220 Map Converter

Version 1.05, November 2013
Developed by Shawn Davison

What it Does:

MapConv is an application built to support the use of newer editors with older Quake engine titles. It is mainly used to convert Version 220 .map files to the format used by games such as HexenII, Quake1, Quake2, and Quake3 (and other titles which use these engines). As well as doing the conversion, it also has support for running user specified compilers on the output .map, helping keep the process as simple and streamlined as possible.

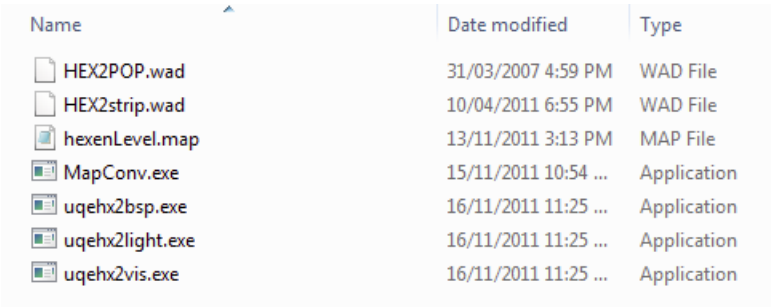
What it Doesn't Do:

The current version of MapConv does still have limitations. While the bulk of functionality is built in, it does not fully support some of the texture alignment options editors working with the 220 format can produce. This means that it is best to avoid rotation and mirroring using texture lock, as well as modifying the align to world/face options. (Move texture lock, and all other alignment methods work fine however. As best practice, just keep to rotating/flipping textures manually.)

Setup (Basic):

To only convert a map, setup is not required (just run the application). If converting a map for a Quake 2/3, you will also need a .tex file in the same directory as MapConv.exe.

To convert and compile, you'll need either the Quake/HexenII format texture wads which are to be compiled into the map or the .tex files for Quake 2/3 games, and the compilers you wish to use in the same directory as MapConv.exe. The compilers can be titled anything as long as they have csg, bsp, light, vis, and map within their names. (Non case sensitive, so bsp, Bsp, or BSP for example, would all work.)



Name	Date modified	Type
HEX2POP.wad	31/03/2007 4:59 PM	WAD File
HEX2strip.wad	10/04/2011 6:55 PM	WAD File
hexenLevel.map	13/11/2011 3:13 PM	MAP File
MapConv.exe	15/11/2011 10:54 ...	Application
uqehx2bsp.exe	16/11/2011 11:25 ...	Application
uqehx2light.exe	16/11/2011 11:25 ...	Application
uqehx2vis.exe	16/11/2011 11:25 ...	Application

Setup (Advanced):

MapConv stores a list of formats which it can convert to within the mapconv.cfg file located within its base directory. With this, custom gametypes can be added, each with an engine type and user specified working directory. These are in the format below:

'*Name ; Engine ; WorkingDirectory*'

-*Name* acts as the name of a gametype and is displayed within the user interface. It is also used as the name of the .tex file if applicable.

-*Engine* determines which engine to convert to for this gametype. Current options are '*hexen*', '*quake*', '*quake2*', and '*quake3*'.

-*WorkingDirectory* allows the user to change where MapConv will save .map files, load .tex files, and run compilers from for a given gametype. This can either be set to a path relative to MapConvs base directory, or a full path. Setting the *WorkingDirectory* as '*none*' will use MapConvs base directory instead. (Working directories are useful to keep files for many different gametypes organized, and also for certain compilers such as q2map which require maps be in a specifically named folder.)

Additional flags can also be set within mapconv.cfg. These are described below:

"reverse_rotation" - Sets weather texture rotations should be reversed or not during conversion. (Default is true) This works independently of the '*Reverse Rotations*' option used when Back Converting.

"remember_mapdir" - When enabled, the last directory a .map file is selected from will be used as the default until MapConv closes. (Default is true) If set to false, the MapConv directory will be used as the default instead.

"use_data_command" - Specifies a data directory taken from the map folders path when running a game. (Default is true) EG '-game fortress +map mapname' if the map directory is ..fortress\maps. (Note that the data command is slightly different for each engine type.)

"rename_mc_entities" - Removes the last two underscores and anything that follows in classnames ending with _mc. (Default is true) EG an entity with the class name of '*item_shells_large_mc*' will be changed to '*item_shells*'.

