



Contents

WadAuthor
Copyright © 1995,1996 Williston Consulting
All Rights Reserved

Welcome to the WadAuthor help file! WadAuthor is intended to accommodate all levels of map design experience. Towards that end, novices looking for quick help should examine the getting started or tutorial topics, while those of greater experience will probably benefit more from the reference. I hope you enjoy using WadAuthor as much as I enjoyed creating it.



[What's New?](#)

Provides a detailed explanation of the changes from the previous release.



[Getting Started](#)

Provides important basic information and an introduction to using WadAuthor. Also provides instructions on accessing the step by step tutorial for creating a new map.



[Reference](#)

Provides detailed information for using WadAuthor. A basic level of experience is assumed for purposes of explanation. If you do not understand something, please consult the getting started section, the tutorial, or the glossary.



[Registration](#)

If you find WadAuthor useful, please examine this topic and register the product. Remember, only you can make shareware work!



[Troubleshooting](#)

When something doesn't seem to work right, consult this topic for answers to common problems and instructions for obtaining technical support.



[About WadAuthor](#)

Provides a release history along with some technical information about WadAuthor's construction.



[Credits And Thanks To...](#)

Software is rarely written in a vacuum, and this product is no different. Consult this topic for a list of people who made important contributions.

For Help on Help, Press F1

Az

Glossary



A

Attribute An attribute is very similar to a [property](#); it is a quality of something. Things and linedefs both have attributes which can affect their placement and behavior at runtime. Refer to the [thing properties dialog](#) or the [linedef properties dialog](#) for more information.

B

Binary space partition A binary space partition (BSP) is the output of a binary space partitioning algorithm. It divides a given space up into geometrically related pieces. Wadgames use a pre-calculated BSP tree to speed hidden surface removal at runtime.

C

Cleric The cleric is one of the three available player classes in Hexen. The cleric features a nearly even balance of fighting and magical ability.

Configuration file Short for wadgame configuration file (WCF), this refers to a data file WadAuthor uses to support different wadgames. Refer to the [configuration files](#) topic for more information.

Context menu Short for context-sensitive menu, this refers to a menu of options relevant to the current selection. Refer to the [context menus](#) topic for more information.

D

Deaf A thing having the deaf attribute will not attack the player without first "seeing" him or being attacked.

DOOM The original best-selling shareware game largely responsible for initiating the veritable slew of first-person perspective shoot-em-up games currently glutting the market.

DOOM][DOOM][: Hell on Earth is the sequel to DOOM.

Door A door, in wadfile terms, is actually a sector with a particular set of properties and linedefs. A "normal" door consists of a single four-sided sector whose ceiling height and floor height are identical; two of the sector's linedefs should also be two-sided, facing in opposite directions, and have their type set to trigger a local door. Click here to learn the simple method for [creating a door](#).

E

Enemy This term refers to any thing object that can move of its own volition to attack the player.

F

Fighter The fighter is one of the three available player classes in Hexen. The fighter features excellent strength and mobility while suffering from a lack of magical ability.

G

H

- Hall of mirrors This refers to the shimmering effect produced when a wadgame has no texture for a given wall. Click here for an explanation of [why the hall of mirrors effect happens](#).
- Heretic One of the wadgames for which WadAuthor is designed.
- Hexen One of the wadgames for which WadAuthor is designed.

I

- ID Software ID Software makes some of the best state-of-the-art games in for the PC. For more information call 1-800-IDGAMES.
- Impassable A linedef having the impassable attribute will not permit anything to cross.

J

K

L

- Linedef A linedef defines a line segment between two vertexes. Each linedef may have one or two sidedefs associated with it based on whether the player can ever see the back side.

M

- Mage The mage is one of the three available player classes in Hexen. The mage features excellent magical ability while suffering from poor fighting skills.
- Map A map is stored within a wadfile as a well-defined list of resources. Each map corresponds to a single "level" or "mission" when played. Refer to the [understanding wadmaps](#) topic for more information.

N

- Node building When a map is saved, quite a bit of information must be generated before the map can be used in a game. This information allows the game engine to quickly choose which walls need to be drawn for the player, avoiding time intensive computations at run-time. This process is referred to as node building.

O

- O The fifteenth letter of the English alphabet.
- Object Software and philosophy debates notwithstanding, this term refers to an object within a wadmap. An object is always one of the following types: a thing, vertex, linedef, sidedef, or sector. Of these types, things, vertexes, linedefs, and sectors may be directly manipulated from the [map editing view](#); sidedefs are grouped with linedefs for simplicity.

P

- Pcode Pcode is short for pseudo-code. It refers to an interpreted series of instructions, usually compiled from a higher level language.
- Polyobject A polyobject is a complex architectural feature whose location is resolved at map load time. Polyobjects were added with the release of Hexen to support

behavior otherwise not allowed within the constraints of previous wadgame engines. Swinging doors, rotating walls, et. al. are examples of polyobject use.

Property A property is a piece of information or setting relevant to a given object. For example, thing properties include a class, type, attributes, and other information.

Q R

Raven Software A company with some ties to Id Software. They are partially responsible for producing the Heretic and Hexen wadgames.

Runner For WadAuthor's purposes, at least, this term refers to the portion of a step facing the player when ascending a staircase.

Rubberband In WadAuthor, this term refers to an operation by which the user indicates a rectangle within the current map for further processing. The term is used because of the flexible way in which the rectangle is specified.

S

Script A series of instructions to be executed. Scripts were introduced with the Hexen wadgame.

Secret A sector having the secret attribute counts towards the evaluation presented to the player at the end of each level. A linedef with the secret attribute will appear as a normal one-sided linedef on the player's map at runtime — this can be very useful when [creating a secret room](#).

Sector A sector defines a room within a map. It contains lighting conditions, floor and ceiling heights, and other information.

Sidedef A sidedef defines how a linedef appears when viewed from a given side.

Stairwell For WadAuthor's purposes, at least, this term refers to the side walls of a staircase. For example, when converting a sub-sector into a rising staircase, the stairwell will be the newly formed walls protruding from the parent sector's floor.

T

Thing A thing defines a single object within a map. Most things interact with the player in some way, although some are strictly scenery. A few examples are weapons, ammunition, and enemies.

Texture A texture is a bitmap image applied to a geometric surface by the wadgame engine.

Texture mapping Texture mapping, in the general sense, refers to an algorithm that applies a bitmap to a surface. In the wadgame specific sense, it refers to the process of applying the linedef and sector images to the walls, ceiling, and floor to create a believable environment.

Track When talking about a door, this term refers to the generally one-sided linedefs along which the door rises when opened.

U

Undo The undo feature allows the user to reverse the effects of the last editing operation. WadAuthor maintains an undo operation for every editing operation

performed since the last time the map was saved.

V

Vertex

A vertex defines a single point within a map via Cartesian coordinates. For those who don't remember geometry, a Cartesian coordinate is a simple pair of orthogonal values usually denoted by (x,y).

W

Wadfile

A data file intended for use with several games from id software. A wadfile consists of a header, resources, and a resource directory. Refer to the [understanding wadfiles](#) topic for more information.

Wadgame

Wadgame is a term I have coined to refer to any game utilizing a wadfile. A few examples are DOOM, DOOM][, and HERETIC.

X

Y

Z

No Help Available

Sorry, but there is no help topic available for the current context.



About WadAuthor

WadAuthor came about during the process of crafting my first wadfile for DOOM. I truly prefer the Windows environment to DOS, and it frustrated me greatly that the only editor I could really count on was a DOS application! As I learned more about the DOOM engine during the construction process, I started thinking fondly about creating my own editor. Because several of the technical issues involved were largely unfamiliar ground to me, I approached the entire thing as one big research project. WadAuthor is the result of that process.

Date

Release notes

- 7/26/96 Release 1.30. If an external wadfile is used for additional images, it will now be included when running the map. Fixed a focus problem with the image browser dialog incremental searching. Improved zoom under WindowsNT to a maximum of 400%. Improved zoom under Windows95 to a maximum of 250% by limiting map size to 16000 units square. Changed the tag dialog to center the map editing view on the selected object. Added the make column option to the new rectangular sector dialog and the new polygonal sector dialog. Added the current zoom setting to the status bar. The image browser dialog now appends an asterisk to the name of an image if it is not supplied by the main wadfile. When running a map, WadAuthor will now create a response file for passing arguments if the command line length is greater than the operating system allows. Added panning on Ctrl+Shift primary click. This feature allows the user to scroll the map view by grabbing a location and dragging rather than by using the scroll bars. Added the run map dialog for customizing options immediately prior to launching the game. Added the tip of the day dialog to provide on-going help and amusement.
- 6/21/96 Release v1.22. Fixed duplicate thing description problem with Hexen. Changed shift-primary-click to add an object to the selection if the user does not drag a rubberband selection. Changed primary-click on a selected object to select exclusively if a move operation is not performed. Changed the automatic tag assignment feature to select first available tag rather than use the highest numbered tag plus one. Changed the rubberband selection to add to the existing selection rather than clearing it first. Changed the current object handling to clear automatically when the cursor is not over an object. Added code for handling long filenames correctly when running a map or executing user-defined tools. Made waiting for user-defined tool execution an option. User-defined tool replaceable parameters now include long filename macros for passing long filenames to 32-bit Windows applications. Added the How To Make A Lift help topic. Added the ability to make one linedef parallel to another, optionally aligning it at a fixed distance.
- 5/31/96 Release v1.21. Fixed a GPF when setting external image handling to the current document with an unsaved document. Fixed a bug that caused thing types to appear multiple times within the thing property dialog. Fixed a bug preventing the save of a read-only file under a new filename. Fixed a selection list bug causing a GPF during map reloading. Fixed image acceleration code causing a GPF during map reload. Fixed GPF during image browsing for flat images of sizes other than 64 x 64 pixels. Fixed stair motif floor and ceiling runner image list controls to browse textures rather than flats. Fixed invalid Heretic defaults for stair motifs. Fixed detection of flats within a graphics PWAD using FF_START/FF_END. Added 'X' keystroke shortcut for splitting selected linedefs. Added 'J' keystroke shortcut for joining selected objects. Changed the tag handling in property dialogs to allow selection from a list of available tags. Added the ability to set linedef lengths.
- 2/19/96 Release v1.20. Flipping linedefs now forces a node rebuild during the next save operation. Script import and export dialog file types now work under Windows95. The missing Floor_ForceLightning special has been added to the Hexen configuration file. The Hexen Arc of Death thing has been re-classified as a weapon. The object insertion dialog has been fixed to request a location from the user. The map checking routine has been fixed to prevent interpreting a local door as having an invalid tag number. Fixed a problem with clipboard pasting that prevented the front sides of linedefs from getting properly fixed up with the correct surrounding sector number. The center on map object dialog no longer allows an object number one larger than the maximum. When deleting a sector, WadAuthor no longer deletes two-sided linedefs still in use by another sector. A bug that caused texture alignment to fail has been fixed. Added the ability to abort running the map if the WARUN.EXE utility is not present. Improved performance when browsing thing images by the addition of a multi-threaded preload routine at startup. Under Win32s (which does not support threads), the data will still be preloaded, it will simply delay application startup. When deleting a sector, WadAuthor now goes to much greater lengths to make sure all remaining one-sided linedefs have a main texture. Changed thing drawing to display facing angle for the appropriate classes. Added incremental searching to all image name edit controls. Clicking on an unused portion of the

map with the primary mouse button begins a rubberband selection. Changed insertion of rectangular sectors to allow separate specification of horizontal and vertical widths. Changed the center on map object dialog to select the number control when any of the type controls are selected to simply keyboard use. Added the ability to disable snap to grid on the fly during a drag operation by holding down the shift key. Improved drawing to make a distinction between one-sided and two-sided linedefs. By default, one-sided linedefs are drawn in a lighter color. Removed almost all restrictions on door conversion; any sector connected to at least one other sector may now be converted to a door. Changed insertion of rectangular sectors to snap upper left vertex to grid only if the snap to grid feature is enabled. Changed insertion of polygonal sectors and things to snap center to grid only if the snap to grid feature is enabled. Changed Hexen sector to door conversion to stop using up a tag number. Changed check map dialog image problem fixing to use images from the linedef to be fixed, if available, before relying upon the current sector motif. Made image caching defaults sensitive to operating environment: 75 under Windows/NT, 50 under Windows95, and 25 for all other environments. Image acceleration code has been extended to support all known modes (256 colors, 64k colors, 16M colors, and true color). Added Windows95 spin controls where appropriate to improve mouse interface. Changed polygonal sector calculations to produce normalized geometric figures. Canceling a drag and drop operation now restores the previous view settings. Optimized screen redraw performance. Added a floor and ceiling image alignment grid. To enable or disable this feature, select the Show Floor Grid option from the context menu. To change the default setting, choose Options from the Tools menu; the default setting may be changed from the Views page. Added a bookmark feature. WadAuthor now supports additional images within the current document or within an external wadfile. Added a document properties dialog. Added a feature to force a node rebuild during the next save operation. Added complete color customization. Added a full-screen view feature. Added a tag editing and assignment option to the context menu. The helpfile has been converted to a more Windows95 friendly format. WadAuthor now "remembers" the last open map for each wadfile.

- 12/18/95 Release v1.11.1. Fixed several bugs that went undetected during testing. Released without updating documentation as no feature changes occurred.
- 12/11/95 Release v1.11. Fixed the check map dialog problem list to use the correct system-wide highlighted text color setting. Fixed a GPF when trying to add a SCRIPTS resource to maps that did not previously have one. Fixed a bug that incorrectly identified maps as not being ready to run. Fixed a bug that prevented scripts from being compiled when using the 32-bit version under Win32s. Fixed a bug that prevented the script dialog buttons from appearing correctly when using the 32-bit version under Win32s. Fixed a bug that caused the grid setting to be 2 units when using the 32-bit version under Win32s. Fixed a spelling error in the helpfile about topic. Fixed a bug in the thing properties dialog that caused the thing type to be set incorrectly. Fixed a bug that prevented the available script numbers from being updated after a successful script editing session. Fixed a bug with running Hexen maps. Added manual entry of thing angle to support polyobjects. Added a helpfile topic on understanding polyobjects and expanded the understanding scripts topic. Added import and export buttons to the scripts dialog. Improved sector determination. Added full source-level script decompilation. Added an object zoom feature. Added some example Hexen wads to the product.
- 11/17/95 Release v1.10. Fixed several bugs in vertex and linedef joining. All known restrictions have been removed. Fixed a bug that occurred only when saving a new map failed. If the initial save failed for any reason, future attempts to save would fail. Fixed a bug when pasting a copied sub-sector into an unused portion of the map. Previously, the sub-sector's linedefs would not be correctly updated as one-sided. Fixed a bug in the 16-bit version that would cause all linedefs within a wadfile to be incorrectly described (usually as local doors or something similar). Pasting and undo operations in a wad image name edit control now properly update the associated viewer (if any). An object property editing bug has been fixed to prevent generation of an empty undo operation. Reduced map load times by block reading wadmap objects. Reduced internal node-building times slightly by internally caching more data. Added a drag cursor setting to the view options. Uncheck it if the normal drag cursor obscures the drag destination to an unacceptable degree. Added a default grid size setting to the view options. Added new keyboard accelerators '[' and ']' for increasing and decreasing the grid size. Some systems, particularly when running Windows95, would not recognize the Shift+Plus and Shift+Minus default accelerators. Added new undo and check map buttons to the standard toolbar. Decreased the minimum grid setting to two units. Improved sector determination to ignore linedefs whose front and back face the same sector. This solves identification problems with sectors that fold back upon themselves or contain "tripwire" linedefs. Changed secondary mouse button usage to be more consistent with Windows95 interface guidelines. Joining objects now re-checks the current map if the check map dialog is active. Added a fix all button to the check map dialog. When pressed, it will fix all errors capable of being fixed without input from the user. Replaced the new thing dialog with the standard thing properties dialog. Improved the thing properties dialog to allow graphical selection of the thing type. Clicking the thing preview will invoke the image browser dialog, providing an alphabetically sorted list of the available things. The tags dialog now displays the tags in ascending sorted numerical order. Added

- a user-defined tools section to the tools menu. Added new view and user toolbars. Added a new toolbars page to the options dialog. It allows the user to specify which toolbars should be visible. Optionally, the user may right click on any toolbar to change these settings without opening the options dialog. Changed all toolbars to support docking to the frame window and each other. Also added code to remember the toolbar layout between WadAuthor sessions. Added a dialog box for inserting a rectangular sector to allow the user to specify the side length, optionally using the current grid setting. Added raw data editing for single objects. This allows power users to directly specify the raw data that will be saved to disk. This feature should be used with care. Added limited support for Hexen, the newest wadgame from Raven/Id Software. Added support for editing and creating maps for the fourth episode of Ultimate DOOM. Clipboard support has been enhanced to support exchange of data across all game types. When transferring data from Hexen to other wadgames, some conversion may be necessary, but existing architecture will be preserved as much as possible.
- 9/29/95 Maintenance release v1.03. Rewrote document saving code to avoid several GPF's for maps with lots of sidedefs. Fixed bug that prevented the selected door motif from being applied during conversion. Fixed GPF when trying to join two-sided lines with an invalid back sector mapping. Fixed bug preventing drag and drop joining of two-sided linedefs. Added support for using an external node building utility. Added specific error messages for failed node building, failed application of wadfile changes, and new wadfile creation. Added the ability to set the sector mapping for linedef sidedefs. Enhanced the map statistics dialog. Added a progress dialog for save operations.
- 8/25/95 Maintenance release v1.02. Joining linedefs has been enhanced to allow joining of any two linedefs -- as long as the sectors involved are valid. Speeded up linedef drawing when zoomed in. Fixed duplicate vertex identification bug. Added views page to options dialog and renamed maps page to files page. When snap to grid is off, keyboard-based object insertion no longer snaps to grid. Using keyboard to set current object now updates status bar. Added multi-sector rotation and scaling with optional inclusion of things therein. Sectors will now be joined when joining vertexes. Several floating point errors in the 16-bit version have been fixed. Fixed bug with clipboard when copying a sub-sector into another sector. Fixed bug preventing maps with > 8192 objects from loading in 16-bit windows version. Add ability to rename the current map within the current wadfile. Added the *Select All* option to the *Edit* menu. Added the *Tutorial*, *Troubleshooting*, and *What's New?* items to the *Help* menu. Improved the map checking routine to report linedefs with an invalid tag number. Using the check map dialog's fix button to repair problems with the wadfile was not properly allowing the action to be undone, nor marking the document as changed; this bug has been fixed. Fixed a bug in the check map dialog that prevented objects numbered zero from being viewed. A double-click in the type listbox of the new thing dialog exits the dialog, placing the selected thing.
- 7/17/95 Maintenance release v1.01. Added a snap to grid option, added color thing printing for color printers, changed polygonal sector limits, fixed a bug that prevented the *File/Save As...* option from working, fixed a palette handling bug that allowed WadAuthor's display to be corrupted when switching back from another program, fixed the VERSIONINFO resource, expanded the text for some linedef floor codes, added a check for the WARUN.EXE utility when running a map, added options dialog, and fixed a bug in the 16-bit version of the REJECT resource creation that prevented maps with more than 217 sectors from working.
- 6/30/95 Initial public release. Completed the keyboard interface, added thing display dialog, added texture alignment, added tags dialog, enhanced check map dialog, added printing and print preview, expanded the help file, improved current object determination for sectors, and breathed a big sigh of relief!
- 5/18/95 Released the second beta to some friends. Added sector scaling and rotation, added stair conversion, added door and stair motifs, added undo functionality, greatly expanded the help file, tied context-sensitive help into the app, added configuration files for DOOM [I] and HERETIC, added support for color depths above 256, added object filtering, and fixed quite a few bugs.
- 4/12/95 Released the first beta to some friends. Major functionality seems intact; lacking good documentation and some polish.

Technical Information

WadAuthor was born as a collection of humble 16-bit DOS utilities, migrated into a couple of Windows 3.x utilities, and finally became an MFC application using Microsoft Visual C++ v1.52. With the availability of Microsoft Visual C++ v2.0, WadAuthor started straddling the 32/16 bit fence as a Windows/NT native application. It has been tested in the Windows 3.1, Windows for Workgroups v3.11, Windows/NT v3.0-v4.0 (beta), Windows95 Preview, and Windows95 final release environments.

The main WadAuthor executable is compiled from 72,000+ lines of C++ code, roughly one-third of which is specific to wadfile editing. It relies on a couple hundred lines of AWK code used for generation of the wadgame configuration files and other miscellany. The help file, developed with the assistance of RoboHelp for Windows95, is compiled from around seventy separate document files and over one-hundred images.

Join Objects?

If two objects occupy the same position as the result of a drag and drop operation, WadAuthor will ask if the objects should be joined. Press the *Yes* button to join the objects, or press the *No* button to keep them separate.

WadAuthor Requires At Least 256 Colors

{button ,JI(`wauthor.HLP`,`Why_won_t_WadAuthor_run_with_a_sixteen_color_video_driver!')} [Related Topics](#)

WadAuthor has determined that your current video mode supports less than 256 colors. WadAuthor requires at least 256 colors for normal operation. Please select a different video mode and try again. Press the *OK* button to dismiss the dialog box and close the application.

WadAuthor Requires A Configuration File

{button ,KL(`Configuration Files',0,`,`')} [Related Topics](#)

WadAuthor requires a valid wadgame configuration file for normal operation. Please locate one or re-install if necessary before trying again. Press the *OK* button to dismiss the dialog box and close the application.

Close The Check Map Dialog?

Some operations require that the check map dialog box be closed. Press the *Yes* button to allow the check map dialog box to closed, or press the *No* button to cancel the pending operation.

The Map Must First Be Saved

Some operations require that the map be saved. You have not allowed the software to do so; thus, the operation will be canceled. Press the *OK* button to dismiss the dialog box.

Not A Valid Wadfile

The wadfile in question has an invalid header, an invalid directory, or contains corrupt data. Please select a different wadfile. Press the *OK* button to dismiss the dialog box.

Remove The Motif?

{button ,KL(`Motifs',0,`,`')} [Related Topics](#)

WadAuthor needs confirmation that you want to remove the currently selected motif. Please press the *Yes* button to confirm, or press the *No* button to cancel the removal.

Invalid Motif Name

{button ,KL(`Motifs',0,`,`')} [Related Topics](#)

The motif name you have entered is invalid. Motif names cannot be blank. Please press the *OK* button to dismiss the dialog box and enter a valid motif name.

Remove The Tag?

{button ,KL(`Motifs',0,`,`')} [Related Topics](#)

WadAuthor needs confirmation that you want to remove the curenly selected tag. Please press the *Yes* button to confirm, or press the *No* button to cancel the removal.

Reassign The Tag?

{button ,KL(`Tags',0,`,`')} [Related Topics](#)

The object already has a different tag. Press the Yes button to assign it the current tag number, or press the *No* button to cancel the operation.

Assign A New Tag?

{button ,KL(`Tags',0,`,`')} [Related Topics](#)

Either the current map uses no tags, or you do not have one selected. Press the *Yes* button to create a new tag and assign it to the object, or press the *No* button to cancel the operation.

Invalid Software License

The software license file is either missing or corrupt. Reinstall the software and try again.

Registration Attempt Failed

The data supplied to the register dialog was invalid. Please try again. Remember, the information is case sensitive and must be typed exactly as it appears in order to work properly.

Could Not Execute

{button ,KL(`The Run Map Feature Fails',0,`,`')} [Related Topics](#)

The run map feature has failed. Check the related topics for more information.

WARUN.EXE Is Missing

WadAuthor could not locate WARUN.EXE in the same directory as the WadAuthor executable. The WARUN.EXE utility is used to launch the appropriate wadgame for running the current map, avoiding certain operating system problems in the process. This message is informational in nature; WadAuthor will still try to run the map, but it may fail for reasons WARUN.EXE was designed to prevent. You may press the *OK* button to try anyway, or press the *Cancel* button to abort running the map.

Could Not Write To File

This error message, not surprisingly, means that WadAuthor tried to write data to a given file, and could not. When this occurs, it is usually due to one of several possible causes. These are listed below.

- The disk to which WadAuthor was trying to write is full.
- The disk to which WadAuthor was trying to write is copy protected.
- The file to which WadAuthor was trying to write is marked read only.

Confirm Map Replace

This message appears when trying to rename the current map if the new name already exists within the wadfile. Pressing the *Yes* button will cause the pre-existing map to be replaced. Pressing the *No* button cancels the operation.



Renaming the map, like any other editing operation, may be undone. WadAuthor will not actually change the map name until the file is saved, even though the title bar may seemingly indicate otherwise.

Node Build Failed

{button ,KL(`File Options Dialog',0,`,`')} [Related Topics](#)

This error message means that the node building portion of the save operation was not successful. If node building is set to *External*, check to make sure the supplied command is valid. If node building is set to *Internal*, then WadAuthor has suffered an internal failure; please contact the author and report the bug.

The Map Cannot Be Run

{button ,KL(`File Options Dialog',0,`,`')} [Related Topics](#)

This error message means that the current map does not have the runtime data necessary for the wadgame to use it. Because this data is generated during the node building process, this error message usually means one of the two things listed below.

- n The node building process failed during the last save operation.
- n Node building has been set to *None*.

Check the node building settings and save the current map before trying the run option again.

Wad Create Failed

This error message means that a new wadfile could not be created during the save operation. This is usually due to a lack of available disk space.

Save Failed

This error message means that WadAuthor could not create the entire wadfile on disk. During a save operation, WadAuthor copies all unchanged portions of the wadfile from the previous version, saving the changed portions directly from memory. If it cannot safely complete this process, the save operation will fail with this error message. This is usually due to a lack of available disk space.

Could Not Run Tool

This error message usually means that the command line or command-line parameters of the specified user-defined tool is incorrect. Check the tool parameters before trying again.

Could Not Save User Tools

This error message usually means that the user-defined tools file has been marked read-only or a disk error has occurred. Check to make sure that the file is not marked read-only and make sure enough disk space is available before trying again.

Wrong Configuration File

This error message means that you have tried to open a wadfile for which WadAuthor cannot use the current wadgame configuration file. For example, Hexen, unlike the other wadgames, uses different data structures for various map objects; this prevents WadAuthor from editing a Hexen map with a configuration file designed for another wadgame. Select a different wadgame configuration file before trying again.

Conversion Necessary

This message means that the data originated from a wadgame that uses different data structures for various map objects. Because of this, it is necessary to convert the data before proceeding. Some data loss can be expected.

Confirm Closing Tag Tool

In order to continue with the current operation, the tag tool must be closed. If you choose not to close the tag tool, the current operation will abort.

No Errors Detected

This message means that the last operation completed successfully without encountering any error conditions.

Missing Script Compilation Files

This error message means that one or more of the files required for compiling script code was not present in the WadAuthor directory. Please make sure the following files are present in the WadAuthor directory before trying again.

- n WARUN.EXE
- n ACC.EXE

Could Not Compile

This error message usually means that there wasn't enough memory to launch the script code compilation process. Check the amount of free memory before trying again.

Confirm Old Compiled Code

This message means that the script code has changed since the last time it was compiled. If you choose to continue anyway, the old compiled code saved to the wadfile will not well represent the newer script code.

Could Not Set Tag

This error message means that the specified object cannot be tagged. Please make sure you are specifying the correct object before trying again.

Field Cannot Be Blank

The specified data field cannot be left blank. Data is required for correct operation. Please enter valid data before continuing.

Confirm Map Name

The map name you have supplied could not be found in the main wadfile for the current wadgame. If you choose to continue, the map will not function with the current wadgame.

Confirm Script Code Editing

{button ,KL(`Scripts',0,`,`')} [Related Topics](#)

The current map did not contain a resource from which script code could be obtained when it was last loaded from disk. The map did, however, contain compiled pcode. If you create script code and save this map, it will overwrite the old pcode — possibly rendering the map unplayable.

Wait For Process

When running the 32-bit version of WadAuthor under Microsoft's Win32s, it is not possible for WadAuthor to correctly wait for a process to finish without jumping through some very obscure technical hoops.

To insure that WadAuthor waits until the process is finished to take further action, this message is displayed. When the process has finished, press the *OK* button to let WadAuthor know it is safe to proceed. Pressing the button early may cause the operation to fail.

Import/Export Failed

WadAuthor could not import or export the script code. When importing this usually means that the file is too big. When exporting this usually means that the filename supplied is already in use by a read-only file, or the disk is full.

Confirm Abandoning Changes

This message means that the data has been changed without being saved. Press the *Yes* button to continue and abandon the changes, or press the *No* button to abort the operation.

Replace Bookmark?

This message means that the bookmark being set already exists. Press the *Yes* button to replace the bookmark, or press the *No* button to abort the operation.

Undo Operation Failed

This message means that the undo operation could not restore the correct state prior to the last editing operation. It is probably wise to close the application and restart.

Thank You For Registering!

Thank you very much for registering WadAuthor! You are making shareware work.

Could Not Fix Error

This message means that one of the errors in the check map dialog could not be fixed. You will have to fix it manually.

Confirm Force Node Rebuild

Forcing a node rebuild, unlike other editing operations, cannot be undone. It will cause WadAuthor to remove the node data so that the map cannot be played until a node building operation is performed. Press the *Yes* button to continue, or press the *No* button to abort the operation.

Confirm Texture Use

{button ,KL(`Custom Images',0,`,`')} [Related Topics](#)

This message means that WadAuthor has detected additional textures to be used for editing but no associated panel names resource was found. You may press the *Yes* button to use the additional images, or press the *No* button to use only those images present in the main wadgame wadfile.

File Contains No Maps

This message means that WadAuthor could not find any maps in the given wadfile. Please select a different wadfile and try again.



Credits And Thanks To...

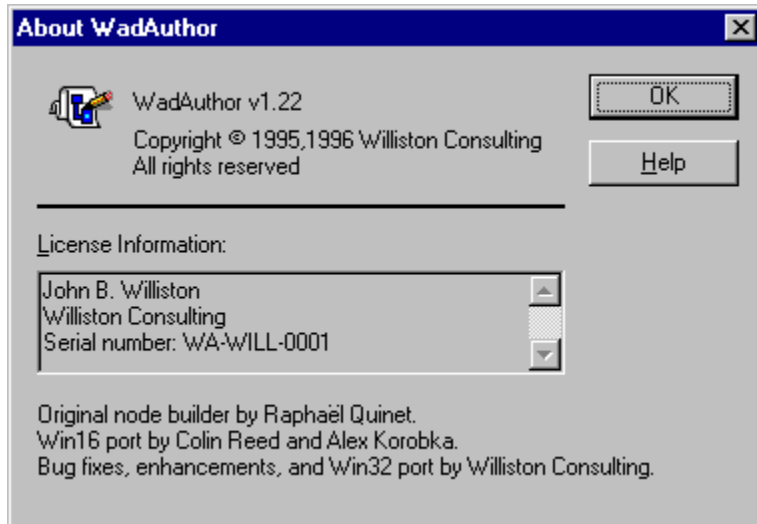
- n My wife, for her beauty, love, and patience.
- n Tim "let me know when it's done so I can use it" Walker, for his input, beta-testing, and letting me borrow DOOM][for a while.
- n Steve [Captain] Hoek, for loads of interface suggestions and constant reality checks.
- n Pat "I have no idea how much it's costing us" Burrell, for helping me arrive at the final mouse interface.
- n Aaron Kamphuis for letting me borrow Heretic for a while.
- n Raphaël Quinet and any contributors to the Doom Editing Utilities (DEU), for providing source code for the node building process and several other items of interest.
- n Colin Reed and Alex Korobka for the initial porting of the node building code to Windows.
- n Ben Morris, author of The Doom Construction Kit (DCK), for making the best freeware DOS editor available.
- n Jack and Mike Vermeulen for help with the Hexen thing data.
- n Colin Caird for helping me understand polyobjects.
- n Luc Cluitmans for taking my script disassembler and turning it into a decompiler.
- n Olivier Montanuy for helping me understand how best to support custom images.
- n Steve Benner for providing a wealth of good user-interface suggestions.
- n Jan-Albert B. van Ree and WTF Productions for providing me with a special version of their editing guide.
- n ID Software, for creating one of the most thoroughly enjoyable, albeit theologically ridiculous, games I've ever played.

Most of all my thanks goes to the Lord Jesus Christ, for doing that which I could not; truly "Thine is the kingdom, and the power, and the glory, forever. Amen." (NAS Mat 6:13)

I would like to dedicate WadAuthor to the memory of my father, John A. Williston, who for some reason decided to end his life over the weekend of March 16-17, 1996. He helped me to build my first computer as a twelve year old boy, and I am well within the mark when I say that I would never have become the man I am today without him.

About Dialog

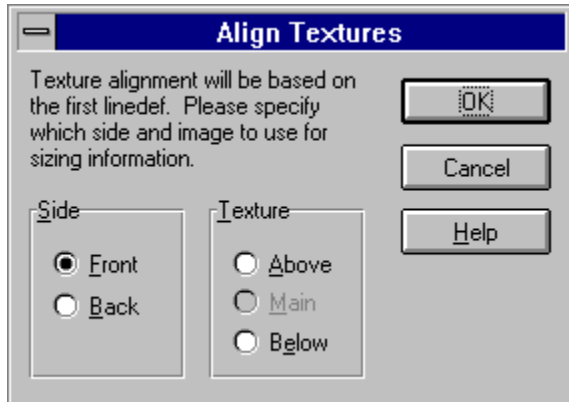
The about dialog box, shown in the illustration below, provides some basic information about WadAuthor as well as the software license information.



