Guide to BOOK Chapter 6 Examples:

Victor Lazzarini

Time-domain Audio Programming

These examples will build on most systems, and have been tested on OS X, Linux and Windows. To build the examples you will first need a C compiler. I recommend using **gcc**, if you have a choice (on OSX and Linux you'll get it by default). On Windows **gcc** can be got by either installing **MinGW/MSYS** or **Cygwin** (www.cygwin.com).

Examples are built using **scons**, which is a handy utility for code maintenance and programming projects. It looks after re-building any files that have been modified and gets all the right components together to build the whole project. See the 'scons-guide' document for more details. If you don't have scons, you can get it from www.scons.org, and you will also need Python, which you can get from www.python.org. Both programs are easy to install.

You will also need to install **libsndfile**, if you do not have it already. In fact, **scons** will look for it and tell you if you don't have it. Then you can get it from www.mega-nerd.com/libsndfile. It's all very simple to build (if you need to) and install, all the instructions for your system are provided.

For the VST plugins you will need to download the VST SDK (2.4) from http://www.steinberg.net/en/company/3rd_party_developer.html and drop the unzipped directory inside the examples directory.

Once your system is ready, all you need to do is type the following at your shell/terminal (`$` is the prompt), and all examples will be built.

```
$ scons
```

You can also build each program example separately just by invoking the **g++** command directly from the shell (for VST plugins please see below):

```
$ g++  -o <prgname>  <sources>  -I.  -I/usr/local/include
                               -L/usr/local/lib -lsndfile
```

where `<prgname>` is the name of the program and `<sources>` are the required source files (see below). The `-I` flag tells gcc to look for extra header files in the current directory and /usr/local/include and the `-L` tells it to do the same for libraries. The required library is then supplied with the `-l` flag.

All examples display a `USAGE` message if run without any arguments, for example:

```
$ oscillator
```

Oscillator

This is a simple example demonstrating table lookup oscillators in audio and control functions, producing a vibrato sinusoidal sound. The required sources are: *main1.cpp*, *oscil.cpp*, *soundio.cpp* and *ftables.cpp*.

Envelope

This demonstrates the use of envelopes for amplitude and frequency. Its source files are: *main2.cpp*, *envel.cpp*, *oscil.cpp*, *soundio.cpp* and *ftables.cpp*.

Filter

This next example is a demonstration of the use of a filter in a subtractive synthesis example, producing an arpeggio of harmonics by sweeping the center frequency of the filter. The required sources are: *main3.cpp*, *envel.cpp*, *filter.cpp*, *oscil.cpp*, *soundio.cpp* and *ftables.cpp*.

Delay

This example shows a variable delay in action, implementing a flanging effect. An envelope is used to vary the delay time from a minimum to the user-defined maximum. Source files are: *main4.cpp*, *envel.cpp*, *filter.cpp*, *delay.cpp* and *soundio.cpp*.

Conv

The final program example implements time-domain convolution of an input file and an impulse response file. Required sources are: *main5.cpp*, *delay.cpp* and *soundio.cpp*.

MyPlug

The VST plugin example implements a delay-line pitch-shifter with feedback. To build it from the command line you can use the following g++ commands:

Linux

```
g++ -shared -o myplug.so myplug.cpp
      vstsdk2.4/public.sdk/source/vst2.x/audioeffect.cpp
      vstsdk2.4/public.sdk/source/vst2.x/audioeffectx.cpp
      vstsdk2.4/public.sdk/source/vst2.x/vstplugmain.cpp
      -D__cdecl=""
      -I vstsdk2.4/public.sdk/source/vst2.x/ -I vstsdk2.4
```

Windows

```
g++ -shared -o myplug.dll myplug.cpp
      vstsdk2.4/public.sdk/source/vst2.x/audioeffect.cpp
      vstsdk2.4/public.sdk/source/vst2.x/audioeffectx.cpp
      vstsdk2.4/public.sdk/source/vst2.x/vstplugmain.cpp
      -I vstsdk2.4/public.sdk/source/vst2.x/ -I vstsdk2.4
```

OS X

```
g++ -bundle -o myplug myplug.cpp
      vstsdk2.4/public.sdk/source/vst2.x/audioeffect.cpp
      vstsdk2.4/public.sdk/source/vst2.x/audioeffectx.cpp
      vstsdk2.4/public.sdk/source/vst2.x/vstplugmain.cpp
      -I vstsdk2.4/public.sdk/source/vst2.x/ -I vstsdk2.4
```

On OS X we have to drop the dynamic module (bundle) inside a certain directory structure (also called a bundle), together with a couple of files. The bundle is already present in these examples, so all you need to do is to copy the relevant files:

```
cp myplug myplug.vst/Contents/MacOS
cp info.plist myplug.vst/Contents/
```