefs has a upper, middle and lower texture which are sometimes required depending on the heights of the sectors and all share the same texture offsets.

Sector

The sector defines an area on the map. It has properties for the floor and ceiling and can have special effects for the environment. The shape of the sector is defined by the sidedefs and the linedefs they are on and should always be a closed area or multiple closed areas. There are exceptional cases, however, where authors use a non-closed sector to create a special effect by exploiting the Doom engine.

Thing

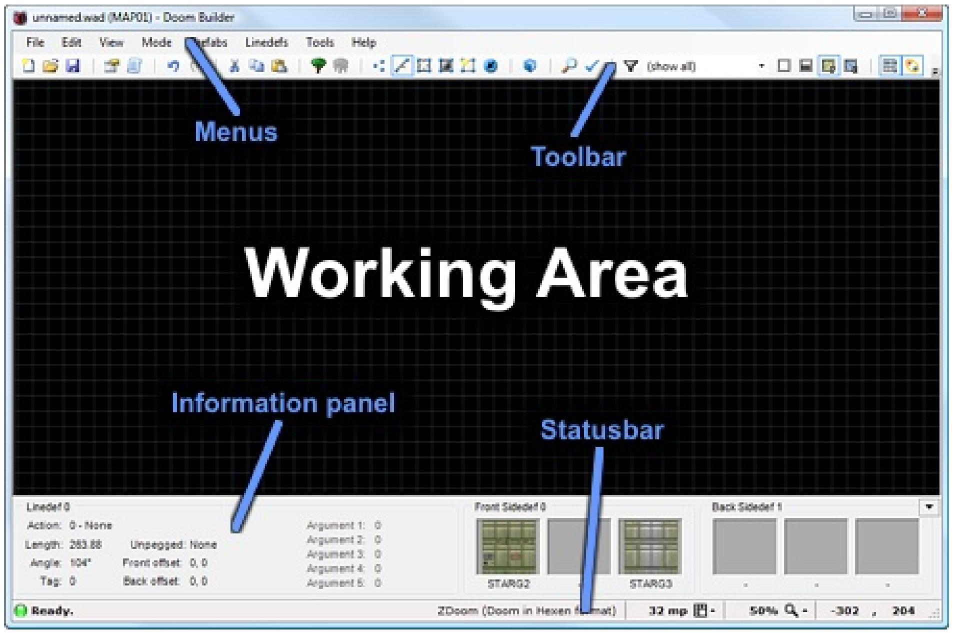
Things are basically any object in the map that is not part of its geometry. They can be decorations, items, monsters, player starts or even indicators for the sourceport to do something in that location. They have X and Y coordinates and in Hexen format they also have a Z coordinate (which often is relative to the sector floor).

About the User Interface

Doom Builder's user interface exists of one main editing window in which most of the map editing will be done. Doom Builder also has a window for scripting and several built-in dialogs for editing and setting up your preferences and much more. This part of the Reference Manual will help you explain all the buttons and controls on every window.

Main Window

The main interface window consists of the following parts;


unnamed.wad (MAP01) - Doom Builder
File Edit View Mode Prefabs Linedefs Tools Help
(show all)
Menus
Toolbar
Working Area
Information panel
Statusbar

Linedef 0
Action: 0 - None
Length: 283.88 Unpegged: None
Angle: 104° Front offset: 0, 0
Tag: 0 Back offset: 0, 0
Argument 1: 0
Argument 2: 0
Argument 3: 0
Argument 4: 0
Argument 5: 0
Front Sidedef 0
STARG2 - STARG3
Back Sidedef 1
- - -
Ready. ZDoom (Doom in Hexen format) 32 mp 50% -302 , 204

Menus

The menus are very straight-forward and you'll find most common features in the usual places. File related actions in the File menu, view related actions in the View menu and editing related actions in the Edit menu.

Toolbar

Your most needed actions and options are on the toolbar, right below the menu.

Information panel

When highlighting or targeting a specific element in the working area, a small summary about that element is displayed in this panel. You can minimize the information panel by clicking on the little arrow button on the right to maximize your working area.

Statusbar

Doom Builder shows you the current status in the statusbar. On the left is a small LED that turns green when Doom Builder is idle. When the LED turns yellow, it means Doom Builder is doing some background work. When the LED is red, then Doom Builder is busy performing an action. Right next to the status LED is a description of the current status. It also shows a result description and flashes the LED when performing and action.

On the right of the statusbar are grid size (in mappixels), the zoom percentage and the current mouse coordinates. You can click the buttons next to the grid size and the zoom percentage to bring up a related menu that allows you to change any of these settings.

Custom Fields Editor

The UDMF format allows any number of custom properties on every element in the map. With the custom fields editor, you can edit the known properties and add custom ones yourself. The game configuration has known properties that are supported by the sourceport and are automatically shown in the list. Grayed items are known, but have not been set (and thus have their default value). To add your own custom properties, you can click on the text **Click to add custom field** and type the name for the new property. Then you can change the type and set its value. To remove a custom property, just select it and press Delete. Known properties cannot be deleted, pressing Delete with a known property selected will just reset it back to its default value.

Errors and Warnings Window

When errors or warnings occur during certain operations these will be reported in this window. If the window does not automatically show, you can find it in the Tools menu or by pressing the default key F11. Click the **Copy Selection** button to copy the selected errors to the clipboard. Click the **Clear** button to remove all the errors from the window.

You can choose to open this window automatically when errors occur after operations such as loading a map by checking the option **Show this window when errors occur**.

Game Configurations Window

The Game Configurations window is accessible from the Tools menu, or with the default key F6. On the left of this window you can select the game configuration for which you want to change settings. Choose from the tabs on the right what you want to change.

Resources

This is a list of resources that will be loaded before any other resources. You should add the IWAD and other project-related resources here so that they are automatically loaded with every map you make with this game configuration. These resources, from top to bottom, will be loaded first before any other resources. You can drag the resource items to change their order. See also the Resource Options Window.

Nodebuilder

Building the BSP tree and other additional information such as BLOCKMAP and REJECT is useful to optimize the in-game performance and for most older sourceports and vanilla Doom it is even required. This is where you select the nodebuilder you want to run in certain cases and which settings to use. You can choose a nodebuilder to run when saving the map and when testing the map. Consult the documentation of the sourceport you are using to find out which nodebuilder and which settings you need. Advanced users may want to add more specific settings by writing a nodebuilder configuration.

Testing

No matter how well an editor works, testing to see how your map looks and performs in the sourceport is essential. Here you can choose the sourceport to test with and the parameters to use. You can also set the default skill level, but this can also be easily changed from the test dropdown menu in the toolbar of the main window. You generally do not need to specify custom parameters (the default ones should work fine) but advanced users may wish to change these parameters.

The following special placeholders can be used in the parameters;

%F	WAD file with the map that is to be tested. NOTE: this is a temporary file and not the file you opened or saved.
%WP	IWAD resource file with full path included. This is the first (highest) IWAD file that is found in the resources list.
%WF	IWAD resource filename only, without path. This is the first (highest) IWAD file that is found in the resources list.
%L	Map lump name as is set in the map options window.
%L1	The first number found in the map lump name (as is set in the map options window). This is for use with the -warp parameter.
%L2	The second number found in the map lump name (as is set in the map options window). This is for use with the -warp parameter.
%AP	All resource files, except the first IWAD, with full paths included. The resources are separated by spaces and when this placeholder is enclosed within quotes (**"%AP"**) then the quotes are repeated for every resource.
%S	Skill number at which to test.
%NM	This is either **-nomonsters** when you choose to test without monsters, or nothing at all.

When the checkbox **Use short path and file names (MSDOS 8.3 format)** is checked, all the above placeholders that output filenames and/or paths will use the short version that is compatible with MSDOS.

