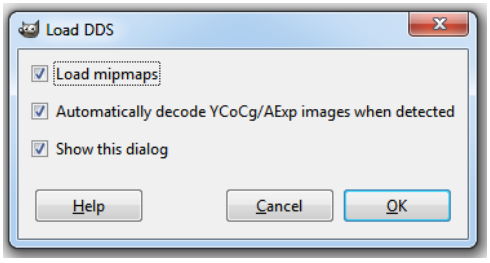
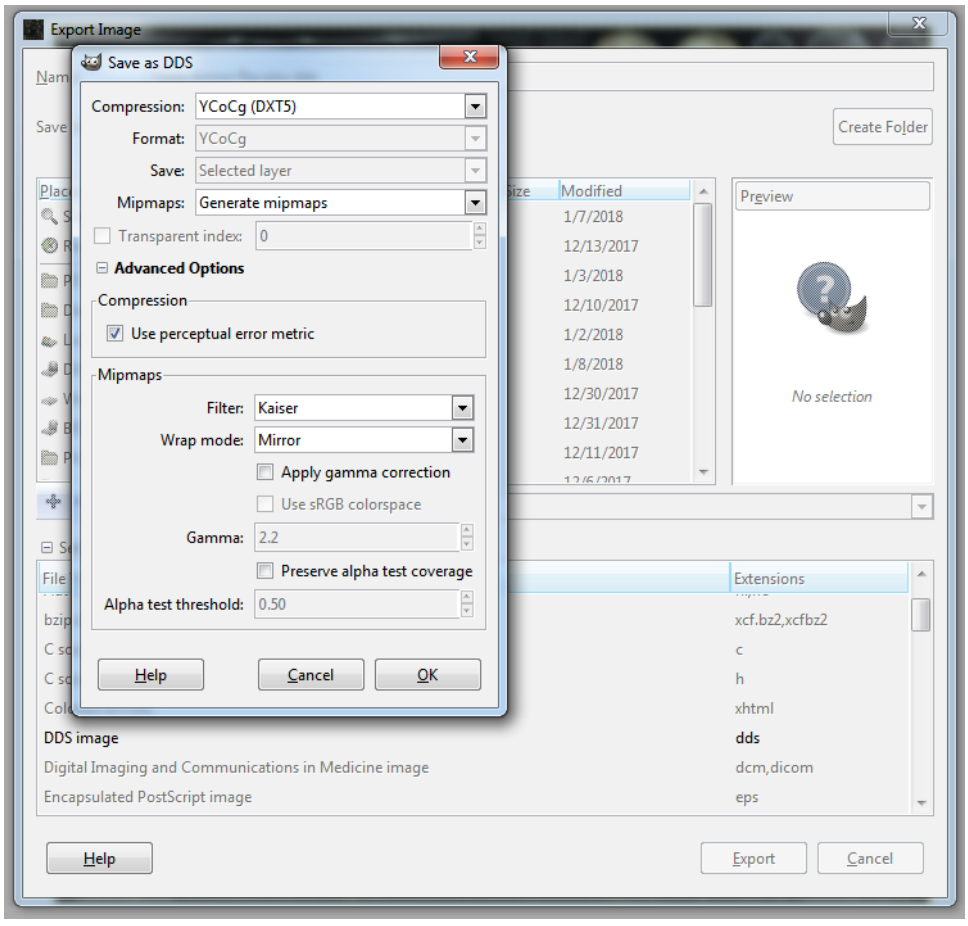
DDS Plugin Guide for Gimp 2.x.x

Version 0.2





By Viktor Geist, aka Visitant (visitant@yahoo.com)

Purpose

Of particular importance to digital artists, game authors, and modders of games like Skyrim and Fallout is the ability to create textures for models and games, and save them as DDS files. And, when needed, to open those DDS files and edit them. This ability to create and edit DDS files has been a prominent factor when comparing Photoshop to The Gimp. Most have gravitated toward Photoshop because they are completely unaware that The Gimp does in fact have a plugin that allows the creation and editing of DDS files, and more. Well, it does! And you can do a lot with it. For example, you can also create Cube Maps and Volume Maps with The Gimp's DDS plugin.