"ogier_map" - Removes the underscore from '*_angles*' and '*_mangle*' when converting. (Default is false) This can be used if building Quake levels with the Ogier editor.

"modify_texture_scale" - Multiplies texture scale by a specified value and updates x / y alignment accordingly. If set as 0, no change will take place. (Default is 0) This can be used when working with editors which have hard coded default texture scales you wish to override.

"suppress_tex_errors" - Suppresses printed errors when textures in a map being converted do not have a match in the .tex file. (Default is false)

"convert_detail_brushes" - Converts any entitys named '*detail*' to brushes with the detail content flag checked. Only has an effect when converting maps to Quake 2 or Quake 3 engine games. (Default is true)

"detail_content_value" - The content value (integer) used for detail brushes in Quake2 games. (Default is 134217728)

"convert_surfaces" - Converts faces with specific rotation values to light emitting surfaces or presets from the mapconv.cfg file. (Default is false)

"reload_config" - Reloads flags, surfaces, gametypes from the mapconv.cfg file each conversion. (Default is false) This can be useful if you wish to make regular modifications of the cfg file without relaunching MapConv.

Also in this file, custom flag values can be specified which may be applied or added to any surface from within an editor. These offer an additional way to modify textures in Quake2 engine games. (More information on this can be found in the below sections.) These are written as below:

'*surface number ; add or replace existing flags ; flags*'

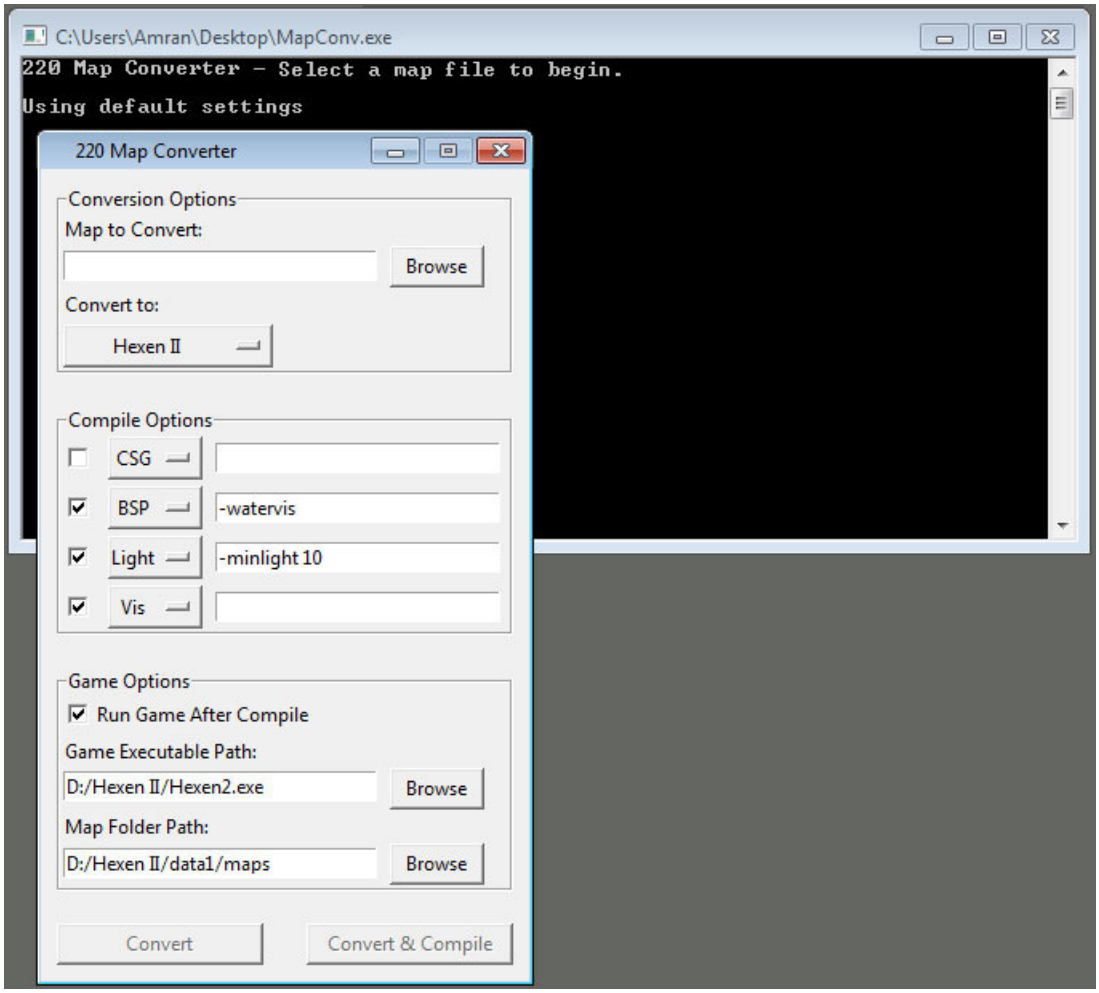
-*surface number* is the value which can be used in editor to link a surface to this set of flags. These can be from '*surf01*' to '*surf99*'.

