



PVRTexTool

User Manual

Copyright © Imagination Technologies Limited. 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
Version : PowerVR SDK REL_3.3@2855377a External Issue
Issue Date : 26 Mar 2014
Author : Imagination Technologies Limited

Contents

1.	Introduction	5
1.1.	Software Overview.....	5
1.1.1.	PVRTexToolGUI.....	5
1.1.2.	PVRTexTool Command-Line	5
1.1.3.	PVRTexTool Plug-ins	5
1.1.4.	PVRTexLib	5
1.2.	Document Overview	5
2.	PVRTexTool GUI	6
2.1.	Installation.....	6
2.1.1.	From Installer	6
2.2.	Main Interface	6
2.2.1.	View Panels.....	7
2.2.2.	Diff View Panel	7
2.2.3.	Status Bar	7
2.2.4.	Display Options	8
2.2.5.	Texture Information	10
2.2.6.	Surface Browser	10
2.2.7.	The Toolbox	11
2.2.8.	Resize.....	11
2.2.9.	Flip.....	12
2.2.10.	Rotate	12
2.2.11.	Border.....	13
2.2.12.	Generate MIP-Map Chain	13
2.2.13.	Resize Canvas	14
2.2.14.	Generate Normal Map.....	15
2.2.15.	Bleed	15
2.2.16.	Pre-Multiply Alpha	16
2.2.17.	Colour MIP Levels	16
2.2.18.	Channel Swap	16
2.3.	Toolbars.....	18
2.3.1.	Quick Access Bar	18
2.4.	Menus	19
2.4.1.	File Menu.....	19
2.4.2.	Edit Menu	21
2.4.3.	View Menu.....	22
2.4.4.	Window Menu.....	23
2.4.5.	Help Menu	23
2.5.	Dialogs.....	24
2.5.1.	Wrap Raw Data	24
2.5.2.	Create Cube Map	26
2.5.3.	Create Texture Array.....	27
2.5.4.	Create Font Texture	28
2.5.5.	Change Grid Size	28
2.5.6.	Change Background Colour	29
2.5.7.	New Texture	29
3.	PVRTexTool Command-Line	30
3.1.	Installation.....	30
3.1.1.	From Installer	30
3.1.	Usage Instructions.....	30
3.2.	Examples	30
3.3.	Command-Line Options.....	30
4.	PVRTexTool Plug-ins.....	38
4.1.	Adobe Photoshop	38
4.2.	Autodesk 3D Studio MAX	38

4.3.	Autodesk Maya	38
4.4.	Microsoft Windows Explorer	38
5.	PVRTexLib	40
5.1.	Library Overview	40
5.2.	Installation.....	41
5.2.1.	From Installer	41
5.2.2.	Accessing the Library	41
5.2.3.	Using the DLL (Windows Only)	41
5.3.	PVR Container Format	41
5.3.1.	File Header Structure	41
5.3.2.	Meta-Data.....	43
5.4.	Example Code	44
5.4.1.	Read and Decompress an Image	44
5.4.2.	Pre-Process, Transcode (Compress), and Save an Image	44
5.4.3.	Read an Image and Resize the Canvas	45
5.4.4.	Creating an Image from a Header and Data	46
5.4.5.	Accessing Meta-Data	46
6.	Related Materials	47
7.	Contact Details	48

List of Figures

Figure 2-1	PVRTexToolGUI Main Interface	6
Figure 2-2	Diff View Panel	7
Figure 2-3	Status Bar	7
Figure 2-4	Display Options.....	8
Figure 2-5	Texture Information.....	10
Figure 2-6	Surface Browser	10
Figure 2-7	The Toolbox.....	11
Figure 2-8	Flip	12
Figure 2-9	Rotate	12
Figure 2-10	Add Border.....	13
Figure 2-11	Generate MIP-Map Chain.....	13
Figure 2-12	Resize Canvas.....	14
Figure 2-13	Generate MIP Levels	15
Figure 2-14	Bleed.....	15
Figure 2-15	Pre-Multiply Alpha	16
Figure 2-16	Colour MIP Levels	16
Figure 2-17	Channel Swap	17
Figure 2-18	Quick Access Bar	18
Figure 2-19	File Menu	19
Figure 2-20	Edit Menu.....	21
Figure 2-21	View Menu	22
Figure 2-22	Window Menu	23
Figure 2-23	Help Menu	23
Figure 2-24	Wrap Raw Data Dialog	24
Figure 2-25	Create Cube Map Dialog	26
Figure 2-26	Create Texture Array	27
Figure 2-27	Create Font Texture	28
Figure 2-28	Change Grid Size	28

Figure 2-29 Change Background Colour	29
Figure 2-30 New Texture	29

1. Introduction

1.1. Software Overview

PVRTexTool is a suite of utilities for compressing textures, an important technique that ensures the lowest possible texture memory overhead at application run-time. PVRTexTool's components include a library, command line and GUI tools, and a set of plug-ins. Plug-ins are available for Autodesk 3DSMax and Maya, and Adobe Photoshop.

Each component is capable of converting to a variety of popular compressed texture formats such as PVRTC and ETC, as well as all of the core texture formats for a variety of different APIs. They also include a number of advanced features to pre-process the image data; for example border generation, colour bleeding and normal map generation.

Textures can be saved to DDS, KTX, or PVR, Imagination's PowerVR Texture Container format; a container with a full public specification, support for custom meta-data, and complete, optimised, resource loading code in the PVRTools.

1.1.1. PVRTexToolGUI

PVRTexToolGUI is the graphical version of PVRTexTool; it is available for Windows, Linux and Mac OS. It allows the user to manipulate texture data in an interactive graphical environment with immediate visual feedback.

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 calls from a 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).

1.1.4. PVRTexLib

PVRTexLib is a library for the management of textures. It occupies the `'pvrtexture'` namespace and allows users to access the same PVRTexTool functionality in a library, for easy integration with existing tool chains.

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

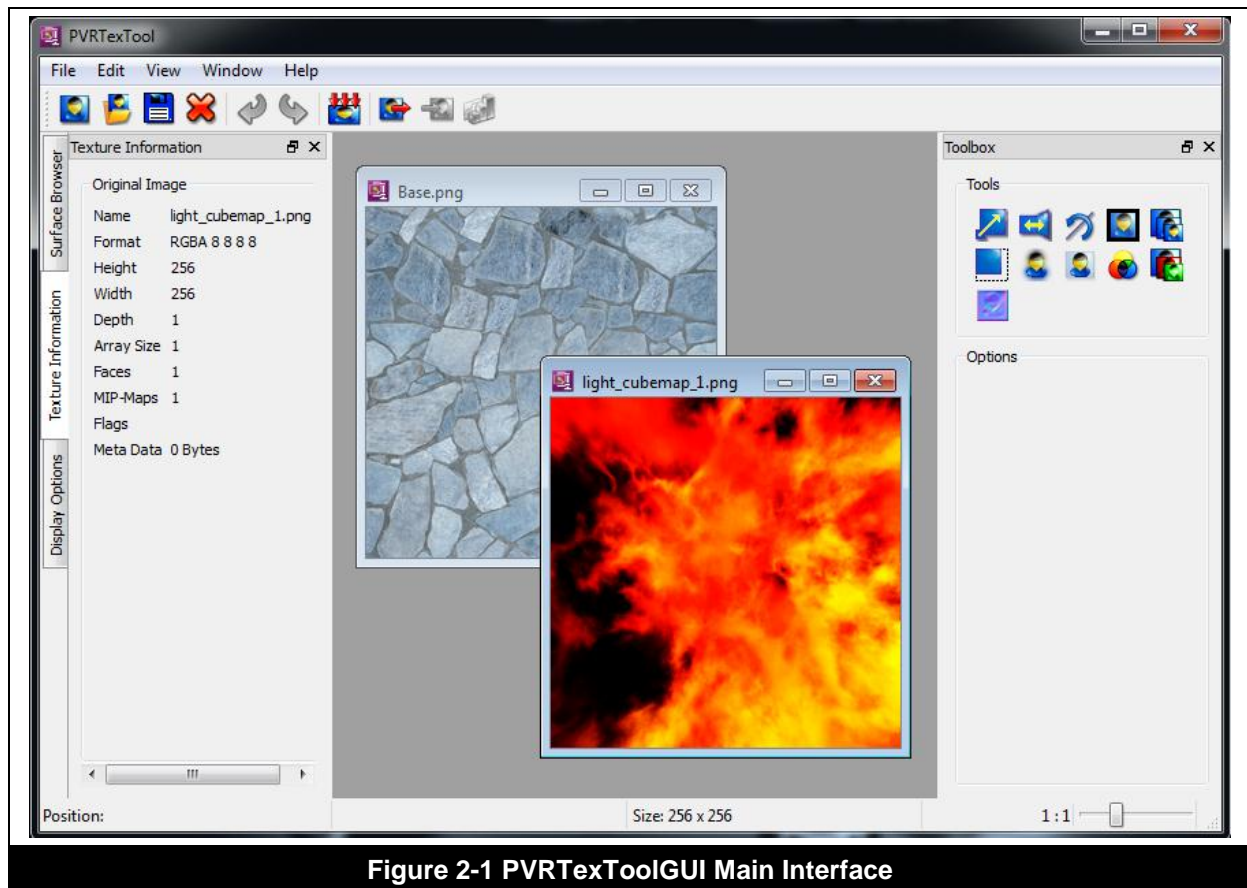
2.1.1. From Installer

Download the PowerVR Insider and follow the on screen instructions. Once the package has successfully installed the application is available in:

```
<InstallDir>\PVRTexTool\GUI\<PLATFORM>\
```

2.2. Main Interface

As an MDI (multiple document interface) application, PVRTexToolGUI may have multiple files open simultaneously displaying a View Panel for each. In these instances any action performed is performed on the View Panel that currently has focus, as shown in Figure 2-1.



2.2.1. View Panels

Each image viewing window renders a single texture to the viewport, allowing users to verify that the texture looks correct both before and after encoding.

If the images have transparency, they are alpha blended with a background. Textures that have been pre-multiplied in the GUI are blended as such. The images can be moved by clicking and dragging if they are not fully visible, and zoomed by scrolling the mouse wheel.

