Contents

1. Introduction .................................................................................................................................................. 4
   1.1. Document Overview ......................................................................................................................... 4
   1.2. Software Overview ............................................................................................................................. 4
      1.2.1. Texture Formats ......................................................................................................................... 4

2. PVRTexTool GUI ......................................................................................................................................... 6
   2.1. The Basics ........................................................................................................................................ 6
      2.1.1. Compatibility ............................................................................................................................. 6
      2.1.2. User Interface Layout ................................................................................................................ 6
   2.2. Menu Bar ........................................................................................................................................... 7
      2.2.1. File Menu .................................................................................................................................... 7
      2.2.2. Edit Menu .................................................................................................................................... 8
      2.2.3. View Menu ................................................................................................................................... 9
      2.2.4. Window Menu .............................................................................................................................. 11
      2.2.5. Help Menu .................................................................................................................................. 11
   2.3. Texture Details, Display and Comparison ......................................................................................... 12
      2.3.1. Surface Browser .......................................................................................................................... 12
      2.3.2. Texture Information ................................................................................................................... 13
      2.3.3. Display Options ........................................................................................................................... 13
      2.3.4. Texture Comparison .................................................................................................................... 15
   2.4. Texture Pre-Processing ...................................................................................................................... 17
      2.4.1. General Steps .............................................................................................................................. 17
      2.4.2. Options and Settings .................................................................................................................. 18
   2.5. Texture Creation and Loading .......................................................................................................... 21
      2.5.1. Create a New Texture .................................................................................................................. 21
      2.5.2. Create a Cube Map ..................................................................................................................... 21
      2.5.3. Create a Texture Array .............................................................................................................. 22
      2.5.4. Create a Font Texture ................................................................................................................ 23
      2.5.5. Wrap Raw Data ........................................................................................................................... 24
   2.6. Texture Encoding ............................................................................................................................... 25
      2.6.1. Encoding Modes .......................................................................................................................... 26

3. PVRTexTool CLI ........................................................................................................................................... 28
   3.1. Usage Instructions ............................................................................................................................... 28
   3.2. Examples ............................................................................................................................................ 28
   3.3. Command-Line Options .................................................................................................................... 28

4. PVRTexTool Plugins ................................................................................................................................... 36
   4.1. Plugin for Adobe Photoshop ............................................................................................................. 36
      4.1.1. Installation .................................................................................................................................... 36
      4.1.2. Uninstallation ............................................................................................................................... 36
   4.2. Plugin for Autodesk 3ds Max ............................................................................................................ 37
      4.2.1. Installation .................................................................................................................................... 37
      4.2.2. Uninstallation ............................................................................................................................... 37
   4.3. Plugin for Autodesk Maya ................................................................................................................. 37
      4.3.1. Installation .................................................................................................................................... 37
      4.3.2. Uninstallation ............................................................................................................................... 38
   4.4. Plugin for Windows Explorer ........................................................................................................... 38
      4.4.1. Installation .................................................................................................................................... 38
      4.4.2. Uninstallation ............................................................................................................................... 38

5. PVRTexTool Library ..................................................................................................................................... 39
   5.1. Library Overview ............................................................................................................................... 39
      5.1.1. Accessing the Library .................................................................................................................. 39
      5.1.2. Using the DLL (Windows only) .................................................................................................... 40
   5.2. Example Code ....................................................................................................................................... 40
      5.2.1. Read and Decompress an Image .................................................................................................. 40
      5.2.2. Pre-Process, Transcode (Compress) and Save an Image ............................................................. 40
List of Figures
Figure 1. General layout of the GUI ................................................................. 6
Figure 2. File menu ...................................................................................... 7
Figure 3. Edit menu .................................................................................... 9
Figure 4. View menu .................................................................................. 10
Figure 5. Window menu ............................................................................ 11
Figure 6. Help menu .................................................................................. 11
Figure 7. Surface Browser window ............................................................ 12
Figure 8. Texture Information window ...................................................... 13
Figure 9. Display Options window .............................................................. 14
Figure 10. Colour diff results .................................................................. 15
Figure 11. Tolerance diff results ................................................................. 16
Figure 12. Performing a diff by blending textures ........................................ 16
Figure 13. Error Metrics dialog box ......................................................... 17
Figure 14. Toolbox window ...................................................................... 17
Figure 15. Create New Texture dialog box ................................................ 21
Figure 16. Compose Cubemap dialog box .................................................. 22
Figure 17. Create Texture Array dialog box .............................................. 22
Figure 18. Create Font Texture dialog box ............................................... 23
Figure 19. Wrap Raw Data dialog box ...................................................... 25
Figure 20. Encode Texture dialog box ...................................................... 26

List of Tables
Table 1. Settings for resizing a texture ....................................................... 18
Table 2. Settings for flipping a texture along an axis ................................. 18
Table 3. Settings for rotating a texture by 90 degrees ............................... 18
Table 4. Settings for adding a mirrored border to a texture ...................... 19
Table 5. Settings for generating a MIP map chain for a texture .............. 19
Table 6. Settings for resizing a texture canvas ........................................ 19
Table 7. Settings for editing individual channels ..................................... 20
Table 8. Settings for creating a normal map .............................................. 21
Table 9. Settings for creating a font texture ............................................. 23
Table 10. Settings used for wrapping raw data ......................................... 25
Table 11. Command-line options .............................................................. 28
Table 12. Provision for plugins ................................................................. 36
1. Introduction

1.1. Document Overview
The purpose of this document is to serve as a complete user manual for PVRTexTool. It includes installation instructions, functionality explanations and useful tips on how to make the most out of the applications. The document also provides a list of all command-line options and covers information about PVRTexTool plugins and library.

1.2. Software Overview
PVRTexTool is a utility for compressing textures, an important technique that ensures the lowest possible texture memory overhead at application run-time. The components of PVRTexTool include a library, Command Line Interface (CLI) and Graphical User Interface (GUI) tools, and a set of plugins. Plugins are available for Autodesk 3ds Max 2010 and later editions, Autodesk Maya 2010 and later editions, as well as Adobe Photoshop.

Each component is capable of converting to a variety of popular compressed texture formats such as PowerVR Texture Compression (PVRTC) and Ericsson Texture Compression (ETC), as well as all of the core texture formats for a range of different APIs. PVR, the container format used by PVRTexTool, has full public specification, support for custom metadata, as well as complete, optimised, resourced loading code in the cross-platform tools library supplied in the PowerVR Graphics SDK.

PVRTexTool also includes a number of advanced features to pre-process the image data, e.g., border generation, colour bleeding and normal map generation, amongst others. An overview of the various components of PVRTexTool is provided next:

- **PVRTexTool GUI**: This is the Graphical User Interface of PVRTexTool. It is available for Windows, Linux and OS X, and allows the user to manipulate texture data in an interactive graphical environment with immediate visual feedback.

- **PVRTexTool CLI**: This is the Command Line Interface of PVRTexTool. It is available for Windows, Linux and OS X, where 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.

- **PVRTexTool Plugins**: These are designed to give various programs access to the functionality of PVRTexTool. For example, with the plugins, Adobe Photoshop has the ability to load and save PowerVR (PVR) texture files. In the case of Autodesk 3ds Max and Autodesk Maya, the plugins allow the use of PVR texture files when applying materials and also enable saving rendered images in PVR format (at 32 bits per pixel only).

- **PVRTexTool Library**: This 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.

1.2.1. Texture Formats
PVRTexTool offers a variety of texture compression formats depending on the user requirements, detailed below:

- **PVRTC**: PVRTC is PowerVR’s proprietary texture compression format. It uses a sophisticated amplitude modulation scheme to encode textures. Texture data is encoded as two low-resolution images along with a full resolution, low-bit-precision modulation signal.

