Developing Audio Plugins with Cabbage and Csound

Rory Walsh

Dundalk Institute of Technology

Overview

- This talk describes a novel new approach to developing audio plugins with Csound. It discusses the basics of the system, with some exploration of the working principles.

- The talk will conclude with a look at the IDE being used for said plugin development and demonstrate a few simple examples of different Cabbage plugins.

Introduction

• In an industry dominated by commercial and closed-source software, audio plugins represent a rare opportunity for developers to extend the functionality of their favourite digital audio workstations

• Plugin developers can concentrate solely on signal processing tasks rather than low-level audio and MIDI communication.

• The latest version of Cabbage seeks to provide for the first time a truly cross-platform, multi-format Csound plugin solution. Cabbage allows users to generate plugins under three major frameworks: the Linux Native VST, Virtual Studio Technology (VST), and Apple's Audio Unit.

Background information

• The software system presented today is a an amalgamation of several previous projects which have been rewritten and partially redesigned in order to take full advantage of today's new breed of DAW.

• Before looking at the system as it stands today it's worth looking over the two main projects it derives from...

Cabbage 2008

• Cabbage was first presented to the audio community at the Linux Audio Conference in 2008.

• The framework provided Csound programmers with no low-level programming experience with a simple, albeit powerful toolkit for the development of standalone cross-platform audio software.

• The first version of Cabbage had no support for plugin development.

csLADSPA / csVST

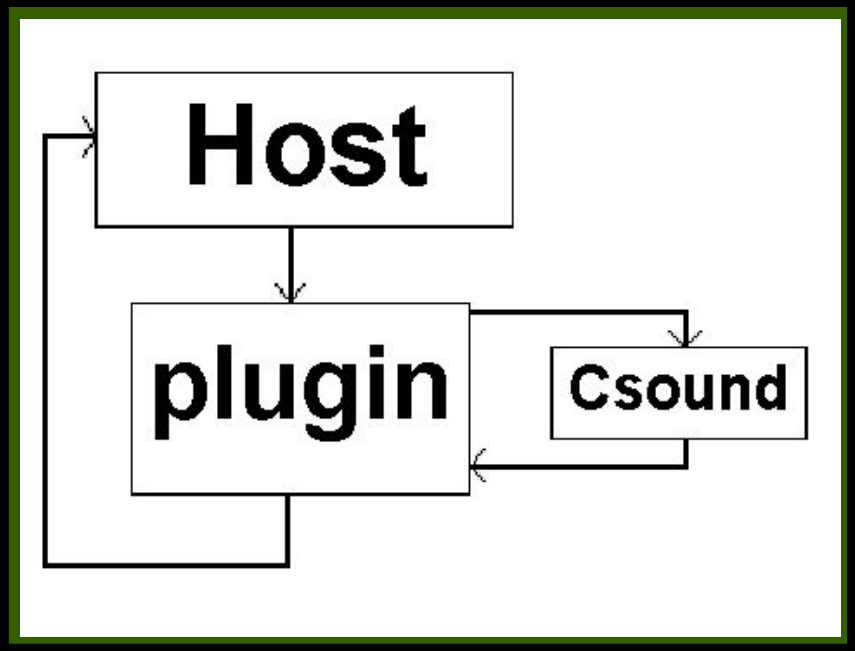
• csLADSPA and csVST are two lightweight audio plugin systems that make use of the Csound API.

• Both toolkits were developed so that musicians and composers could harness the power of Csound within a host of different DAWs.

• The concept behind the toolkits is simple and although each toolkit made use of a different SDK they both worked in the same way.

csLADSPA / csVST

- A basic model of how the plugins work is shown in below:



Cabbage 2010

• The latest version of Cabbage consolidates the aforementioned projects into one user-friendly cross-platform interface for developing audio plugins.

• By combining the GUI capabilities of earlier versions of Cabbage with the lightweight plugin systems of csLADSPA and csVST Csound users can now develop high-end professional audio plugins armed with nothing more than a rudimentary knowledge of Csound.