Textures

You can group your textures into categories called Texture Sets. There are some fixed Texture Sets that appear in the browsers automatically (such as the "All" set and the sets for each resource) and you can create custom Texture Sets here. You can also copy Texture Sets from one game configuration to another with the **Copy** and **Paste** buttons. With the **Add Default Sets** button you can add the default Texture Sets that are provided with the game configuration (if any). For more information about creating your own Texture Sets, see the Texture Set window.

Modes

Here you can choose what editing modes you can use when editing with the selected game configuration. This is useful when plugins are installed that replace certain editing modes or plugins that add editing modes for a specific sourceport only. Note that some editing modes may require other editing modes from the same plugin to work together. When plugins are installed, you should consult the provided documentation to see what changes should be made here. Only recommended for advanced users.

Grid Setup Window

On this window you can configure the grid and background settings. Click the grid icon in the statusbar or select Grid Setup from the Edit menu to access this window. The grid helps aligning your map elements and the 64 mappixels grid is important to Doom as it indicates how flats are aligned. You can change the size of the grid you want to work with at the top of this window. Note that the grid can also easily be changed with the actions **Grid Increase** and **Grid Decrease**.

The background is useful when drawing over an image to copy its shape. Check the **Show background image** checkbox and select a texture, flat or file to be used as the background image. You can also change the offset and the scale.

Image Browser Window

Use this browser to look for textures or flats. On the left you have the list with your Texture Sets. The list also contains a set named "All" which, obviously, contains all textures or flats. A Texture Set for each loaded resource has also been added. This is useful if you know in which resource the image resides. Once you have chosen your Texture Set, you can use the text field at the bottom to enter the texture name. You only have to enter the name partly and the list of images will adjust to show only the images that match with the entered name. Select an image to view its dimensions at the bottom. When the focus is on the images list, you can press the **Tab** key to jump between the same image in the "Used textures" and "Available textures" areas. Doubleclick the image or click **OK** to make your selection or **Cancel** to close the list.

To make your own Texture Sets, see the Textures tab on the Game Configurations Window.

Linedef Properties Window

With this dialog window you can edit all linedef properties. When a selection of multiple linedefs is made, some fields may appear grayed or empty in case they are different for some of the selected elements. Setting a value in grayed or empty field will apply this to the entire selection.

Properties

On this tab you will find all the general settings for the linedef. These setting apply to both sides of the linedef, unless specified otherwise.

The **Settings** area provides options for the line such as *Impassable* which blocks players and monsters from crossing the line, and *Doublesided* which is automatically set or unset by Doom Builder to indicate if the line has one or two sidedefs. Depending on the game configuration, this area can feature many more options for the behaviour of the linedef.

Below that is the **Action** area, which allows you to set a special action that is executed on a specific trigger method such as a switch or proximity trigger. In Hexen format, the action can take some arguments and the trigger method can be specified independently from the action. In UDMF format, the trigger method can be any combination of methods.

In Doom and UDMF formats, the line can also be tagged with a number that identifies the linedef. This number can then be used by Thing actions or scripts to perform actions on the linedef.

Sidedefs

Here you can change the properties for each side of the line. If you wish to add or remove a sidedef on the linedef, just check or uncheck the **Front Side** or **Back Side** box. In the **Sector Index** field you can type the sector index that you which this sidedef to be bound to. Note that you generally do not have to change this as Doom Builder will make valid sectors for you. Changing the Sector Index is only recommended for advanced users. On the right are the textures that are on the sidedef. Left-click on the box to browse for a texture using the Image Browser, right-click on the box to clear the texture (making it a single dash). If you know the texture name, you can also type the name directly into the text field below the texture box.

In UDMF format, you can click the **Custom Fields** buttons to change the custom fields for the sidedefs. See Custom Fields Editor for more information.

Custom

This tab allows you to edit the custom fields on the linedef. Only available in UDMF format. See Custom Fields Editor for more information.

Map Options Window

This dialog window allows you to change some options that apply to the entire map and how it is loaded. You can access it from the Edit menu, or with the default key F2. At the top you can choose the **Game Configuration** that you wish to use when editing this map. Please note that changing to a configuration of a different file format may cause a loss of information in your map, possibly breaking it. Below the Game Configuration field is the **Level Name** field. Here you can enter the lump name that is used to store your map in the WAD file. An example level name for a Doom 1 map could be E2M3 which indicates the map is or episode 2, level 3. An example for a Doom 2 map is MAP03 which is level 3.

Resources

The Resources area allows you to specify which resources are loaded when editing this map. The **Strictly load patches between P_START and P_END only for this file** checkbox applies to the wad file you are editing (where your map is in). When this is checked, Doom Builder will resolve texture patch conflicts by strictly loading patch lumps only from between the P_START and P_END lumps as it was intended by id Software, but most sourceports do not require this (as result, older maps have texture patch lumps that are not in between P_START and P_END).

In the list of resources you can see all the resources that will be loaded. Doom Builder shows the items from the game configuration in gray at the top of the list, because those are always loaded first. You can change the resources loaded according to the game configuration in the Game Configurations Window. You can drag the resource items (except the ones from the game configuration) to change their order. See also the Resource Options Window.

Open Map Window

When opening a map from a WAD file, this is the dialog that lets you choose what level to open and what options to use. At the top you can choose the **Game Configuration** that you wish to use when editing this map. When you select the game configuration you need, Doom Builder will show you a list of maps from the WAD file that can be opened with the selected configuration. You must select a map from the list to open.

Resources

The Resources area allows you to specify which resources are loaded when editing this map. The **Strictly load patches between P_START and P_END only for this file** checkbox applies to the wad file you are editing (where your map is in). When this is checked, Doom Builder will resolve texture patch conflicts by strictly loading patch lumps only from between the P_START and P_END lumps as it was intended by id Software, but most sourceports do not require this (as result, older maps have texture patch lumps that are not in between P_START and P_END).

In the list of resources you can see all the resources that will be loaded. Doom Builder shows the items from the game configuration in gray at the top of the list, because those are always loaded first. You can change the resources loaded according to the game configuration in the Game Configuration Window. You can drag the resource items (except the ones from the game configuration) to change their order. See also the Resource Options Window.

Preferences Window

The Preferences window provides options that apply to Doom Builder, regardless of the game configuration or map you choose to edit. You can access this window from the Tools menu, or with the default key F5. The options are categorized in different tabs;

Interface

This tab contains some interface related options.

Default view:

This sets the default view mode that you wish to use when opening or creating a new map.

Preview image size:

The size of all preview images in the editor and in the Image Browser. Smaller preview images consume less system memory.

Autoscroll speed:

During some actions, such as drawing and dragging, Doom Builder can automatically scroll the working area when you move your mouse towards the edges. Use this setting to set the scrolling speed to use. Drag the slider completely to the left to turn off automatic scrolling.

Script Editor:

Here you can set the font and other options regarding the Script Editor. Check the box **Always on top of main window** if you want the script editor to float above the Main Window even when the script editor does not have the focus.

Visual Modes:

This area contains several settings for the Visual Mode. To improve performance, you may want to decrease the **View distance** setting.

Controls

This is where you configure your keyboard and mouse shortcut keys. On the left is a list of all Actions in Doom Builder with the current mouse or key combination shown next to them. Select an action to view a summary of this action on the right. Press any key combination to set to this action, or select a special mouse combination from the dropdown box. Below that, you can see which other actions share the same key combination so that you can verify any conflicts it may have. Note that actions used in different editing modes can safely share the same key combination.

Appearance

On this tab you can change some settings related to visual aesthetics. On the top and the left of this tab you can change colors that are used in the Script Editor and Working Area.

Passable lines transparency:

This controls how much translucent the passable lines are. 0% is completely opaque, which is the same as impassable lines. Passable or impassable can be set with a linedef setting, but is automatically set by the editor for singlesided lines and removed for doublesided lines.

Textures and Flats brightness:

With this setting you can increase the brightness of the textures and flats. This is useful when viewing the texture against a white background, or if you are using a dark monitor.

Black background in image browser:

Turn this on to create a black background in the Image Browser.