- **ETC**: Ericsson Texture Compression is a lossy texture compression scheme developed by Ericsson Research. ETC1 lacks Alpha component support, which was rectified with the release of the ETC2 scheme.

- **ASTC**: Adaptive Scalable Texture Compression is a lossy block-based texture compression format developed by ARM. It has been an official extension of OpenGL and OpenGL ES since 2012. The block sizes that are available range from 4x4 to 12x12 pixels for 2D textures, and 3x3x3 to 6x6x6 pixels for 3D textures, with each block being 128 bits. Therefore, the larger the block size, the smaller the size and lower the quality of the texture.
Compressing to ASTC formats

If you wish to compress to ASTC formats, you will require the ASTC Evaluation Codec (astcenc) to be in your path when working with PVRTexTool. This codec is freely available from ARM, and can be added to your path by following the listed instructions:

**On Windows**
1. Navigate to your System Settings, and select the “Advanced” option,
2. Select “Environment Variables”,
3. Select the “PATH” option and click on “Edit”,
4. Type in the directory of the ASTC encoder.
5. Click “OK”.

**On Mac OS X**
1. Open up Terminal.
2. Run `sudo nano /etc/paths`,
3. When prompted, enter your password,
4. Go to the bottom of the file and input the directory of the ASTC encoder,
5. Quit the file,
6. Type “Y” to save the modification.

**On Linux**
1. Modifying the path on Linux depends on your shell.
2. If using Bash, Sh, or Ksh, at the shell prompt type:

   ```
   export PATH=$PATH:/path/to/dir1
   export PATH=$PATH:/path/to/dir1:/path/to/dir2
   ```

   Or:

   ```
   PATH=$PATH:/path/to/dir1; export PATH
   ```

3. To have the path persist across terminal sessions, add the above commands to a startup script such as `.bashrc`.

**Note:** In all examples, `path/to/dir1` should be replaced with the directory containing the ASTC encoder.
2. PVRTexTool GUI

2.1. The Basics

2.1.1. Compatibility
For the most efficient performance, it is recommended that PVRTexTool is installed on machines running at least Windows 7, Mac OS 10.8 (Mountain Lion), or Linux (Ubuntu 12.04).

2.1.2. User Interface Layout
The general layout of the GUI is illustrated in Figure 1. It is a multi-document interface that allows working with various files simultaneously.

The interface is split into the following sections:

- **Menu bar** (Figure 1a): This section enables access to several options for file, edit, view, window and help.
- **View window area** (Figure 1b): This section displays all the view windows for files that have been opened in the GUI.
- **Texture detail area** (Figure 1c): This section contains multiple switchable windows which mainly display texture details, amongst others.
- **Status bar** (Figure 1d): This section details basic information about the status of a texture.
- **Quick access bar** (Figure 1e): This section provides options for performing the most common actions while working with textures.
- **Toolbox window** (Figure 1f): This section provides options and settings which are useful during the pre-processing of textures.
2.2. Menu Bar

2.2.1. File Menu

Figure 2 illustrates the File menu. The menu provides options for creating and manipulating textures, opening, saving and closing files, as well as exiting the GUI.

![Figure 2. File menu](image)

**Create a New Texture**

To create a new texture, click **File -> New...** (Figure 2). This option is also available from the Quick access bar. This will open a dialog box for specifying the settings of the new texture (see Section 2.5.1).

**Open a Recent File**

To open a recently accessed file, click **File -> Open Recent File** (Figure 2). This will display a list of the recently accessed files from which the desired file can then be selected for opening. The list can contain up to 10 recently opened files for quick access.

**Clear Recent Files**

To clear the list of recently accessed files, click **File -> Open Recent File** (Figure 2). An option called **Clear recent files** will become available to use.

**Open a File**

To open a texture or image file, click **File -> Open...** (Figure 2). This option is also available from the Quick access bar. This will open a dialog box for browsing to the required file. Keyboard shortcuts can also be used to open files. PVRTexTool supports three texture formats: PowerVR texture (PVR); Khronos texture (KTX); and Microsoft Direct Draw Surface (DDS). The tool is also able to read the following image formats: BMP, JPEG and PNG.

**Create a Cube Map**

To create a cube map texture, click **File -> Create Cubemap...** (Figure 2). This will open a dialog box which allows the specification of image files for the six faces of the cube (see Section 2.5.2).

**Create a Texture Array**

To create a texture array, click **File -> Create Texture Array...** (Figure 2). This will open a dialog box allowing the specification and ordering of image files to produce the texture array (see Section 2.5.3).
Create a Font Texture

To create a font texture, click File -> Create Font Texture... (Figure 2). This will open a dialog box to guide the user through the process of generating texture atlases for use as fonts with PVRUIRenderer (see Section 2.5.4).

Wrap Raw Data

To load raw image data from a bitmap or corrupt texture file, click File -> Wrap Raw Data... (Figure 2). This will open a dialog box for specifying the required settings (see Section 2.5.5).

Reload a Texture

To reload a texture, click File -> Reload Texture (Figure 2). Using this option 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, as appropriate.

Save a Texture

Encoded data can be saved to a texture file by using the options File -> Save or File -> Save As... (Figure 2). This option is also available from the Quick access bar. Saving can be done in the following formats: PVR, KTX and DDS. If the texture has not yet been encoded and has been modified, the GUI prompts the user to select an encoding method.

Save a Texture in a Legacy Format

To save a texture in the legacy PVR v2 format, click File -> Save As Legacy... (Figure 2).

Warning: PVR v2 is deprecated and with limited support. Please update applications to use PVR v3.

Save as an Image File

To save a texture as an image file, click File -> Save Image... (Figure 2). This option allows the user to save the currently displayed texture 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 will be lost. This can be used to save each MIP map level individually if needed. It also automatically appends the dimensions of the image to the end of the filename. Images can be saved in the following formats: BMP, JPEG and PNG.

Close a Currently Opened Texture

To close a currently opened texture, click File -> Close (Figure 2). This option is also available from the Quick access bar.

Close all Opened Textures

To close all currently opened textures, click File -> Close All (Figure 2).

Exit the GUI

To close the GUI, click File -> Exit (Figure 2).

2.2.2. Edit Menu

Figure 3 illustrates the Edit menu which provides options for undoing and redoing actions, encoding textures and padding file headers with metadata.
Figure 3. Edit menu

**Undo an Action**
To undo the last performed action, click Edit -> Undo (Figure 3). This option is also available from the Quick access bar.

**Redo an Action**
To redo the last undone action, click Edit -> Redo (Figure 3). This option is also available from the Quick access bar.

**Encode a Texture**
To encode a texture from one format into another, click Edit -> Encode (Figure 3). This option is also available from the Quick access bar. This will then open a dialog box providing various encoding options (see Section 2.6).

**Align Texture Data**
To pad a file header by aligning the start of the texture data with a given byte boundary, click Edit -> Align Texture Data (Figure 3) and select the desired option from the displayed list:

- **Align to 2 Byte Boundary**: This option pads the file header with empty data so that the start texture data aligns to a 2 byte boundary.
- **Align to 4 Byte Boundary**: This option pads the file header with empty data so that the start texture data aligns to a 4 byte boundary.
- **Align to 8 Byte Boundary**: This option pads the file header with empty data so that the start texture data aligns to an 8 byte boundary.

**2.2.3. View Menu**
Figure 4 illustrates the View menu which provides options for performing texture comparisons and customizing the layout of the GUI, amongst others.

Note: The GUI layout can also be customized by dragging and dropping the individual dockable windows to either the right or left hand sides of the interface or as standalone windows.
Show Gridlines in a View Window
To show gridlines on a currently opened file, click View -> Show Grid (Figure 4). The size of the gridlines can be set by using the View -> Change Grid Size... option.

Change Grid Size in a View Window
To change the size of the gridlines, click View -> Change Grid Size... (Figure 4). This will open a dialog box from which the grid size (in pixels) can be specified.