Technical details

• Earlier versions of Cabbage were written using the wxWidgets C++ GUI library.

• Whilst this provided a more than adequate array of widgets and other useful classes it quickly became obvious that creating plugins with wxWidgets was going to be more trouble than it was worth.

• Cabbage now usees the JUCE Class library which as well as providing an extensive array of GUI widgets provides robust classes for audio and MIDI IO.

Architecture

• The architecture of Cabbage has undergone some dramatic changes since 2008.

• Originally Cabbage produced standalone applications in which everything, including the .csd file was embedded into a binary executable that could then be distributed as a single application.

• Instead of creating a new standalone application for each instrument Cabbage is now a plugin host.

The Cabbage native host

- The Cabbage native host loads and runs Cabbage instruments/plugins from disk.

- The function of the Cabbage host is twofold.

  a) it provides a standalone player for running GUI based Csound instruments.

  b) it provides a platform for developing and testing audio plugins. Any instrument that runs in the Cabbage native host can be exported as an audio plugin.

  Cabbage also provides a virtual MIDI keyboard to help test and prototype MIDI based plugins.

Cabbage Syntax

• The Cabbage syntax has changed slightly in recent years, although in remains completely backwards compatible.

• The syntax used to create GUI controls is quite straightforward and should be provided within special xml-style tags, i.e., <Cabbage> and </Cabbage> at the top of a unified Csound file.

• Each line of Cabbage specific code relates to one GUI control only and the syntax is non case-sensitive.

Example

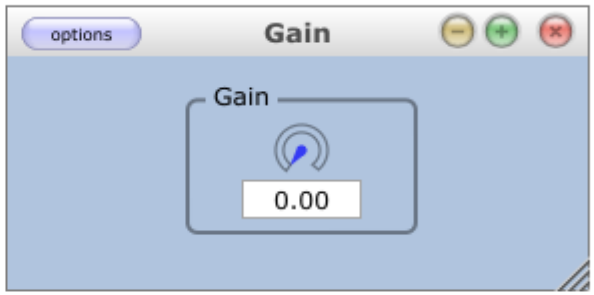
- The following example shows a basic Cabbage instrument and how it appears when loaded with the Cabbage native host

```
<Cabbage>
form caption("Gain"), size(293, 122), colour("LightSteelBlue"),
rslider size(120,80), pos(87,13), channel("gain"), caption("Gain"), min(0), max(10)
</Cabbage>
<CsoundSynthesizer>
<CsOptions>
-d -n
</CsOptions>
<CsInstruments>
; Initialize the global variables.
sr = 44100
ksmps = 32
nchnls = 2

instr 1
k1 chnget "gain"
a1 inch 1
outs (a1*k1), (a1*k1)
endin


</CsInstruments>
<CsScore>
f1 0 4096 10 1
i1 0 1000
</CsScore>
</CsoundSynthesizer>

```



Cabbage and WinXound

• The most straightforward way to start developing Cabbage plugins and instrument is with WinXound. WinXound is a free and open-source Front-End GUI Editor for Csound 5.

• Unlike other Csound frontends WinXound uses native operating system languages and libraries rather than a cross-platform toolkit.

• Another distinct feature of WinXound is that it spawns Csound as a process rather than using the Csound API.

