Texture Format Support and Settings

Reference for supported texture formats and file types and their configuration.

One of the biggest contributors to any digital projects memory footprint comes from the size and amount of Textures used. Luckily Unreal Engine has a very robust system for non-destructively reducing Texture size across all your projects Textures. In the following page we will take a look at these systems and how you can use them to reduce your projects Texture memory requirements.

Texture Resolution

Unreal Engine supports texture resolutions from 1×1 up to 8192×8192 with some slight modifications to .INI files. Current DirectX video adapters and game consoles support various texture resolutions from 1×1 to 2048×2048 and up to 8192×8192. The highest texture resolution supported by a specific hardware device varies by manufacturer, model and available texture memory. There are a number of features and settings in Unreal Engine 4 for managing the texture resolutions that are rendered for various areas such as world geometry or the user interface.

Engine Texture Resolution Limit

Unreal Engine 4 defaults to limiting the maximum number of texture mips to 14, which effectively limits the largest rendered texture to 8192 (1×1 to 8192×8192 is 14 mips).
This has the side-effect that imported 8192 textures will only render up to mip1 of 4096. The constant MAX_TEXTURE_MIP_COUNT which defaults to 13 in the engine source files can be modified to a value of 14 to support 8192 texture rendering. This constant is defined in the following source files (as of QAMar09, be sure to verify on other QA versions).

```
Src\D3D10Drv\Src\D3D10Device.cpp
Src\Engine\Inc\RHI.h
Src\Engine\Inc\UnTex.h
Src\Engine\Src\RHI.cpp
Src\Engine\Src\TextureCube.cpp
```

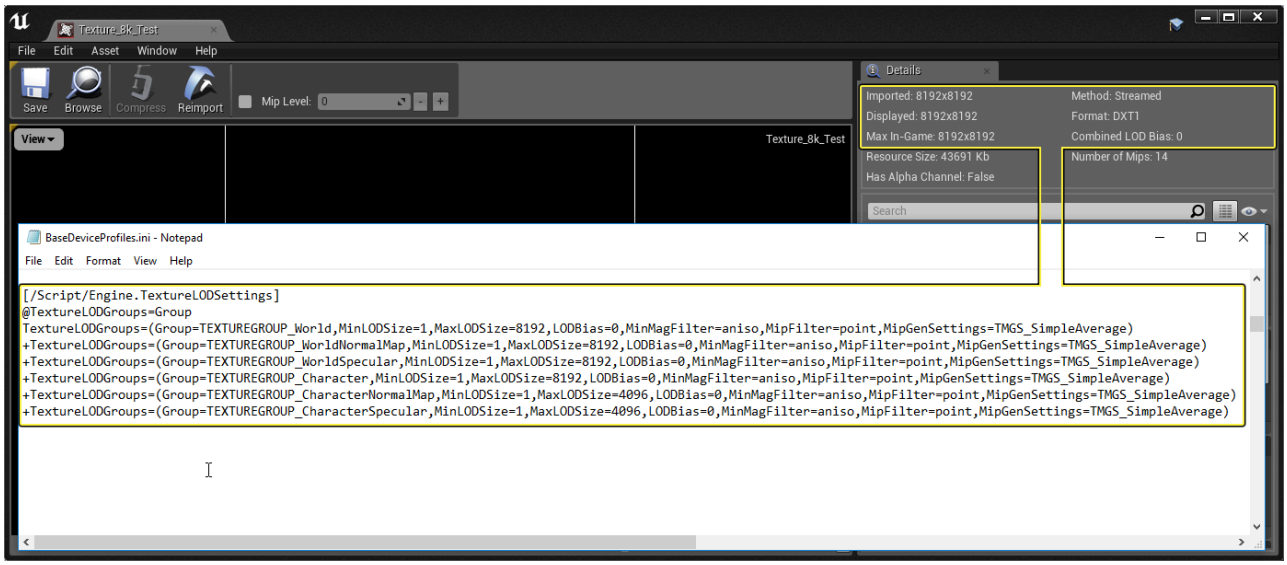
Copy full snippet

With the release of UE 4.8 you can now modify your projects to use Textures up to 8192 in size without having to modify the C++ code by adding the following text to your projects **BaseDeviceProfiles.ini** file and setting the **MaxLODsize** to **8192**

```
[/Script/Engine.TextureLODSettings]
TextureLODGroup_World=
(MinLODSize=1,MaxLODSize=8192,LODBias=0,MinMagFilter=aniso,MipFilter=point)
```

Copy full snippet

Once you have added the section you want to increase the size for, save the file and restart the Editor. When the Editor restarts any Textures that were imported at a size of 8192 should now show 8192 as the size of LOD 1, instead of being clamped to a maximum of 4096. In the following example image we have modified the BaseDeviceProfiles.ini file in a project to allow for the use of Textures that are up to 8192 in size. When the texture, Texture_8k_Test is loaded we can see that both the imported and displayed texture size are at 8192.