Align Textures Dialog

{button ,KL(`Textures',0,`,`')} [Related Topics](#)

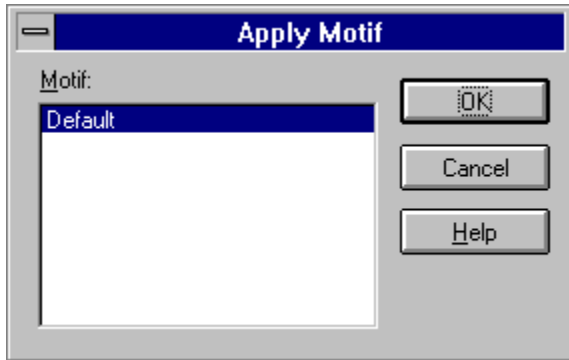
The align textures dialog, shown in the illustration below, allows the user to specify which side of the linedefs to align and which texture to use for sizing information.



Apply Motif Dialog

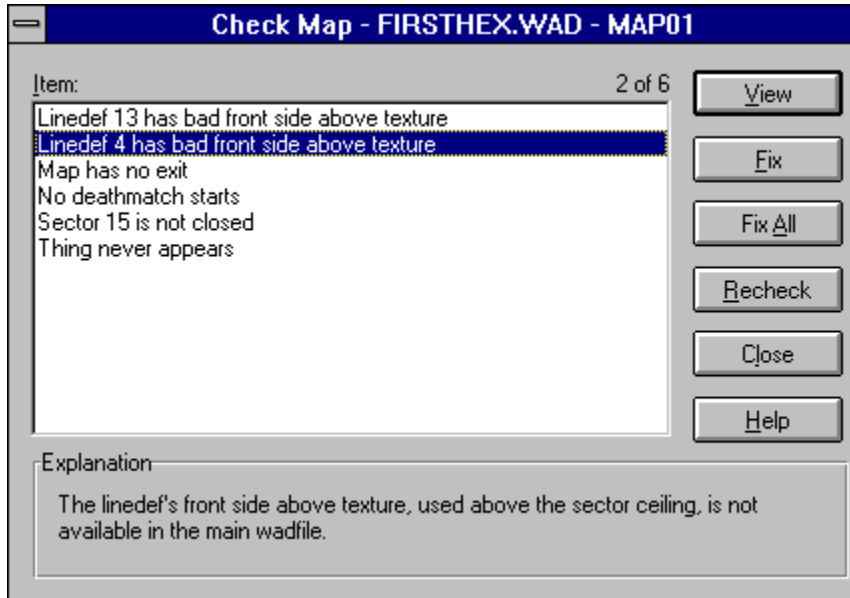
{button ,KL(`Motifs',0,`,`')}` [Related Topics](#)

The apply motif dialog, shown in the illustration below, allows the user to select a motif to be applied to selected sectors or used when converting a sector to a door. Select the desired motif and press the *OK* button, or press the *Cancel* button to abort.



Check Map Dialog

The check map dialog box, shown in the illustration below, simplifies finding and fixing map problems. You may click on the controls in the illustration below for more specific help.



Item List

This control contains the problems detected with the map. Problems that have already been fixed appear in gray.

Explanation

This control provides a brief explanation of the current problem.

View Button

This control centers the map on the current problem when pressed.

Fix Button

This control applies a default corrective action to solve the current problem when pressed.

Fix All Button

When pressed, this control applies a default corrective action to solve all of the problems for which a default corrective action exists.

Recheck Button

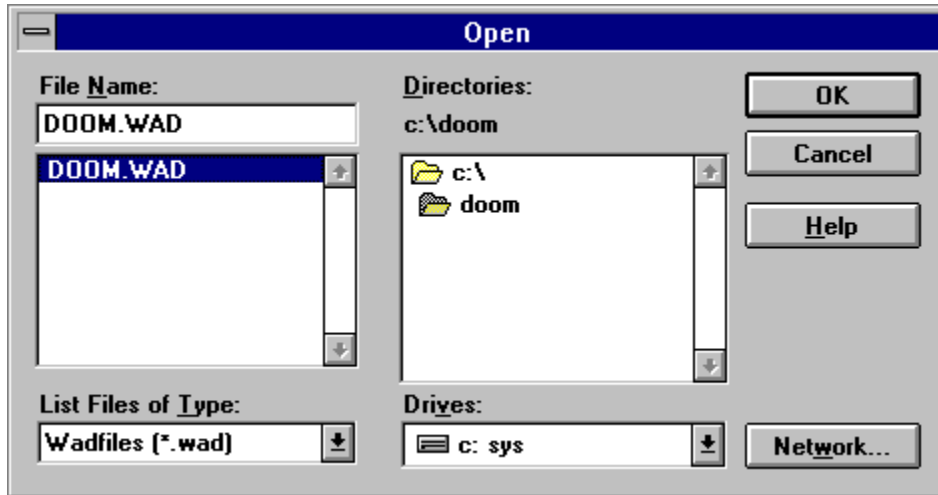
This control rechecks the map and updates the item list.

Close Button

This control closes the dialog when pressed.

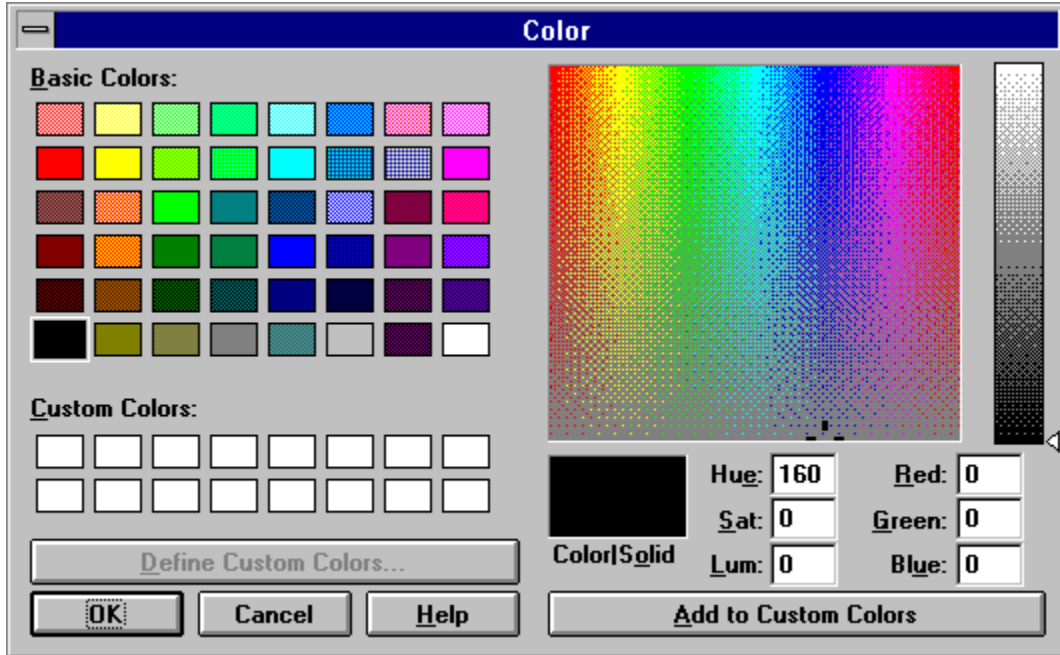
File Open/Save Common Dialog

The file open/save common dialog box, shown in the illustration below, is one of the common dialog boxes provided by Windows. WadAuthor uses the Windows common dialog box functions to provide a consistent file selection interface. For help on using the common dialogs, consult your Windows manual.



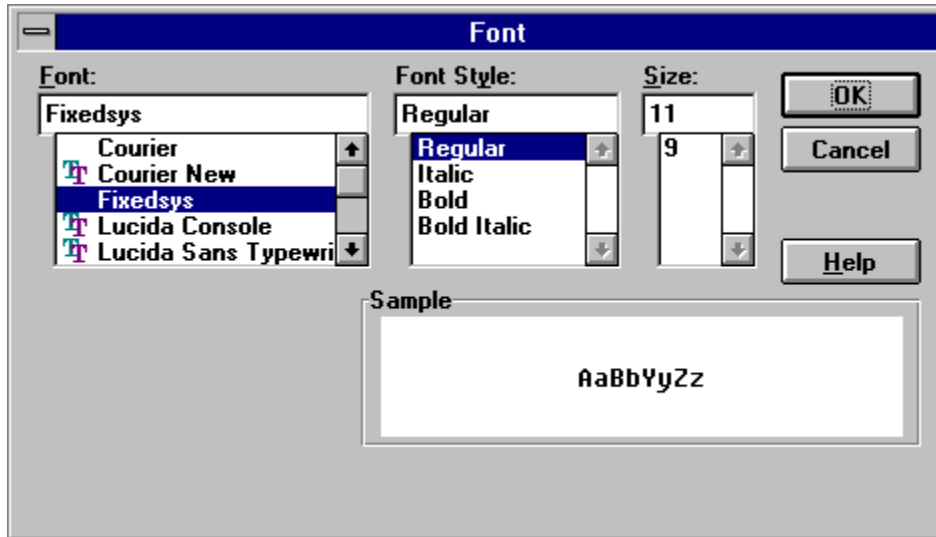
Color Selection Dialog

The color selection dialog, shown in the illustration below, is one of the common dialog boxes provided by Windows. For more information about the color selection dialog, consult your Windows manual.



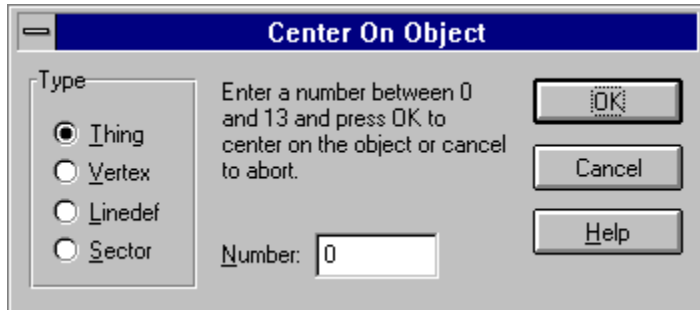
Font Selection Dialog

The font selection dialog, shown in the illustration below, is one of the common dialog boxes provided by Windows. For more information about the font selection dialog, consult your Windows manual.



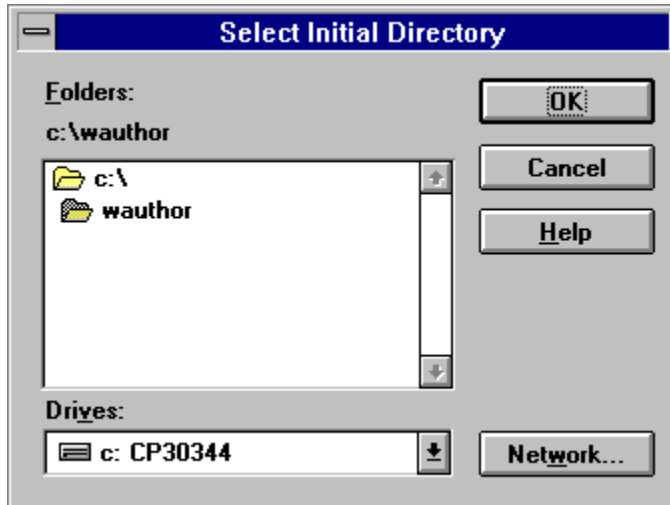
Center On Object Dialog

The center on object dialog, shown in the illustration below, prompts the user for an object type and a number. Press the *OK* button to center the map on the specified object, or press the *Cancel* button to abort.



Directory Selection Dialog

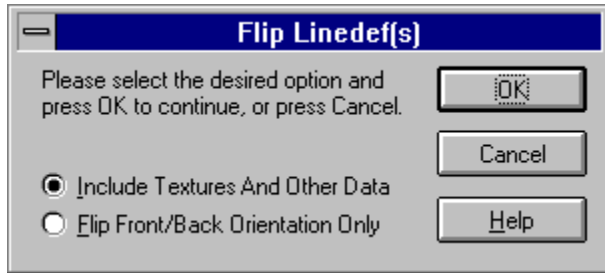
The directory selection dialog, shown in the illustration below, allows the user to choose a directory via a familiar folder-based interface.



Flip Linedef(s) Dialog

The flip linedefs dialog box, shown in the illustration below, allows the user to reverse one or more linedefs. The user must select one of the two options and press the *OK* button to continue, or press the *Cancel* button to abort. The two options are explained below.

- n **Include Textures And Other Data** is the more complete option, exchanging not only the starting and ending vertexes but also the sidedef texture information.
- n **Flip Front/Back Orientation Only** means that only the starting and ending vertexes will be exchanged. This has the effect of reversing the front and back sides. For example, if a linedef goes from vertex six to vertex three, selecting this option will change the linedef to be defined from vertex three to vertex six.

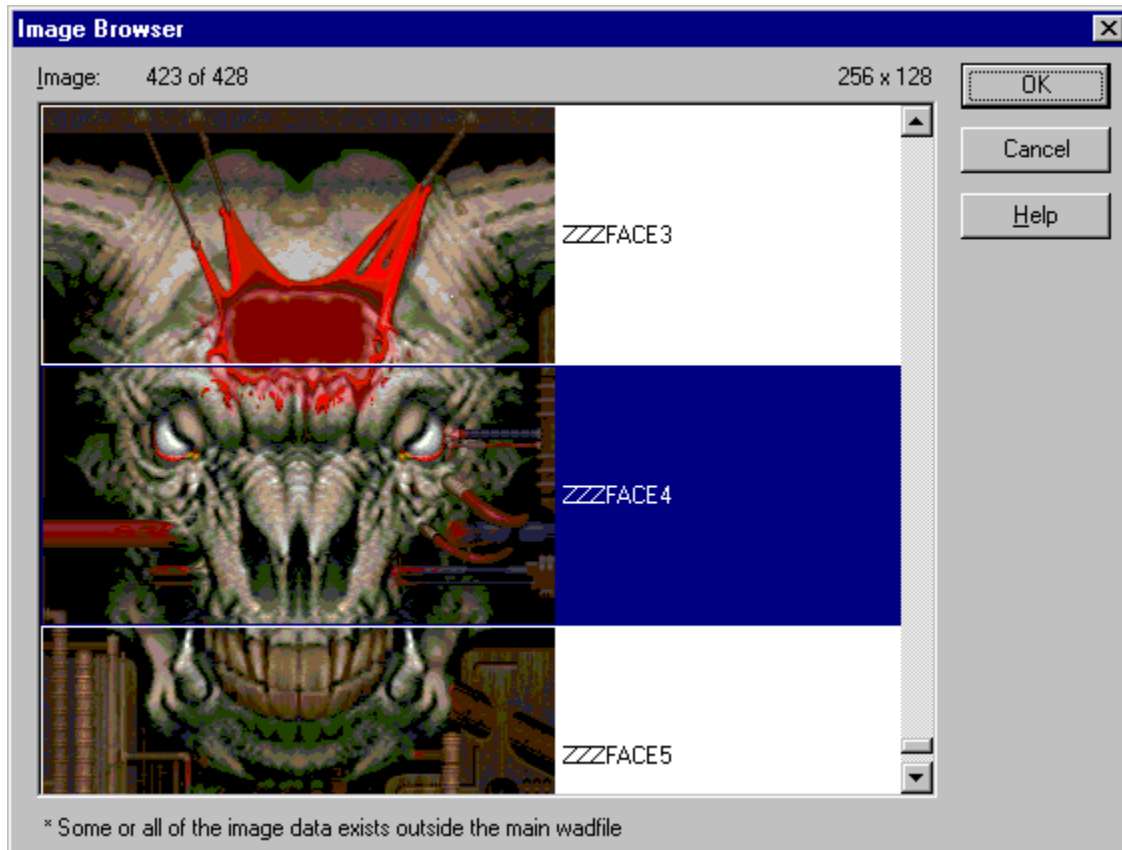


This feature should be used carefully, as it can cause map errors when used improperly. For example, flipping only the front/back orientation of a single-sided linedef will leave the undefined back sidedef facing into the given sector.

Image Browser Dialog

The image browser dialog, shown in the illustration below, is used throughout WadAuthor to simplify image selection. You may select images by using the mouse to navigate the listbox, or by using the incremental search feature. Press the *OK* button to accept the selected image, or press the *Cancel* button to close the dialog box without accepting the selected image.

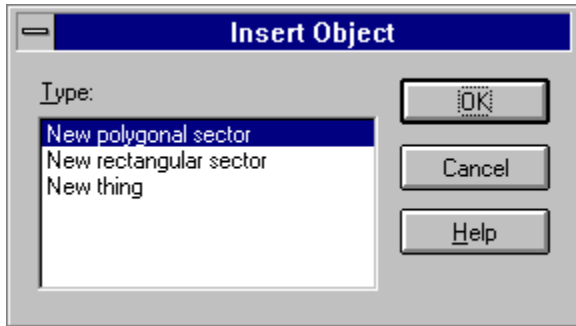
To use the incremental search feature, move to the image listbox and begin typing the first few characters of the desired image. A small edit window will appear near the listbox where the characters you type will be displayed. As you type more of the name, the listbox selection is repositioned accordingly. Press the escape key to cancel the incremental search, or press the enter key to accept the current results of the search.



If an asterisk follows the image name, this indicates that the image data is not supplied by the main wadfile.

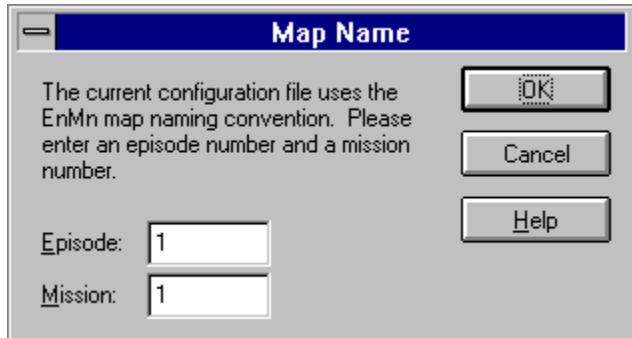
Insert Object Dialog

The insert object dialog, shown in the illustration below, is displayed to allow the user to select the type of object to be inserted. Select the desired object and press the *OK* button to continue, or press the *Cancel* button to abort.



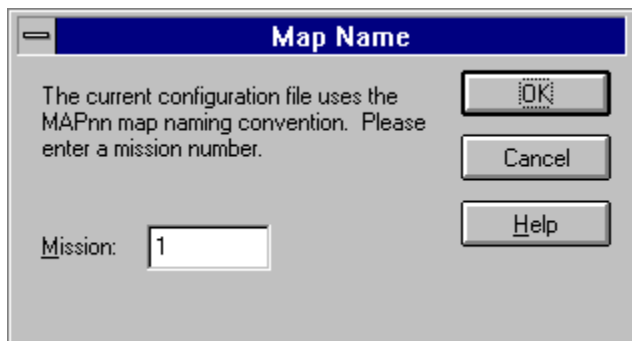
Map Name Dialog

The map name dialog, shown in the illustrations below, changes to accommodate the map naming convention of the current wadgame. In each case, the dialog box will supply instructions for providing a map name.



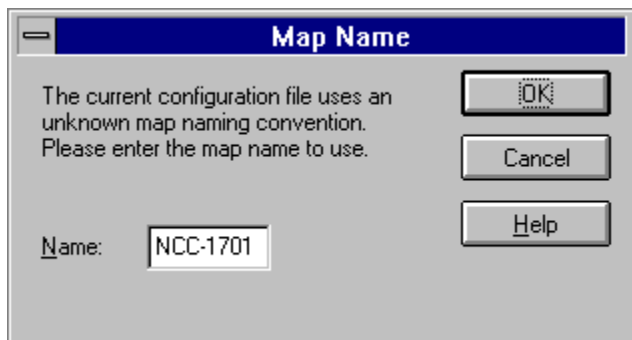
The dialog box titled "Map Name" has a blue header bar with a minus sign icon on the left. The main text reads: "The current configuration file uses the EnMn map naming convention. Please enter an episode number and a mission number." Below this text are two input fields: "Episode:" with the value "1" and "Mission:" with the value "1". To the right of the text are three buttons: "OK", "Cancel", and "Help".

The dialog box to the left is used for wadgames using the *EnMn* map naming convention.



The dialog box titled "Map Name" has a blue header bar with a minus sign icon on the left. The main text reads: "The current configuration file uses the MAPnn map naming convention. Please enter a mission number." Below this text is one input field: "Mission:" with the value "1". To the right of the text are three buttons: "OK", "Cancel", and "Help".

The dialog box to the left is used for wadgames using the *MAPnn* naming convention.



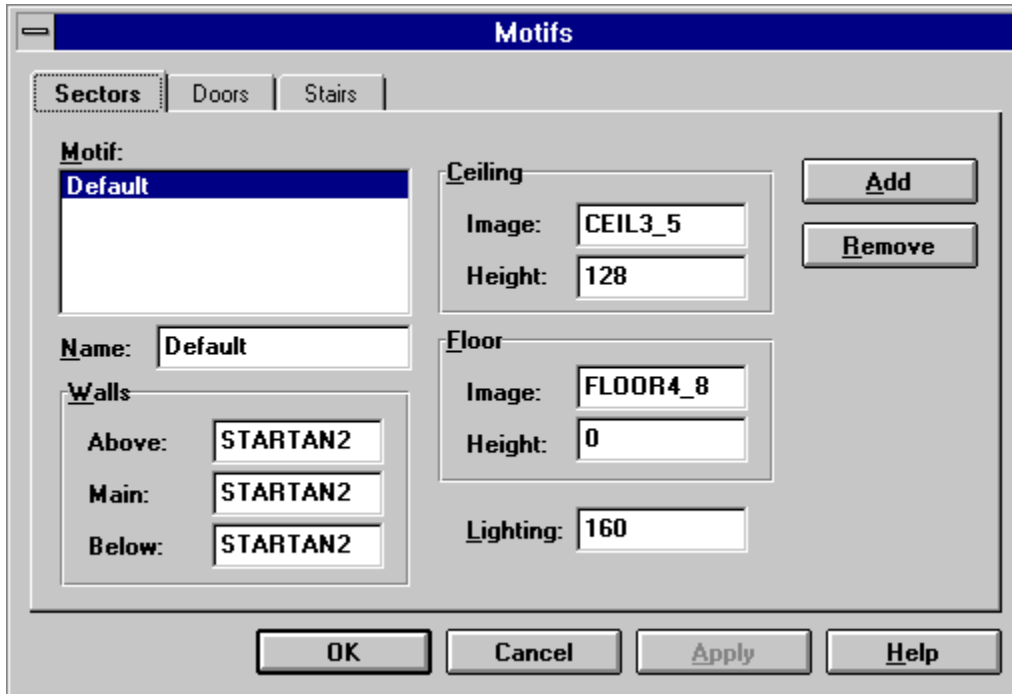
The dialog box titled "Map Name" has a blue header bar with a minus sign icon on the left. The main text reads: "The current configuration file uses an unknown map naming convention. Please enter the map name to use." Below this text is one input field: "Name:" with the value "NCC-1701". To the right of the text are three buttons: "OK", "Cancel", and "Help".

The dialog box to the left is used only when WadAuthor cannot understand the new map naming convention as specified in the current wadgame configuration file. It is intended to provide for future extensibility without requiring a new version of WadAuthor.

Sector Motifs Dialog

{button ,KL(` Motifs;Configuration Files',0,`,`')} [Related Topics](#)

The motifs dialog box sector motifs page, shown in the illustration below, allows the user to select a current sector motif, add new sector motifs, and delete or change existing sector motifs. You may click on the controls in the illustration below for more specific help.



The screenshot shows a dialog box titled "Motifs" with three tabs: "Sectors", "Doors", and "Stairs". The "Sectors" tab is active. The dialog is divided into several sections:

- Motif:** A list box containing "Default".
- Name:** A text field containing "Default".
- Walls:** Three text fields labeled "Above:", "Main:", and "Below:", each containing "STARTAN2".
- Ceiling:** Two text fields labeled "Image:" (containing "CEIL3_5") and "Height:" (containing "128").
- Floor:** Two text fields labeled "Image:" (containing "FLOOR4_8") and "Height:" (containing "0").
- Lighting:** A text field containing "160".

On the right side of the dialog, there are two buttons: "Add" and "Remove". At the bottom of the dialog, there are four buttons: "OK", "Cancel", "Apply", and "Help".

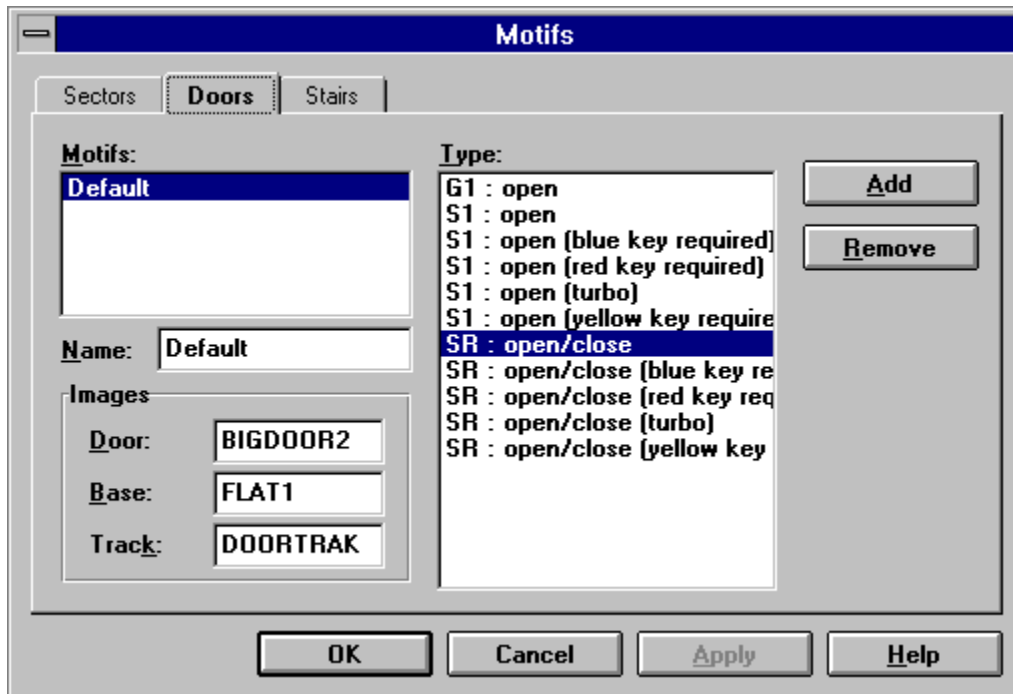


Because motifs contain wadgame specific information, a different motif file is used for each configuration file. Check the related topics for more information.

Door Motifs Dialog

{button ,KL(` Motifs;Configuration Files',0,`,`')} [Related Topics](#)

The motifs dialog box door motifs page, shown in the illustration below, allows the user to select a current door motif, add new door motifs, and delete or change existing door motifs. You may click on the controls in the illustration below for more specific help.



Because motifs contain wadgame specific information, a different motif file is used for each configuration file. Check the related topics for more information.

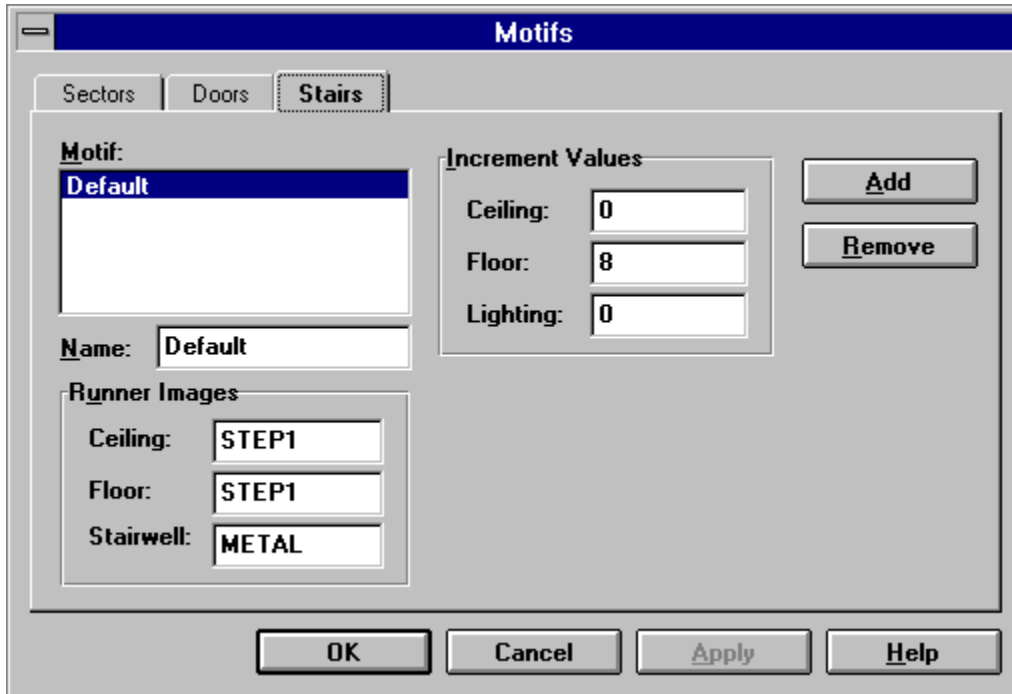
Door Activation

This listbox displays a list of the available door activation types. These refer to [linedef types](#) or specials depending upon the wadgame for which the map is intended.

Stair Motifs Dialog

{button ,KL(` Motifs;Configuration Files',0,`,`') } [Related Topics](#)

The motifs dialog box stair motifs page, shown in the illustration below, allows the user to select a current stair motif, add new stair motifs, and delete or change existing stair motifs. You may click on the controls in the illustration below for more specific help.



The screenshot shows a dialog box titled "Motifs" with three tabs: "Sectors", "Doors", and "Stairs". The "Stairs" tab is selected. The dialog is divided into several sections:

- Motif:** A list box containing "Default".
- Name:** A text field containing "Default".
- Increment Values:** Three input fields: "Ceiling:" with value "0", "Floor:" with value "8", and "Lighting:" with value "0".
- Runner Images:** Three input fields: "Ceiling:" with value "STEP1", "Floor:" with value "STEP1", and "Stairwell:" with value "METAL".

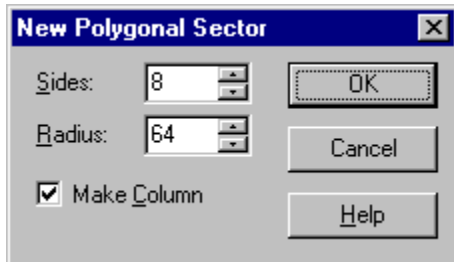
Buttons for "Add" and "Remove" are located to the right of the "Increment Values" section. At the bottom of the dialog are buttons for "OK", "Cancel", "Apply", and "Help".



Because motifs contain wadgame specific information, a different motif file is used for each configuration file. Check the related topics for more information.

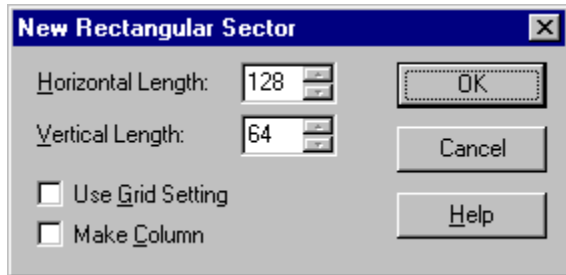
New Polygonal Sector Dialog

The new polygonal sector dialog, shown in the illustration below, allows the user to specify the number of sides and the radius when creating new polygonal sectors. If make column is checked, the linedefs of the new sector will be one-sided and impassable facing outward from the center; otherwise, the linedefs of the new sector will be two-sided and facing inward allowing the player to freely cross.



New Rectangular Sector Dialog

The new rectangular sector dialog, shown in the illustration below, allows the user to specify the length to use for each side, optionally using the current grid setting, when creating new rectangular sectors. If make column is checked, the linedefs of the new sector will be one-sided and impassable facing outward from the center; otherwise, the linedefs of the new sector will be two-sided and facing inward allowing the player to freely cross.



Object Filter Dialog

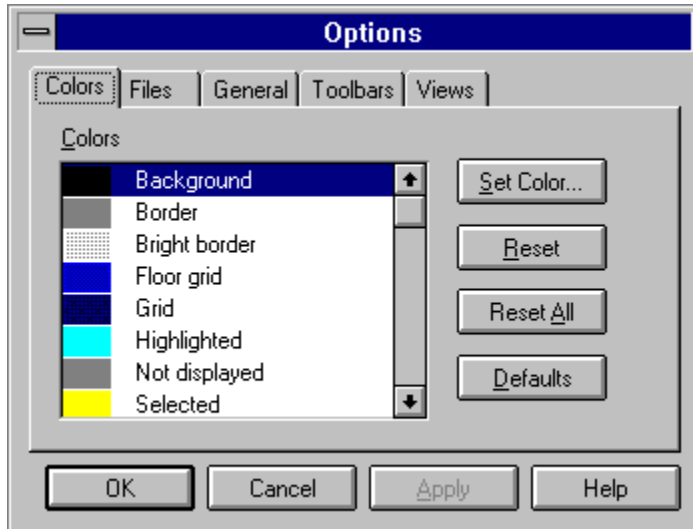
The object filter dialog, shown in the illustration below, allows the user to control which type of objects are affected by selection operations.



Colors Options Dialog

{button ,KL(`Options',0,`,`')} [Related Topics](#)

The options dialog colors page, shown in the illustration below, allows the user to configure the colors used by WadAuthor. You may click on the controls in the illustration below for more specific help.



Colors

The list of colors available for modification is displayed here. The usage of each color is explained in the following list.

- n **Background** is used for the map editing view background color.
- n **Border** is used for drawing two-sided linedefs and other general outlining.
- n **Bright Border** is used for one-sided linedefs.
- n **Floor Grid** is used to draw the floor and ceiling alignment grid markers.
- n **Grid** is used to draw the regular grid lines.
- n **Highlighted** is used for the current object and other highlighted objects.
- n **Not Displayed** is used for rendering things that do not appear at the current filter settings.
- n **Selected** is used to outline selected objects.
- n **Tagged** is used to outline tagged objects.

