

PVRTexTool

User Manual

Copyright © Imagination Technologies Ltd. All Rights Reserved.

This publication contains proprietary information which is subject to change without notice and is supplied 'as is' without warranty of any kind. Imagination Technologies and the Imagination Technologies logo are trademarks or registered trademarks of Imagination Technologies Limited. All other logos, products, trademarks and registered trademarks are the property of their respective owners.

Filename : PVRTexTool.User Manual.1.0.8.External.doc
Version : 1.0.8 External Issue (Package: POWERVR SDK REL_2.10@863921)
Issue Date : 19 Mar 2012
Author : Imagination Technologies Ltd

Contents

| | |
|---|-----------|
| 1. Introduction | 4 |
| 1.1. Software Overview..... | 4 |
| 1.1.1. PVRTexToolGUI..... | 4 |
| 1.1.2. PVRTexTool Command-Line | 4 |
| 1.1.3. PVRTexTool Plug-ins | 4 |
| 1.2. Document Overview | 4 |
| 2. PVRTexTool GUI | 5 |
| 2.1. Installation..... | 5 |
| 2.1.1. From Installer | 5 |
| 2.1.2. From GZIP..... | 5 |
| 2.2. Main Interface | 6 |
| 2.2.1. Multiple Document Interface..... | 6 |
| 2.3. Texture View Window | 7 |
| 2.3.1. View Panels..... | 7 |
| 2.3.2. Status Bar | 7 |
| 2.3.3. Display Tab | 8 |
| 2.3.4. MIP-Map Tab..... | 9 |
| 2.3.5. Info Tab | 9 |
| 2.4. Toolbars | 10 |
| 2.4.1. Quick Access Toolbar | 10 |
| 2.4.2. MIP-Map Toolbar..... | 12 |
| 2.5. Menus | 13 |
| 2.5.1. File Menu | 13 |
| 2.5.2. View Menu | 15 |
| 2.5.3. Edit Menu | 15 |
| 2.5.4. Window Menu..... | 16 |
| 2.5.5. Help Menu | 16 |
| 2.6. Dialogs | 17 |
| 2.6.1. Wrap Raw Data | 17 |
| 2.6.2. Compose Cube Map | 17 |
| 2.6.3. Load Separate Channels..... | 18 |
| 2.6.4. Options | 18 |
| 2.6.5. Pre-Process Texture | 19 |
| 2.6.6. Transform | 20 |
| 2.6.7. Encode Texture | 21 |
| 2.6.8. Properties | 23 |
| 2.6.9. Encoding Statistics | 23 |
| 2.6.10. Generate Font | 24 |
| 3. PVRTexTool Command-Line | 25 |
| 3.1. Installation..... | 25 |
| 3.1.1. From Installer | 25 |
| 3.1.2. From GZIP..... | 25 |
| 3.1. Usage Instructions | 25 |
| 3.2. Examples | 25 |
| 3.3. Command-Line Options..... | 26 |
| 4. PVRTexTool Plug-ins..... | 28 |
| 4.1. Adobe Photoshop | 28 |
| 4.2. Autodesk 3D Studio MAX | 28 |
| 4.3. Autodesk Maya | 28 |
| 5. Related Materials | 29 |
| 6. Contact Details | 30 |
| Appendix A. Texture Format Reference | 31 |
| A.1. DirectX 10 Formats..... | 33 |

| | | |
|------|--------------|----|
| A.2. | OpenVG | 35 |
|------|--------------|----|

List of Figures

| | | |
|-------------|-------------------------------------|----|
| Figure 2-1 | PVRTexToolGUI Main Interface | 6 |
| Figure 2-2 | View Panels | 7 |
| Figure 2-3 | Status Bar | 7 |
| Figure 2-4 | Display Tab | 8 |
| Figure 2-5 | MIP-Map Tab | 9 |
| Figure 2-6 | Info Tab | 9 |
| Figure 2-7 | Open | 10 |
| Figure 2-8 | Close | 10 |
| Figure 2-9 | Save | 10 |
| Figure 2-10 | Save All | 10 |
| Figure 2-11 | Undo | 10 |
| Figure 2-12 | Redo | 10 |
| Figure 2-13 | Encode | 10 |
| Figure 2-14 | Pre-Process | 10 |
| Figure 2-15 | Transform | 10 |
| Figure 2-16 | Image Properties | 11 |
| Figure 2-17 | Encoding Statistics | 11 |
| Figure 2-18 | Load Level | 12 |
| Figure 2-19 | Load Channel | 12 |
| Figure 2-20 | Save Level | 12 |
| Figure 2-21 | Generate MIP-Maps | 12 |
| Figure 2-22 | File Menu | 13 |
| Figure 2-23 | View Menu | 15 |
| Figure 2-24 | Edit Menu | 15 |
| Figure 2-25 | Window Menu | 16 |
| Figure 2-26 | Help Menu | 16 |
| Figure 2-27 | Wrap Raw Data Dialog | 17 |
| Figure 2-28 | Compose Cube Map Dialog | 17 |
| Figure 2-29 | Load Separate Channels Dialog | 18 |
| Figure 2-30 | Options Dialog | 18 |
| Figure 2-31 | Pre-Process Texture Dialog | 19 |
| Figure 2-32 | Transform Dialog | 20 |
| Figure 2-33 | Encode Texture Dialog | 21 |
| Figure 2-34 | Properties Dialog | 23 |
| Figure 2-35 | Encoding Statistics Dialog | 23 |
| Figure 2-36 | Generate Font Dialog | 24 |

1. Introduction

1.1. Software Overview

PVRTexTool is a utility designed to convert image files into hardware-friendly texture files. PVRTexTool is also able to perform some simple pre-processing effects on textures such as normal map and cube map generation, colour bleeding and more. It comes in three forms; PVRTexTool GUI, a graphical application; PVRTexTool Command-Line, a command-line tool; and a series of plug-ins for Adobe Photoshop, Autodesk 3D Studio Max, and Autodesk Maya.

1.1.1. PVRTexToolGUI

PVRTexToolGUI is the graphical version of PVRTexTool; it is available for Windows, Linux and Mac OS. Only the executable is required to run on Windows, Linux and Mac OS require X11.

1.1.2. PVRTexTool Command-Line

PVRTexToolCL is the command-line version of PVRTexTool; it is available for Windows, Linux, and MacOS; only the executable is required. Its purpose is to allow the easy batching of texture conversion and compression operations via call from script or batch file.

1.1.3. PVRTexTool Plug-ins

The PVRTexTool plugins are designed to give various programs access to the functionality of PVRTexTool. Photoshop gains the ability to load and save .pvr files; 3D Studio Max and Maya gain the ability to use .pvr files when applying materials and the ability to save rendered images in .pvr format (at 32 bits per pixel only).

1.2. Document Overview

The purpose of this document is to serve as a complete user manual for PVRTexTool. It includes installation instructions, a guide to the functionality of the application, a complete listing of all interface options and preferences for the GUI, as well as a listing of all command-line options available for PVRTexToolCL.

2. PVRTexTool GUI

2.1. Installation

2.1.1. From Installer

Download either the PowerVR Insider SDK or the individual PVRTexTool package and follow the on screen instructions. Once the package has successfully installed the application will be available in:

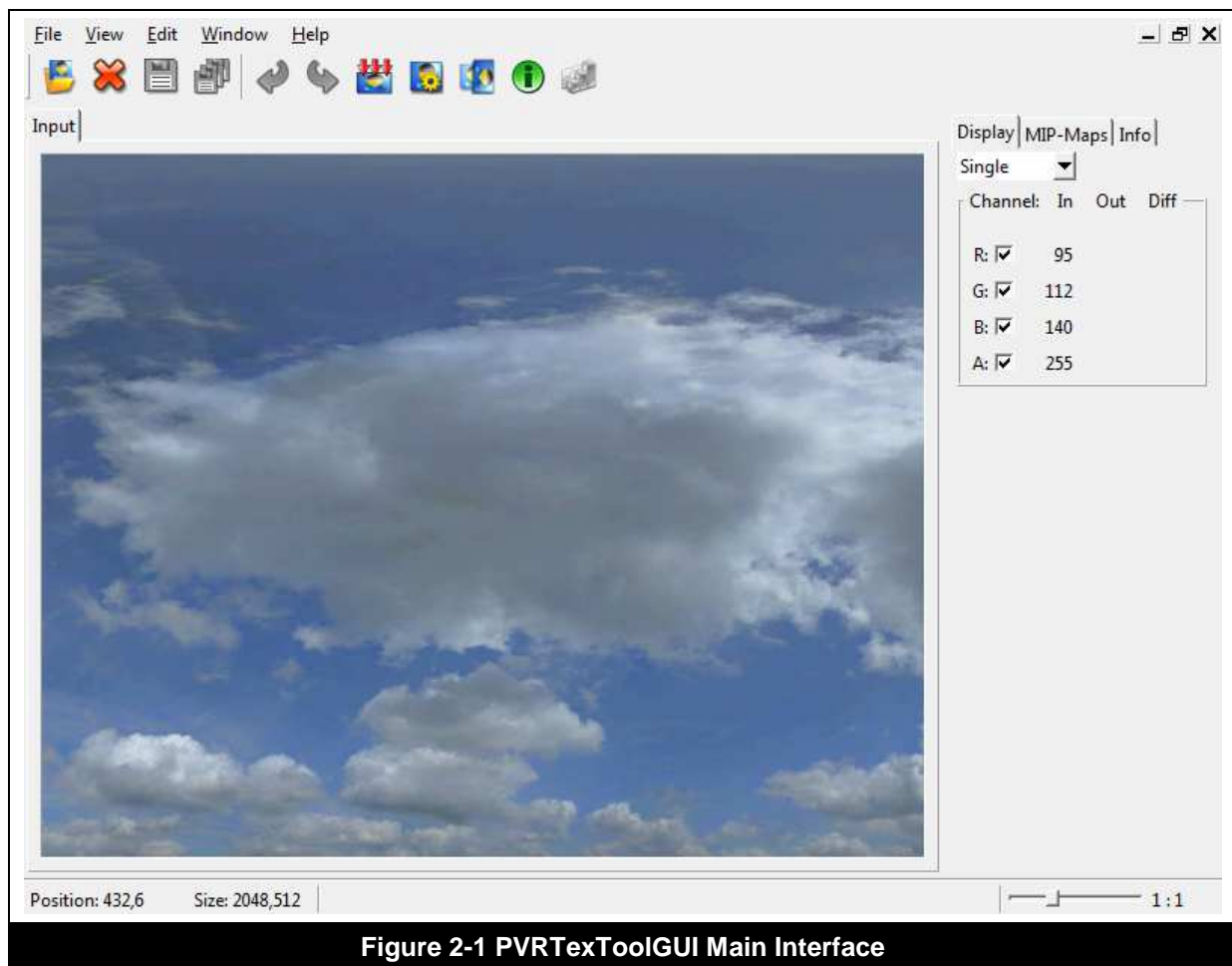
```
<SDK_ROOT>\Utilities\PVRTexTool\PVRTexToolGUI\<PLATFORM>\
```