2.2.2. Diff View Panel

The 'Diff View Panel' is similar in function to the default 'View Panel', it can't be edited, nor can a texture be saved from it. Its primary purpose to display the difference between two selected images based on the options set in the 'Display Options'. Images are treated as 'Input' and 'Output' rather than the 'Left' and 'Right' of traditional diffs as the purpose of this view is to highlight differences between input textures and encoded outputs, as portrayed in Figure 2-2.

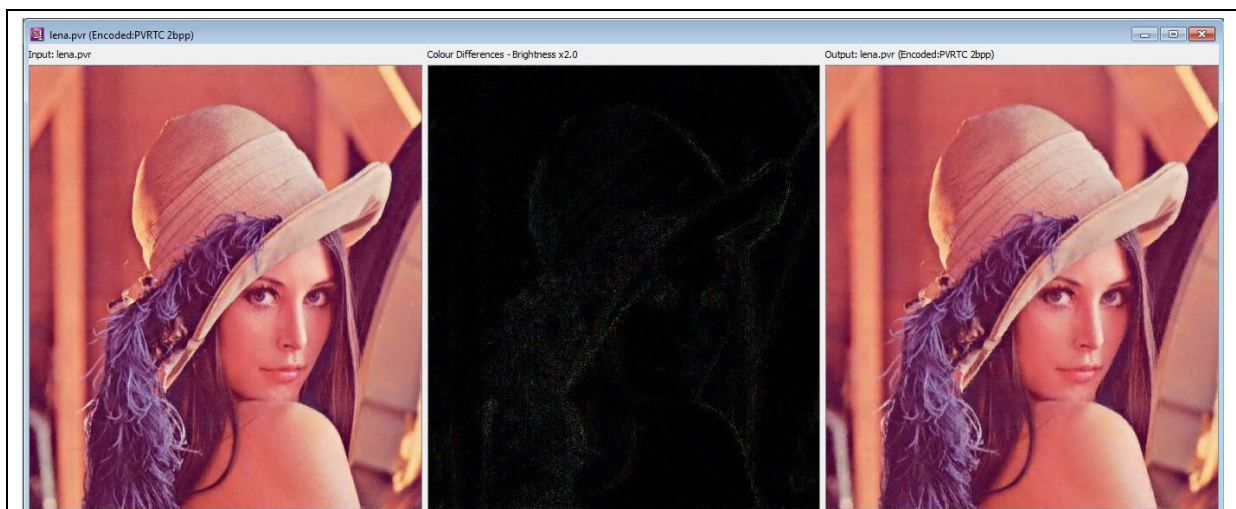


Figure 2-2 Diff View Panel

2.2.3. Status Bar

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

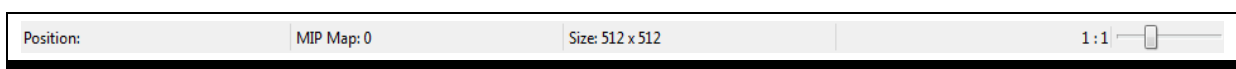


Figure 2-3 Status Bar

Position

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

Size

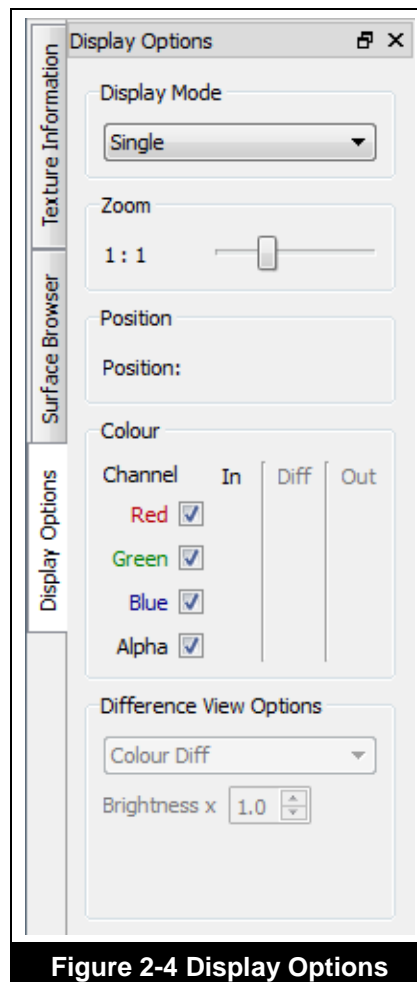
For quick reference the size of the texture being viewed is displayed here.

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 View Panel.

2.2.4. Display Options

The 'Display Options' tab on the left of the screen opens up the display options toolbar, as displayed in Figure 2-4.



Display Mode

This dropdown allows the user to select how the texture in a given view panel is displayed. Three options exist:

- 'Single' – Display a single instance of the texture regardless of the shape or size of the view panel.
- 'Tiled' – Tile the texture across the view panel.
- 'Cube' – Display the texture rendered into a 3D Cube Map, allowing the user to move the camera around as if it was in the centre of a cube with the texture as one of the faces. If multiple cube map faces are specified in the texture these are loaded on to the appropriate faces of the cube.

Zoom

This slider sets the zoom level of the texture.

Position

The position of the cursor within the texture; {0, 0} is represented as the top left corner.

Colour

This table displays the RGBA value of the texture under the cursor. While in a 'Diff View' this displays the information for both textures and display the difference between the two values.

The tick boxes can be used to toggle on or off the displaying of a given colour channel.

Difference View Options

This dropdown allows the user to select the form of diff that is performed in the currently selected 'Diff View'. Three options exist:

- 'Colour Diff' – A standard diff of the colour channels of two images.
- 'Tolerance Diff' – A diff that displays the difference between the original image and the second image as one of three colours. Differences higher than the tolerance are displayed as Red, differences lower than the tolerance are displayed in blue, and any pixels with no difference are displayed as black.
- 'Blend Textures' – This option is not a diff in the classic sense, but is useful for performing side by side comparisons. It performs a blend between the two textures.

Brightness

The 'Brightness' value is used when performing a 'Colour Diff' as a multiplier for the difference values, this assists in highlighting areas where the difference is minimal.

Tolerance

The 'Tolerance' value is the value under which any differences in a 'Tolerance Diff' is ignored.

Blend Visibility

The 'Blend Visibility' sets the amount of each image used when 'Blend Textures' is selected.

2.2.5. Texture Information

The 'Texture Information' panel shows a variety of information in regards the currently loaded texture, such its name, format, dimensions, and number of MIP-Maps etc.

When a 'Diff View' is selected a second box appears on the lower half of the 'Texture Information' panel with the information of the second image.

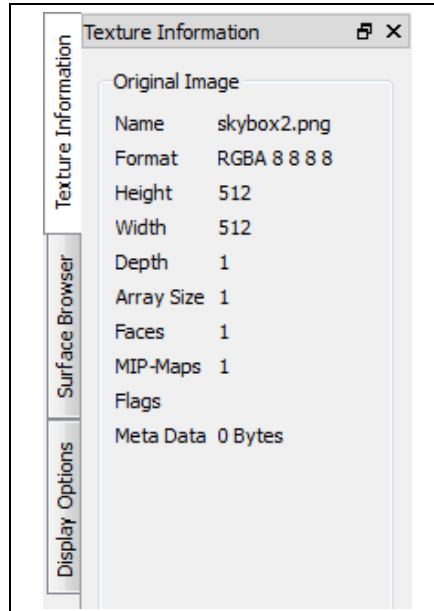


Figure 2-5 Texture Information

2.2.6. Surface Browser

The 'Surface Browser' panel show details of the MIP-Map and Cube map surfaces included in the texture.

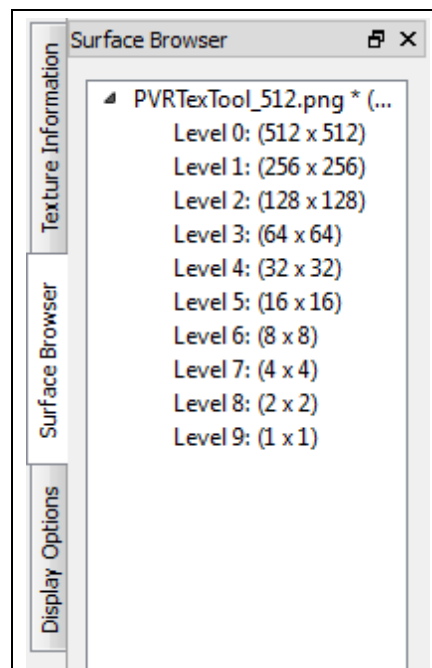


Figure 2-6 Surface Browser

2.2.7. The Toolbox

The Toolbox is PVRTexTool's panel of pre-processing tools.

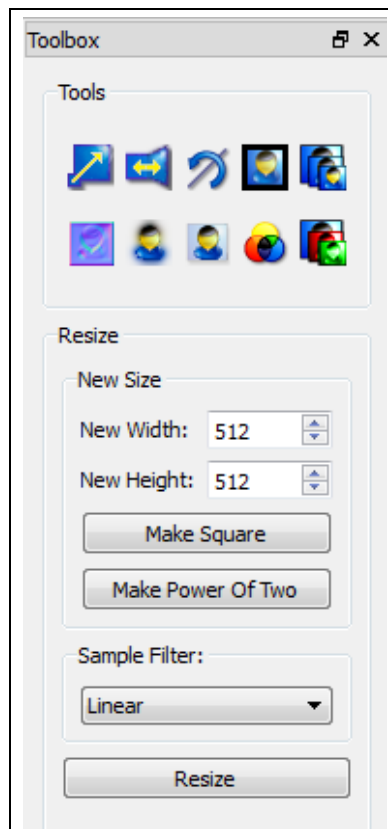


Figure 2-7 The Toolbox

2.2.8. Resize

This offers options for resizing the currently selected texture.

New Size

Sets the desired height and width for the texture to be resized to.

Make Square

Sets the new width and height values to be the same, giving a square texture.

Make Power Of Two

Sets the new width and height values to be the next largest power of two.

Sample Filter

Sets the filter used during resizing. Three options exist:

- Nearest
- Linear
- Cubic

Resize

Resizes the image to the size set under 'New Size' using the filter set under 'Sample Filter'.