Set Color

The set color button, when pressed, displays the color selection common dialog box.

Reset

The reset button, when pressed, returns the currently selected color to its initial value when the dialog was first displayed.

Reset All

The reset button, when pressed returns all colors to their initial values when the dialog was first displayed.

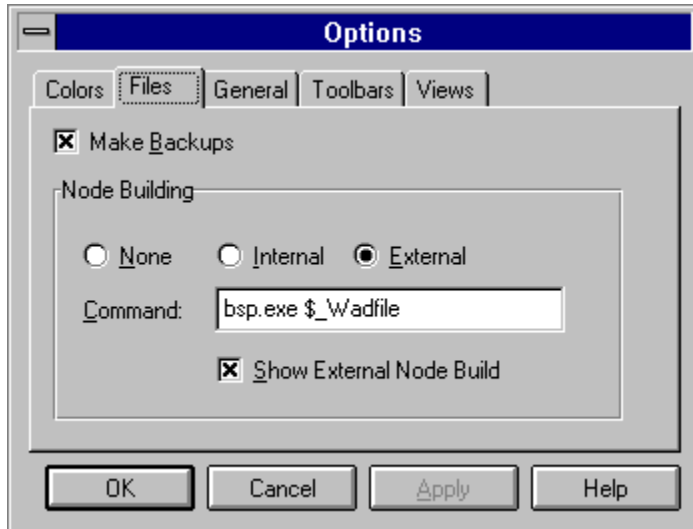
Defaults

The defaults button, when pressed, sets all colors to their default values.

File Options Dialog

{button ,KL(`Options',0,`,`')} [Related Topics](#)

The options dialog files page, shown in the illustration below, allows the user to configure various aspects of wadfile management in WadAuthor. You may click on the controls in the illustration below for more specific help.



Make Backups

The make backups option, when checked, causes WadAuthor to retain the previous version of a map during a save operation.

Node Building

The node building options determine how WadAuthor will handle a map that needs a node rebuild when saving. The options are explained below.

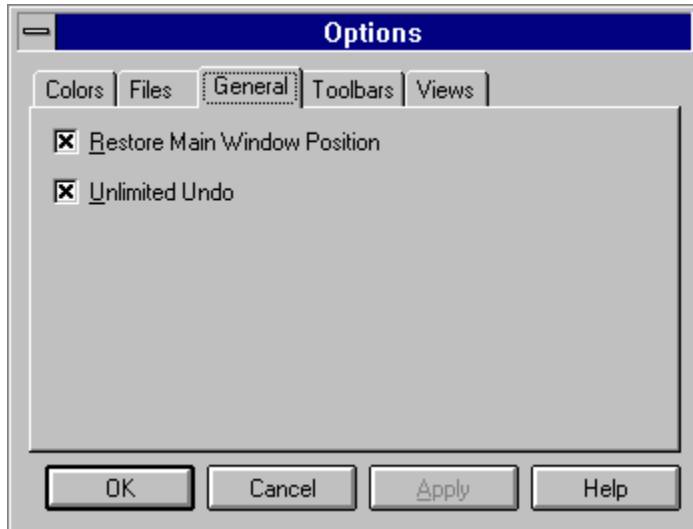
- n **None** will result in much faster save times — particularly when working with large maps. However, the map will not be playable until a node rebuild is performed.
- n **Internal** will cause WadAuthor to use its own internal node building code. This is the default option, and is probably the best in most circumstances.
- n **External** will cause WadAuthor to execute the user-specified command whenever a node rebuild is required. The command may use four macros, `$_Wadfile`, `$_Wadmap`, `$_Mapname`, and `$_Comspec`, to specify the fully qualified pathname of the current map, the command-line parameter appropriate to the map name, the actual map name, and the command shell processor.

When using *External* node building, checking the *Show External Node Build* option will make the node-building child process visible to the user.

General Options Dialog

{button ,KL(`Options',0,`,`')} [Related Topics](#)

The options dialog general page, shown in the illustration below, allows the user to configure various aspects of WadAuthor. You may click on the controls in the illustration below for more specific help.



Restore Main Window Position

The restore main window position option, when checked, will cause the location and sizing parameters of WadAuthor's main window to be remembered from session to session.

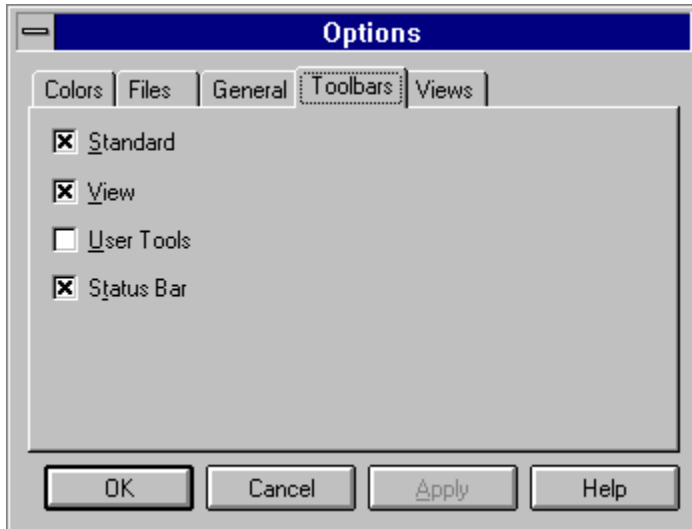
Unlimited Undo

The unlimited undo option, when checked, causes WadAuthor to maintain a complete undo history to the time of the last file save. Disabling this option, however, can save quite a bit of memory during long editing sessions.

Toolbar Options Dialog

{button ,KL(`Control Bars;Options',0,`,`')} [Related Topics](#)

The options dialog toolbar toolbars page, shown in the illustration below, allows the user to determine which toolbars are available for use. Toolbars may also be shown or hidden by right clicking on a toolbar and selecting from the resulting menu.

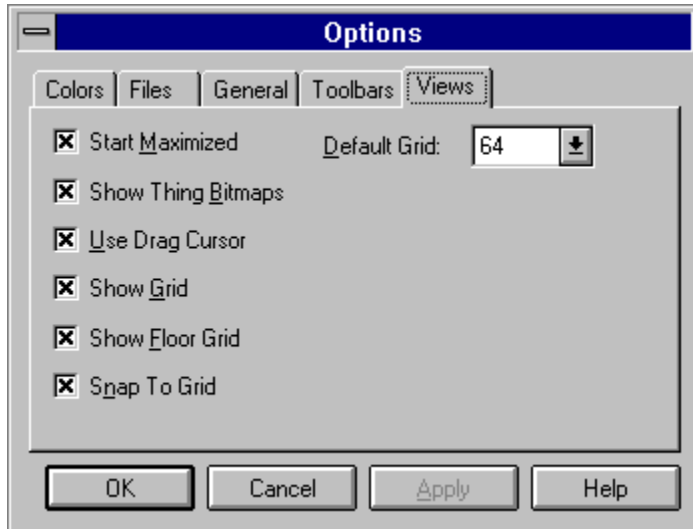


The view and user toolbars are not available in the 16-bit version of the software. This is due to a limitation of the 16-bit Microsoft Foundation Classes.

View Options Dialog

{button ,KL(`Options',0,`,`')} [Related Topics](#)

The options dialog views page, shown in the illustration below, allows the user to configure various aspects of the views in WadAuthor. You may click on the controls in the illustration below for more specific help.



Start Maximized

The start maximized option, when checked, causes new map editing views to start in a maximized state.

Show Grid

The show grid option, when checked, causes all new views to initially display the grid.

Show Floor Grid

The show floor grid option, when checked, causes all new views to initially display the floor and ceiling image alignment grid.

Show Thing Bitmaps

The show thing bitmaps option, when checked, causes all new views to initially display sprite bitmaps whenever possible to do so.

Snap To Grid

The snap to grid option, when checked, causes all new views to initially snap objects to grid whenever appropriate.

Use Drag Cursor

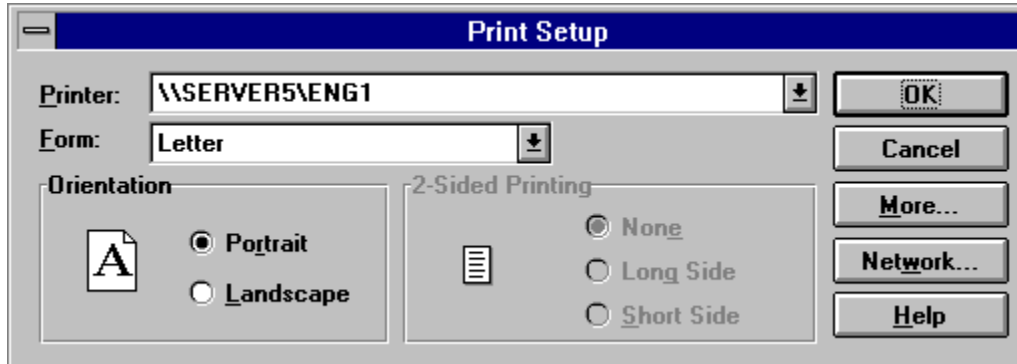
The use drag cursor option, when checked, causes WadAuthor to display the standard single-document or multi-document cursor during a drag operation. If this obscures the drop location marker to an unacceptable degree, uncheck the option.

Default Grid

The default grid setting determines the grid setting used for all new views.

Print Setup Dialog

The print setup dialog box, shown in the illustration below, allows you to select the destination printer and its connection as explained by the options following the illustration.



Printer

Select the printer you want to use. Choose the *default printer*; or choose the *specific printer* option and select one of the current installed printers shown in the box. You install printers and configure ports using the Windows Control Panel.

Orientation

Choose *portrait* or *landscape*.

Paper Size

Select the size of paper that the document is to be printed on.

Paper Source

Some printers offer multiple trays for different paper sources. Specify the tray here.

Options

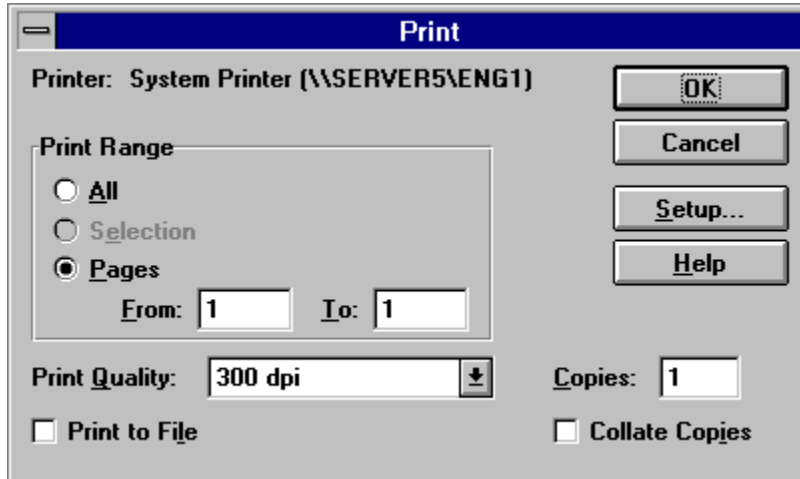
Displays a dialog box where you can make additional choices about printing, specific to the type of printer you have selected.

Network...

Choose this button to connect to a network location, assigning it a new drive letter.

Print Dialog

The print dialog box, shown in the illustration below, allows you to specify how the document should be printed as explained by the options following the illustration.



Printer

This is the active printer and printer connection. Choose the Setup option to change the printer and printer connection

Setup

Displays the print setup dialog box so you can select a printer and printer connection.

Print Range

Specify the pages you want to print:

- All** Prints the entire document.
- Selection** Prints the currently selected text.
- Pages** Prints the range of pages you specify in the From and To boxes.

Copies

Specify the number of copies you want to print for the above page range.

Collate Copies

Prints copies in page number order, instead of separated multiple copies of each page.

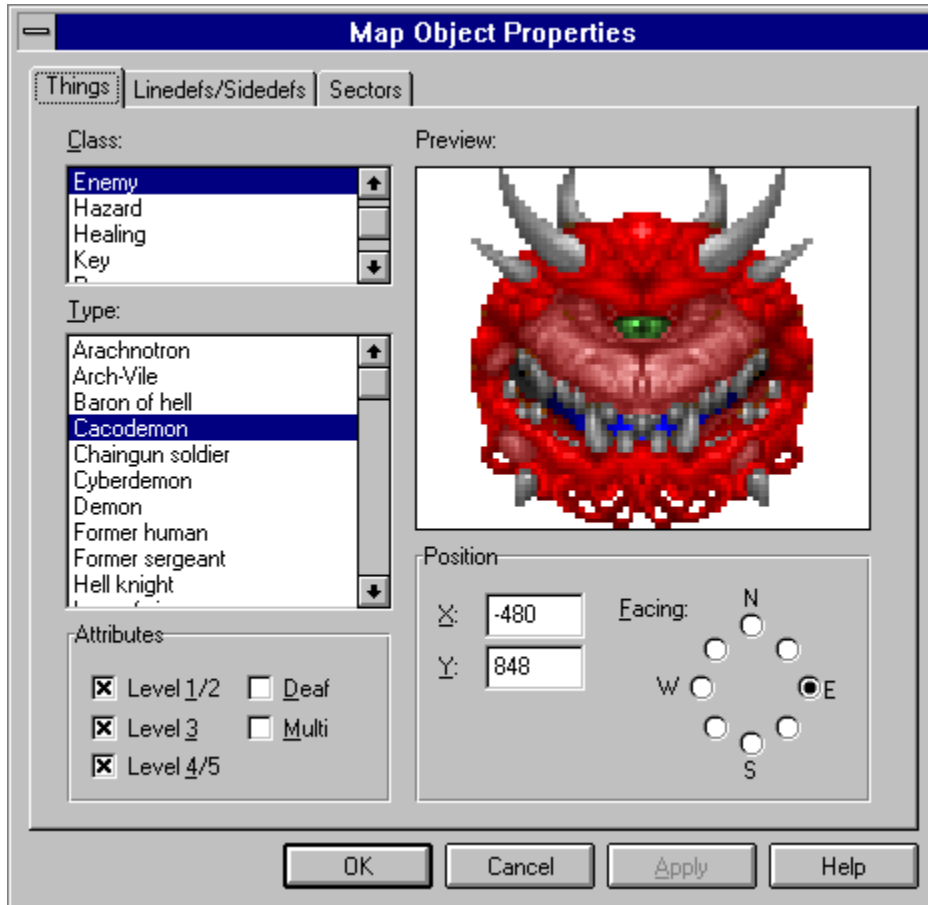
Print Quality

Select the quality of the printing. Generally, lower quality printing takes less time to produce.

Thing Properties Dialog

{button ,KL(` Properties;Textures',0,`,`')} [Related Topics](#)

The map object properties dialog thing page, shown in the illustration below, allows editing of multiple things. You may click on the controls in the illustration below for more specific help.



Thing Class

This is a somewhat arbitrary distinction made by WadAuthor to reduce the large number of thing types to a manageable list. The classes are defined by the current wadgame configuration file.

Thing Type

This uniquely defines the appearance and behavior of a thing at runtime. The types are defined by the current wadgame configuration file.

Thing Attributes

These determine on which skill levels and in which modes of play the given thing will be present at runtime. The attributes are explained in the table below.

Attribute	Description
Level 1/2	This thing will be present on the lowest two skill levels of play.
Level 3	This thing will be present on the third skill level of play.
Level 4/5	This thing will be present on the highest two skill levels of play.
Deaf	This thing will not respond to sound; it will not attack the player until the player attacks or enters its field of vision. This attribute is only meaningful for enemies.
Multi	This thing will only appear in multi-player mode.

Thing Preview

An image of the currently selected thing will be displayed here. If no image is available, a large red circle with a diagonal line through it will appear instead. The images displayed are sprites extracted from the wadgame's main wadfile; the association of sprite name to thing type is made by the current wadgame configuration file. Clicking the image will display the [image browser dialog](#) to allow graphical selection of the desired thing type.



The images are zoomed to a maximum of three times their actual size or minimized to a minimum of half their actual size to best utilize the available screen real estate.

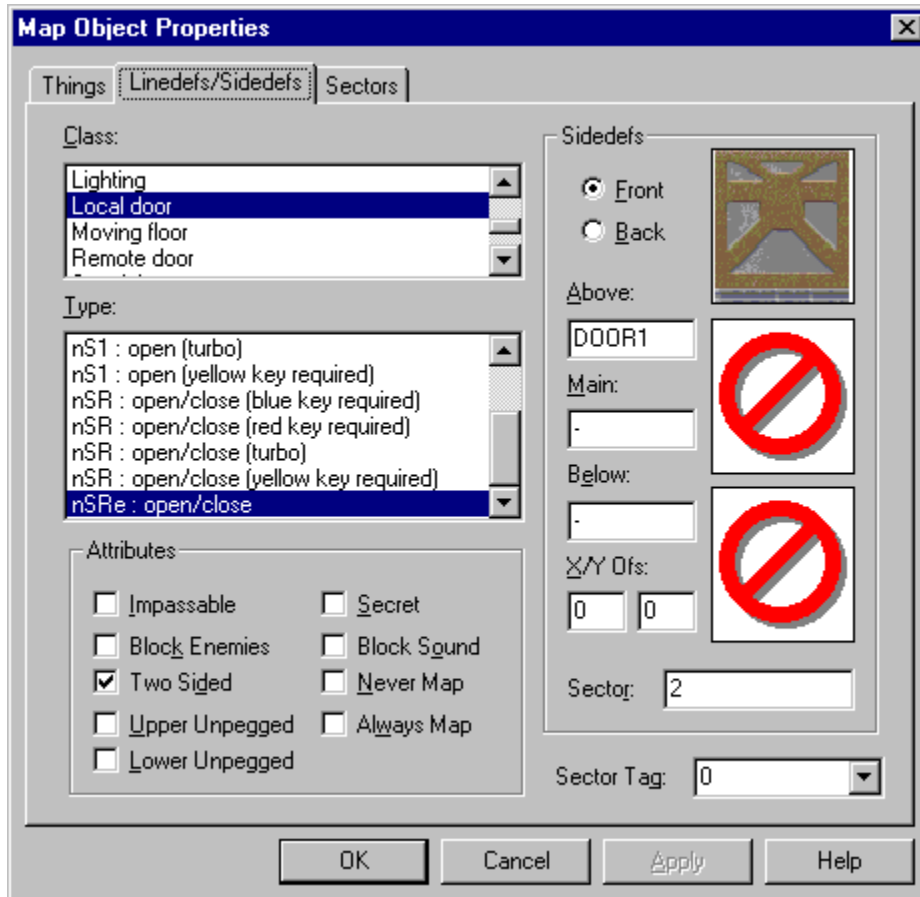
Thing Position And Facing

The Cartesian coordinates of the thing, along with the direction in which it faces are specified here. The coordinates are given in units, while the direction is given by selecting from a series of radio-buttons arranged roughly like the points of a compass.

Linedef Properties Dialog

{button ,KL(` Properties;Textures;Tags',0,`,`') } [Related Topics](#)

The map object properties dialog linedef page, shown in the illustration below, allows editing of multiple linedefs. You may click on the controls in the illustration below for more specific help.



Linedef Class

This is a somewhat arbitrary distinction made by WadAuthor to reduce the large number of linedef types to a manageable list. The classes are defined by the current wadgame configuration file.

Linedef Type

This uniquely defines any special behavior of the linedef at runtime. The types are defined by the current wadgame configuration file. The codes displayed at the beginning of the linedef type determine the linedef's method of activation and are explained in the following table.

<u>Code</u>	<u>Meaning</u>
n	No sector tag is required; this linedef provides a simple effect.
W	Linedef is activated when the player walks across the front side.
S	Linedef is activated when the player "uses" it.
G	Linedef is activated when the player shoots it.
1	Linedef can only be activated once.
R	Linedef can be activated repeatedly.
L	Changes to the affected sectors will be locked for the duration of game play.
e	Linedef can be activated by enemies.

Linedef Sector Tag

This associates a linedef or group of linedefs to a sector or group of sectors. When a linedef of a type that requires a sector tag number is triggered, all sectors having the same tag number are affected.

Linedef Attributes

These determine whether the linedef can be crossed and how and when it appears on the map. They also alter the manner in which images are displayed for the linedef's front and back sides. The attributes are explained in the table below.

<u>Attribute</u>	<u>Description</u>
Impassable	Neither players nor enemies can cross this linedef.
Block Enemies	Enemies cannot cross this linedef.
Two Sided	The linedef can be crossed in both directions.
Upper Unpegged	The above image is rendered top down instead of bottom up.
Lower Unpegged	The main and below images are rendered bottom up instead of top down.
Secret	The linedef will appear on the player's map in red. This is useful for protecting secret doors and other hidden features. This has nothing to do with the percentage of secrets found given at the successful conclusion of a map.
Block Sound	Sound will be blocked by the second linedef of this type encountered. This can be used to facilitate ambush situations and the like.
Never On Map	This linedef never appears on the player's map — even if the map powerup is acquired.
Always On Map	When the map is begun, this linedef will be on the player's map before the player has seen it. This is a good way to provide a clue towards what may lie ahead.

Linedef Sidedef Selector

These two radio-buttons control which sidedef's data, front or back, is available for modification.

Linedef Images

These [image list controls](#) determine how the linedef appears from the player's perspective. The name of the image used at runtime can be entered here.

Linedef Image Offsets

Offset values for the horizontal and vertical image rendering may be supplied here.

Linedef Image Previews

These preview controls display the image whose name appears in the corresponding image list control. Clicking a preview control will display the [image browser dialog](#) for the corresponding [image list control](#).



The images are zoomed to a maximum of three times their actual size or minimized to a minimum of half their actual size to best utilize the available screen real estate.

Linedef Sidedef Sector

Each valid sidedef must correctly reference the sector it faces. The sector number may be specified here to correct map errors or achieve special effects.

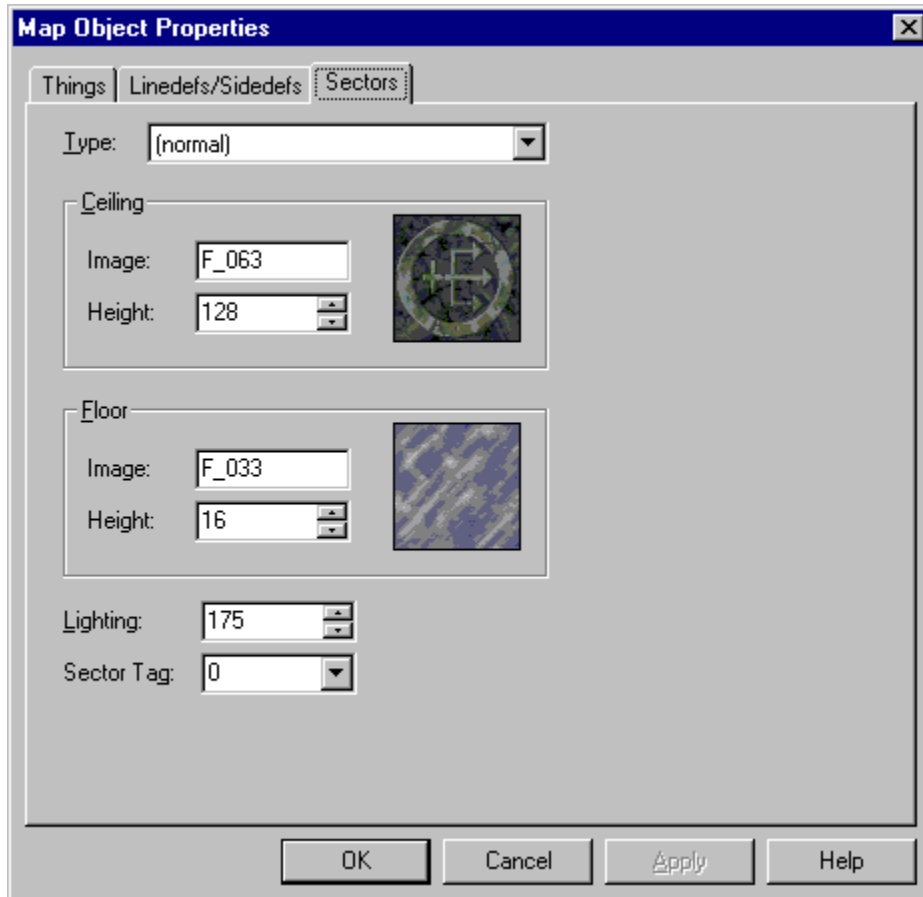


Entering invalid data can render the map unplayable. Please exercise caution when modifying the values displayed here.

Sector Properties Dialog

{button ,KL(` Properties;Textures;Tags',0,`,`')}` [Related Topics](#)

The map object properties dialog sector page, shown in the illustration below, allows editing of multiple sectors. You may click on the controls in the illustration below for more specific help.



Sector Type

This uniquely defines any special behavior of the sector at runtime. The types are defined by the current wadgame configuration file.

Sector Images

These [image list controls](#) determine how the sector's ceiling and floor appear from the player's perspective. The name of the image used at runtime can be entered here.

Sector Heights

These [relative change controls](#) set the ceiling and floor height for the sector.

Sector Image Previews

These preview controls display the image whose name appears in the corresponding image list control. Clicking a preview control will display the [image browser dialog](#) for the corresponding [image list control](#).

Sector Lighting

This [relative change control](#) determines the lighting level for the sector. The valid values are from 0 (completely dark) to 255 (excessively bright) and may be affected by the sector type.

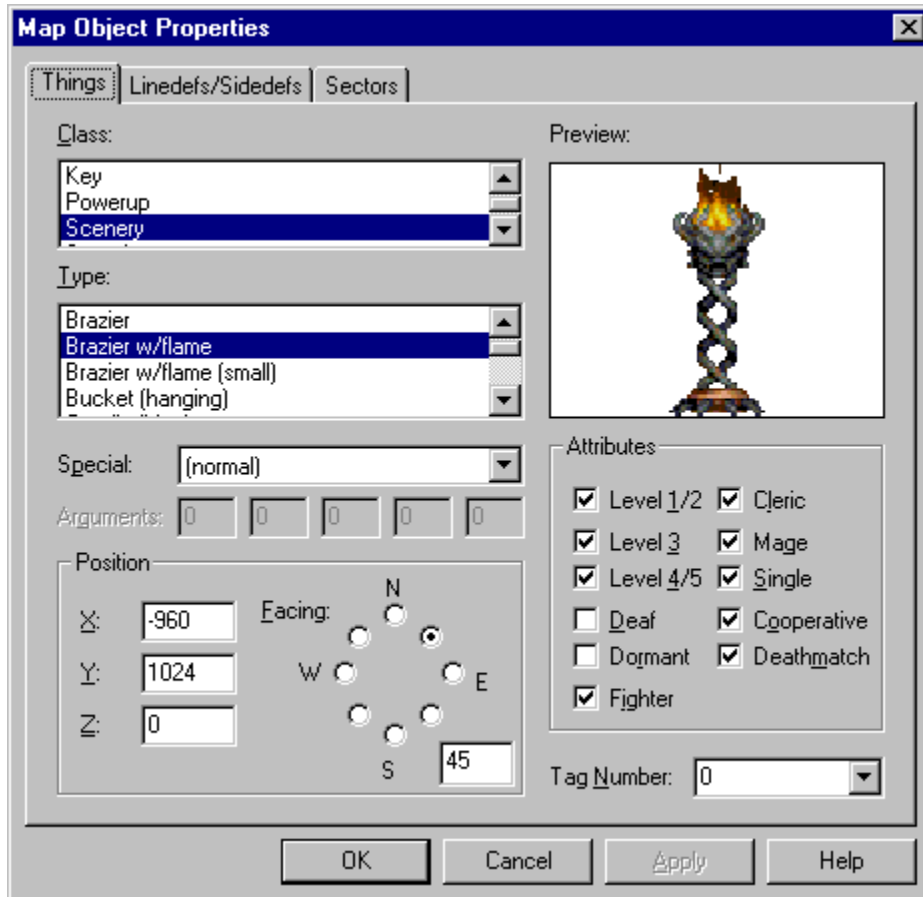
Sector Tag

This associates a sector or group of sectors to a linedef or group of linedefs. When a linedef of a type that requires a sector tag number is triggered, all sectors having the same tag number are affected.

Thing Properties Dialog (Hexen)

{button ,KL(` Properties;Textures',0,`,`') } [Related Topics](#)

The map object properties dialog thing page (Hexen variant), shown in the illustration below, allows editing of multiple things. You may click on the controls in the illustration below for more specific help.



Thing Attributes

These determine on which skill levels and in which modes of play the given thing will be present at runtime. The attributes are explained in the table below.

Attribute	Description
Level 1/2	This thing will be present on the lowest two skill levels of play.
Level 3	This thing will be present on the third skill level of play.
Level 4/5	This thing will be present on the highest two skill levels of play.
Deaf	This thing will not respond to sound; it will not attack the player until the player attacks or enters its field of vision. This attribute is only meaningful for enemies.
Dormant	This thing must be explicitly activated by a special. It will not function until then.
Fighter	This thing will be present when the user is playing as a fighter.
Cleric	This thing will be present when the user is playing as a cleric.
Mage	This thing will be present when the user is playing as a mage.
Single	This thing will be present when the user is playing a single-player game.
Cooperative	This thing will be present when the user is playing a co-operative multi-player game.
Deathmatch	This thing will be present when the user is playing a deathmatch multi-player game.

Tag Number

This identifier may be used to group things together for manipulation by specials.

Thing Facing Angle

To better support polyobject creation, the user may manually specify the facing angle directly within this field. Polyobject anchors and starting positions are paired by supplying matching polyobject numbers in the thing angle field. Click here for more information about [Understanding Polyobjects](#).

Linedef Properties Dialog (Hexen)

{button ,KL(` Properties;Textures;Tags',0,`,`)} [Related Topics](#)

The map object properties dialog linedef page (Hexen variant), shown in the illustration below, allows editing of multiple linedefs. You may click on the controls in the illustration below for more specific help.



Special Type

The special type specifies the action that will be initiated when the map object is activated.

Special Arguments

The special arguments are parameters passed to the special execution. Each argument may range in value from 0 to 255. Unused arguments will be disabled, preventing the user from changing them.

Activate

The activate control allows the user to specify under what condition the linedef is activated.

Linedef Attributes

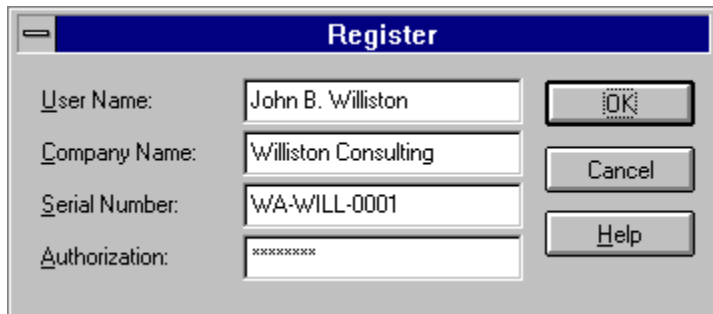
These determine whether the linedef can be crossed and how and when it appears on the map. They also alter the manner in which images are displayed for the linedef's front and back sides. The attributes are explained in the table below.

<u>Attribute</u>	<u>Description</u>
Impassable	Neither players nor enemies can cross this linedef.
Block Enemies	Enemies cannot cross this linedef.
Two Sided	The linedef can be crossed in both directions.
Upper Unpegged	The above image is rendered top down instead of bottom up.
Lower Unpegged	The main and below images are rendered bottom up instead of top down.
Secret	The linedef will appear on the player's map in red. This is useful for protecting secret doors and other hidden features. This has nothing to do with the percentage of secrets found given at the successful conclusion of a map.
Block Sound	Sound will be blocked by the second linedef of this type encountered. This can be used to facilitate ambush situations and the like.
Never On Map	This linedef never appears on the player's map — even if the map powerup is acquired.
Always On Map	When the map is begun, this linedef will be on the player's map before the player has seen it. This is a good way to provide a clue towards what may lie ahead.
Repeatable	Is the linedef special (if any) a repeatable?

Register Dialog

{button ,KL(`Registration',0,`,`')} [Related Topics](#)

The register dialog, shown in the illustration below, allows the user to update the software license. Upon payment for the software, the required information for completing the dialog box will be made available. Simply enter the information into the dialog fields and press the *OK* button to authorize your copy of WadAuthor.



The image shows a Windows-style dialog box titled "Register". It contains four text input fields and three buttons. The fields are labeled "User Name:", "Company Name:", "Serial Number:", and "Authorization:". The "User Name" field contains "John B. Williston", "Company Name" contains "Williston Consulting", and "Serial Number" contains "WA-WILL-0001". The "Authorization" field contains a series of asterisks. The buttons are labeled "OK", "Cancel", and "Help".

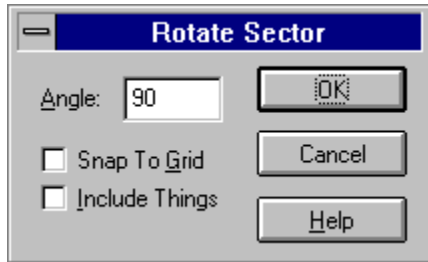
Field Label	Value
User Name:	John B. Williston
Company Name:	Williston Consulting
Serial Number:	WA-WILL-0001
Authorization:	*****



If the registration dialog will not accept the information, an error has probably occurred during entry. Check the data very carefully, making sure the case is correct (the data is case-sensitive), making sure you're not typing zeroes for the letter 'O', etc.