Cabbage and WinXound

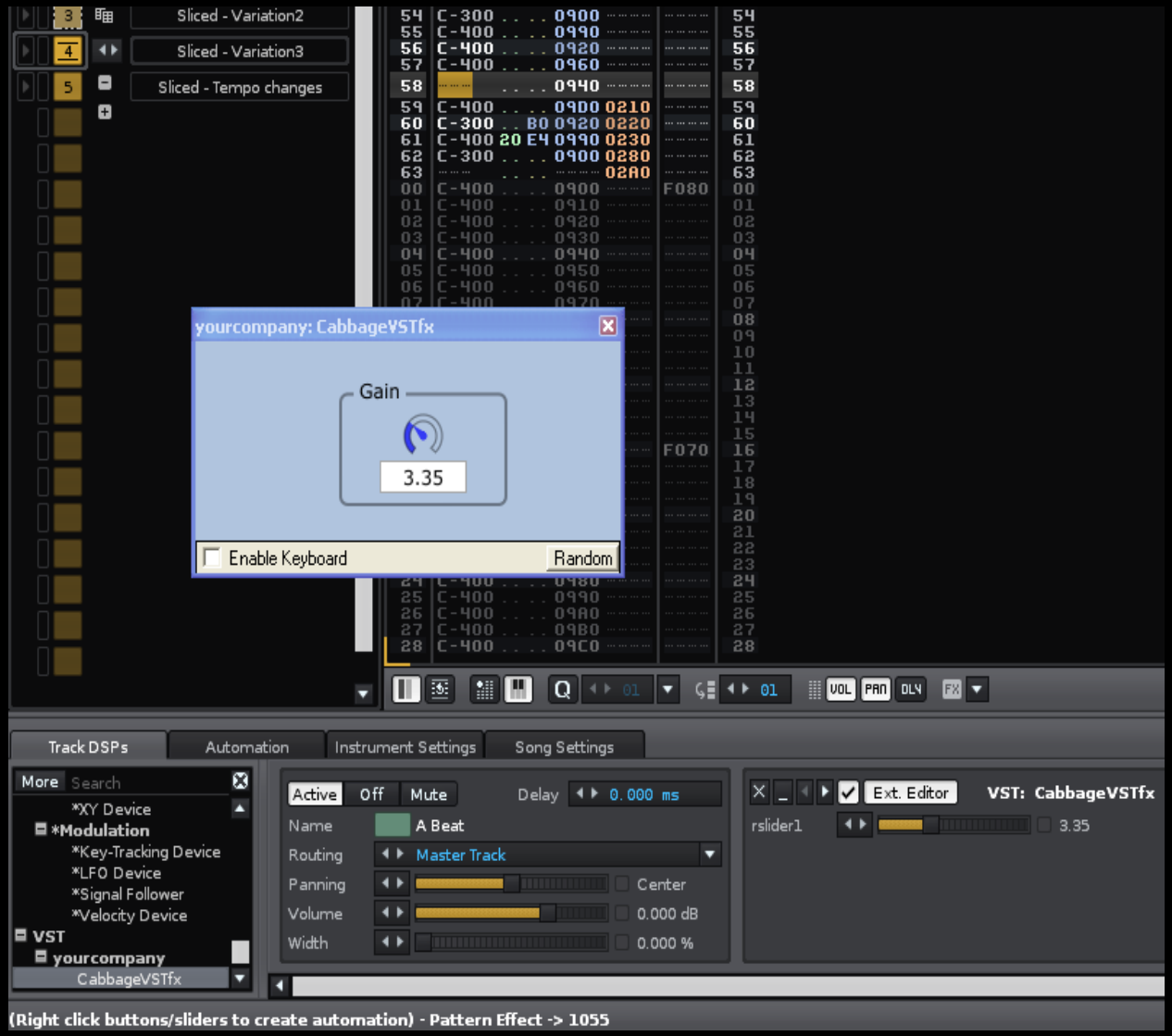
• Communication between Cabbage and WinXound is made possible through interprocess communication. Named pipes are set up and messages are sent to and from each application.

• Before getting started users must first set the Cabbage directory in WinXound. This is accessible through the Settings menu command.

• Once this has been set users can use the Cabbage tools menu to update Cabbage and export Cabbage patches directly to different plugin formats.