Change Background Colour of a View Window
To change the background colour for a particular view window, click View -> Change Background Colour... (Figure 4). This will open a colour picker from which the required colour can be selected.

View Error Metrics after Texture Comparison
To view error statistics after a texture comparison activity, click View -> Error Metrics (Figure 4). This will open a dialog box for viewing the participating metrics (see Section “Display Error Metrics for an Encoded Texture”).

Select Diff Input for Texture Comparison
To select an input texture for texture comparison, click View -> Select Diff Input (Figure 4). After the input has been selected it is then possible to choose another texture for comparing against (see Section 2.3.4).

Select a Texture against which to Diff
To select a texture against which the input texture is to be compared, click View -> Diff Against <Input> (Figure 4). A window will then open to aid the diff analysis process (see Section 2.3.4).

Show or Hide Toolbox Window
To show or hide the Toolbox window (see Section 2.4), toggle View -> Toolbox (Figure 4).

Show or Hide Surface Browser Window
To show or hide the Surface Browser window (see Section 2.3.1), toggle View -> Surface Browser (Figure 4).

Show or Hide Texture Information Window
To show or hide the Texture Information window (see Section 2.3.2), toggle View -> Texture Information (Figure 4).
Show or Hide Display Options Window
To show or hide the Display Options window (see Section 2.3.3), toggle View -> Display Options (Figure 4).

Show or Hide Quick Access Bar
To show or hide the Quick access bar (see Figure 1e), toggle View -> Quick Access Bar (Figure 4).

2.2.4. Window Menu
Figure 5 illustrates the Window menu, which allows controlling the display of view windows in the View window area (see Figure 1b).

![Window menu](image)

**Figure 5. Window menu**

- **Tile Windows**
  To arrange the view windows opened in the View window area in a tile mode, click Window -> Tile Windows (Figure 5).

- **Cascade Windows**
  To arrange the view windows opened in the View window area in a cascading mode, click Window -> Cascade Windows (Figure 5).

- **Fit Window to Texture**
  To fit a selected view window to the full size of the image it contains, click Window -> Fit Window to Texture (Figure 5).

2.2.5. Help Menu
Figure 6 illustrates the Help menu, which provides options for accessing help assets, sending feedback, checking for software updates and viewing general release information.

![Help menu](image)

**Figure 6. Help menu**

- **View PVRTexTool User Manual**
  To view the “PVRTexTool User Manual” document, click Help -> PVRTexTool Help (Figure 6).

- **View PVR File Format Specification Document**
  To view the “PVR File Format Specification” document, click Help -> PVR File Format Specification (Figure 6).
Submit Feedback
To provide feedback, click Help -> Feedback... (Figure 6). This will open a dialog box where instructions are displayed on how to post feedback and request for support.

Check for Updates
To force-check for software updates, click Help -> Check for Updates (Figure 6). This will check for the latest version of PVRTexTool available in our servers.

About PVRTexTool
To view basic information about PVRTexTool release information such as versioning and contact details, click Help -> About PVRTexTool (Figure 6).

2.3. Texture Details, Display and Comparison
When a texture is opened in the GUI, the Texture detail area displays texture and surface browser information, texture display options as well as texture comparison options (see Figure 8, Figure 7 and Figure 9).

2.3.1. Surface Browser
The Surface Browser window shows details of the MIP map and cube map surfaces within a texture as well as the names of other files that are opened in the GUI (Figure 7). To view the individual levels, such as surfaces within a texture, expand the texture name in the window and select the desired level. This will then show the surface in the view window of the selected texture.

Load File to a Surface
The Surface Browser window can be used to load a file to a surface. This can be achieved by following these steps:

1. Right-click the name of the texture (or one of its levels, faces or members in the case of a texture with surfaces).
2. Click the Load file to surface... option. This will open a dialog box for browsing to the required file.
3. Select the required file to load.

Note: Loading a file to a surface can be useful after the creation of a texture (see Section 2.5.1).

Note: It is also possible to load texture data from a file by editing the individual channels of a texture (see Section “Edit Individual Channels”) or by wrapping raw data (see Section 2.5.5).
2.3.2. Texture Information

The Texture Information window (Figure 8) shows a variety of information in regards to the currently loaded texture, such as its name, format, dimensions, and number of MIP maps, amongst others.

Note: When a texture diff is performed (see Section 2.3.4), the Texture Information window captures the details of the two textures being compared.

![Figure 8. Texture Information window](image)

2.3.3. Display Options

A texture is displayed in a view window which renders a single texture to the viewport, allowing users to verify that the texture looks correct for encoding or other purposes. Textures that have transparency, or have been pre-multiplied in the GUI, are alpha blended with a background. Options concerned with texture display are present in the Display Options window (Figure 9). Texture display in the GUI can be controlled in a number of ways, which are discussed in the next section.
Adjust the Display Mode of a Texture

The display mode of a texture in a view window can be adjusted by using the Display Mode settings (Figure 9a). The settings allow for different visualization modes to be used while working with textures:

- The Single display mode option is used to visualize a single instance of a texture regardless of the shape or size of its view window.
- The Tiled display mode option is used to tile the texture across the view panel.
- The Cube display mode option is used to visualize the texture rendered in a 3D cube map, allowing the user to move the camera around as if it were in the centre of a cube with the texture as one of the faces. If multiple cube map faces are specified in a texture then these are loaded onto the appropriate faces of the cube.

Zoom in and out of a Texture

Textures can be zoomed in and out for inspection. There are a number of ways to achieve this, namely:

- Use the slider provided in the Display Options window (Figure 9b).
- Use the slider provided in the Status bar (see Figure 1d).
- Use the mouse wheel on a view window.

View the Size of a Texture

The size of a texture, in pixels, is displayed at the bottom of the GUI in the Status bar (see Figure 1d).

View Cursor Position over a Texture

When the cursor is positioned over a texture, the system displays the coordinates, in pixels, of the cursor position, where 0, 0 corresponds to the coordinates of the top left hand corner of the image. The cursor position can be viewed from the Display Options window (Figure 9c) as well as in the Position parameter present in the Status bar (see Figure 1d).
Toggle the Colour Channel of a Texture
When the cursor is positioned over a texture, the system displays the RGBA value at that point (Figure 9d). To toggle a specific channel on or off and view its resultant effect on a texture, use its corresponding checkbox.

Note: When a texture diff is performed (see Section 2.3.4), the Colour section of the Display Options window captures the colour channel details of both textures and also displays the difference between the values from the textures being compared.

2.3.4. Texture Comparison
The purpose of texture comparison (i.e., texture diff) is to highlight the differences between input textures and their corresponding encoded outputs - see Section 2.6 for details of how to perform texture encoding. There are various ways in which the user is able to visualize these differences, notably by performing a colour diff, tolerance diff or diff by blending textures:

- **Colour diff**: A colour diff is a standard comparison of the colour channels of two images.
- **Tolerance diff**: A tolerance diff is a comparison that displays the difference between the original image and the second image as one of three colours. Differences higher than the tolerance are displayed in red, differences lower than the tolerance are displayed in blue, and any pixel with no difference is displayed in black.
- **Diff by blending textures**: A diff between two textures can be achieved by understanding how the two textures blend together. This is useful for performing side by side comparisons.

When a texture diff is performed between two files, a new view window is created to display the two images as well as the differences between them. The new window cannot be edited nor can a texture be saved from it. The following steps should be taken in order to perform a texture diff:

1. In the View window area, click the view window for the desired input file against which to diff.
2. From the Menu bar, click View -> Select Diff Input. This option is also available from the Quick access bar.
3. In the View window area, click the view window of the target file.
4. From the Menu bar, click View -> Diff Against <Input>. This option is also available from the Quick access bar. A new window will be presented showing the results of the diff (Figure 10). By default, a colour diff is shown.

