Using texturetool to Compress Textures

The iOS SDK includes a tool to compress your textures into the PVRTC or ASTC formats, aptly named `texturetool`. If you have Xcode installed with the iOS 7.0 SDK or later, then `texturetool` is located at: `/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/usr/bin/texturetool`.

`texturetool` provides various compression options, with tradeoffs between image quality and size. You need to experiment with each texture to determine which setting provides the best compromise.

> **Note:** The encoders, formats, and options available with `texturetool` are subject to change. This document describes those options available as of iOS 7.

texturetool Parameters

The parameters that may be passed to `texturetool` are described in the rest of this section.

```
user$ texturetool -h


Usage: texturetool [-hl]
       texturetool -c <reference_image> <input_image>
       texturetool [-ms] [-e <encoder>] [-p <preview_file>] -o <output> [-f <format>]
<input_image>


first form:

    -h       Display this help menu.

    -l       List available encoders, individual encoder options, and file formats.


second form:

    -c       Compare <input_image> to <reference_image> and report differences.


third form:

    -m               Generate a complete mipmap chain from the input image.

    -s               Report numerical differences between <input_image> and the
encoded image.

    -e <encoder>     Encode texture levels with <encoder>.

    -p <preview_file> Output a PNG preview of the encoded output to <preview_file>.
Requires -e option

    -o <output>      Write processed image to <output>.

    -f <format>      Set file <format> for <output> image.
```

> **Note:** The `-p` option indicates that it requires the `-e` option. It also requires the `-o` option.

Listing C-1 Encoding options

```
user$ texturetool -l
Encoders:


PVRTC
 --channel-weighting-linear
 --channel-weighting-perceptual
 --bits-per-pixel-2
 --bits-per-pixel-4
 --alpha-is-independent
 --alpha-is-opacity
 --punchthrough-unused
 --punchthrough-allowed
 --punchthrough-forced
ASTC
 --block-width-4
 --block-width-5
 --block-width-6
 --block-width-8
 --block-width-10
 --block-width-12
 --compression-mode-veryfast
 --compression-mode-fast
 --compression-mode-medium
 --compression-mode-thorough
 --compression-mode-exhaustive
 --srgb-yes
 --srgb-no
 --block-height-4
 --block-height-5
 --block-height-6
 --block-height-8
 --block-height-10
 --block-height-12

Formats:


Raw
PVR
KTX
```

`texturetool` defaults to `--bits-per-pixel-4`, `--channel-weighting-linear` and `-f Raw` if no other options are provided.

The `--bits-per-pixel-2` and `--bits-per-pixel-4` options create PVRTC data that encodes source pixels into 2 or 4 bits per pixel. These options represent a fixed 16:1 and 8:1 compression ratio over the uncompressed 32-bit RGBA image data. There is a minimum data size of 32 bytes; the compressor never produces files smaller than this, and at least that many bytes are expected when uploading compressed texture data.

When compressing, specifying `--channel-weighting-linear` spreads compression error equally across all channels. By contrast, specifying `--channel-weighting-perceptual` attempts to reduce error in the green channel compared to the linear option. In general, PVRTC compression does better with photographic images than with line art.

The `-m` option automatically generates mipmap levels for the source image. These levels are provided as additional image data in the archive created. If you use the Raw image format, then each level of image data is appended one after another to the archive. If you use the PVR archive format, then each mipmap image is provided as a separate image in the archive.

The (`-f`) parameter controls the format of its output file. The default format is Raw. This format is raw compressed texture data, either for a single texture level (without the `-m` option) or for each texture level concatenated together (with the `-m` option). Each texture level stored in the file is at least 32 bytes in size and must be uploaded to the GPU in its entirety. The PVR format matches the format used by the PVRTexTool found in Imagination Technologies's PowerVR SDK. To load a PVR-compressed texture, use the `GLKTextureLoader` class.

The `-s` and `-c` options print error metrics during encoding. The `-s` option compares the input (uncompressed) image to the output (encoded) image, and the `-c` option compares any two images. Results of the comparison include root-mean-square error (`rms`), perceptually weighted `pRms`, worst-case texel error (`max`), and compositing-based versions of each statistic (`rmsC`, `pRmsC`, and `maxC`). Compositing-based errors assume that the image's alpha channel is used for opacity and that the color in each texel is blended with the worst-case destination color.

