Unreal Tournament

Audio and Visuals Tweaking Guide

Document revision:	1.5
Author:	Raynor.Z
Publishing date:	March 2013

Latest version available at www.ut99.org community portal.

Foreword

This guide is for all the Unreal Tournament players who like to get just a bit more visual and audio quality out of this masterpiece. This guide will cover clean installing Unreal Tournament (from now on referred as UT), updating the game with essential patches and then tweaking numerous graphical options that are possible thanks to community created video renderers. I will also cover some UT audio settings that should improve sound quality and fidelity without causing distorted or otherwise abnormal side effects. This guide is divided into sections for easier reading and each section covers one specific area. This unofficial guide is provided „as is" without any further warraties. I apologize for any grammatical or spelling errors, because english isn't my native language, therefore some mistakes can happen.

Thank You.

Introduction

Unreal Tournament is already over a decade old, but regardless of that, still popular. It has tons of community created content that has greatly helped extending it's lifecycle. While gameplay has lasted, visuals may have not so well. Fear not, UT community has some really talented individuals, who have provided us new feature rich video renderers that will provide additional visual effects, stability, operating system compatibility and pefrormance. And what's most important, all tweaks provided in this guide are 100% online compatible so you can jump straight into action without worrying about package mismatch errors etc.

Things you will need:

- Unreal Tournament game discs (CD1 and CD2)
- Some technical knowledge
- A bit of time and patience ☺

Unreal Tournament works perfectly on modern hardware and OS (operating system) environments if configured properly. I have tested UT on Windows XP, Windows Vista, Windows 7 and Windows 8 platforms (both 32-bit and 64-bit versions) and they all performed equally fine.

Important Things to Know

Here are some important things to acknowledge, before you continue:

- UT has been tested and works fine with modern Windows operating systems (and of course these include Windows 7 and Windows 8).

- In some unfortunate cases (that more likely applies for systems with AMD CPUs and laptop computers) you need to disable AMD Cool'n'Quiet or Intel SpeedStep features from system BIOS to maintain smooth gameplay (where game does not exceed normal framerates so it will run at hectic and inconstant speeds). It happens because the nature of first UT engine was coded. Back in 1999 all home PC-s were single-core systems and didn't automatically adjust CPU clock speeds. UT1 engine has no idea about multi-core CPUs, therefore the occasional gameplay speed issues.

- This guide will cover clean installing UT using original game CD1 and CD2. Today you can also buy UT from various digital channels (including Good Old Games and Steam), but since I do not have these versions available, I cannot comment how much and in which way they are different from retail versions.

- Definitely do NOT use S3TC textures for mapping. Only use standard CD1 textures for mapping. If you deal with level creation, it is recommanded that you keep two separate UT folders – one for playing and another one for mapping.

- As a precaution I would not recommand downloading any non-official, pre-configured or so called „optimized“ UT versions from the internet, because you can never be sure what do you actually get and what kind of settings are applied to that package (in worst case scenario it may contain malware).

Part One

Installing and Patching UT

Unreal Tournament comes in two discs. First one contains UT game itself, second one contains higher resolution S3TC textures (GOTY version has also ChaosUT and Rocket Arena mods included although they are outdated and you can download updated versions from the internet).

Install Unreal Tournament game itself from CD1.

Right after your UT install is complete, be sure to patch your game. The amount of patches you need to install depends on what version of UT you have. **If you got original 1999 release, then you need to download and install the following:**

UT patch version 436. This is the latest official patch for clients. There is also patch 451, but it is for servers and not for clients playing online.

http://files.filefront.com/UT+436+No+Delta+Patch/;1405841;/fileinfo.html

UT Bonus Pack 1

http://www.ut-files.com/index.php?dir=Patches/&file=utbonuspack.zip

UT Bonus Pack 2

http://files.filefront.com/UTBonusPack2zip/;1405825;/fileinfo.html

UT Bonus Pack 3

http://files.filefront.com/UT+Windows+Bonus+Pack+3+The+Inoxx+Pack/;2130744;/fileinfo.html

UT Bonus Pack 4

http://files.filefront.com/Unreal+Tournament+Bonus+Pack+4/;1405828;/fileinfo.html

If you got UT Game of the Year Edition, then you just need to download Bonus Pack 4, because bonus packs 1-3 are already included. Some older GOTY versions are version 432. In this case you need also patch 436.