-*add or replace* offer two different options for how flags will be applied to a surface. '*flags_add*' will add the flag values to any which already exist in the .tex file, while '*flags_replace*' will replace any already specified flags entirely. Placing a minus before a flag value and setting it to '*flags_add*' can also be used to decrease the values of an already existing set of flags.

-*flags* is the actual value to be added or replaced. EG, *0 1 200*.

Usage:

When you open MapConv, you'll see both a command window used for displaying information about the conversion and compilation processes, as well as a GUI window for user input.



"Map to Convert" - The file path to the version 220 map to be converted or compiled. Use the browse button to select a map file, or type in the path.

"Convert to" - The destination map type. Currently supports Quake, Quake2, and Quake3 engines and variants.

"Compile Options" - Which compilers, if any, to run on the map after conversion. Compilers should either be in the same directory as the application or in the user specified working directory. These must contain csg, bsp, light, vis, and map within their names (non case sensitive). A text field for entering optional arguments is included next to each compile option.

"Run Game After Compile" - If selected, the game will run after compilation has finished and the bsp will be loaded. The '*Game Executable Path*' should point to either the exe or a shortcut file of the game (MapConv is able to pull out and use any args specified in shortcuts). The '*Map Folder Path*' is where the compiled bsp will be copied to before being loaded. (Note that by default MapConv will automatically set the data folder argument based on where this map folder is located. EG if the map folder path is '*D:/Hexen II/data1/maps*', the game would be run with '*-data1 +map mapname.bsp*'.)

"Convert, Convert & Compile" - Convert only converts a map to the destination format, placing the output in the same directory as the application or the user specified working directory. Convert & Compile works similarly, but runs the above specified compilers on the output file following conversion.

Saving & Loading Settings - MapConvs currently chosen settings can be saved to the config file by pressing the '*F1*' key. These settings will automatically be loaded the next time the application is run. Pressing the '*F2*' key at any time will temporarily reset all options to default.

Behind the Scenes:

So, what exactly happens during the conversion process?

- Removes MapVersion220 key from the worldspawn if it exists.
- Corrects the worldspawn's spawnflags line if map contains any. (From '*spawnflags2*' to '*spawnflags*'.)
- Removes full file paths of each wad in the wad key, also replaces '*.hlwad*' with '*.wad*' in any of the file names. The wad key is removed entirely if being converted to a Quake2 or Quake3 type file. (Example: *'\sdk\hex2pop.hlwad;\sdk\hex2.hlwad'* becomes *'hex2pop.wad;hex2.wad'*) Note that not all compilers support multiple wads per map.
- Removes underscore from '*_angles*' and '*_mangle*' if the ogier_map cfg flag is set.
- Renames any entitys which use classnames ending in '*_mc*' if the rename_mc_entities cfg flag is set.
- Texture scale and alignment are updated if the modify_texture_scale is set in the .cfg.
- Performs conversion from 220 to old format on each line containing texture information. This involves separating and reordering the texture alignment values within the .map file while also applying the below game specific changes. Note that any x or y alignment which contains a decimal will be rounded to the nearest whole number. All texture rotations will also be reversed unless otherwise specified.

  Quake Engine
  -No additional changes.

  Hexen II
  -An extra entry of -1 is appended to the end of each line containing texture information.