Figure 10. Colour diff results
5. Under the Difference View Options in the Display Options window (see Figure 9e), type in a suitable brightness multiplier value to reveal the colour differences, as required. This assists in highlighting areas where the difference is minimal.

Note: Colour channels can also be toggled on and off after performing a colour diff.

If it is required to perform a tolerance diff, under the Difference View Options in the Display Options window (see Figure 9e), choose Tolerance Diff from the dropdown menu. The result will then be visible from the diff view window (Figure 11). Also adjust the tolerance value, between 0 and 1, in order to set the threshold under which any difference is ignored.

![Figure 11. Tolerance diff results](image)

If it is required to perform a diff by blending textures, under the Difference View Options in the Display Options window (see Figure 9e), choose Blend Textures from the dropdown menu. The result of the diff will then be visible from the diff view window (Figure 12). Adjust the blend visibility setting, which is a percentage value, to define the amount that each texture contributes to the resultant blended texture.

![Figure 12. Performing a diff by blending textures](image)

Compare Texture Information
During a texture comparison activity, it is possible to view the texture details of the two files by switching to the Texture Information window (see Figure 8).
Display Error Metrics for an Encoded Texture

After performing a texture diff, error metrics can be displayed to further help the texture comparison task. In order to view the error metrics, perform the following:

1. Perform a colour diff, tolerance diff or diff by blending textures.
2. From the Menu bar, click View -> Error Metrics. This option is also available from the Quick access bar. This will then open the Error Metrics dialog box which tabulates statistical information (Figure 13).

![Figure 13. Error Metrics dialog box](image)

2.4. Texture Pre-Processing

The GUI provides a set of tools for performing pre-processing tasks on textures and images. The tools can be accessed from the Toolbox window in the Tools section (Figure 14). Depending on the tool being chosen, different options are displayed below the Tools section.

![Figure 14. Toolbox window](image)

2.4.1. General Steps

The core steps for pre-processing a texture are the following:

1. Select the texture by clicking its view window.
2. From the Toolbox window (Figure 14), select the desired option. This will then display the corresponding settings.
3. Use the necessary settings and apply changes.

2.4.2. Options and Settings

Resize Texture
This option is used to resize a texture to the given size. Table 1 identifies the participating settings.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Width</td>
<td>Sets the desired resized width in pixels.</td>
</tr>
<tr>
<td>New Height</td>
<td>Sets the desired resized height in pixels.</td>
</tr>
<tr>
<td>Make Square</td>
<td>Sets the new width and height values to be the</td>
</tr>
<tr>
<td></td>
<td>same, giving a square texture.</td>
</tr>
<tr>
<td>Make Power Of Two</td>
<td>Sets the new width and height values to be the</td>
</tr>
<tr>
<td></td>
<td>next largest power of two.</td>
</tr>
<tr>
<td>Sample Filter: Nearest</td>
<td>Sets the filtering method to Nearest.</td>
</tr>
<tr>
<td>Sample Filter: Linear</td>
<td>Sets the filtering method to Linear.</td>
</tr>
<tr>
<td>Sample Filter: Cubic</td>
<td>Sets the filtering method to Cubic.</td>
</tr>
</tbody>
</table>

Flip along an Axis
This option is used to flip a texture over a given axis. Currently only the X and Y axes are supported, providing a standard 2D flip, either vertically or horizontally. Table 2 identifies the participating settings.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flip Horizontal</td>
<td>Flips the texture horizontally.</td>
</tr>
<tr>
<td>Flip Vertical</td>
<td>Flips the texture vertically.</td>
</tr>
</tbody>
</table>

Rotate by 90 Degrees
This option is used to rotate a texture around a given axis. Currently only the Z axis is supported, providing a standard 2D rotate. Table 3 identifies the participating settings.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rotate Clockwise</td>
<td>Rotates the texture clockwise by 90 degrees.</td>
</tr>
<tr>
<td>Rotate Anti-Clockwise</td>
<td>Rotates the texture anti-clockwise by 90 degrees.</td>
</tr>
</tbody>
</table>

Add a Mirrored Border
The purpose of adding a small mirrored border is to improve the quality of compressed textures whose compression algorithm assumes that the texture wraps. This technique should be used for non-tiling textures and with UV coordinates adjusted to not use the border. Table 4 identifies the settings that can be applied to add a mirrored border to a texture.
Table 4. Settings for adding a mirrored border to a texture

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Border Size: PVRTC 2bpp (8x4)</td>
<td>This is a pre-defined border size specifically designed to improve the quality of non-wrapped PVRTC 2 bits per pixel.</td>
</tr>
<tr>
<td>Border Size: PVRTC 4bpp (4x4)</td>
<td>This is a pre-defined border size specifically designed to improve the quality of non-wrapped PVRTC 4 bits per pixel.</td>
</tr>
<tr>
<td>Border Size: User Defined:</td>
<td>This allows a user-defined border to be specified for other texture compression formats. The width and height of the border can be adjusted.</td>
</tr>
</tbody>
</table>

Create a MIP Map Chain
This option is used to generate MIP maps for the current texture. Table 5 identifies the participating settings.

Table 5. Settings for generating a MIP map chain for a texture

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample Filter: Nearest</td>
<td>Sets the filtering method to Nearest.</td>
</tr>
<tr>
<td>Sample Filter: Linear</td>
<td>Sets the filtering method to Linear.</td>
</tr>
<tr>
<td>Sample Filter: Cubic</td>
<td>Sets the filtering method to Cubic.</td>
</tr>
<tr>
<td>Number Of MIP Maps</td>
<td>This dropdown box allows the number of generated MIP maps to be adjusted. The default setting is that of Full MIP Map Chain.</td>
</tr>
</tbody>
</table>

Resize Canvas
This option is used to resize a texture to the given size, without changing the image data. Table 6 identifies the participating settings.

Table 6. Settings for resizing a texture canvas

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Width</td>
<td>Sets the desired resized canvas width in pixels.</td>
</tr>
<tr>
<td>New Height</td>
<td>Sets the desired resized canvas height in pixels.</td>
</tr>
<tr>
<td>Make Square</td>
<td>Makes the canvas square by increasing the smallest height or width value to equal the largest height or width value.</td>
</tr>
<tr>
<td>Make Power Of Two</td>
<td>Increases the new height or width values to the next power of two, e.g., 256 pixels, 512 pixels, 1024 pixels, etc.</td>
</tr>
<tr>
<td>X Offset</td>
<td>Offsets the current texture from the top left of the new canvas. The offset amount is a value, in pixels, in the positive X-axis direction.</td>
</tr>
<tr>
<td>Y Offset</td>
<td>Offsets the current texture from the top left of the new canvas. The offset amount is a value, in pixels, in the negative Y-axis direction.</td>
</tr>
<tr>
<td>Centre</td>
<td>Centres the texture within the new canvas.</td>
</tr>
</tbody>
</table>
Optimise Transparent Areas
Optimising transparent areas 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 pass done on the texture. If the textures being used have no useful data in their alpha channel, it is strongly recommended that this option be used. To optimise the transparent areas of a texture, use the appropriate option and click the Bleed button.

Multiply Colours by the Alpha Channel
Run-time optimisation of a texture can be achieved by pre-multiplying the colour channels of its pixels by its alpha channels. To multiply colours by the alpha channel, use the appropriate option and click the Pre-Multiply Alpha button to complete the procedure.