Bilinear filtering in classic modes:

This enables bilinear texture filtering for all classic Editing Modes (2D). Bilinear texture filtering removes the large square blocks when zooming in, but can also make it look blurry.

Bilinear filtering in visual modes:

This enables bilinear texture filtering for all visual Editing Modes (3D). Bilinear texture filtering removes the large square blocks when zooming in, but can also make it look blurry.

High quality display rendering:

This improves the display with the use of Pixel Shader Model 2.0. Also adds anti-aliasing to the classic Editing Modes (2D). Performance may be improved by turning off this setting (this also gives a more oldschool Doom Builder 1.68 look to the working area). When Pixel Shader Model 2.0 support is not available, Doom Builder automatically turns off this setting and disables this option.

Square things:

Things in Doom use a square shape for collision detection and you may want to turn this option on to get a better view on the bounding boxes of things in the classic Editing Modes (2D).

Resource Options Window

Use this dialog window to browse for a resource and change any options on how to use the resource. Doom Builder supports 3 types of resources;

WAD File

This is the most common resource known with Doom. You can find more information about the WAD format on the Doom Wiki. When the **Strictly load patches between P_START and P_END only for this file** checkbox is checked, Doom Builder will resolve texture patch conflicts by strictly loading patch lumps only from between the P_START and P_END lumps in this WAD file as it was intended by id Software, but most sourceports do not require this (as result, older maps have texture patch lumps that are not in between P_START and P_END).

Directory

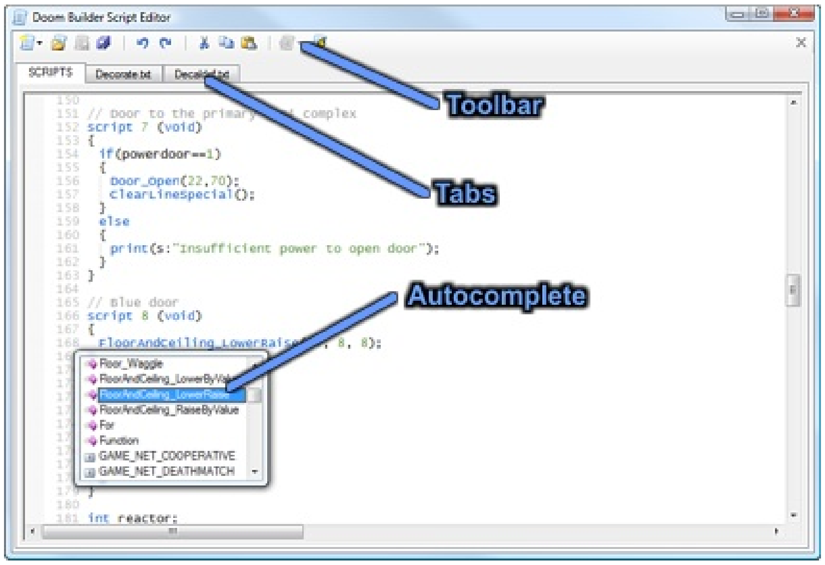
Doom Builder can load resources from a directory in different ways. The directory is expected to use the PK3 directory structure (for more information see the ZDoom Wiki) but you can use it to simply load images from the directory root as textures or flats by turning on the options **Load images in directory root as textures/flats**.

PK3 File

The PK3 file is a zipped directory structure and can be used as WAD file replacement by several sourceports. For more information see the ZDoom Wiki.

Script Editor Window

The script editor is a powerful text editor with syntax highlighting and autocomplete. It allows you to open and edit multiple documents at the same time, even if they do not reside in your map.



```

// Door to the primary      complex
script 7 (void)
{
  if(powerdoor==1)
  {
    Door_Open(22,70);
    clearLinespecial();
  }
  else
  {
    print(s:"Insufficient power to open door");
  }
}

// Blue door
script 8 (void)
{
  FloorAndCeiling_LowerRaise     , 8, 8);

}

int reactor;
```

Each opened script has its own tab page. When there are text lumps in the map format (such as the SCRIPTS lump in Hexen formats), they will be opened automatically and cannot be closed.

Shortcut Keys

CTRL+O	Browse for a script file to open.
CTRL+S	Save the current script to file. This does not work for internal script lumps, save the map to save the internal script lumps.
CTRL+Space	Autocompletes the current word. If there is more than one possibility, it will pop up a list of keywords to choose from.
CTRL+F	Opens the Find and Replace dialog window.
F3	Finds the next occurence of the most recent search with the Find and Replace dialog window.
F2	If supported for the type of script you are editing, this opens a website with information about the current keyword that your cursor is at.

Sector Properties Window

On this dialog you can change all the sector properties. When a selection of multiple sectors is made, some fields may appear grayed or empty in case they are different for some of the selected elements. Setting a value in grayed or empty field will apply this to the entire selection.

Properties

You can manually set the **Ceiling height** and **Floor height** at the top of this window. On the right are the flats (textures) that are on the floor and ceiling. Left-click on the box to browse for a flat using the Image Browser. If you know the flat name, you can also type the name directly into the text field below the flat box.

In the **Effects** area you can set a special effect for the sector and set the **Brightness** level to use.

Below that you can enter a tag number to identify the sector. This number can then be used by Linedef actions or scripts to perform actions on the sector.

Custom

This tab allows you to edit the custom fields on the sector. Only available in UDMF format. See Custom Fields Editor for more information.

Texture Set Window

This dialog window lets you make your own Texture Set. On the top of this window you can enter the name of the Texture Set, this is how it will be displayed in the Game Configurations Window and the Image Browser Window. It is recommended that you make your Texture Sets while the map you are editing is open in Doom Builder, so that you can see the results immediately.

Filters

In this area you set up what you want to include in this Texture Set. You can use wildcards to include more than one texture at once. Use the question mark **?** to indicate exactly one character (no matter which character) and use the asterisk ***** to indicate zero or more characters (no matter which characters). Each Texture Set has its own filters, so it is possible for Texture Sets to overlap and include textures that are also in other Texture Sets.

Results

If you have a map open in Doom Builder while you are setting up your Texture Sets, you can instantly see the result in the Results area. By clicking the buttons **Show Matches** and **Show Not Matching** you can see which textures are included and which are not. When viewing the texture that are **not** included, you can double-click the texture image to add it to the filters list and include it in the Texture Set.

Thing Properties Window

This dialog allows you to edit all the thing properties. When a selection of multiple things is made, some fields may appear grayed or empty in case they are different for some of the selected elements. Setting a value in grayed or empty field will apply this to the entire selection.

Properties

On the left of this tab you can select the **Thing Type**. Either select it from the list or enter the type number in the field below the list. If it is a thing type that is in the Game Configuration, the information known about that thing type is displayed as well.

On the right you can change the **Settings** for the thing. The settings available depend on the Game Configuration you are using. Below that you can change the **Angle**. In Hexen and UDMF formats, you can also specify the **Z height** of the thing. The Z height is usually relative to the sector floor, but for some things the Z height is absolute (refer to the sourceport documentation).

Action

This tab is only available in Hexen and UDMF formats. You can set the thing **Tag** that is used to refer to this thing. Below that is the **Action** area, which allows you to set a special action that is executed by the thing (for monsters this is usually when it dies). Please note that some things use the action arguments for their own properties and that setting an action on those things may give an unexpected result (refer to the sourceport documentation).

Custom

This tab allows you to edit the custom fields on the thing. Only available in UDMF format. See Custom Fields Editor for more information.

Things Filters Window

With the Things Filters dialog window, you can set up your own Things filters that you can use to show only relevant things in the editor. You can access this dialog from the View menu. Select a filter on the left to edit it, or click on **New Filter** to create a new one. You can remove the selected filter by clicking on the **Delete Selected** button. On the right is the **Filter Settings** are where you can edit the selected filter.

