Introduction

This document is meant to describe the differences between making plugins for the pre-Universe versions of Electric Image and Electric Image Universe.

NOTE: Plugins compatible with Electric Image 2.9 should continue to work unmodified; however, these plugins will not work on platforms other than the Macintosh.

Electric Image Universe plugins use Macintosh-style resource files (on non-Mac platforms, the resources are placed into a MacBinary file). You must use a Macintosh to create these resource files. See the section “Resources” below for more information.

Platform

You must define one of the platform symbols in order to properly compile. The valid platform symbols are __MACINTOSH__, __WIN32__ and __SUN__. One of these symbols must be defined “outside” of the source (e.g. in a prefix file, in a precompiled header, as a command line argument, or some other method.).

The header file “PluginHeadersMac.h” can be used as a prefix file for building Mac plugins in CodeWarrior. PluginHeadersMac.h is located in “Mac:Code:Libraries:Headers:.”

The Main Function

Your plugin should export a single function, called “Main.” Main should be declared like this:

```
EXPDLL long Main(unsigned long theCommand,
                 long theCommandRecSize,
                 void* theCommandRec);
```

The details of how this function is exported from your plugin varies by platform and development system.

On the Macintosh, using CodeWarrior, your Main function is exported using an “.exp” file. The export file to use is part of the SDK, and can be found in Mac:Code:Libraries:Source:. To use this export file, drag it into your CodeWarrior project, then open up your project settings. Go to the “PPC PEF” panel, and choose “Use “.exp” file” from the “Export Symbols:” popup menu.

On the PC, using Visual C++, your Main function is exported via the EXPDLL declaration placed before your function definition. (Note that on

platforms other than the PC, the EXPDLL macro expands to the empty string, and thus has no effect on the declation.)

NOTE: If you are developing your plugin using CodeWarrior on the PC, the EXPDLL will cause your Main function to be exported.

The source file containing Main should include “PluginClient.h.” This file provides the definition of EXPDLL, as well as a prototype for the Main function.

Plugin Type

Previously, the type of a plugin was encoded in its Macintosh file type. This is no longer a sufficient way to distinguish the type of a plugin, since the Windows file system does not support file types. Therefore, you must use filename extensions. The extension for model plugins is “.plm.” The extension for light flare plugins is “.plf.” You should name your plugins with these extensions even on the Macintosh. In fact, your plugin should be named the same on every platform you support (if your plugin uses different names on different platforms, it may not work correctly with Renderama).

Resources

The resources used by a plugin can be divided into two groups: platform-specific resources, and platform-independent resources.

The platform-independent resources are those actually required for the plugin’s code to work. These resources include the WINT resources you create with Interface Builder, the contents of the file “PluginUtilLibs.rsrc” (which is supplied as part of the plugin SDK), the plugin’s toolbar icon (which is displayed in the Object palette), and any custom resources (if any) you create and which your plugin loads and uses at runtime.

Platform-specific resources are those that are required by a given operating system. On the Macintosh, these resources include ‘vers,’ ‘BNDL,’ and ‘icl8.’

Previously in Electric Image plugins, both kinds of resources were present in the same file: the plugin file itself. However, with Electric Image Universe, these resources should be separated into two files. Platform-specific resources continue to be stored in the plugin file, while platform-independent resources should be stored in a separate file. This new file should use the same name as the plugin, but with a different extension. That extension is “.rsc.”

You create the .rsc file on the Macintosh, using whatever method you like. One way is to put a new build target into your plugin’s CodeWarrior project and add all platform-indendent resource files to that target. You use that target to build the plugin’s .rsc file.

The following library resource files should be included in the .rsc file of your plugin:

- FlarePlugin.rsrc or ModelPlugin.rsrc, depending on what type of plugin you are building
- PluginUtilLibs.rsrc

On Windows, the .rsc file is really a MacBinary version of the Macintosh .rsc file. One way to create the Windows version of the .rsc file is as follows:

- build the .rsc file on a Macintosh
- stuff the .rsc file on a Macintosh using DropStuff to generate a .rsc.sit file
- transfer the .rsc.sit file to a PC (make sure to use a binary transfer so no conversions occur to the file)
- use Stuffit Expander to expand the .rsc.sit into a MacBinary file

Make sure when expanding the .rsc.sit file that Stuffit Expander is set to always expand resource forks into separate MacBinary files.

When delivering your plugin, the plugin and the .rsc file should both be placed in the EI Sockets directory.

No More DLOGs And DITLs.

User interfaces must now be created with Interface Builder. Programmatic control of the user interface is provided through the Electric Image User Interface API. See the document “EI UI API” for more information.

No More 68K Support

There is no more distinguishing between a 68K and PowerPC plugins. The Macintosh version of Electric Image Universe will support only PowerPC plugins. Instead of using pluginModelPPCType and pluginFlarePPCType, you should use pluginModelType and pluginFlareType.

Deprecated Host Callbacks

The following host callbacks are now deprecated:

hostOSEvent

hostOpenResource

hostCloseResource

hostColorPicker

You should never need to use hostOSEvent. If you use hostOpenResource or hostCloseResource, please use EI_OpenResourceFile and EI_CloseResourceFile instead. If you use hostColorPicker, please use EI_PickColor or EI_PickColorWithAlpha.

These callbacks are curently supported in the Macintosh version of EI Universe, for backwards compatibility only. The Windows and Unix versions of Universe do not support these callbacks, and future Macintosh versions of Universe will not support these callbacks.