Edit Individual Channels
The contents of a colour channel (alpha, red, green and blue) within a texture can be set based on either a channel within the current image, another image, an integer or float. Table 7 lists the source settings that can be applied to edit the individual channels of a texture.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source: Current Texture</td>
<td>Sets the current texture as the source. The required channel can then be chosen from the Red, Green, Blue, Alpha, Luminance or Intensity options in the Channel dropdown box.</td>
</tr>
<tr>
<td>Source: File</td>
<td>Sets another texture as the source by allowing the user to browse to the file. The required channel can then be chosen from the Red, Green, Blue, Alpha, Luminance or Intensity options in the Channel dropdown box.</td>
</tr>
<tr>
<td>Source: Integer</td>
<td>Sets an integer value as the source value. Integer values can be in the range 0 to 255.</td>
</tr>
<tr>
<td>Source: Float</td>
<td>Sets a float value as the source value. Float values can be in the range 0 to 1.</td>
</tr>
</tbody>
</table>

For example, it is possible to set the red channel of a texture to be the same as its blue channel by identifying the Destination: Red settings, choosing the Current Texture from the Source dropdown box and selecting the Blue option from the Channel dropdown box. Similarly, swapping the contents of the red channel of a texture with the blue channel from another texture can be done by identifying the Destination: Red settings, choosing the File option from the Source dropdown box, selecting the desired file and then selecting the Blue option from the Channel dropdown box.

Colourise MIP Map Levels
It is possible to colourize the MIP map levels of a texture using false colouring in order to aid the visual inspection of a texture when using the Surface Browser window (see Section 2.3.1). This is mostly useful for testing, so that MIP map levels can be spotted easily. To colourise MIP map levels, use the appropriate option and click the Colour MIP Map Levels button to complete the procedure.

Create a Normal Map
A normal map can be created for a texture by assuming that it is a height map, based on the intensity of its RGB channels. Table 8 identifies the settings that can be applied to create a normal map for a texture.
### Table 8. Settings for creating a normal map

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Height Scale</td>
<td>Sets the scaling of the height map from which the normal map is generated.</td>
</tr>
<tr>
<td>Channel Order</td>
<td>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. Channel values are X Axis, Y Axis, Z Axis and N/A.</td>
</tr>
</tbody>
</table>

### 2.5. Texture Creation and Loading

Textures can be created on demand using the GUI. Basic textures, cube maps, texture arrays and font textures are supported.

#### 2.5.1. Create a New Texture

To create a new texture, perform the following steps:

1. **Click** File -> New in the Menu bar. This option is also available from the Quick access bar. This will open the Create New Texture dialog box (Figure 15).

![Create New Texture dialog box](image)

**Figure 15. Create New Texture dialog box**

2. Adjust the necessary settings in the dialog box. Values can be specified for the MIP map levels, number of faces, number of array members and the size of the texture in pixels.

3. Click the OK button. This will create an empty texture canvas of the specified size.

**Note:** To load a file to a surface, follow the steps identified in Section “Load File to a Surface”. It is also possible to load a file while editing the individual channels of a texture (see Section “Edit Individual Channels”).

#### 2.5.2. Create a Cube Map

A cube map is a texture that uses the faces of a cube as the map shape and is a simple and computationally efficient means of performing environmental mapping. To create a cube map, perform the following steps:

1. **Click** File -> Create Cubemap in the Menu bar. This will open the Compose Cubemap dialog box (Figure 16).

2. For each of the six faces of the cube, browse to and select the desired file to use. The texture for each face is made square and resized to the same size as the texture used on the X-Positive face, all with a linear filter. Any additional cube face, surface or MIP map level in a texture used on a face is discarded.

3. Click the OK button to complete the procedure. The cube map can then be inspected in the Surface Browser window.
2.5.3. Create a Texture Array

Texture arrays are collections of textures stored within the same file and can be used for several purposes including as an alternative to texture atlases and for environmental mapping, amongst others. To create a texture array, perform the following steps:

1. Click File -> Create Texture Array... in the Menu bar. This will open the Create Texture Array dialog box (Figure 17).

2. Click the Add Texture(s) button to include individual textures making up the texture array. As textures are added, the dialog box will list them in the order that the files have been selected. The textures are resized to the same size as the texture in the first position with a linear filter. Any additional cube face, surface or MIP map level in a texture is discarded.

3. The order of the textures within the texture array can be changed by dragging and dropping the file names above or below one another in the dialog box. Individual files can also be removed by selecting them and clicking the Remove Texture(s) button.
4. Once all the required files have been added, click the OK button to complete the procedure. The texture array can then be inspected in the Surface Browser window.

### 2.5.4. Create a Font Texture

It is possible to render a set of text into a texture, in a font specified by the user, with supporting metadata about how the characters should be handled. This is used primarily for PVRUIRenderer but can be used elsewhere. OTF and TTF font files are supported, as are a variety of different character sets. To create a font texture, perform the following steps:

1. **Click File -> Create Font Texture...** in the Menu bar. This will open the Create Font Texture dialog box (Figure 18).

![Create Font Texture dialog box](image)

**Figure 18. Create Font Texture dialog box**

2. Specify the location of the font file.
3. Adjust the font settings, as appropriate. Table 9 lists the parameters that can be adjusted when creating a font texture.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>Sets the font size in pixels.</td>
</tr>
<tr>
<td>Border</td>
<td>Sets the size, in pixels, of a border to be added around each character in the font texture.</td>
</tr>
<tr>
<td>Grid Fitting</td>
<td>When this option is ticked, the height and width of the font characters are modified to align with the set pixel width of screen display.</td>
</tr>
<tr>
<td>Anti Aliasing</td>
<td>When this option is ticked, aliasing is prevented and the appearance of the font texture is improved.</td>
</tr>
</tbody>
</table>
### Setting

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character Set</td>
<td>This determines the set of characters that will be rendered into the font texture. Three options are available:</td>
</tr>
<tr>
<td></td>
<td>- ASCII Character Set (7-Bit): Renders the full set of 128 ASCII characters to the texture.</td>
</tr>
<tr>
<td></td>
<td>- User-Defined String: Presents a text box that the user can type or paste a custom string. All characters in the string will be rendered into the font in the most appropriate order. Note that duplicate characters are only rendered once. Unicode characters are supported.</td>
</tr>
<tr>
<td></td>
<td>- Text File: Allows the user to specify a text file which contains a custom string for rendering. The string is treated in the same way as for a User Defined String.</td>
</tr>
</tbody>
</table>

4. If choosing a character set, pick an option from the Character Set dropdown box.
5. Click the OK button to complete the procedure.

#### 2.5.5. Wrap Raw Data

Wrapping raw data is a texture loading task, whereby it is possible to manually specify how PVRTexTool should read texture data. Raw data can be loaded to a texture by wrapping the data to it.

*Note: Other supported texture data loading tasks are through loading a file to a surface (see Section “Load File to a Surface”) or editing the individual channels of a texture (see Section “Edit Individual Channels”).*

To wrap raw data, perform the following steps:

1. Click File -> Wrap Raw Data… in the Menu bar. This will open the Wrap Raw Data dialog box (Figure 19).
2. Load the required data file by browsing to it. The data should be in the order set in Section “Texture Data” of the “PVR File Format Specification” document, and is as follows:

```plaintext
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
```
3. Adjust the required settings in the dialog box. Table 10 provides a list of the settings that can be applied.

Table 10. Settings used for wrapping raw data

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>The width and height of the texture represented by the raw data in the data file.</td>
</tr>
<tr>
<td>Surfaces</td>
<td>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 array members in the case of a texture array.</td>
</tr>
<tr>
<td>Pixel Format</td>
<td>The format of the image. If it is uncompressed data, this also states the bit rate and channel order of the image.</td>
</tr>
<tr>
<td>Variable Type</td>
<td>The data type of the raw data held within the data file.</td>
</tr>
<tr>
<td>Colour Space</td>
<td>The colour space that the raw data in the data file is in.</td>
</tr>
<tr>
<td>Pre-Multiplied</td>
<td>This is used to indicate whether the data is pre-multiplied by its alpha channel or not.</td>
</tr>
<tr>
<td>Data Offset</td>
<td>The data offset is used to adjust the offset into the raw data at which the actual texture data starts, within the data file. This is particularly useful for restoring data within corrupted headers, where the size of the header can be entered into the offset, and the header skipped. Assume Header Size indicates whether a PVR3 header should be assumed to exist at the beginning of the file.</td>
</tr>
</tbody>
</table>

4. Click the OK button to complete the procedure.

2.6. Texture Encoding

Texture encoding (or compression) is the primary function of PVRTexTool, whereby images are encoded using fixed rate compression schemes for efficient rendering in graphics systems. In the GUI, textures can be encoded in a wide range of formats. In order to encode a texture, perform the following steps:
1. Click Edit -> Encode... in the Menu bar. This option is also available from the Quick access bar. This will open the Encode Texture dialog box which provides the necessary workflow to help the user carry out the encoding task (Figure 20).