Rotate Sector Dialog

The rotate sector dialog, shown in the illustration below, allows the user to specify the angle, in degrees, by which the selected sector(s) should be rotated. Positive values cause counter-clockwise rotation; negative values cause clockwise rotation.



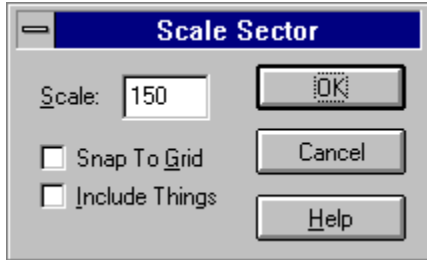
The *Snap To Grid* option, when checked, causes WadAuthor to snap each vertex of the selected sector(s) to the current grid after performing the operation. The *Include Things* option, when checked, causes WadAuthor to preserve the positioning of things within the selected sector(s).



The *Include Things* option can greatly increase the time required for the operation.

Scale Sector Dialog

The scale sector dialog, shown in the illustration below, allows the user to specify the percentage by which the selected sector(s) should be scaled. For example, a value of two-hundred percent would double the sector size(s).



The *Snap To Grid* option, when checked, causes WadAuthor to snap each vertex of the selected sector(s) to the current grid after performing the operation. The *Include Things* option, when checked, causes WadAuthor to preserve the positioning of things within the selected sector(s).

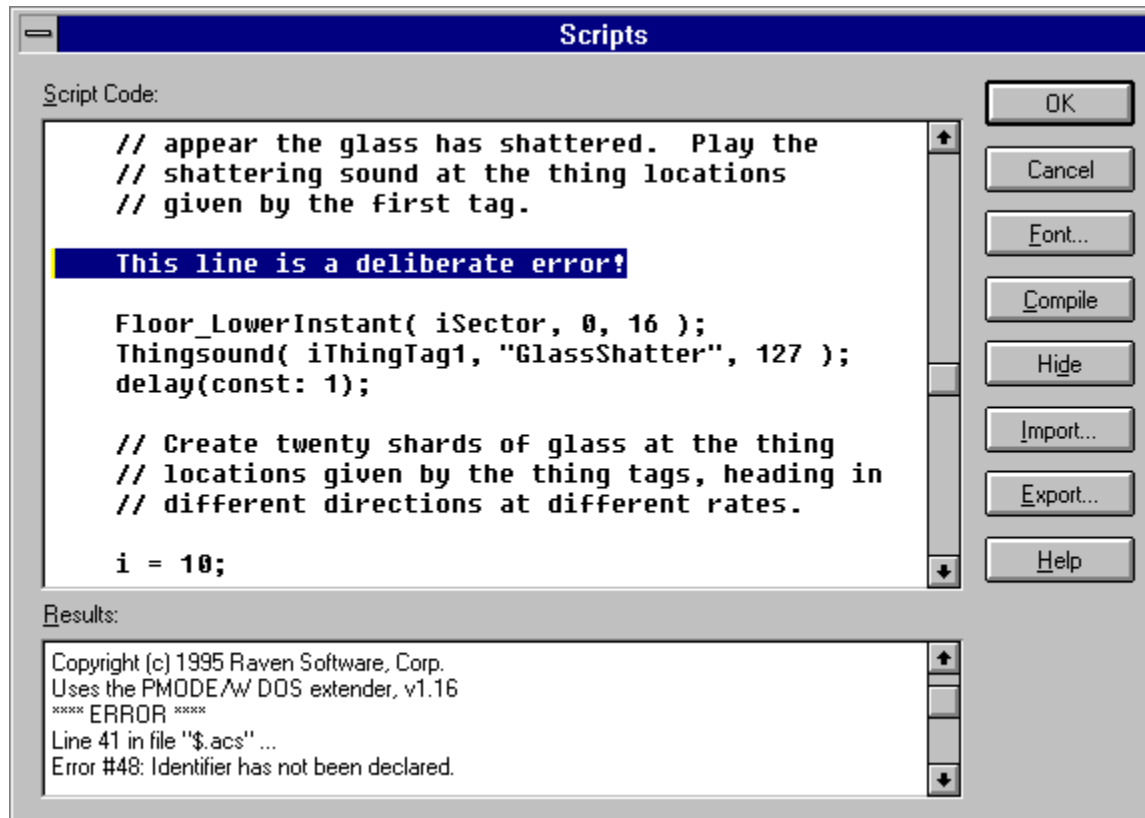


The *Include Things* option can greatly increase the time required for the operation.

Scripts Dialog

{button ,KL(' Scripts',0,``,``)} [Related Topics](#)

The scripts dialog, shown in the illustration below, allows the user to develop and compile script code for use with Hexen-style wadgames. You may click on the controls in the illustration below for more specific help.



Script Code

The script code is entered and editing here. If errors are detected during a compile, they are highlighted as shown in the illustration.

Results

If errors are detected during a compile, the results are displayed here to allow the user to quickly diagnose and correct the error condition.

Font Button

The font button, when pressed, will invoke the font selection common dialog box to allow the user to select a fixed-pitch font for displaying the script code. The system font is used by default.

Compile Button

The compile button, when pressed, compiles the script code as it currently exists above. If errors are detected, the results will be displayed and the offending line highlighted (if possible).

Show/Hide Button

This button, when pressed, toggles the state of the results window, showing or hiding it as appropriate.

Import Button

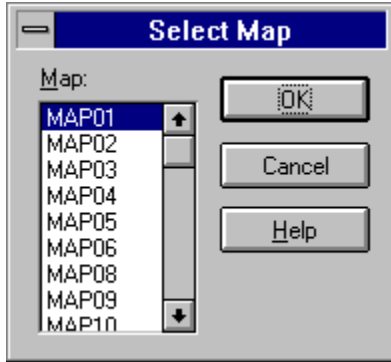
This button, when pressed, allows the user to specify a filename from which the script code will be loaded.

Export Button

This button, when pressed, allows the user to specify a filename to which the script code will be saved.

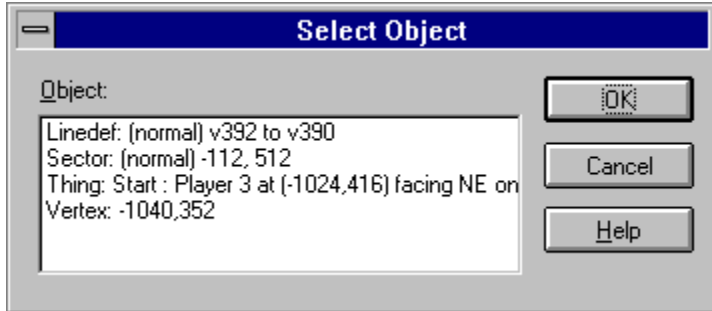
Select Map Dialog

The select map dialog, shown in the illustration below, allows the user to select which map within a wadfile to edit. Obviously, this only applies to wadfiles containing more than a single map.



Select Object Dialog

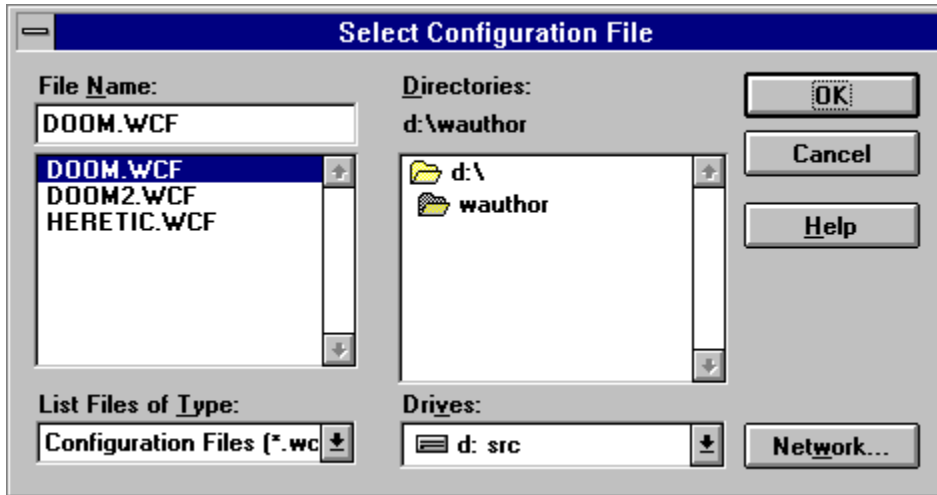
The select object dialog, shown in the illustration below, allows the user to disambiguate a mouse click within a densely populated section of the current map. Select a single object from the list and press *OK* to continue the current operation, or press the *Cancel* button to abort.



Select Configuration File Dialog

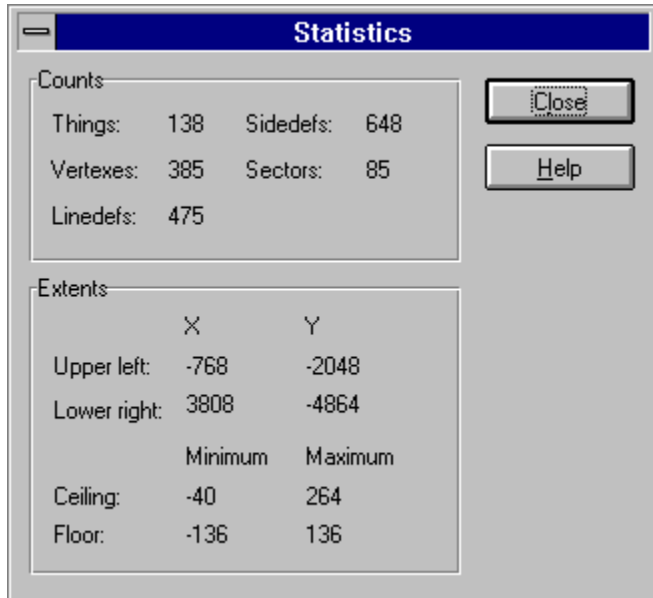
{button ,KL(` Configuration Files',0,`,`')} [Related Topics](#)

WadAuthor supports different wadgames by using different wadgame configuration files. At startup, if WadAuthor cannot locate a valid wadgame configuration file, the dialog box shown below will be invoked to allow the user to select one. This same dialog is also used when the user selects a different wadgame configuration file during normal operation. If some or all of the required wadgame information is invalid, the wadgame configuration dialog will be invoked to allow easy editing.



Statistics Dialog

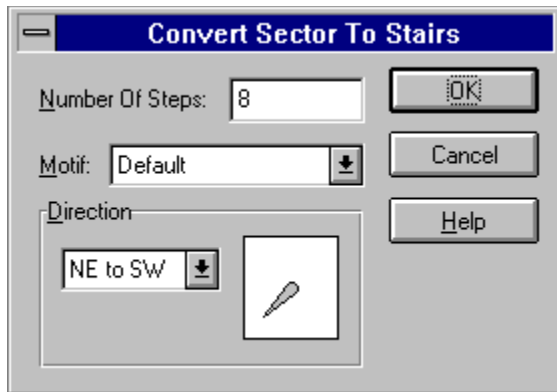
The statistics dialog, shown in the illustration below, provides some useful information about the current map. The counts supply the total number of each object type in the current map. The extents supply the map rectangle in x-axis and y-axis coordinates, along with the minimum and maximum floor and ceiling heights in z-axis coordinates.



Convert Sector To Stairs Dialog

{button ,KL('Stairs',0,'')} [Related Topics](#)

The convert sector to stairs dialog, shown in the illustration below, prompts the user for information when converting a sector to stairs. The number of steps determines the number of steps into which the sector will be divided. The motif contains the floor and ceiling height changes, the lighting change, and the images used for the stairwell and runners. The direction specifies the path along which the stairs should be created.

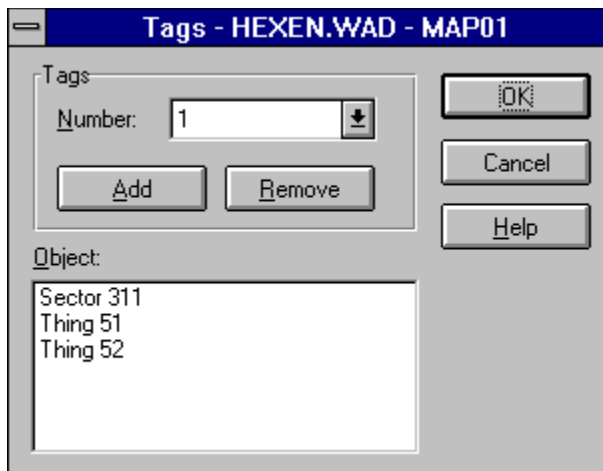


Tags Dialog

{button ,KL(`Tags',0,`,`')}` [Related Topics](#)

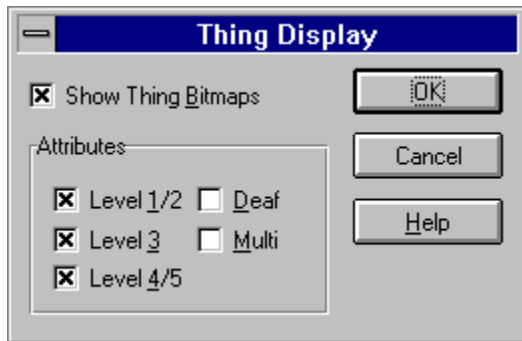
The tags dialog, shown in the illustration below, allows the user to examine and modify the existing tag relationships between linedefs, sectors, and things (when working with Hexen maps). Pressing the *Add* button will add a new tag to those available, and pressing the *Remove* button will remove the current tag entirely. Clicking on the objects in the listbox will highlight them within the map. Clicking on object on the map will add or remove them to or from the listbox.

When working with Hexen maps, it is important to understand that not all linedefs can be tagged. WadAuthor will only allow you to tag linedefs whose special type requires a tag. In this respect, linedefs are different from sectors and things which may be freely tagged.



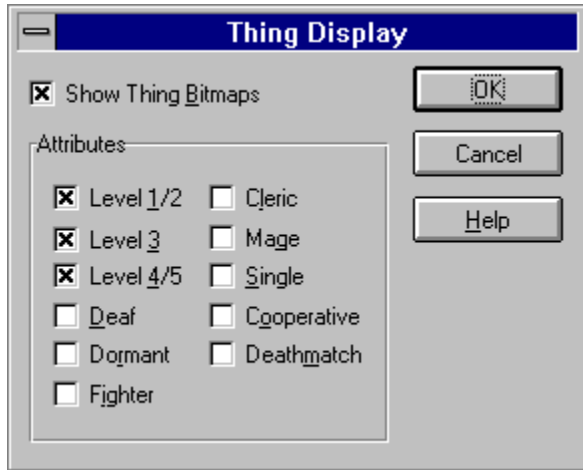
Thing Display Dialog

The thing display dialog, shown in the illustration below, allows the user to configure how things are displayed and which things are displayed. The *show thing bitmaps* checkbox determines whether or not bitmap images will be used when possible. The attribute checkboxes specify the attributes a thing must have in order to be displayed.



Thing Display Dialog (Hexen)

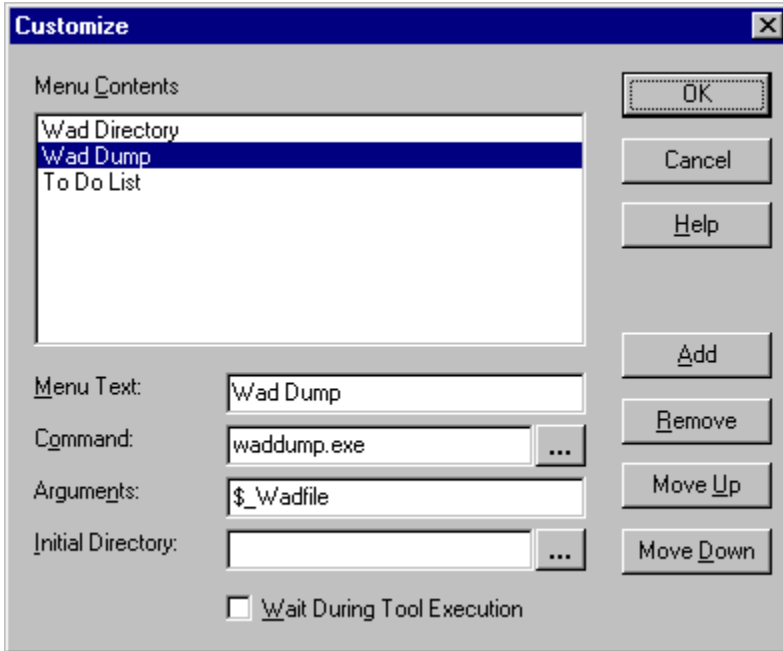
The thing display dialog, shown in the illustration below, allows the user to configure how things are displayed and which things are displayed. The *show thing bitmaps* checkbox determines whether or not bitmap images will be used when possible. The attribute checkboxes specify the attributes a thing must have in order to be displayed.



User Tools Dialog

{button ,KL(`Customize',0,`,`')}` [Related Topics](#)

The user tools dialog, shown in the illustration below, allows the user to configure the entries appended to the *Tools* menu. You may click on the controls in the illustration below for more specific help.



The following list of replaceable parameters may be used within the command, arguments, and initial directory fields.

Parameter	Meaning
\$_Comspec	Command shell executable
\$_Appdir	The directory from which WadAuthor is executing
\$_LAppdir	The long filename of the directory from which WadAuthor is executing.
\$_Dir	The current working directory
\$_LDir	The long filename of the current working directory
\$_Wadfile	The filename containing the current map
\$_LWadfile	The long filename containing the current map
\$_Mapname	The name of the current map ("MAP01" for example)
\$_Wadmap	The warp parameter of the current map ("01" for example)

Tools List

The user-defined tools are listed here as they will appear when appended to the *Tools* menu.

Menu Text

The text used to identify the tool should be entered here. This field cannot be blank.

Command

The command line to execute for the tool should be entered here. Pressing the button to the immediate right will invoke a browser dialog, allowing the user to navigate the directory tree selecting from a list of available programs.

Arguments

The arguments to be passed to the tool should be entered here.

Initial Directory

The directory from which the tool should be launched. Pressing the button to the immediate right will invoke a browser dialog, allowing the user to navigate the directory tree.

Add

Adds a new tool to the list when clicked.

Remove

Removes the currently selected tool from the list when clicked.

Move Up

Moves the current selected tool up one slot when clicked.

Move Down

Moves the current selected tool down one slot when clicked.

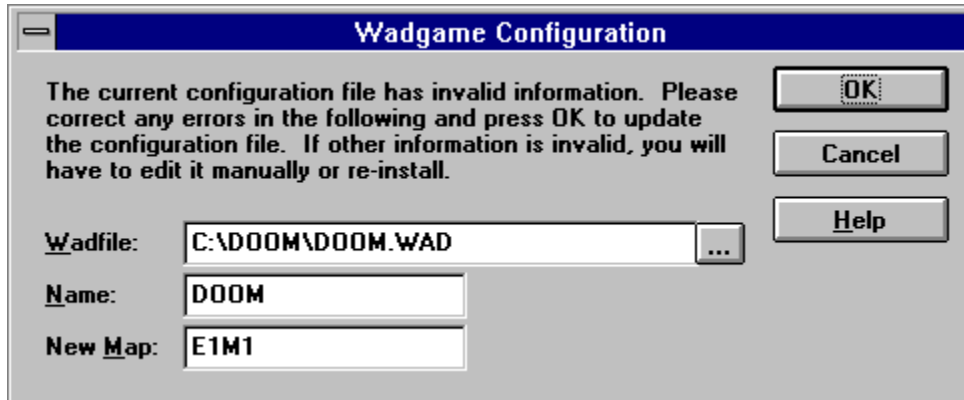
Wait During Tool Execution

When checked, WadAuthor will wait until the tool has finished executing before allowing any further editing to occur.

Wadgame Configuration Dialog

{button ,KL(' Configuration Files',0,'','')} [Related Topics](#)

If any of the required wadgame information in the current configuration file is invalid, this dialog will be presented on startup. The user must supply correct information for the wadgame for which maps are to be created or edited before continuing. Pressing the *Cancel* button will invoke the configuration file dialog, allowing you to choose a different configuration file altogether. You may click on the controls in the illustration below for more specific help.



Wadfile

The wadfile specifies the main data file for the wadgame. It is from this file that the images displayed during editing are extracted.



The button with the ellipses is for browsing the files on your drive(s). Clicking it allows you to navigate your directory structure to select a wadfile.

Name

The name field identifies the game for which the [wadfile](#) is intended.

New Map

The new map specifies the default name used when creating new maps. It is based on the wadgame for which the map is intended. For example, DOOM uses the *EeMm* naming convention for maps where *e* is the episode number and *m* is the mission number (i.e. episode one mission seven will have a map name of "E1M7").

Example Wadfiles

The Hexen wadgame introduces so much new power for the wad designer, that I thought it necessary to include some example wadfiles. The following list briefly describes the examples that come with WadAuthor, providing links to the appropriate help topics.

EX_GLASS.WAD

The purpose of this wadfile is to demonstrate use of the breaking glass script that WadAuthor uses by default. Click here for help understanding the [Breaking Glass](#) script. This wadfile is designed only for Hexen and will not work with any previous wadgame.

EX_POLYO.WAD

The purpose of this wadfile is to demonstrate use of a simple polyobject that provides swinging doors. Click here for help [Understanding Polyobjects](#). This wadfile is designed only for Hexen and will not work with any previous wadgame.



Getting Started

If you are here because you want to get a new map created as quickly as possible, click the *Tutorial* button above for a step by step walkthrough. Otherwise, I suggest you take a few moments to peruse the following topics first.

Using WadAuthor

[Basic Vocabulary](#)

[Understanding Wadfiles](#)

[Understanding Wadmaps](#)

[Creating New Objects](#)

[Selecting Objects](#)

[Moving Objects](#)

[Editing Existing Objects](#)

[Deleting Objects](#)

[Making Things Happen](#)

[Understanding Polyobjects](#)

[Understanding Scripts](#)

[Understanding Texture Mapping](#)

Common Questions

[Why Does The "Hall Of Mirrors" Effect Happen?](#)

[Why Can't I Just Draw Lines On The Map?](#)

[Why Won't WadAuthor Run With A Sixteen Color Video Driver?](#)

[Why Is There No Sound When Running A Map?](#)

[Can I Customize The Game Graphics?](#)



Important reminders, tips, and other useful bits of information are set off by the graphic used in this example note. Make sure you read items with this designation carefully.

Basic Vocabulary

It is important to establish a minimum working vocabulary to use WadAuthor effectively. The terms you need to understand before proceeding are summarized in the following table. Consult the glossary for a more comprehensive list.

Term	Definition
Linedef	A linedef defines a wall within a map. A linedef has a starting vertex, an ending vertex, and various other information.
Map	A map is stored within a wadfile as a well-defined list of resources. Each map corresponds to a single "level" or "mission" when played.
Node building	When a map is saved, quite a bit of information must be generated before the map can be used in a game. This information allows the game engine to quickly choose which walls need to be drawn for the player, avoiding time intensive computations at run-time. This process is referred to as node building.
Sector	A sector defines a room within a map. It contains lighting conditions, floor and ceiling heights, and other information.
Sidedef	A sidedef defines how a wall appears from a given side. Each linedef may have one or two sidedefs associated with it based on whether the player can ever see the back side.
Thing	A thing defines a single object within a map. Most things interact with the player in some way, although some are strictly scenery. A few examples are weapons, ammunition, and enemies.
Vertex	A vertex defines a single point within a map via Cartesian coordinates. For those who don't remember geometry, a Cartesian coordinate is a simple pair of orthogonal values usually denoted by (x,y).
Wadfile	A data file intended for use with several games from id software. A wadfile consists of a header, resources, and a resource directory.
Wadgame	Wadgame is a term I have coined to refer to any game utilizing a wadfile. A few examples are DOOM, DOOM][, and HERETIC.

Understanding Wadfiles

A wadfile is simply a data file that adheres to the same format as the original DOOM.WAD file released with DOOM. A wadfile must have a header, a number of resources, and a directory. Presumably, wadfiles were given their name because they provide a simple way to store "wads of stuff." The header format, resource format, and directory format are described in the sections that follow. Non-technical types may wish to skip to the next topic; all you really need to know to use WadAuthor effectively is that maps are stored within wadfiles.

Header

At the beginning of every wadfile, a twelve-byte header structure contains a four-byte signature. The signatures of which I am aware are listed in the following table.

Signature	Description
IWAD	Possibly short for "internal wadfile", this denotes a main wadfile designed for use with a wadgame. The aforementioned DOOM.WAD is an example of an IWAD wadfile.
PWAD	Possibly short for "patch wadfile", this denotes an add-on wadfile. All new wadfiles created by WadAuthor are initialized with this signature.
TWAD	Short for template wadfile, this signature was introduced by DoomCAD, another wad editing application for Windows.

Following the four-byte signature is a four-byte number which provides the number of directory entries in the wadfile directory. The remaining four bytes is a number which provides the offset, from the beginning of the wadfile, at which the directory exists.

Resources

The resources within a wadfile exist in many formats, depending upon their purpose. For example, map data, image data, sound data, and textual data are all stored differently. Because of this, I will offer no further explanation here. For those interested in learning more about the various resources, I would suggest that you visit various internet sites or other on-line services for more detail.

Directory

A wadfile directory consists of a number of directory entries; the exact number, as explained above, is given by the wadfile header. Each directory entry is a sixteen-byte structure. The first four bytes is a number providing the offset, from the beginning of the wadfile, at which the resource exists. The second four bytes is a number providing the size, in bytes, of the resource. The remaining eight bytes provides a textual name for the resource.

Understanding Wadmaps

A single wadmap corresponds to a single mission or level within a given wadgame. Wadmaps are stored within a given wadgame's main wadfile until needed at runtime. The required wadmap resources and geometry are explained in the sections that follow.

Resources

A wadmap is defined by a fixed series of wadfile resources. Their names, listed in the order in which they appear in the wadfile directory, and a brief description of their purpose are given in the following table.

Name	Description
<i>Mapname</i>	The map naming conventions vary between different wadgames. For example, DOOM uses the <i>EnMn</i> convention replacing the <i>n</i> 's with the episode and mission numbers, while DOOM][uses the <i>MAPnn</i> convention, replacing the <i>n</i> 's with the two-digit mission number.
THINGS	The things resource contains the thing data for the map.
LINEDefs	The linedefs resource contains the linedef data for the map.
SIDEDefs	The sidedefs resource contains the sidedef data for the map.
VERTEXES	The vertexes resource contains the vertex data for the map.
SEGS	The segs resource is used to define ssectors.
SSECTORS	The ssectors resource divides all of the sectors into convex polygons.
NODES	The nodes resource contains branches in a binary space partition (BSP) that partition the map. It is used to determine which walls are in front of others.
SECTORS	The sectors resource contains the sector data for the map.
REJECT	The reject resource contains a two-dimensional matrix that determines whether enemies in a given sector can detect or attack players in another sector.
BLOCKMAP	The blockmap resource contains a series of tables that simplify collision detection between walls and moving objects.
BEHAVIOR	The behavior resource is only present in maps created for Hexen. It contains compiled script data for the map.
SCRIPTS	The scripts resource is only present in maps created for Hexen and is optional. WadAuthor uses this resource to maintain the script code for a given map. This resource is a proposed standard accepted by some wad editors.

WadAuthor maintains the things, linedefs, sidedefs, vertexes, and sectors resources for a given map in memory. It is only when the map is saved that the segs, ssectors, nodes, reject, and blockmap resources are generated on disk. Because the node building process can be quite lengthy for large maps, WadAuthor only does it when necessary.

You can use this information to your advantage by approaching map construction in two separate phases: creating and modifying. If you complete your vertex-related editing chores during the creating phase, saving your map during the modifying phase will be much quicker. In general, any time you add a new vertex or delete or move an existing vertex, your next save will incur the time hit for a complete node rebuild.

Geometry

You remember geometry, right? Well, don't worry, WadAuthor takes care of virtually all of the geometry involved in creating new maps. It is important, however, that you understand how the wadmap resources relate to each other; otherwise, you risk creating a map that cannot be played due to one or more architectural errors. The following list provides a geometric overview of the relevant map object types.

- n **Vertexes** define locations and are essentially points in the Cartesian plane.
- n **Linedefs** are strung between vertexes and are essentially line segments.
- n **Sidedefs** define how the front and back of a linedef appears from the player's perspective. They also determine which sectors the front and back of a linedef face. They have no geometric equivalent, true, but they do provide the crucial link between linedefs and sectors.
- n **Sectors** define some of the attributes of a "room" and are essentially polygons.

These simple relationships have some very important implications for budding map designers. Some of the most important points to grasp are listed below.

- n Deleting a vertex will delete any linedefs that reference it.
- n A linedef can be one-sided or two-sided.
- n A one-sided linedef is essentially a brick wall; it cannot normally be crossed and only has a front side. A one-sided linedef that has its back side facing the sector with which it is associated is going to produce the hall of mirrors effect at runtime.
- n A two-sided linedef can normally be crossed. Because the sectors faced by its front and back sides can have different floor and ceiling heights, great care must be taken in specifying the below, main, and above images to avoid the hall of mirrors effect at runtime.
- n Deleting a linedef can leave a sector open. All sectors must be closed polygons for the node building process to work.
- n Deleting a sector affects its associated linedefs and vertexes. One-sided linedefs are deleted, two-sided linedefs become one-sided, and any vertexes no longer in use are deleted.
- n WadAuthor is doing an awful lot of work for you behind the scenes!

Creating New Objects

{button ,KL(' New Objects',0,'','')} [Related Topics](#)

WadAuthor allows you to create two basic types of objects: sectors and things. Sectors may be created anywhere within a map, while things must be created within an existing sector. Sectors created in an unused portion of the map have one-sided linedefs, separating them from the rest of the map. Sectors created within other sectors have two-sided linedefs, acting as sub-sectors through which the player can freely travel. Step by step instructions follow.

Creating A New Thing

- n Right-click within an existing sector where the thing should be placed.
- n Select the *New Thing..* option from the context menu.
- n Enter the desired parameters in the resulting dialog box and press the *OK* button to complete the new thing, or press the *Cancel* button to abort.

Creating A New Rectangular Sector

- n Right-click anywhere on the map roughly where the sector should be centered.
- n Select the *New Rectangular Sector* option from the context menu.
- n Enter the desired parameters in the resulting dialog box and press the *OK* button to complete the new sector, or press the *Cancel* button to abort.



The vertexes of the new sector will be snapped to grid.

Creating A New Polygonal Sector

- n Right-click anywhere on the map roughly where the sector should be centered.
- n Select the *New Polygonal Sector...* option from the context menu.
- n Enter the desired parameters in the resulting dialog box and press the *OK* button to complete the new sector, or press the *Cancel* button to abort.



The vertexes of the new sector will not be snapped to grid. This step is omitted to preserve the most accurate shape for the given number of sides and radius.

Selecting Objects

{button ,KL(`Interface',0,`,`')} [Related Topics](#)

WadAuthor provides several ways to select objects. Most of them require setting the current object and pressing a key or clicking with the mouse. For more information, consult the related topics.



When a sector is selected, all of the linedefs associated with it are also selected. This is mainly to provide some kind of visual feedback to the user, although it is also useful for various editing operations.

Moving Objects

{button ,KL(`Interface',0,`,`')} [Related Topics](#)

WadAuthor supports the drag and drop interface found in many Windows programs. To move an object or group of objects, select them, then click and hold the primary mouse button on one of the selected objects. Move the mouse and use the helpful wire-frame display WadAuthor supplies to position the objects at the correct final location. Release the mouse button to complete the move operation, or press the escape button at any time to abort the procedure.



When the snap to grid feature is enabled, WadAuthor allows the user to temporarily disable it during a drag operation by holding down the shift key.

Editing Existing Objects

{button ,KL(`Interface:Properties',0,`,`')} [Related Topics](#)

Editing existing objects is a two step process of selection and manipulation. Whenever possible, WadAuthor supports direct manipulation of an unlimited number of objects of varying types. The most frequently useful editing operations are available through the context menu. Once the desired objects are selected, you may manipulate them. The following list describes some of the most useful operations made available by the context menu. For more information, consult the related topics.

- n **Flipping a linedef** can reverse only the starting and ending vertexes or reverse the sidedef data as well. Avoid this feature unless you know what you're doing!
- n **Splitting a linedef** (or linedefs) inserts a new vertex at the center of the line. This allows the user to quickly reshape a given sector as desired.
- n **Joining two linedefs** is a fast means for connecting two sectors; select the first linedef, select the linedef to which it should be joined, and select the join option from the context menu. The first line will "move" to join the second.
- n **Joining two vertexes** is a fast means for getting rid of a linedef or fixing a problem with an un-closed sector. Select the first vertex, select the second vertex to which it should be joined, and select the join option from the context menu. The first vertex will "move" to join the second.
- n **Creating a door** manually is difficult at best; using WadAuthor is much simpler. Select any single four-sided sector that connects two other sectors and select the door conversion option from the context menu.
- n **Scaling or rotating a sector** usually implies moving each associated vertex manually. Select any single sector, and select the desired option from the context menu.
- n **Creating a staircase** is a lot of work; using WadAuthor is much simpler. Select any single four-sided sector and select the stair conversion option from the context menu.
- n **Editing object properties** is probably the most common operation. Select any number of objects and select the *Properties...* option from the context menu.

Deleting Objects

{button ,KL('Check Map Dialog',0,`,`,'')} [Related Topics](#)

Deleting objects deserves its own help topic because of the relationships between the different object types. Deleting the wrong objects can render a map unplayable, and require you to delete entire regions of the map to fix the problem. So, it is important that you understand the ramifications of deleting something. The following sections provide more information.