Another interesting fact is, many (if not most) Photoshop plugins have to be purchased. Not so with plugins for The Gimp; they are free. Best of all, you can in fact install and use most Photoshop plugins in The Gimp via the "PSPI" plugin (limited to the 32-bit versions). For example, with this plugin I can continue to use my entire Flaming Pear suite without issue. Not counting the fact that I sometimes use Flaming Pear when creating DDS textures, I am greatly relieved that I can still use those plugins because I paid for the software.

The biggest problem with the DDS plugin for The Gimp is there is very little information that describes how to use it, and even less information on the various options available to the user in its menus.

This document is meant to fill the gap in this lack of information. I will add to it as I can, over time.

~V~

Test Information

Date of test: 2/12/2018

OS: Windows 7 Ultimate 64-bit.

Gimp version: 2.8.22, 64-bit

Download site: https://code.google.com/archive/p/gimp-dds/downloads

Plugin install path: C:\Program Files\GIMP 2\lib\gimp\2.0\plug-ins

NOTE: Plugin also works in: .gimp-2.8\plug-ins

Acknowledgments

This document was created using personal knowledge gleaned from experimentation with DDS files created and modified using The Gimp, and by examination of information concerning DDS files and their various compression types that is freely available on the internet. In every case wherein the content of this document was derived from the internet I have made sure to link to the source pages in question. In this way the authors get the credit they deserve. I would like to thank those authors for the information they chose to share with the world via the internet.

A list of links to these internet sources are found on the last page of this document.

For best viewing, print this document out.
Please feel free to distribute this document as you see fit for the good of The Gimp community.

Main Menu

Compression:

1. BC1 / DXT1
2. BC2 / DXT3
3. BC3 / DXT5
4. BC3nm / DXT5nm
5. BC4 / ATI1 (3Dc+)
6. BC5 / ATI2 (3Dc)
7. RXGB (DXT5)
8. Alpha Exponent (DXT5)
9. YCoCg (DXT5)
10. YCoCg scaled (DXT5)

DXT is a texture compression format, meaning it will stay compressed in video memory, allowing the artist to use more or larger textures. There are five DXT formats: DXT1, DXT2, DXT3, DXT4, and DXT5. In DirectX 10 it is known as Block Compression and comes in five basic flavors: BC1, BC2, BC3, BC4, and BC5.

The DXT compression format can be stored inside container files like DDS, VTF, and so on, but DDS is the most common file-type used for this purpose.

Source images must be power of two in size (i.e. 4x4, 32x8, 256x512, 1024x1024, etc.), and the smallest size is 4x4. If you attempt to utilize a smaller size (like 1x1 or 1x256) the file will be padded out to 4 pixels wide and/or tall.

BC1 / DXT1

Generally the best DXT format for textures with just black and white in the Alpha channel, no grays.

- In DirectX 10 it is called BC1.
- Compressed, with an optional 1-bit Alpha (just black and white, no grays).
- Smallest file size, half the size of DXT3 and DXT5. DXT1a is same file size as DXT1c.
- Doesn't work very well on images that have stark color changes, like pixel art.
- DXT1a and DXT1c are the same format, just an internal switch for enabling alpha or not.

NOTE: the alpha is actually stored in the RGB data, so the black parts of the Alpha will replace the RGB color completely. Enabling or disabling the alpha channel seems to be via selecting the *Format:* option first. RGB8 for DXT1c, and RGB8A8 for DXT1a.

BC2 / DXT2

- In DirectX 10 it is called BC2.
- Same format as DXT3, except it assumes the alpha is not pre-multiplied.
- Rarely used because it requires extra processing in the shader.

NOTE: Not obviously available in Gimp DDS plugin.

BC2 / DXT3

Generally the best DXT format for textures with a sharp Alpha channel.

1. In DirectX 10 it is called BC2.
2. Compressed, with "explicit" Alpha.
    - Color is compressed the same as DXT1.
    - In the alpha channel, each pixel is only one of 16 levels of gray.
    - This works well if the alpha values are mostly black and mostly white, with thin anti-aliasing between them.
3. Twice the file size of DXT1, same file size as DXT5.

BC3 / DXT4

1. In DirectX 10 it is called BC3.
2. Same format as DXT5, except it assumes the alpha is not pre-multiplied.
3. Rarely used because it requires extra processing in the shader.