Enter the name of your filter in the **Name** field. If you want to show only Things from a specific category, select the category in the **Filter by category** box. Below that you can select the settings by which you want to filter the Things (**Filter by settings)**. A grayed/filled checkbox means it is not used for the filter. If the checkbox is empty, it means that the setting may not be set on the Things to pass the filter. If the checkbox is checked, it means that the setting must be set on the Things to pass the filter.

Things Filters are stored separately for each Game Configuration.

Vertex Properties Window

This dialog allows you to edit the exact vertex coordinates. When a selection of multiple vertices is made, some fields may appear grayed or empty in case they are different for some of the selected elements. Setting a value in grayed or empty field will apply this to the entire selection.

Custom

This tab allows you to edit the custom fields on the vertex. Only available in UDMF format. See Custom Fields Editor for more information.

About Editing Modes

With Doom Builder, you are always editing in a specific mode which depends on what you want to do and allows you to use the same mouse buttons and key controls for different purposes. You will find yourself switching editing modes all the time, so it is a good thing to remember which keys you have assigned to these modes.

Brightness Mode

With this mode you can easily change the brightness levels in your map. Make a selection of sectors with **LMB**. Then hold **RMB** and drag up or down to increase or decrease the brightness levels of the selected sectors. The brightness changes to the nearest supported brightness level (according to the chosen game configuration), but you can hold Shift to override this behavior.

The order in which you select the sectors is important to the **Make Brightness Gradient** feature as it creates a gradient from the first selected sector to the last selected sector (the first and last are not modified). When you select the sectors, the order will be displayed with numbers in the sectors.

Curve Linedefs Mode

This mode shows a small dialog window that you can use to create a curve from a linedef or multiple linedefs. The linedefs will be split in a number of smaller linedefs that make up the curve. Although this mode shows a dialog window, you can still use the main interface and zoom/move your view around the map.

The curve linedefs editing mode is one of the classic (2D) editing modes. This mode is volatile, which means that this mode returns to the previous stable mode when the map is saved or closed, either accepting or discarding your preview changes. You can access this mode through the Linedefs Mode.

Default Controls

Enter	Accept and apply the changes, and return to the previous mode.
Escape	Discard the changes and return to the previous mode.

Draw Geometry Mode

This editing mode allows you to draw your geometry. Draw lines by clicking with the left mouse button (**LMB**) where you want your vertices. Depending on the toolbar buttons, the vertex you are drawing next will snap to the grid and/or to nearby geometry. You can change this behavior by holding down Shift and/or Control. Your drawing ends when you draw onto the first drawn vertex (making a closed polygon) or when you press your right mouse button (**RMB**).

Contrary to drawing in older editors, it is not required to draw in any specific way, such as drawing clockwise or drawing a complete polygon. You also do not need to traverse your existing lines when drawing a sector adjacent to existing geometry. The editor will solve any lines your drawing crosses, split sector you draw across and creates new sectors where needed.

This editing mode is one of the classic (2D) editing modes. This mode is volatile, which means that this mode returns to the previous stable mode when the map is saved or closed, either accepting or discarding your preview changes. You can access this mode by pressing **Control+D** in any classic mode.

Default Controls

Control+D	Starts this drawing mode (available from any classic mode).
Enter	Accept and apply the changes, and return to the previous mode.
Escape	Discard the changes and return to the previous mode.
Backspace	Removes the last drawn vertex.
LMB	Draws a new line by placing a vertex at the mouse cursor. Use in combination with Shift and/or Control to change the snapping behavior.
LMB	Ends and applies the drawing, and returns to the previous editing mode.

Edit Selection Mode

With this editing mode you can perform some more advanced operations on a selection of geometry or things. This editing mode is one of the classic (2D) editing modes. This mode is volatile, which means that this mode returns to the previous stable mode when the map is saved or closed, either accepting or discarding your preview changes. You can access this mode by making a selection or highlight and pressing **E** in most classic modes. Clicking on the map outside the selection will return to the previous mode, applying your changes.

Functions

Drag selection

You can drag the selection by holding **LMB** on the selection rectangle to grab it and move it around. When grabbing the selection, the nearby vertex is highlighted and will be used to snap to other geometry while dragging.

Flip Horizontally

Accessible from the toolbar or by shortcut key, you can flip the entire selection horizontally. It is recommended to do this before rotating the selection.

Flip Vertically

Accessible from the toolbar or by shortcut key, you can flip the entire selection vertically. It is recommended to do this before rotating the selection.

Resize

Grab one of the handles on the edges of the selection rectangle using **LMB** and drag it to resize the selection. You can drag your mouse away from the selection and a ruler will appear that allows you to align the grabbed edge of the selection with existing geometry.

Rotate

Grab one of the handles on the corners of the selection rectangle using **LMB** and drag it to rotate the selection. The selection will normally only rotate by the nearest 45 degrees. You can hold Shift to rotate it freely.

Default Controls

Enter	Accept and apply the changes, and return to the previous mode.
Escape	Discard the changes and return to the previous mode.

Find & Replace Mode

Looking for something? This is the mode you want to use to quickly find specific elements in your map. This mode is accessible from any classic mode using **F3** and shows a dialog window that you can use to find items. Select the type of search you want to perform at the top of the dialog window. Then enter the value you are looking for. Depending on the type of search, you can click the browse button to select from a browser. Check the **Replace** checkbox if you want to replace the value of the found elements with another value. Click the Find / Replace button to perform the search.

The results will be displayed in a list that opens. Click on one of the results to zoom in on that particular element. You can also focus the main window and scroll or zoom the map to see the selected elements. Click the **Edit Selection** button to edit the selected elements.

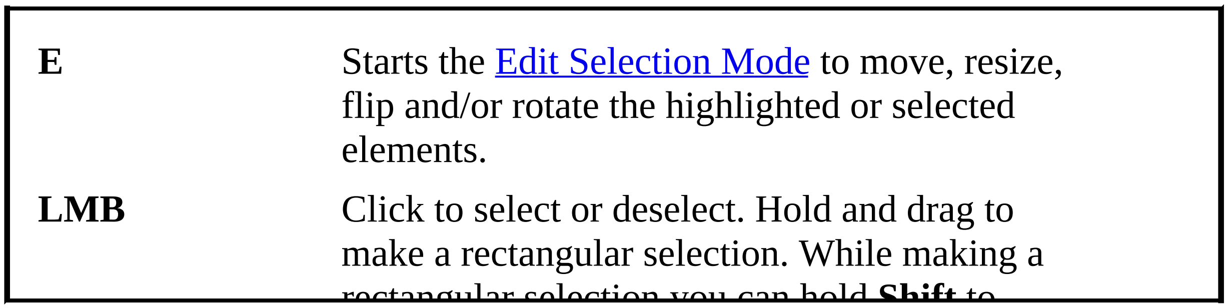
Linedefs Mode

With this mode you can edit the linedefs in your map. Every linedef has a front (right) and back (left) side which make up the boundaries of your sectors. You can edit the sidedef properties through the linedefs in this mode. Drag the linedefs to move them and snap them to the grid and other geometry. Hold Shift and/or Control to disable snapping to the grid and geometry.

The linedefs editing mode is one of the classic (2D) editing modes.

Default Controls

L	Switches from any other classic editing mode to this mode.
Insert	Starts the Draw Geometry Mode.
Delete	Deletes the selected linedefs. Note that this may break your sectors.
F	Flips the linedef and changes the sidedefs so that they remain in their correct position.
Shift+F	Flips the sidedefs to the other side of the linedef.
Shift+S	Splits the linedef with a new vertex on the linedef, nearest to your mouse position.
Shift+1	Keeps only the single-sided linedefs selected.
Shift+2	Keeps only the double-sided linedefs selected.
Shift+C	Starts the Curve Linedefs Mode to make curves with the selected or highlighted linedefs.



