PVR File Format

Specification

Filename	:	PVR File Format.Specification
Version	:	PowerVR SDK REL_3.4@3164023a External Issue
Issue Date	:	30 Sep 2014
Author	:	Imagination Technologies Limited

Contents

List of Figures

List of Tables

1. Introduction

1.1. Document Overview

The purpose of this document is to act as a specification for the PVR file format (PVR specification version 3.0.0).

1.2. PVR Format Overview

PVR files contain three elements:

- One Header, of 52 bytes length.
- Zero or more meta-data elements, whose length can be determined from the header.
- One texture data element, whose length can be determined from the header.

The file is laid out as shown in Figure 1.

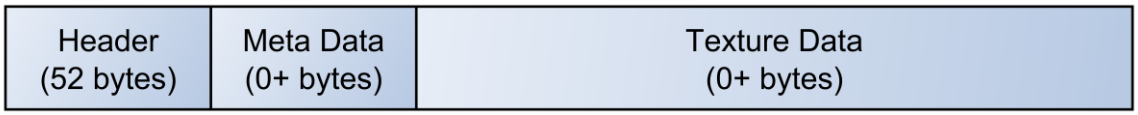

Figure 1. File layout

2. Format Description

2.1. Header Format

Table 1 lists details of the header format.

Table 1. Header format

Name	Size (bytes)	Data Format
Version	4	Unsigned 32bit Integer
Flags	4	Unsigned 32bit Integer
Pixel Format	8	Unsigned 64bit Integer
Colour Space	4	Unsigned 32bit Integer
Channel Type	4	Unsigned 32bit Integer
Height	4	Unsigned 32bit Integer
Width	4	Unsigned 32bit Integer
Depth	4	Unsigned 32bit Integer
Num. Surfaces	4	Unsigned 32bit Integer
Num. Faces	4	Unsigned 32bit Integer
MIP-Map Count	4	Unsigned 32bit Integer
Meta Data Size	4	Unsigned 32bit Integer

2.1.1. Version

'Version' contains the version of the PVR header format. The exact value of 'Version' will be one of the following depending upon the endianess of the file, and the computer reading it:

- 0x03525650, if endianess does match.
- 0x50565203, if endianess does not match.

2.1.2. Flags

The purpose of the 'Flags' 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 flags identified in Table 2 are currently supported.

Table 2. Flags

Name	Value	Description
No Flag	0	No flag has been set.
Pre-multiplied	0x02	When this flag is set, colour values within the texture have been pre-multiplied by the alpha values.

2.1.3. Pixel Format

'Pixel Format' is a 64bit unsigned integer containing the pixel format of the texture data, where the most significant 4 bytes have been set to '0' and the least significant 4 bytes will contain a 32bit unsigned integer value identifying the pixel format. The values are listed in Table 3.

Table 3. Pixel format

Formats	Value
PVRTC 2bpp RGB	0
PVRTC 2bpp RGBA	1
PVRTC 4bpp RGB	2
PVRTC 4bpp RGBA	3
PVRTC-II 2bpp	4
PVRTC-II 4bpp	5
ETC1	6
DXT1	7
DXT2	8
DXT3	9
DXT4	10
DXT5	11
BC1	7
BC2	9
BC3	11
BC4	12
BC5	13
BC6	14
BC7	15
UYVY	16
YUY2	17
BW1bpp	18
R9G9B9E5 Shared Exponent	19
RGBG8888	20
GRGB8888	21
ETC2 RGB	22
ETC2 RGBA	23
ETC2 RGB A1	24
EAC R11 Unsigned	25
EAC R11 Signed	25
EAC RG11 Unsigned	26
EAC RG11 Signed	26

If the most significant 4 bytes contain a value, the full 8 bytes are used to determine the pixel format. The least significant 4 bytes contain the channel order, each byte containing a single character, or a null character if there are fewer than four channels, e.g., {'r', 'g', 'b', 'a'} or {'r', 'g', 'b', '\0'}. The most significant 4 bytes state the bit rate for each channel in the same order, each byte containing a single 8bit unsigned integer value, or zero if there are fewer than four channels, e.g., {8, 8, 8, 8} or {5, 6, 5, 0}.

2.1.4. Colour Space

'Colour Space' is a 32bit unsigned integer that specifies which colour space the texture data is in. The two valid values are listed in Table 4.

Table 4. Colour space

Colour Space	Value	Description
Linear RGB	0	Texture data is in the Linear RGB colour space
sRGB	1	Texture data is in the Standard RGB colour space

2.1.5. Channel Type

'Channel Type' is a 32bit unsigned integer that determines the data type of the colour channels within the texture data. Valid values are listed in Table 5.

Table 5. Channel type

Data Type	Value
Unsigned Byte Normalised	0
Signed Byte Normalised	1
Unsigned Byte	2
Signed Byte	3
Unsigned Short Normalised	4
Signed Short Normalised	5
Unsigned Short	6
Signed Short	7
Unsigned Integer Normalised	8
Signed Integer Normalised	9
Unsigned Integer	10
Signed Integer	11
Float	12

2.1.6. Height

'Height' is a 32bit unsigned integer representing the height of the texture stored in the texture data, in pixels.

2.1.7. Width

'Width' is a 32bit unsigned integer representing the width of the texture stored in the texture data, in pixels.