Part Two

Installing Higher Resolution S3TC Textures

In this section we are going to install **higher resolution S3TC textures from CD2**, but before you do that be sure to **back up your current City.utx, ShaneSky.utx and SkyBox.utx files from your UnrealTournament\Texures folder**, because they're improperly masked and you get some malformed skyboxes. Put those three files into some temporary folder until you need them.

Right after you have installed S3TC texures from CD2 go to your UnrealTournament\Texures folder and take out the following files and put them besides other previously backed up files - **Female1Skins.utx**, **Female2Skins.utx**, **Male1Skins.utx**, **Male2Skins.utx**, **Male3Skins.utx** and **SkTrooperSkins.utx**.

The reason you'll do that is because sometimes you get „sliding zombie players“ effect in some improperly configured servers which is visually really annoying bug.

Now **download S3TC_Fix.zip**, extract it to UnrealTournament\Texures folder and run S3Fix.exe (Depending on your UT install location, for example Program Files folder, **you might need to run this utility as administrator** (right click and select „Run as administrator“ option). This utility will correct that „sliding zombie players“ bug by patching some textures.

http://www.ut99.org/download/file.php?id=52

After the process is complete (Command Prompt window closes after patching is finished), be sure to **take all your previously backed up files and move them into Textures folder answering „Yes“ to overwrite existing ones**.

Part Three

System Updates and New OpenGL Renderer

Be sure to have up-to-date video card drivers for latest performance enhancements and bug fixes.

For AMD (ATI) drivers, go to:

www.amd.com

and for NVIDIA drivers, go to:

www.nvidia.com

I'ts also a good practice to have latest DirectX runtime libraries installed. You can get the installer from

http://www.microsoft.com/en-us/download/details.aspx?id=35

After that download updated video renderer for UT that has capability to add extra visual details and graphical options, that including S3TC textures support. I particularly recommand Chris Dohnal's OpenGL renderer. There are also other renderers like Direct3D8 and Direct3D9, but at this point OpenGL is most feature rich and configurable.

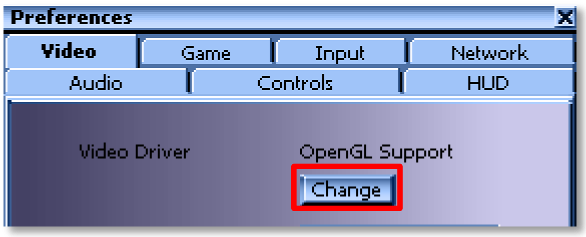
You can get OpenGL version 3.7 from here:

http://www.cwdohnal.com/utglr

and look for a file named **utglr37.zip**.