2.1.2. From GZIP

Download either the PowerVR Insider SDK or the individual PVRTexTool package. Unzip the .tar.gz file, and then untar the .tar file. From the ensuing folder browse to:

```
<SDK_ROOT>\Utilities\PVRTexTool\PVRTexToolGUI\<PLATFORM>\
```

2.2. Main Interface



2.2.1. Multiple Document Interface

The main interface of PVRTexToolGUI contains the Texture View Window. As an MDI (multiple document interface) application PVRTexToolGUI may have multiple files open simultaneously displaying a Texture View Window for each. In these instances any action performed will be performed on the Texture View Window that currently has focus.

2.3. Texture View Window

This window can display the currently selected MIP-map level of a texture in a number of different ways, along with a variety of useful information.

2.3.1. View Panels

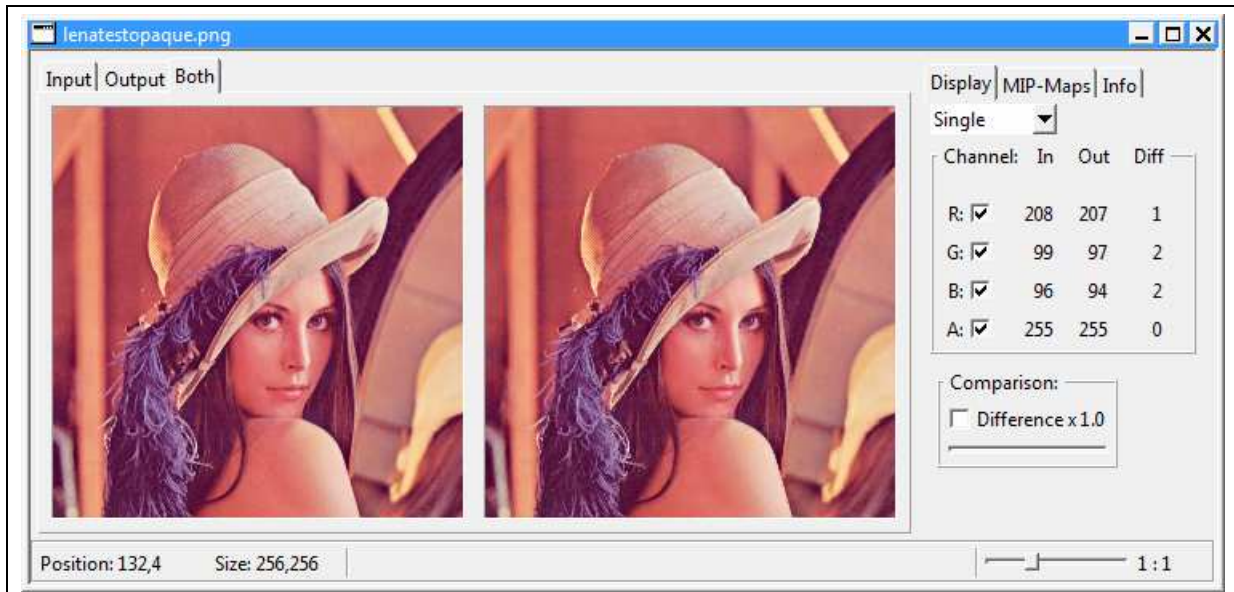


Figure 2-2 View Panels

Each image viewing window shows the current image data before (Input) and after (Output) encoding or one of each state via tab selection.

The Input view panel shows the loaded texture data before the encoding step.

The Output image is a decoded version of the encoded data to allow by-eye assessment, and is analogous to how the image will look when displayed on hardware.

If the images have transparency, they will be alpha blended with a background. The images can be moved by clicking and dragging if they are not fully visible.

2.3.2. Status Bar

At the bottom of the window is a status bar detailing basic information about the current status of the texture.



Figure 2-3 Status Bar

Position

As the cursor moves over the image, this details the mouse position within each texture.

Size

For quick reference the size of the texture being viewed is displayed here.

Zoom

Use this slider to zoom in and out of an image. The zoom feature can also be operated using the mouse wheel on any of the view panels, and is specific to each Texture View Window.

2.3.3. Display Tab

This tab shows information and options for the current display windows.

Display Mode

This dropdown menu is used to select how the image should be displayed. If the file is a normal 2D texture, you can choose between a single and a tiled view. If you are viewing a cube map, you can choose between a single view of the current face and a 3D preview.

Channels

Each channel provides an output value for the pixel that the mouse is currently situated above, its corresponding output pixel, and a difference value. Un-checking each channel will prevent this channel from drawing.

Difference

Activating the 'Difference' checkbox displays a representation of the difference between the original image and the image after encoding (multiplied by the value set with the scale slider), to highlight possible encoding and compression artifacts.

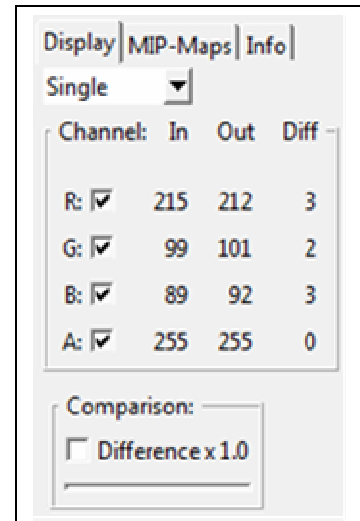


Figure 2-4 Display Tab

2.3.4. MIP-Map Tab

This tab shows information and options for MIP-Map levels present in the current texture.

MIP-Map Browser

This collapsible menu shows a list of all MIP-map levels in the texture. When a cube map is loaded each face of the cube map appears as a separate collapsible menu with its own MIP-map levels.

Click on a MIP-map to view it in the Texture View Window. Right clicking on a MIP-map will bring up a menu displaying the following options:

- 'Load Into MIP Level...' – See Section 2.4.2 MIP-Map Toolbar – Load Level.
- 'Save from MIP Level...' – See Section 2.4.2 MIP-Map Toolbar – Save Level.
- 'Generate MIP Levels from here...' – See Section 2.4.2 MIP-Map Toolbar – Generate MIP-Maps.

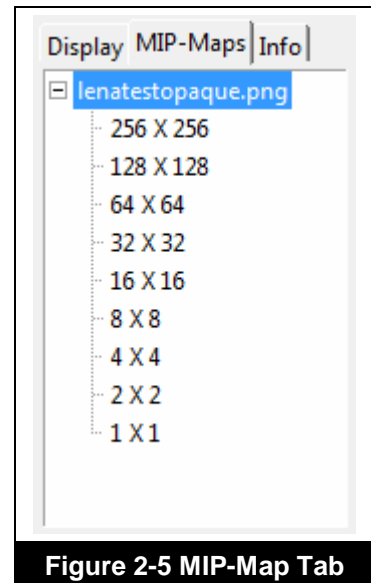


Figure 2-5 MIP-Map Tab

2.3.5. Info Tab

This tab displays some basic information about the textures in the view panels. Further information can be viewed in the full properties dialog found in 'View -> Get Properties...'

Texture Name

This area displays the filename of the current texture.

Original Image

'Original Image' displays information about the image being converted. The data shown includes the original dimensions of the image, the file size, how many MIP-map levels were originally present and the pixel format (for more information see Appendix A. Texture Format Reference).

Encoded Texture

This area displays information about the encoded image. The data shown includes the dimensions of the converted image, the new file size, how many MIP-map levels are now present, and the pixel format (for more information see Appendix A. Texture Format Reference).

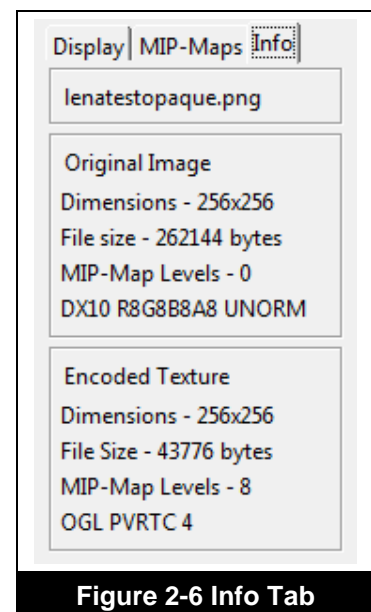


Figure 2-6 Info Tab

2.4. Toolbars

2.4.1. Quick Access Toolbar

Open

'Open' opens a texture for editing.