  Quake2 & Quake3 Engine
  -Texture names are replaced with full paths from the .tex file.
  -Any extra information specified for a specific texture in the .tex file (surface flags, etc) is inserted following the line where it appears.
  -When using convert_detail_brushes, converts any entity in the map named '*detail*' to a standard brush and adds the '*detail*' content flag to its surfaces.
  -When using convert_surfaces, applies additional custom flags to any face with specific rotation values set.

- Saves the converted map file either in the same directory as the MapConv application, or in the user specified working directory.
- Runs selected compilers, if any, on the converted map file. Texture wads and .tex files are loaded from this same directory as the map.
- If set to run after compile, copies the bsp to the maps folder before loading the game and level. (This will include any arguments specified by a .lnk (shortcut) file, as well as a data command and map name by default.)

Tex Files:

The versions of Hammer which MapConv is intended to work with have two setbacks when working with Quake2 and Quake3 engine levels: Texture names cannot be longer than 15 characters, and no support for setting surface flags within the editor. Both of these issues are worked around using .tex files which are loaded during the conversion process.

A typical line within a .tex file contains the name of a texture as is displayed in the .hlwad, followed by a semicolon and the full texture path. When converting, any instance of the .hlwad name will be replaced with the full path.
EG: *01brtrim1;Andoria/brtrim1;* (Replaces *01brtrim1* with *Andoria/brtrim1* during conversion.)

Additional values can be placed following the second semicolon which are to be appended to the line in the converted file. This is used for setting surface flags, light values, and etc which are linked to that texture. In the format of '*0 0 0*', either(or both) of the first two 0 s should be replaced by the sum of the flags wanted, while the third 0 is used to contain a value for surfaces with flags such as '*light*' or '*anim speed*'.
EG: *m_chain;Mines/chain;134217728 16781312 5* (Replaces *m_chain* with *Mines/chain* during conversion, and adds the flag values.)

Lists of these flags and further documentation can be found within the provided .tex files. (Open using Notepad or another text editor.)

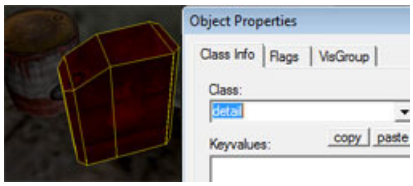
Generating Tex Files:

There is a built in feature to generate .tex files within MapConv, press '*F4*' to use this. The resulting file will be written to '*generatedfile.tex*', while any errors will be printed in the console (these are also added to the file in the form of comments).

> **"Should prefixes be added to the texture names?"** - If '*Yes*' is selected, textures will be prefixed with '*00*' for the base directory, and '*01*', '*02*', '*03*', etc for each additional directory. This prevents duplicate texture names.
>
> **"Should the source texture files be renamed?"** - If '*Yes*' is selected, the texture files in the chosen directory will be renamed to contain the generated prefixes. This makes them easy to quickly build a matching .hlwad from.

Note that a notex texture is also added near the top of .tex files. Using this texture in a map on non visible surfaces will decrease the conversion time.

Detail Brushes:

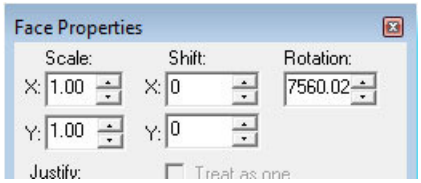


Both the Quake2 and Quake3 engine support detail brushes specified using content flags. A shortcut is built into MapConv to make creating such objects easy.

First select any brush or brushes, and then make them into an entity. The entity type should be '*detail*', and should not have any keys specified. MapConv will automatically convert these to standard brushes and add the detail flag when you convert a map to either a Quake2 or Quake3 engine type.

It's best to avoid using this with any textures which already have the detail flag specified in a customized .tex file, as the content value will not be correct after conversion.

Light Emitting Textures & Changing Surface Flags:



Tex files are one way of adding flags to surfaces, but an alternate method of specifying these values from within the editor using the rotation field also exists. When rotating a texture, the first portion of the rotation can be used to trigger an effect, while the two decimal numbers act as a value. This system may be a bit unwieldy, but it does offer a way to more quickly adjust surface flags while working on levels for Quake2 engine games. These trigger values function on an increase or decrease of 360 degrees, so they can also be added to any rotated textures without changing their in-editor appearance.

EG: A trigger might be a rotation of *7560*. If you have a surface that already has a rotation of *45* degrees, this can be added together as *7605*. Negative values work the same way, so if a surface has a rotation of *-45* degrees, *-7605* could be used.