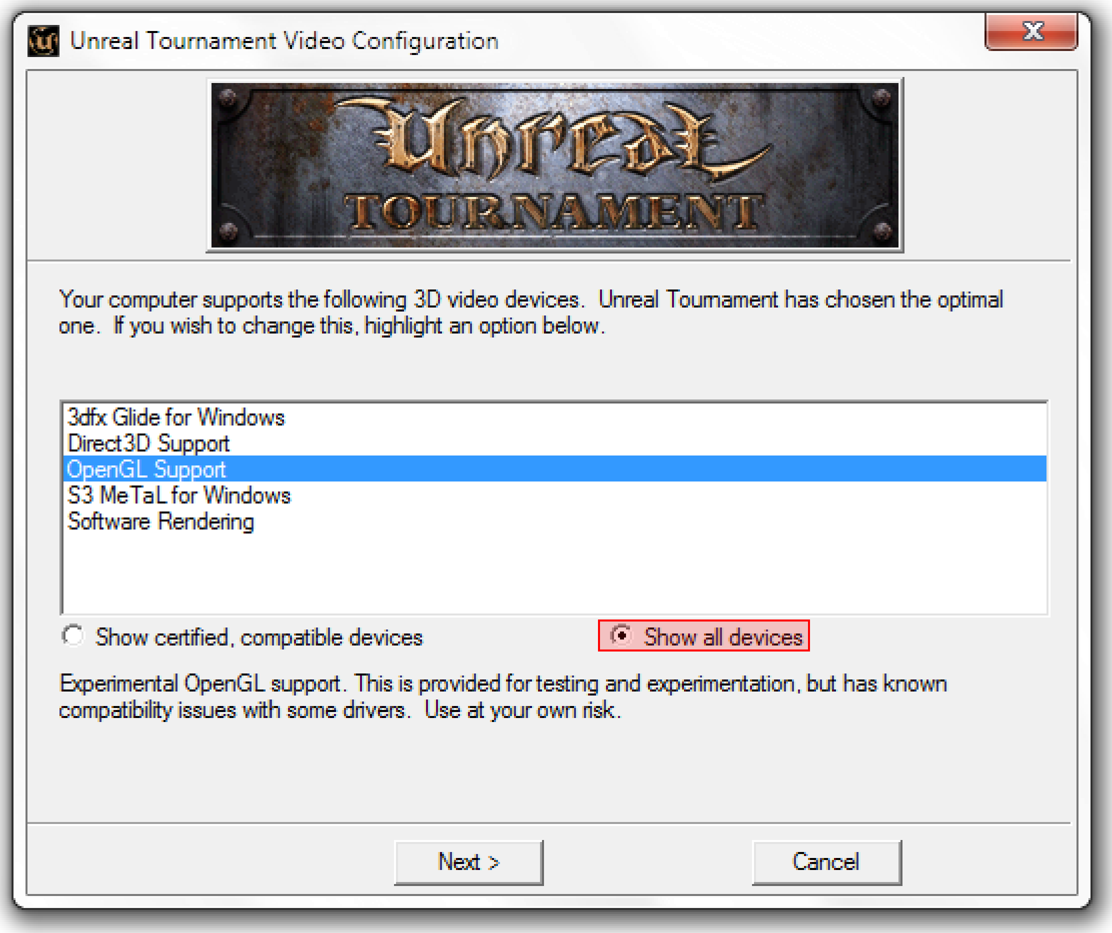
After that, **extract its contents to UnrealTournament\System** and overwrite existing one.

Now it is time to start up UT and configure it to use just installed OpenGL renderer. If you have already passed first run configuration screen, you can change video renderer by going **Options → Preferences → Video** and click on **Change** button.



Video Driver OpenGL Support

On **First-Time Configuration** screen, select **OpenGL**. If you cannot see this option, make sure you have **Show all devices** radio button selected.


Unreal Tournament Video Configuration

Your computer supports the following 3D video devices. Unreal Tournament has chosen the optimal one. If you wish to change this, highlight an option below.

3dfx Glide for Windows
Direct3D Support
OpenGL Support
S3 MeTaL for Windows
Software Rendering

Show certified, compatible devices [✓ Show all devices]

Experimental OpenGL support. This is provided for testing and experimentation, but has known compatibility issues with some drivers. Use at your own risk.

You can always later change the renderer from the in-game the Options menu. Now click Next → Run to start the game.

If the game starts up it means, that new OpenGL video renderer works fine. Exit the game and let's begin with advanced visual and audio settings configuration.

Part Four

Advanced Engine and OpenGL Settings

UT stores most of its user configurable settings in two files:

- UnrealTournament.ini
- User.ini

Both of them are located in UnrealTournament\System folder.

Open UnrealTournament.ini in Notepad or other similar text editing application.

IMPORTANT: Pay close attention to backslash marks (//) and the text following after that. These marks are only comments. Do not add them into your UnrealTournament.ini file!

First lets increase system cache for UT. Go to **[Engine.GameEngine]** and change CacheSizeMegs default value from 4 to 16 so it will look like this:

```
[Engine.GameEngine]
CacheSizeMegs=16
```