Figure 2-7 Open

Close

'Close' closes the texture.



Figure 2-8 Close

Save

'Save' saves the texture.



Figure 2-9 Save

Save All

'Save All' saves all the textures currently open in PVRTexToolGUI.



Figure 2-10
Save All

Undo

'Undo' undoes the last performed action.



Figure 2-11
Undo

Redo

'Redo' redoes the last undone action.



Figure 2-12
Redo

Encode

'Encode' launches the Encode Texture dialog.



Figure 2-13
Encode

Pre-Process

'Pre-Process' launches the Pre-Process Texture dialog.



Figure 2-14 Pre-
Process

Transform

'Transform' launches the Transform dialog.



Figure 2-15
Transform

Image Properties

'Image Properties' launches the Properties dialog.



Figure 2-16 Image Properties

Encoding Statistics

'Encoding Statistics' launches the Encoding Statistics dialog.



Figure 2-17 Encoding Statistics

2.4.2. MIP-Map Toolbar

Load Level

'Load Level' loads an image into the currently selected MIP-map level. This image must have the same dimensions as the selected MIP level.



Figure 2-18
Load Level

Load Channel

'Load Channel' loads one or more images into colour channels for the currently selected MIP-map level. Images must be the same size as the selected MIP-map level.



Figure 2-19
Load Channel

Save Level

This option saves an image file of the current MIP-map level.



Figure 2-20
Save Level

Generate MIP-Maps

'Generate MIP-Maps' uses the currently selected MIP-map to generate all smaller MIP-maps.



Figure 2-21
Generate MIP-Maps

2.5. Menus

2.5.1. File Menu

Open

'Open...' opens a texture or image file. PVRTexTool supports the texture formats:

- .pvr - PowerVR texture files
- .ktx - Khronos texture files
- .dds - Microsoft Direct Draw Surface files

It can also read the following image formats: BMP, TGA, GIF, PCX, JPG, and PNG.

Open Recent

This shows a list of up to 10 recently opened files for quick access.

Compose Cube Map...

Use this option to create a cube map texture by combining existing images. It allows you to specify image files from which to load each of the six faces of the cube.

Generate Font...

This option launches the dialog, used to generate texture atlases for use as fonts with Print3D.

Open to Separate Channels...

This menu option opens the Load Separate Channels dialog; it allows several image files to be combined into a single texture with different channels being taken from different images. Integer values can also be used instead of filenames to set an image-wide value for a given channel.

Wrap Raw Data

This option opens the 'Wrap Raw Data' dialog; a dialog used to load raw image data from a bitmap file or a corrupt texture file.

Reload From File

'Reload From File...' reloads the current texture from disk reverting any pre-processing already carried out. If the file has been updated in some way by another program since being opened this also allows the texture in memory to be updated to that which is currently stored on disk. Any encoded data produced is discarded by this operation.

In instances where multiple files have been used, or the file has been processed from raw data, the relevant dialog box will be launched instead of the file automatically reloading. This allows for channels to be rearranged and options to be adjusted etc. in the case of a mistake being made.

Save/Save As

These options will save encoded data to a texture file, and can be saved in one of the following formats:

- .pvr - PowerVR texture files
- .ktx - Khronos Texture files
- .dds - Microsoft Direct Draw Surface files
- .h - C/C++ Header file storing PVR data in an array of type 'unsigned long'

If the texture has not yet been encoded, PVRTexTool will prompt the user to select an encoding method from the Encode Texture dialog (see Section 2.6.7 Encode Texture)

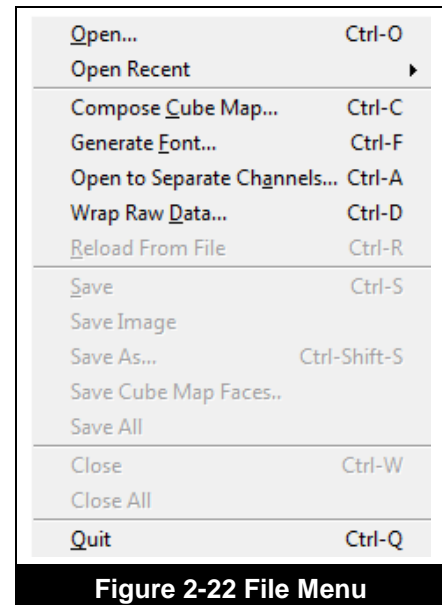


Figure 2-22 File Menu

Save Image

This option allows the user to save the currently displayed image to an image file, rather than a texture file. Note that it will save out the current MIP-map level if a lower level is being displayed and all other MIP-map data will be lost. This can be used to save out each MIP-map level individually if needed. This will automatically append the dimensions of the image to the end of the filename.

Images can be saved as the following formats:

- .bmp
- .tga
- .gif
- .pcx
- .jpg
- .png

Save Cube Map Faces

This option is similar to Save Image with the exception that it will ask to save each face of the cube map individually for the currently selected MIP-Map level. This will automatically append the name of each face to the image file, as well as the image dimensions.

Save All

This option performs the same operation as Save, but for all open textures.

Close

'Close' closes the current texture

Close All

'Close All' closes all the currently open textures.

Quit

'Quit' closes the application.

2.5.2. View Menu

This Menu shows options for displaying information or changing the appearance of some elements.

View Grid

This option will overlay a grid onto the current Texture View Window. The size of this grid is determined via the Options dialog.

Get Properties...

This option opens the Properties dialog.

Get Encoding Statistics...

This option opens the Encoding Statistics dialog.

Options...

'Options...' opens the Options dialog.

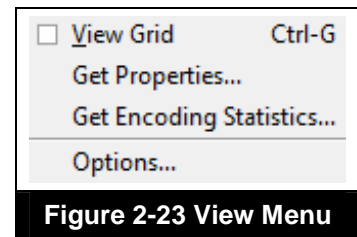


Figure 2-23 View Menu

2.5.3. Edit Menu

Undo

'Undo' undoes the last performed action.

Redo

'Redo' redoes the last undone action.

Encode Texture...

This option opens the Encode Texture dialog.

Pre-process...

This option opens the Pre-Process Texture dialog.

Transform...

'Transform...' opens the Transform dialog.

Load/Swap Channels

'Load/Swap Channels' opens the Load Separate Channels dialog.

MIP-Map Level

'MIP-Map Level' is an additional sub-menu containing operations which apply solely to the currently selected MIP-map level. These are:

- Load MIP-level... - Load an image into a specific MIP-map level; the image must be the same height and width as the MIP-map level being replaced.
- Save MIP-level... - Save an image from a specific MIP-map level; the filename will have the dimensions of the level appended to it.

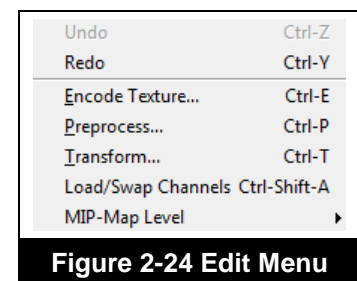


Figure 2-24 Edit Menu

2.5.4. Window Menu

The window menu contains options pertaining to the displaying of multiple Texture View Windows.

Tile Horizontally

'Tile Horizontally' tiles the currently open Texture View Windows horizontally.

Tile Vertically

'Tile Vertically' tiles the currently open Texture View Windows vertically.

Cascade

'Cascade' displays all the open Texture View Windows in a cascade style.

Currently Open

The bottom of the menu lists all the currently open Texture View Windows.

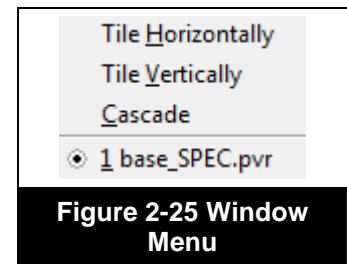


Figure 2-25 Window Menu

2.5.5. Help Menu

Help

'Help...' opens this document.

About

'About...' opens an about page containing version information, contact details etc.

Feedback

'Feedback' opens a panel for giving feedback on the application.

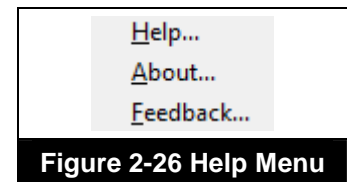


Figure 2-26 Help Menu

2.6. Dialogs

2.6.1. Wrap Raw Data

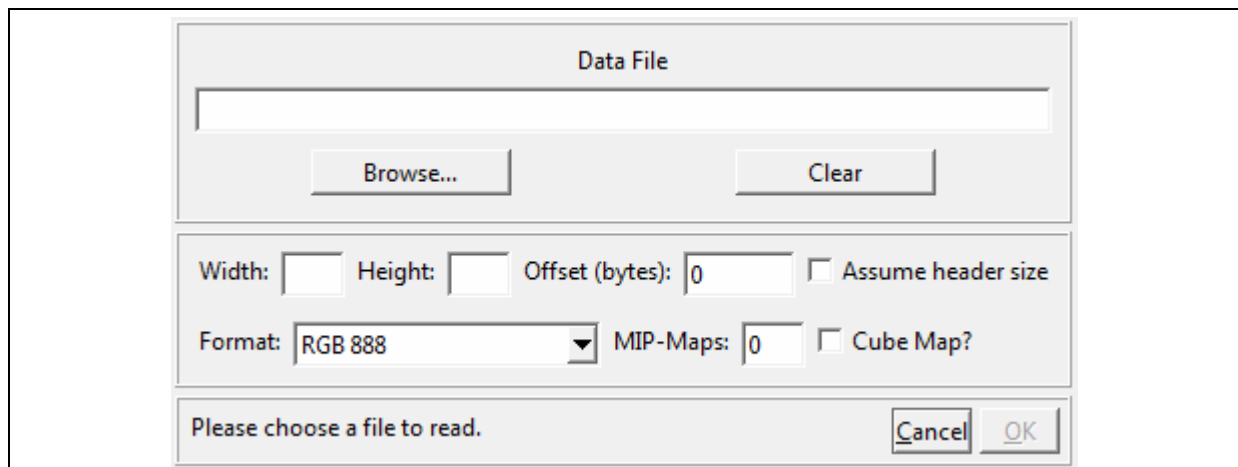


Figure 2-27 Wrap Raw Data Dialog

This dialog is used to load raw image data from a bitmap file or a corrupt texture file. The user must input the size of the image, pixel format, number of MIP-maps, and whether or not it is a cube map (6 surfaces).