![Encode Texture dialog box](image)

2. Use the provided filters to choose the required format for compression (Figure 20a). As options are selected from the dropdown boxes, the corresponding encoding formats meeting the filter criteria are listed (Figure 20b).
3. Select the desired encoding format from the list.
4. Select the required encoding mode if the selected format supports it (Figure 20c).
5. Choose additional options, as appropriate (Figure 20d). For example, ticking the Generate MIPMaps checkbox creates all MIP map levels for the resultant texture. Similarly, ticking the Show As Diff View checkbox generates the new encoded texture and additionally displays a diff view window comparing the original texture against the newly compressed one.
6. Click the Encode button to complete the procedure (Figure 20e). This will create the new encoded texture which can then be saved.

**Note:** It is possible to reset the filters at any time by using the Reset Filters button (Figure 20f).

### 2.6.1. Encoding Modes

When encoding a texture, there are several modes to choose from depending on the format that the texture is being encoded into. The majority of formats (such as RGBA, LA or RG) have only one mode of encoding, and so the option shown in Figure 20c does not appear.

**PVRRTC**

When working with this format, the available encoding modes are:

- **Fastest:** This mode should be utilized when encoding a texture quickly than in high quality.
- **Fast**: When quick but good quality results are required, the *Fast* mode is better suited than *Fastest* (which has lower quality) and *Normal* (which takes longer).
- **Normal**: This mode offers results of good quality that take a reasonable time to encode.
- **High**: Providing the second-best compression quality available, *High* mode pays for this benefit by being slower to encode textures than the preceding options.
- **Very High**: Offering the best quality textures, this mode takes time to effectively encode them.

**ETC**
The available encoding modes for this format are:

- **Fast**: Similar to the mode used for encoding PVRTC, the *Fast* mode produces a relatively low quality result at the fastest possible speed using standard image metrics. This mode is useful for encoding non-image data such as normals.
- **Fast Perceptual**: This mode uses non-standard image metrics, making it more suitable for encoding image data unlike the standard modes.
- **Slow**: This mode takes longer to achieve results, but it produces higher quality work after encoding.
- **Slow Perceptual**: Similar to the regular *Slow*. This mode utilizes non-standard image metrics which are more suited to encoding image data.

**ASTC**
The available modes of compression are:

- **Very Fast**: This runs the compression at a high speed, though this results in a relatively large loss of quality.
- **Fast**: Running the operation in this mode results in minor quality loss whilst still providing results in a timely manner.
- **Medium**: Using this mode results in a balance between quality and time taken to compress.
- **Thorough**: This mode provides greater detail in the result than the preceding modes.
- **Exhaustive**: Running the operation in Exhaustive modes provides the greatest level of detail in the results, but it takes the longest time to run.
3. PVRTexTool CLI

3.1. Usage Instructions
The CLI can be used to process and compress textures, or by using a batch file. The syntax for the command-line is described next. 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 to run multiple instances of the tool in parallel.

3.2. Examples
In the next example, an image named Example.bmp is taken and a full chain of MIP maps is generated for it and encoded to ARGB 1555. The file Example.pvr is created.

```
PVRTexToolCLI -m -f alr5g5b5 -i Example.bmp
```

In the next example, a cube map is generated from files named skybox<n>.bmp, encoded to PVRTC1 4bpp and saved as a data structure in a Direct Draw Surface. The file skybox.dds is created.

```
PVRTexToolCLI -i skybox1.bmp,skybox2.bmp,skybox3.bmp,skybox4.bmp,skybox5.bmp,skybox6.bmp -cube
-m -f pvrtc1_4 -o skybox.dds
```