Cascade Effect

Deleting a single object can cause many other objects to be deleted, and I call this the cascade effect. For example, a linedef requires two vertexes, so deleting a vertex will necessitate deleting any linedefs that are attached to it. Sectors are made up of linedefs (actually they're referenced by sidedefs which are referenced by linedefs), so if deleting a vertex results in the deletion of any remaining linedefs in a given sector, the sector will also be deleted.



WadAuthor properly cascades a delete operation for you. For any objects that you delete, WadAuthor will make sure that any orphaned objects are similarly removed.

Corruption

It is all too easy a thing to corrupt a map. The most common source of problems is from deleting something that needs to be there. For example, a novice might see the linedef between two connecting sectors and figure it shouldn't really be there — after all, the player can't walk through walls, right? If you suspect that a delete operation may have corrupted your map, use the map checking tool *before saving the map* to find out for sure. If you have open sectors or other errors of that nature, you should probably undo the delete operation.

Making Things Happen

{button ,KL(`Tags',0,`,`')} [Related Topics](#)

The wadgame environment is a dynamic one; that is, architectural features can change over time independent of or in response to a variety of conditions. In every case, however, achieving a desired effect is a matter of configuring linedefs and sectors. The following sections group these dynamic features into three categories: sector effects, simple linedef effects, and complex linedef effects.

Sector Effects

Depending on the wadgame, different sector types can be used to achieve a several effects. Flickering lights, floors that damage the player, and ceilings that drop or rise are common across all wadgames, while only the most recent allow the map designer to express ideas like moving water or wind. Sector-based effects are achieved by selecting one of the non-normal sector types from the sector properties dialog.

Simple Linedef Effects

Simple linedef effects are those which do not require the use of tag numbers. Examples of these include horizontally scrolling wall images and local doors. To achieve a simple linedef effect, select the desired linedef type from the linedef properties dialog — that's it. Granted, this isn't much of an explanation, but then these are supposed to be simple...

Complex Linedef Effects

Complex linedef effects are those which require the use of tag numbers. Examples of these include remote doors, lifts, et. al. Achieving a complex linedef effect is a matter of choosing the linedefs that trigger it, choosing the sectors it should affect, setting the tag numbers to match, and enjoying the results. For a step by step example on creating a remote door, check the related topics.

The Hexen wadgame adds a new realm of possibilities for wadfile design. By avoiding the use of a fixed set of codes for linedef actions in favor of a system of specials and scripts, there is little the wadfile designer cannot achieve. This comes, of course, at the expense of simplicity.

Understanding Texture Mapping

{button ,KL('Textures',0,``,``)} [Related Topics](#)



Understanding wadgame texture mapping is, perhaps, the most difficult concept involved in creating good maps. More specifically, understanding how to achieve the desired texture alignment is somewhat complicated. The following sections explain wadgame texture mapping and the factors that affect it. For purposes of illustration, I have created a default texture, 128 units square, shown to the left. I will use this texture in all my examples. Because the following explanation is fairly lengthy, you may click on any of the sections listed below to quickly jump to that section.

Sections

[Sidedef Textures - Basics](#)

[Sidedef Textures - Main](#)

[Sidedef Textures - Below And Above](#)

[Sidedef Textures - Manual Adjustments](#)

[Sidedef Textures - Common Applications](#)

[Sector Textures](#)

Sidedef Textures - Basics

Every sidedef, for purposes of texture rendering, is broken into three distinct areas: the area below the sector's floor, the area between floor and ceiling, and the area above the ceiling. Each area may have a different texture name assigned to it. If no texture is needed, a hyphen is used for the null texture. The texture names cannot be left blank — this will cause an error at runtime. WadAuthor refers to the textures used for these areas as the *Below*, *Main*, and *Above* textures.

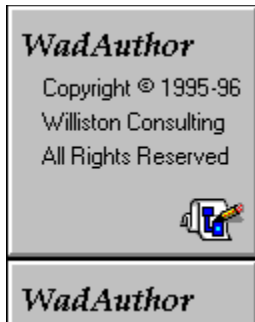
Wadgames render each area by tiling its texture horizontally and vertically until the entire area is covered. The texture will be truncated in either direction if the area dimensions are not an even multiple of the dimensions of the texture.



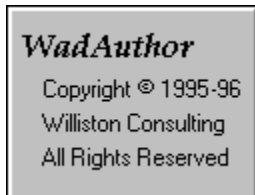
Textures whose vertical dimension is anything other than 64 or 128, stair runners for example, do not always tile well vertically. They sometimes result in small pink or multi-colored "cracks" appearing between the tiling. This is due to a bug in the rendering engine.

Sidedef Textures - Main

The vertical axis is handled differently for the three areas. Let's begin with the main area, since it is the most commonly used. By default, the main texture is mapped from top to bottom. This can cause problems if the overall sector height, the difference between floor and ceiling, is not equivalent to the texture's vertical size. The following illustrates two possible cases.



Here the sector is taller than the texture. Notice how the texture is tiled. If the texture had some distinctive feature at its bottom end that was designed to meet the floor, we'd have a real problem.



Here the sector is shorter than the texture. Notice how the texture is truncated. In this case, a distinctive feature at the bottom end of the texture would be truncated.

As you can see, potentially undesirable results occur in both cases. Fortunately we have a way to modify the rendering behavior. Linedefs have two attributes that change the manner in which textures are vertically rendered. The *lower unpegged* attribute applies to the main and below textures, and the *upper unpegged* attribute applies to the above texture. The lower unpegged attribute will cause the main and below textures to be reverse rendered, from bottom to top. The following table illustrates the previous cases with the lower unpegged attribute set.



Notice how setting the lower unpegged attribute has made the bottom of the texture meet the floor. This can be particularly important when working with high ceilings.



In this case, the aforementioned distinctive feature would still appear connected to the floor. Our player may be crouching, but at least the walls look good...

As you can see, the texture is now being mapped from bottom to top. Setting the lower unpegged attribute can quickly solve most main texture alignment problems between sectors with the same floor height but different ceiling heights.



Setting the main texture to something other than the null texture can provide an interesting effect with a two-sided linedef. By setting one side of a linedef's main texture to null and the other side to something else, you effectively create a wall that isn't. It appears to be solid from one side, but is transparent from the other. Imagine the player's surprise when something big and nasty comes tromping through the wall after him!

Sidedef Textures - Below And Above

The issue is further complicated by the below and above textures. A linedef may be two-sided, having different sectors on either side. If the floor and ceiling heights of the first sector match that of the second, there's no problem. When they differ, the below and above textures become important. The below texture will be required when looking into a sector whose floor height is greater than that in which the player is standing. The above texture will be required when looking into a sector whose ceiling height is less than that in which the player is standing.



If the ceiling texture of both sectors is F_SKY1, the upper textures between them will always be invisible. This is, apparently, a special case to allow map designers to include outdoor scenes.

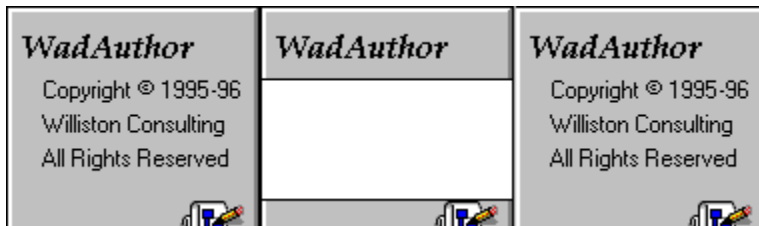


By default, the below texture is mapped from top to bottom, while the above texture is mapped from bottom to top. If this seems fairly innocent, consider the illustration to the left, a simple example of a window. A window is usually a small sector whose floor and ceiling are higher and lower than the surrounding sectors. In this event, the textures will be improperly aligned with the surrounding non-windowed walls.



As shown to the left, by setting the upper and lower unpegged attributes, we can fix this problem. The above texture will be rendered top to bottom, while the below texture will be rendered bottom to top.

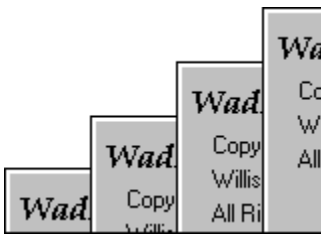
Just when you thought the pain was over... I am sorry to say there is a single remaining complication with vertical texture alignment. In the above example, the sector in which the player is standing is the same height as the texture. If this is not the case, the textures will still be aligned, but perhaps not as one might suspect. If the sector in which the player is standing is shorter than the texture height, the bottom of the textures will be clipped as shown in the following illustration.



This probably seems confusing; after all, I just explained how lower unpegged renders from bottom to top, right?

Here's the catch: rendering from bottom to top does not necessarily start at the bottom of the given texture. It selects its starting point within the texture based on the distance from floor to ceiling. It may seem confusing, but it's all done to allow nice alignment of sidedef textures.

Now for the easy part. In each area, the texture mapping proceeds along the horizontal axis from the left edge of the wall to the right, as seen from the player's perspective. This is not usually a problem except in a few cases. For example, a staircase usually consists of many small sectors with consistently varying heights connected together. If the sides of the staircase are visible, the horizontal textures will probably need to be aligned. The following illustration demonstrates a rising staircase using the example texture for the below texture. Notice the horizontal and vertical alignment problems.



The vertical alignment can be fixed by applying the appropriate unpegged attribute. The horizontal alignment problem provides a good lead in for the next section...

Sidedef Textures - Manual Adjustments

There are some situations in which it is necessary to make manual adjustments to the texture alignment. The previously mentioned horizontal alignment problem with a rising staircase, for example, requires an offset value for the textures to be tiled properly. The offsets determine how far into the texture rendering starts, and they may be positive or negative. Values larger than the size of the texture will wrap. WadAuthor provides editing of these offsets via the linedef properties dialog.

Calculating the horizontal offset is usually pretty simple. Calculating the vertical offset requires more work. The three factors to consider are the ceiling height of the current sector, the ceiling height of the adjacent sector, and the height of the texture. The desired offset will almost always be the difference between two of these values.

Sidedef Textures - Common Applications

Having provided such a wealth of information, it's time to get to the good stuff: applying the knowledge. The following list details the most common situations in which texture alignment can be a problem.

- n **Doors** are usually implemented as a sector where the ceiling height is the same as the floor height. When the door is shut, the player sees the above texture. When the door opens, the ceiling of the door sector rises. None of the unpegged attributes are set; this lets the door image rise with the ceiling since the above texture is based at the bottom edge of the rising ceiling. The sidedefs making up the inside tracks of the door are usually one-sided and thus need only a main texture. Because main textures are rendered from the ceiling down, the inside tracks will rise as the door opens. This is usually not desirable. By setting the lower unpegged attribute for the inside tracks, they will be based on the non-changing floor and will remain stationary.
- n **Secret doors** add a little complexity if the door is to blend in with the surrounding wall. When the door is closed, the player normally sees the above texture, from floor to ceiling. Since the upper texture is rendered from the bottom up, the bottom of the door, where it meets the floor, will show the bottom of the texture. If the room has a non-standard ceiling height, the adjacent sidedef will not be vertically aligned with the closed door. The simplest solution is to set the lower unpegged attribute for all adjacent sidedefs, making them render bottom up as well.
- n **Switches** get shifted towards the floor if the sidedef to which they are applied is shorter than the switch texture. To place the switch at a better vertical position, set the lower unpegged attribute. The sidedef will render bottom up, and the switch will always be at a constant height.

Sector Textures

The floor and ceiling images are all 64 units square. Unlike sidedef textures, their alignment is very simple; they are aligned with an underlying 64 unit grid. Because of this, floor and ceiling textures will always be tiled seamlessly within any given sector. The only problem arises with images that contain special markings, teleporter pads for example. The sector in which these textures appear must be exactly aligned to a 64 unit

grid to insure proper alignment.

Image List Control

WadAuthor uses image list controls to provide a consistent method for image selection. Image list controls behave like a standard Windows edit control with a few significant enhancements. The following table summarizes the additions to the normal keyboard interface.

Key	Results
F2	Invokes the image browse dialog for full-size side-by-side graphical selection. The remaining keystrokes allow navigation of the list displayed by the image browse dialog without the graphical overhead.
Up Arrow	Selects the previous image.
Down Arrow	Selects the next image.
Page Up	Selects the image one page above the current image.
Page Down	Selects the image one page below the current image.
Ctrl+Home	Selects the first image in the list.
Ctrl+End	Selects the last image in the list.

When typing an image name, the image list control will perform an incremental search to help find the desired texture with the fewest number of keystrokes.



Remember, the null texture is represented as a hyphen and is only allowed for walls, not ceilings or floors. Click here for more information about [texture mapping](#).

Relative Change Control

WadAuthor uses relative change controls to provide some shortcuts for assigning numeric values. Relative change controls behave like a standard Windows edit control with a few enhancements. The following table summarizes the numeric prefixes understood by a relative change control.

<u>Prefix</u>	<u>Results</u>
++/--	A double plus or minus prefix will add or subtract the specified value from the original value. For example, when setting a sector light level, entering a value of ++10 would increase the existing value by 10 units.
+++/--	A triple plus or minus prefix will add or subtract the specified value to an appropriate reference value. For example, when setting a sector ceiling height, entering a value of +++128 would make the ceiling 128 units above the floor.



The triple plus or minus prefixes are only meaningful when a reference is available. For example, a floor can be adjusted relative to the ceiling, but a lighting level has no reference — a triple plus or minus would be interpreted in the same way as a double plus or minus.

[Files] Section

This section maintains data file settings. The following table explains the entries that may appear in this section.

<u>Entry</u>	<u>Default</u>	<u>Description</u>
ConfigFile	DOOM.WCF	Specifies which wadgame configuration file to use.
MakeBackups	1	Determines whether WadAuthor creates backup files during a save operation.
NodeBuilding	1	Determines whether WadAuthor doesn't build nodes, uses its internal code, or calls an external program for node building during a save operation.
ExternalNodeBuildCmd	(none)	Specifies the command line to be executed when performing an external node build.
ExternalNodeBuildShow	0	Determines whether the child process launched for an external node build is visible.

[Caching.Images] Section

WadAuthor accesses all wadfile images through a caching system that maintains the most frequently used images as bitmaps. More specifically, two separate caches are used, separating images into two categories: sprites and everything else. Sprite images are used for displaying things and are accessed quite frequently when sprite bitmap viewing is enabled.

The caching systems are initialized whenever a wadgame configuration file is read. This will occur once at startup and again whenever you select a different wadgame configuration file. The following table explains the entries that may appear in this section.

<u>Entry</u>	<u>Default</u>	<u>Description</u>
Sprites	25 for Windows 3.x 50 for Windows95 75 for Windows/NT	Sets the number of sprite images maintained as bitmaps.
Other	25 for Windows 3.x 50 for Windows95 75 for Windows/NT	Sets the number of other images maintained as bitmaps.



A value of zero will disable the given cache. A value of negative one will cause the cache to be allocated large enough to cache all applicable images within the wadgame's main wadfile. Using negative one can be very effective when using Windows/NT.

[General] Section

WadAuthor maintains general options in this section. The following table explains the entries that may appear in this section.

<u>Entry</u>	<u>Default</u>	<u>Description</u>
UnlimitedUndo	1	Determines whether WadAuthor allows unlimited undo up to the limits of system memory.
RestoreState	1	Determines whether WadAuthor remembers the initial frame window position and extents.

[Recent File List] Section

This section is used to maintain the list of most recently accessed files on the *File* menu. The entries in this section will always be like the sample shown below.

FileN = Filename

In the example, *N* provides the number by which *Filename* will be listed in the most recently used section of the *File* menu.

[Views] Section

This section is used to maintain defaults for the wad editing views. The following table explains the entries that may appear in this section.

<u>Entry</u>	<u>Default</u>	<u>Description</u>
StartMaximized	0	Determines if WadAuthor will maximize document windows when opened.
ShowGrid	1	Determines if WadAuthor initially shows the grid in new map editing views.
ShowThingBitmaps	1	Determines if WadAuthor initially shows things as bitmaps in new map editing views.
SnapToGrid	1	Determines if WadAuthor initially snaps objects to the current grid in new map editing views.
UseDragCursor	1	Determines if WadAuthor displays the standard document cursors during a drag operation.

[Window.X] And [Dialog.X] Sections

Many of the dialog boxes and windows maintain their position information or other private data. This information is stored in separate sections within the INI file. The section name, as suggested by the title of this help topic, is always formed by appending the internal name of the window or dialog to the appropriate identifier, separated by a period. For example, the check map dialog stores its private data in a section named *[Dialog.CheckMap]*. The following table explains some of the entries that may commonly appear in these sections.

Entry	Default	Description
Position	(none)	Specifies the screen position in x,y format at which the given window or dialog will appear.
Extents	(none)	Specifies the width and height in cx,cy format to which the given window or dialog will be sized.
Show	(none)	Specifies the manner in which the window or dialog will initially be shown (normal, maximized, minimized, etc.).



Position and extent data may be adjusted if you have changed screen resolutions since the last time it was updated. Every effort will be made to restore the previous settings, but the position and extents will always be clipped, if necessary, to fit the available screen resolution.

Map Editing View

{button ,KL(`Interface',0,`,`')}` [Related Topics](#)

The map editing view provides the primary interface in WadAuthor. It is a graphical representation of a map that allows the user to manipulate the things, vertexes, linedefs, sidedefs, and sectors therein. You may click on any of the following topics for more information about the map editing view.

[Map Editing View Legend](#)

[Toolbars](#)

[Toolbar Docking Areas](#)

[Status Bar](#)

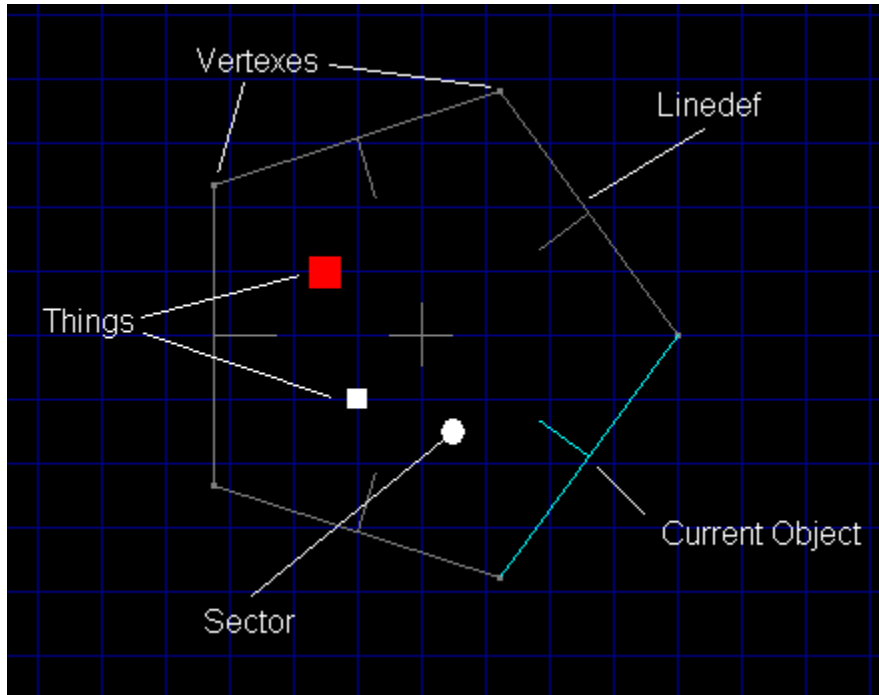
[Configuring The View](#)

[The Current Object](#)

[Manipulating Objects](#)

Map Editing View Legend

The illustration below shows the view for a rather trivial map which I have constructed. The sections that follow provide some additional information about the various objects that appear within the image.



Things

The tiny colored squares represent things. The color and size used for each thing is specified in the wadgame configuration file. By learning the different colors and relative sizes, one can quickly get an overall impression of how a given map is populated.

When the zoom is set high enough (usually at maximum or one step less), actual bitmap images of the things will be displayed if thing bitmaps are enabled. WadAuthor will only display the bitmap images at high zoom levels to preserve reasonably accurate placement information.

Vertexes

The almost imperceptible rectangles that appear at the ends of linedefs represent vertexes. Vertexes may be moved, deleted, or joined, but they have no properties to edit.

Linedefs

Linedefs are probably the most visible objects, since they provide the map outline. Notice that each linedef has a proportionately smaller perpendicular line extending from its center; this indicates the direction in which the line is facing. The smaller line protrudes from the front sidedef of the linedef; the opposite side is, of course, the back sidedef.

Sectors

Sectors are actually made visible by their associated linedefs. In the image above, there is a single pentagonal sector in which the things exist.

Toolbars

{button ,KL(` Full Screen Command',0,`,`')} [Related Topics](#)

Toolbars are normally displayed across the top of the application window, below the menu bar. They provide quick mouse access to many useful commands in WadAuthor. You may click on any of the buttons in the images below for details about the command it invokes.

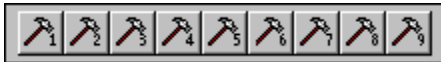
Standard Toolbar



View Toolbar



User-Defined Toolbar



Full Screen Toolbar

The full screen toolbar, shown in the illustration below, is only visible when full screen mode is enabled; check the related topics for more information about full screen mode.



Toolbar Docking Areas

{button ,KL(`Interface',0,`,`')} [Related Topics](#)

The toolbars in WadAuthor may be docked at the left, right, top, or bottom of the main window, or they may float singly or in groups. To dock a toolbar, drag it near one of the toolbar docking areas and drop it. To undock a toolbar, drag it away from the docking area to which it is currently attached.



Holding down the control key while dragging a toolbar will prevent it from docking.

Status Bar

The status bar, shown in the illustration below, is displayed at the bottom of the WadAuthor window. You may click on any of the panes in the image below for details about the information displayed within.



Configuring The View

{button ,KL(`Interface',0,`,`')} [Related Topics](#)

Configuring the map editing view involves scrolling, zooming, and enabling or disabling a few options. For details, consult the related topics or click the following button for a list of view commands.

{button List of View Commands ,KL(`View Commands',0,`,`')}

The Current Object

{button ,KL(`Interface;Object Filter Dialog;Tags',0,`,`')} [Related Topics](#)

As the mouse cursor moves over objects on the map editing view, the object underneath the cursor is highlighted. An abbreviated textual description of the object also appears in the status bar. This highlighted object is called the current object. WadAuthor supplies this feature to make object selection easier — particularly in crowded regions of the map. Understanding how WadAuthor selects the current object can help you get to a desired object more quickly. The following sections explain how the current object is chosen.

No Object Filtering

By default, WadAuthor starts without an object filter setting. This allows the user to work with all of the different object types at once, without requiring separate modes of operation for each type. Each time the mouse is moved, WadAuthor finds the nearest thing, vertex, and linedef; it also determines over which sector, if any, the mouse cursor is positioned. From these objects, WadAuthor then selects the closest object, as calculated from the object's center, and makes it current.

Using Object Filtering

If you are interested in a particular type of object, you may set the object filter. Once the filter is set to a specific type, WadAuthor confines its search for a nearest object to that type. Because distance calculations will only be performed for a single type, this can greatly improve performance on slower machines.



When the current object is tagged, WadAuthor will highlight all other objects sharing the tag number in a different color. This makes it easy to see which objects trigger actions.

Manipulating Objects

WadAuthor supplies features for creating, deleting, and editing objects. For more information, click on one of the following topics.

[Creating New Objects](#)

[Selecting Objects](#)

[Moving Objects](#)

[Editing Existing Objects](#)

[Deleting Objects](#)

Message Pane

The message pane describes actions of menu items as you use the arrow keys to navigate through menus. This area similarly shows messages that describe the actions of toolbar buttons as you depress them, before releasing them. If after viewing the description of the toolbar button command you wish not to execute the command, then release the mouse button while the pointer is off the toolbar button. This area is also used to display various messages relevant to the current operation.

Wadgame Pane

The wadgame pane contains the name of the current wadgame, as defined by the current wadgame configuration file.

Object Filter Pane

The object filter pane displays the object filter setting for the active view.

Thing Display Pane

The thing display pane displays the current thing display attributes.

Grid Spacing Pane

The grid spacing pane displays the grid spacing setting for the active view.

Zoom Pane

The zoom pane displays the current zoom setting for the active view.

Cursor Position Pane

The cursor position pane displays the location of the cursor, in wadmap coordinates, within the active view.

File Menu

Clicking a top level menu expands or contracts the list of menu items; clicking a menu item displays the help topic for the item.

File Menu

[New](#)

[Open](#)

[Select Configuration File](#)

[Close](#)

[Save](#)

[Save As](#)

[Print](#)

[Print Preview](#)

[Print Setup](#)

[Most Recently Used Files](#)

[Exit](#)

Edit Menu



[View Menu](#)



[Tools Menu](#)



[Window Menu](#)



[Help Menu](#)

Edit Menu

Clicking a top level menu expands or contracts the list of menu items; clicking a menu item displays the help topic for the item.



[File Menu](#)



[Edit Menu](#)

[Undo](#)

[Cut](#)

[Copy](#)

[Paste](#)

[Delete](#)

[Select All](#)

[Force Node Build](#)

[Properties](#)



[View Menu](#)



[Tools Menu](#)



[Window Menu](#)



[Help Menu](#)

View Menu

Clicking a top level menu expands or contracts the list of menu items; clicking a menu item displays the help topic for the item.



[File Menu](#)



[Edit Menu](#)



[View Menu](#)

[Full Screen](#)

[Zoom In](#)

[Zoom Out](#)

[Best Fit](#)

[Go To Bookmark](#)

[Set Bookmark](#)

[Increase Grid](#)

[Decrease Grid](#)

[Object Filter](#)

[Thing Display](#)

[Select Map](#)

[Statistics](#)



[Tools Menu](#)



[Window Menu](#)



[Help Menu](#)

Tools Menu

Clicking a top level menu expands or contracts the list of menu items; clicking a menu item displays the help topic for the item.



[File Menu](#)



[Edit Menu](#)



[View Menu](#)



[Tools Menu](#)

[Customize](#)

[Options](#)

[Motifs](#)

[Tags](#)

[Check Map](#)

[Run Map](#)



[Window Menu](#)



[Help Menu](#)

Window Menu

Clicking a top level menu expands or contracts the list of menu items; clicking a menu item displays the help topic for the item.



[File Menu](#)



[Edit Menu](#)



[View Menu](#)



[Tools Menu](#)



[Window Menu](#)

[New Window](#)

[Cascade](#)

[Tile Horizontally](#)

[Tile Vertically](#)

[Arrange Icons](#)

[Refresh](#)

[Window Numbers](#)



[Help Menu](#)

Help Menu

Clicking a top level menu expands or contracts the list of menu items; clicking a menu item displays the help topic for the item.



[File Menu](#)



[Edit Menu](#)



[View Menu](#)



[Tools Menu](#)



[Window Menu](#)



[Help Menu](#)

[Help Topics](#)

[Search For Help On](#)

[How To Use Help](#)

[Register](#)

[About WadAuthor](#)

Undo Command

The undo command reverses the effects of the last operation. WadAuthor features multiple levels of undo, limited only by system memory.



When a map is saved, the undo history is emptied.

Keyboard: Ctrl+Z
 Alt+Backspace

Cut Command

The cut command removes the current selection and places it on the clipboard.

Keyboard: Ctrl+X

Copy Command

The copy command places a copy of the current selection on the clipboard.

Keyboard: Ctrl+C

Paste Command

The paste command inserts the contents of the clipboard into the current map at a user-specified position. WadAuthor will allow you to specify the position by using the arrow keys to move the insertion point, or by clicking the primary mouse button at the desired location. Pressing the escape key will abort the operation.



The paste location determines which sector will be used as the parent for any linedefs requiring fixup during the paste operation.

Keyboard: Ctrl+V

Delete Command

The delete command removes the contents of the current selection.

Keyboard: Delete

Select All Command

The select all command selects all objects in the current map as defined by the current object filter and thing display options. For example, if the object filter is set to linedef, all linedefs within the current map will be selected.

Force Node Build Command

The force node rebuild command will cause a node building operation to be performed the next time the map is saved — whether it is actually needed or not.

Properties Command

{button ,KL(` Properties',0,`,`')} [Related Topics](#)

The properties command edits the properties of the current selection. If there is no current selection, it displays the document properties instead.

Keyboard: Alt+Enter

Join Vertexes Command

The join vertexes command joins the first selected vertex to the second. This command can be used to reduce the number of sides in a sector or closing open sectors.

Keyboard: J

Join Linedefs Command

The join linedefs command joins the first selected linedef to the second. This command can be used to connect two sectors so that the player may pass between them.

Keyboard: J

Split Linedef(s) Command

The split linedefs command divides all selected linedefs in half, inserting a new vertex at each center point.

Keyboard: X

Align Linedef Textures Command

`{button ,KL(`Textures',0,`,`')}` [Related Topics](#)

The align linedef textures command adjusts the texture offsets to insure that the images will blend seamlessly from the player's perspective.

Flip Linedef(s) Command

`{button ,KL(`Flip Linedef(s) Dialog',0,`,`')}` [Related Topics](#)

The flip linedefs command displays the flip linedefs dialog, allowing the user to specify how the front and back sides of the linedef should be exchanged.

Apply Sector Motif Command

{button ,KL(`Motifs',0,`,`')} [Related Topics](#)

The apply sector motif command displays the apply motif dialog, allowing the user to select a motif to be applied to the selected sectors.

Rotate Sector Command

{button ,KL(`Rotate Sector Dialog',0,`,`')} [Related Topics](#)

The rotate sector command displays the rotate sector dialog, allowing the user to specify the angle by which the selected sector should be rotated.

Scale Sector Command

{button ,KL(`Scale Sector Dialog',0,`,`')} [Related Topics](#)

The scale sector command displays the scale sector dialog, allowing the user to specify the percentage by which the selected sector should be scaled.

Convert Sector To Stairs Command

{button ,KL(`Stairs',0,`,`')} [Related Topics](#)

The convert sector to stairs command displays the convert sector to stairs dialog, allowing the user to change the selected sector into a staircase as specified by the current stair motif.

Convert Sector To Door Command

{button ,KL(` Doors',0,`,`')} [Related Topics](#)

The convert sector to door command changes the selected sector into a door as specified by the current door motif.

Map Name Command

```
{button ,KL(`Map Name Dialog',0,`,`')} Related Topics
```

The map name command invokes the map name dialog box to allow the user to change the name of the current map.

Scripts Command

{button ,KL(`Scripts',0,`,`')} [Related Topics](#)

The scripts command invokes the scripts dialog box to allow the user to change the script code for the current map.

Edit Raw Data Command

{button ,KL(`Raw Data',0,`,`')} [Related Topics](#)

The edit raw data command invokes one of the raw editing dialogs to allow the user to access the actual data saved to disk for a given map object. This feature should be used with caution.

Create Motif From Sector Command

{button ,KL(`Motifs',0,`,`')} [Related Topics](#)

The create motif from sector command creates a new motif based on the selected sector. To invoke the command, select the sector, then click the secondary mouse button on it to invoke the context menu and choose the *Create Motif From Sector* option.

Set Linedef Length(s) Command

{button ,KL(`Set Linedef Length(s) Dialog',0,`,`')} [Related Topics](#)

The set linedef length command displays the set linedef length(s) dialog, allowing the user to specify a length for the selected linedefs.

Make Linedefs Parallel Command

{button ,KL(`Make Linedefs Parallel Dialog',0,`,`')} [Related Topics](#)

The make linedefs parallel command displays the make linedefs parallel dialog, allowing the user to specify options to be used in performing the command.

New Command

The new command creates a new map and a new window in which to edit the map.

Keyboard: Ctrl+N

Open Command

{button ,KL(`File Open/Save Common Dialog',0,`,`')} [Related Topics](#)

The open command displays the file open common dialog to allow the user to open a map from an existing wadfile.

Keyboard: Ctrl+O

Select Configuration File Command

{button ,KL(` Configuration Files',0,`,`')} [Related Topics](#)

The select configuration file command displays the select configuration file dialog to allow the user select a different wadgame configuration file.

Close Command

The close command closes the currently active map.

Keyboard: Ctrl+F4

Save Command

{button ,KL(` File Open/Save Common Dialog',0,`,`')} [Related Topics](#)

The save command saves the currently active map. If the map has not previously been saved, file save common dialog will be displayed to allow the user to supply a filename.

Keyboard: Ctrl+S

Save As Command

{button ,KL(`File Open/Save Common Dialog',0,`,`')} [Related Topics](#)

The save as command displays file save common dialog and saves the currently active map under a new filename.

Print Command

{button ,KL(`Print Dialog',0,`,`')} [Related Topics](#)

The print command displays print dialog to print the currently active map.

Keyboard: Ctrl+P

Print Preview Command

The print preview command displays a preview of the currently active map as it would appear when output to the current printer.

Print Setup Command

{button ,KL(`Print Setup Dialog',0,`,`')} [Related Topics](#)

The print setup command displays the print setup dialog to configure the current printer.

Most Recently Used File Command

The most recently used file commands open the wadfiles listed on the *File* menu.

Exit Command

The exit command closes the WadAuthor application.

Keyboard: Alt+F4

Help Topics Command

The contents command displays the help topics for the WadAuthor help file.

Keyboard: F1



When invoked from the keyboard, this will display help for the current context, if any, before defaulting to the contents topic.

Search For Help On Command

The search for help on command allows the user to search the indexed list of available topics in the WadAuthor help file.

How To Use Help Command

The how to use help command displays the Windows help on using the help file viewer.

Register Command

{button ,KL(`Register Dialog',0,`,`')} [Related Topics](#)

The register command displays the register dialog which allows the user to authorize the software license.

Tip Of The Day Command

{button ,KL(`Tip of the Day Dialog',0,`,`')} [Related Topics](#)

The tip of the day command displays the tip of the day dialog which provides useful information.

About WadAuthor Command

{button ,KL(`About Dialog',0,`,`')} [Related Topics](#)

The about WadAuthor command invokes the about dialog.