If there is a header, the user can either specify the size or let PVRTexTool calculate it based on the file size. This header data will then be skipped when the image is loaded. It should be noted that automatic calculation will not work correctly if there is additional data at the end of the file.

2.6.2. Compose Cube Map

This dialog is used to create a cube map texture by combining existing images. It allows you to specify image files from which to load each of the six faces of the cube. All of these source images must be of identical, square dimensions.

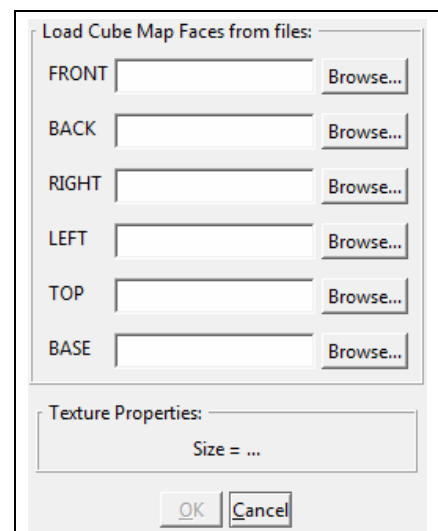


Figure 2-28 Compose Cube Map Dialog

2.6.3. Load Separate Channels

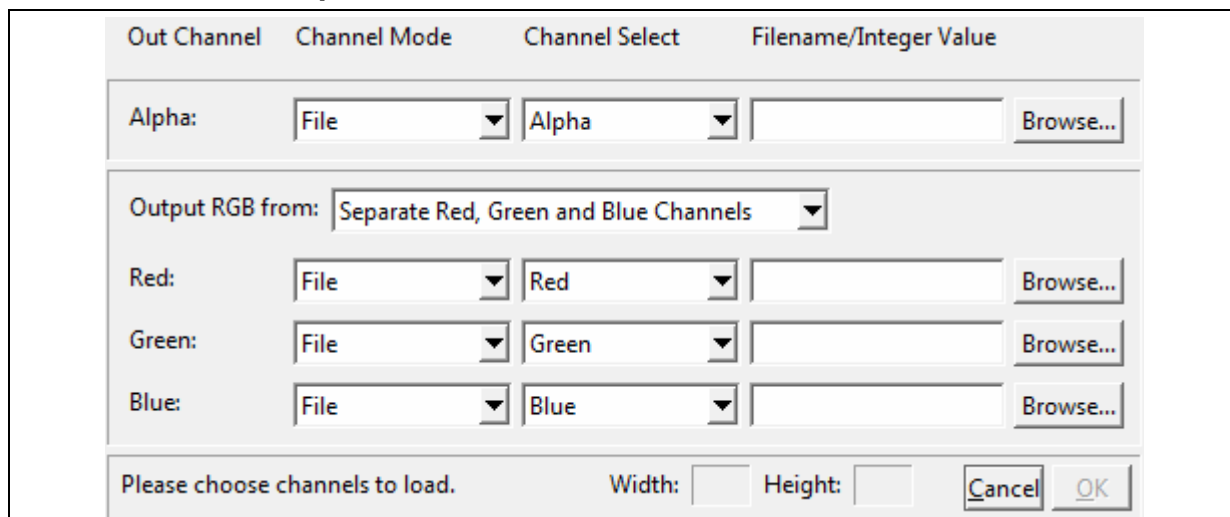


Figure 2-29 Load Separate Channels Dialog

For each channel a source ('Current', 'File' or 'Value') can be selected from the first drop down. The second drop down indicates which channel from the source you wish to load into the destination. The final entry box can contain either an integer value in the range 0 to 255 (if 'Value' is selected from the first drop down), or a file path (if 'File' is selected from the first drop down) either entered manually or selected using the 'Browse...' button. The 'Current' option is only available when a file is open and allows channels to be swapped within the current image.

Finally, it is possible for the red, green and blue channels to all be loaded from the same file, this can be done by changing the value in the 'Output RGB from:' drop down.

2.6.4. Options

These options allow the user to set some basic GUI options: the background colour in the Texture View Windows and the size of the grid displayed in the View Panels when it is activated from the View Menu.

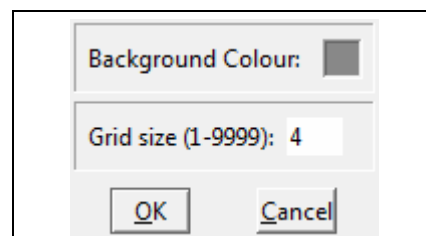


Figure 2-30 Options Dialog

2.6.5. Pre-Process Texture

Create Normal Map

A normal map is a texture that stores normal vectors instead of colours. The X component of the normal vector is stored in the red channel of the texture, the Y component in the green channel and the Z component in the blue channel. Normal maps are commonly used with Dot3 bump mapping. The normal map is calculated from a grey scale height map that indicates the roughness of the surface (or the red channel of the height map if the height map is not grey scale). Large values in the map mean taller heights, while small values mean lower heights. The values come from the red channel of the input texture.

A normal vector is calculated per pixel using the difference between the intensity of adjacent pixels and then is compressed into the format selected. The scale value modifies how it interprets the difference between the pixels; the higher the scale, the bigger the difference in normal values. This is analogous to the scale used by a height map to generate terrain. Note that internally, the format of normal maps is automatically raised to 32-bit floating point precision.

Once the top MIP-map level has been converted to a normal map, lower MIP-map levels are regenerated from this image using the standard 2 by 2 averaging algorithm. This will give physically correct results for per pixel lighting on matte surfaces.

Make Square

Checking this option will take the shortest dimension of the image and resize it to equal the longest side of the image. In short, it will scale up the image to be square. The resize operation will use the rescaling method set in the drop down menu above.

Expand to Power of 2

This function will expand any size of texture so that its dimensions have values of powers of two, although the dimensions may not necessarily be equal (unless the Make Square option is chosen). Again, the resize operation uses the rescaling method set in the drop down menu above.

Colour MIP-maps

Sometimes, for testing purposes, it's useful to mark each MIP-map in a MIP-map chain by giving it a colour. This option goes through the existing MIP-map chain and colours the levels green, red, yellow, pink, cyan, and blue in descending order. It leaves the top MIP-map level unchanged.

Generate MIP-maps

In most cases, PVRTexTool automatically generates MIP-maps upon loading an image, however if they are not present or corrupt, this option can be checked to regenerate them.

Alpha Bleed

When mapping certain parts of a texture on an object, the texture may contain an invisible void between useful parts. This void creates discontinuities around the textured parts that can impair the texture compression with certain formats. Uncompressed textures can also benefit from this bleeding operation as texture filtering (bilinear and/or MIP-mapping) can cause undesired adjacent texels to contribute to the final colour being sampled. To avoid this issue, the bleeding process transforms these frontiers by filling the void with a mix from nearby pixels.

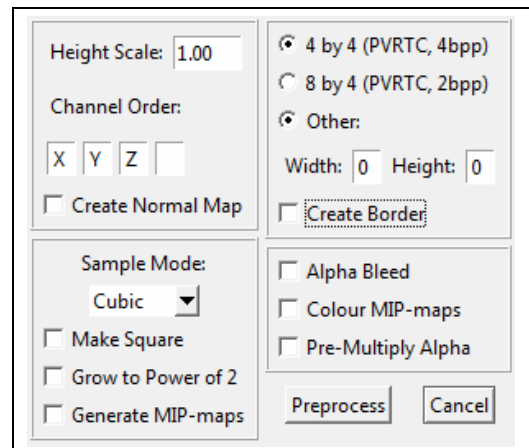


Figure 2-31 Pre-Process Texture Dialog

Add Border

PVRTC texture data is assumed to be continuous across texture edges, which is a common case in graphic applications where various material textures like rock, brick, grass etc. are tiled to represent high texture detail for a larger area than the texture itself. However, this can occasionally result in minor compression artefacts along the edges of texture data that does not tile.

Whenever results are deemed unsatisfactory, this issue can be resolved by adding a border around the original texture data. This border will absorb any possible artefacts related to tiling.

This option has two possible functions depending on the width and height of the texture:

- Width and height are a power of 2 – In this case the user can select one of two options. The algorithm will scale the texture down by 8 pixels in the vertical direction (4 on each side) and either 8 or 16 pixels in the horizontal direction, and then add mirrored borders.
- Width and/or height are not a power of 2 – In this case the border is directly added so as to expand the image dimensions to their nearest respected powers of 2.