NOTE: Not obviously available in Gimp DDS plugin.

BC3 / DXT5

Generally the best DXT format for textures with a smooth Alpha channel.

1. In DirectX 10 it is called BC3.
2. Compressed, with interpolated Alpha.
    - Color is compressed the same as DXT1. In the alpha channel, each 4x4 block is compressed separately from the others.
    - Two of the pixels are stored in 256 levels of gray, while the other 14 pixels in the block are interpolated between those two, using 8 levels of gray.
    - This works well when each block has a fairly smooth gradation of values, rather than sharp transitions.
3. Twice the file size of DXT1, same file size as DXT3.

BC3nm / DXT5nm

This is a variation of BC3 that is used to represent normal maps by encoding the X and Y components as follows: R=1, G=Y, B=0, A=X, this swizzle is used to facilitate decompression. Normal mapping has significant performance benefits over bump mapping, in that far fewer operations are required to calculate the surface lighting.

BC4 / ATI1 (3Dc+)

This is a block compression format that only contains a single alpha block.

BC5 / ATI2 (3Dc)

This is a block compression format that contains two alpha blocks. It's typically used to compress normal maps.

RXGB (DXT5)
A variant of BC3 / DXT 5, it modifies how the texture is read by the renderer. Requires the shader to be rewritten especially for this kind of texture. Rarely used.

Alpha Exponent (DXT5)
A DXT5 texture with a highly compressed alpha channel. Allows a DXT5 texture to render high quality images.

YCoCg (DXT5)
High-quality compression of color images can be achieved by using the DXT5 format after converting the RGB_ data to CoCg_Y. In other words, the luma (Y) is stored in the alpha channel and the chroma (CoCg) is stored in the first two of the 5:6:5 color channels. For color images, this technique results in a 4:1 compression ratio with very good quality – generally better than 4:2:0 JPEG at the highest quality setting.

YCoCg scaled (DXT5)
Same as above, except has more instructions. Improves pixel quality.

NOTE 1: Unless the texture requires another format, I would use DXT5nm as the default for creating and saving normal maps.

NOTE 2: When creating and saving textures using the DXT5 format, I think it best to use AExp or, preferably, one of the YCoCg-types over the typical DXT5 format when possible With modern hardware, the performance hit would be negligible.

Format:

1. Default
2. RGB8 – 8 bits per channel, no alpha channel, 24-bit depth
3. RGB8A8 – 8 bits per channel, with alpha channel, 32-bit depth
4. BGR8 – channels re-ordered, 8 bits per channel, no alpha channel, 24-bit depth
5. ABGR8 – channels re-ordered, 8 bits per channel, with alpha channel, 32-bit depth
6. R5G6B5 – RGB with 5, 6, and 5 bits per channel (respectively), no alpha channel, 16-bit depth
7. RGBA4 – 4 bits per channel, with alpha channel, 16-bit depth
8. RGB5A1 – RGB channels with 5 bits each, 1 bit alpha channel, 16-bit depth
9. RGB10A2 – RGB channels with 10 bits each, 2 bit alpha channel, 32-bit depth
10. R3G3B2 – Red and Green channels 3 bits, Blue channel 2 bits, no alpha channel, 8-bit depth
11. A8 – export as 8-bit alpha channel
12. L8 – export as 8-bit greyscale alpha channel, most often used to enhance luminance
13. L8A8 – export as 16-bit greyscale alpha channel, most often used to enhance luminance
14. Aexp – see Alpha Exponent explanation in previous section
15. YCoCg – see YCoCg explanations in previous section.

NOTE: Color depth, also known as bit depth, is either the number of bits used to indicate the color of a single pixel in a bitmapped image or video frame buffer, or the number of bits used for each color component of a single pixel.When referring to a pixel, the concept can be defined as bits per pixel (bpp), which specifies the number of bits used.

Save:

1. Selected Layer

This box is grayed out unless you are making a Cube Map or Volume Map.

Cube Maps: Creating a Cube Map requires six layers that are all the same size and color depth, with or without Alphas. Each layer may be uniquely named, but must have one of the correct identifiers in the name (Front, Back, Left, Right, Top, Bottom). For example: woodland_front, woodland_back, etc. When exporting as DDS the *Save:* box becomes active, and if the naming is correct it will automatically understand that you are using the layers to create a Cube Map, and the option (*As cube map*) will be available.