2.2.9. Flip

Options for flipping the current texture are displayed in Figure 2-8.

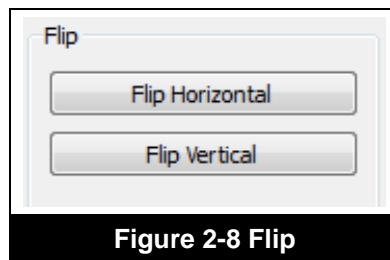


Figure 2-8 Flip

Flip Horizontal

Flips the current texture horizontally.

Flip Vertical

Flips the current texture vertically.

2.2.10. Rotate

Options for rotating the texture are displayed in Figure 2-9.

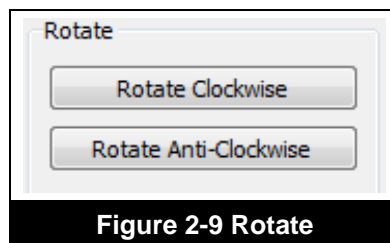


Figure 2-9 Rotate

Rotate Clockwise

Rotates the current texture clockwise.

Rotate Anti-Clockwise

Rotates the current texture anti-clockwise.

2.2.11. Border

This adds a small border to the outside of an image. The purpose of this border is to improve the quality of compressed textures whose compression algorithm assumes the texture wraps. This technique should be used for non-tiling textures, with UV coordinates adjusted to not use the border. Figure 2-10 displays the 'Add Border' dialog.

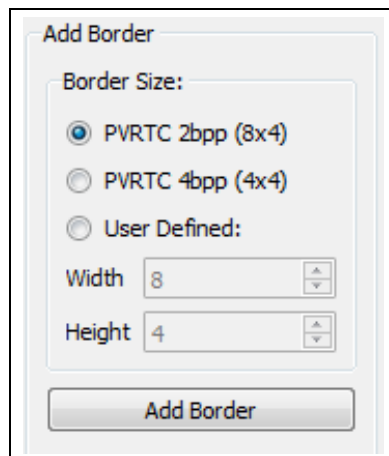


Figure 2-10 Add Border

Border Size

Two border sizes come pre-defined; these are specifically designed to improve the quality of non-wrapped PVRTC 2 & 4 bits per pixel. It also possible for a user to define borders for other texture compression formats.

Add Border

Confirms the settings and applies the border.

2.2.12. Generate MIP-Map Chain

A set of options for generating a MIP-Map chain for your texture, as shown in Figure 2-11.

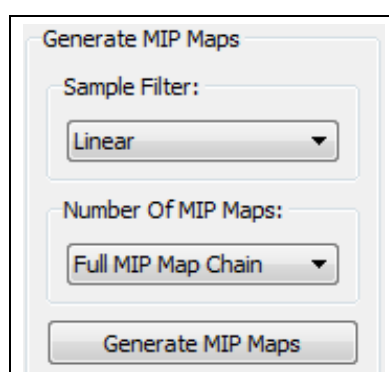


Figure 2-11 Generate MIP-Map Chain

Sample Filter

Sets the filter used during resizing. Three options exist:

- Nearest
- Linear
- Cubic

Number of MIP Maps

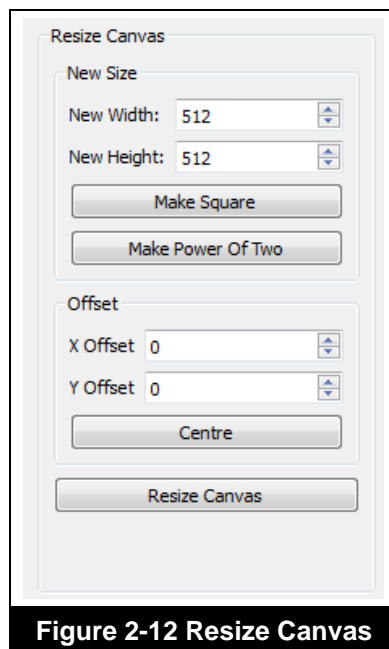
This dropdown sets the number of MIP Maps to be generated; by default the full set of MIP-Maps is generated.

Generate MIP Maps

Confirms the settings and generates MIP Maps

2.2.13. Resize Canvas

This option resizes the canvas without changing the image. Figure 2-12 displays the 'Resize Canvas' dialog.

**New Size**

Sets the new width and new height of the canvas, in pixels.

Make Square

Makes the canvas square by increasing the smallest height/width value to equal the largest height/width value.

Make Power Of Two

Increases the new height/width values to the next power of two. For example; 256 pixels, 512 pixels, 1024 pixels, etc.

Offset

Offsets the current texture from the top left of the new canvas.

Centre

Centres the current texture in the new canvas.

Resize Canvas

Resizes the current canvas using the size and offsets entered above.

2.2.14. Generate Normal Map

This option (illustrated in Figure 2-13) assumes your texture is a height map (based on the intensity of your RGB channels) and generates a normal map based on this information.

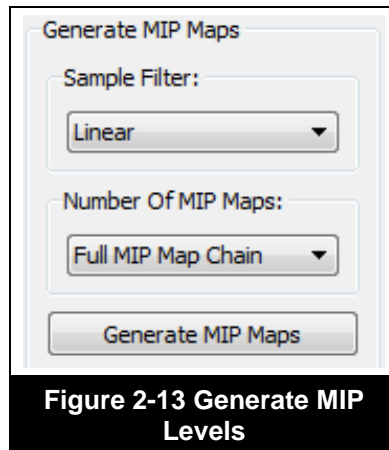


Figure 2-13 Generate MIP Levels

Height Scale

Sets the scaling of the height map from which the normal map is generated.

Channel Order

Sets which channel should map to which axis in the normal map. Optionally, a fourth channel can be used which can contain the original height map.

Generate Normal Map

Generates the normal map based on the above settings.

2.2.15. Bleed

Figure 2-14 displays this option, which improves the quality of opaque areas of an image by bleeding the colour channels into the unused colour channels of transparent areas. This improves both compression quality and any bilinear filtering passes done on the texture. If the textures you use have no useful data in their alpha channel it is strongly recommended that you use this.

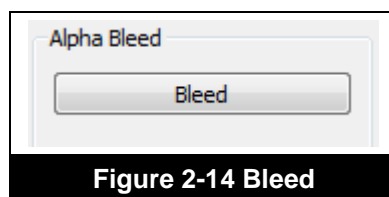


Figure 2-14 Bleed

Bleed

Confirms and applies the bleed.

2.2.16. Pre-Multiply Alpha

This option pre-multiplies the colour channels of a pixel by its alpha channels. This works as a run-time optimisation, as blending each pixel avoids multiplication.

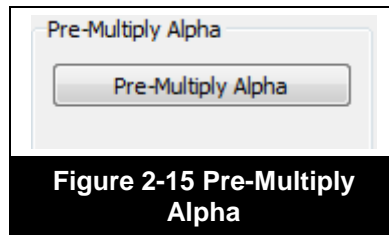


Figure 2-15 Pre-Multiply Alpha

Pre-Multiply Alpha

Confirms and pre-multiplies the current texture.

2.2.17. Colour MIP Levels

This option adds false colours the MIP Map Levels for the currently selected texture to a variety of primary colours. This is mostly useful when testing so that the current MIP Map level can be spotted easily.

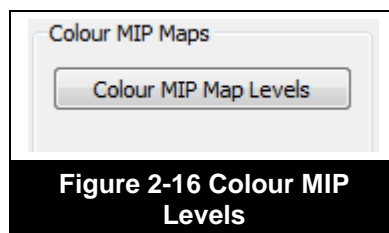


Figure 2-16 Colour MIP Levels

Colour MIP-Map Levels

Confirms and applies the false colouring.

2.2.18. Channel Swap

The channel editor panel allows the user to set the contents of a channel within a texture based on either a channel within the current image, another image, an integer or a float.

For example, setting the current texture's red channel to be the same as the current textures blue channel would mean selecting 'Source: Current', 'Channel: Blue' under Destination: Red. To swap the contents of the red channel in the current texture with the blue channel from another texture, under Destination: Red select 'Source: File', select the desired file, then select 'Channel: Blue'. For Integer and floating point sources, integer values can be in the range [0-255], float values in the range of [0-1].

Channel Editor

Destination: Alpha

Source:

File

Channel:

Alpha

Destination: Red

Source:

Integer

Value: 0

Destination: Green

Source:

Float

Value: 0.00

Destination: Blue

Source:

Current

Channel:

Blue

Swap Channels

Figure 2-17 Channel Swap

2.3. Toolbars

2.3.1. Quick Access Bar

The Quick Access Bar is displayed in Figure 2-18.



New

Icon (a) opens the New Texture dialog.

Open

Icon (b) opens a texture for editing.

Save

Icon (c) saves the texture.

Close

Icon (d) closes the texture.

Undo

Icon (e) undoes the last performed action.

Redo

Icon (f) redoes the last undone action.

Encode

Icon (g) launches the Encode dialog.

Select Diff Input

Icon (h) selects the input texture for a diff.

Diff Against Input

Icon (i) diffs the selected texture against the texture currently selected as the diff input.

Error Metrics

Icon (j) launches the Error Metrics dialog.

2.4. Menus

2.4.1. File Menu

Clicking on 'File' opens up the File Menu.

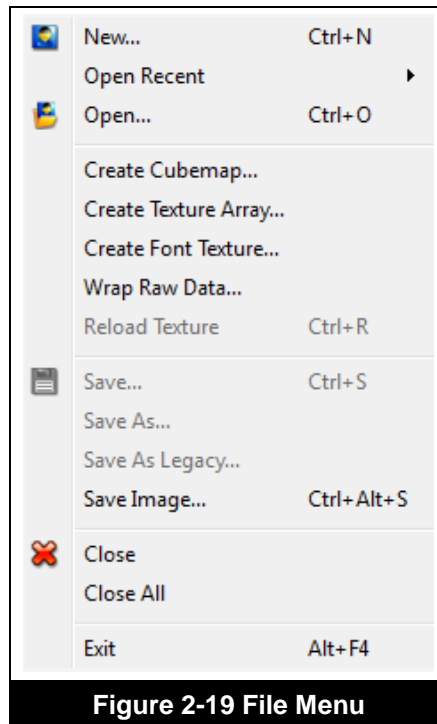


Figure 2-19 File Menu

Open

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

- PVR - PowerVR texture
- KTX - Khronos texture
- DDS - Microsoft Direct Draw Surface

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

Open Recent

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

Create Cube Map...