Next go to **[WinDrv.WindowsClient]** section and make it look like this:

```
[WinDrv.WindowsClient]
WindowedViewportX=640
WindowedViewportY=480
WindowedColorBits=32
FullscreenViewportX=1920 // Set YOUR horizontal screen resolution!
FullscreenViewportY=1200 // Set YOUR vertical screen resolution!
FullscreenColorBits=32
Brightness=0.400000 // Set brightness level (you can also set this option in-game)
MipFactor=1.000000
UseDirectDraw=True
UseJoystick=False
CaptureMouse=True
StartupFullscreen=True
CurvedSurfaces=True
LowDetailTextures=False
ScreenFlashes=True
NoLighting=False
SlowVideoBuffering=True
DeadZoneXYZ=True
DeadZoneRUV=False
InvertVertical=False
ScaleXYZ=1000.000000
ScaleRUV=2000.000000
MinDesiredFrameRate=1.000000
```

```
Decals=True
NoDynamicLights=False
UseDirectInput=True
ParticleDensity=0
NoFractalAnim=False
SkinDetail=High
TextureDetail=High
```

Now lets configure some audio settings for higher quality and fidelity. Go to **[Galaxy.GalaxyAudioSubsystem]** where you can set all Galaxy audio renderer settings.

```
[Galaxy.GalaxyAudioSubsystem]
UseDirectSound=True
UseFilter=True
UseSurround=True //Will make UT sound better even with 2.0 stereo configurations.
UseStereo=True
UseCDMusic=False
UseDigitalMusic=True
UseSpatial=False
UseReverb=True
Use3dHardware=False
LowSoundQuality=False
ReverseStereo=False
Latency=45 // Default value 40 will occasionally cause sound stutter and crackling.
OutputRate=44100Hz // Increase sample rate for improved sound quality.
EffectsChannels=32 // Change from default 16 channels for richer audio effects.
DopplerSpeed=9000.000000
MusicVolume=160
SoundVolume=224
AmbientFactor=0.700000
```

A note about alternative audio renderers

There are other community made audio renderers available like OpenAL and FMod. While in theory they should provide more accurate audio effects and settings, I personally cannot recommand them at this point. I have tested all of them and each one has some „quirks“ like too loud announcer voice or muffled weapon sounds etc. There may be possibility of some sort of hardware/software conflict, but various systems have resulted me recurring issues. None of these quirks have came up with Galaxy audio renderer so I’ve decided to concentrate on that as most „safe“ solution. You are free to try those alternative renderers and see if they provide better results for you.

Now the most important part – OpenGL graphical settings that are located under **[OpenGLDrv.OpenGLRenderDevice]**. If you cannot find [OpenGLDrv.OpenGLRenderDevice] section in your UnrealTournament.ini then add it yourself, because UT have not created it yet automatically.

More detailed descriptions for each setting are at the end of this document.

```
[OpenGLDrv.OpenGLRenderDevice]
ZRangeHack=True
NoAATiles=True
NumAASamples=4 // Level of anti-aliasing, valid values are 0, 2, 4 and 8.
UseAA=True //Enable anti-aliasing.
MaskedTextureHack=True
SmoothMaskedTextures=False
SceneNodeHack=True
FrameRateLimit=62 // Do not go over 100 or you'll get some inconstant game speeds.
SwapInterval=0 // 0 means Vsync off, 1 means Vsync on (will cause mouse imput lag).
UseFragmentProgram=True
UseMultiDrawArrays=True
TexDXT1ToDXT3=False
DynamicTexIdRecycleLevel=100
CacheStaticMaps=True
UseTexPool=True
UseTexIdPool=True
UseSSE2=True
UseSSE=True
BufferTileQuads=True
SinglePassDetail=False
SinglePassFog=True
ColorizeDetailTextures=False
DetailClipping=False
DetailMax=2
RefreshRate=75 // Set maximum refresh rate your display supports.
MaxTMUnits=0
NoFiltering=False
MaxAnisotropy=16 // Level of anisotrophic filtering.
Use16BitTextures=False
UseS3TC=True // Enable higher resolution S3TC textures.
UseAlphaPalette=True
UseTrilinear=True
UsePrecache=True
ShareLists=False
UsePalette=True
UseMultiTexture=True
UseBGRATextures=True
UseZTrick=False
MaxLogTextureSize=12
MinLogTextureSize=0
OneXBlending=False
```