Volume Maps: The method for creating a Volume Map is the same as creating a Cube Map, with one exception. You are not limited to the number of layers you want to use. You do not need to identify any of the layers (Front, Back, etc…). On the other hand, all layers must be the same size and color depth. Export as DDS and select *As volume map* for the *Save:* box.

Mipmaps:

1. No mipmaps
2. Generate mipmaps
3. Use existing mipmaps

Self-explanatory.

Transparent index:

Note: Tick-box is grayed out as is variable box. Variable box set to 0.

I believe that when activated, it will adjust the opaqueness of a layer or alpha channel.

Advanced Options Menu

Compression

NOTE: *Use perceptual error metric* becomes available when some type of compression method from main menu (Compression:) is selected.

The *Use perceptual error metric* is supposed to enhance visual quality. I would suggest experimenting with the texture in a rendering program to see how it affects the model according to the texture type (Compression:) in question.

Mipmaps

NOTE: *Warp mode:*, *Apply gamma correction*, and *Preserve alpha test coverage* become available in the Filter: menu when *Generate mipmaps* from main menu (Mipmaps:) is selected.

Filter:

1. Default – no filtering is applied to image. Possibly/probably causes visual anomalies such as excessively sharp edges and stair-stepping artifacts.
2. Nearest – bilinear filter, sharp switching between mipmaps?
3. Box – is a polyphase box filter. Outside of the Lanczos and Kaiser filters, it's a good choice for most cases. It's also much faster than the other filters.
4. Triangle – uses a triangle filter. The kernel has a larger width and thus produces blurrier results than the box filter.
5. Quadratic – gaussian-type filter?
6. B-Spline – trilinear filter?
7. Mitchell – sinc-type filter. Used primarily to reduce loss of detail while upsampling.
8. Lanczos – similar to the Kaiser filter, the results are usually indistinguishable between the two. Lanczos may handle ringing and artifacts better. Can be tuned with Warp filter selections.
9. Kaiser - Kaiser-windowed sinc filter. It's generally considered the best choice for downsampling filters, but in order to obtain best results it is advisable to experiment with Warp modes as is done with the Lanczos filter. Otherwise the resulting images could suffer from ringing artifacts or the result may not be as sharp as desired.

Warp mode:

1. Default
2. Mirror
3. Repeat
4. Clamp

When evaluating the color of texels that are near the border, most the filters usually sample outside of the texture. The *Default* setting (1) generally looks good. From what I have observed, I think nothing is done to the image. However, better results can be achieved by explicitly specifying the desired wrapping mode.

Mirror (2) is probably best since it is used for many other image processing operations, like normal map generation. *Repeat* and *Clamp* modes (3 and 4, respectively) are self-explanatory, and I would not use either unless the texture dictates it would be best to do so.

Apply gamma correction
NOTE: ticking this box activates *Use sRGB colorspace* and *Gamma:* parameter. The default value for Gamma parameter is 2.2.

Gamma, simply explained, adjusts the reflectiveness of an object while also lightening (adding whiteness) the entire scene in which the object is found. Applying this would require fine-tuning of the model appearance within a rendering program capable of varying the lighting conditions within a given scene.

Preserve alpha test coverage
NOTE: ticking this box will activate Alpha test threshold parameter. Default parameter is 0.50.

I have yet to experiment with this, so I can offer no opinion.

Sources

https://en.wikipedia.org/wiki/S3_Texture_Compression
http://wiki.polycount.com/wiki/DXT
http://www.nvidia.com/object/real-time-normal-map-dxt-compression.html
https://github.com/castano/nvidia-texture-tools/wiki/ApiDocumentation
http://number-none.com/product/Mipmapping,%20Part%201/index.html
http://number-none.com/product/Mipmapping,%20Part%202/index.html
http://www.cs.ucf.edu/~sumant/publications/sig99.pdf
https://en.wikipedia.org/wiki/Gamma_correction
http://www.nvidia.com/object/real-time-ycocg-dxt-compression.html
https://code.google.com/archive/p/gimp-dds/wikis/DxtCompressionQuality.wiki
https://en.wikipedia.org/wiki/Color_depth