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

Create Texture Array...

Use this option to create a texture array by combining existing images into a single large texture. The Create Texture Array dialog allows you to specify which images files from which to load and the order in which to load them.

Create Font Texture...

This option launches the 'Create Font Texture' dialog, used to generate texture atlases for use as fonts with Print3D. The first dialogue requests the location of the font file, in either a '.otf' or '.ttf' format, and then for the values for the size of the font and an associated border to be

inputted into the provided fields. This border will put padding between the letters to avoid bilinear bleeding from one letter to another.

This dialogue also permits you to configure how the font will translate into a texture atlas with a toggle for 'Grid Fitting' to force the arrangement into a regular grid; otherwise the tool will try its best fit. The 'Anti Aliasing' toggle will allow you to smooth the edges of the fonts if required. There is also a drop down menu to allow you to specify the character set where you can add your own customised set.

Clicking 'OK' will generate a texture atlas featuring the characters from the '.ttf' or '.otf' file.

The atlas must then be converted into an optimal deployment format, which depends on the framework you wish to use. Before encoding, the 'Vertical Flip' option must be disabled.

The 'Generate MIPMaps' option should only be used if your text will be rendered in 3D.

Wrap Raw Data...

This option opens the 'Wrap Raw Data' dialog; a dialog used to load raw image data from a bitmap or a corrupt texture file. Further information can be found in section 2.6.1.

Reload Texture

'Reload Texture' 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 is 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 saves encoded data to a texture file, and can be saved in one of the following formats:

- PVR - PowerVR Texture Container v3
- KTX - Khronos Texture files
- DDS - Microsoft Direct Draw Surface files
- H - C/C++ Header file storing PVR data in an array of type 'unsigned byte'

If the texture has not yet been encoded and has been modified, PVRTexTool prompts the user to select an encoding method.

Save As Legacy... (Deprecated)

This option is used to save files to the legacy PVR v2 format. PVR v2 is deprecated and support is removed in the future, please update applications to use PVR v3.

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 saves the current MIP-map level if a lower level is being displayed and all other MIP-map data is lost. This can be used to save out each MIP-map level individually if needed. This automatically appends the dimensions of the image to the end of the filename.

Images can be saved as the following formats:

- BMP
- JPG
- PNG

Close

'Close' closes the current texture

Close All

'Close All' closes all the currently open textures.

Exit

'Exit' closes the application

2.4.2. Edit Menu

Clicking 'Edit' will open the Edit Menu.

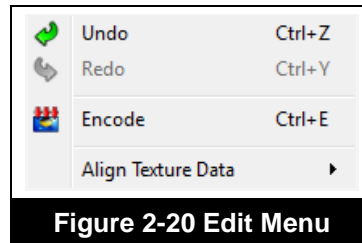


Figure 2-20 Edit Menu

Undo

'Undo' undoes the last performed action.

Redo

'Redo' redoes the last undone action.

Encode Texture...

This option opens the Encode dialog, used to encode textures from one format to another.

Align Texture Data

These options add meta data to pad the file header, aligning the start of the texture data with a given byte boundary.

Align to 2 Byte Boundary

'Align to 2 Byte Boundary' pads the file header with empty meta-data so that the start texture data aligns to a 2 byte boundary.

Align to 4 Byte Boundary

'Align to 4 Byte Boundary' pads the file header with empty meta-data so that the start texture data aligns to a 4 byte boundary.

Align to 8 Byte Boundary

'Align to 8 Byte Boundary' pads the file header with empty meta-data so that the start texture data aligns to a 8 byte boundary.

2.4.3. View Menu

Clicking 'View' opens the View Menu.

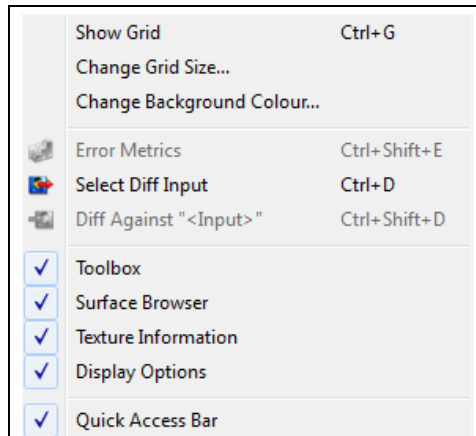


Figure 2-21 View Menu

Show Grid

This option overlays a grid onto the current Texture View Window. The size of this grid is determined via the Change Grid Size dialog.

Change Grid Size...

This option opens the Change Grid Size dialog.

Change Background Colour...

This option opens the Change Background Colour dialog, used to change the background, behind the image from the default grey.

Error Metrics

Opens the Error Metrics Dialog when a diff view window is current.

Select Diff Input

Selects the input texture for a diff.

Diff Against Input

Differs the selected texture against the texture currently selected as the diff input.

Toolbox

Toggles displaying of the Toolbox tab.

Surface Browser

Toggles displaying of the Surface Browser tab.

Texture Information

Toggles displaying of the Texture Information tab.

Display Options

Toggles displaying of the Display Options tab.

Quick Access Bar

Toggles displaying of the Quick Access Bar.

2.4.4. Window Menu

The Window Menu contains options pertaining to the displaying of multiple Texture View Windows.

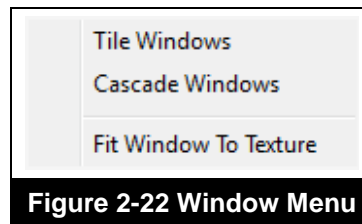


Figure 2-22 Window Menu

Tile Windows

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

Cascade Windows

'Cascade Windows' cascades all the currently open Texture View Windows.

Fit Window to Texture

'Fit Window to Texture' fits the selected view panel to the size of the texture it contains.

2.4.5. Help Menu

Clicking on 'Help' will open the Help Menu.

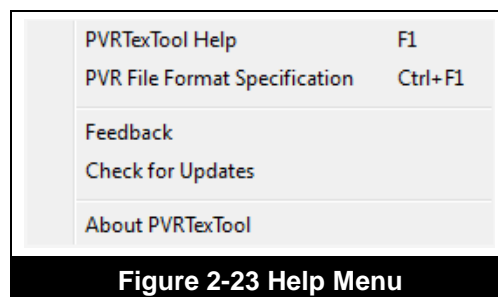


Figure 2-23 Help Menu

PVRTexTool Help

Opens the PVRTexTool User Manual (this document).

PVR File Format Specification

Opens the PVR File Format Specification document.

Feedback

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

Check for Updates

As of SDK release 3.0 PVRTexTool can auto-update. 'Check for Updates' is used to force an update, though PVRTexTool checks for updates automatically each time the program is run.

About PVRTexTool

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

2.5. Dialogs

2.5.1. Wrap Raw Data

The 'Wrap Raw Data' option opens up the Wrap Raw Data Dialog.

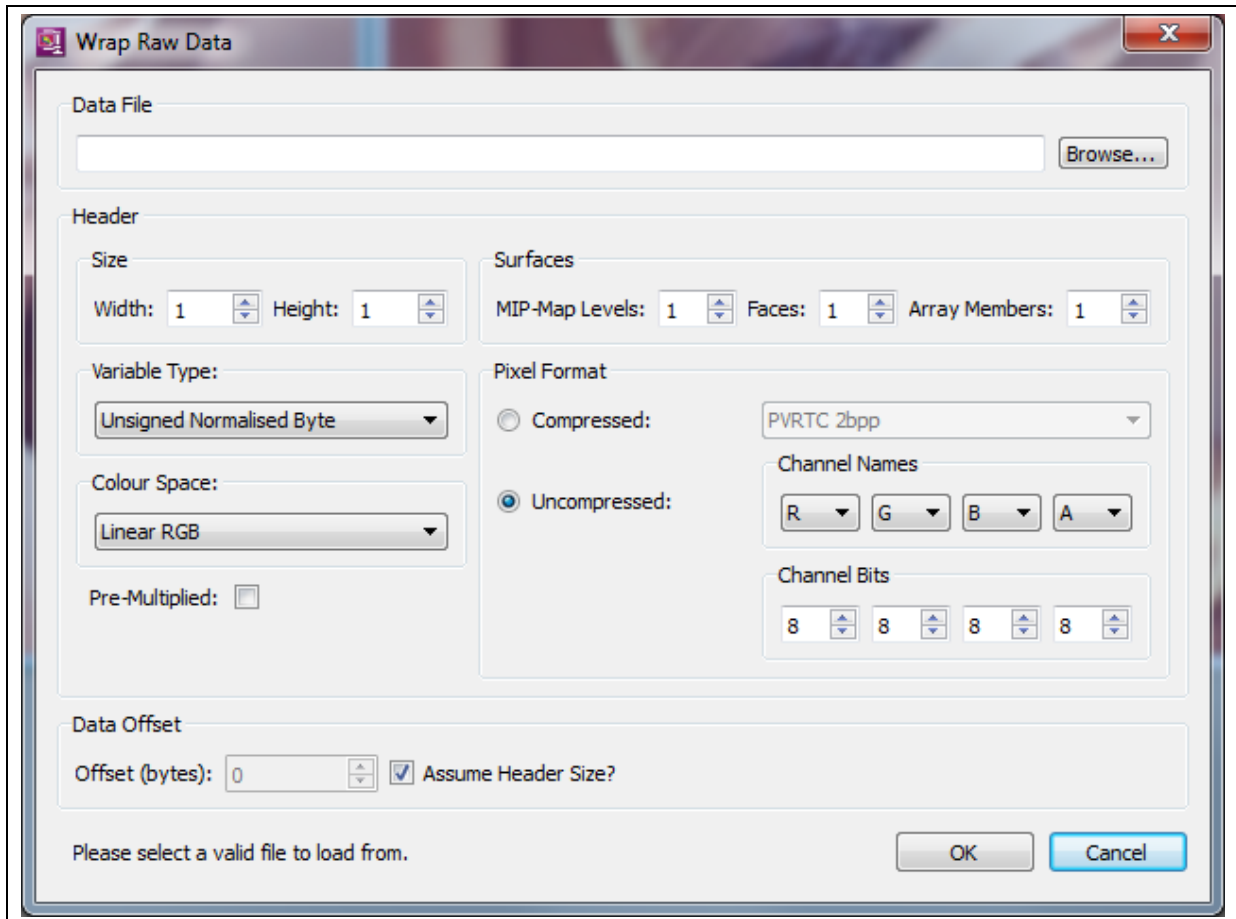


Figure 2-24 Wrap Raw Data Dialog

Data File

'Data File' represents the location of the file from which to load the raw data. Data should be in the order set in the Texture Data section of the PVR File Format Specification, which is as follows:

```

for each MIP-Map Level in MIP-Map Count
  for each Surface in Num. Surfaces
    for each Face in Num. Faces
      for each Slice in Depth
        for each Row in Height
          for each Pixel in Width
            Byte data[Size Based On PixelFormat]
          end
        end
      end
    end
  end
end
end
end
end

```


Header

Size

The width and height of the texture represented by the raw data in the data file.