3.3. Command-Line Options
Table 11 lists the command-line options.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description, usage and example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input file</td>
<td>[REQUIRED] Sets the input file or files. Must be a JPEG, PNG, BMP, PVR, KTX, DDS or ASTC file. If either the cube map or texture array flag is set, multiple files should be explicitly specified.</td>
</tr>
<tr>
<td>Usage:</td>
<td>-i [filepath],&lt;additionalfiles...&gt;</td>
</tr>
<tr>
<td>Example:</td>
<td>-i picture.jpg, otherpicture.png</td>
</tr>
<tr>
<td>Output file</td>
<td>Sets 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.</td>
</tr>
<tr>
<td>Usage:</td>
<td>-o &lt;filepath&gt;</td>
</tr>
<tr>
<td>Example:</td>
<td>-o texture.pvr</td>
</tr>
<tr>
<td>Option</td>
<td>Description, usage and example</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>Decompress output</td>
<td>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 cube maps and texture arrays, behaviour is special; a series of decompressed files is output in the following format: <code>&lt;Name&gt;.Face&lt;FaceNumber&gt;.Array&lt;ArrayNumber&gt;.&lt;Extension&gt;</code>, e.g., <code>Texture.Face1.Array12.png</code></td>
</tr>
<tr>
<td>Usage: <code>−d &lt;filepath&gt;</code></td>
<td>Example: <code>−d decompressed.png</code></td>
</tr>
<tr>
<td>Cube map</td>
<td>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.</td>
</tr>
<tr>
<td>Usage: <code>−cube &lt;faceorder&gt;</code></td>
<td>Example: <code>−cube +X,−X,+Y,−Y,+Z,−Z</code></td>
</tr>
<tr>
<td>Texture array</td>
<td>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.</td>
</tr>
<tr>
<td>Usage: <code>−array</code></td>
<td>Example: <code>−array</code></td>
</tr>
<tr>
<td>Pad metadata</td>
<td>Adds padding 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.</td>
</tr>
<tr>
<td>Usage: `−pad [2</td>
<td>4</td>
</tr>
<tr>
<td>Legacy PVR (Deprecated)</td>
<td>Forces the output file to be saved out as a legacy PVR file (PVR v2) for backwards compatibility purposes.</td>
</tr>
<tr>
<td>Usage: <code>−legacypvr</code></td>
<td>Example: <code>−legacypvr</code></td>
</tr>
<tr>
<td>Resize</td>
<td>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.</td>
</tr>
<tr>
<td>Usage: <code>−r [width],[height]</code></td>
<td>Example: <code>−r 512,256</code></td>
</tr>
<tr>
<td>Resize square</td>
<td>Forces the texture into a square. A single character parameter, − or +, can be specified to tell whether it is resized smaller (i.e., −) or larger (i.e., +). Incompatible with standard resize.</td>
</tr>
<tr>
<td>Usage: `−square &lt;+</td>
<td>−&gt;`</td>
</tr>
<tr>
<td>Option</td>
<td>Description, usage and example</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>Resize power of two</td>
<td>Forces the texture into power of two dimensions. A single character parameter, − or +, can be specified to tell whether it is resized smaller (i.e., −) or larger (i.e., +). Incompatible with standard resize.</td>
</tr>
</tbody>
</table>
| **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 and − 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 metadata 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. Alternatively, border sizes can be chosen manually for the width and height. Specifying just the width is allowed, and results in the vertical border to have 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 |
<table>
<thead>
<tr>
<th>Option</th>
<th>Description, usage and example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal map generation</td>
<td>Uses the input texture as a height map to generate a normal map (by creating an intensity texture from the r, g and b channels). 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 and the default is xyz.</td>
</tr>
<tr>
<td></td>
<td>Usage: <code>-n [scale],&lt;channelorder&gt;</code></td>
</tr>
<tr>
<td></td>
<td>Example: <code>-n 1.0,xyz</code></td>
</tr>
<tr>
<td>MIP map generation</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>Usage: <code>-m &lt;numberofmipmaps&gt;</code></td>
</tr>
<tr>
<td></td>
<td>Example: <code>-m 9</code></td>
</tr>
<tr>
<td>MIP map filter</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>Usage: `-mfilter [nearest</td>
</tr>
<tr>
<td></td>
<td>Example: <code>-mfilter cubic</code></td>
</tr>
<tr>
<td>Colour MIP maps</td>
<td>Saturates the tail of the MIP map chain with colours for debugging purposes. If the original colour of a texture cannot be seen, then it is known that it is using MIP maps at all times and some upper levels can be removed from the texture to reduce memory consumption.</td>
</tr>
<tr>
<td></td>
<td>Usage: <code>-c</code></td>
</tr>
<tr>
<td></td>
<td>Example: <code>-c</code></td>
</tr>
<tr>
<td>Option</td>
<td>Description, usage and example</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>Encode format</td>
<td>[REQUIRED] Sets the format to encode to. The first argument (required) is the format, which can be either a compressed format or a non-compressed format in the form r8g8b8a8. Any number of channels up to four 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 and defaults to normalised unsigned byte. Specifying a type is generally not required for compressed formats, but generally is for uncompressed formats. The third and final argument specifies the colour space, which accepts either sRGB or lRGB and the default is lRGB for linear RGB. Valid formats:</td>
</tr>
<tr>
<td></td>
<td>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_RGB_A, ETC2_RGB_A1, EAC_R11, EAC_RG11, ASTC_4x4, ASTC_5x4, ASTC_5x5, ASTC_6x5, ASTC_6x6, ASTC_8x5, ASTC_8x6, ASTC_8x8, ASTC_10x5, ASTC_10x6, ASTC_10x8, ASTC_10x10, ASTC_12x10, ASTC_12x12, ASTC_3x3x3, ASTC_4x3x3, ASTC_4x4x3, ASTC_4x4x4, ASTC_5x4x4, ASTC_5x5x4, ASTC_5x5x5, ASTC_6x5x5, ASTC_6x6x5, ASTC_6x6x6</td>
</tr>
<tr>
<td></td>
<td>Valid variable types:</td>
</tr>
<tr>
<td></td>
<td>UB, UBN, SB, SBN, US, USN, SS, SSN, UI, UIN, SI, SIN, UF, SF</td>
</tr>
<tr>
<td></td>
<td>Key:</td>
</tr>
<tr>
<td></td>
<td>First Char: S=Signed, U=Unsigned</td>
</tr>
<tr>
<td></td>
<td>Second Char: B=Byte, S=Short, I=Integer, F=Float</td>
</tr>
<tr>
<td></td>
<td>Third Char (optional): N=Normalised.</td>
</tr>
<tr>
<td></td>
<td>Valid Colour Spaces:</td>
</tr>
<tr>
<td></td>
<td>IRGB, sRGB</td>
</tr>
<tr>
<td>Usage:</td>
<td>−f [format],&lt;variabletype&gt;,&lt;colourspace&gt;</td>
</tr>
<tr>
<td>Example:</td>
<td>−f PVRTC1_2,UBN,lRGB</td>
</tr>
<tr>
<td>Note:</td>
<td>PVRTexTool requires ASTC Evaluation Codec (astcenc) to be in your path to be able to compress to ASTC formats. This tool is freely available from ARM.</td>
</tr>
<tr>
<td>Encode quality</td>
<td>Sets the quality level to compress with. Only currently useful with ETC and PVRTC formats.</td>
</tr>
<tr>
<td>Usage:</td>
<td>−q [compressorquality]</td>
</tr>
<tr>
<td>Example:</td>
<td>−q pvrtcfast</td>
</tr>
<tr>
<td>Option</td>
<td>Description, usage and example</td>
</tr>
<tr>
<td>------------</td>
<td>---------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Dither</td>
<td>Tells the compressor to dither the texture before compression to avoid banding artefacts.</td>
</tr>
<tr>
<td></td>
<td><strong>Usage:</strong> <code>-dither</code></td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> <code>-dither</code></td>
</tr>
<tr>
<td>Silence</td>
<td>Tells the utility to not output messages of any kind.</td>
</tr>
<tr>
<td></td>
<td><strong>Usage:</strong> <code>-shh</code></td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> <code>-shh</code></td>
</tr>
<tr>
<td>Help</td>
<td>Requests help for either a list of commands or for help on a specified argument, if the argument is a parameter.</td>
</tr>
<tr>
<td></td>
<td><strong>Usage:</strong> <code>-help &lt;commandargument&gt;</code></td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> <code>-help &quot;flip&quot;</code></td>
</tr>
<tr>
<td>Red channel</td>
<td>Sets the red channel in the input texture to match the channel specified in a second image.</td>
</tr>
<tr>
<td></td>
<td>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 <code>r, g, b, a, l, i</code>. These represent red, green, blue, alpha, luminance and intensity.</td>
</tr>
<tr>
<td></td>
<td><strong>Usage:</strong> <code>-red [filename],&lt;channelname&gt;</code></td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> <code>-red Red.png,g</code></td>
</tr>
<tr>
<td>Green channel</td>
<td>Sets the green channel in the input texture to match the channel specified in a second image.</td>
</tr>
<tr>
<td></td>
<td>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 <code>r, g, b, a, l, i</code>. These represent red, green, blue, alpha, luminance and intensity.</td>
</tr>
<tr>
<td></td>
<td><strong>Usage:</strong> <code>-green [filename],&lt;channelname&gt;</code></td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> <code>-green Green.dds,b</code></td>
</tr>
<tr>
<td>Blue channel</td>
<td>Sets the blue channel in the input texture to match the channel specified in a second image.</td>
</tr>
<tr>
<td></td>
<td>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 <code>r, g, b, a, l, i</code>. These represent red, green, blue, alpha, luminance and intensity.</td>
</tr>
<tr>
<td></td>
<td><strong>Usage:</strong> <code>-blue [filename],&lt;channelname&gt;</code></td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> <code>-blue Blue.pvr,r</code></td>
</tr>
<tr>
<td>Option</td>
<td>Description, usage and example</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>Alpha channel</td>
<td>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: <code>-alpha [filename],&lt;channelname&gt;</code> Example: <code>-alpha Alpha.bmp,i</code></td>
</tr>
</tbody>
</table>
| 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` |
<p>| 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 <code>8192x8192</code> are supported. Option is incompatible with square or power of two canvas resize options. Usage: <code>-rcanvas [width],[height]</code> Example: <code>-rcanvas 512,256</code> |
| Resize canvas square| Forces the texture into a square, without changing the image data. This takes effect after resizing. A single character parameter, <code>−</code> or <code>+</code>, can be specified to tell whether it is resized smaller (i.e., <code>−</code>) or larger (i.e., <code>+</code>). Incompatible with standard resize. Usage: <code>-squarecanvas &lt;+|-&gt;</code> Example: <code>-squarecanvas +</code> |
| 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, <code>−</code> or <code>+</code>, can be specified to tell whether it is resized smaller (i.e., <code>−</code>) or larger (i.e., <code>+</code>). Incompatible with standard resize. Usage: <code>-potcanvas &lt;+|-&gt;</code> Example: <code>-potcanvas +</code> |</p>
<table>
<thead>
<tr>
<th>Option</th>
<th>Description, usage and example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offset canvas</td>
<td>Sets the offset when performing a canvas resize (including square or power of two sizes). Accepts two signed integer parameters: <code>xoffset</code> and <code>yoffset</code>. Values between -8192x-8192 and 8192x8192 are supported. Incompatible with centre canvas. Usage: <code>-offsetcanvas [xoffset],[yoffset]</code> Example: <code>-offsetcanvas -12,56</code></td>
</tr>
<tr>
<td>Centre canvas</td>
<td>Sets the offset when performing a canvas resize (including square or power of two sizes) so that the image resides in the centre of the canvas. Incompatible with canvas offset. Usage: <code>-centrecanvas</code> Example: <code>-centrecanvas</code></td>
</tr>
</tbody>
</table>
4. PVRTexTool Plugins

The Plugins are available for various tools and supported across different platform versions as identified in Table 12.