A texture pre-processed in this way will have an edge of 4x4 pixels for the PVRTC 4bpp case and an edge of 8x4 for the PVRTC 2bpp case. This texture has to be remapped correctly to restore it to its original size. This remapping can be calculated as follows:

- PVRTC 4bpp:
 - $u = ((4+1)/ResX) + u * (1 - (2*(4+1)/ResX))$
 - $v = ((4+1)/ResY) + v * (1 - (2*(4+1)/ResY))$
- PVRTC 2bpp:
 - $u = ((8+1)/ResX) + u * (1 - (2*(8+1)/ResX))$
 - $v = ((4+1)/ResY) + v * (1 - (2*(4+1)/ResY))$

(Note that 1 is added to the border size to avoid bilinear bleeding. ResX and ResY are the original texture resolution)

Pre-Multiply Alpha

For alpha-blended rendering it is sometimes useful to have the other, opaque channels of a texture encoded with their value pre-multiplied by the alpha value for each pixel. Checking this option will carry this out.

2.6.6. Transform

The transform dialog allows basic geometric transformations to be applied to the current texture.

Resize

Enter the desired dimensions, in pixels, into the Width and Height boxes, choose the scaling algorithm you desire and click 'Go'. 'Nearest Neighbour', 'Bilinear' or 'Bicubic' scaling is available.

Mirror

Basic reversal in the horizontal and vertical axis is available through this section.

Rotate

Click the options in this section to rotate the current texture by 90 degrees in a clockwise or anti-clockwise direction.

For cube map textures, only the mirror and rotate options are available; these may be applied to the currently selected surface in the MIP-Map Tab or all surfaces at once.

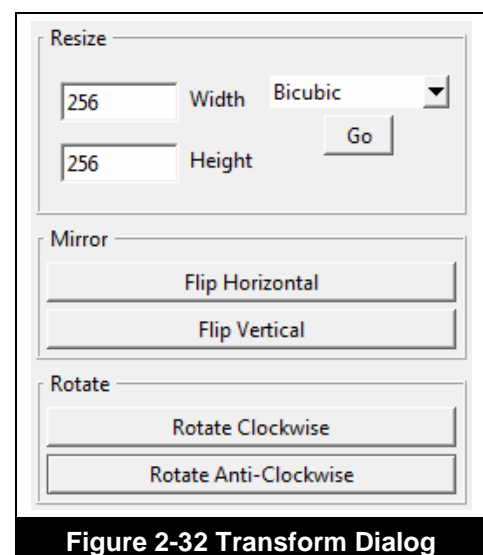


Figure 2-32 Transform Dialog

2.6.7. Encode Texture

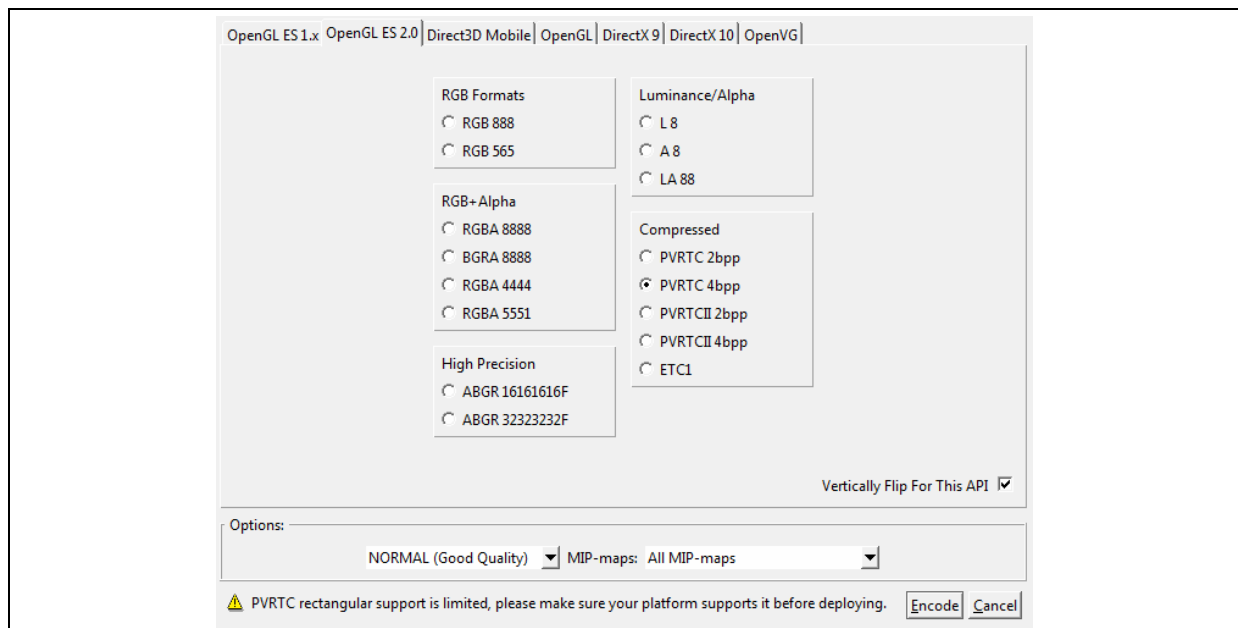


Figure 2-33 Encode Texture Dialog

APIs

The several tabs along the top allow selection of the target API.

Pixel Formats

This area lists the formats that can be encoded for each API.

MIP-Map Levels

It is possible to choose the number of MIP-maps to be encoded within a texture. This can be selected from the 'MIP-maps:' drop down.

Vertical Flip

For each API you can set whether the texture is flipped vertically or not. By default in the PowerVR SDK OpenGL, OpenGL ES 1.x and OpenGL ES 2.0 textures are flipped.

OpenVG textures should be correctly orientated for use with this API and do not have a flip option.

NB: This option will not change how the texture appears visually, merely the ordering of data.

PVRTC Quality

Four compression methods are available for PVRTC: 'Fast (Medium Quality)', 'Normal (High Quality)', 'High (Very High Quality)' and 'Best (Best Quality)'. Fast compression is generally considered the most useful for development, as it combines reasonable quality with high speed and is useful for quick asset iteration. Normal quality is recommended for general use as it takes a short amount of time and provides great quality images. High and Best should only be used if textures are particularly prone to artefacts, as these are much slower compression methods.

ETC Quality

This option varies the setting used by the ETC compressor. Note that Medium and Slow settings may produce higher quality results, but can take very long periods to complete encoding, even on modern hardware.

Dithering

When encoding to non-compressed formats that have channel precision of less than 8 bits, a dithering option is available; this can be activated or deactivated by ticking or un-ticking the 'Dithering' tick box.

2.6.8. Properties

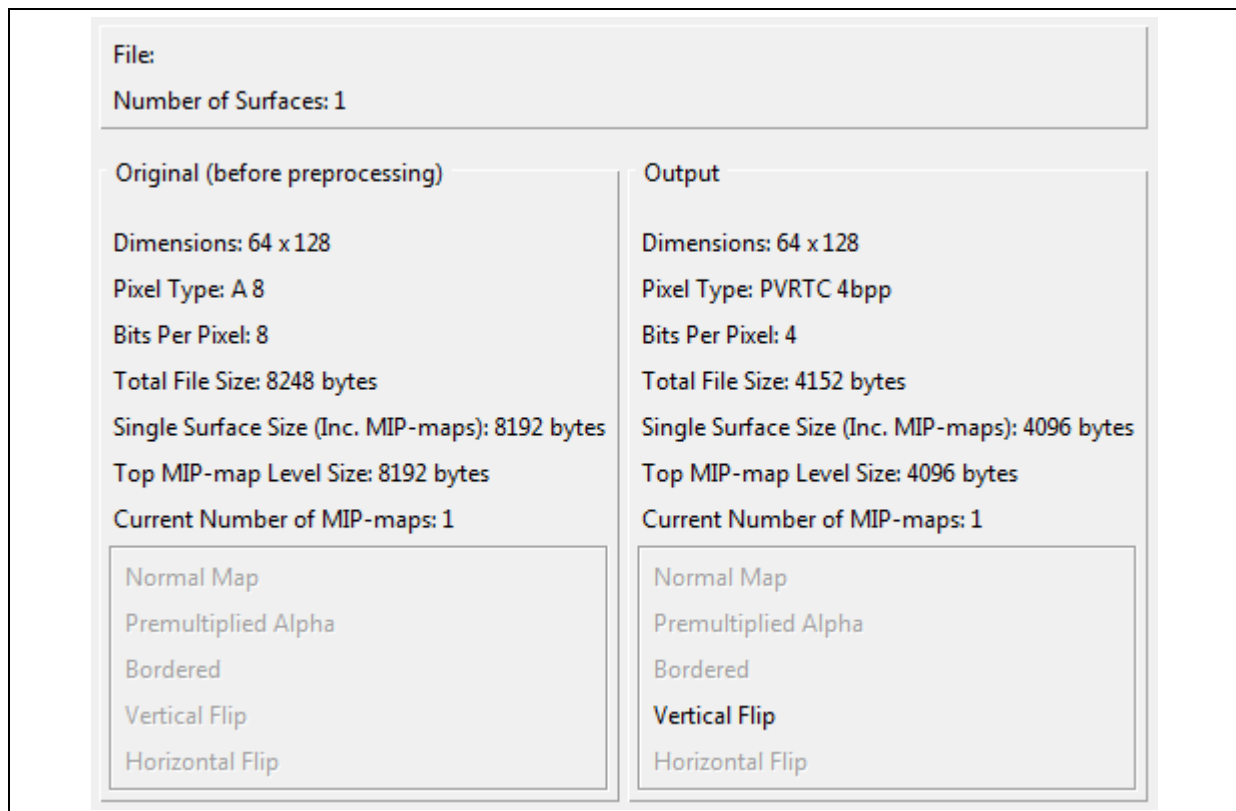


Figure 2-34 Properties Dialog

This option displays information about the currently active texture in a floating dialog. More information is presented here than in the Info Tab of the Texture View Window.