E	Starts the Edit Selection Mode to move, resize, flip and/or rotate the highlighted or selected elements.
LMB	Click to select or deselect. Hold and drag to make a rectangular selection. While making a rectangular selection you can hold **Shift** to

Make Sectors Mode

With this mode you can create new sectors from existing geometry only. This is useful to fix broken sectors, split the islands from a single sector into multiple sectors, or create sectors from enclosed void areas. When moving the mouse over the map, the potential sector is highlighted. Use **LMB** to create the new sector.

The Make Sectors editing mode is one of the classic (2D) editing modes.

Default Controls

M	Switches from any other classic editing mode to this mode.
LMB	Click to create a new, closed sector.
RMB	Click to create a new, closed sector and edit its properties.

Map Analysis Mode

This mode helps finding problems in your map. It shows a small dialog window with the checks you wish to perform. Press **F4** in any classic mode to switch to this mode. Select the checks to perform and click the **Start Analysis** button. The problems that are found are displayed in a list. Click on of the problems to zoom in on the subject and show a description of the problem below the list. For some problems, buttons appear below the description that you can use to quickly fix the problem.

Sectors Mode

In this mode you can edit the sectors in your map. Every sector is formed by the sidedefs around the sector (which are the sides of the linedefs). Drag the sectors to move them and snap them to the grid and other geometry. Hold Shift and/or Control to disable snapping to the grid and geometry. You can also create a stairs or gradient brightness by using the buttons on the toolbar.

The order in which you select the sectors is important to some of the functions of the editing mode and, as long as you stay in an editing mode that keeps sectors selected, the order will stay the same. When you select the sectors, the order will be displayed with numbers in the sectors.

The sectors editing mode is one of the classic (2D) editing modes.

Default Controls

S	Switches from any other classic editing mode to this mode.
Insert	Starts the Draw Geometry Mode.
Delete	Deletes the selected sectors and associated sidedefs. When deleting a sector that is enclosed inside a larger sector, this will create a hole (or 'pillar') inside that sector. If you wish to remove the sector without creating a void area, use the Linedefs Mode to delete the lines instead.
J	Joins to selected sector to become one sector. The properties of the first selected sector will be retained.
Shift+J	Merge the selected sectors to become one sector. This is the same as joining, with the addition that lines in between the selected sectors are automatically removed.
Shift+D	Create a door from the selected or highlighted sectors. This will show a dialog which allows you to choose textures and flats for the door.
G	Make gradient brightness levels from first selected sector to last selected sector.
Shift+ScrollDown	Lowers the ceiling of the selected or highlighted sectors by 8 mappixels.
Shift+ScrollUp	Raises the ceiling of the selected or highlighted sectors by 8 mappixels.
Shift+G	Make gradient ceiling heights from first selected sector to last selected sector.
	Lowers the floors of the selected or

Control+ScrollDown	highlighted sectors by 8 mappixels.
Control+ScrollUp	Raises the floors of the selected or highlighted sectors by 8 mappixels.
Control+G	Make gradient floor heights (stairs) from first selected sector to last selected sector.

E	Starts the Edit Selection Mode to move, resize, flip and/or rotate the highlighted or selected elements.
LMB	Click to select or deselect. Hold and drag to make a rectangular selection. While making a rectangular selection you can hold **Shift** to

Things Mode

With this editing mode you edit the things in your map. Drag the things to move them and snap them to the grid. Hold Shift to disable snapping to the grid. You can use the Things Filter in the toolbar to show only things with specific properties (such as multiplayer things or things that only appear at a hard skill level).

The things editing mode is one of the classic (2D) editing modes.

Default Controls

T	Switches from any other classic editing mode to this mode.
Insert	Inserts a new thing.
Delete	Deletes the selected or highlighted things.
E	Starts the Edit Selection Mode to move, resize, flip and/or rotate the highlighted or selected elements.
LMB	Click to select or deselect. Hold and drag to make a rectangular selection. While making a rectangular selection you can hold **Shift** to

Vertices Mode

This editing mode allows you to edit the vertices in your map. You can insert new vertices that automatically split linedefs and deleting a vertex that only has two linedefs attached will reconnect the linedefs properly. You can drag the vertices to move them and snap them to the grid and geometry. Hold Shift and/or Control to disable snapping to the grid and geometry.

The vertices editing mode is one of the classic (2D) editing modes.

Default Controls

V	Switches from any other classic editing mode to this mode.
Insert	Inserts a new vertex at the mouse coordinates. When near a linedef, the linedef will be split with this vertex.
Delete	Deletes the selected vertices. Note that this may also remove attached linedefs and break your sectors.
E	Starts the Edit Selection Mode to move, resize, flip and/or rotate the highlighted or selected elements.
LMB	Click to select or deselect. Hold and drag to make a rectangular selection. While making a rectangular selection you can hold **Shift** to

Visual Mode

The Visual Mode is different from the other modes. In this mode you can walk and fly through your map in 3D and see it as it would look like in game. With the crosshair you can aim at objects (floors, ceilings, walls and things) and edit them instantly. You can also select objects and perform actions on all objects together. Note that not all actions apply to an entire selection.

Default Controls

W	Switches from any classic editing mode to this mode and back to the previous mode.
LMB	Select the targeted object. For walls, hold down this button to drag the texture offsets.
RMB	Edit the targeted object's properties.
MMB	Pastes the copied texture image or flat image onto the target or selection.
Control+RMB	Opens the Image Browser to select a new texture or flat for the targeted or selected object.
Shift+MMB	Flood-fills with the copied texture image or flat image onto the target. This fills all adjacent textures or flats that are identical to the original. Note that this action is only applied to the targeted object and does not work for a selection.
C	Clears the selection.
E	Move forward. Hold Shift to move faster.
D	Move backward. Hold Shift to move faster.
S	Strafe left. Hold Shift to move faster.
F	Strafe right. Hold Shift to move faster.
G	Toggle gravity on/off.
B	Toggle full-brightness on/off.
L	Toggle lower unpegged linedef flag on/off.
U	Toggle upper unpegged linedef flag on/off.
	Toggle things on/off/boxed. When things are set to boxed, you will see the

T	bounding box of things around them in the color of their category.
A	Auto-aligns the neighbouring textures horizontally, until a wall is encountered that has different textures. Note that this action is only applied to the targeted object and does not work for a selection.
Shift+A	Auto-aligns the neighbouring textures vertically, until a wall is encountered that has different textures. The vertical alignment takes the ceiling height differences between sidedefs into account. Note that this action is only applied to the targeted object and does not work for a selection.
Shift+C	Copies the targeted texture offsets for pasting. Note that this works only for the targeted object, the selection is ignored.
Control+C	Copies the targeted texture image or flat image for pasting. Note that this works only for the targeted object, the selection is ignored.
Control+Shift+C	Copies the targeted object properties for pasting. Note that this works only for the targeted object, the selection is ignored.
Shift+V	Pastes the copied texture offsets onto the target or selection.
Control+Shift+V	Pastes the copied object properties onto the target or selection.
ScrollUp	Raises the targeted or selected object by 8 mappixels.
ScrollDown	Lowers the targeted or selected object by 8 mappixels.
Shift+ScrollUp	Raises the targeted or selected object by 1 mappixel.
	Lowers the targeted or selected object

Shift+ScrollDown	by 1 mappixel.
Control+ScrollUp	Increases the brightness level of the targeted or selected object by 8.
Control+ScrollDown	Decreases the brightness level of the targeted or selected object by 8.
Up	Moves the targeted or selected texture up by 1 pixel.
Down	Moves the targeted or selected texture down by 1 pixel.
Left	Moves the targeted or selected texture left by 1 pixel.
Right	Moves the targeted or selected texture right by 1 pixel.
Shift+Up	Moves the targeted or selected texture up by 8 pixels.
Shift+Down	Moves the targeted or selected texture down by 8 pixels.
Shift+Left	Moves the targeted or selected texture left by 8 pixels.
Shift+Right	Moves the targeted or selected texture right by 8 pixels.
Shift+R	Resets the texture offsets to 0, 0 on the targeted or selected object.