Surfaces

The number of surfaces of the texture represented by the raw data in the data file, this includes the number of MIP-Maps, the number of faces of a cube map, and the number of members if the data represents a texture array.

Pixel Format

The format of the image, if it is uncompressed data this also states the bit rate and channel order of the image.

Variable Type

The data type of the raw data held within the data file.

Colour Space

This drop down menu allows the user to select the colour space the raw data in the data file is in.

Pre-Multiplied

This toggle is used to indicate whether the data is pre-multiplied by its alpha channel or not.

Data Offset

The data offset is used to set the offset into the raw data within the data file at which the actual texture data starts. This is particularly useful for restoring data with corrupted headers, the size of the header can be entered into the offset, and the header skipped.

Offset

The actual offset in bytes.

Assume Header Size

This toggle states whether a PVR3 header should be assumed to exist at the beginning of the file.

2.5.2. Create Cube Map

The Create Cube Map dialog allows the user to create a cube map from six separate images. The textures are made square, and resized to the same size as the texture in the first position, all with a linear filter. Any additional cube faces, surfaces or MIP Map levels in a texture are discarded.

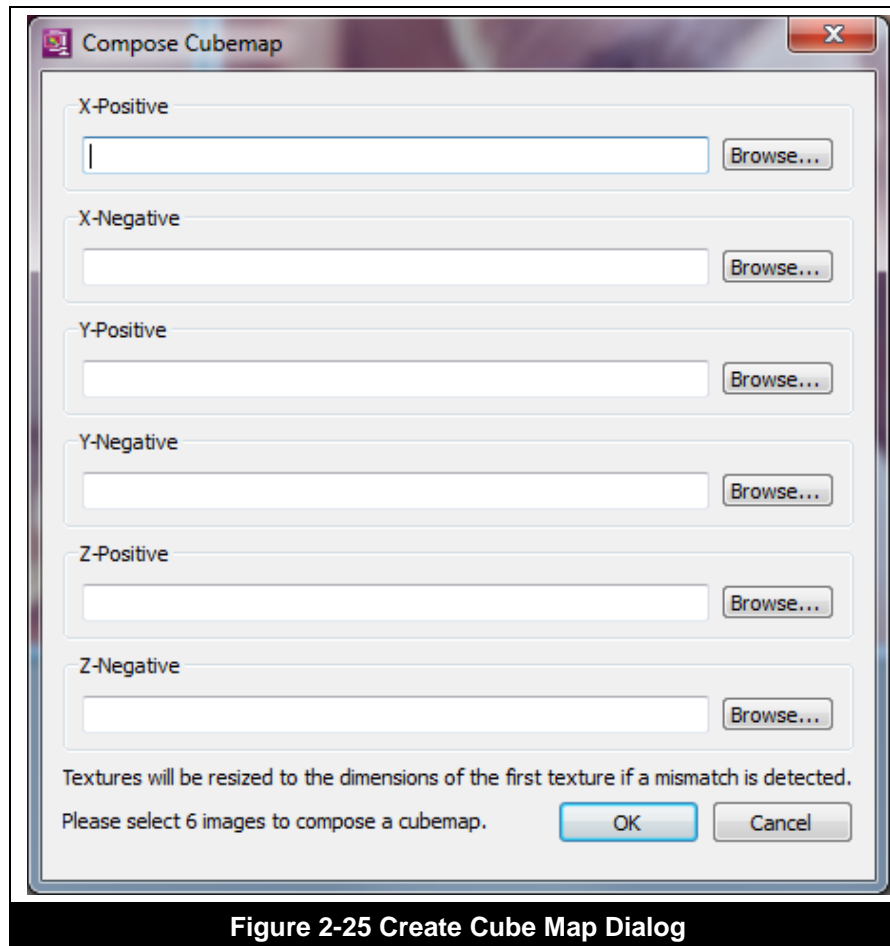


Figure 2-25 Create Cube Map Dialog

2.5.3. Create Texture Array

The Create Texture Array dialog allows the user to create a texture array from any number of textures. The textures are resized to the same size as the texture in the first position with a linear filter. Any additional cube faces, surfaces or MIP Map levels in a texture are discarded.

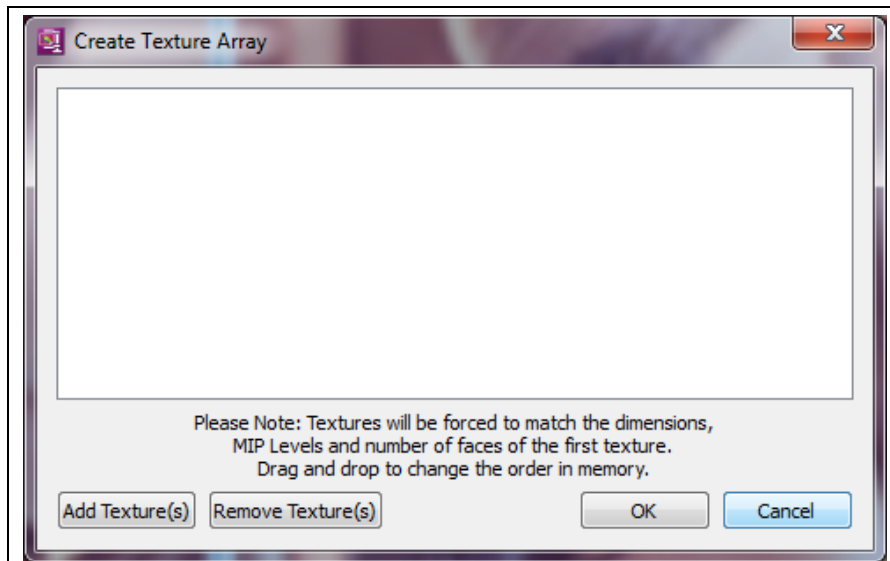


Figure 2-26 Create Texture Array

2.5.4. Create Font Texture

This dialog allows the user to render a font and a set of text to a texture, with supporting meta data about how the characters should be handled. This is used primarily for Print3D but can be used elsewhere. OTF and TTF font files are supported, as are a variety of different character sets.

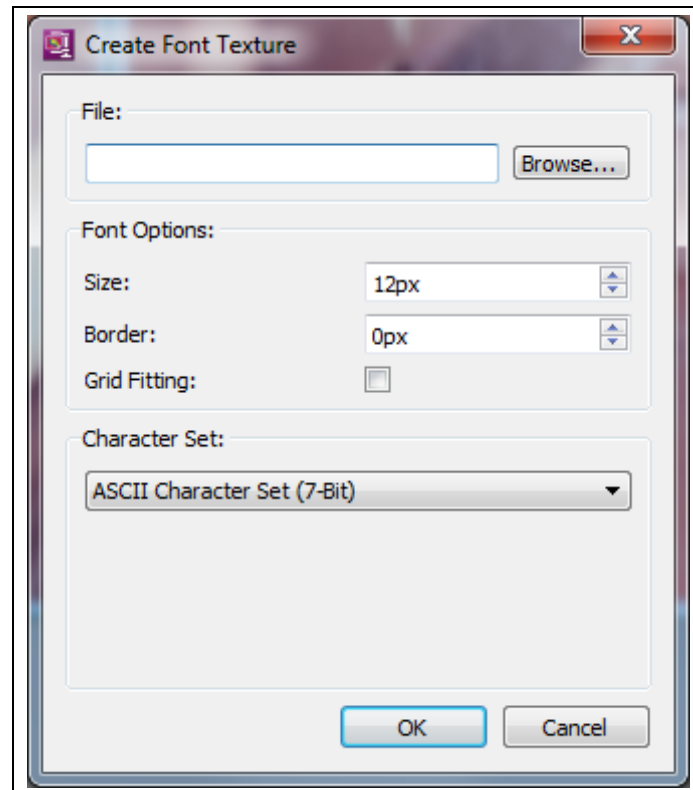


Figure 2-27 Create Font Texture

2.5.5. Change Grid Size

Sets the size of the grid that is displayed when 'Show Grid' is selected.

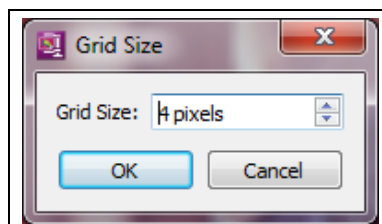


Figure 2-28 Change Grid Size

2.5.6. Change Background Colour

Allows the user to set the background colour of each view panel.

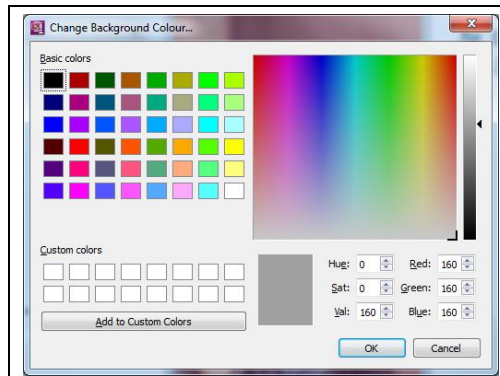


Figure 2-29 Change Background Colour

2.5.7. New Texture

Creates a new texture based on a series of parameters set by the user.