2.6.9. Encoding Statistics

| MIP-map level: 0 Face: 0 | | | | | |
|--------------------------|-----------|-----------|-----------|----------|-----------|
| Encoding Error Metrics | Red | Green | Blue | Alpha | All |
| Mean Error | 3.504913 | 3.512497 | 3.828842 | 0.000000 | 2.711563 |
| Standard Deviation | 5.974660 | 5.973378 | 6.014302 | 0.000000 | 4.490585 |
| Mean Square Error | 4.983157 | 4.949515 | 5.406291 | 0.000000 | 3.834741 |
| PSNR db | 41.155758 | 41.185181 | 40.801811 | n/a | 42.293442 |
| Maximum Difference | 51 | 48 | 50 | 0 | 51 |

Figure 2-35 Encoding Statistics Dialog

This shows some error statistics concerning the difference between input and output images of the current texture, the mean error, the standard deviation, the mean square error, the maximum difference and the peak signal-to-noise ratio.

2.6.10. Generate Font

The 'Generate Font' dialog is used to generate texture atlases for use with Print3D; part of the PVRTools found within the PowerVR Insider SDK for displaying text within an application. It uses a font from a TTF or OTF to generate the texture and atlas data required for this purpose.

Once all options are set as required and 'Generate' has been clicked a 'Save File' dialog will appear. The file saved from this dialog contains the texture and header information needed by Print3D.

Font

Font File

A file path representing the location of the font file to be used in the texture atlas can be entered here, either manually, or through the use of the 'Browse...' option.

Size

This dropdown box allows the user to select the font size that is to be written into the texture atlas.

Border

'Border' represents the size of the border that should be left around each character within the texture atlas.

Grid Fitting

'Grid Fitting' is a form of font hinting that may produce a more desirable output font, varying from font to font, and by user preference.

Character Set

The three options in this section detail the character set that is to be placed into the texture atlas. By default the 7-bit ASCII character set is used, a text file containing a list of the characters may also be used, or the desired characters (including non-English characters) may be pasted into the 'Input' box.

Texture

Generate MIP-maps

With 'Yes' ticked, a full MIP-map chain will be generated for the output file.

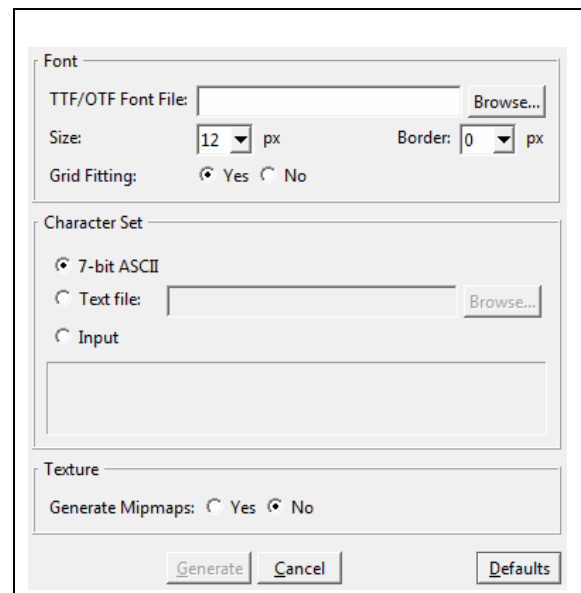


Figure 2-36 Generate Font Dialog

3. PVRTexTool Command-Line

3.1. Installation

3.1.1. From Installer

Download either the PowerVR Insider SDK or the individual PVRTexTool package and follow the on screen instructions. Once the package has successfully installed the application will be available in:

```
<SDK_ROOT>\Utilities\PVRTexTool\PVRTexToolCL\<PLATFORM>\
```

3.1.2. From GZIP

Download either the PowerVR Insider SDK or the individual PVRTexTool package. Unzip the .tar.gz file, and then untar the .tar file. From the ensuing folder browse to:

```
<SDK_ROOT>\Utilities\PVRTexTool\PVRTexToolCL\<PLATFORM>\
```

3.1. Usage Instructions

PVRTexTool can be used from the command-line to be able to process and compress textures using a batch file. The syntax for the command-line is as follows:

```
PVRTexTool -f<format> -i<inputfilename> [-a<alphafilename>] [-b<factor>] [-border] [-d] [-dds*] [-dither] [-e] [-h] [-help] [-l] [-m] [-nt] [-ngt] [-o<outputfilename>] [-ob<channelorder>] [-p] [-premultalpha] [-pvrctmethod] [-pvrctfast] [-pvrctnormal] [-pvrcthigh] [-pvrctbest] [-pvrlegacy] [-q<level>] [-r<algorithm>] [-silent] [-square] [-s<filename>] [-x<width>] [-y<height>] [-yflip<0,1>]
```

*dds is only available in the Windows version of PVRTexTool.

PVRTexTool allows the following three methods of option input:

No space (-f<format>), spaced (-f <format>) or equality (-f=<format>).

It should be noted that using a filename value for -i or -o which begins with '=' will work only with the last two options, e.g. '-i=Example.bmp' will read the filename as "Example.bmp" even if the file is called '=Example.bmp'. In this case '-i =Example.bmp' or '-i==Example.bmp' will work correctly.

3.2. Examples

To encode the file Example.bmp as a binary PVR file with pre-generated MIP-maps in ARGB 1555 format.

```
PVRTexTool -m -f1555 -iExample.bmp
```

To encode a sky box from files named skybox_n.bmp as an include header in OpenGL PVRTC 4bpp format:

```
PVRTexTool -h -s -m -foglrvrtc4 -iskybox1.bmp
```

3.3. Command-Line Options

| Option | Notes |
|-------------------|--|
| -a | Input alpha filename (.pvr also BMP, JPG, PNG, GIF or TGA file). |
| -b[factor] | Bump/Normal map. This option calculates the normal map from a height map passed as input. [factor] is a multiplication factor for the normal map. Default value is 2.0. |
| -border | Pre-processes texture with a mirrored border around the texture. Works the same as the GUI option. A texture with dimensions that are non-power of two will be expanded to power of two dimensions. For power of two input images the border will be of 4 pixels all round, unless the chosen format is PVRTC2 or OGLPVRTC2 in which case the border generated is 4 pixels at top and bottom and 8 pixels at the sides of the texture. |
| -d | Create output file(s) with decompressed texture data. |
| -dds | Create a Microsoft Direct Draw Surface file. This option is only available on the Windows version of PVRTexTool. |
| -dither | When encoding to non-compressed formats that have channel precision of less than 8 bits, this option enables dithering |
| -e | Creates false colour MIP-map levels rather than true levels. The -m must also be set or no MIP-map levels will be generated for false colouring. |
| -f | Output file format. For example -f PVRTC4 |
| -h | Create include file with texture data. |
| -help | Displays information about PVRTexTool similar to that presented here. |
| -i | Input filename (BMP, JPG, PNG, GIF or TGA file, or DDS on Windows). |
| -l | Apply a bleed filter to the texture and its MIP-maps based on the alpha channel. |
| -m | Automatically generate all MIP-map levels. |
| -nt | No twiddle. Using this option whilst compressing to a PVRTC format will have no effect as PVRTC compressed textures are always twiddled. |
| -o | Output filename. If this option is not used the output filename will be the same as the input filename with the extension .pvr or .h depending on the relevant options also passed. PVRTexTool will override an incorrect extension. |
| -ob[channelorder] | Used in conjunction with the '-b' flag, this flag allows the channel order to be set. E.g. yzx. |
| -p | Create binary PVR file with texture data. This is the default setting if the output type is not specified. |
| -premultalpha | Multiplies the RGB values in the texture by the alpha values in the pre-processing phase of texture encoding. |
| -pvrctmethod | Values 0-3 choose a mode for PVRTC compression. The default value is 0, which is Normal compression. 1=Fast Compression, 2=High Quality Compression and 3=Best Quality Compression. |
| -pvrctfast | Uses the fast, medium quality PVRTC compressor. |
| -pvrctnormal | Uses the normal, high quality PVRTC compressor. This is the default. |
| -pvrcthigh | Uses the very high quality PVRTC compressor. |
| -pvrctbest | Uses the best quality PVRTC compressor. |
| -pvrlegacy | Saves the output texture as a legacy PVR file. |

| Option | Notes |
|---------|---|
| -q | Quality mode for ETC compression. 0 = Fast, 1 = Medium, 2 = Slow, 3 = Fast Perceptual, 4 = Medium Perceptual, 5 = Slow Perceptual. Default is 3. |
| -r | Choose a resizing algorithm. 1 = nearest, 2 = bilinear, 3 = bicubic. Default is bicubic. |
| -s | Compress a skybox. Files must be named XXXXXn where n=1-6. |
| -silent | Force PVRTexTool into "silent" mode. |
| -square | Convert the texture into a square texture with power of two dimensions. |
| -x | Define a new width for the output texture. |
| -y | Define a new height for the output texture. |
| -yflip | Flips the texture vertically. Use -yflip 1 to force flipping; -yflip 0 for no flip. NB: This option will not change the texture visually if viewing in an image viewer; it only changes the data ordering. |

4. PVRTexTool Plug-ins

Plugins are available for Adobe Photoshop, Autodesk 3D Studio Max, Autodesk Maya on Windows; 32bit versions are available for all three, 64bit versions are available for Maya and 3DS Max; the 64bit plugins for Maya and 3DS Max only support version 2010 onwards.

