Filewrap

User Manual

Filename	:	Filewrap.User Manual
Version	:	PowerVR SDK REL_3.5@3523383a External Issue
Issue Date	:	17 Apr 2015
Author	:	Imagination Technologies Limited

Contents

1. Introduction

1.1. Document Overview

The purpose of this document is to serve as a complete user manual for Filewrap. It includes installation instructions, a guide to the functionality of the application, and a complete listing of all command-line options.

1.2. Software Overview

Filewrap is a tool used to convert a resource file of an application into a source/header file that can be integrated into that same application. This allows executables and resources to be combined into a single file for distribution or obfuscation purposes. It can also be used to create a memory file system for instances where no file system is present.

2. Installation

Filewrap is a standalone executable that is available as part of the PowerVR Graphics SDK which can be downloaded from the PowerVR Insider website.

2.1.1. From Installer

Filewrap can be installed from the PowerVR SDK installer:

1. Navigate to the PowerVR Insider webpage (powervrinsider.com) and download the SDK.
2. Launch the installer and follow the on-screen instructions. Ensure that at least one SDK is selected.
3. Once the SDK has been successfully installed, Filewrap will be available in the `SDK` folder.

2.1.2. From GZIP

Download the PowerVR Graphics SDK. Unzip the `.tar.gz` file, and then un-tar the TAR file. From the ensuing folder, browse to the location identified next. This folder will contain the Filewrap executable.

```
<InstallDir>\SDK_<version>\Utilities\Filewrap\<PLATFORM>\
```

3. Usage Instructions

3.1. Wrapping a File

Wrapping a file is performed as follows:

```
Filewrap –o OutFile.cpp inputFile1 inputFile2 inputFile3
```

This example shows three input files being wrapped. Multiple input files can be wrapped and output to a single output file safely, as loading is based on the original file name as opposed to the output file name. As these original file names serve as the index in the memory file system, it is important that they are unique.

3.2. Command-Line Options

Table 1 lists the command-line options that can be used for file wrapping.

Table 1. Command-line options

Option	Description
`-o[a] outFile`	Outputs [by appending] to `outFile`, e.g., `-oa OutFile.cpp InputFile.txt` will wrap `InputFile.txt` and append it to `OutFile.cpp`.
`-b`	Do not append `0` byte (for binary data only).
`-s`	String literal mode.
`-h`	Create header file (does not register file in the memory file system).
`-be`	Use big endian mode (default is little endian).

3.3. Compile Time Wrapping

The wrapping of files can be automated at compile time using makefiles or custom build steps, e.g., in Visual Studio. An example of this is captured in the following steps:

1. For a given project, add the desired file that needs to be wrapped.
2. Select the file in the `solution explorer`.
3. Right-click the file and select `properties`.
4. Click `Custom Built Tool`.
5. Set `Command-Line` to:

```
Filewrap.exe –o "$(InputDir)Outputfile.cpp" "$(InputPath)"
```

6. Set `Outputs` to:

```
$(InputDir)Outputfile.cpp
```

This process can also be automated by editing the Visual Studio solution file with a script, thereby ensuring that it does not need to be done for each individual file that is to be wrapped.

3.4. Loading a File

All file read access in the PVRTools library uses the `CPVRTResourceFile` class. When loading a file, `CPVRTResourceFile` first looks in the read path set by the application before checking for the file by name in the memory file system. This order of precedence allows files that have been wrapped and linked into the executable to be overridden. An example of reading a file using the `CPVRTResourceFile` class is provided next:

```
// Get and set the read path for content files

CPVRTResourceFile::SetReadPath((char*)PVRShellGet(prefReadPath));

// Get and set the load/release functions for loading external files.
// In the majority of cases the PVRShell will return NULL function pointers
// implying that nothing special is required to load external files.

CPVRTResourceFile::SetLoadReleaseFunctions(  PVRShellGet(prefLoadFileFunc),
                      PVRShellGet(prefReleaseFileFunc) );

// Set the error string, and the file to be opened

CPVRTString* const pReturnError = "";
const char* const pszSrcFile = "fileName.txt"

CPVRTResourceFile TextFile(pszSrcFile);
if (!TextFile.IsOpen())
{
        *pReturnError += CPVRTString("Failed to open file ") + pszSrcFile + "\n";
        return PVR FAIL;
}

const char* pTextData = (const char*) TextFile.DataPtr();
```

It is important to note that when reading a file from the memory file system, only the original pre-wrapping file name is used. It is thus very important that these filenames be unique even if multiple are stored within a single wrapped file. In order to access the `CPVRTResourceFile` class and for it to function correctly, the application must be linked to the PVRTools library for the target API.

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