The error metrics used with the `-s` and `-c` options and by the encoder when optimizing a compressed image treat the image's alpha channel as an independent channel by default (or when using the `--alpha-is-independent` option). The `--alpha-is-opacity` option changes the error metric to one based on a standard blending operation, as implemented by calling `glBlendFunc( GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA )`.

PVR Texture compression supports a special punchthrough mode which can be enabled on a per 4x4 block basis. This mode limits the color gradations that can be used within the block, but introduces the option of forcing the pixel's alpha value to 0. It can defeat PVRTC smooth color interpolation, introducing block boundary artifacts, so it should be used with care. The three punchthrough options are:

- `--punchthrough-unused` — No punchthrough (the default option).
- `--punchthrough-allowed` — The encoder may enable punchthrough on a block by block basis when optimizing for image quality. This option generally improves the objective (per-pixel) error metrics used by the compression algorithm, but may introduce subjective artifacts.
- `--punchthrough-forced` — Punchthrough is enabled on every block, limiting color gradation but making it possible for any pixel in the block to have an alpha of 0. This option is provided principally for completeness, but may be useful when the results can be compared visually to the other options.

Important: Source images for the encoder must satisfy these requirements:

- Height and width must be at least 8.
- Height and width must be a power of 2.
- Must be square (height==width).
- Source images must be in a format that Image IO accepts in OS X. For best results, your original textures should begin in an uncompressed data format.

Important: If you are using PVRTexTool to compress your textures, then you must create textures that are square and a power of 2 in length. If your app attempts to load a non-square or non-power-of-two texture in iOS, an error is returned.

Listing C-2 Encoding images into the PVRTC compression format

```
Encode Image.png into PVRTC using linear weights and 4 bpp, and saving as
ImageL4.pvrtc
user$ texturetool -e PVRTC --channel-weighting-linear --bits-per-pixel-4 -o
ImageL4.pvrtc Image.png

Encode Image.png into PVRTC using perceptual weights and 4 bpp, and saving as
ImageP4.pvrtc
user$ texturetool -e PVRTC --channel-weighting-perceptual --bits-per-pixel-4 -o
ImageP4.pvrtc Image.png

Encode Image.png into PVRTC using linear weights and 2 bpp, and saving as
ImageL2.pvrtc
user$ texturetool -e PVRTC --channel-weighting-linear --bits-per-pixel-2 -o
ImageL2.pvrtc Image.png

Encode Image.png into PVRTC using perceptual weights and 2 bpp, and saving as
ImageP2.pvrtc
user$ texturetool -e PVRTC --channel-weighting-perceptual --bits-per-pixel-2 -o
ImageP2.pvrtc Image.png
```

Listing C-3 Encoding images into the PVRTC compression format while creating a preview

```
Encode Image.png into PVRTC using linear weights and 4 bpp, and saving the output as
ImageL4.pvrtc and a PNG preview as ImageL4.png
user$ texturetool -e PVRTC --channel-weighting-linear --bits-per-pixel-4 -o
ImageL4.pvrtc -p ImageL4.png Image.png

Encode Image.png into PVRTC using perceptual weights and 4 bpp, and saving the output
as ImageP4.pvrtc and a PNG preview as ImageP4.png
user$ texturetool -e PVRTC --channel-weighting-perceptual --bits-per-pixel-4 -o
ImageP4.pvrtc -p ImageP4.png Image.png

Encode Image.png into PVRTC using linear weights and 2 bpp, and saving the output as
ImageL2.pvrtc and a PNG preview as ImageL2.png
user$ texturetool -e PVRTC --channel-weighting-linear --bits-per-pixel-2 -o
ImageL2.pvrtc -p ImageL2.png Image.png

Encode Image.png into PVRTC using perceptual weights and 2 bpp, and saving the output
as ImageP2.pvrtc and a PNG preview as ImageP2.png
user$ texturetool -e PVRTC --channel-weighting-perceptual --bits-per-pixel-2 -o
ImageP2.pvrtc -p ImageP2.png Image.png
```

Note: It is not possible to create a preview without also specifying the `-o` parameter and a valid output file. Preview images are always in PNG format.

To load a PVR-compressed texture, use the `GLKTextureLoader` class.

For an example of working with PVR-compressed data directly, see the *PVRTextureLoader* sample.