About Configurations

Doom Builder is a very flexible editor that can be customized for your own sourceport or mapping project. Most of the settings that are different among the sourceports are in configuration files. You can find these configurations in the subdirectories of your Doom Builder program directory. This part of the Reference Manual helps you writing your own configurations, but is only recommended for advanced users.

You can safely create your own configurations for your mapping project or sourceport. If there are any errors in your configuration, Doom Builder will show these in the Errors & Warnings dialog. Doom Builder is installed with several configurations which can be great examples to see how these things work. It is recommended to always create new configurations in separate files. Do not modify the files that are installed by Doom Builder, as these will be overwritten when a new version is installed.

When you made changes to the configurations in the Doom Builder subdirectories, you must restart Doom Builder to reload the configurations with your changes.

Configuration Syntax

All configurations used with Doom Builder follow a specific, structured syntax. The configurations are all in plain text format and can be edited with any plain text editor. Every setting in a configuration is in the following form:

```
settingname = value;
```

Here are a few examples:

```
doublesidedflag = 4;
defaulttexturescale = 1.0f;
scaledtextureoffsets = true;
gamename = "Doom";
```

There are a few rules to the setting names and values:

- The setting name may not contain any spaces, tabs, newlines, dots or (back)slashes.
- The setting name must be unique within the configuration, or within the structure it is in.
- A decimal value must contain a dot and must end with '**f**'
- Boolean settings may use the keywords **true** and **false**.
- Strings (texts) must begin with a doublequote (**"**) and end with a doublequote. To include a doublequote in the string, prefix it with a backslash (**\\"**). To use a backslash in a string, also prefix it with a backslash.

Some settings do not require a value and their precense or absense alone is enough. Such a case would look like this:

```
enablelighting;
```

C-style comments can be inserted by using double slashes (***//***) for singleline comments and ***/**** and ****/*** for block comments. These comments will be completely ignored by Doom Builder. You can hide writings about your wildest dreams in configuration files!

Structures

Configurations can also contain **structures**. Think of them as a collection of settings that have their own name space. The structure begins with a name and opens with an opening bracket (**{**). At the end of the structure it closes with a closing bracktet (**}**). It is common to ident the settings inside the structure with a single tab to make it easier to see that those settings belong in a structure. An example of a structure with settings:

```
winningnumbers
{
	car = 55;
	washmachine = 40;
	tools = 30;
}
```

Structures are often used by Doom Builder in cases where multiple collections of settings can be found. The name of a structure can also be just a number, so that a structure can describe information about a specific index number. Here is an example of such a case:

```
things
{
	1
	{
		name = "Player 1 start";
		color = "green";
	}

	2
```

```
        {
                name = "Player 2 start";
                color = "green";
        }

        138
        {
                name = "Hideous monster";
                color = "red";
        }
}
```

Please note that in some structures, the order of the settings and structures is important.

Including

Some configurations (such as the Game Configurations) can become very large and complex, while some share the same values in many settings. For example, Eternity is a game engine that inherits features from Doom and Boom and adds on top of that. This is where including pieces from other configuration files becomes interesting. With the **include** function, we can insert features from Doom, then insert (and possible override) features from Boom and then add the Eternity ones. This saves us rewriting all the Doom and Boom features that already exists in other configurations.

The include function takes two arguments. The first (mandatory) is the filename (path relative to the current file) and the second (optional) is the name of the structure to include. If the second argument is not given, the entire file will be included. Here is an example that shows how such an include functions look like:

```
include("commonsettings.cfg");
include("extras.cfg", "skills");
```

Below are a few examples that show what the results are when including settings that override settings with the same names among other things. For these examples we will be including a file named "extras.cfg", which contains the following settings:

```
maxtexturenamelength = 8;
skyflatname = "F_SKY1";
```

```
options
{
	1 = "Low";
	2 = "Medium";
	3 = "High";
}
```

This example shows how the include function includes an entire file. This is the most basic include:

```
thingflags
{
	1 = "Easy";
	2 = "Medium";
	4 = "Hard";
}

include("extras.cfg");
```

Result:

```
thingflags
{
	1 = "Easy";
	2 = "Medium";
	4 = "Hard";
}

maxtexturenamelength = 8;
skyflatname = "F_SKY1";

options
{
	1 = "Low";
	2 = "Medium";
```

```
        3 = "High";
}
```

The following example shows how a single structure from extras.cfg is included. Notice how only the contents of the structure are included and not the structure container itsself!

```
thingflags
{
        1 = "Easy";
        2 = "Medium";
        4 = "Hard";
}

include("extras.cfg", "options");
```

Result:

```
thingflags
{
        1 = "Easy";
        2 = "Medium";
        4 = "Hard";
}

1 = "Low";
2 = "Medium";
3 = "High";
```

If we want our included settings in a structure, we have to put the include function in the structure where we want our settings included. See the following example:

```
thingflags
```

```
{
	1 = "Easy";
	2 = "Medium";
	4 = "Hard";
}

options
{
	0 = "None";

	include("extras.cfg", "options");

	4 = "Ultra";
}
```

Result:

```
thingflags
{
	1 = "Easy";
	2 = "Medium";
	4 = "Hard";
}

options
{
	0 = "None";

	1 = "Low";
	2 = "Medium";
	3 = "High";

	4 = "Ultra";
}
```

The following example demonstrates how values can be overridden by using the same same setting name. It also shows that this does NOT change the order of the items. The first time an item is defined (either by including or because it is written) is where its position will be. See this example:

```
thingflags
{
	1 = "Easy";
	2 = "Medium";
	4 = "Hard";
}

options
{
	0 = "None";

	include("extras.cfg", "options");

	2 = "Average";
	4 = "Ultra";
}
```

Result:

```
thingflags
{
	1 = "Easy";
	2 = "Medium";
	4 = "Hard";
}

options
{
	0 = "None";
```

```
        1 = "Low";
        2 = "Average";
        3 = "High";

        4 = "Ultra";
}
```

Notice how the definition of setting "2" ("Average") does not move the already defined "2" ("Medium"), but instead only changes its value to "Average".

About Compiler Configurations

B

About Game Configurations

These configurations are the largest and most complex ones. A Game Configuration contains all the settings that determine Doom Builder's behavior when editing your map and optional features that are specific to certain map formats can be enabled or disabled here. A Game Configuration also keeps all information about hardcoded actions and things in the game, which Doom Builder cannot find in the WAD files. You can find the Game Configurations in "Configurations" directory, which is a subdirectory of your Doom Builder's program directory. Subdirectories are not searched by Doom Builder for configurations, but you can use them for include files.

Many settings in Doom Builder's user interface as bound to the available Game Configurations. This is because the user often wants to use different resources and compilers for a different project. When Doom Builder starts up and detects a new Game Configuration file, it will automatically create default settings for that configuration. The following default settings are read from the Game Configuration (only when first detected by Doom Builder): testparameters, testshortpaths, defaultsavecompiler, defaulttestcompiler, thingsfilters and texturesets.

Game Configuration - Basic Settings

type (string)
This indicates the type of configuration to prevent accedential use of a different configuration. Must always be the string "Doom Builder 2 Game Configuration".

game (string)
The name that is displayed in Doom Buider for your Game Configuration.

engine (string)
Game engine/sourceport name. This is used as the UDMF namespace for UDMF map format interface. It currently has no other function.

defaultlumpname (string)
Default lump name suggested when creating a new map and selecting this configuration.

testparameters (string)
Default parameters used to launch the test game engine. See Game Configurations window for the available parameter placeholders.

testshortpaths (boolean)
Set to **true** to use MSDOS 8.3 format paths and filenames by default. Default is **false**. The user can still change this in the Game Configurations window.

defaultsavecompiler (string)
Name of the Nodebuilder Compiler Configuration structure to use as the default settings for the compiler that is used when saving the map. The user can still change this in the Game Configurations window.

defaulttestcompiler (string)
Name of the Nodebuilder Compiler Configuration structure to use as the default settings for the compiler that is used when testing the map. The user can still change this in the Game Configurations window.

skills (structure)
This defines the skill options the user has available with this game engine/project. The settings in this structure are expected to be numbers with string values (the descriptive name for the skill level).