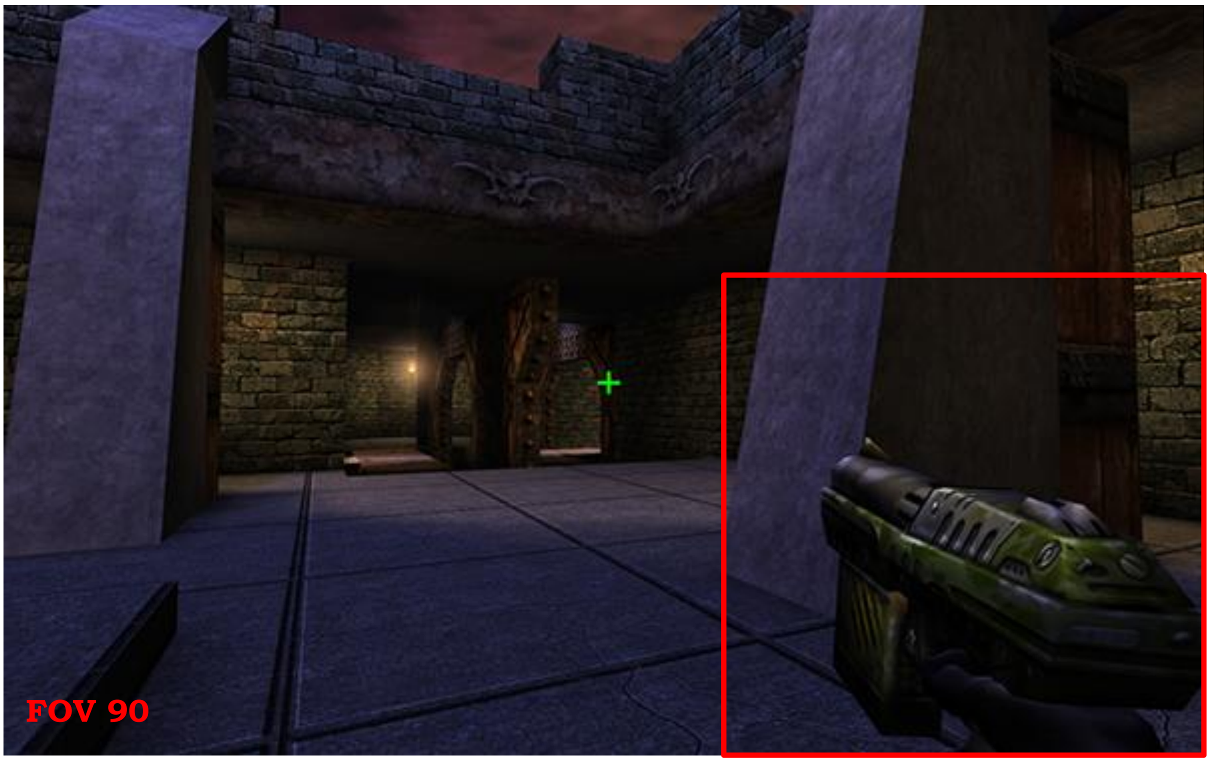
```
GammaCorrectScreenshots=False
GammaOffsetBlue=0.000000
GammaOffsetGreen=0.000000
GammaOffsetRed=0.000000
GammaOffset=0.000000
LODBias=-1.000000 // Use negative values to sharpen textures, positive to blur them.
DetailTextures=True
DescFlags=0
Description=ATI Radeon HD 4800 Series // Add your GPU description if you want to.
HighDetailActors=True
Coronas=True
ShinySurfaces=True
VolumetricLighting=True
```