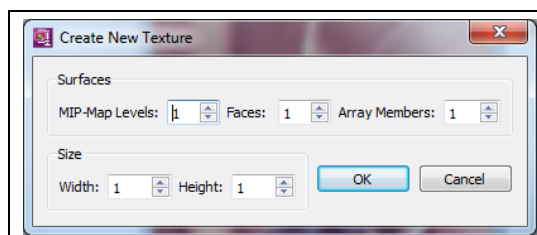


Figure 2-30 New Texture

3. PVRTexTool Command-Line

3.1. Installation

3.1.1. From Installer

Download the PowerVR Insider SDK package and follow the on screen instructions. Once the package has successfully installed the application is available in:

```
<InstallDir>\PVRTexTool\CL\<PLATFORM>\
```

3.1. Usage Instructions

PVRTexToolCL can be used from the command-line to process and compress textures, or by using a batch file. The syntax for the command-line is described below. Note that when compressing a large number of textures to PVRTC format, it is most efficient to let the utility work on each texture in order. If compressing to another format, it is advisable for you to parallelise the threading.

3.2. Examples

Taking an image (Example.bmp), generating a full chain of MIP-Maps for it and encoding it to ARGB 1555:

```
PVRTexToolCL -m -f a1r5g5b5 -i Example.bmp
```

Generating a cube map from files named skybox_n.bmp, encoding to PVRTC 1 4bpp, and saving it as a data structure in a C/C++ header:

```
PVRTexToolCL -i skybox1.bmp,skybox2.bmp,skybox3.bmp,skybox4.bmp,skybox5.bmp,skybox6.bmp -cube  
-m -f pVRTc1_4 -o skybox.h
```

3.3. Command-Line Options

Input File

REQUIRED

Set the input file or files. Must be a JPEG, PNG, BMP, PVR, KTX or DDS file. If either the cube map or texture array flag is set, multiple files should be explicitly specified.

Usage:

```
-i [filepath],<additionalfiles...>
```

Example:

```
-i picture.jpg, otherpicture.png
```

Output File

Set the output file destination. If specified, must be a PVR, KTX or DDS file. Otherwise the application outputs a file with the same name as the first input file.

Usage:

```
-o <filepath>
```

Example:

```
-o texture.pvr
```

Decompress Output

Whether or not to save a decompressed file alongside the input. A filename can be specified as JPEG, PNG or BMP. Otherwise the file is decompressed to a file with the same name as the first input file. For cubemaps and texture arrays, behaviour is special - a series of decompressed files is output in the following format:

```
<Name>.Face<FaceNumber>.Array<ArrayNumber>.<Extension>  
- e.g. Texture.Face1.Array12.png
```

Usage:

-d <filepath>

Example:

-d decompressed.png

Cube Map

Constructs a cube map from available input files. If present, the input file argument must contain at least 6 textures, or a multiple of 6 textures if the array flag is present. Textures of different sizes are resized with a linear filter to the size of the original texture, or the specified size if resizing.

Usage:

-cube <faceorder>

Example:

-cube +X,-X,+Y,-Y,+Z,-Z

Texture Array

Constructs a texture array from available input files. If present, the input file argument should contain multiple files, or a multiple of 6 textures if the cube map flag is enabled.

Usage:

-array

Example:

-array

Pad Meta Data

Adds padding to the meta data, so that the texture data sits on a 2,4 or 8 byte boundary, according to the parameter specified. Must have a parameter in the form of an integer, valid values are 2,4 or 8.

Usage:

-pad [2|4|8]

Example:

-pad 4

Legacy PVR (Deprecated)

Forces the output file to be saved out as a legacy pvr file (PVR v2) for backwards compatibility purposes.

Usage:

-legacypvr

Example:

-legacypvr

Resize

Resizes a texture to the given size. Accepts two unsigned integer parameters - width and height. Values up to 8096x8096 are supported. Option is incompatible with Square or Power of Two resize options.

Usage:

-r [width],[height]

Example:

-r 512,256

Resize Square

Forces the texture into a square. A single character parameter, '-' or '+', can be specified to specify whether it is resized smaller(-) or larger(+). Incompatible with standard resize.

Usage:

-square <+|->

Example:

-square +

Resize Power of Two

Forces the texture into power of two dimensions. A single character parameter, '-' or '+', can be specified to specify whether it is resized smaller(-) or larger(+). Incompatible with standard resize.

Usage:

-pot <+|->

Example:

-pot +

Resize Filter

By default, a linear filter is used to resize textures. Setting this flag to 'nearest', 'linear' or 'cubic' forces PVRTexTool to use the specified filter to resize textures.

Usage:

-rfilter [nearest|linear|cubic]

Example:

-rfilter cubic

Rotate

Rotate the texture around a given axis 'x', 'y' or 'z'. Currently only 'z' is supported - this is a standard 2D rotate. Also requires a second argument '-' or '+' to choose the rotate direction. In the context of a 2D rotation, + is clockwise, - is anti-clockwise.

Usage:

-rotate [z],<+|->

Example:

-rotate z,+

Flip

Flips the texture over a given axis 'x', 'y', or 'z'. Currently only 'x' and 'y' are supported providing a standard 2D flip.

Also accepts an optional second argument, "flag", that adds meta-data to the texture marking it as being flipped. This is useful when knowledge of the orientation is required ahead of time, for example, displaying a flipped texture in its original orientation. Usage:

-flip [x|y],<"flag">

Example:

-flip y,flag

Add Border

Adds a mirrored border to the texture. If no arguments are specified, PVRTexTool chooses an appropriate border size for the texture. Alternative, border sizes can be chosen manually for the width and height. Specifying just the width is allowed, and results in the vertical border having a height of 0.

Usage:

-b <width>,<height>

Example:

-b 4,4

Pre-Multiply Alpha

Pre-Multiplies the texture by its alpha value.

Usage:

-p

Example:

-p

Alpha Bleed

Discards any data in fully transparent areas to optimise the texture for better compression.

Usage:

-l

Example:

-l

Normal Map Generation

Using the input texture as a height map (by creating an intensity texture from the r, g and b channels), generates a normal map. Accepts two arguments: A positive float which determines the scale that the height map is assumed to be on, and a string of a combination of the four characters 'x', 'y', 'z' and 'h'. These specify the channel order as saved out into the texture. x, y and z specify these components, and h specifies the original height value used. Duplicate channels are not allowed, but channels can be missed off. This argument is optional, the default is 'xyz'.

Usage:

-n [scale],<channelorder>

Example:

-n 1.0,xyzh

MIP-Map Generation

Generates MIP-Maps for the current texture. An optional unsigned integer can be added to specify the number of MIP Map levels which should be generated, otherwise a full chain is created.

Usage:

-m <numberofmipmaps>

Example:

-m 9

MIP-Map Filter

By default, a linear filter is used to generate MIP-Maps. Setting this flag to 'nearest', 'linear' or 'cubic' forces it to use the specified filter instead.

Usage:

-mfilter [nearest|linear|cubic]

Example:

-mfilter cubic

Colour MIP-Maps

Saturates the tail of the MIP-Map chain with colours for debugging purposes - if you never see the original colour of a texture, then you know it's using MIP-Maps at all times and some upper levels can be removed from the texture to reduce memory consumption.

Usage:

-c

Example:

-c

Encode Format

REQUIRED

Sets the format to encode to.

First argument (required) is the format, which can be either a compressed format (see Usage for a list) or a non-compressed format in the form 'r8g8b8a8'. Any number of channels up to 4 can be specified, but must be matched with a size (in bits) at all times. Valid channel names are 'r', 'g', 'b', 'a', 'i', 'l' or 'x'. Valid sizes range from 1 to 32, but the total of all sizes must be a multiple of 8 (byte aligned).

The second argument is the channel type. This is optional - defaults to Normalised Unsigned Byte. Specifying a type is generally not required for compressed formats, but generally is for uncompressed formats. See usage table for valid values.

The third and final argument specifies the colour space, which accepts either sRGB or IRGB - default is IRGB for linear rgb.

Usage:

-f [format],<variabletype>,<colourspace>

Example -f PVRTC1_2,UBN,IRGB

Valid Formats:

- PVRTC1_2
- PVRTC1_4
- PVRTC1_2_RGB
- PVRTC1_4_RGB
- PVRTC2_2
- PVRTC2_4
- ETC1
- BC1
- BC2
- BC3
- UYVY
- YUY2
- 1BPP
- RGBE9995
- RGBG8888
- GRGB8888
- ETC2_RGB
- ETC2_RGBA
- ETC2_RGB_A1
- EAC_R11
- EAC_RG11

Valid Variable Types:

- UB
- UBN
- SB
- SBN
- US
- USN
- SS
- SSN
- UI
- UIN
- SI
- SIN
- UF
- SF

Key:

First Char- S=Signed, U=Unsigned.

Second Char- B=Byte, S=Short, I=Integer, F=Float.

Third Char (optional) N=Normalised.

Valid Colour Spaces:

- IRGB
- sRGB

Encode Quality

Sets the quality level to compress with. See usage table for valid options. Only currently useful with ETC and PVRTC formats.

Usag:

-q [compressorquality]

Example:

-q pvrctfast

Dither

Tells the compressor to dither the texture before compression to avoid banding artifacts.

Usage:

-dither

Example:

-dither

Silence

Tells the utility to not output messages of any kind.

Usage:

-shh

Example:

-shh

Help

Requests help either a list of commands, or for help on a specified argument, if the argument is a parameter.

Usage:

-help <commandargument>

Example:

-help "flip"

Red Channel

Sets the Red channel in the input texture to match the channel specified in a second image. A filename is specified for the source, and an optional channel name (single character) can be specified to select the source. By default, the channel draws from its equivalent in the new texture. E.g. Red draws from red, green from green etc. Valid source channels are 'r','g','b','a','l','i'. These represent red, green, blue, alpha, luminance and intensity.