Example:

```
skills
{
        1 = "I'm too young to die";
        2 = "Hey, not too rough";
        3 = "Hurt me plenty";
        4 = "Ultra-Violence";
        5 = "Nightmare!";
}
```

linetagindicatesectors (boolean)
When **true**, Doom Builder will highlight sectors associated with the

same tag number when a line is highlighted. This is only really useful for Doom format maps, because Hexen format and UDMF format has no single tag on linedefs (in those formats, the arguments of the linedef's action can be tags, which also works to highlight sectors). The default is **false**.

soundlinedefflag (integer or string)

This lets Doom Builder know the linedef flag that indicates where sound should be blocked. Doom Builder uses this to give the line a special color and plugins can use this information to perform operations related to blocking sound lines. For map formats that use numeric flags (Doom and Hexen) this must be an integer specifying the flag value of the Block Sound flag. For map formats that use named flags (UDMF), this must be a string indicating the name of the Block Sound flag.

singlesidedflag (integer or string)

This lets Doom Builder know the linedef flag that indicates a line with only one side. Doom Builder will set this flag value on a linedef when it changes a line to become single sided and removes the flag from a linedef when it becomes double sided. Plugins can also use this information to perform operations on linedefs. For map formats that use numeric flags (Doom and Hexen) this must be an integer flag value. For map formats that use named flags (UDMF), this must be a string indicating the name of the flag.

doublesidedflag (integer or string)

This lets Doom Builder know the linedef flag that indicates a line with two sides. Doom Builder will set this flag value on a linedef when it changes a line to become double sided and removes the flag from a linedef when it becomes single sided. Plugins can also use this information to perform operations on linedefs. For map formats that use numeric flags (Doom and Hexen) this must be an integer flag value. For map formats that use named flags (UDMF), this must be a string indicating the name of the flag.

impassableflag (integer or string)

This lets Doom Builder know the linedef flag that indicates a line which blocks players and monsters. Doom Builder uses this to give the line a special color and plugins can use this information to perform operations related to blocking sound lines. For map formats that use numeric flags (Doom and Hexen) this must be an integer specifying the flag value of the Impassable flag. For map formats that use named flags (UDMF), this must be a string indicating the name of the Impassable flag.

makedoortrack (string)

Name of a texture to use on the walls when making a door.

makedooraction (integer)

Linedef action number to put on the lines when making a door.

makedoorarg# (0 .. 4) (integer)

Arguments for the linedef action number to put on the lines when making a door.

doomlightlevels (boolean)
Set this to **false** to use linear lighting in Doom Builder. Normally Doom Builder uses a simulation of Doom's light levels. Default value is **true**.

start3dmode (integer)
Thing type number that Doom Builder will use to keep your Visual Mode camera position stored in the map. Doom Builder will place a single thing of this type in your map and move it along as you move in Visual Mode.

skyflatname (string)
Name of the flat that is interpreted as sky (meaning there is no ceiling). Doom Builder and plugins can use this information for various purposes.

maxtexturenamelength (integer)
Maximum length of texture names in characters. This is used by Doom Builder to limit the input fields in the user interface and to check the validity of texture names in resources. This does NOT determine the actual limitation on the texture names in the map file format. Default value is 8.

Game Configuration - Map Format Settings

formatinterface (string)

Interface class name in Doom Builder that is used to read and write the map data.

- For Doom map format, use "DoomMapSetIO". This is a map format that uses numeric flags.
- For Hexen map format, use "HexenMapSetIO". This is a map format that uses numeric flags.
- For UDMF map format, use "UniversalMapSetIO". This is a map format that uses named flags.

gamedetect (structure)

This is used to determine if the Game Configuration is suitable for the opened WAD file. None of the settings in this structure have any impact on the actual editing, limitations or storage of the map, but only assist the user when opening WAD files that have not been opened with Doom Builder before (when no .dbs file is available). The setting names in this structure are the lump names that must be checked for. The value of these settings (integers) indicates which rule applies to the lump name for a positive check. Valid values are:

- **1** - At least one of these lumps must exist.
- **2** - None of these lumps must exist.
- **3** - All of these lumps must exist.

Example:

```
gamedetect
{
	TEXTMAP = 2;
	ENDMAP = 2;
	EXTENDED = 2;
	BEHAVIOR = 3;
	E1M1 = 2;
	E1M2 = 2;
	E1M3 = 2;
	MAP01 = 1;
	MAP02 = 1;
	MAP03 = 1;
}
```

maplumpnames (structure)
This structure describes the lumps that make up the complete map, including scripts and nodes. It must also indicate what Doom Builder is supposed to do with these lumps and/or where the lumps come from. These lumps are normally written by the map format interface class, but some could be generated by the nodebuilder compiler or stored by Doom Builder directly. In this structure, there should be a structure for every map lump. The name of the structure should be the lump name, in uppercase. For the map header (which name differs) you can use the name **~MAP**. The following settings should be in the lump structure (unless the default value applies):

- **required** (boolean).
  True to indicate that this lump is a required lump for the map. Default is **false**.

- **blindcopy** (boolean).

Set this to **true** when Doom Builder should copy this lump along with the map without using it. Default is **false**.

- **nodebuild** (boolean).
  When set to **true**, this indicates that the nodebuilder compiler generates or modifies this lump. Default is **false**.

- **allowempty** (boolean).
  Set to **true** to allow the nodebuilder to leave this lump empty. Default is **false**.

- **script** (string).
  When this is set, it indicates that this lump can be edited with the script editor. The contents of this lump will be loaded in the script editor automatically. Set the value to the .cfg filename (without path) of a Script Configuration to use.

Example:

```
maplumpnames
{
	~MAP
	{
		required = true;
		blindcopy = true;
		nodebuild = false;
	}

	THINGS
	{
		required = true;
		nodebuild = true;
		allowempty = true;
	}

	LINEDEFS
	{
		required = true;
		nodebuild = true;
		allowempty = false;
	}

	SIDEDEFS
	{
		required = true;
		nodebuild = true;
		allowempty = false;
	}

	VERTEXES
	{
		required = true;
		nodebuild = true;
		allowempty = false;
	}

	SEGS
	{
		required = false;
		nodebuild = true;
		allowempty = false;
	}

	SSECTORS
	{
		required = false;
		nodebuild = true;
		allowempty = false;
	}
```

```
        NODES
        {
                required = false;
                nodebuild = true;
                allowempty = false;
        }

        SECTORS
        {
                required = true;
                nodebuild = true;
                allowempty = false;
        }

        REJECT
        {
                required = false;
                nodebuild = true;
                allowempty = false;
        }

        BLOCKMAP
        {
                required = false;
                nodebuild = true;
                allowempty = false;
        }

        SCRIPTS
        {
                required = false;
                nodebuild = false;
                script = "ZDoom_ACS.cfg";
        }
}
```

Game Configuration - Resource Settings

decorategames (string)

Fill this to the game names to support DECORATE actors from. Only the DECORATE actors who's game name is in this string will be loaded. If this setting is not set, DECORATE lumps are not loaded.

Example:

```
decorategames = "heretic raven";
```

mixtexturesflats (boolean)

Doom Builder will allow the use of flats on walls and textures on floors when this is set to **true**. Textures and flats will be mixed with priority to the original purpose. For textures this means that textures override flats with the same name, for flats this means that flats override textures with the same name. Default is **false**.

defaulttexturescale (decimal)

The scale of textures when no scale is known. This is a scalar value: 0.5 is half the original size and double the resolution. Default is 1.0.

defaultflatscale (decimal)

The scale of flats when no scale is known. This is a scalar value: 0.5 is half the original size and double the resolution. Default is 1.0.

scaledtextureoffsets (boolean)

Determines if texture offsets are in world coordinates (unscaled by texture scale) or texture coordinates. Set to **true** to use texture coordinates. Default is **false**.

textures (structure)
This lists the marker lump names that indicate the begin and end of a list of textures that Doom Builder should load. There must be a separate structure for each range, for which the structure name doesn't matter. The range must have a 'start' setting and an 'end' setting of which the values must be the names of the start and end lumps (strings). Please note that PNAMES, TEXTURE1 and TEXTURE2 lumps do not need to be in the game configuration, they are always loaded when available.