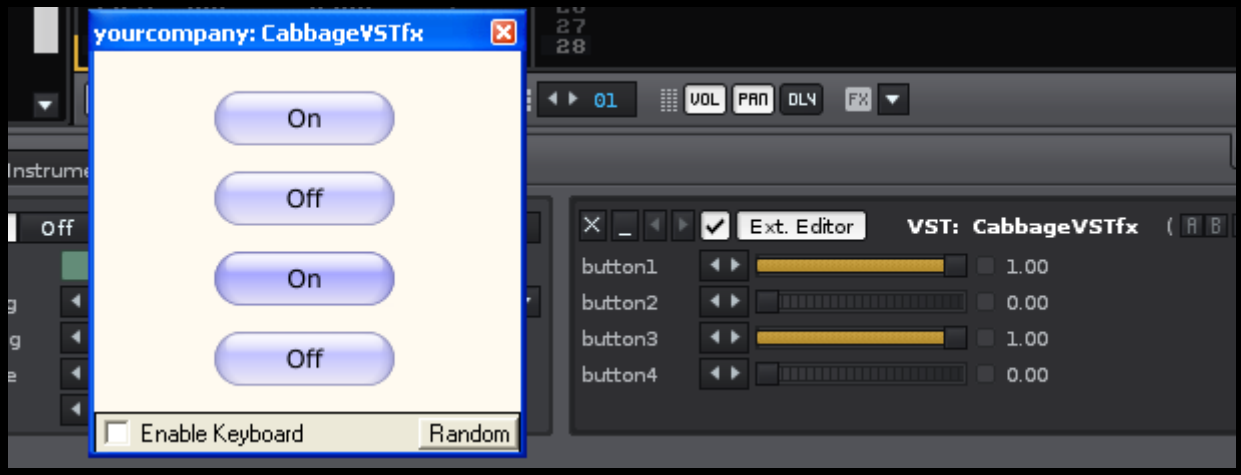
Plugin example

- The previous example showed a simple 'gain' instrument loaded into the Cabbage standalone player. In the screen-shot to the left you can see the same instrument running as a plugin in Renoise.



Native plugin parameters

• Most plugin hosts implement a native interface for displaying plugin parameters. Usually this consists of a bank of sliders as can be seen in the screen-shot below.



• In order for Cabbage plugins to avail of MIDI-learn functionality each Cabbage widgets must be mapped to a native slider. In the case above sliders will jump between 0 and 1 whenever a button is pressed.

Native plugin parameters

- In the case of widgets such as comboboxes where user have multiple choices of selection sliders range will be split to reflect the number of choices available to users.

- If for example a user create a combobox with 5 elements, the corresponding native slider will jump a fifth each time the user increments the current selection.

CsOptions – *plugin effects*

• When running Cabbage patches, be it in the native Cabbage host or in some other plugin host there are certain Csound command line flags that must be used:

```
<CsOptions>
-d -n
</CsOptions>
```

• The -d prevents Csound from trying to load any external graphics windows and the -n causes Csound to bypass writing of audio to any devices.