Usage

-red [filename],<channelname>

Example

-red Red.png,g

Green Channel

Sets the Green channel in the input texture to match the channel specified in a second image. A filename is specified for the source, and an optional channel name (single character) can be specified to select the source. By default, the channel draws from its equivalent in the new texture. E.g. Red draws from red, green from green etc. Valid source channels are 'r','g','b','a','l','i'. These represent red, green, blue, alpha, luminance and intensity.

Usage

```
-green [filename],<channelname>
```

Example

```
-green Green.dds,b
```

Blue Channel

Sets the Blue channel in the input texture to match the channel specified in a second image. A filename is specified for the source, and an optional channel name (single character) can be specified to select the source. By default, the channel draws from its equivalent in the new texture. E.g. Red draws from red, green from green etc. Valid source channels are 'r','g','b','a','l','i'. These represent red, green, blue, alpha, luminance and intensity.

Usage

```
-blue [filename],<channelname>
```

Example

```
-blue Blue.pvr,r
```

Alpha Channel

Sets the Alpha channel in the input texture to match the channel specified in a second image. A filename is specified for the source, and an optional channel name (single character) can be specified to select the source. By default, the channel draws from its equivalent in the new texture. E.g. Red draws from red, green from green etc. Valid source channels are 'r','g','b','a','l','i'. These represent red, green, blue, alpha, luminance and intensity.

Usage

```
-alpha [filename],<channelname>
```

Example

```
-alpha Alpha.bmp,i
```

Difference

Calculates the difference between the input and a supplied file, providing error metrics, and is incompatible with performing compression (-f). A visual representation of the differences can be output by selecting an output mode:

- 'Colour' Outputs the absolute delta of each channel into a texture. The modifier multiplies the deltas to highlight any issues (Default: 1.0).
- 'Tolerance' diffs using the modifier as threshold (Default: 0.1): Deltas of 0 are black, above the threshold are red, below are blue.
- 'Blend' blends the images using the modifier as a weighting of the first texture against the second. (Default: 0.5).
- 'None' (Default) will suppress any output, so that it only provides metrics.

Usage

```
-diff [filename],<mode>,<modifier>
```

Example

```
-diff Other.png,Tolerance,0.5f
```

Resize Canvas

Resizes a texture to the given size, without changing the image data. This takes effect after resizing. Accepts two unsigned integer parameters - width and height. Values up to 8192x8192 are supported. Option is incompatible with Square or Power of Two canvas resize options.

Usage

```
-rcanvas [width],[height]
```

Example

```
-rcanvas 512,256
```

Resize Canvas Square

Forces the texture into a square, without changing the image data. This takes effect after resizing. A single character parameter, '-' or '+', can be specified to specify whether it is resized smaller(-) or larger(+). Incompatible with standard resize.

Usage

```
-squarecanvas <+|->
```

Example

```
-squarecanvas +
```

Resize Canvas Power of Two

Forces the texture into power of two dimensions, without changing the image data. This takes effect after resizing. A single character parameter, '-' or '+', can be specified to specify whether it is resized smaller(-) or larger(+). Incompatible with standard resize.

Usage

```
-potcanvas <+|->
```

Example

```
-potcanvas +
```

Offset Canvas

Sets the offset when performing a canvas resize (including square or pot resizes). Accepts two signed integer parameters - xoffset and yoffset. Values from -8192x-8192 to 8192x8192 are supported. Incompatible with canvas centring.

Usage

```
-offsetcanvas [xoffset],[yoffset]
```

Example

```
-offsetcanvas -12,56
```

Centre Canvas

Sets the offset when performing a canvas resize (including square or pot resizes) so that the image resides in the centre of the canvas. Incompatible with canvas offset.

Usage

```
-centrecanvas
```

Example

```
-centrecanvas
```

4. PVRTexTool Plug-ins

Plug-ins are available on Microsoft Windows for Adobe Photoshop, Autodesk 3D Studio Max, Autodesk Maya and Microsoft Windows Explorer.

32bit versions are available for all four plug-ins; 64bit versions are available for the Windows Explorer plug-in, and the Maya and 3DSMax plug-ins from 2010 versions onwards.

4.1. Adobe Photoshop

This plug-in allows Photoshop to save and load PVR files, with the option to encode when saving.

Installation

```
Copy <SDK_ROOT>\Utilities\PVRTexTool\Photoshop\<PLATFORM>\PVRFormat.8bi  
To <PHOTOSHOP_DIR>\plug-ins\File Formats\
```

Uninstallation

```
Delete <PHOTOSHOP_DIR>\plug-ins\File Formats\PVRFormat.8bi
```

4.2. Autodesk 3D Studio MAX

This plug-in allows 3DSMax to save and load PVR files. PVR becomes available as a material type similar to bitmaps, and users is able to save final rendered images in this format.

Installation

```
Copy <SDK_ROOT>\Utilities\PVRTexTool\3DSMax\<PLATFORM>\  
PVRTexTool3DSMax<VERSION>.dle  
To <3DSMAX_DIR>\plug-ins\
```

Uninstallation

```
Delete <3DSMAX_DIR>\plug-ins\PVRTexTool3DSMax<VERSION>.dle
```

4.3. Autodesk Maya

This plug-in allows 3DSMax to save and load PVR files. PVR becomes available as a material type similar to bitmaps, and users is able to save final rendered images in this format.

Installation

```
Copy <SDK_ROOT>\Utilities\PVRTexTool\3DSMax\<PLATFORM>\  
PVRTexToolMaya<VERSION>.dll  
To <MAYA_DIR>\bin\plug-ins\image\
```

Uninstallation

```
Delete <3DSMAX_DIR>\plug-ins\PVRTexToolMaya<VERSION>.dll
```

4.4. Microsoft Windows Explorer

This plug-in allows you to preview PVR files in Windows Explorer. The icon for PVR files are replaced with a thumbnail, and the preview pane displays the texture rendered to the pane. To install the Windows Explorer Plug-ins, one of run the following commands:

Installation

```
Command Line (Administrator Access): Regsvr32.exe <SDK_ROOT>  
\PVRTexTool\WindowsExplorer_x86_32\PVRTextureViewer.dll
```

Or

```
Command Line (Administrator Access): Regsvr32.exe <SDK_ROOT>  
\PVRTexTool\WindowsExplorer_x86_64\PVRTextureViewer.dll
```

Uninstallation

Command Line (Administrator Access): Regsvr32.exe /u <SDK_ROOT>
\PVRTexTool\WindowsExplorer_x86_32\PVRTextureViewer.dll

Or

Command Line (Administrator Access): Regsvr32.exe <SDK_ROOT>
\PVRTexTool\WindowsExplorer_x86_64\PVRTextureViewer.dll

5. PVRTexLib

5.1. Library Overview

PVRTexLib is a library for the management of PVR textures. It occupies the `'pvrtexture'` namespace and provides the facility to:

- Load and save PVR files.
- Transcode to and from many different texture formats.
- Perform a variety of pre-process techniques on decompressed pixel data.
- Provide information about a texture file loaded by the library.

The following static library files are provided:

- `'PVRTexLib.lib'` – Windows library
- `'libPVRTexLib.a'` – Linux object archive
- `'libPVRTexLib.a'` – Mac OS object archive

The following dynamic library files are provided:

- `'PVRTexLib.dll'` – Windows dynamic link library
- `'libPVRTexLib.so'` – Linux shared object library
- `'libPVRTexLib.dylib'` – Mac OS dynamic library

Also present are a number of header files that define PVRTexLib's functionality:

- `'PVRTexLibVersion.h'`
- `'PVRTexture.h'`
- `'PVRTextureHeader.h'`
- `'PVRTextureUtilities.h'`
- `'PVRTextureDefines.h'`
- `'PVRTextureFormat.h'`

Finally, a number of header files from the PowerVR SDK Tools libraries are present:

- `'PVRTGlobal.h'`
- `'PVRTErrors.h'`
- `'PVRTArray.h'`
- `'PVRTMap.h'`
- `'PVRTString.h'`
- `'PVRTTexture.h'`

These header files are included in the PVRTexLib package so that separate installation of the tools libraries is not required. Reference material on the PowerVR SDK Tools libraries can be found in the `'Tools'` folder in the PowerVR Insider SDK directory.

`'PVRTextureUtilities.h'` is the primary header file, and including it in your project includes all other header files required for PVRTexLib to function.

Note: Any texture passed to a pre-processing function must be in one of PVRTexLib's standard formats, which is R8G8B8A8 unsigned normalised byte, R16G16B16A16 unsigned normalised short, R32G32B32A32 unsigned normalised integer or signed float. This excludes the Transcode function.

5.2. Installation

5.2.1. From Installer

Download one of the PowerVR Insider SDKs and run the installer following the on screen instructions. Once the package has successfully installed, the library files is available in the SDK folder at:

```
<InstallDir>\PVRTexTool\Library\
```

5.2.2. Accessing the Library

To access the functionality of the library within an application the library must be linked against and the header files included in the application at build time.

Finally, if the resulting program is linked against the shared libraries then these libraries must be present on the target system.

5.2.3. Using the DLL (Windows Only)

When developing for Windows using the PVRTexLib libraries the pre-processor define

'_WINDLL_IMPORT' must be set.

5.3. PVR Container Format

The PVR file format is composed of a header, followed by any amount of meta-data (both of which combine to form the 'CPVRTextureHeader' class); which is then followed by texture data. The header contains 11x 32-bit unsigned integers, and 1x 64-bit unsigned integer (52 bytes total), and is designed to include all the information required to load a given texture.