2.1.8. Depth

'Depth' is a 32bit unsigned integer representing the depth of the texture stored in the texture data, in pixels.

2.1.9. Num. Surfaces

'Num. Surfaces' is used for texture arrays. It is a 32bit unsigned integer representing the number of surfaces within the texture array.

2.1.10. Num. Faces

'Num. Faces' is a 32bit unsigned integer that represents the number of faces in a cube map.

2.1.11. MIP-Map Count

'MIP-Map Count' is a 32bit unsigned integer representing the number of MIP-Map levels present including the top level. A value of one, therefore, means that only the top level texture exists.

2.1.12. Meta Data Size

'Meta Data Size' is a 32bit unsigned integer representing the total size (in bytes) of all the metadata following the header.

2.2. Metadata Format

Table 6 lists the details of the metadata format. Metadata allows for the creator of a PVR to store custom information within the PVR file relating to the storage.

Table 6. Metadata format

Name	Size(bytes)	Data Format
FourCC	4	Four Byte Array
Key	4	Unsigned 32bit Integer
Data Size	4	Unsigned 32bit Integer
Data	Variable	Variable
Padding	8 or more, if defined	Variable

2.2.1. FourCC

'FourCC' is a four byte identifier (consisting of single byte characters or integers) whose value, combined with the value of 'Key', is used to determine how 'Data' should be handled. The values {'P', 'V', 'R', 0} to {'P', 'V', 'R', 255} (and their numerical equivalents) are reserved and must not be used except as described in this specification.

2.2.2. Key

'Key' is an unsigned 32bit integer, which, when coupled with 'FourCC' determines how 'Data' should be handled.

2.2.3. Data Size

'Data Size' is an unsigned 32bit integer representing the size of 'Data' in bytes.

2.2.4. Data

'Data' is an array of user defined information of size determined from 'Data Size' of a data type and purpose determined from the value of 'FourCC' and 'Key'.

2.2.5. Pre-Defined Metadata

The metadata elements identified in Table 7 are pre-defined by this specification.

Table 7. Pre-defined metadata

FourCC	Key	Data Size	Data Description
‘P’, ‘V’, ‘R’, 3	0	Variable	An array of integers describing the position and sizes of each texture within a texture atlas. Each sequence of four integers represents the information for a single texture within the atlas and appear in the order: 1. X Position 2. Y Position 3. Width 4. Height
‘P’, ‘V’, ‘R’, 3	1	8	Specifies that the file contains normal map information. The 8 bytes are in the form of a 32bit float representing the scale of the normal map, followed by a four character array describing the order of the channels, for example {‘x’, ‘y’, ‘z’, ‘h’}. Use of ‘h’ as the representation for a given channel denotes that the channel in question contains the original height map.
‘P’, ‘V’, ‘R’, 3	2	6	Specifies that the file contains a cube map and the order of the faces within that cube map. The 6 bytes represent a six character string. This string shows the order the cube map faces are stored in the texture data, for example ‘XxYyZz’. Uppercase letters refer to a positive axis position while lowercase refer to a negative axis position. Not all axes must be present.

‘P’, ‘V’, ‘R’, 3	3	3	Specifies the logical orientation of the texture within the texture data. This does not affect the mapping from pixels to texture coordinates. Each byte is a Boolean value representing the orientation for a single axis in the order X, Y, Z. The values are as follows:
			
			• X Axis
			  • Non-zero value = X values increase to the left
			  • Zero value = X values increase to the right
			• Y Axis
			  • Non-zero value = Y values increase upwards
			  • Zero value = Y values increase downwards
			• Z Axis
			  • Non-zero value = Z values increase outwards
			  • Zero value = Z values increase inwards

‘P’, ‘V’, ‘R’, 3	4	12	Specifies whether the texture has a border. The 12 bytes are broken down into three unsigned 32bit integers. The three integers represent the size of the border of the image, in pixels, on the X, Y and Z axes, respectively. These values are used to offset texture reads by the size of the border in order to obtain the actual texture data. It should be noted that only three border sizes are given, this means that the border size for X is applied to both the left and right of the image, Y to the top and bottom and Z to the front and back.

FourCC	Key	Data Size	Data Description
‘P’, ‘V’, ‘R’, 3	5	Variable	Specifies that this block contains padding data. The size of data varies in order to align the texture data with a convenient block boundary. The contents of data are left undefined. This block should be skipped during parsing.

3. Texture Data

The remainder of the file, after the header and metadata, is texture data. The format and size of this texture data can be found in the header (see Section 2.1).

3.1.1. Uncompressed Texture Data Structure

The uncompressed texture data is laid out 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
```

3.1.2. Compressed Texture Data Structure

All compressed data formats have a "minimum width/height" which is the lowest number of pixels that can be represented by any given region in a compressed image. See the `PVRTGetFormatMinDims()` function in PVRTools, which lists the minimum height/width dimensions for each format.

Note: The bits per pixel are listed using the `PVRTGetBitsPerPixel()` *function, which helps determine the region size.*

The compressed texture data is laid out 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 Region by aligned Depth (Based_On_PixelFormat)
                for each Region by aligned Height (Based_On_PixelFormat)
                    for each Region by aligned Width (Based On PixelFormat)
                        Byte data[Size Based On PixelFormat]
                    end
                end
            end
        end
    end
end
```

4. 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