Click image for full size.

Compressed Texture Memory Requirements

DXT uses lossy compression based on packing pixels into 4×4 blocks with paletted colors and interpolated colors. This results in an 8:1 DXT1 and 4:1 DXT5 constant compression file size. Since video memory and texture pool resources are fixed for a specific platform and hardware, a balance must be struck between texture resolution and resource usage. The following table lists the texture memory requirements for DXT1 and DXT5 textures at various common resolutions with full mips (1×1 up to full native mip0). Note that the memory requirements are near-constant multiples of the texture resolution ratio, and that DXT5 textures require near-twice the memory of their DXT1 counterpart.

Since the resolution to compression ratio is a constant, to calculate the memory requirements for a texture resolution not listed here, simply multiply the resolution ratios. For example, a 1024×512 texture would be one-half the memory requirements of a 1024×1024 texture.

The table data was compiled from textures created by ATI's Compressonator using Box-Filter mip generation and DirectX Texture Compression.

Resolution	Total Mips from 1x1	DXT1	DXT5
16×16	5 mips	312 bytes	496 bytes
32×32	6 mips	824 bytes	1.48kb (1,520 bytes)

Resolution	Total Mips from 1x1	DXT1	DXT5
64×64	7 mips	2.80kb (2,872 bytes)	5.48kb (5,616 bytes)
128×128	8 mips	10.8kb (11,064 bytes)	21.4kb (22,000 bytes)
256×256	9 mips	42.8kb (43,832 bytes)	85.4kb (87,536 bytes)
512×512	10 mips	170kb (174,904 bytes)	341kb (349,680 bytes)
1024×1024	11 mips	682kb (699,192 bytes)	1.33MB (1,398,256 bytes)
2048×2048	12 mips	2.66MB (2,796,344 bytes)	5.33MB (5,592,560 bytes)
4096×4096	13 mips	10.6MB (11,184,952 bytes)	21.3MB (22,369,776 bytes)
8192×8192	14 mips	42.6MB (44,739,384 bytes)	85.3MB (89,478,640 bytes)

Engine Config TextureGroup Properties

The minimum and maximum LOD (mip) supported for specific game TextureGroups is defined in a number of engine configuration files.
The source set of configuration settings files is located in the `[Unreal Engine 4 Install Location]\Engine\Config\BaseDeviceProfiles.ini` file under the [/Scripts/Engine.TextureLODSettings] section.

For developing games, the `[your_game]\Config\DefaultDeviceProfiles.ini` file also contains a mirror set of the base properties in the `Engine\Config\` folder and should be the copy that is normally modified for your game's specific settings.

Note that there are independent sets of TextureGroup entries for the Unreal Editor and in-game. These two sets are respectively located in the [SystemSettingsEditor] and [SystemSettings] sections in the config files.

