PVRTexLib

User Manual

Filename	:	PVRTexLib.User Manual.1.0.17.External.doc
Version	:	1.0.17 External Issue (Package: POWERVR SDK REL_2.10@905358)
Issue Date	:	14 Mar 2012
Author	:	Imagination Technologies Ltd

Contents

List of Figures

Error! No table of figures entries found.

1. Introduction

1.1. Document Overview

The purpose of this document is to serve as a partial reference for the PVRTexLib libraries, and its associated PVR container format. Full header file reference information can be found in the HTML documentation found in the `'…/PVRTexLib/Documentation/…'` folder. It will also provide a number of examples of common uses of PVRTexLib with associated code samples.

1.2. Library Overview

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

- Load and save .pvr files.
- Compress and decompress many different pixel formats.
- Perform a variety of processes on decompressed pixel data.
- Provide information about a texture file loaded by the library.

The following library files are provided:

- `'PVRTexLib.dll'` – Windows DLL
- `'PVRTexLib.lib'` – Linux library file
- `'libPVRTexLib.a'` – Mac OS library file

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

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

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

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

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

`'PVRTexture.h'` is the primary header file, and including it will include all other header files required for PVRTexLib to function.

2. Installation

PVRTexLib is a series of libraries that are available as part of the PowerVR Insider SDK which can be downloaded from the PowerVR Insider website.

2.1. From Installer

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

```
 <SDK_ROOT>\Utilities\PVRTexLib\
```

2.2. From GZIP

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

```
<SDK_ROOT>\Utilities\PVRTexLib\<PLATFORM>\
```

This folder will contain the PVRTexLib libraries. The associated header files can be found in the folder:

```
<SDK_ROOT>\Utilities\PVRTexLib\
```

2.3. Accessing the Library

To access the functionality of the library within an application the library must be linked to and the header files included; additionally when developing for Windows using the PVRTexLib libraries the pre-processor define `'_WINDLL_IMPORT'` must be set.

Finally, if the resulting program is to be run on Windows, the .dll file must be present.

3. PVR Container Format

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

3.1. File Header Structure

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

u32Version

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

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

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

u32Flags

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

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

u64PixelFormat

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

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

This allows the '`u64PixelFormat`' to perform a double purpose; when '`Low`' is '`0`' '`PixelTypeID`' can be used to read an enum of compressed pixel types (note that TexLib may not support all of the pixel types within the '`EPVRTPixelFormat`' enum for every API, check '`PVRTTexture.h`' for more information):

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

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

u32ColourSpace

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

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

u32ChannelType

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

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

u32Height/Width/Depth

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

u32NumSurfaces

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

u32NumFaces

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

u32MIPMapCount

`‘u32MIPMapCount’` is the number of MIP-map sublevels present in addition to the top level: 0 means that only a top level non-MIP-mapped texture exists, 1 means the top level plus an extra MIP-map level, etc.

u32MetaDataSize

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

3.2. Meta-Data

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

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

DevFOURCC

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

u32Key

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

u32DataSize

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

Data

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

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

4. Example Code

4.1. Read and Decompress an Image

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

```
#include "PVRTexture.h"

using namespace pvrtexlib;

CPVRTString filePath = "test.pvr";

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

// Decompress cTexture to the standard RGBA8888 format.
Transcode(
               sOriginalTexture,
               PVRStandard8PixelType,
               ePVRTVarTypeUnsignedByteNorm,
               PVRTCSpacelRGB
          );
// cTexture should now be in the format RGBA8888, with each channel being of the type
// unsigned integer, in the lRGB colour space.
```

4.2. Pre-Process, Encode, and Save an Image

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

```
#include "PVRTexture.h"

using namespace pvrtexlib;

CPVRTString filePath = "test.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);

// False colour the MIP-map levels
ColourMIPMaps(cTexture);

// Compress to PVRTC 4bpp.
PixelType PVRTC4BPP (ePVRTPF_PVRTCI_4bpp_RGB);
Transcode(cTexture, PVRTC4BPP, ePVRTVarTypeUnsignedInteger, PVRTCSpacelRGB);

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

4.3. Creating an Image from a Header and Data

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

```
#include "PVRTexture.h"

using namespace pvrtexlib;

// The pixel data is a pointer called ‘pData’.
// Create the pixelType.
PixelType PVRTC4BPP (ePVRTPF_PVRTCI_4bpp_RGB);
CPVRTextureHeader header(
                              PVRTC4BPP,
                              512,
                              512,
                         );

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

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

4.4. Accessing Meta-Data

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

```
#include "PVRTexture.h"

using namespace pvrtexlib;

CPVRTString filePath = "test.pvr";

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

// Get the header
CPVRTextureHeader* pHeader = 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(pHeader.hasMetaData(aCC, aKey))
{
       // Handle the block based on ‘aCC’ and ‘aKey’
       DoSomething();
}
```

5. Related Materials

Software

- PVRTexTool
- PVR Tools

Documentation

- PVRTC & Texture Compression Usage Guide

6. Contact Details

For further support contact:

devtech@imgtec.com

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

Tel: +44 (0) 1923 260511
Fax: +44 (0) 1923 277463

Alternatively, you can use the PowerVR Insider forums:

www.imgtec.com/forum

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

www.imgtec.com