Guide to DVD Chapter 41 Examples: *Richard Boulanger, Tim Lukens, and Max Mathews*

Wrapping It Up with Qt: Adding a GUI

Prerequisites & Known Issues

Most of these examples require the PortMIDI library, the PortAudio library, or both to run.

In the *Examples/Required_Libraries* folder, we have included pre-compiled library and header files for both. They were compiled under OS X 10.5, but should run fine on any Intel-based Mac running 10.4 or 10.5.

NOTE: If you are running OS X 10.6, PortAudio should still run fine, but on July 20, 2010, there are issues with PortMIDI and Snow Leopard. You will recompile and install PortMIDI following the instructions on this page: http://github.com/halfbyte/portmidi

Installing the Pre-Built Libraries on OS X 10.4 and 10.5

1. Launch the Terminal Application

2. In the Terminal, `cd` to the *Examples/Required_Libraries* directory and type the following: (note: the $ sign is the command-line prompt.)

```
$ sudo cp libportaudio.dylib /usr/local/lib
$ sudo cp libportmidi.dylib /usr/local/lib
$ sudo cp portaudio.h /usr/include
$ sudo cp portmidi.h /usr/include
$ sudo cp porttime.h /usr/include
```

Installing the Libraries on OS X 10.6 or Other Operating Systems

If you are running any OS other than OS X 10.4 or 10.5, you will need to manually download and install PortAudio and PortMIDI from http://portmedia.sourceforge.net/

Installing Qt Creator

Qt Creator is also required to run these examples. All of the coding done in this chapter was done inside of Qt Creator which can be downloaded from Nokia's page http://qt.nokia.com/products/

Contents

In the printed text we discuss the MIDI_Monitor application, but in this examples folder we also present Algo_Comp_Csound, RingMod, Module, and Radio_Baton.

MIDI_Monitor

To compile: Open the *MIDI_Monitor.pro* file in Qt Creator and then, from the Build menu, chose "Build Project MIDI_Monitor" (Command-B) followed by the Run Command (Command-R). Note that this command is also found under the Build menu.

The details of *MIDI_Monitor* are covered in the chapter. This program requires only the PortMIDI library. It allows the user to select a MIDI input and output device. It then prints all incoming messages to the screen. If the THRU check-box is checked, it will duplicate all incoming messages and send them out to the output device.

AlgoComp

To compile: Open the *AlgoComp.pro* file in Qt Creator and then, from the Build menu, chose "Build Project AlgoComp" (Command-B) followed by the Run Command (Command-R).

AlgoComp is an algorithmic composer for Csound. It gives the user several different options that they can set to generate a *.csd* file that can, in turn, be run or rendered with Csound (or QuteCsound). To take full advantage of this program, Csound must be installed. Csound can be downloaded at http://csounds.com/

Note: When the *.csd* file is generated, it is automatically saved it in the same directory as the executable. But, if you run the program from inside of Qt Creator, it behaves oddly. If you are on OS X and run the program from inside Qt Creator, to access the .csd file you will need to locate the AlgoComp app in the Finder, right-click, click Show Package Contents, and then move to the Contents/MacOS directory. This happens because when the program is being run from Qt Creator, it treats the binary file inside the .app bundle as the executable and therefore saves the .csd file there. However, when you run the program from the Finder, by double-clicking on the app, it treats the *.app* file as the executable and thus saves the *.csd* file in the folder just as you'd expect. Thus, in this case, it is best to test your application from the Finder.

RingMod

To compile: Open the *RingMod.pro* file in Qt Creator, and then, from the Build menu, chose "Build Project RingMod" (Command-B) followed by the Run Command (Command-R).

RingMod is a ring modulator that processed the computer's "default" input device. This program is adapted from Jonathan Bailey's *HelloOsc* program from Richard Boulanger and Jonathan Bailey's DVD chapter. The user can select several different waveforms and change the frequency to modify the ring modulation. This program only requires the PortAudio library.

Module

To compile: Open the *Module.pro* file in Qt Creator, and then, from the Build menu, chose "Build Project Module" (Command-B) followed by the Run Command (Command-R).

Module is a modular synthesizer written for DVD Chapter 39: *Real-time MIDI Control and Audio Processing in C*. This program allows the user to change the waveshape, the frequency, the attack and decay times, and to control automatic note triggering. Only two DSP modules are being used in the program, but many more have been included in the *module.c* file. This is in hopes that you, the programmer, will take advantage of patching them together to make different types of sounds, as well as adding more widgets to the GUI and thus give the user more control over the synthesizer. This program requires only the PortAudio Library.

Radio_Baton

To compile: Open the *Radio_Baton_2.pro* file in Qt Creator, and then, from the Build menu, chose "Build Project Radio_Baton_2" (Command-B) followed by the Run Command (Command-R).

The Radio Baton is a wireless 3D MIDI controller created by Max Mathews, the father of computer music. The Baton comes with a piece of software called *Conduct* – an "expressive" ASCII-based sequencer, that allows the user to notate and perform their own or anyone's music using text files. Once "compiled", these text-files can be "conducted" and their volume and rhythms adjusted in real-time. This project "wraps" the *Conduct* program in a GUI as well as adding a few new features such as: *Matrix Mode* and *Raw DATA Mode*.

The Radio Baton GUI is the 2010 Berklee College of Music, Electronic Production and Design, Bachelor's Thesis Project of Tim Lukens, and the software is still in beta; but will be getting regular updates (which can be found at http://csounds.com/apb). This rather advanced, and not particularly stable, example has been included for the curious programmer who wishes to study the *Conduct* program and see how one might go about wrapping code of this magnitude and complexity.

A Final Important Note: Readying an Application for Deployment

Mac OS

Let's say you have just finished up an awesome Qt GUI application and you want to share it with your friends or release it online to the public. There are a few simple steps you need to follow in order to "bundle up" your application. Since Qt programs need to have certain Qt frameworks included with them, as well as any libraries that are being dynamically linked by your code, you must use a command-line utility that comes with Qt to bundle it all into a working *.app* file. To bundle an app on OS X, follow these steps:

1. Open the *Terminal*

2. `cd` to the directory where your compiled application resides
3. Type “macdeployqt project.app” where project.app is the name of your application

It is important to keep in mind that if your program is called “Module” it will actually be called “Module.app” and must be addressed as such in Terminal.

Windows

Deploying a Qt application on Windows is slightly different. For this OS, one can find the instructions at the following URL: http://doc.trolltech.com/4.6/deployment-windows.html