The TextureLODGroup settings entries in the `DefaultDeviceProfiles.ini` file will look similar to this. Note that older QA versions may not include the MinMagFilter and MipFilter properties for each setting.

```
[/Script/Engine.TextureLODSettings]
; NOTE THAT ANY ITEMS IN THIS SECTION WILL AFFECT ALL PLATFORMS!!!
@TextureLODGroups=Group
TextureLODGroups=(Group=TEXTUREGROUP_World,MinLODSize=1,MaxLODSize=4096,LODBi
+TextureLODGroups=
(Group=TEXTUREGROUP_WorldNormalMap,MinLODSize=1,MaxLODSize=4096,LODBias=0,Mir
+TextureLODGroups=
(Group=TEXTUREGROUP_WorldSpecular,MinLODSize=1,MaxLODSize=4096,LODBias=0,MinM
+TextureLODGroups=(Group=TEXTUREGROUP_Character,MinLODSize=1,MaxLODSize=4096,
+TextureLODGroups=
(Group=TEXTUREGROUP_CharacterNormalMap,MinLODSize=1,MaxLODSize=4096,LODBias=0
+TextureLODGroups=
(Group=TEXTUREGROUP_CharacterSpecular,MinLODSize=1,MaxLODSize=4096,LODBias=0,
+TextureLODGroups=(Group=TEXTUREGROUP_Weapon,MinLODSize=1,MaxLODSize=4096,LOD
+TextureLODGroups=
(Group=TEXTUREGROUP_WeaponNormalMap,MinLODSize=1,MaxLODSize=4096,LODBias=0,Mi
+TextureLODGroups=
(Group=TEXTUREGROUP_WeaponSpecular,MinLODSize=1,MaxLODSize=4096,LODBias=0,Mir
+TextureLODGroups=(Group=TEXTUREGROUP_Vehicle,MinLODSize=1,MaxLODSize=4096,LO
+TextureLODGroups=
(Group=TEXTUREGROUP_VehicleNormalMap,MinLODSize=1,MaxLODSize=4096,LODBias=0,M
+TextureLODGroups=
(Group=TEXTUREGROUP_VehicleSpecular,MinLODSize=1,MaxLODSize=4096,LODBias=0,Mi
+TextureLODGroups=(Group=TEXTUREGROUP_Cinematic,MinLODSize=1,MaxLODSize=4096,
+TextureLODGroups=(Group=TEXTUREGROUP_Effects,MinLODSize=1,MaxLODSize=4096,LO
+TextureLODGroups=
(Group=TEXTUREGROUP_EffectsNotFiltered,MinLODSize=1,MaxLODSize=4096,LODBias=0
+TextureLODGroups=(Group=TEXTUREGROUP_Skybox,MinLODSize=1,MaxLODSize=4096,LOD
+TextureLODGroups=(Group=TEXTUREGROUP_UI,MinLODSize=1,MaxLODSize=4096,LODBias
+TextureLODGroups=(Group=TEXTUREGROUP_Lightmap,MinLODSize=1,MaxLODSize=4096,L
+TextureLODGroups=
(Group=TEXTUREGROUP_Shadowmap,MinLODSize=1,MaxLODSize=4096,LODBias=0,MinMagFi
+TextureLODGroups=
(Group=TEXTUREGROUP_RenderTarget,MinLODSize=1,MaxLODSize=4096,LODBias=0,MinMa
+TextureLODGroups=
(Group=TEXTUREGROUP_MobileFlattened,MinLODSize=1,MaxLODSize=4096,LODBias=0,Mi
```

```
25 +TextureLODGroups=
   (Group=TEXTUREGROUP_Terrain_Heightmap,MinLODSize=1,MaxLODSize=4096,LODBias=0,
26 +TextureLODGroups=
   (Group=TEXTUREGROUP_Terrain_Weightmap,MinLODSize=1,MaxLODSize=4096,LODBias=0,
27 +TextureLODGroups=(Group=TEXTUREGROUP_Bokeh,MinLODSize=1,MaxLODSize=256,LODBi
28 +TextureLODGroups=(Group=TEXTUREGROUP_Pixels2D,MinLODSize=1,MaxLODSize=4096,L
```

Copy full snippet

PC AppCompat Buckets

AppCompat is used to override various SystemSettings based on objective and empirical evidence gathered at startup. When app compatibility is enabled (PC only), the system measures machine capability, and then overwrites the Engine.ini values with preset values from one of 5 "buckets". See BaseCompat.ini in the `Engine\Config\` folder for an example of this usage.

AppCompat is intended to only be checked ONCE when the game is first run (not the editor). It detects this by checking for the existence of an [AppCompat] section in [game]Engine.ini, which contains the previously computed scores for the machine. If AppCompat has already been applied once, it is not changed again to allow custom changes to be made by users without being overwritten every time.

AppCompat is specifically disabled for the editor so machine specs do not affect how assets are viewed on various machines during development. This is the reason for the split between SystemSettings and SystemSettingsEditor.

You can effectively disable AppCompat by supplying an empty DefaultCompat.ini for your game, which causes it to initialize all buckets from [SystemSettings] in Engine.ini. In this case, the system operates exactly as it did before AppCompat was introduced.