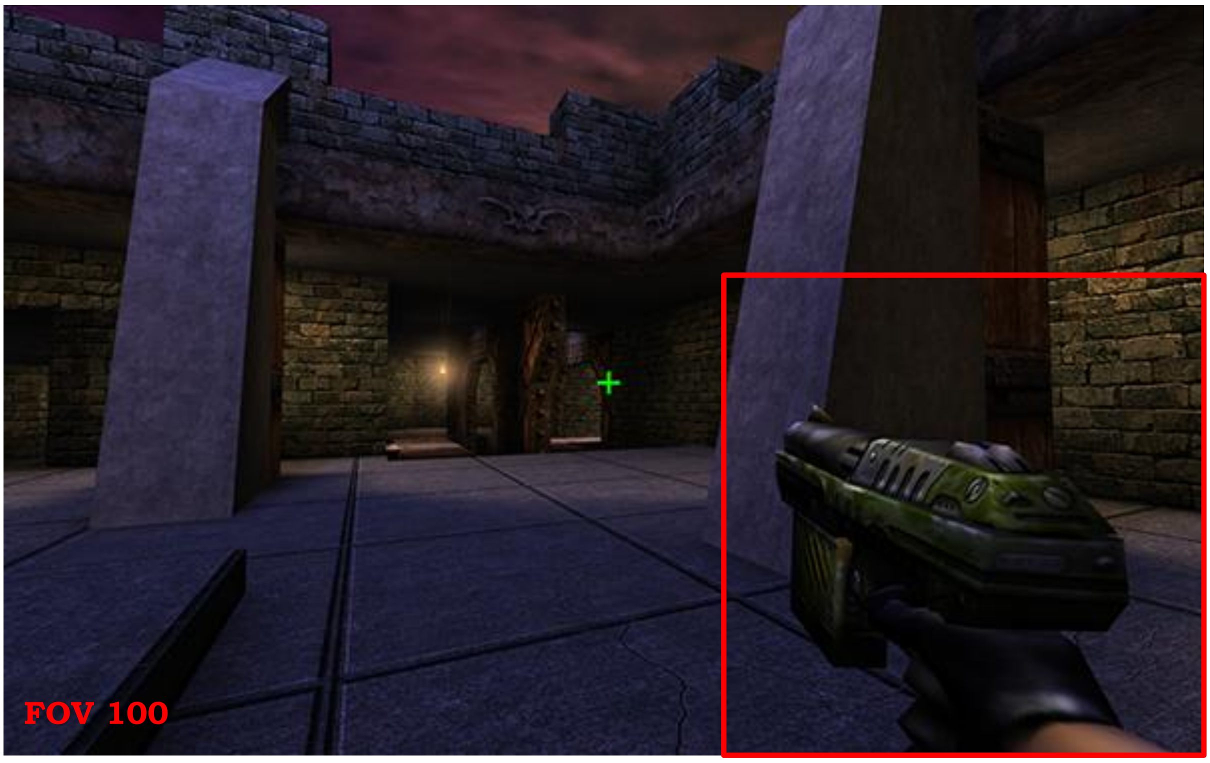
Field of View Bug

Since most of today's computer displays are widescreen, FOV (Field of View) value becomes an issue with older game titles like UT. Some players prefer higher FOV while others like the default setting, but there is another minor bug associated with UT.

Default FOV for UT is 90 degrees. With widescreen displays this can cause one minor issue that might not be right noticeable. I'm specifically talking about player weapon model you'll see when you play. If you're running default FOV 90 and using widescreen display and resolution, weapon model becomes slightly cut off from the edge of display. While this may seem as minor thing for some people, I find it to be somewhat immersion breaking. To demonstrate this issue, I have taken two comparing screenshots, first one has default FOV value 90 while second has FOV value 100. Resolution was set to 1280x800.

Go to next page to see the screenshots.


FOV 90


FOV 100

As you can see, Enforcer model or more specifically player’s hand holding the Enforcer is not visible because of too narrow FOV value (default 90) when using widescreen resolution. To correct this, open **User.ini** and find **[Engine.PlayerPawn]** section.

```
[Engine.PlayerPawn]
DesiredFOV=95.000000
DefaultFOV=95.000000
```

Setting FOV is really a personal preference so you are free to experiment with different values to find something, that really suits your needs.

Part Five

Detailed Descriptions for Each OpenGL Setting

Chris Dohnal, author of updated OpenGL renderer, has provided very nice descriptions for each option you can tweak under OpenGL renderer. All the following is from his website:

http://www.cwdohnal.com/utglr/settings.html

UseTrilinear - [True/False]
Controls the use of trilinear texture filtering.

NoFiltering - [True/False]
Can disable filtering on all textures. Useful as a debug option.

MaxAnisotropy - [Integer]
Controls the use and level of anisotropic texture filtering. Disabled if set to 0. Should make no difference if set to 1 (isotropic texture filtering). If set to greater than 1, specifies the maximum degree of anisotropy to use for texture filtering.

UseS3TC - [True/False]
Enables the use of high resolution S3TC compressed textures if they are installed.