<table>
<thead>
<tr>
<th>Plugin for:</th>
<th>Windows</th>
<th>OS X</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>32-bit</td>
<td>64-bit</td>
</tr>
<tr>
<td>Adobe Photoshop</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Autodesk 3ds Max</td>
<td>✓ (Up to 2013)</td>
<td>✓ (2010 onwards)</td>
</tr>
<tr>
<td>Autodesk Maya</td>
<td>✓ (Up to 2013)</td>
<td>✓ (2010 onwards)</td>
</tr>
<tr>
<td>Windows Explorer</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 12. Provision for plugins

In the examples listed throughout this section, placeholder variables are used to facilitate ease of use. These variables are:

- `<SDK_ROOT>`: This variable indicates the root directory in which the SDK is installed.
- `<PLATFORM>`: This variable indicates the platform being used, e.g., Windows, OS X, etc.
- `<VERSION>`: This variable indicates the version of the utility the plugin is being used for.

4.1. Plugin for Adobe Photoshop

This plugin allows Adobe Photoshop to save and load PVR files, with the option to encode when saving.

4.1.1. Installation

Copy:

```
<SDK_ROOT>\PowerVR_Tools\PVRTexTool\Plugins\Photoshop\<PLATFORM>\PVRFormat.8bi
```

To:

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

4.1.2. Uninstallation

Delete:

```
<PHOTOSHOP_DIR>\plug-ins\File Formats\PVRFormat.8bi
```
4.2. **Plugin for Autodesk 3ds Max**

This plugin allows Autodesk 3ds Max to save and load PVR files. PVR becomes available as a material type similar to bitmaps and users are able to save final rendered images in this format.

**4.2.1. Installation**

Copy:

```markdown
<SDK_ROOT>\PowerVR_Tools\PVRTexTool\Plugins\3DSMax\<YEAR>\<PLATFORM>\PVRTexTool_<VERSION>.dle
```

to:

```markdown
<3DSMAX_DIR>\plug-ins\PVRTexTool_<VERSION>.dle
```

**4.2.2. Uninstallation**

Delete:

```markdown
<3DSMAX_DIR>\plug-ins\PVRTexTool_<VERSION>.dle
```

4.3. **Plugin for Autodesk Maya**

This plugin allows Autodesk Maya 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.

**4.3.1. Installation**

For Windows, copy:

```markdown
<SDK_ROOT>\PowerVR_Tools\PVRTexTool\Plugins\Maya\<YEAR>\<PLATFORM>\PVRTexTool_<VERSION>.mll
```

to:

```markdown
<MAYA_DIR>\bin\plug-ins\image\PVRTexTool_<VERSION>.mll
```

For OS X, copy:

```markdown
<SDK_ROOT>\PowerVR_Tools\PVRTexTool\Plugins\Maya\<YEAR>\<PLATFORM>\PVRTexTool_<VERSION>.bundle
```

to:

```markdown
/Users/Shared/Autodesk/maya/<MAYA_VERSION>/plug-ins/
```

Once the plugin has been installed it must be activated in the Autodesk Maya Plug-in Manager.

4.3.2. Uninstallation
For Windows, delete the following file:

```
<MAYA_DIR>\bin\plug-ins\image\PVRTexTool_<VERSION>.mll
```

For OS X, delete the following file:

```
/Users/Shared/Autodesk/maya/<MAYA_VERSION>/plug-ins/PVRTexTool_<VERSION>.bundle
```

4.4. Plugin for Windows Explorer
This plugin allows the user to preview PVR files in Windows Explorer. The icon for PVR files is replaced with a thumbnail and the preview pane displays the texture rendered to the pane.

4.4.1. Installation
To install the Windows Explorer plugin, run one of the following commands with administrator access depending on the version of Windows running, i.e., 32-bit or 64-bit.

For 32-bit Windows, run:

```
Regsvr32.exe <SDK_ROOT>\PowerVR_Tools\PVRTexTool\WindowsExplorer_x86_32\PVRTextureViewer.dll
```

For 64-bit Windows, run:

```
Regsvr32.exe <SDK_ROOT>\PowerVR_Tools\PVRTexTool\WindowsExplorer_x86_64\PVRTextureViewer.dll
```

4.4.2. Uninstallation
To uninstall the Windows Explorer plugin, run one of the following commands with administrator access depending on the version of Windows running, i.e., 32-bit or 64-bit.

For 32-bit Windows, run:

```
Regsvr32.exe /u <SDK_ROOT>\PowerVR_Tools\PVRTexTool\WindowsExplorer_x86_32\PVRTextureViewer.dll
```

For 64-bit Windows, run:

```
Regsvr32.exe /u <SDK_ROOT>\PowerVR_Tools\PVRTexTool\WindowsExplorer_x86_64\PVRTextureViewer.dll
```
5. **PVRTexTool Library**

5.1. **Library Overview**

The Library is used 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-processing techniques on decompressed pixel data.
- Provide information about a texture file loaded by the library.

The following dynamic library files are provided:

- `PVRTexLib.dll` *(Windows dynamic link library).*
- `libPVRTexLib.so` *(Linux shared object library).*
- `libPVRTexLib.dylib` *(OS X dynamic library).*

Also present are a number of header files that define functionality of the Library:

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

Finally, a number of helper header files are also present:

- `PVRTGlobal.h`
- `PVRTError.h`
- `PVRTArray.h`
- `PVRTMap.h`
- `PVRTString.h`
- `PVRTTexture.h`

`PVRTexUtility.h` is the primary header file, and including it in a project includes all other header files required for the Library to function.

*Note: Further documentation for the Library is available in the Documentation folder of PVRTexTool installation directory.*

5.1.1. **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. If the resulting program is linked against the shared libraries then these libraries must be present on the target system.
5.1.2. Using the DLL (Windows only)
When developing for Windows using the Library, the pre-processor define _WINDLL_IMPORT must be set.

5.2. Example Code

5.2.1. 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.

```cpp
#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,
PVRStandard8PixelType,
ePVRTVarTypeUnsignedByteNorm,
ePVRTCSpace1RGB
);
// cTexture should now be in the format RGBA8888, with each channel being of the type unsigned integer, in the lRGB colour space.
```

5.2.2. 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 PVRTC1, 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 the Library’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 "PVRTTexture.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, ePVRTF_PVRTCI_4bpp_RGB, ePVRTVarTypeUnsignedByteNorm, ePVRTCspace1RGB);

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

5.2.3. 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 "PVRTTexture.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");

5.2.4. Create 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 "PVRTTexture.h"

using namespace pvrtexture;

// The pixel data is a pointer called 'pData'.
// Create the header.
CPVRTextureHeader cHeader( ePVRTF_PVRTCI_4bpp_RGB,
                          512,
                          512
                        );

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

// Save the file
cTexture.saveFile("out.pvr");
5.2.5. **Accessing Metadata**

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

```cpp
#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. 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 Tools and SDK and Insider programme, please visit:
http://www.powervrinsider.com

For general enquiries, please visit our website:
http://imgtec.com/corporate/contactus.asp