Context Help Command

The context help command changes the cursor to the context help cursor. Help will be provided for the next menu item, button, dialog box, or other control upon which the user clicks.

Keyboard: Shift+F1

New Rectangular Sector Command

{button ,KL(`New Objects',0,`,`')} [Related Topics](#)

The new rectangular sector command displays the new rectangular sector dialog and inserts the specified sector into the map.

New Polygonal Sector Command

{button ,KL(`New Objects',0,`,`')} [Related Topics](#)

The new polygonal sector command displays the new polygonal sector dialog and inserts the specified sector into the map.

New Thing Command

`{button ,KL(`New Objects',0,`,`')}` [Related Topics](#)

The new thing command displays the thing properties dialog and inserts the specified thing into the map.

Insert Object Command

{button ,KL(`New Objects',0,`,`')} [Related Topics](#)

The insert object command displays the insert object dialog allowing the user to select a type of object to be inserted into the map.

Keyboard: Insert

Customize Command

{button ,KL(`Customize',0,`,`')} [Related Topics](#)

The customize command displays the user tools dialog to allow the user to customize the list of user tools.

Options Command

{button ,KL(`Options',0,`,`')} [Related Topics](#)

The options command displays options dialog to allow the user to modify WadAuthor configuration settings.

Motifs Command

{button ,KL(`Motifs',0,`,`')} [Related Topics](#)

The motifs command displays the motifs dialog for the current wadgame configuration file.

Keyboard: Ctrl+M

Tags Command

{button ,KL(`Tags',0,`,`')} [Related Topics](#)

The tags command displays the tags dialog for the current map. This allows the user to add, modify, or remove tags.

Keyboard: Ctrl+T

Check Map Command

{button ,KL(`Check Map Dialog',0,`,`')} [Related Topics](#)

The check map command examines the currently active map for errors and, if any are found, displays check map dialog to aid in fixing them.

Keyboard: Ctrl+K

Run Map Command

The run map command runs the currently active map using the run command specified in the current wadgame configuration file.

Keyboard: Ctrl+R

User Defined Tool Command

{button ,KL(`Customize',0,`,`')} [Related Topics](#)

The user-defined tool commands execute the user-defined command line and arguments.

Full Screen Command

The full screen command toggles the state of the full screen display mode. Full screen mode hides the main window caption, menu, toolbar docking frames, and status bar in order to allow open views to occupy the entire screen. When using full screen mode, the special full screen floating toolbar, shown in the illustration below, will be displayed. Clicking the button will toggle full screen mode.



Keyboard: Ctrl+U

Zoom In Command

{button ,KL(`Interface',0,`,`')} [Related Topics](#)

The zoom in command increases the map editing view viewing scale.

Keyboard: Plus

Zoom Out Command

{button ,KL(`Interface',0,`,`')} [Related Topics](#)

The zoom out command decreases the map editing view scale.

Keyboard: Minus

Zoom In Max Command

{button ,KL(`Interface',0,`,`')} [Related Topics](#)

The zoom in max command increases the map editing view viewing scale to the allowed maximum.

Keyboard: Ctrl+Plus

Zoom Out Max Command

{button ,KL(`Interface',0,`,`')} [Related Topics](#)

The zoom out max command decreases the map editing view scale to the allowed minimum.

Keyboard: Ctrl+Minus

Object Zoom Command

{button ,KL(`Interface',0,`,`')} [Related Topics](#)

The object zoom command configures the map editing view scale to best fit the current object.

Keyboard: Z

Clear Object Zoom Command

{button ,KL(`Interface',0,`,`')} [Related Topics](#)

The clear object zoom command restores the map editing view scale to its original settings prior to the first use of the object zoom command.

Keyboard: Shift+Z

Best Fit Command

{button ,KL(`Interface',0,`,`')} [Related Topics](#)

The best fit command centers the map within the map editing view and adjusts the scale so that the entire map can be viewed at once.

Keyboard: Home

Go To Bookmark Command

{button ,KL(`Interface',0,`,`')} [Related Topics](#)

The go to bookmark commands restore the view position and zoom settings from the given bookmark number.

Keyboard: Ctrl+1 through Ctrl+9

Set Bookmark Command

{button ,KL(`Interface',0,`,`')} [Related Topics](#)

The set bookmark commands record the view position and zoom settings for the given bookmark number.

Keyboard: Ctrl+Shift+1 through Ctrl+Shift+9

Increase Grid Command

{button ,KL(`Interface',0,`,`')} [Related Topics](#)

The increase grid command increases the map editing view grid spacing.

Keyboard: Shift+Plus
]

Decrease Grid Command

{button ,KL(`Interface',0,`,`')} [Related Topics](#)

The increase grid command decreases the map editing view grid spacing.

Keyboard: Shift+Minus
 [

Object Filter Command

{button ,KL(`Interface',0,`,`')} [Related Topics](#)

The object filter command displays the map editing view to specify the type of objects to edit.

Keyboard: Ctrl+F

Thing Display Command

```
{button ,KL(`Interface;Thing Display Dialog',0,`,`')}
```

[Related Topics](#)

The thing display command displays the thing display dialog to specify which attribute bits a thing should have to be displayed.

Select Map Command

{button ,KL(`Select Map Dialog',0,`,`')} [Related Topics](#)

The select map command displays the select map dialog to choose which map to edit within a multi-map wadfile.

Statistics Command

{button ,KL(`Statistics Dialog',0,`,`')} [Related Topics](#)

The statistics command displays the statistics dialog which provides some useful information about the current map.

Show Grid Command

{button ,KL(`Interface',0,`,`')} [Related Topics](#)

The show grid command shows or hides the grid. The current grid setting will be used for all alignment operations; it simply will not appear within the map editing view.

Show Floor Grid Command

{button ,KL(`Interface',0,`,`')} [Related Topics](#)

The show floor grid command shows or hides the floor image alignment grid. This grid setting may be used to properly place sectors whose floor or ceiling images must be tiled correctly.

Show Thing Bitmaps Command

{button ,KL(`Interface',0,`,`')} [Related Topics](#)

The show thing bitmaps command enables or disables the use of thing bitmaps at high zoom settings. Enabling thing bitmap display provides a clearer picture of exactly what objects are within a given area, but disabling it improves performance and minimizes graphics resource usage.

Snap To Grid Command

{button ,KL(`Interface',0,`,`')} [Related Topics](#)

The snap to grid command determines whether or not WadAuthor will align objects with the current grid at the conclusion of a successful drag and drop move operation.

Scroll Commands

{button ,KL(`Interface',0,`,`')} [Related Topics](#)

A page scrolling command moves half the current map distance in the specified direction. A line scrolling command moves one-twentieth the current map distance in the specified direction.

Keyboard

Key	Effect
Up Arrow	Scrolls the view up by one line.
Down Arrow	Scrolls the view down by one line.
Left Arrow	Scrolls the view left by one line.
Right Arrow	Scrolls the view right by one line.
Ctrl+Up Arrow	Scrolls the view up by one page.
Ctrl+Down Arrow	Scrolls the view down by one page.
Ctrl+Left Arrow	Scrolls the view left by one page.
Ctrl+Right Arrow	Scrolls the view right by one page.
Page Up	Scrolls the view up by one page.
Page Down	Scrolls the view down by one page.



The current map distance is derived from the current zoom setting. As the zoom increases, scroll commands change position by smaller amounts; as the zoom decreases, scroll commands change position by larger amounts.

Object Filter Commands

{button ,KL(`Object Filter Dialog',0,`,`')} [Related Topics](#)

An object filter command sets the object filter to a specified object type.

Keyboard

Key	Effect
L	Sets the object filter to linedefs.
N	Sets the object filter to none.
S	Sets the object filter to sectors.
T	Sets the object filter to things.
V	Sets the object filter to vertexes.

New Window Command

The new window command opens another window into the currently active map. All editing operations are synchronized across multiple windows into a single map.

Cascade Command

The cascade command re-arranges the open windows in a cascading fashion.

Tile Horizontally Command

The tile horizontally command re-arranges the open windows in a tiled fashion in the horizontal direction.

Tile Vertically Command

The tile vertically command re-arranges the open windows in a tiled fashion in the vertical direction.

Arrange Icons Command

The arrange icons command re-arranges any minimized windows within the client area of the main window.

Refresh Command

The refresh command causes the currently active map to perform a complete redraw.

Keyboard: F5

Window Number Command

The window number command selects from the list of open windows on the *Window* menu.

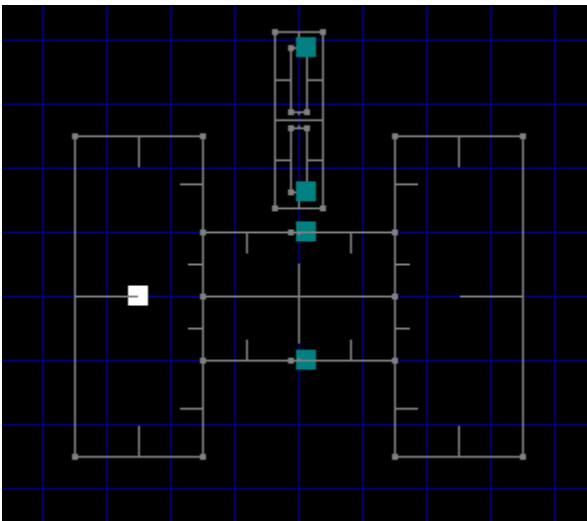
Understanding Polyobjects

Polyobjects were introduced with the release of Hexen. They allow the wad designer to create effects that are simply not possible with other wadgames due to the self-imposed limitations of the game engines. A polyobject is a complex architectural feature whose location is resolved at map load time. In English this means that a polyobject is a polygonal object defined at one location on the map that appears in a different location at run time.

There are several important points to making a polyobject work. Without following them the map may refuse to load, or the game may crash when the polyobject is triggered. The important points are listed below.

- n You must select a number between 1 and 255 to uniquely identify the polyobject to be created. This number is used during the construction.
- n You must define the polyobject. This is done so that the wadgame knows which linedefs to manipulate as part of the object. The simplest method to accomplish this is to let WadAuthor create the polyobject as a polygonal sector of the desired number of sides, then apply the Polyobj_StartLine special to one of the linedefs — supplying the polyobject number selected in the first step. It is possible to manually specify the linedefs, but it's much easier this way.
- n You must provide an anchor for the polyobject. This supplies the wadgame with the point around which the polyobject will rotate, slide, or whatever. This is done by placing a polyobject anchor thing at the desired position near the polyobject and setting the thing's angle to the polyobject number selected earlier.
- n You must provide a destination on the map to which the anchor will be translated at map load time. This supplies the wadgame with the final destination for the polyobject anchor. This is done by placing a start spot (crushing or non-crushing) at the location which the polyobject anchor should occupy at runtime and setting the thing's angle to the polyobject number selected earlier.

The following illustration is taken from the EX_POLYO.WAD that ships with WadAuthor. You may click on portions of the illustration below for more specific help.



Door Polyobjects

These two sub-sectors are what will appear as doors from the player's perspective. The polyobject anchors and map start spots are used to provide the wadgame engine with enough information to correctly move the polyobjects into place.

Polyobject Anchors

The polyobject anchors are used as the points at which the polyobjects will be attached once they have been translated to their final positions within the map. In the case of doors, these points define the “hinges” around which the door polyobject swings.

Each anchor point is associated with a start spot by matching up the numbers that appear in each thing's facing angle. For example, the anchor with an angle value of one will get translated to the start spot with an angle value of one.

If this seems like a bit of a hack, don't worry — it is. According to the specifications, polyobjects were already supported before the thing tag code was in place, so the map designers were using the thing angle as an interim solution. By the time the thing tag code was ready, the designers did not want to go back and change all their work.

Start Spots

The start spots are used as final destination points for the polyobject anchors. Each anchor point is associated with a start spot by matching up the numbers that appear in each thing's facing angle. For example, the anchor with an angle value of one will get translated to the start spot with an angle value of one.

If this seems like a bit of a hack, don't worry — it is. According to the specifications, polyobjects were already supported before the thing tag code was in place, so the map designers were using the thing angle as an interim solution. By the time the thing tag code was ready, the designers did not want to go back and change all their work.

Raw Data Editing Dialogs

The raw data editing facilities of WadAuthor are only for wadfile editing experts. When a single object is selected, the user may select the *Edit Raw Data...* option from the context menu. Doing so will invoke one of the raw editing dialogs, allowing the user to change the actual data stored to disk for each map object. The feature should be used with extreme caution as it can easily render a map un-playable.



Reference

The reference section is a series of topics designed to explain various aspects of WadAuthor's operation. You may click on any of the topics listed below for more specific help.

[Commands](#)

[Configuration Files](#)

[Context Menus](#)

[Dialogs](#)

[Glossary](#)

[Initialization File](#)

[Keyboard Interface](#)

[Menus](#)

[Motifs](#)

[Mouse Interface](#)

[Properties](#)

[Understanding Polyobjects](#)

[Understanding Scripts](#)

[Understanding Texture Mapping](#)

[Using Custom Images](#)

Commands

The following button provides an alphabetized reference to the WadAuthor commands for which help topics exist.

```
{button List of Commands ,KL(`Commands',0,`,`')}
```

Configuration Files

{button ,KL(' Configuration Files',0,`,`')} [Related Topics](#)

WadAuthor configuration files (WCF files) contain information specific to a given wadgame. WadAuthor uses them to locate the wadgame's main wadfile, provide the naming convention for new maps, supply a textual name of the game to the user, generate default motifs, and obtain data for all the different types of things, linedefs, and sectors available in the given wadgame. The format is somewhat similar to Windows initialization files (INI files).



The configuration file data is loaded into memory during startup. If you edit the file, you must either exit the application and restart or read use the *Select Configuration File* command on the *File* menu to use your changes.

Wadgame Section

A valid WCF file must contain a wadgame section at the top of the file. The following example is taken from the DOOM.WCF file that ships with the product.

```
[WadGame]
Name=DOOM
IWAD=DOOM.WAD
NewMap=E1M1
Directory=C:\DOOM
Run=$_Comspec /c doom.exe -file $_Wadfile -warp $_Wadmap -nosound
```

The *Name* entry provides a textual name for the wadgame, the *IWAD* entry provides WadAuthor with the filename of the wadgame's main wadfile, the *NewMap* entry provides the naming convention for new maps, and the *Directory* entry provides the location where the wadgame is installed. If any entry (except *Run*) in this section is invalid, the wadgame configuration dialog will be invoked to allow the user to supply correct information.

The *Run* entry cannot easily be validated; fortunately, it rarely requires alteration. The *Run* entry is used to launch the current wadgame to run the current map. The *\$_Wadfile*, *\$_Wadmap*, and *\$_Comspec* macros are replaced at run time with the fully qualified pathname of the current map, the command-line parameter appropriate to the map name, and the command shell processor. You may wish to change the *Run* entry if a special PIF file is required, or if you wish to perform batch-file processing when running a map.

The *-nosound* parameter is present by default to avoid crashing when running wadgames under Windows 3.1, Windows for Workgroups 3.11, or Windows/NT. Most users of Windows95 should be able to safely remove this parameter, allowing the wadgame to supply sound effects and music during play.



The *\$_Wadmap* parameter is formed by eliminating all non-numeric characters from the map name, discarding leading and trailing spaces. For example, *E2M7* is transformed into *27* at run time. If a wadgame that violates this convention is released, you will need to obtain an updated copy of the software or modify the *Run* entry manually.

Default Sections

A valid WCF file must also contain some default sections. These are used to generate default sector, door, and stair motifs in the event that a motif file cannot be located at startup. The following examples are taken from the DOOM.WCF file that ships with the product.

```
[Default.Sector]
Above=STARTAN2
Main=STARTAN2
Below=STARTAN2
Ceiling=CEIL3_5
Floor=FLOOR4_8
CeilingHeight=128
```

FloorHeight=0
Lighting=160

[Default.Door]
Base=FLAT1
Door=BIGDOOR2
Track=DOORTRAK
Type=1

[Default.Stair]
FloorRunner=STEP1
CeilingRunner=STEP1
Stairwell=METAL
FloorInc=8
CeilingInc=0
LightingInc=0

Other Sections

The data for all the different types of things, linedefs, and sectors available in the given wadgame follows these sections. The format of the information is documented within the WCF file and is subject to change in future releases. You may edit this data to suit your preferences, but I strongly urge you to create a backup copy of the original WCF file first and proceed with caution.

Context Menus

{button ,KL(` Menu',0,`,`')} [Related Topics](#)

Context menus, short for context-sensitive menus, are menus whose options are limited to those that are appropriate within the current context. To display an object's context menu, select the object and press *Shift+F10*, or click the object with the secondary mouse button. The following sections describe the context menu options that will appear when a given object is clicked.

Map Editing View

Menu Item

[New Rectangular Sector](#)

[New Polygonal Sector](#)

[New Thing](#)

[Zoom In](#)

[Zoom Out](#)

[Best Fit](#)

[Increase Grid](#)

[Decrease Grid](#)

[Show Grid](#)

[Show Floor Grid](#)

[Show Thing Bitmaps](#)

[Snap To Grid](#)

[Map Name](#)

[Scripts](#)

[Properties](#)

Description

Inserts a new square sector the size of a single grid, centered at the menu position. The sector is snapped to grid.

Displays the new polygonal sector dialog, allowing the user to insert a new sector of a specified radius and number of sides. The sector is centered at the menu position, but not snapped to grid.

Displays the new thing dialog, allowing the user to insert a new thing at the menu position.

Increases the map viewing scale.

Decreases the map viewing scale.

Alters zoom and position to fit the map to the available display area.

Increases the grid spacing.

Decreases the grid spacing.

Shows or hides the grid.

Shows or hides the floor and ceiling image alignment grid.

Enables or disables the use of thing bitmaps at high zoom settings.

Enables or disables alignment of objects with the grid.

Allows editing of the map name.

Allows editing of map script code.

Allows editing of document properties.

Map Object

Menu Item

[New Rectangular Sector](#)

[New Polygonal Sector](#)

[New Thing](#)

[Join Vertexes](#)

[Join Linedefs](#)

[Split Linedef\(s\)](#)

[Flip Linedef\(s\)](#)

[Set Linedef Length\(s\)](#)

[Make Linedefs Parallel](#)

[Align Linedef Textures](#)

[Apply Sector Motif](#)

Description

Inserts a new square sector the size of a single grid, centered at the menu position. The sector is snapped to grid.

Displays the new polygonal sector dialog, allowing the user to insert a new sector of a specified radius and number of sides. The sector is centered at the menu position, but not snapped to grid.

Displays the new thing dialog, allowing the user to insert a new thing at the menu position.

This item is only available when exactly two vertexes are selected. Joins the first vertex selected to the second.

This item is only available when exactly two one-sided linedefs are selected. Joins the first linedef selected to the second, connecting the two sectors.

Splits all selected linedefs in half by inserting a new vertex at each center.

Displays the flip linedefs dialog, allowing the user to reverse the selected linedef direction and, optionally, sidedef data.

Displays the set linedef length(s) dialog, allowing the user to specify the length and anchor points for the selected linedefs.

Displays the make linedefs parallel dialog, allowing the user to specify the anchor and distance for the selected linedefs.

Displays the align textures dialog, allowing the user to specify which side to align and which texture to use for sizing information.

This item is only available when at least one sector is selected.

Displays the apply motif dialog and applies the motif selected to

the sectors.

[Rotate Sector](#)

This item is only available if a single sector is selected. Displays the rotate sector dialog and rotates the sector by the specified angle.

[Scale Sector](#)

This item is only available if a single sector is selected. Displays the scale sector dialog and scales the sector by the specified percentage.

[Create Motif From Sector](#)

This item is only available if a single sector is selected. It creates a new motif based on the properties of the sector.

[Convert Sector To Stairs](#)

This item is only available if a single four-sided sector is selected. Displays the convert sector to stairs dialog, allowing the user to select some options and convert the sector to a staircase.

[Convert Sector To Door](#)

This item is only available if a single that connects to at least one other sector is selected. Displays the apply motif dialog, allowing the user to select a door motif and convert the sector to a door.

[Edit Tags](#)

This item is only available if a single object capable of being tagged is selected. It allows the user to display the objects sharing the given tag.

[Edit Raw Data](#)

This item is only available if a single object is selected. It allows the user to edit the actual data saved to disk for a given map object.

[Properties](#)

This item is only available if at least one object other than a vertex is selected. Displays the appropriate property editing dialog.

Dialogs

The following button provides an alphabetized reference to the WadAuthor dialog boxes for which help topics exist.

```
{button List of Dialog Boxes ,KL(^ Dialogs',0,';^')}
```

Selecting A Different Configuration File

{button ,KL(` Configuration Files',0,`,` `')} [Related Topics](#)

To create or edit maps for a different wadgame, you must tell WadAuthor which configuration file to use. Close all open maps and choose the *Select Configuration File...* option from the *File* menu. The select configuration file dialog box will be invoked to allow you to specify which configuration file WadAuthor should use.

Initialization File

WadAuthor makes use of a standard Windows initialization file (INI file) to retain most of its settings. The file exists in the same directory as WadAuthor and may be safely deleted at any time — it will be recreated the next time WadAuthor is run. The following topics explain each section of the INI file and the various entries found therein.

[\[Files\] Section](#)

[\[Caching.Images\] Section](#)

[\[General\] Section](#)

[\[Recent File List\] Section](#)

[\[Views\] Section](#)

[\[Window.X\] and \[Dialog.X\] Sections](#)

Keyboard Interface

{button ,KL(`Interface',0,`,`')} [Related Topics](#)

Although WadAuthor is, perhaps, at its best with a mouse or other pointing device, I have always been most efficient when working from the keyboard. The following sections describe the keyboard interfaces available within WadAuthor.

Global Commands

The following commands are available from anywhere within the application.

Key	Command	Description
F1	Context Help	Displays the help file for the current context.
Shift+F1	Context Help	Enables mouse-driven context-sensitive help.

Main Frame Window

The main frame window is the main application window. It is the window that contains all the open documents and provides the toolbar and status bar. The following commands are available when working with the main frame window.

Key	Command	Description
Ctrl+N	New	Creates a new document.
Ctrl+O	Open	Opens an existing document.
Alt+F4	Exit	Exits the application.

Map Editing View

Map editing views provide the primary WadAuthor functionality. They display wadmaps and allow the user to manipulate the objects within.

Map Editing View - File Commands

Key	Command	Description
Ctrl+N	New	Creates a new document.
Ctrl+O	Open	Opens an existing document.
Ctrl+F4	Close	Closes the current document.
Ctrl+S	Save	Saves the current document.
Ctrl+P	Print	Prints the current document.

Map Editing View - Tool Commands

Key	Command	Description
Ctrl+M	Motifs	Displays the motifs dialog.
Ctrl+T	Tags	Displays the tags dialog.
Ctrl+K	Check Map	Checks the current map for errors.
Ctrl+R	Run Map	Runs the current map.

Map Editing View - View Commands

Key	Command	Description
Plus	Zoom In	Increases the viewing scale.
Minus	Zoom Out	Decreases the viewing scale.
Home	Best Fit	Alters zoom and position to fit the map to the available display area.
Shift+Plus or]	Increase Grid	Increases the grid spacing.
Shift+Minus or [Decrease Grid	Decreases the grid spacing.
Ctrl+Plus	Zoom In Max	Zooms in to the maximum allowed scale.

Ctrl+Minus	Zoom Out Max	Zooms out to the minimum allowed scale.
Up Arrow	Scroll Up	Scrolls the view up by one line.
Down Arrow	Scroll Down	Scrolls the view down by one line.
Left Arrow	Scroll Left	Scrolls the view left by one line.
Right Arrow	Scroll Right	Scrolls the view right by one line.
Ctrl+Up Arrow	Page Up	Scrolls the view up by one page.
Ctrl+Down Arrow	Page Down	Scrolls the view down by one page.
Ctrl+Left Arrow	Page Left	Scrolls the view left by one page.
Ctrl+Right Arrow	Page Right	Scrolls the view right by one page.
Page Up	Page Up	Scrolls the view up by one page.
Page Down	Page Down	Scrolls the view down by one page.
F5	Refresh	Redraws the current map.
Z	Object Zoom	Zooms the map to best fit the current object.
Shift+Z	Clear Object Zoom	Restores the original map settings prior to the object zoom command.
Ctrl+U	Full Screen	Toggles the state of full screen display.

Map Editing View - Selection Commands

Key	Command	Description
Tab	Next Object	Sets the current object to the next object.
Shift+Tab	Previous Object	Sets the current object to the previous object.
Ctrl+F	Object Filter	Sets the object selection filter.
Ctrl+M	Select Map	Selects a map from a multi-map wadfile.
F2	Center On Object	Centers the current map display on a specified object.
Shift+F10	Context Menu	Displays the context menu for the selection.
T	Filter Things	Sets the object filter to things.
V	Object Filter Commands	Sets the object filter to vertexes.
L	Object Filter Commands	Sets the object filter to linedefs.
S	Filter Sectors	Sets the object filter to sectors.
N	Filter None	Sets the object filter to none.

The following thing display commands are only valid for wadgames other than Hexen.

1, 2	Thing Display	Toggles the level 1/2 attribute for thing display.
3	Thing Display	Toggles the level 3 attribute for thing display.
4, 5	Thing Display	Toggles the level 4/5 attribute for thing display.
D	Thing Display	Toggles the deaf attribute for thing display.
M	Thing Display	Toggles the multi-player attribute for thing display.

The following thing display commands are only valid for Hexen.

1, 2	Thing	Toggles the level 1/2 attribute for thing display.
------	-----------------------	--

3	Display Thing Display	Toggles the level 3 attribute for thing display.
4, 5	Thing Display	Toggles the level 4/5 attribute for thing display.
D	Thing Display	Toggles the deaf attribute for thing display.
R	Thing Display	Toggles the dormant attribute for thing display
F	Thing Display	Toggles the fighter attribute for thing display
C	Thing Display	Toggles the cleric attribute for thing display
M	Thing Display	Toggles the mage attribute for thing display.
I	Thing Display	Toggles the single-play attribute for thing display
O	Thing Display	Toggles the cooperative-play attribute for thing display
H	Thing Display	Toggles the deathmatch-play attribute for thing display

Map Editing View - Edit Commands

Key	Command	Description
Delete	Delete	Removes the contents of the selection.
Insert	Insert Object	Inserts an object into the map.
Alt+Enter	Properties	Edits properties of the selection.
Ctrl+C	Copy	Copies the contents of the selection to the clipboard.
Ctrl+V	Paste	Pastes the contents of the clipboard into the current document.
Ctrl+X	Cut	Removes the contents of the selection to the clipboard.
Ctrl+Z	Undo	Reverses the effects of the last editing operation.
J	Join Vertexes or Join Linedefs	Joins selected vertexes or linedefs.
X	Split Linedefs	Splits selected linedefs.

Menus

{button ,KL(`Interface',0,`,`')} [Related Topics](#)

Clicking a top level menu expands or contracts the list of menu items; clicking a menu item displays the help topic for the item.



[File Menu](#)



[Edit Menu](#)



[View Menu](#)



[Tools Menu](#)



[Window Menu](#)



[Help Menu](#)

Motifs

{button ,KL(`Motifs',0,`,`')} [Related Topics](#)

Motifs allow the map designer to more easily maintain a consistent look and feel when creating new architecture. When new sectors are created, the current motif is applied to all appropriate sidedefs. Motifs may also be applied to existing sectors if the user has a decorative change of heart.

Because motifs are directly tied to a specific wadgame, the available motifs will depend upon the current configuration file. WadAuthor looks for motif files (MTF files) in the directory from which it runs when reading data from the current configuration file. It uses the base name of the wadgame's main wadfile to determine which motif file to use. For example, if the wadgame's main wadfile is DOOM.WAD, WadAuthor will look for DOOM.MTF. If WadAuthor cannot find a motif file for the current wadgame, it will create a default motif file.

To edit the list of available motifs or to select the current motif, choose the *Motifs...* option from the *Tools* menu. To manually apply a motif, select the sectors to which the motif should be applied, and choose the *Apply Motif...* option from the context menu.

Mouse Interface

{button ,KL(`Interface',0,`,`')} [Related Topics](#)

WadAuthor is, perhaps, at its best when used with a mouse or other pointing device. I find the mouse to be the most direct means for some operations. The following sections describe the mouse interfaces available within WadAuthor.

Map Editing View

Map editing views provide the primary WadAuthor functionality. They display wadmaps and allow the user to manipulate the objects within.

Primary Mouse Button (Usually Left)

Action	Effect
Click	Clicking an object will do one of two things: if selected, a drag and drop operation will begin; otherwise, the object will become selected. Clicking an empty area of the map will clear the selection.
Ctrl+Click	Holding down the control key while clicking an object will toggle its selection state; clicking an empty area of the map has no effect.
Alt+Click	Holding down the alt key while clicking allows for very precise selection. If objects are too close together for the software to distinguish, use this method. Whenever more than one object is "near enough," a dialog box will allow the user to pick the specific object of interest.
Shift+Click	Holding down the shift key begins a rubberband selection. This allows the user to select a range of objects in a single operation. If the user does not drag a rubberband selection, the object clicked will be added to the selection.
Double click	A double click will clear the selection, select the object clicked (if any), and display the properties dialog for the given object.
Ctrl+Double click	Holding down the control key while double-clicking will center the map editing view at the given location.

Secondary Mouse Button (Usually Right)

Action	Effect
Click	Clicking an object will do one of two things: if selected, the context menu for all selected objects will be displayed; otherwise, the object will become selected and the context menu for the specific object will be displayed. Clicking an empty area of the map will display the context menu for the map itself.
Alt+Click	As with the primary mouse button, holding down the alt key while clicking allows for very precise selection. Refer to the primary mouse button Alt+Click explanation for more detail.

Properties

{button ,KL(` Properties',0,`,`')} [Related Topics](#)

The term properties refers to the innate attributes or qualities of an object. For example, a sector has a type, a floor image and height, a ceiling image and height, a lighting level, and a tag number. Property editing refers to the process of changing object properties. WadAuthor provides very powerful property editing through various dialog boxes. To edit an object's properties, select the object and choose *Properties...* from the context menu, choose *Properties...* from the Edit menu, or press *Alt+Enter* to display the appropriate dialog box.



Registration

{button ,KL('Registration',0,`, `') } [Related Topics](#)

WadAuthor is not free. If you use it, you must pay for it. I believe in the shareware system because the “try before you buy” approach is a great way to make sure you’re getting something you want. In allowing you the free use of this program for a time, I am extending you a degree of trust; please do not violate this trust. Remember, only you can make shareware work.

You may use WadAuthor free of charge for 15 days. If, at the end of this time period, you decide that WadAuthor is not for you, please stop using it and remove it from your system — better yet, pass it along to a friend (or enemy if you really don’t like it). If you decide to keep WadAuthor, you must register through one of the means listed at the bottom of this topic.

In exchange for registration, you will receive technical support, upgrades for the life of the product, access to handy wadfile utilities, and the author’s gratitude — not to mention that pleasant lack of guilt that comes with a clear conscience! By registering WadAuthor, you will also get the chance to shape its future through your comments and complaints. Upon receipt of payment, you will be sent the information required to authorize the software license.

CompuServe

Users of CompuServe can register WadAuthor while on-line. To register WadAuthor via CompuServe, log on, GO SWREG at the ! prompt, and follow the instructions. The information you will need to complete the process is displayed below.


Program Title:	WADAUTHOR
Registration ID:	6379
Fee (US\$):	20.00

Snail Mail

To register WadAuthor by snail mail, please send a check or money order, payable to Williston Consulting, in the amount of \$20 (U.S.) to the following address.

Williston Consulting
12704 Bloomfield Ave. Apt. 126
Norwalk CA 90650

Along with your payment, you must enclose a message including the product name (WadAuthor), your name, your company (if any), address, e-mail address (if any), and any comments you care to add. A pre-prepared order form is available by clicking on this topic’s title bitmap or by clicking below.

 [Click here to view the WadAuthor order form.](#)

Registration payments from countries outside the United States of America must be capable of being cashed by a U.S. bank. I will make every attempt to cash whatever I receive, but if I cannot do so, I will be forced to return your order. Alternatively, you may contact me to make any special arrangements for wire transfer of funds. I do not wish to discourage potential foreign users in any way; I simply have not found an alternative.

Understanding Scripts

The scripting capabilities introduced in Hexen add a whole new dimension to wadfile editing. At the cost of a little simplicity of understanding, the flexibility of the wadgame engine has been dramatically enhanced. The purpose of this topic is to give a wadfile-literate user a good introduction to scripts. You may click on the following topics for more information.

General

[What Is A Script?](#)

[Why Are Scripts Compiled?](#)

[Where Is The Data Stored?](#)

[How Do I Use A Script?](#)

[What Is The Scripting Language?](#)

[Where Can I Get More Information?](#)

Example

[Breaking Glass](#)

[How Do I Use The Breaking Glass Script?](#)

What Is A Script?

An *Action Code Script*, or ACS for short, is a series of instructions to be carried out. Originally the script is defined in terms of a C-like programming language. Prior to run time, however, the script code must be compiled into *pcode* form and stored with the map.

The term *pcode* is a kind of shorthand for pseudo-code. It refers to instructions that are executed via an interpreter. Pcode is often used where a fully compiled set of instructions is undesirable, but fully interpreted speed is unacceptable. Visual Basic programs, for example, are actually compiled as a bunch of *pcode* instructions of a different sort.

Why Are Scripts Compiled?

The scripts are compiled because Id/Raven Software decided to do it! All kidding aside, I suspect the scripts must be compiled for sake of performance and simplicity in the runtime environment. As mentioned earlier, the script data must be compiled prior to running the map, or else the script will not execute correctly.

Where Is The Data Stored?