Use16BitTextures - [True/False]
Selects lower quality and more compact formats for a number of textures, which will often speed things up. In many cases, there is only minor quality loss. In other cases, like with various skyboxes and coronas, there is often major quality loss.

UseBGRATextures - [True/False]
Allows textures to be uploaded in BGRA format rather than RGBA format if the GL_EXT_bgra extension is supported. This can improve texture upload performance. This option should always be enabled unless it causes problems.

LODBias - [Floating point]
Allows mipmap selection bias to be adjusted. Use negative values to pseudo sharpen textures. Use positive values to blur textures and potentially improve performance at the expense of blurry textures.

UseTNT - [True/False]
A workaround for buggy TNT/TNT2 drivers. Alters texture scaling and mipmap generation behavior. If you really want to know all the details, check the source code.

TexDXT1ToDXT3 - [True/False]
A workaround for poor image quality on NVIDIA GeForce1 - GeForce4 series hardware when using DXT1 format S3TC compressed textures. If enabled, converts all DXT1 textures to DXT3 textures on upload. This improves image quality on the previously mentioned NVIDIA hardware at the expense of twice as much texture memory usage for these textures. The NVIDIA DXT1 image quality problems or most noticeable on certain skybox textures. Keep this in mind when deciding whether or not to trade image quality for speed here. This option should not be enabled on any hardware that draws DXT1 textures with the same quality as DXT3 textures of course.

UseMultiTexture - [True/False]
Controls the use of multitexturing. Should always be enabled as the renderer has a few glitches when it is not. I might try to track these down some day.

UsePrecache - [True/False]
Controls texture precaching. Texture precaching may improve performance by initializing internal data structures for a number of world textures and most likely getting them loaded into video memory at level load time. It will also slow level loading down some.

MaxTMUnits - [Integer]
Used to limit the number of texture units used by the renderer. Useful as a debug option. Disabled if set to 0.

UsePalette - [True/False]
Controls the use of paletted textures. If there is hardware support for paletted textures, using them can significantly improve performance.

UseAlphaPalette - [True/False]
A workaround for very old buggy GeForce drivers. If set to False, will not upload masked textures as paletted. If there is hardware support for paletted textures, this option should be set to True unless it causes any problems.

MaskedTextureHack - [True/False]
Enabling this option can prevent rendering problems with masked textures when the same texture is applied to different polygons that do not have the masked attribute set consistently across all of them. Likely examples of masked texture problems are rendering errors with solid colored boxes around railings and trees that can often times be fixed with the flush command. There is some risk to using this option, which is why it's called a hack option. It's likely to be very safe, but not completely safe. Implementing it the completely safe way is a lot of extra work, so it uses the simple solution. If it does happen to fail, there will be some completely incorrect textures on some objects.

GammaOffset - [Floating point]
Offset for gamma correction. Can be used to adjust gamma correction even more if you hit the end of the Brightness slider in Video options. The default value of 0.0 causes no change. Use negative values for darker or positive values for brighter. If adjusting this setting for the first time, I'd recommend starting with small values such as -0.3 or 0.3.

GammaCorrectScreenshots - [True/False]
If enabled, will apply gamma correction to screen shots.

GammaOffsetRed - [Floating point]
GammaOffsetGreen - [Floating point]
GammaOffsetBlue - [Floating point]
Fine tuning parameters for gamma correction. These allow different offsets to be specified for each color channel. These offsets are never applied when gamma correcting screen shots, even if GammaCorrectScreenshots is enabled.

OneXBlending - [True/False]
If enabled, matches what the D3D renderer does for blending in multitexture mode when applying lightmaps to world geometry. I can't say for sure which way is correct. In single texture mode, the D3D renderer does appear to do blending like the OpenGL renderer in single texture mode or multitexture mode without OneXBlending enabled.

RefreshRate - [Integer]
Can be used to request a specific refresh rate when running full screen. If set to 0, a default refresh rate is used. If this value is set to an invalid or unsupported refresh rate based on video card or monitor capabilities, the renderer will fail to initialize.

SwapInterval - [Integer]
Controls V Sync. If set to the default value of -1, the default buffer swapping method is used. Set to 0 to disable V Sync. Set to 1 to enable V Sync. Set to higher values for one frame every N screen refreshes. Not all video drivers support values higher than 1.

