Guide to DVD Chapter 5 Examples: *Richard Dobson*

Audio Plug-ins in C++

In this chapter we move (mostly) away from the command-line into IDE-based development – *Xcode* on OS X and *Visual Studio Express* (v 2005 or later) for Windows. The reason for this (as discussed in the chapter) is that the test and debugging requirements are very different. A plug-in has to run within a host application (most likely GUI-based, not least to support plug-ins with custom editors); these environments support launching an external application in order to debug a shared library.

However, the chapter does begin with command line programs – try to resist the temptation to go directly to the plug-in project!

Command Line Programs: *device*, *miniplugin*, *fbtest*

Build these in the usual way at the command-line, using *make*. Neither *device* nor *miniplug-in* requires any external resources to run. From the *examples* directory:

```
cd device
make
./device

cd ../miniplugin
make
./miniplugin
```

The filter bank test program *fbtest* uses (and builds) *portsf* in the familiar way. From the *examples* directory:

```
cd fbtest
make
```

Usage:

```
fbtest infile outfile botfreq topfreq Q gain freqwarp

botfreq : lower frequency limit of filterbank: 100 <= botfreq < nyquist

topfreq : upper frequency limitm of filterbank: 0 < topfreq < nyquist

Q       : resonance factor for filters: 0 < Q < 1000
gain    : output amplitude factor: gain > 0

freqwarp: warp factor for filter frequencies: 1.0 = no warp
          warp is relative to topfreq.
          small deviations from 1.0 recommended.
```

```
deviations above 1.0 will force lower filters to cut out
topfreq and freqwarp can vary over time
Up to 24 filters are available.
NB: minimum permitted interval between adjacent filters = 1
semitone.
```

Note that only mono soundfiles are supported.

Example:

One source soundfile is provided: *sawnv.wav*
Two breakpoint files are supplied: *topfreq.brk*, *warp.brk*

```
./fbtest sawnv.wav sawfb.wav 100 topfreq.brk 100 1 warp.brk
```

As a matter of interest (see the DVD Chapter): this is indeed the program written to develop and test the filter bank before recasting it as a VST plug-in.

Harmbank

Before the example plug-in, *harmbank*, can be built, you need to setup the examples development environment for your platform. Copy the whole *examples* directory, as is, to your hard disk. Do not rearrange the structure of this directory! You will also need a MIDI keyboard or similar controller to drive the synthesizer that generates audio for the plug-in. Alternative methods include the use of modular environments supporting soundfile playback or other synthesis sources.

The VST 2.4 SDK

Download the SDK from Steinberg's 3rd-party developer website:

http://www.steinberg.net/en/company/3rd_party_developer.html

You will be required to register as a developer – there is no cost involved. The development project files supplied in this chapter require v2.4 of the SDK. The unpacked SDK (in a directory called *vstsdk2.4*) should be placed in the *examples* directory (i.e. at the same level as *harmbank*); the various VST source files will then be correctly located by the projects.

Test Hosts

A test host has been provided for each platform. In each case they have been developed by their authors expressly to support plug-in development. You are of course free to use any other host of your choice, whether a modular patching environment such as *Audiomulch* or *Bidule* (both highly recommended for this purpose), or a DAW application. Note however that for repeated build and test cycles, the speed with which the host application launches becomes an important practical consideration. Unfortunately some hosts prefer to prevent debuggers from launching

them, which makes them unsuitable for live debugging. Both the VSE and Xcode projects are supplied pre-configured to invoke their respective host application as described below.

Windows:

VSTHost from Hermann Seib.

At the time of writing the current version is 1.45. This is provided in the zip file *vsthost.zip*. Unpack this into the examples directory. Refer to the author's website for more information about the latest version and for many other resources for VST development:

http://www.hermannseib.com/english

OS X:

VSTLord from Arne Scheffler.

At the time of writing the current version is 0.3.1 (Universal Binary). Open the disk image file *vstlord031.dmg*, and copy the files to the examples directory. Note that one of these is a free synthesizer plug-in, *zr3.vst*. See Arne's website for the latest developments:

http://arne.knup.de/

His website is a resource of considerable value and importance for all VST developers.

Alternatively:

Audiomulch by Ross Bencina see http://www.audiomulch.com/.

At the time of writing, V2 has been released. Originally only available for Windows, Audiomulch is now available for OS X. Ross Bencina is one of the developers of *portaudio*.

Bidule: see http://www.plogue.com/ Available for both platforms.

Both applications are made available in time-limited demo versions. By inspecting the properties pages for *harmbank* in your IDE, you will see how to configure it for a different host application.

Driving the Plug-ins – Audio Sources

Windows:

VSTHost includes a built-in soundfile player. Looping playback is supported. Once launched, you need to open the required effect (File –> NewPlug-in), or use the button adjacent to the drop-down list control in the toolbar. If you later close *VSTHost*, leaving the plug-in widget in the window, the next time the application is launched, the plug-in will automatically be loaded,

ready for use. See the documentation for details. Of most interest will probably be the ability to create a chain of effects (e.g. VST instrument and plug-ins).

OS X:

The *VSTLord* distribution includes a VST instrument (VSTi) modeling a drawbar organ, *zr3.vst*. There is no on-screen virtual keyboard, so an external controller such as a physical keyboard is required. This instrument is suitable for testing *harmbank*, but you can of course load any other VSTi of your choice. You must first select a VSTi via *File –>New Synth*. Once the instrument is loaded you can then load an effect via *VST –>Add Effect*.

IDE Project Files

Project files for both Windows and OS X are found in the *examples/harmbank* directory:

Windows: *harmbank.sln* (with *harmbank.vcproj*)

OS X: *harmbank.xcodeproj*

Building the Examples

If all installation steps have been completed as described, double-click the appropriate project file to open the project in the IDE. All the source files (including the VST SDK files) should be visible in the project. Make sure you are setup to build the Debug version of the plug-in. You should now be able to build *harmbank* immediately. Assuming the build is successful, use the IDE to launch the test host (*VSE*: F5; *Xcode*: click the Debug widget). Note that the Xcode project is configured to create the plug-in as a Universal Binary. Thus it reports two build cycles – one for each platform.

VSTHost:

Open any suitable soundfile (a sustained sound is recommended, to hear the effects of a filter bank). A short sawtooth test file, *sawnv.wav*, is available in the *fbtest* directory, though you may prefer a more musical source. Noise sources can also prove effective for test purposes. Alternatively, open your favorite VST instrument, verify it responds to MIDI input, and then load *harmbank*.

VSTLord:

This can take several seconds to load – don't panic! The debug window status bar will indicate that *gdb* has started. Follow the instructions above to load *harmbank*.