4.1. Adobe Photoshop

Copy: `<SDK_ROOT>\Utilities\PVRTexTool\Photoshop\<PLATFORM>\PVRFormat.8bi`

To: `<PHOTOSHOP_DIR>\plug-ins\File Formats\`

Once installed .pvr will be available as one of the formats supported for loading and saving images.

4.2. Autodesk 3D Studio MAX

This plug-in will allow 3DSMax to load the .pvr file format. When applying a material, this format will be available in the supported list and it will be displayed properly in the view-ports and final rendered images.

To install this plug-in:

Copy: `<SDK_ROOT>\Utilities\PVRTexTool\3DSMAX\<PLATFORM>\
PVRTexTool3DSMAX<VERSION>.dle`

To: `<3DSMAX_DIR>\plug-ins\`

4.3. Autodesk Maya

This plug-in will allow Maya to load the .pvr file format. When applying a material, this format will be available in the supported list and it will be displayed properly in the view-ports and final rendered images.

To install this plug-in:

Copy: `<SDK_ROOT>\Utilities\PVRTexTool\Maya\<PLATFORM>\
PVRTexToolMAYA<VERSION>.dll`

To: `<MAYA_DIR>\bin\plug-ins\image\`

5. Related Materials

Software

- PVRTexLib

Documentation

- PVRTexLib User Manual

White Papers

- PVR Texture Compression

6. Contact Details

For further support contact:

devtech@imgtec.com

PowerVR Developer Technology
Imagination Technologies Ltd.
Home Park Estate
Kings Langley
Herts, WD4 8LZ
United Kingdom

Tel: +44 (0) 1923 260511

Fax: +44 (0) 1923 277463

Alternatively, you can use the PowerVR Insider forums:

www.imgtec.com/forum

For more information about PowerVR or Imagination Technologies Ltd. visit our web pages at:

www.imgtec.com

Appendix A. Texture Format Reference

Although some of the formats below are for use in specific colour spaces, PVRTexTool is not colour space aware and the user must ensure that data in the correct colour space is used with PVRTexTool.

Please note that greyed out formats, while present in the PixelType enum, are not supported by PVRTexTool at this time.

| Format | Description | Command-Line Identifier e.g. -f4444 |
|-------------------------|---|--|
| ARGB 4444 | Good 16-bit format when smooth translucency is needed. | 4444 |
| ARGB 1555 | Punch-through 16-bit translucent format. | 1555 |
| RGB 565 | Best quality 16-bit opaque format. | 565 |
| RGB 555 | As 1555 format but alpha is ignored. Good channel balance. | 555 |
| RGB 888 | 24-bit opaque format with 8 bits for each colour channel. | 888 |
| ARGB 8888 | Best quality 32-bit format, but size and performance are worse than 16-bit formats. | 8888 |
| ARGB 8332 | High quality translucency 16-bit format. | 8332 |
| I 8 | 8-bit intensity only format. | 8 |
| AI 88 | 16-bit alpha and intensity format. | 88 |
| 1BPP | One bit per pixel. | 1_BPP |
| (V,Y1,U,Y0) | YUV 16-bit format. Used for streaming movies. Good for photographic quality textures. | VY1UY0 |
| (Y1,V,Y0,U) | YUV format. | Y1VY0U |
| PVRTC2 | PVRTC compression format. 2-bit per pixel. | PVRTC2 |
| PVRTC4 | PVRTC compression format. 4-bit per pixel. | PVRTC4 |
| OpenGL ARGB 4444 | Good 16-bit format when smooth translucency is needed. | OGL4444 |
| OpenGL ARGB 1555 | Punch-through 16-bit translucent format. | OGL1555 |
| OpenGL ARGB 8888 | Best quality 32-bit format, but size and performance are worse than 16-bit formats. | OGL8888 |
| OpenGL RGB 565 | Best quality 16-bit opaque format. | OGL565 |
| OpenGL RGB 555 | As 1555 format but alpha is ignored. Good channel balance. | OGL555 |
| OpenGL RGB 888 | 24-bit opaque format with 8 bits for each colour channel. | OGL888 |
| OpenGL I 8 | 8-bit intensity only format. | OGL8 |
| OpenGL AI 88 | 16-bit alpha and intensity format. | OGL88 |
| OpenGL PVRTC2 | PVRTC compression format. 2-bit per pixel. | OGLPVRTC2 |
| OpenGL PVRTC4 | PVRTC compression format. 4-bit per pixel. | OGLPVRTC4 |
| OpenGL | An OpenGL ES extension-only format which is essentially a reordered RGBA8888 | OGLBGRA8888 |

| Format | Description | Command-Line Identifier e.g. -f4444 |
|-----------------------|--|--|
| BGRA 8888 | | |
| DXT1 | Microsoft S3TC format, 4 bits per pixel with no alpha. (Windows Only) | DXT1 |
| DXT2 | Microsoft S3TC format, 8 bits per pixel. Good for sharp alpha transitions. Alpha is considered pre-multiplied. (Windows Only) | DXT2 |
| DXT3 | Microsoft S3TC format, 8 bits per pixel. Good for sharp alpha transitions. (Windows Only) | DXT3 |
| DXT4 | Microsoft S3TC format, 8 bits per pixel. Good for gradient alpha transitions. Alpha is considered pre-multiplied. (Windows Only) | DXT4 |
| DXT5 | Microsoft S3TC format, 8 bits per pixel. Good for gradient alpha transitions. (Windows Only) | DXT5 |
| RGB 332 | 8-bit opaque format. | 332 |
| AL 44 | 8-bit alpha & luminance format. | AL44 |
| LVU 655 | Signed integer luminance and bump map format. | LVU655 |
| XLVU 8888 | Signed integer luminance and bump map format. | XLVU8888 |
| QWVU 8888 | Signed 8bit format designed for bump mapping. | QWVU8888 |
| ABGR 2101010 | 10-bit precision format with 2 bits for alpha. | ABGR2101010 |
| ARGB 2101010 | Another 10-bit precision format with 2 bits for alpha. | ARGB2101010 |
| AWVU 2101010 | 10-bit precision signed format with 2 bits for alpha. Designed for bump maps. | AWVU2101010 |
| GR 1616 | 2-channel 16-bit per channel format. | GR1616 |
| VU 1616 | 2-channel 16-bit per channel format. Designed for bump maps. | VU1616 |
| ABGR 16161616 | 64-bit format with alpha. | ABGR16161616 |
| R 16F | Single channel 16-bit floating point format. | R16F |
| GR 1616F | 2-channel 16-bit floating point format. | GR1616F |
| ABGR 16161616F | 64-bit floating point format with transparency. | ABGR16161616F |
| R 32F | Single channel 32-bit floating point format. | R32F |
| GR 3232F | 2-channel 32-bit floating point format. | GR3232F |
| ABGR 32323232F | 128-bit floating point format with transparency. | ABGR32323232F |
| ETC | Ericsson Texture Compression, 4 bits per pixel with no alpha. | ETC |
| | Ericsson Texture Compression, 4 bits per pixel with explicit alpha like DXT3. | |
| | Ericsson Texture Compression, 4 bits per pixel with interpolated alpha like DXT5. | |
| A 8 | 8-bit alpha only format. | A8 |
| VU 88 | 2-channel 8-bit per channel format. Designed for bump maps. | VU88 |
| L16 | 16-bit luminance only format. | L16 |
| L8 | 8-bit luminance only format | L8 |
| AL 88 | 16-bit alpha and luminance format. | AL88 |
| UYVY | YUV colour space, pixel pair format. | UYVY |
| YUY2 | YUV colour space, pixel pair format. | YUY2 |

