World Machine2.2 PDK

By Stephen Schmitt ©2005-2010

License Agreement

The License Agreement for the PDK is a supplement to the License Agreement within World Machine itself. You must agree to both the original terms and the additional ones below to use this PDK. Any creations that have used the PDK libraries, source headers, or examples contained herein signify implicit acceptance of this license.

SOFTWARE LICENSE ADDENDUM FOR PLUGIN DEVELOPMENT KIT

1. Terms

This Agreement is a legal agreement between you, the end user, and World Machine Software (hereinafter the "Provider"). By installing and/or using this software program (World Machine 2 Plugin Development Kit hereinafter the "Software"), you are agreeing to become bound by the terms of both this Agreement and the original agreement provided with the World Machine installation package.

2. Limited Source Code License

A limited license to the use of any and all source and library code provided within the plugin development kit, including but not limited to example projects, source file headers, and compiled libraries is provided, subject to:

a. Any of the provided source may not used with and/or compiled into any program except the World Machine executable program.
b. You may not attempt to circumvent the registration system or any limitations on the Basic Edition of World Machine.
c. You may not, in whole or in part, copy, reproduce, translate, reverse engineer, derive source code, modify, disassemble, decompile, create derivative works based on World Machine without the prior consent, in writing, of the Provider.

Creating a Plugin

This help file is a skeletal, bare-bones explanation of creating plugins for World Machine.

Consult the PDK forum on the official World Machine forums for more information!

http://forum.world-machine.com/

Most things should be self explanatory through examining the provided example project, which consists of 4 devices:

SimpleGenerator, SimpleOutput, SimpleWorldEffect, and Inverter.

A brief explanation of several points follows below:

A Simplified Model of World Machine's Plugin Loading:

On startup, World Machine iterates through all .DLL files in the plugins/devices/ folder. Your .DLL must implement the required exported functions shown below for World Machine to load it.

A DLL can contain 1 or more plugins. The plugins are enumerated by the GetDevPlugin() function; each time the GetDevPlugin() function is called by the WM Core, it will check the return value to determine if there are any more plugins to be found. Return false when all plugins have been enumerated.

The required exported functions are:

```
int  GetHeaderVersion();
void SetupGlobalAccess(WMGlobalStruc *global);
bool GetDevPlugin(const int i, DevPluginStruc *result);
```

For implementation details of what you need to do with these, consult the DLLExample.cpp file.

Toolbar Bitmaps

Each device can include a 32x32 bitmap to be added to the parts toolbar. A plugin DLL contains a single bitmap image that contains each individual device's image stacked side-by-side, ex: a 4-device DLL would have a 128x32 bitmap image.

Toolbar bitmap graphics for your plugin are handled by including a bitmap resource named **"IDB_TOOLBAR_GFX"** (quotes included) in the DLL. The framework will look for this graphic during the DLL load process.

If you do not include a toolbar bitmap, or a bitmap not large enough to supply all of the devices, WM will supply its own default icons for your device(s).

In the /resources folder of the PDK are the basic 32x32 icon backgrounds you can use to match the rest of the WM icons.

Data Packets (Heightfields, Bitmaps, etc)

Heightfields should be acquired by calling the global retrieve function

```
	GetPacket(WMP_hfield);
```

This has been convenienced by a C macro to be simply: GetNewHF();

Heightfields are Reference Counted. If your device is keeping a pointer to a heightfield, make sure you call Heightfield->AddRef(). When you are done with a heightfield, You should release your reference by calling Release().

For more information on the methods available in the Heightfield class, consult the Heightfield.h file.

Bitmaps are handled identically, except they are of type `WMP_rgb`. You can retrieve a new Bitmap by calling GetNewBMP();

Parameters

You can create Parameters for your Device by using the Generic Parameter system. Basic use of the Parameter system is quite straightforward; consult the examples for usage.

Ports

The ports available for Input/Output in your device are governed by the SetLinks and related functions.

You may flag a port as auxillary (not needed to build: shows up in a different color in the GUI) by calling Setflag(1) for the given port. For example:

```
in(0)->Setflag(1); // Sets the first port for the device to be optional.
```

To provide names for your Input & Output ports, you may do either:

In your device's constructor, name each port like so:

```
	in(0)->SetName("Primary Input");
	in(1)->SetName("Doodad mask Input");
	out(0)->SetName("Primary Output");
```

Or you may override the appropriate function in the Device class:

```
	virtual char *GetInputName(int slot);
```

```
virtual char *GetOutputName(int slot);
```

and for the given slot you want to rename, simply return a pointer to a static C-string. For any port numbers you do not handle, let the Base class implementation handle it.

Scale/Location information and Explorer Mode

A well written device should produce consistent results that respect the world's aspect ratio, panning and scaling.

The easiest way to do this is to use the pair of functions:

```
CoordF TransformCoord_HFToWorld(SizeData &sd, CoordF &device_loc);
CoordF TransformCoord_WorldToHF(SizeData &sd, CoordF &world_loc);
```

They translate a floating point coordinate pair to and from world/heightfield space, using the Sizing data provided by any heightfield.

See the example plugin SimpleWorldEffect for an example of how this is implemented!

Compiler Compatibility

To compile your plugin for World Machine 2, you must be using the MSVC 2005 compiler.