5.3.1. File Header Structure

```
struct PVRTextureHeaderV3 {
    PVRTuint32    u32Version;
    PVRTuint32    u32Flags;
    PVRTuint64    u64PixelFormat;
    PVRTuint32    u32ColourSpace;
    PVRTuint32    u32ChannelType;
    PVRTuint32    u32Height;
    PVRTuint32    u32Width;
    PVRTuint32    u32Depth;
    PVRTuint32    u32NumSurfaces; //For texture arrays
    PVRTuint32    u32NumFaces;   //For cube maps
    PVRTuint32    u32MIPMapCount;
    PVRTuint32    u32MetaDataSize;
};
```

u32Version

'u32Version' contains the version information for the header file. This is checked using the following information from 'PVRTextureDefines.h':

```
const PVRTuint32 PVRTEX3_IDENT      = 0x03525650; // 'P''V''R'3
const PVRTuint32 PVRTEX3_IDENT_REV  = 0x50565203;
const PVRTuint32 PVRTEX_CURR_IDENT  = PVRTEX3_IDENT;
const PVRTuint32 PVRTEX_CURR_IDENT_REV = PVRTEX3_IDENT_REV;
```

It is good practice to test against 'PVRTEX_CURR_IDENT' as this is used in future revisions of the library and always refers to the most current version.

u32Flags

The purpose of the 'u32Flags' field is to allow for future proofing of the header format, giving the format the ability to specify flags that can dictate how the texture data is stored.

The following flag is currently supported:

```
const PVRTuint32 PVRTEX3_PREMULTIPLIED = (1<<1);
```

- 'PVRTEX3_PREMULTIPLIED' – When this flag is set, the colour values within the texture data have been pre-multiplied by the alpha channel.

u64PixelFormat

'u64PixelFormat' is a 64-bit unsigned integer containing the pixel format of the texture data contained in the file. This field is best read using the union, PixelType:

```
union PixelType
{
    struct LowHigh
    {
        uint32 Low;
        uint32 High;
    } Part;
    uint64 PixelTypeID;
    uint8 PixelTypeChar[8];
};
```

This allows the 'u64PixelFormat' to perform a double purpose; when 'High' is '0' 'PixelTypeID' can be used to read an enum of compressed pixel types*:

```
enum EPVRTPixelFormat
{
    ePVRTCI_2bpp_RGB,
    ePVRTCI_2bpp_RGBA,
    ePVRTCI_4bpp_RGB,
    ePVRTCI_4bpp_RGBA,
    ePVRTCII_2bpp,
    ePVRTCII_4bpp,
    ...
};
```

Or 'PixelTypeChar' can be used to read up to 4 characters representing the channel order, for example 'r', 'g', 'b', 'a', and up to 4 unsigned 8-bit integers representing the channel bit rate, for example '8, 8, 8, 8' or '10, 10, 10, 2'.

*PVRTexLib may not support all of the pixel types within the 'EPVRTPixelFormat' enum for every API, check 'PVRTTexture.h' for more information.

u32ColourSpace

'u32ColourSpace' represents the colour space the texture data is in. The value to test against is taken from the enum 'EPVRTColourSpace':

```
enum EPVRTColourSpace
{
    ePVRTCSpacelRGB,
    ePVRTCSpacesRGB
};
```

u32ChannelType

'u32ChannelType' is used to determine what data type the colour channels within the format use:

```
enum EPVRTVariableType
{
    ...
    ePVRTVarTypeUnsignedByte,
    ePVRTVarTypeSignedByte,
    ePVRTVarTypeUnsignedShortNorm,
    ...
};
```

u32Height/Width/Depth

These three entries in the header each represent the length of a dimension of the texture.

Contrary to some image libraries, Depth in this instance refers to the number of z-slices in a 3D texture.

u32NumSurfaces

'u32NumSurfaces' is used for texture arrays; it represents the number of textures in the texture array.

u32NumFaces

'u32NumFaces' records the number of faces in a cube map.

u32MIPMapCount

'u32MIPMapCount' is a 32-bit unsigned integer representing the number of MIP-map levels present including the top level. A value of 1 means that only a top level non-MIP-mapped texture exists, 2 means the top level plus an extra MIP-map level, etc.

u32MetaDataSize

Version 3 of the PVR header format includes support for arbitrary meta-data. This flag represents the combined size of all meta-data in the file.

5.3.2. Meta-Data

Within a PVR file, the header is immediately followed by an amount of meta-data as specified in u32MetaDataSize. This meta-data is split into a series of blocks containing 3x 32-bit unsigned integers and 1x 8-bit unsigned integer pointer. The format of each meta-data block is as follows:

```
struct MetaDataBlock
{
    uint32 DevFOURCC;
    uint32 u32Key;
    uint32 u32DataSize;
    uint8* Data;
};
```

DevFOURCC

'DevFOURCC' is a four character descriptor representing the creator of the meta-data. This value, coupled with 'u32Key' is used to correctly read the value of 'Data'. The values 'P' 'V' 'R' '0' to 'P' 'V' 'R' '255' are reserved for use by Imagination Technologies and should not be used.

u32Key

'u32Key' contains a value that indicates the type of data contained in the 'Data' field. This value, coupled with 'DevFOURCC' is used to correctly read the value of 'Data'. This is done in two separate fields so that the value of 'u32Key' may be reused without causing collisions or unknown behaviour.

u32DataSize

'u32DataSize' records the size of 'Data' so that the correct amount of memory can be accessed.

Data

The 'Data' field is a pointer to the head of an array that can contain any data of the user's choice. The loader must be able to read this data based on the values of 'u32Key' and 'DevFOURCC'.

Further information can be found on all of the above in the header files 'PVRTextureDefines.h' and 'PVRTextureHeader.h'.

5.4. Example Code

5.4.1. Read and Decompress an Image

In this example, an existing PVR file is read and decompressed, possibly for later processing, access to the image data or re-encoding.

```
#include "PVRTexture.h"

using namespace pvrtexture;

CPVRTString filePath = "example.pvr";

// Open and read a pvr texture from the file location specified by filePath
CPVRTexture cTexture(filePath);

// Decompress cTexture to the standard RGBA8888 format.
Transcode(    cTexture,
              PVRStandard8PixelFormat,
              ePVRTVarTypeUnsignedByteNorm,
              ePVRTCSpacelRGB
            );

// cTexture should now be in the format RGBA8888, with each channel being of the type
// unsigned integer, in the lRGB colour space.
```

5.4.2. Pre-Process, Transcode (Compress), and Save an Image

In this example, an image in an uncompressed format is converted into a normal map of the same dimensions, and a full MIP-map chain is generated; it is then encoded into PVRTC 1, 4 bits per pixel and saved to an output file.

It should be noted that the standard texture formats in OpenGL ES and OpenGL are treated as normalised; only when specified are they read as integer values. As such, the 'Variable Type' passed to Transcode(...) should be of the type 'Norm' when targeting these APIs.

Note: Any texture passed to a pre-processing function must be in one of PVRTexLib's standard formats, which is R8G8B8A8 unsigned normalised byte, R16G16B16A16 unsigned normalised short, R32G32B32A32 unsigned normalised integer or signed float. This excludes the Transcode function.

```
#include "PVRTexture.h"

using namespace pvrtexture;

CPVRTString filePath = "example.pvr";

// Open and reads a pvr texture from the file location specified by filePath
CPVRTexture cTexture(filePath);

// Convert the image to a Normal Map with a scale of 5.0, and y/z/x channel order
GenerateNormalMap(cTexture, 5.0, "yzx");

// Generate MIP-map chain
GenerateMIPMaps(cTexture, eResizeLinear);

// Compress to PVRTC 4bpp.
Transcode(cTexture, ePVRTPF_PVRTCI_4bpp_RGBA, ePVRTVarTypeUnsignedByteNorm, ePVRTCSpacelRGB);

// Save the file
cTexture.saveFile("out.pvr");
```

5.4.3. Read an Image and Resize the Canvas

In this example, an existing PVR file is read in an uncompressed format, the canvas is resized to 512x256, leaving the (original) canvas in the top left of the texture. A full MIP-map chain is then generated and the texture is saved to a PVR file.

```
#include "PVRTexture.h"

using namespace pvrtexture;

CPVRTString filePath = "example.pvr";

// Open and reads a pvr texture from the file location specified by filePath
CPVRTexture cTexture(filePath);

// Resize canvas
ResizeCanvas(cTexture, 512, 256, 1, 0, 0, 0);

// Generate MIP-map chain
GenerateMIPMaps(cTexture, eResizeLinear);

// Save the file
cTexture.saveFile("out.pvr");
```

5.4.4. Creating an Image from a Header and Data

In this example, pixel data in a compressed format is added to a header to create a 'CPVRTexture' which is then saved to a file.

```
#include "PVRTexture.h"

using namespace pvrtexture;

// The pixel data is a pointer called 'pData'.
// Create the header.
CPVRTextureHeader cHeader(    ePVRTPF PVRTCI 4bpp RGB,
                              512,
                              512
                              );

// Create the image.
CPVRTexture cTexture(cHeader, pData);

// Save the file
cTexture.saveFile("out.pvr");
```

5.4.5. Accessing Meta-Data

In this example, meta-data is read from the header of a texture and interpreted based on values of 'DevFOURCC' and 'u32Key'.

```
#include "PVRTexture.h"

using namespace pvrtexture;

CPVRTString filePath = "test.pvr";

// Open and reads a pvr texture from the file location specified by filePath
CPVRTexture cTexture(filePath);

// Get a reference to the header
const CPVRTextureHeader& rHeader = cTexture.getHeader();

// As the developer, we choose our own values for DevFOURCC and u32Key
// As such, we know what they are, in this case 'aCC' and 'aKey' respectively.
if(rHeader.hasMetaData(aCC, aKey))
{
    // Handle the block based on 'aCC' and 'aKey'
    DoSomething();
}
```

6. Related Materials

Specifications

- [PVR File Format Specification](#)

White Papers

- [PVR Texture Compression](#)
- [PVRTC & Texture Compression User Guide](#)

7. Contact Details

For further support, visit our forum:

<http://forum.imgtec.com>

Or file a ticket in our support system:

<https://pvrsupport.imgtec.com>

To learn more about our PowerVR Graphics SDK and Insider programme, please visit:

<http://www.powervrinsider.com>

For general enquiries, please visit our website:

<http://imgtec.com/corporate/contactus.asp>

Imagination Technologies, the Imagination Technologies logo, AMA, Codescape, Enigma, IMGworks, I2P, PowerVR, PURE, PURE Digital, MeOS, Meta, MBX, MTX, PDP, SGX, UCC, USSE, VXD and VXE are trademarks or registered trademarks of Imagination Technologies Limited. All other logos, products, trademarks and registered trademarks are the property of their respective owners.