UQE Hexen II

Installation Notes - v1.18

Patching Hexen II

This section applies if you do not have the mission pack.
Make sure that your commercial version of *Hexen II* is patched to the latest official version. If your version of *Hexen II* is not v1.11 you need to run the patch file "*ph2v111.exe*".

Decompress the *UQE Hexen II* archive to a clean folder. (ex: "*c:\hexen2*")

Next we need to copy the original (source) *Hexen II* data files from its original location to the newly decompressed location of (destination) *UQE Hexen II*.

The following files needs to be copied:
[source]/data1/pak0.pak > [destination]/data1/pak0.pak
[source]/data1/pak1.pak > [destination]/data1/pak1.pak

With these actions performed your copy of *Hexen II* is completely upgraded to *UQE Hexen II*.

Patching Hexen II: Portal of Praevus

This section applies if you have the mission pack.
Your commercial version of *Hexen II* & the *Portal of Praevus* mission pack should already be patched to v.1.11 when the mission pack were installed.

Decompress the *UQE Hexen II* archive to a clean folder. (ex: "*c:\hexen2*")

Next we need to copy the original (source) *Hexen II* data files from its original location to the newly decompressed location of (destination) *UQE Hexen II*.

The following files needs to be copied:
[source]/data1/pak0.pak > [destination]/data1/pak0.pak
[source]/data1/pak1.pak > [destination]/data1/pak1.pak
[source]/portals/pak0.pak > [destination]/portals/pak0.pak

With these actions performed your copy of *Hexen II: Portal of Praevus* is completely upgraded to *UQE Hexen II*.

Engine Features

Using OGG, MP3 or WAV music files

Supported in the engine is the ability to execute OGG, MP3 and WAV playback for music. The engine takes preference when loading music files in the order of OGG, MP3, WAV if the base file name is found to be the same.

Under the "*data1*" / "*portals*" game data directories or your own custom game data directories, create a "*music*" directory.

A good idea would be to convert your copy of *Hexen II* and *Hexen II: Portal of Praevus* CD Audio to OGG. Name the *Hexen II* CD Audio OGG files from "*track02.ogg*" to "*track17.ogg*". Name the *Hexen II: Portal of Praevus* CD Audio OGG files from "*mp_track02.ogg*" to "*mp_track12.ogg*". Place these files in your respective "*music*" directories. You could also package the files into PAK files. The engine will auto-select the music type with a priority given to file-based playback, if available.

[mp_][track][02][.ogg] = [mission pack indicator][track][track number][format]

Using Quake2-style skyboxes

The method to implement a skybox is exactly the same as how it's done in Quake 2.

Under the "*data1*" or "*portals*" game data directories or your own custom game data directories, create a "*skies*" directory.

Skyboxes works with custom levels. Add a new key namely "*sky*" to your "*worldspawn*" entity with a string value of "*trent*" for example. In the "*skies*" directory, place the six skybox textures with the naming conventions "*trent_bk.tga*" (back), "*trent_dn.tga*" (down), "*trent_ft.tga*" (front), "*trent_lf.tga*" (left), "*trent_rt.tga*" (right), "*trent_up.tga*" (up). The engine supports skybox formats TGA and JPG with a width and height up to 1024x1024.

[trent][_][bk][.tga] = [sky name][seperator][direction][format]

Using External BSP Textures

Under the "*data1*" or "*portals*" game data directories or your own custom game data directories, create a "*textures*" directory.

The engine supports two texture formats namely TGA and JPG.

There's a few special naming conventions to be taken note of when adding new external textures. Warping textures like water textures normally a named for example "**water1*" inside the BSP. To replace the internal texture with an external one, name the texture file "*#water1.jpg*". The "#" sign indicates to the engine that it's a warped texture.

The JPEG library used by Quake III Arena was ported to the engine and there was found that certain JPG files have a tendency to crash the engine. If this does happen, please make sure the JPG files are being encoded using a standard and generally supported format. Alternatively use TGA files as they tend to load much faster.

[#][water1][.jpg] = [warp indicator][original name][format]

Animated textures with 4 animation frames like for example "*rune1*" can be represented as external textures as "*+0rune1.jpg*" (frame 1), "*+1rune1.jpg*" (frame 2), "*+2rune1.jpg*" (frame 3), "*+3rune1.jpg*" (frame 4). The "+" sign indicates to the engine that it's an animated texture and the number following is the frame number.