A.1. DirectX 10 Formats

| Format | Channel Type | Description | Command-Line Identifier |
|----------------------|--|---|---------------------------------------|
| RGBA 32323232 | Float | High precision formats with alpha support | <code>DX10_R32G32B32A32_FLOAT</code> |
| RGBA 32323232 | unsigned int | | <code>DX10_R32G32B32A32_UINT</code> |
| RGBA 32323232 | signed int | | <code>DX10_R32G32B32A32_SINT</code> |
| RGB 323232 | float | High precision formats with no alpha support | <code>DX10_R32G32B32_FLOAT</code> |
| RGB 323232 | unsigned int | | <code>DX10_R32G32B32_UINT</code> |
| RGB 323232 | signed int | | <code>DX10_R32G32B32_SINT</code> |
| RGBA 16161616 | float | 16-bit precision formats with alpha support | <code>DX10_R16G16B16A16_FLOAT</code> |
| RGBA 16161616 | unsigned normalised int | | <code>DX10_R16G16B16A16_UNORM</code> |
| RGBA 16161616 | unsigned int | | <code>DX10_R16G16B16A16_UINT</code> |
| RGBA 16161616 | signed normalised int | | <code>DX10_R16G16B16A16_SNORM</code> |
| RGBA 16161616 | signed int | | <code>DX10_R16G16B16A16_SINT</code> |
| RG 3232 | float | High precision two channel formats | <code>DX10_R32G32_FLOAT</code> |
| RG 3232 | unsigned int | | <code>DX10_R32G32_UINT</code> |
| RG 3232 | signed int | | <code>DX10_R32G32_SINT</code> |
| RGBA 1010102 | unsigned normalised int | 10-bit precision format with 2 bit alpha support. | <code>DX10_R10G10B10A2_UNORM</code> |
| RGBA 1010102 | unsigned int | | <code>DX10_R10G10B10A2_UINT</code> |
| | float | | |
| RGBA 8888 | unsigned normalised int | 32-bit formats with alpha support | <code>DX10_R8G8B8A8_UNORM</code> |
| RGBA 8888 | unsigned normalised int, sRGB colour space | | <code>DX10_R8G8B8A8_UNORM_SRGB</code> |
| RGBA 8888 | unsigned int | | <code>DX10_R8G8B8A8_UINT</code> |
| RGBA 8888 | signed normalised int | | <code>DX10_R8G8B8A8_SNORM</code> |
| RGBA 8888 | signed int | | <code>DX10_R8G8B8A8_SINT</code> |
| RG 1616 | float | 16-bit precision two channel formats | <code>DX10_R16G16_FLOAT</code> |
| RG 1616 | unsigned normalised int | | <code>DX10_R16G16_UNORM</code> |
| RG 1616 | unsigned int | | <code>DX10_R16G16_UINT</code> |
| RG 1616 | signed normalised int | | <code>DX10_R16G16_SNORM</code> |
| RG 1616 | signed int | | <code>DX10_R16G16_SINT</code> |
| R 32 | float | 32-bit single channel formats | <code>DX10_R32_FLOAT</code> |
| R 32 | unsigned int | | <code>DX10_R32_UINT</code> |
| R 32 | signed int | | <code>DX10_R32_SINT</code> |
| RG 88 | unsigned normalised int | 8-bit precision two channel formats | <code>DX10_R8G8_UNORM</code> |
| RG 88 | unsigned int | | <code>DX10_R8G8_UINT</code> |
| RG 88 | signed normalised int | | <code>DX10_R8G8_SNORM</code> |
| RG 88 | signed int | | <code>DX10_R8G8_SINT</code> |
| R 16 | float | 16-bit single channel formats | <code>DX10_R16_FLOAT</code> |
| R 16 | unsigned normalised int | | <code>DX10_R16_UNORM</code> |
| R 16 | unsigned int | | <code>DX10_R16_UINT</code> |
| R 16 | signed normalised int | | <code>DX10_R16_SNORM</code> |
| R 16 | signed int | | <code>DX10_R16_SINT</code> |

| Format | Channel Type | Description | Command-Line Identifier |
|------------------|---|--|-------------------------|
| R 8 | unsigned normalised int | 8-bit single channel formats | DX10_R8_UNORM |
| R 8 | unsigned int | | DX10_R8_UINT |
| R 8 | signed normalised int | | DX10_R8_SNORM |
| R 8 | signed int | | DX10_R8_SINT |
| A 8 | unsigned normalised int | 8-bit single channel alpha format | DX10_A8_UNORM |
| R 1 | unsigned normalised int | 1-bit per pixel texture format | DX10_R1_UNORM |
| RGBE 9995 | | | |
| RGBG 8888 | unsigned normalised int | | |
| GRGB 8888 | unsigned normalised int | | |
| BC 1 | unsigned normalised int | Microsoft S3TC format, 4 bits per pixel with no alpha information. (Windows Only) | DX10_BC1_UNORM |
| BC 1 | unsigned normalised int sRGB colour space | Microsoft S3TC format, 4 bits per pixel with no alpha information. (Windows Only) | DX10_BC1_UNORM_SRGB |
| BC 2 | unsigned normalised int | Microsoft S3TC format, 8 bits per pixel. Good for sharp alpha transitions. (Windows Only) | DX10_BC2_UNORM |
| BC 2 | unsigned normalised int sRGB colour space | Microsoft S3TC format, 8 bits per pixel. Good for sharp alpha transitions. (Windows Only) | DX10_BC2_UNORM_SRGB |
| BC 3 | unsigned normalised int | Microsoft S3TC format, 8 bits per pixel. Good for smooth alpha transitions. (Windows Only) | DX10_BC3_UNORM |
| BC 3 | unsigned normalised int sRGB colour space | Microsoft S3TC format, 8 bits per pixel. Good for smooth alpha transitions. (Windows Only) | DX10_BC3_UNORM_SRGB |
| BC 4 | unsigned normalised int | | |
| BC 4 | signed normalised int | | |
| BC 5 | unsigned normalised int | | |
| BC 5 | signed normalised int | | |

A.2. OpenVG

All OpenVG formats are treated by PVRTexLib as ePREC_INT8.

| Format | Description | Command-Line Identifier |
|---------------------------|--|-------------------------|
| RGBX 8888 sRGB | 32 bits per pixel, no alpha support, sRGB colour space | OVG_RGBX_8888_SRGB |
| RGBA 8888 sRGB | 32 bits per pixel, alpha support, sRGB colour space | OVG_RGBA_8888_SRGB |
| RGBA 8888 sRGB PRE | 32 bits per pixel, pre-multiplied alpha support, sRGB colour space | OVG_RGBA_8888_SRGB_PRE |
| RGB 565 sRGB | 16 bits per pixel, no alpha support, sRGB colour space | OVG_RGB_565_SRGB |
| RGBA 5551 sRGB | 16 bits per pixel, punch-through alpha support, sRGB colour space | OVG_RGBA_5551_SRGB |
| RGBA 4444 sRGB | 16 bits per pixel, alpha support, sRGB colour space | OVG_RGBA_4444_SRGB |
| L 8 sRGB | Single channel 8 bits per pixel format, sRGB colour space | OVG_L_8_SRGB |
| RGBX 8888 IRGB | 32 bits per pixel, no alpha support, IRGB colour space | OVG_RGBX_8888_LRGB |
| RGBA 8888 IRGB | 32 bits per pixel, no alpha support, IRGB colour space | OVG_RGBA_8888_LRGB |
| RGBA 8888 IRGB PRE | 32 bits per pixel, pre-multiplied alpha support, sRGB colour space | OVG_RGBA_8888_LRGB_PRE |
| L 8 IRGB | Single channel 8 bits per pixel format, IRGB colour space | OVG_L_8_LRGB |
| A 8 | Alpha texture 8 bits per channel | OVG_A_8 |
| 1 BPP | Single bit per pixel B&W texture | OVG_1_BPP |
| XRGB 8888 sRGB | 32 bits per pixel, no alpha support, sRGB colour space | OVG_XRGB_8888_SRGB |
| ARGB 8888 sRGB | 32 bits per pixel, alpha support, sRGB colour space | OVG_ARGB_8888_SRGB |
| ARGB 8888 sRGB PRE | 32 bits per pixel, pre-multiplied alpha support, sRGB colour space | OVG_ARGB_8888_SRGB_PRE |
| ARGB 1555 sRGB | 16 bits per pixel, punch-through alpha support, sRGB colour space | OVG_ARGB_1555_SRGB |
| ARGB 4444 sRGB | 16 bits per pixel, alpha support, sRGB colour space | OVG_ARGB_4444_SRGB |
| XRGB 8888 IRGB | 32 bits per pixel, no alpha support, IRGB colour space | OVG_XRGB_8888_LRGB |
| ARGB 8888 IRGB | 32 bits per pixel, alpha support, IRGB colour space | OVG_ARGB_8888_LRGB |
| ARGB 8888 IRGB PRE | 32 bits per pixel, pre-multiplied alpha support, IRGB colour space | OVG_ARGB_8888_LRGB_PRE |
| BGRX 8888 sRGB | 32 bits per pixel, no alpha support, sRGB colour space | OVG_BGRX_8888_SRGB |
| BGRA 8888 sRGB | 32 bits per pixel, alpha support, sRGB colour space | OVG_BGRA_8888_SRGB |
| BGRA 8888 sRGB PRE | 32 bits per pixel, pre-multiplied alpha support, sRGB colour space | OVG_BGRA_8888_SRGB_PRE |
| BGR 565 sRGB | 16 bits per pixel, no alpha support, sRGB colour space | OVG_BGR_565_SRGB |
| BGR 5551 sRGB | 16 bits per pixel, punch-through alpha support, sRGB | OVG_BGR_5551_SRGB |

| Format | Description | Command-Line Identifier |
|---------------------------|--|-------------------------|
| | colour space | |
| BGRA 4444 sRGB | 16 bits per pixel, alpha support, sRGB colour space | OVG_BGRA_4444_SRGB |
| BGRX 8888 IRGB | 32 bits per pixel, no alpha support, IRGB colour space | OVG_BGRX_8888_LRGB |
| BGRA 8888 IRGB | 32 bits per pixel, alpha support, IRGB colour space | OVG_BGRA_8888_LRGB |
| BGRA 8888 IRGB PRE | 32 bits per pixel, pre-multiplied alpha support, IRGB colour space | OVG_BGRA_8888_LRGB_PRE |
| XBGR 8888 sRGB | 32 bits per pixel, no alpha support, sRGB colour space | OVG_XBGR_8888_SRGB |
| ABGR 8888 sRGB | 32 bits per pixel, alpha support, sRGB colour space | OVG_ABGR_8888_SRGB |
| ABGR 8888 sRGB PRE | 32 bits per pixel, pre-multiplied alpha support, sRGB colour space | OVG_ABGR_8888_SRGB_PRE |
| ABGR 1555 sRGB | 16 bits per pixel, no alpha support, sRGB colour space | OVG_ABGR_1555_SRGB |
| ABGR 4444 IRGB | 16 bits per pixel, alpha support, sRGB colour space | OVG_ABGR_4444_SRGB |
| XBGR 8888 IRGB | 32 bits per pixel, no alpha support, IRGB colour space | OVG_XBGR_8888_LRGB |
| ABGR 8888 IRGB | 32 bits per pixel, alpha support, IRGB colour space | OVG_ABGR_8888_LRGB |
| ABGR 8888 IRGB PRE | 32 bits per pixel, pre-multiplied alpha support, IRGB colour space | OVG_ABGR_8888_LRGB_PRE |