FrameRateLimit - [Integer]
CPU controlled frame rate limiter in frames per second. Set to 0 to disable.

UseAA - [True/False]
Enables multisample antialiasing. For the OpenGL renderer, requires the GL_ARB_multisample extension.

NumAASamples - [Integer]
Specifies the number of samples to use per fragment for antialiasing. 2 and 4 are common values that should work on many video cards.

NoAATiles - [True/False]
Enable this option to disable antialiasing when drawing tiles as seen from the lower half renderer perspective. This should eliminate HUD corruption that can occur when antialiasing is enabled. Some video hardware / drivers do not support the functionality required to enable this option. Note that corruption with antialiasing enabled can still occur on the logo background if using Entry.unr on startup (it's not made of tiles from the renderer perspective).

UseZTrick - [True/False]
Can avoid some z-buffer clears at the expense of cutting z-buffer precision in half. This may improve performance on some video cards. On video cards with z-buffer optimization hardware, enabling this setting may significantly reduce performance as it interferes with some hardware z-buffer optimization implementations.

ZRangeHack - [True/False]
An experimental option that can make the z-buffer work better for far away objects. Might cause unexpected problems, but doesn't seem to break anything major so far. Will fix problems with decals flickering in the distance with 24-bit z-buffers, which is the most you can get on many video cards. Will also fix the issue with the Redeemer covering up part of the HUD. Partially breaks weapon rendering on the first person view one if using wireframe debug mode (will clip near parts of it). Doesn't help enough to make 16-bit z-buffers work correctly.

MinLogTextureSize - [Integer]
Set to 0.

MaxLogTextureSize - [Integer]
Set to 8, or 0.

UseCVA - [True/False]
Enables the use of the compiled vertex array (CVA) extension. It may be useful on video cards without HW T&L. It is likely to slow things down a little bit on video cards with HW T&L.

UseMultidrawArrays - [True/False]
Enables the use of the GL_EXT_multi_draw_arrays extension.

BufferTileQuads - [True/False]
Enables buffering in the DrawTile path. May improve text rendering performance.

UseSSE - [True/False]
Controls the use of SSE instructions. Set to True to auto detect CPU and OS support for SSE instructions and use them if supported. Set to False to disable the use of SSE instructions.

UseVertexProgram - [True/False]
Enables vertex program mode. Consider this an experimental option. It can improve performance in some cases. It can also slow things down a lot if certain other settings are not configured correctly. It is likely to slow things down a lot if detail textures are enabled, but single pass detail texture mode is not enabled. It may not work correctly or may cause crashes with some video drivers.

UseFragmentProgram - [True/False]
Enables fragment program mode. Requires the UseVertexProgram option to also be enabled. May improve performance on newer video hardware. It's generally best to enable or disable UseVertexProgram and UseFragmentProgram together. In a later version of the D3D9 renderer, the UseVertexProgram option is not present and UseFragmentProgram will enable use of both vertex shaders and pixel shaders together if shader model 3 support is available.

UseTexIdPool - [True/False]
Should be set to True.

UseTexPool - [True/False]
Should be set to True.

DynamicTexIdRecycleLevel - [Integer]
Should be set to the default value of 100.

DetailTextures - [True/False]
Enables detail textures.

DetailClipping - [True/False]
Enables the use of a somewhat experimental detail texture mode. It costs more CPU time, but may improve performance in fill rate limited situations.

DetailMax - [Integer]
Set to 2 to enable a second detail texture layer. Set to 0 or 1 for standard one layer detail texturing if detail textures are enabled. The second detail texture layer will not show up unless SinglePassDetail is disabled.

SinglePassDetail - [True/False]
Enables single pass detail texture mode. This should generally be the highest performance detail texture mode. It requires 4 texture units. It also requires the UseDetailAlpha option to be enabled.

SinglePassFog - [True/False]
Enables single pass fog mode. This should generally be the highest performance fog mode. It requires 3 texture units. For the OpenGL renderer, it also requires support for either the GL_ATI_texture_env_combine3 extension or the GL_NV_texture_env_combine4 extension.

ColorizeDetailTextures - [True/False]
Debug option for detail textures. If enabled, adds a green tint to detail textures.

End of Document.