The compiled pcode is stored in a new BEHAVIOR resource for each map. WadAuthor stores the script code in another resource after the BEHAVIOR resource named SCRIPTS. This is a convention proposed by editor authors to allow wadfile designers to share their code and ideas more easily.

When WadAuthor opens a map for which no SCRIPTS resource exists, it tries to decompile the existing BEHAVIOR resource. The decompilation, if successful, will provide source code similar to the code the wadfile designer wrote. WadAuthor cannot determine original variable names or provide the comments, but the essential code will be preserved.

If the BEHAVIOR resource contains no scripts, a default file is read when script editing is desired. Compiling and saving edited script data will overwrite the original scripts for such a map, so caution should be exercised. A map can quickly be rendered unplayable by destroying some of the scripts required at runtime.

How Do I Use A Script?

Each script has a unique identification number that may be used in conjunction with the *ACS_* special codes. For example, to trigger script number one when the player crosses a given linedef, the linedef type should be set to *ACS_Execute*, the arguments should be set to the script number, map number, and script arguments (if any), and the linedef activation should be set to *Player crosses*.

What Is The Scripting Language?

Programmers familiar with the C programming language will immediately recognize the scripting language as a variant of C. It supports many of the standard C language constructs including pre-processing, looping, variables, etc. For more information on the scripting language syntax, consult the official Hexen specifications included with WadAuthor.

Where Can I Get More Information?

The official Hexen specifications provide more information about the scripting language. A copy of the specifications is included with WadAuthor for your convenience, although a newer version may be available electronically.

Breaking Glass

{button ,KL(`Scripts',0,`,`')} [Related Topics](#)

The following script is the default script shipped with WadAuthor, with comments removed for sake of brevity. This topic examines the script in a piece-by-piece fashion, explaining how it works and how to use it most effectively within your Hexen maps. You may click on sections of the script below for more specific help.



This topic assumes that you already understand the scripting language; if you do not, consult the related topics.

```
#include "common.acs"

script 1 ( int iSector, int iThingTag1, int iThingTag2 )
{
    int i;

    Floor_LowerInstant( iSector, 0, 16 );
    Thingsound( iThingTag1, "GlassShatter", 127 );
    delay(const: 1);

    i = 10;
    while ( i-- > 0 )
    {
        Thing_ProjectileGravity( iThingTag1,
            random( T_STAINEDGLASS1, T_STAINEDGLASS0 ),
            random( 0, 255 ), random( 10, 40 ),
            random( 5,20 ) );
        Thing_ProjectileGravity( iThingTag2,
            random( T_STAINEDGLASS1, T_STAINEDGLASS0 ),
            random( 0, 255 ), random( 10, 40 ),
            random( 5,20 ) );
    }
}
```

Common Declarations

```
#include "common.acs"
```

This first line includes the standard declarations used when creating scripts. These were provided with the script compiler from Raven/Id Software. You do not have to use this file, but it sure is easier than creating your own declarations. Placing this line at the top of each script will insure that all specials, things, constants, and other important declarations are available.

Script Declaration

```
script 1 ( int iSector, int iThingTag1, int iThingTag2 )
```

This line declares script number one. The number assigned here is the script number you should use when modifying a map object to trigger it. Notice that I have named the arguments to provides clues to what they do. The first argument has something to do with specifying a sector, while the remaining two specify thing tags. You can name the arguments as you see fit, but it is good programming practice to choose meaningful names.

Variable Declaration

```
int i;
```

This line declares a local variable named `i` that will be used later in the script.

Breaking The Glass

```
Floor_LowerInstant( iSector, 0, 16 );  
Thingsound( iThingTag1, "GlassShatter", 127 );  
delay(const: 1);
```

These lines make the glass appear to be broken. The `Floor_LowerInstant` special will lower the floor of all sectors having the `iSector` tag, thus exposing the main texture of the linedef instead of the below texture.

The `Thingsound` function plays the glass shattering sound all locations on the map where a thing with a tag of `iThingTag1` is located.

Finally, the `delay` function provides a brief pause to make the shattering appear more real.

Providing Fragments

```
i = 10;
while ( i-- > 0 )
{
    Thing_ProjectileGravity( iThingTag1,
        random( T_STAINEDGLASS1, T_STAINEDGLASS0 ),
        random( 0, 255 ), random( 10, 40 ),
        random( 5,20 ) );
    Thing_ProjectileGravity( iThingTag2,
        random( T_STAINEDGLASS1, T_STAINEDGLASS0 ),
        random( 0, 255 ), random( 10, 40 ),
        random( 5,20 ) );
}
```

The variable `i` that was declared earlier is now put to use. By initializing it with a value of ten and decrementing it each time the loop is executed, we cause the two calls to the `Thing_ProjectileGravity` special to be executed ten times.

Thus, a total of twenty randomly selected pieces of glass will be created at the locations on the map given by things having the tag numbers passed to the script. Use of the `random` function makes the effect quite realistic.

How Do I Use The Breaking Glass Script?

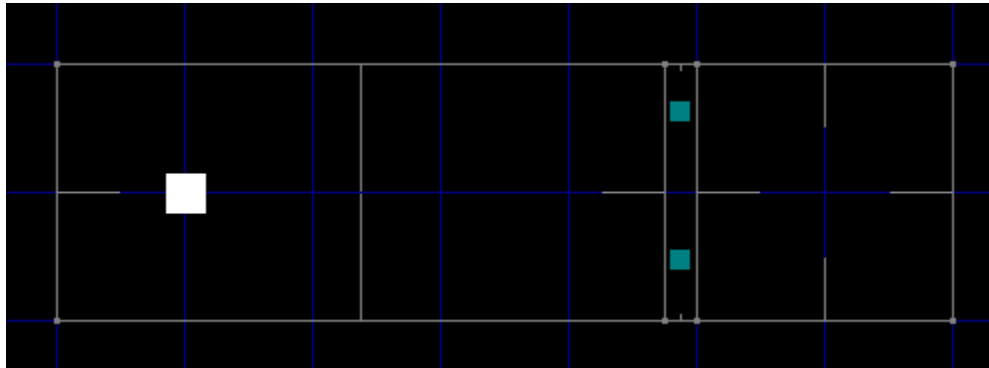
To use the breaking glass script requires some knowledge of how it works along with some specific objects in the map. The basic steps to construct an alcove that is hidden by breakable glass are listed below.

- n Create an alcove sector adjacent and connected to the sector from which you wish the player to see the glass. Give it a unique tag number — this number will be used later.
- n Make the sector's floor height the same as the adjacent sector's ceiling, and make sure the distance between floor and ceiling in the alcove sector is the same as the distance between floor and ceiling in the adjacent sector.
- n Create two map spot things immediately in front of the shared linedef. Set their Z coordinate to 32, and give them both the same tag number — this number will be used later.
- n Edit the linedef shared between the two sectors. Set its main linedefs to a broken glass texture, set its below linedefs to an unbroken glass texture, and set its activation code to trigger when hit by a projectile. Set the type to *ACS_Execute*, and set the special arguments for the current map, desired script number, sector tag, thing tag number, and thing tag number.



Script code for the script number you supply must be available in the wadfile before the effect will work. New wads created by WadAuthor do not have any script code. To compile the default code as part of your wadfile, right click somewhere in an unused portion of the map, select *Scripts...* from the context menu, press the *Compile* button, and press *OK* to accept the data. If you have already edited the script code and deleted the glass breaking script, the code exists within the DEFAULT.ACS file in the WadAuthor directory, and may be pasted into the scripts dialog.

The following illustration is taken from the EX_GLASS.WAD that ships with WadAuthor. You may click on portions of the illustration below for more specific help.



Tagged Map Spot Things

These are map spot things that share a common tag number. I chose to use a common tag number to avoid using more than two things. By assigning the same tag number to both, I can cause the glass shattering sound to play at the same locations from which the glass fragments are created.



Note that these things have their Z coordinate (height from the floor) set so that they are at the player's approximate eye level. This makes the shattering effect more realistic, as the fragments of glass seem to emanate from the point at which the glass was struck.

Tagged Sector

This is the sector whose tag number is passed to the script. The sector is configured with the same overall distance between floor and ceiling as its adjacent sector, but its floor height is equivalent to the ceiling height of the adjacent sector. This is done for a good reason.

When the script is triggered, the sector is lowered, and the shared linedef's main texture is displayed in place of the below texture. By lowering the sector instantly (instead of smoothly like a lift), the breaking effect is instantaneous.

Glass Linedef

This is the linedef that appears to the player as stained glass and cannot be crossed. It cannot be crossed because the tagged sector floor is at the same height as the adjacent sector ceiling. The intact stained glass image is applied to the linedef's below texture. When the tagged sector is lowered, the linedef's main texture will be displayed instead, and the linedef can then be crossed.



Troubleshooting

I really hope you're not reading this. There cannot possibly be any bugs in my code, right? (*sigh*) Oh well, back to reality. The following section contains solutions for common problems, and instructions for getting technical support.

Common Problems - WadAuthor

[The Graphics Are Wrong](#)

[The Run Map Feature Fails](#)

[I Hear No Sound When Running A Map](#)

Common Problems - Maps In General

[Things Are Missing](#)

When All Else Fails...

[Getting Technical Support](#)

The Graphics Are Wrong

{button ,KL(`Getting Technical Support',0,`,`')} [Related Topics](#)

If the wadfile images are not displayed or are displayed incorrectly, it is probably due to an incompatibility between your video card and the image acceleration code WadAuthor employs. Try switching to a different color depth to see if the problem goes away. If the problem does not go away, please consult the related topics for getting technical support.

The Run Map Feature Fails

{button ,KL(` Configuration Files;Getting Technical Support',0,`,`') } [Related Topics](#)

If the run map feature fails, it is probably due to one of the following reasons.

- ⌘ The run command is invalid. Open the current wadgame configuration file in notepad (or some other text editor), carefully examine the *Run* setting in the *WadGame* section, and fix any errors before trying again.
- ⌘ You do not have enough memory to launch the wadgame. Make more memory available or exit Windows before running your map.
- ⌘ There is some kind of problem with the map you're trying to run. Select *Check Map...* from the *Tools* menu and fix any problems that WadAuthor finds before trying again.
- ⌘ The WARUN.EXE utility does not exist in the same directory from which WadAuthor is running. WadAuthor uses this utility to launch the wadgame, and it must be present in the same directory as the WadAuthor executable.

The run map feature was very difficult to support in a consistent manner across Windows, Windows95, and Windows/NT. It is possible that it will fail, even when none of the reasons given above apply. If you simply cannot get it to work, consult the related topics for getting technical support.

Things Are Missing

One of the most common problems for new wadfile designers is that of missing things. A missing thing, in this context, is a thing that is displayed by WadAuthor, but not present during actual game play.

The most common reason for this is that the [thing attributes](#) are set in such a manner that they prevent the thing from appearing. For example, a thing with the *multi* attribute will not appear in a single player game.

Getting Technical Support

If you are a registered user of WadAuthor, or if you are having problems getting the shareware version installed and working, please contact me through one of the means listed below.

CompuServe 70541,1335

Snail Mail Williston Consulting
532 N. Forrest St. Apt. 26
Wayland MI 49348

Make sure to include the following information in your message.

- n The product with which you are having the problem.
- n The version of Windows that you are using.
- n If a message is displayed, please include the full text — including the caption of the dialog box.
- n If the problem is reproducible, please include whatever steps are necessary to recreate the problem.

As a conscientious author, I will make every attempt to fix any bugs reported to me. It is very difficult, however, to fix bugs that I cannot reproduce. So, the more information you can provide, the greater chance I have of finding and squashing the bug.

Tutorial

The purpose of this tutorial is to highlight the best features of WadAuthor while guiding you through the construction of a new map. Novices should follow through the topics in the order in which they appear. Others may wish to use individual topics solely as a reference.



Throughout the tutorial, new terms are set off by the graphic shown to the left. For a more comprehensive list of important terms, consult the glossary.



If you make a mistake when performing any of the instructions in the following topics, don't panic. You can always fix your mistakes by selecting *Undo* from the *Edit* menu. You may continue to select undo until every change you have made has been undone



up to the point at which the map was last saved.

Simple Map Editing

[Creating A New Map](#)

[Creating The First Room](#)

[Changing The Shape Of The Room](#)

[Rotating And Scaling The Room](#)

[Adding Players And An Enemy](#)

[Editing Map Object Properties](#)

[Creating A Hallway](#)

[Connecting The Hallway To The Room](#)

[Creating A Local Door](#)

[Providing An Exit](#)

[Playing The Map](#)

Advanced Map Editing

[Creating A Secret room](#)

[Creating A Remote door](#)

[Creating A Teleporter](#)

[Creating A Staircase](#)

[Creating A Rising Staircase](#)

Suggestions


[Develop Around A Common Theme](#)

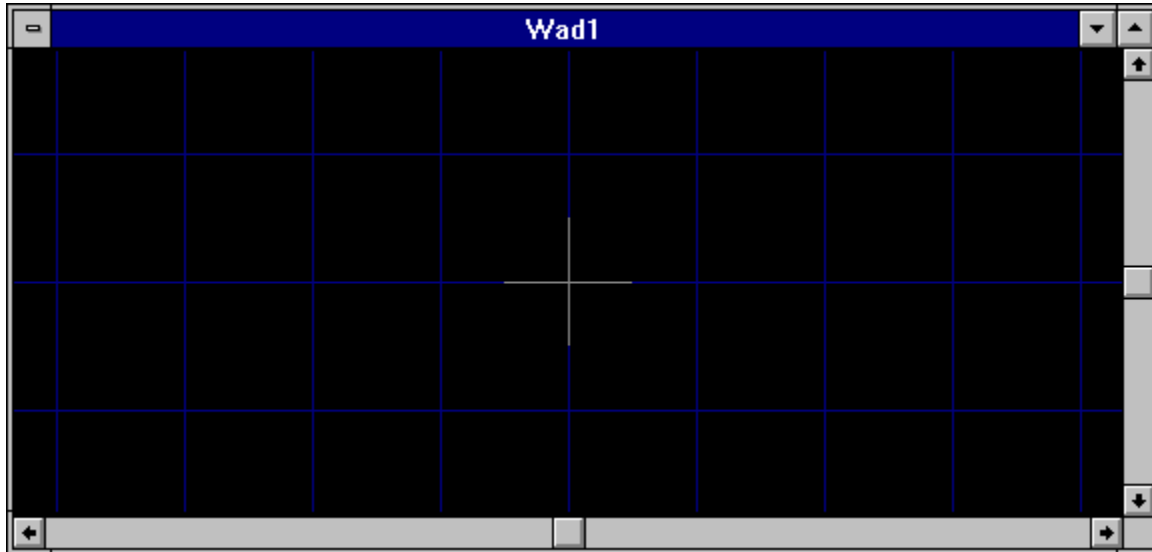
[Consider Multi-Player Modes](#)

[Understand The Third Dimension](#)

[Record A Demo](#)

Creating A New Map

Creating a new wadmap is simple. Press the  toolbar button or select *New* from the *File* menu. A new window will be created with an empty map ready for editing. The window should look something like the illustration below.



Creating The First Room

{button ,KL(`Context Menus;New Objects',0,`,`')} [Related Topics](#)

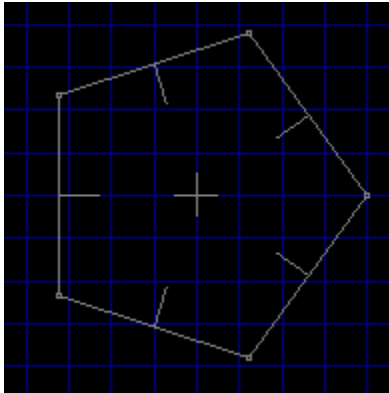
To create the first room, click the secondary mouse button roughly where the center of the room should be. The resulting menu is the context sensitive menu, or context menu for short. This menu will always provide only those operations which are appropriate for the object clicked.

Select the *New Polygonal Sector...* option from the context menu, and enter the requested information in the resulting dialog box. Press the *OK* button to complete the process and create the new sector.



A sector roughly corresponds to a single room. I say roughly because a room, from the player's perspective, can be made up of many different sectors to achieve different effects.

In the illustration below, I created a five-sided sector of radius 256 and re-scaled the map. WadAuthor will scale the map to best fit the available window when you press the *Home* key. Try pressing *Home* and zooming around with the *plus* and *minus* keys until you are satisfied with the display.



Changing The Shape Of The Room

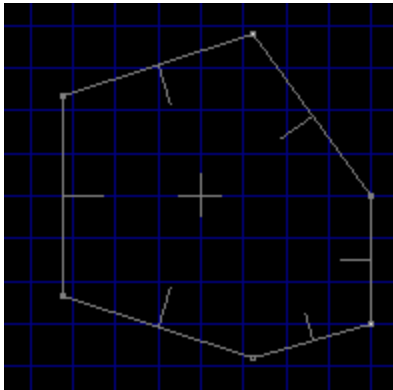
To change the shape of an existing sector, click and hold the primary mouse button over one of the small rectangles where two of a sector's walls meet. This will begin a drag and drop operation that will allow you to position the given corner wherever you like.



The walls of a sector are called linedefs, and the small rectangles where linedefs meet are called vertexes. Linedefs always connect exactly two vertexes, so moving a vertex changes the overall shape of a given sector. Notice that each linedef has a small line at its center protruding inward toward the center of the sector. These lines indicate which side of the linedef is the front side. This will become more important later on in the tutorial.

You may also change the number of sides that a given sector has by changing the number of linedefs. Click the secondary mouse button on any linedef for a given sector and select the *Split Linedef(s)* option from the context menu. This will create a new vertex in the center of the linedef, effectively splitting it in half. You may then position the new vertex, as described above, to modify the shape of the room.

In the illustration below, I have split one of the linedefs and dragged the new vertex.



Rotating And Scaling The Room

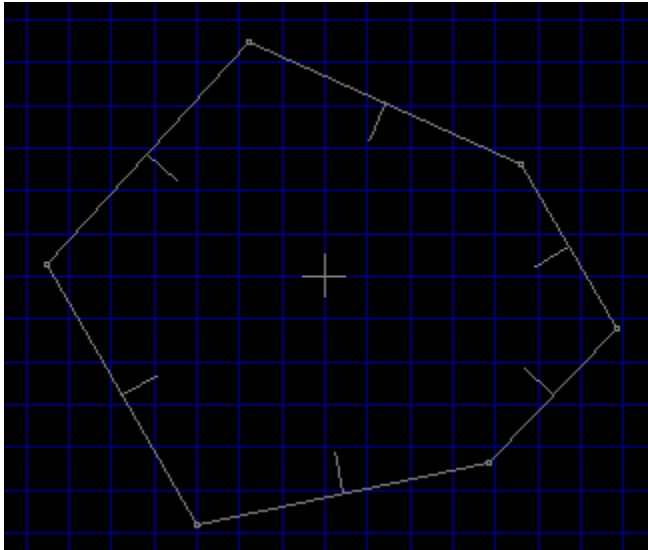
```
{button ,KL(`Rotate Sector Command;Rotate Sector Dialog;Scale Sector Command;Scale Sector Dialog',0,`,`')}  
Related Topics
```

WadAuthor makes it very easy to retain the shape of a room while changing its overall size and orientation. Let's try an example.

Click the secondary mouse button somewhere within the sector, and select the *Rotate Sector...* option from the context menu. In the resulting dialog box, enter an angle and press the *OK* button. The sector has been rotated around its center by the specified number of degrees.

Now let's change the size. Click the secondary mouse button somewhere within the sector, and select the *Scale Sector...* option from the context menu. In the resulting dialog box, enter a percentage and press the *OK* button. The sector has been scaled from its center by the specified value.

By applying different scaling and rotation to different sectors, it should be possible to create virtually any kind of desired shape. In the illustration below, I have applied a 30° rotation to the sector and scaled it by 150%.



Adding Players And An Enemy

To specify where the player will start within the map, we must create a player start thing at the desired location. Click the secondary mouse button somewhere within the sector and select the *New Thing...* option from the context menu.

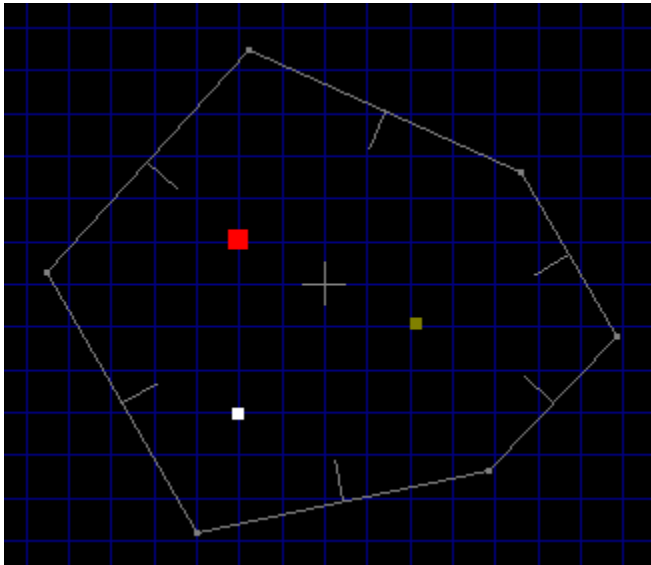
In the resulting dialog box, select *Start* in the class combobox, *Player 1* in the type listbox, and a direction for the player to face. Make sure that the checkboxes labeled *1/2*, *3*, and *4/5* are checked before pressing the *OK* button.

A map without enemies would be very boring indeed, so before continuing on, place an enemy somewhere within the sector. To do this, repeat the above procedure for adding a player start thing, selecting *Enemy* from the class combobox and some nasty opponent in the type listbox.



In wadmap editing terminology, a thing is any object (other than walls, floors, etc.) with which the player may interact. Some things may be picked up and used (ammunition, healing, etc.), others function as scenery (trees, pillars, etc.), and some are downright hazardous to the player's health!

In the illustration below, I have added a player start thing, an enemy, and a weapon.



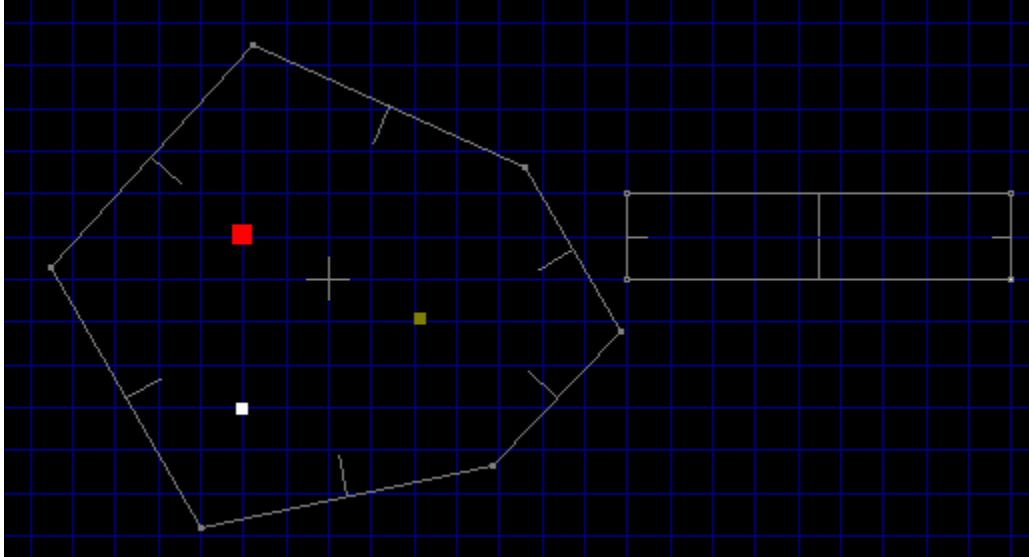
Editing Map Object Properties

{button ,KL(` Properties',0,`,`')} [Related Topics](#)

Property editing is one of the most basic and most flexible operations in WadAuthor. To edit the properties of an existing map object, select it and choose *Properties...* from the context menu, choose *Properties...* from the Edit menu, or press *Alt+Enter* to display the appropriate dialog box.

Creating A Hallway

Click the secondary mouse button on a blank section of the map near the sector, and select the *New Rectangular Sector* option from the context menu. Enter the requested information in the resulting dialog box, and press the *OK* button to complete the process. Click and hold the primary mouse button on one of the linedefs and drag and drop until you have a hallway similar to that shown in the illustration below.



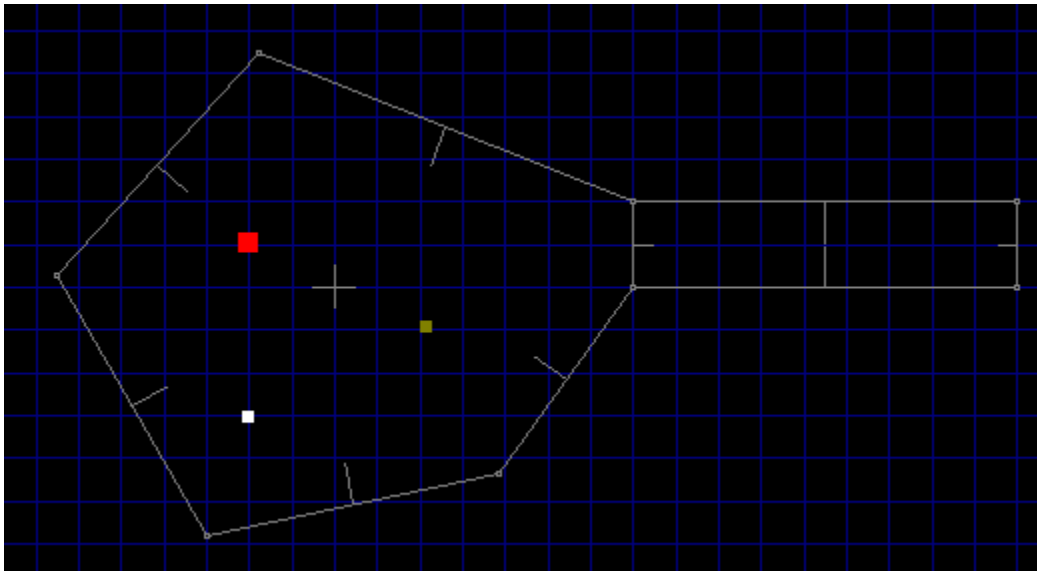
Connecting The Hallway To The Room

Our map now features two sectors: the room and the hallway. We need to connect them so that the player may move between them. WadAuthor automates this process for you by allowing you to join two linedefs. Follow the following instructions closely.

- n Click the primary button on the first linedef to be joined.
- n While holding down the Ctrl key, click the primary button on the second linedef to which the first will be joined.
- n Release the Ctrl key and click the secondary mouse button on either of the two selected linedefs.
- n Select the Join Linedefs option from the context menu.

It is very important that you use this method for connecting sectors on a wadmap. For each linedef, WadAuthor has to know which sector lies on each side. If this information is corrupted, you will not be able to play your map.

In the illustration below, I have joined one of the room's linedefs to the hallway.



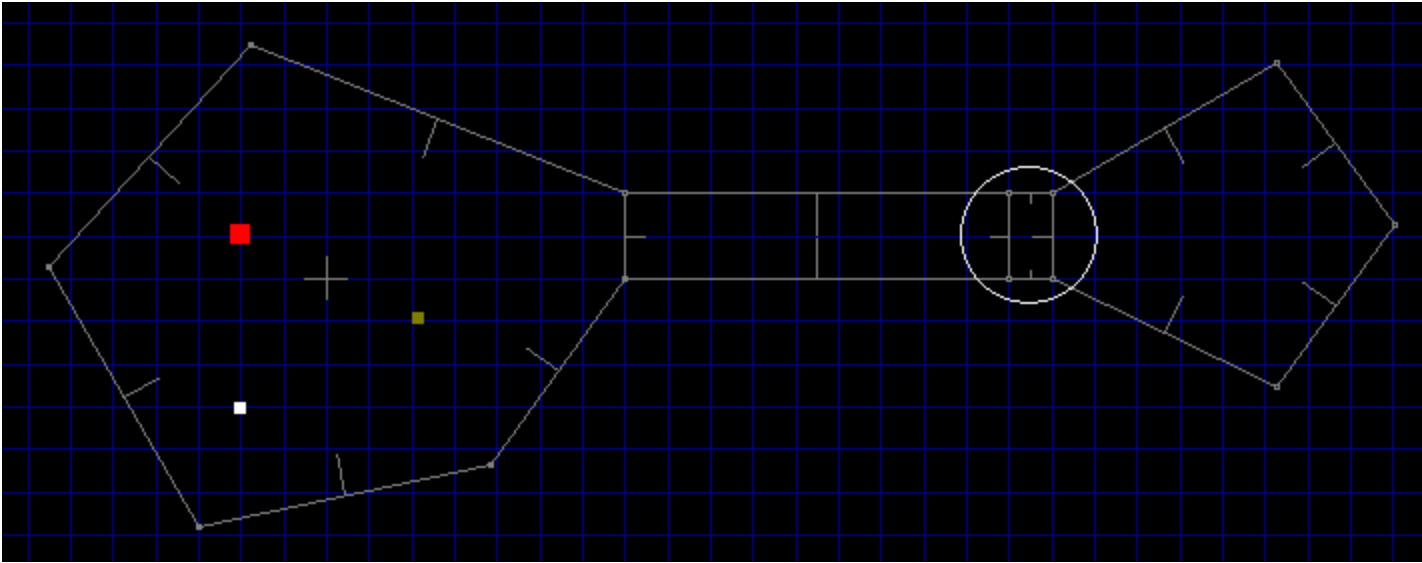
Creating A Local Door

One of the more common architectural features found in wadmaps is the local door. A local door is one which opens when the player walks up to it and triggers the “use” action. WadAuthor simplifies the process of creation by converting an existing sector into a local door. The sector to be converted must have at least one two-sided linedef that joins with a second sector.

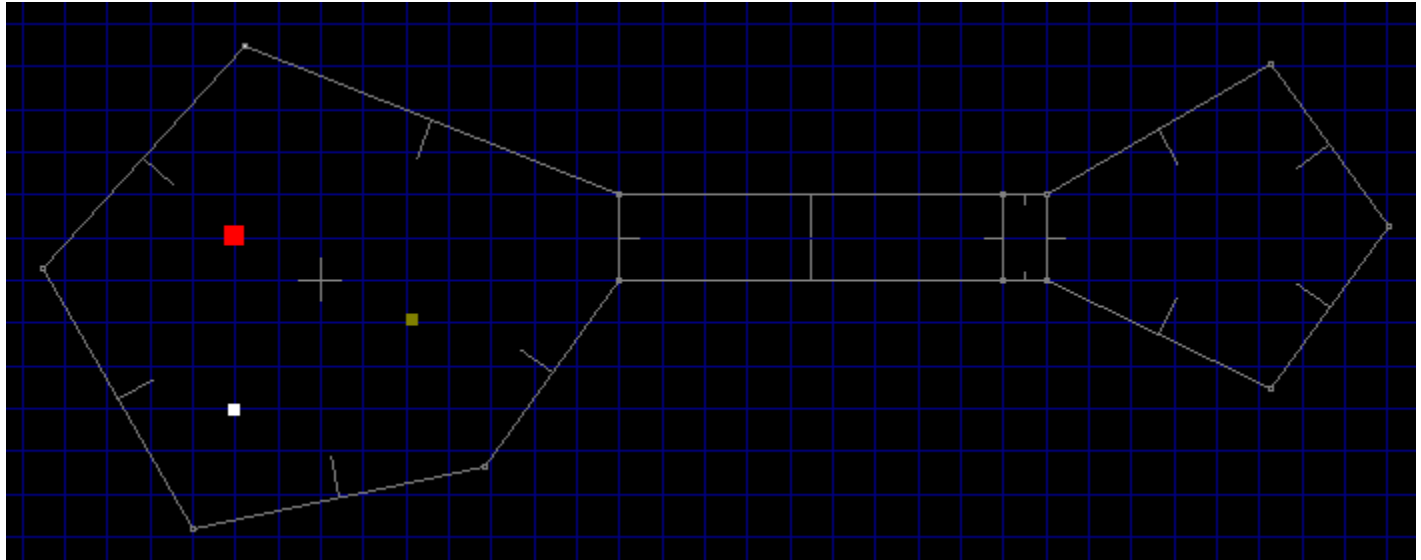


The Hexen wadgame does not make a distinction between local and remote doors as the other wadgames do.

In the illustration below, I have joined two additional sectors (one rectangular, one five-sided with a radius of 256) at the other end of the existing hallway and used paintbrush to circle the sector to be converted into a door.



Click the secondary mouse button on any sector meeting the requirements listed above and select the *Convert Sector To Door...* option from the context menu. The resulting dialog box will allow you to select how the door should appear and behave. In the illustration below, I have converted the previously circled sector to a door. Note that the east linedef’s front and back have been reversed. This is very important! Only the front side of a linedef may be used to trigger a given action — this information will be more useful later on in the tutorial.



Providing An Exit

It's always a good idea to provide a means by which the player can exit a given map and continue on to the next. Otherwise, the only way out is death! To avoid jumping into more advanced topics right now, we'll take this opportunity to create our first sub-sector and set our first linedef actions. We're going to create an area within the second room that ends the level when the player steps inside.

Place a new rectangular sector roughly in the center of the sector into which the door leads. Because you clicked the secondary mouse button within an existing sector, WadAuthor is smart enough to interpret this as a request for a sector within a sector. The new sector will automatically share the floor, ceiling, and lighting attributes of its parent.

Remember that business about the front side of a linedef being the only side that can trigger actions? Here's the first case where this is important. The front sides of the linedefs in our sub-sector all face inward. We're going to need to flip them if walking *into* the sub-sector is going to trigger an exit.