Example:

```
textures
{
	zdoom1
	{
		start = "TX_START";
		end = "TX_END";
	}
}
```

patches (structure)
This lists the marker lump names that indicate the begin and end of a list of patches that Doom Builder should load. There must be a separate structure for each range, for which the structure name doesn't matter. The range must have a 'start' setting and an 'end' setting of

which the values must be the names of the start and end lumps (strings). Note that Doom Builder does not load all patches, only those that are used by the textures.

Example:

```
patches
{
	standard1
	{
		start = "P_START";
		end = "P_END";
	}

	standard2
	{
		start = "PP_START";
		end = "PP_END";
	}
}
```

sprites (structure)
This lists the marker lump names that indicate the begin and end of a list of sprites that Doom Builder should load. There must be a separate structure for each range, for which the structure name doesn't matter. The range must have a 'start' setting and an 'end' setting of which the values must be the names of the start and end lumps (strings). Note that Doom Builder does not load all sprites, only those that are used by the things.

Example:

```
sprites
{
	standard1
	{
		start = "S_START";
		end = "S_END";
	}
}
```

flats (structure)

This lists the marker lump names that indicate the begin and end of a list of flats that Doom Builder should load. There must be a separate structure for each range, for which the structure name doesn't matter. The range must have a 'start' setting and an 'end' setting of which the values must be the names of the start and end lumps (strings).

Example:

```
flats
{
	standard1
	{
		start = "F_START";
		end = "F_END";
	}
}
```

colormaps (structure)

This lists the marker lump names that indicate the begin and end of a list of colormaps that Doom Builder should load. There must be a separate structure for each range, for which the structure name doesn't matter. The range must have a 'start' setting and an 'end' setting of which the values must be the names of the start and end lumps (strings).

Example:

```
colormaps
{
	standard1
	{
		start = "C_START";
		end = "C_END";
	}
}
```

Game Configuration - Sectors Settings

generalizedsectors (boolean)

Set to **true** to support generalized sector effects. This makes the **gen_sectortypes** structure mandatory. Default value is **false**.

sectorbrightness (structure)

This structure provides Doom Builder with a list of sector brightness levels that are most common. Doom Builder will use these levels to increase/decrease the brightness quickly. The structure must contain numeric setting names for the brightness levels. The settings don't need a value and any value will be ignored by Doom Builder.

Example:

```
sectorbrightness
{
        96;
        64;
        32;
        0;
}
```

gen_sectortypes (structure)

Generalized sector types are described in this structure. This structure is required when **generalizedsectors** is set to **true**. For each option there should be a structure. The name of the structure is displayed as the option description. Each option structure should contain a setting for the available choices. The setting name must be a numeric value that is added to the final sector effect value along with the values of the choices from other options (so the final sector effect value is the sum of the choices from every option). The setting must have a string value containing the choice description to be displayed.

Example:

```
gen_sectortypes
{
        secret
        {
                0 = "No";
                128 = "Yes";
        }

        friction
        {
                0 = "Disabled";
                256 = "Enabled";
        }

        wind
        {
                0 = "Disabled";
                512 = "Enabled";
        }
}
```

sectortypes (structure)

This is a simple list of all available sector effects that the user can choose from. The setting names must be numeric (the sector effect number) and the value must be a string containing the description to display.

Example:

```
sectortypes
{
        0 = "Normal";
        1 = "Light Blinks (randomly)";
        2 = "Light Blinks (2 Hz)";
        3 = "Light Blinks (1 Hz)";
        4 = "Damage -10 or 20% health and Light Blinks (2 Hz)";
        5 = "Damage -5 or 10% health";
        7 = "Damage -2 or 5% health";
        8 = "Light Glows (1+ sec)";
        9 = "Secret";
}
```

Game Configuration - Linedefs Settings

generalizedlinedefs (boolean)

Set to **true** to support generalized linedef actions. This makes the **gen_linedeftypes** structure mandatory. Default value is **false**.

linedefflags (structure)

Lists the options that can be set on a linedef. In case of a map format that works with numeric flags, the values of the chosen options are added together to form the final linedef flags value (so each option should use its own bit). Note that with numeric flags, the linedef activation flags are also incorporated in the same value (see **linedefactivations**).
Example for numeric flags:

```
linedefflags
{
	1 = "Impassable";
	2 = "Block Monster";
	4 = "Double Sided";
	8 = "Upper Unpegged";
	16 = "Lower Unpegged";
	32 = "Secret";
	64 = "Block Sound";
}
```

Example for named flags:

```
linedefflags
{
	blocking = "Impassable";
```

```
        blockmonsters = "Block monster";
        twosided = "Doublesided";
        dontpegtop = "Upper unpegged";
        dontpegbottom = "Lower unpegged";
        secret = "Secret";
        blocksound = "Block sound";
}
```

linedefactivations (structure)

This provides a list of choices about how a linedef is activated. Only one of these choices can be selected by the user. In case of a map format that works with numeric flags, the value is part of the linedef flags value and should have it's own range of bits. To separate these bits from the linedef flag options, use the **linedefactivationsfilter** setting. For map formats which use named flags, you can just use names for the settings.

Example for numeric flags:

```
linedefactivations
{
        0 = "Player walks over";
        1024 = "Player presses Use";
        2048 = "Monster walks over";
        3072 = "Projectile hits";
        4096 = "Player bumps";
        5120 = "Projectile flies over";
}
```

linedefactivationsfilter (integer)

Bit mask value that separates the **linedefactivations** bits from the

linedefflags bits. This is required in map formats that use numeric linedef flags, because the bits share the same integer linedef flags value in the map data. This setting is ignored for map formats that use named flags.

linedefflagstranslation (structure)

This is a translation between named (UDMF) linedef flags and the numeric linedef flags that your configuration uses (including linedef activation flags, if any). This structure is mandatory for all Game Configurations that do not use the UDMF map format and should not exist in Game Configurations that use the UDMF map format. Doom Builder uses this translation ot correctly work with copy/paste and prefabs (which are all converted to/from UDMF format). The setting names should be the non-UDMF flags. In case that your Game Configuration uses numeric flags map format, the setting names should be numeric. The setting values must be the equivalent UDMF flag names. The value can be prefixed with **!** to indicate that the value should be inverted (for example, if a setting 32 with value '!raisable' is to be converted, the existance of the bit value 32 will set 'raisable' to **false**).

Example for numeric flags:

```
linedefflagstranslation
{
	1 = "blocking";
	2 = "blockmonsters";
	4 = "twosided";
	8 = "dontpegtop";
	16 = "dontpegbottom";
```

```
        32 = "secret";
        64 = "blocksound";
        512 = "repeatspecial";
        1024 = "playeruse";
        2048 = "monstercross";
        3072 = "impact";
        4096 = "playerpush";
        5120 = "missilecross";
        8192 = "monsteractivate";
}
```

Game Configuration - Things Settings

defaultthingflags (structure)

This defines what the default flags should be first the first new thing when inserted. In map formats that use numeric thing flags, the settings in this structure should be the numeric flags to set. In map formats that use named flags, the settings must be the names of the flags to set. The value of the settings is optional and is ignored by Doom Builder.

Example for numeric flags:

```
defaultthingflags
{
	1;
	2;
	4;
	32;
}
```

Example for named flags:

```
defaultthingflags
{
	skill1;
	skill2;
	skill3;
	single;
	coop;
}
```

About Scripting Configurations

B