A rotation of *7560* or larger (up to *7919*) indicates that the flags on a surface should be changed. A decimal value which matches a surf entry from mapconfig.cfg indicates which flags to add/replace/remove, while no decimal places means all flags should be removed.

EG:
7560 -> Removes all flags
7560.01 -> Applies *surf01*
7560.20 -> Applies *surf20*

Using this same concept, a system is also included to quickly add/modify/remove light emissive surfaces. A rotation of *720* or larger is used for this, while every additional 360 degrees adds another 0 to the end of the light value. (Alternately, another way to think of this is that a light value will have as many digits as time the rotation can be divided by 360.) A decimal value is used as the base light value, while no decimal places means the light flag should be removed.

EG:
720 -> Removes light flag and value
720.01 -> Sets light to *01*
720.10 -> Sets light to *10*
1440.25 -> Sets light to *2500*
2160.50 -> Sets light to *500000*

If a surface is not already set as emissive, this will enable the light flag and place in the light value. If being applied to a surface that already has emissive properties, the light value will simply be updated to the new one.

Worldspawn Spawnflags:

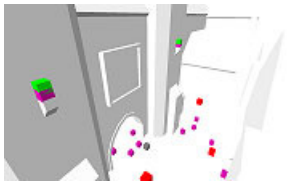
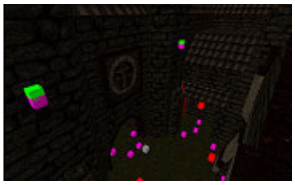
Certain games allow for worldspawn spawnflags, a feature that some editors do not support. To enable a spawnflag in these cases, add a key to the worldspawn called '*spawnflags2*'. MapConv automatically corrects this to '*spawnflags*' when converting.

Using MapConv (_mc) Entities:

MC entities have a special suffix added to them which is removed during conversion while the rename_mc_entitys flag is set within the cfg. This can be utilized to create customized versions of entities and has been used within some of the provided fdg files to display alternate models (such as having entries for both the small and large version of ammunition packs in Quake1, each with a different .mdl being loaded). When picked up by the converter, an entity suffixed by '*_mc*' will have the last two portions of the classname removed.

EG:
An entity with the classname of '*item_shells_large_mc*' will be changed to '*item_shells*' during conversion.

Converting Texture Information From Full Paths (Back Converting):

An additional advanced feature is the ability to convert texture information in both old style and v220 files map files back to the original values specified in a .tex file. Press '*F3*' after setting a map file in the '*Map to Convert*' field and selecting a gametype to use this. This is helpful if you either want to load an original game level into Hammer, or if you wish to switch back and forth between using Hammer and Radiant on a level.

"Reverse Rotations" - Selecting this will reverse the rotations for each surface in a file.

"Ignore Surface Flags" - This check box disables the matching of surface/content flags and will instead link to the first like texture found. This is useful when back converting a map which you do not have a customized .tex file for.

EG: A .tex file has the following entries:
16sftwd033lit;misc/sftwd033;0 1 100
16sftwd033;misc/sftwd033;
16sftwdtrans;misc/sftwd033;0 32 0

By default, any instance of *misc/sftwd033;0 32 0* would be matched to *16sftwdtrans*, however if this flag is checked, it would match to *16sftwd033lit* instead.

"Suppress Tex Errors" -This check box disables error printing regarding missing .tex entries during the back conversion process.

"Import Rotations" - Allows for importing of a v220 .map file which rotations will be restored from. This is meant for cases where custom flags have been set using the rotation field, but have been reset while working on the map in another editor. (EG, taking a map which has rotations using decimal places and re-saving it with an editor such as an old version of Radiant may lose those decimal values. This option can be used to restore these.) Note that this simply matches the position of brush faces in the file, meaning it's best to avoid cloning, adding, or converting brushes to entitys before back converting when using this feature (though this may vary on a per editor basis, modifications to existing objects, entity work, and creating point entities should be fine).

"Source Type" - The type of map file being entered for back conversion. '*Old Format*' is for old style maps (such as those saved by Radiant or Worldcraft 1.6), while '*v220 Format*' is for v220 maps (such as those saved by Hammer). An error stating incorrect file type will be shown if this is not set properly.