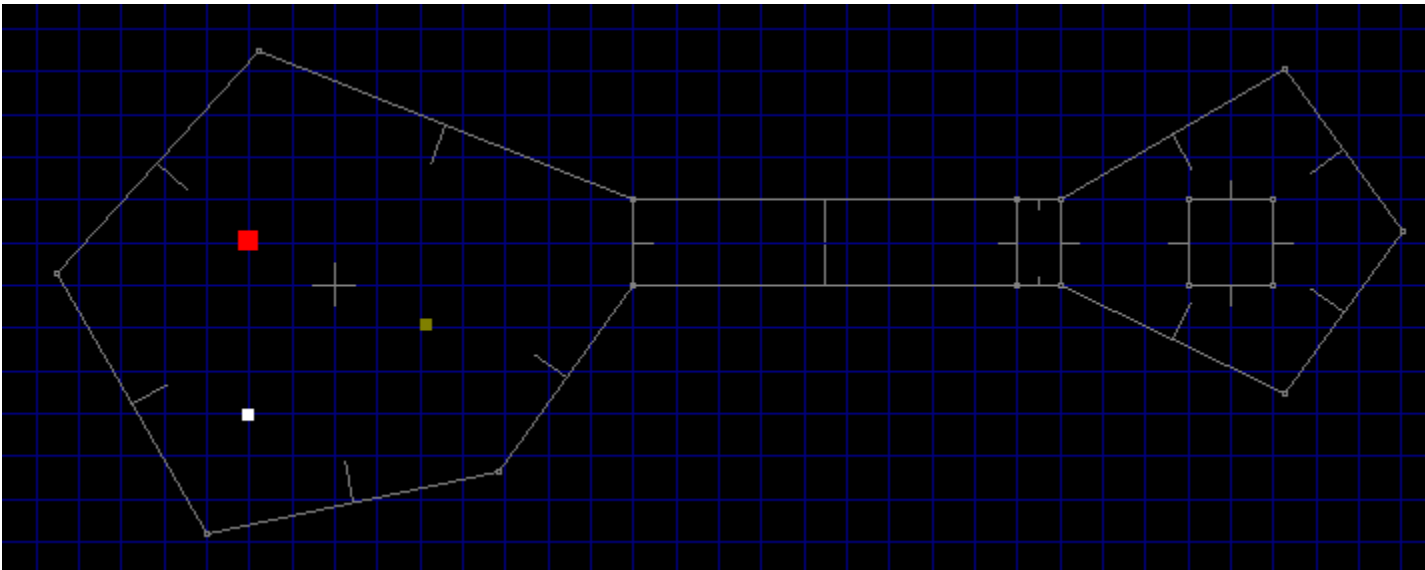
To flip the linedefs, click the secondary mouse button within the sub-sector and select the *Flip Linedef(s)...* option from the context menu. Select the *Include textures and other data* radio-button and press the *OK* button to continue. The linedefs should now be facing the correct direction. All that remains is to configure them to exit the map.

Click the secondary mouse button again within the sub-sector and select the *Properties...* option from the context menu. This will display the map object properties dialog. This dialog box will be covered in greater detail later on in the tutorial. For now, make sure the linedefs page of the dialog is active by clicking the *Linedefs* tab at the top. Select *Exit* in the class listbox, select *W1: end level, go to next* in the type listbox, and press the *OK* button. The linedefs of our sub-sector are now configured to exit the map when the player steps inside.



The Hexen wadgame does not use the same codes as the other wadgames. When creating an exit for a Hexen map, select *Teleport* in the class listbox and the *Teleport_NewMap* special in the type listbox. Be sure to enter the new map number and start position number for the first two special arguments.

In the illustration below, I have created a small rectangular sector within the second room as described in the preceding paragraphs.



Playing The Map

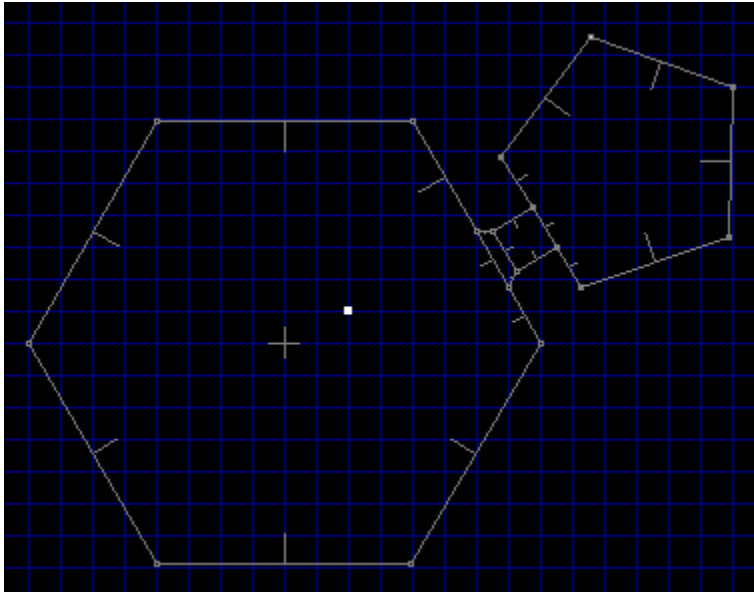
Well, no doubt by now you're getting tired of creating a new map — after all, the fun part is playing it! WadAuthor makes it very easy to test your new map without terminating your editing session. Select the *Run Map...* option from the *Tools* menu and, for now, press the *OK* button in all of the resulting dialog boxes.

WadAuthor will spawn a DOS session and invoke the wadgame specified in the current wadgame configuration file with your new map. Congratulations, you're a WadAuthor!

Creating A Secret Room

{button ,KL(` Properties',0,`,`')} [List of Commands](#)

A secret room is a sort of misnomer; it is usually the entrance to the room that is actually the secret. Setting semantic nitpicking aside, I have created a small map to illustrate making a secret room. Consider the following illustration.



The map above has a large hexagonal room with a door in its northeast wall that leads, through a hallway, into a smaller pentagonal room. Taking the following steps will make the smaller room a secret room.

- n Change the door image (facing into the hexagonal room) to match the surrounding linedefs.
- n Mark the door linedef (facing into the hexagonal room) as secret.
- n Mark the pentagonal sector as secret.



Marking the linedef as secret will only work if the floor/ceiling height of the closed door sector is less than or equal to the floor height of the room sector. If the doorway is a step up from the room, the map will give away the secret by showing the lines immediately beyond the door even before the door is first opened.

Creating A Remote Door

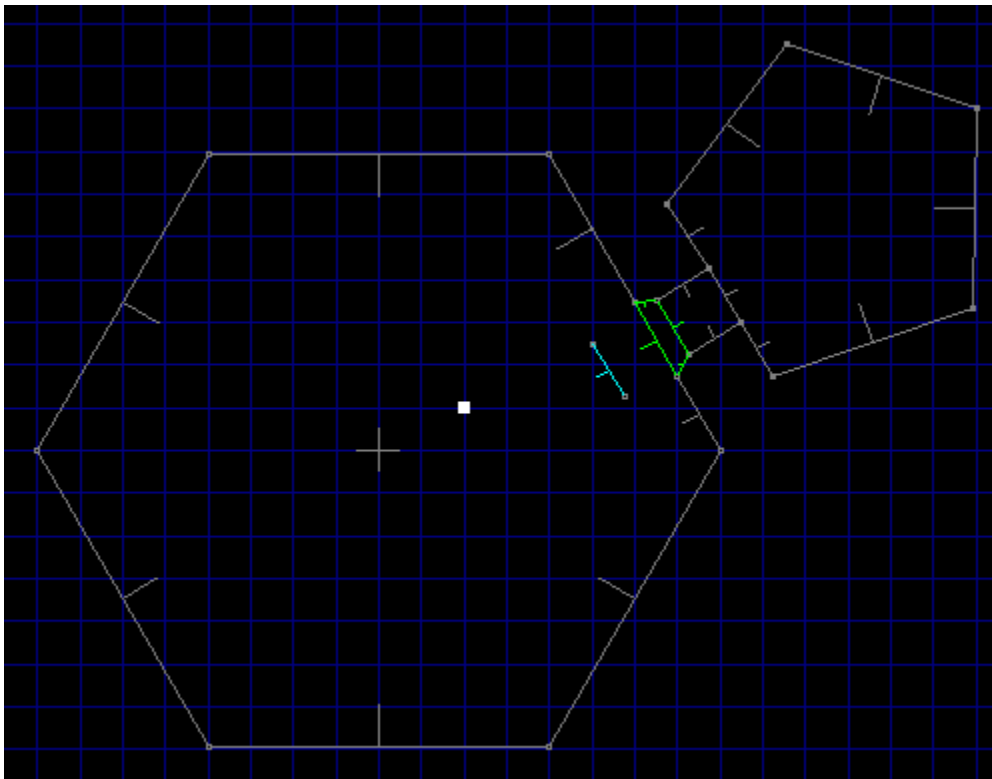
{button ,KL(`Creating A Local Door;Tags',0,`,`')} [List of Commands](#)

Creating a remote door is a simple matter when using WadAuthor. Create a local door, altering the appearance as desired. Invoke the tags dialog, create a new tag, and add the door sector and desired triggering linedef(s) to the list of objects affected. For help creating a normal door or using the tags dialog, consult the related topics.



If you do not wish the player to be able to open remote door locally, be sure to set the door linedef types to normal.

In the following illustration, I have added a single linedef to the map as a trigger for a secret door. The linedef is highlighted as the current object, and the remote door is highlighted as an object with the same tag number.



To add a single linedef to the map, I actually added a new rectangular sector and deleted three of the linedefs. Single linedefs can be a very effective way to provide special effects.

Creating A Teleporter

{button ,KL(`Interface;New Objects;Properties;Tags',0,`,`')} [List of Commands](#)

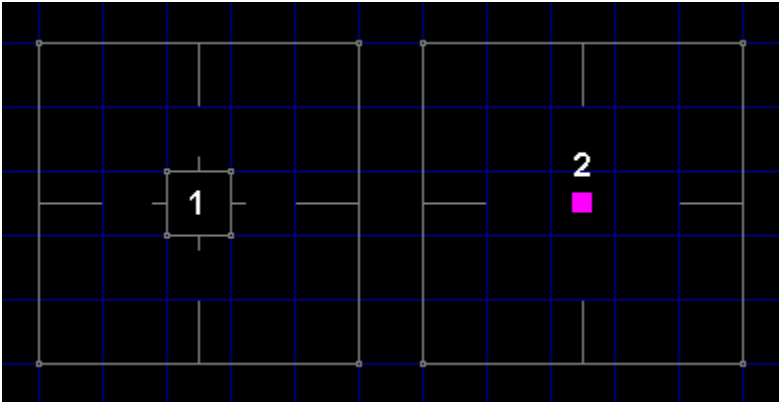
A teleporter, in the most traditional sense (if a traditional sense can even be said to exist for something like this), is a small (usually 64 units square) sub-sector whose linedefs have been set to cause teleportation when crossed. Creating a teleporter, however, also requires a teleport destination thing within a tagged sector. The following steps will create a teleporter platform and destination; for help with the details, consult the related topics.

- n Create a new rectangular sector, optionally setting the floor and ceiling textures or lighting conditions as a visual cue to the player. This will be the teleporter platform.
- n Flip all of the linedefs in the new sector, including textures and other data. To do this, select the new sector and choose the *Flip Linedef(s)...* option from the context menu.
- n Set the teleporter linedefs to the desired action. To do this, edit the teleporter platform linedefs and select the desired teleport type.
- n Create the destination thing. To do this, place a teleport destination within the sector to which the player should be teleported.
- n When creating a teleporter in Hexen, you must tag the teleporter platform linedefs to the destination thing. For all other games, you must tag the teleporter platform linedefs to the destination sector. To do this, invoke the tags dialog, create a new tag, and add the necessary objects to the list.



Always make sure the teleporter platform linedefs are facing the right way. The teleport action occurs when the player crosses the *front* side of a linedef. In the above instructions, flipping all of the teleporter platform linedefs will take care of this.

In the following illustration, I have added a teleporter to allow one-way passage between two otherwise unconnected rectangular rooms. As the player crosses into the teleporter platform at location one, he is immediately teleported to the location and orientation of the destination thing at location two.



Note that the teleporter sector is aligned to match the 64 unit grid. If the teleporter platform floor and ceiling images have a specific pattern, this is necessary to make certain that the pattern appears correctly to the player. For more information about texture mapping, consult the related topics.

Creating A Staircase

{button ,KL(`Interface;New Objects;Stairs',0,`,`')} [List of Commands](#)

WadAuthor automates the process somewhat by allowing you to specify the direction, number of steps, lighting change, floor change, ceiling change, and the various images used to convert a pre-existing rectangular sector into a staircase. Taking the following steps will create a staircase.

- n Create a rectangular sector in the desired location if none already exists.
- n Select the sector to be converted, and choose the *Convert Sector To Stairs...* option from the context menu.
- n Specify the desired number of steps, stair motif, and direction in the resulting convert sector to stairs dialog.



The orientation and attributes of the linedefs are preserved on all sides of the staircase. This can be very important for achieving special effects.

Creating A Rising Staircase

{button ,KL(`Interface;New Objects;Stairs;Tags',0,`,`')} [List of Commands](#)



The Hexen wadgame has pre-defined specials for use in creating rising staircases. The following instructions do not apply to Hexen.

A rising staircase is a staircase that remains hidden until triggered by the player. In architectural terms, it consists of a series of adjacent sectors having certain properties. The first step must be tagged to a linedef whose type is set to raise stairs. When the linedef is triggered, the staircase will rise. For an adjacent sector to rise when the linedef is triggered, the following conditions must be met.

- n The sector must originally have the same floor height as the first step.
- n The sector must be adjacent to a previous step, sharing a linedef with it.
- n The front side of the shared linedef must be facing the previous step, and the linedef must have a lower number than any other linedef facing into the previous step.

Fortunately, WadAuthor makes this fairly simple. The following step by step instructions create a rising staircase.

- n Create a rectangular sector within another sector.
- n Select the new sector and flip all of its linedefs, including textures and other data.
- n Convert the sector to stairs, selecting a motif that has a positive floor height change.
- n Select the step sectors and reset their floor heights to that of the original sector.
- n Tag the first step to the desired linedef and set the linedef type to raise stairs.

A couple of items are worth noting. If a linedef along a side of the staircase faces into a given step sector and has a lower number than any other linedefs in the sector, the results can be surprising. The first step and the surrounding sector will rise, but the remaining stairs will not move. The easiest way to prevent this is by making sure the sides face outward. In the above procedure this is taken care of by flipping all the linedefs.

Always make sure that the last sector terminates the process. This is most easily accomplished by making sure that the last linedef faces out of the last step. In the above procedure this is, once again, taken care of by flipping all the linedefs.

Spiral staircases and other special effects can be achieved by moving the vertexes when finished. Arranging the vertexes in different patterns can make things very interesting for the player. Of course, adding a few nasties at the top of the staircase also works wonders...



If it seems silly to create a rising staircase only to set the floor heights back to the original value, it isn't. WadAuthor analyzes the changing floor height to make sure all the required textures are in place. Using a floor height change of zero will save a single step at the cost of manually applying all of the required textures — without the aid of the check map dialog.

Develop Around A Common Theme

I have played many different third-party maps, and the most enjoyable always have at least one thing in common: they are designed around a common theme. In the same way that following the plot of a well-constructed novel is enjoyable, your audience will appreciate it if your map has a sense of theme and continuity.

A map with sector after sector filled with bad guys to kill and goodies to grab loses its appeal quickly. A map with puzzles to solve, traps to avoid, strategically placed architectural riddles, well-placed monsters, and tactically interesting deathmatch layouts will keep the player coming back again and again.

Consider Multi-Player Modes

When I created my first map, I didn't start thinking about multi-player modes until about halfway through the process. Because of this, I was unable to easily accommodate deathmatch play in my overall scheme. When designing a map to support multi-player modes, remember to keep asking yourself the following questions.

- n Are there enough monsters, weapons, and goodies for each player? Do deathmatch players have easy access to weapons and ammunition near each starting location?
- n If portions of your map can only be accessed by a specific sequence, do they cut deathmatch players off from the rest of the world?
- n Do you have any tricks or traps that require multi-player cooperation? If so, do they make your map impossible for single-player mode?
- n Do your hallways and rooms have enough room for several players to maneuver? Sometimes it can be fun to keep the players from using those heavy weapons as the bad guys are pouring down on them, but it gets old in a big hurry.

Understand The Third Dimension

One of the most enjoyable aspects of wadgame play for me is facing challenging architecture that makes good use of the vertical dimension. Secrets that can only be accessed through jumping, traps that can be avoided by falling over a given linedef, or just making the player take a leap of faith into a dark pit can be very effective. Using these kinds of ideas effectively requires a little understanding, however, to avoid problems.

For starters, wadmaps are not really three-dimensional. The three dimensional effect is achieved by specifying differing altitudes for two dimensional sectors. This has some important ramifications for the wadmap author.

- n Things can be thousands of units apart in the vertical direction, but as far as the game is concerned, they're right next to each other. An unseen enemy can tear a hapless player to shreds from far below because of this.
- n Negative altitudes are possible, but they affect the way certain linedef operations function. For example, linedefs that cause sectors to rise or fall to match the nearest floor or ceiling can be problematic. It seems as if the various game engines resolve the question of which sector is nearest in the positive vertical direction only. If a small sub-sector with a negative altitude is supposed to move down to a slightly lower nearest floor, don't be surprised if it shoots through the roof instead!
- n Early versions of the DOOM engine (1.2 and before?) will crash if the difference between the floor and ceiling altitude is greater than 1000.

Record A Demo

One of the easiest ways to provide help for your audience is to record a demo of your map. The text files that ship with the various wadgames describe the process of recording a demo. Once the demo is finished, you need to insert it into your wadfile with a resource name that varies among the different wadgames.



What's New?

This topic highlights the changes in each new release of WadAuthor. The complete history can always be found in the [About WadAuthor](#) topic, but simply knowing what changed doesn't often help one use those new features. You may click on any of the sections listed below to quickly jump to that section.

[Overview](#)

[Bug Fixes](#)

[Improved Features](#)

[New Features](#)

Overview

The latest release is a minor release. As such, it fixes existing bugs and adds minor features without major alterations from the user's perspective. The list of bug fixes, enhancements and new features follows.

Bug Fixes

- If an external wadfile is used for additional images, it will now be included when running the map.
- Fixed a focus problem with the image browser dialog incremental searching.

Improved Features

- Improved zoom under WindowsNT to a maximum of 400%.
- Improved zoom under Windows95 to a maximum of 250% by limiting map size to 16000 units square.
- Changed the tag dialog to center the map editing view on the selected object.
- Added the make column option to the [new rectangular sector dialog](#) and the [new polygonal sector dialog](#).
- Added the current zoom setting to the status bar.
- The image browser dialog now appends an asterisk to the name of an image if it is not supplied by the main wadfile.
- When running a map, WadAuthor will now create a response file for passing arguments if the command line length is greater than the operating system allows.

New Features

- Added panning on Ctrl+Shift primary click. This feature allows the user to scroll the map view by grabbing a location and dragging rather than by using the scroll bars.
- Added the [run map dialog](#) for customizing options immediately prior to launching the game.
- Added the [tip of the day dialog](#) to provide on-going help and amusement.
- A special version of the WTF Productions DOOM and DOOM][Editing Guide is now included with WadAuthor.

Why Does The Hall Of Mirrors Effect Happen?

As briefly explained in the glossary, this refers to the effect caused by a sidedef without a texture when played within a wadgame. I do not know exactly why it happens (you'd probably have to ask the father of the gaming engine), but reproducing it is, unfortunately, rather easy.

Usually, when you see the hall of mirrors effect, you have a sidedef being viewed from above its ceiling or below its floor without a corresponding texture. The various wadgame engines seem to handle this by not handling it, resulting in the aptly-named hall of mirrors effect.

Why Can't I Just Draw Lines On The Map?

Truthfully, you could if I had taken a different direction in constructing WadAuthor. My favorite wad editor (prior to WadAuthor of course — shameless plug, eh?) allowed the user to create sectors by drawing the linedefs. I got tired of clicking all over the place just to draw a single sector, and I decided that my editor would work differently. I found myself thinking in terms of rooms and the connections between them most of the time; that dictated how WadAuthor works.

There is, actually, another good reason as well: sectors are referenced by sidedefs through linedefs. Maintaining the proper internal data structures in the program was greatly simplified by allowing WadAuthor to do the sector construction. I'm hoping that most people will prefer WadAuthor's methods just as I do.

Why Won't WadAuthor Run With A Sixteen Color Video Driver?

WadAuthor extracts images from the wadgame's main wadfile, and these images are in a 256 color format. While it is a fairly simple thing to display them in color depths greater than 256, it would require the use of a dithering algorithm to present the images in 16 colors with even minimal quality. Because any computer capable of running the wadgames is capable of 256 color presentation, I have chosen to avoid the time spent both in development and speed of execution by requiring a minimum of 256 simultaneous colors.

Why Is There No Sound When Running A Map?

{button ,KL(` Configuration Files',0,`,`')} [Related Topics](#)

The WadAuthor configuration files disable the sound by default to avoid problems. If you are running WadAuthor under Windows95, however, it is quite possible that your hardware is capable of providing sound. To test this, edit the wadgame configuration file with notepad, or some other text editor, and remove the `-nosound` option from the *Run* entry in the *Wadgame* section.



Be sure to make a backup of the original configuration file before editing it.

Can I Customize The Game Graphics?

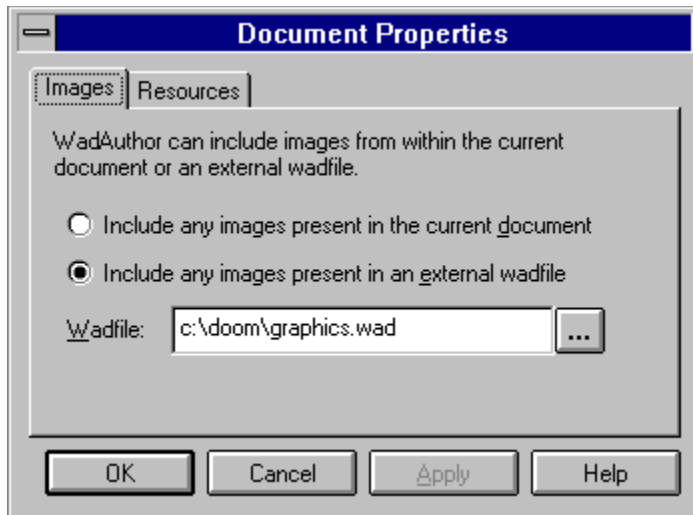
{button ,KL(`Custom Images',0,`,`')} [Related Topics](#)

While WadAuthor does not supply any native functions for creating new images or altering existing images, there are several utilities available that do so. WadAuthor will support the use of additional images through various configuration settings (for more information, consult the related topics).

Document Images Dialog

{button ,KL(`Custom Images;Properties',0,`,`')} [Related Topics](#)

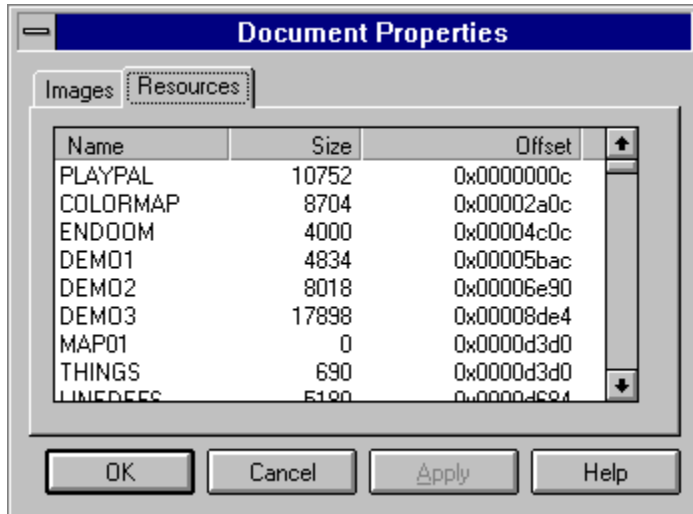
The document properties dialog images page, shown in the illustration below, allows the user to configure how additional images will be handled. By default, WadAuthor will use any images located within the current document. Using a separate wadfile for additional images is supported by checking the appropriate option and supplying the filename.



Document Resources Dialog

{button ,KL(` Properties',0,`,`')} [Related Topics](#)

The document properties dialog resources page, shown in the illustration below, allows the user to browse the list of resources, also known as lumps, in the current document.



Using Custom Images

{button ,KL(`Custom Images',0,`,`')} [Related Topics](#)

WadAuthor supports the use of additional images not present in the main wadgame wadfile. When a wadfile is opened, WadAuthor will default to using any images contained within the wadfile adding to or overriding the images contained in the main wadgame wadfile. If the custom images for a given wadfile are contained in a separate wadfile, for whatever reason, you must tell WadAuthor which wadfile to use for the additional images. You may configure how WadAuthor uses custom images via the document properties dialog.



[Click here for step by step instructions.](#)

The images applied to linedef sidedefs (usually called textures) are actually composed of a number of smaller images (usually called wall patches) which are referenced by number. This allowed the original game designers to avoid wasting space within the main wadfile.

If additional textures are supplied without a corresponding list of panels, WadAuthor will provide a warning message. If the panels present in the main wadgame wadfile are sufficient to create each texture, no problems will occur. If, however, the new textures require new panels, WadAuthor will not be able to display the images. If the warning message appears, it is strongly suggested that you include the panels in the wadfile containing the additional textures.

How To Use Custom Images

- 1 Open the wadfile for which custom images are needed.
- 2 Choose the *Properties* option from the *Edit* menu to display the document properties dialog.
- 3 Change to the *Images* page if necessary.
- 4 Select the desired image handling option. If using images in an external wadfile, be sure to specify a valid file name.

How To Change The Colors

- 1 Choose *Options* from the *Tools* menu to display the options dialog.
- 2 Change to the Colors page if necessary.
- 3 To change a color, double-click its entry in the list, or select it and press the *Set Color* button.

How To Make A Door

- 1 Create and select a new sector (or select an existing sector) to become the door.
- 2 Click the sector with the secondary mouse button to display the context menu.
- 3 Choose the *Convert Sector To Door* option from the context menu.
- 4 Select a motif to be used for creating the door and press the *OK* button.

Note:

If the context menu does not provide the conversion option, make sure you have a single sector selected and make sure it connects to at least one other sector via a two-sided linedef.

How To Make A Staircase

- 1 Create and select a new sector (or select an existing sector) to become the staircase.
- 2 Click the sector with the secondary mouse button to display the context menu.
- 3 Choose the *Convert Sector To Stairs* option from the context menu.
- 4 Select a motif to be used for creating the stairs and press the *OK* button.

Note:

If the context menu does not provide the conversion option, make sure you have a single four-sided sector selected.

How To Align Textures

- 1 Select the linedefs whose textures need to be aligned.
- 2 Click one of the selected linedefs with the secondary mouse button to display the context menu.
- 3 Choose the *Align Textures* option from the context menu.
- 4 Configure the options for sizing information and press the *OK* button.

Tip:

The first texture in the selected list is used for sizing options.

How To Make A Teleporter

- 1 Select the linedef(s) that should cause the teleport.
- 2 Click the linedef(s) with the secondary mouse button to display the context menu.
- 3 Choose the *Properties* option from the context menu.
- 4 Set the linedef type to perform the desired teleport operation and press the *OK* button.
- 5 Click the secondary mouse button within the desired destination sector to display the context menu.
- 6 Choose the *New Thing* option from the context menu.
- 7 Configure the thing to be a teleport destination and press the *OK* button.
- 8 Tag the linedef(s) that cause(s) the teleport to the destination sector.

How To Make A Lift

- 1 Select the linedef(s) that should trigger the lift.
- 2 Click the linedef(s) with the secondary mouse button to display the context menu.
- 3 Choose the *Properties* option from the context menu.
- 4 Set the linedef type to perform the desired lift operation and press the *OK* button.
- 5 Tag the linedef(s) that cause(s) the lift to the lift sector.

Note:

A lift drops down to the nearest floor before rising to its previous position; thus, the lift sector floor must start above the surrounding floor.

How To Tag Objects

- 1 Click one of the objects to be tagged with the secondary mouse button to display the context menu.
- 2 Choose the *Edit Tags* option from the context menu.
- 3 If the object has no tag, press the Yes button when asked if a new tag should be assigned.
- 4 Once the tags dialog is displayed, click the other objects to be tagged with the primary mouse button to add them to the list.

Note:

If you make a mistake and add an undesired item to the list, click it again with the primary mouse button and it will be removed from the list.

How To Edit Objects

- 1 Select the objects to be edited.
- 2 Click one of the selected objects with the secondary mouse button to display the context menu.
- 3 Choose the *Properties* option from the context menu.
- 4 Modify the object parameters in the resulting dialog(s) and press the OK button.

How To Select Objects

- n To select a single object, click it with the primary mouse button.
- n To add or remove an item from the selection, click it with the primary mouse button while holding down the ctrl key.
- n To select a range of objects, click and hold the primary mouse button while holding down the shift key; this will begin a rubberband selection. Drag and release the mouse button to select objects within the rectangle or press the escape key to abort the operation. If you do not drag, the object clicked will be added to the selection.

Tip:

If you're having trouble isolating a single object, try holding down the alt key while clicking; this will provide a list of the objects nearest to the cursor position.

How To Create Objects

- 1 Click the secondary mouse button roughly where the object should be centered to display the context menu.
- 2 Choose the appropriate option to create a new rectangular sector, polygonal sector, or thing.
- 3 Configure the object parameters in the resulting dialog box and press the *OK* button.

Invalid File Name

This error message means that the program cannot use the file name you have supplied. This is usually due to one of the following reasons.

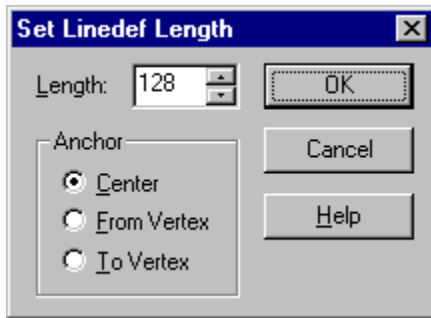
- The name itself is not a valid file name.
- A file having the given name could not be found.
- The file exists but does not contain valid data.

Please check the file name, and try again.

Set Linedef Length(s) Dialog

The linedef length dialog, shown in the illustration below, allows the user to set the length of the selected lines. The anchor is used to determine from which point on the linedef the length will be set. The three options are explained below.

- n **Center** means that the linedef length will be adjusted around the current center point.
- n **From vertex** means that the linedef length will be adjusted from the from vertex.
- n **To vertex** means that the linedef length will be adjusted from the to vertex.



This feature can have unexpected results when used with multiple connected linedefs. When adjusting linedef lengths, the operation proceeds through the selection list one linedef at a time. If any two of the selected linedefs are connected, the vertex connecting them will be adjusted twice.

Make Linedefs Parallel Dialog

The make linedefs parallel dialog box, shown in the illustration below, allows the user to specify options for making one linedef parallel to another. The anchor is used to specify a point on the master linedef with which the moving linedef will be aligned, if any. The four options are explained below.

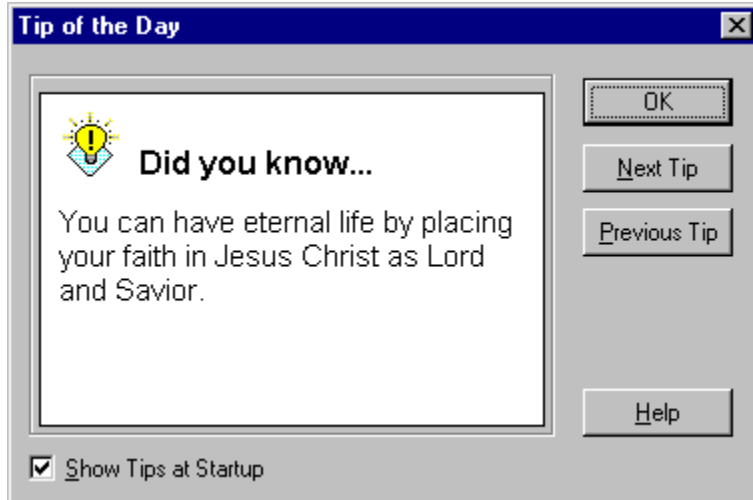
- **Center** means that the moving linedef will be aligned with the center of the master linedef.
- **From vertex** means that the moving linedef will be aligned with the from vertex of the master linedef.
- **To vertex** means that the moving linedef will be aligned with the to vertex of the master linedef.
- **None** means that the moving linedef will be rotated around its current center as necessary.

When using an anchor, the distance field will be used to determine how far from the master linedef the moving linedef should be positioned. For example, when the anchor to center is used, the distance will be the distance from the center of the master linedef to the center of the moving linedef.



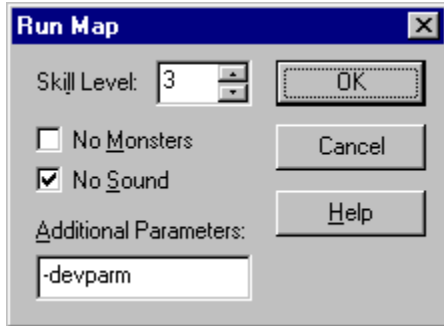
Tip Of The Day Dialog

The tip of the day dialog, shown in the illustration below, provides helpful hints for using WadAuthor and understanding life in general.



Run Map Dialog

The run map dialog, shown in the illustration below, allows the user to customize the game options when running a map. For more information about the additional parameters available for each game, consult the game manual and third party documentation available elsewhere.



To bypass the run map dialog and use the default options, hold down the shift key when invoking the run command.