[+][0][rune1][.jpg] = [animation indicator][frame number][original name][format]

Luma textures gives surfaces a glowing effect that does not get affected by any existing lightmaps creating the illusion of light emanating from the texture. To use a luma texture for the texture "*rtex395.tga*" all that's required is the luma texture needs to be named "*rtex395_luma.tga*".

Using External Colored Lightmaps

Under the "*data1*" or "*portals*" game data directories or your own custom game data directories, create a "*maps*" directory.

Place any colored lightmap files (.lit) in the "*maps*" directory of the appropriate game data directory. If the CVAR "*gl_coloredlight*" is set to a value of "1" the engine will load the appropriate .LIT file that contains the colored lightmap information for the loaded level (map). The .LIT files should carry the same name as their matching .BSP level files.

Using External MDL Skins

Under the "*data1*" or "*portals*" game data directories or your own custom game data directories, create a "*models*" directory.

The engine supports two skin formats namely TGA and JPG.

There's also naming conventions to be taken note of when adding new external skins. The Archer will be used as an example using its "*archer.mdl*" model with two skins namely "*skin0*" and "*skin1*". To replace the internal MDL skins with external ones, name the skin files "*archer_0.jpg*" (skin0), "*archer_1.jpg*" (skin1).

[archer][_][0][.jpg] = [original name][seperator][skin number][format]

Using External Sprite Textures

Under the "*data1*" or "*portals*" game data directories or your own custom game data directories, create a "*sprites*" directory.

The engine supports two sprite texture formats namely TGA and JPG.

Important to note is that the external sprite texture frames should be identical in width and height per-frame, since the engine actually replaces each internal sprite texture frame pixel with the external sprite texture frame pixel. In a future release of the engine real 32bit sprite support will be added via a custom SPR32 format.

There's also a naming conventions to be taken note of when adding new external sprite textures. The yellow spark sprite will be used as an example using its "*spark.spr*" sprite file with its ten animation frames. To replace the internal sprite textures with external ones, name the sprite textures from "*spark_0.tga*" (frame 1) to "*spark_9.tga*" (frame 10).

[spark][_][0][.tga] = [original name][seperator][frame number][format]

Using PAK and PK3 (zip) files

The engine can load any content from PAK or PK3 files, even if they are mixed. Take note that identical content found in both PAK and PK3 files will result in the matching content being loaded from the PK3 instead of the PAK as PK3 files are loaded after PAK files irrespective of its base file names. Base file name order only gets taken into account where the file extensions match.

To ensure loading order and engine stability, please note that all original base game content plus original UQE base data should all be either PAK or PK3 data. Anything else being added can be in either PAK or PK3 format. Its recommended not to mix PAK and PK3 files if you encounter any engine stability issues resulting from a mixture of PAK and PK3 files.

Using Classic Rendering Mode

Since the software renderer has been omitted as from version 1.15 an alternative method has been developed for the OpenGL renderer to emulate the look and feel the dated software renderer provided.

To enable the classic rendering mode, make sure the GPU driver settings does not force any settings related to textures filtering and anti-aliasing. Once that is done, on the options menu change the texture mode to "*point sampled*". To complete the experience, start-up the engine at a lower resolution like 640x480.

Toolset Features

Extracting Textures from BSP Files

Using the tool *uqehx2bsp.exe*, there are two methods available to extract internal BSP textures to external WAL files.

The first method is to use the "*-textures*" command line switch. Using this switch will extract internal BSP textures to external WAL textures when in map compile-time. The new WAL textures will be placed in a directory called "*textures*".

The second method, "*-onlytextures*", is similar to using the only entities command line switch, with the only difference is that it only reads an existing BSP file (eg: no map source) and only extract the texture information from it. The new WAL textures will be placed in a directory called "*textures*". There is no in-engine support to use the format.

Minimum Lighting Lightmap Options

Using the tool *uqehx2light.exe*, there are several options available to manipulate light in the map you are lighting. To create a minimum light level in your map, use the command line switch "*-minlight 200*", where 200 is the minimum light level. Another way in achieving this is by adding a new key called "*light*" to the "*worldspawn*" entity with a value of 200, which will be the minimum light level.