TEXTUREGROUP Properties

Each TextureGroup entry defines the texture properties for a specific texture set as used in the game rendering. Grouping textures into common sets allows for better control over the texture memory pool use by various game texture resources.

Property	Description
MinLODSize	Minimum mip size that will be rendered, specified in pixels, range of 1 to 8192 as power-of-two's, must be less than MaxLODSize.
MaxLODSize	Maximum mip size that will be rendered, specified in pixels, range of 1 to 8192 as power-of-two's, must be greater than MinLODSize.
LODBias	A negative or positive value that determines the number of mip levels to offset prior to uploading for render, clamped within MinLODSize and MaxLODSize.
MinMagFilter	Specifies the texture filter type when textures are minified or magnified by the GPU. See the chart below.
MipFilter	Specifies whether the GPU should blend two mips together when viewing the texture from a distance or at a grazing angle. See the chart below.
NumStreamedMips	The number of mips that are allowed to be streamed in or out. If a texture has 10 mips and NumStreamedMips is 2, only the 2 highest mips will be allowed to stream in or out. The texture will therefore have 8-10 mips in memory at any given time. Setting NumStreamedMips to 0 means that no mips will be streamed and the textures using this LOD group will always be fully loaded. Setting NumStreamedMips to -1 means that all mips are allowed to be streamed in or out (there are still other restrictions that apply though). NumStreamedMips is an optional setting that defaults to -1.

Filtering

MinMagFilter	MipFilter	filter type
point		Point

MinMagFilter	MipFilter	filter type
linear	point	Bilinear
linear		Trilinear
aniso	point	Anisotropic Point
aniso		Anisotropic Linear

TextureGroup, LODGroup and LODBias

The TextureGroup and LODBias settings specified in the config ini files, along with the LODGroup and LODBias settings specified in the Texture Properties determine the final set of texture mips used for an individual texture.

An example TextureGroup entry in the [your_game]Engine.ini may look like this:

```
Group=TEXTUREGROUP_World,MinLODSize=1,MaxLODSize=4096,LODBias=0,MinMagFilter
```

Copy full snippet

Any textures assigned to the TEXTUREGROUP_World LODGroup will use these settings to determine the mip range used for rendering.
The additional LODBias setting in the Texture Properties is additive with the LODBias specified in the config ini file TextureGroup.

The LODBias *biases* or offsets which mip is chosen for rendering. The LODBias is calculated before the LODGroup Min/Max range. The LODBias in the Texture Properties is added to the LODBias in the TextureGroup to determine the final LODBias value used.
A LODBias of 0 is the main (native) texture resolution. A LODBias of 1 is the first mip down for the texture, a LODBias of 2 is the second mip down, etc. For example, a 1024×1024 texture that has a LODBias of 1 results in the 512×512 mip being chosen for rendering.

The LODBias specified in the Texture Properties for each individual texture can be positive or negative, so that it can offset the TextureGroup's default LODBias to either higher or lower mip values.
For example:

- A TextureGroup LODBias of 0 and a Texture Properties LODBias of 0 would result in a final LODBias of 0.
- A TextureGroup LODBias of 0 and a Texture Properties LODBias of 1 would result in a final LODBias of 1.
- A TextureGroup LODBias of 1 and a Texture Properties LODBias of 1 would result in a final LODBias of 2.
- A TextureGroup LODBias of 1 and a Texture Properties LODBias of -1 would result in a final LODBias of 0.

After the final LODBias is calculated, then the texture mip is checked to see that it fits into the TextureGroup's Min/Max LODSize range, and it is adjusted if necessary. This allows a simple config ini file change to effectively clamp a specific TextureGroup to within a set min/max LOD range.

For example, a 1024×1024 texture with LODBias of 1 uses the 512×512 mip, if it is in the TEXTUREGROUP_World LODGroup as shown above, it is then checked to see if it fits within the TextureGroup's Minimum and Maximum LODSize range, which in this case is 256 and 1024. Since each game title will have its own unique TextureGroup settings, artists and level designers should be aware of the MinLODSize and MaxLODSize for each group. It would increase distributable package size with no rendering quality benefit if a game shipped with 2048 textures assigned to a TextureGroup with a MaxLODSize of 1024.

Texture Properties

For an explanation of the meaning of the various texture properties, see the Texture Properties page.