Its also possible to add negative lighting to the world by using the command line switch "*-nominlimit*". By not capping lower light levels it allows for negative light values to be compiled.

Console Commands

cmdlist

Lists all console commands that are available for use.

cvarlist

Lists all console variables that are available for use.

fmod_restart

Stops any playback and restarts FMOD sound system.

fmod_playmusic

Start audio file playback. fmod_playmusic *[filename]*

fmod_stopmusic

Stop audio file playback.

fmod_pausemusic

Pauses the currently playing audio file.

fmod_resumemusic

Resumes playback of the currently paused audio file.

Console Variables

cl_mlook

Replaces the *mlook* (mouse look) command and can also be set on the options menu. This console variable is also better known as *free look* in later Quake engines like Quake 2. Default "*1*". cl_mlook *[0 or 1]* (archiving)

cl_slook

Also known as *slope look*. This console variable enables or disables the ability to adjust the player's viewpoint according to a upwards or downwards sloping part of a map in front of the player. Default "*1*". cl_slook *[0 or 1]* (archiving)

gamma

Hardware gamma control (brightness). Based on Quake III Arena gamma control. Default: "*2.35*". gamma *[0.5 to 2.75]* (archiving)

gl_overbrightbits

Sets the amount of gamma overbright bits to use. Default: "*0*". gl_overbright *[0 to 2]* (archiving)

gl_skytype

Determines whether to render the standard scrolling sky or render the skybox. Where *0* renders the classic scrolling sky and *1* renders the skybox. See "*Using Quake2-style skyboxes*" in the "*Engine Features*" section on how to setup a skybox. Default: "*0*". gl_skytype *[0 or 1]* (archiving)

gl_texturemode

Sets texture modes from point sampled to the highest anisotropic filtering your graphics processing unit (GPU) supports. Default: "*0*". gl_texturemode *[0 to *]* (archiving)

gl_wireframe

Sets world rendering mode from textured to wireframe mode. Default: "*0*". gl_wireframe *[0 to 1]*

r_fps

Displays the current Frames Per Second (FPS) rate. Default: "*0*". r_fps *[0 or 1]* (archiving)

r_dynamic_sidemark

Blocks most of any dynamic lights from shining through walls by testing if a dynamic light source's origin is infront of the plane of a face that is being considered for dynamic lighting. The current implementation doesn't yet consider any visibility list data. Default: "*0*". r_dynamic_sidemark *[0 or 1]*

gl_lumatex_render

Enables or disables luma texture rendering. Default: "*0*". gl_lumatex_render *[0 or 1]* (archiving)

gl_texture_non_power_of_two

Enables or disables whether non-pow2 are acceptable. Default: "*1*". gl_texture_non_power_of_two *[0 or 1]* (archiving)

cl_maxfps

Sets the maximum frames per second rendering rate. Default: "*110*". cl_maxfps *[1 to 500]* (archiving)

sv_fps

Sets the server's frame rate. Default: "*20*". sv_fps *[1 to 20]* (archiving)

gl_coloredlight

Enables or disables colored lightmap rendering. Default: "*0*". gl_coloredlight *[0 or 1]* (archiving)

gl_overbright

Enables or disables lightmap overbright rendering. Default: "*1*". gl_overbright *[0 or 1]* (archiving)

r_particle_dot

Renders non-specific particles as a dot (standard *Hexen II*) when value is set to 1. Default: "*1*".
r_particle_dot *[0 or 1]* (archiving)

Runtime Environment

If there are issues getting the engine to start the “Visual C++ Redistributable for Visual Studio 2015” package installation may be required.
https://www.microsoft.com/en-us/download/details.aspx?id=48145

Thanks

A special thank you goes to Lyubomir Ivanov for helping with rigorous playtesting and identifying bugs.
https://quake.fandom.com/wiki/User:Vorknkx

A huge thank you goes to Ozkan Sezer for providing gamecode and entity positioning fixes from his engine source port project, Hammer of Thyrion (v1.5.7).
http://uhexen2.sourceforge.net

Ultimate Quake Engine

Jacques Krige
https://www.jacqueskrige.site
https://www.moddb.com/mods/uqe-hexen-ii
https://github.com/jacqueskrige/uqe-hexen-ii
https://quake.fandom.com/wiki/Ultimate_Quake_Engine