CsOptions – *plugin instruments*

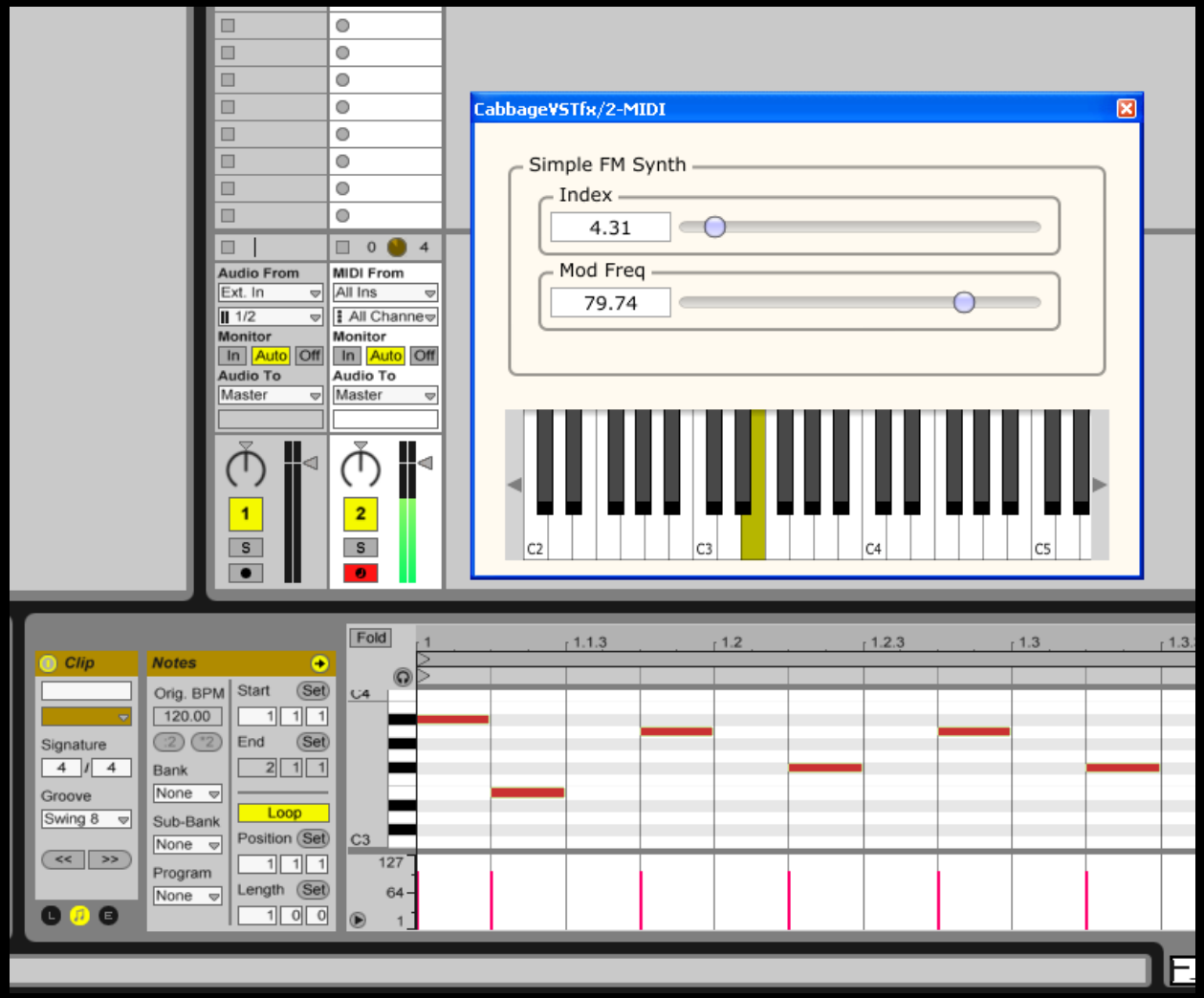
• When building plugin instruments you will need the following CsOptions

```
<CsOptions>
-d -n -+rtmidi=null -M0 --midi-key-cps=4 --midi-velocity-amp=5
</CsOptions>
```

• The –midi flags allow MIDI data to be routed directly to a Csound p-field. In the case above every instance of p4 in the Csound instrument will be replaced by the frequency of the MIDI note and every instance of p5 will be replaced by a MIDI velocity expressed in raw amplitude.

VSTi example

- The plugin on the left shows FM synth running in Ableton Live as a VSTi. The virtual MIDI keyboard is invaluable when it comes to proto-typing instruments in the native Cabbage host before exporting as plugins.



A typical user session

1. Launch WinXound and create a Cabbage .csd file

2. Press Ctrl+Alt+i to launch the Cabbage host

3. Edit the Csound code so that it interacts with Cabbage through the use of named software buses

4. Once you are happy with your Cabbage patch you can either continue to run it in the Cabbage host or export is as a plugin.

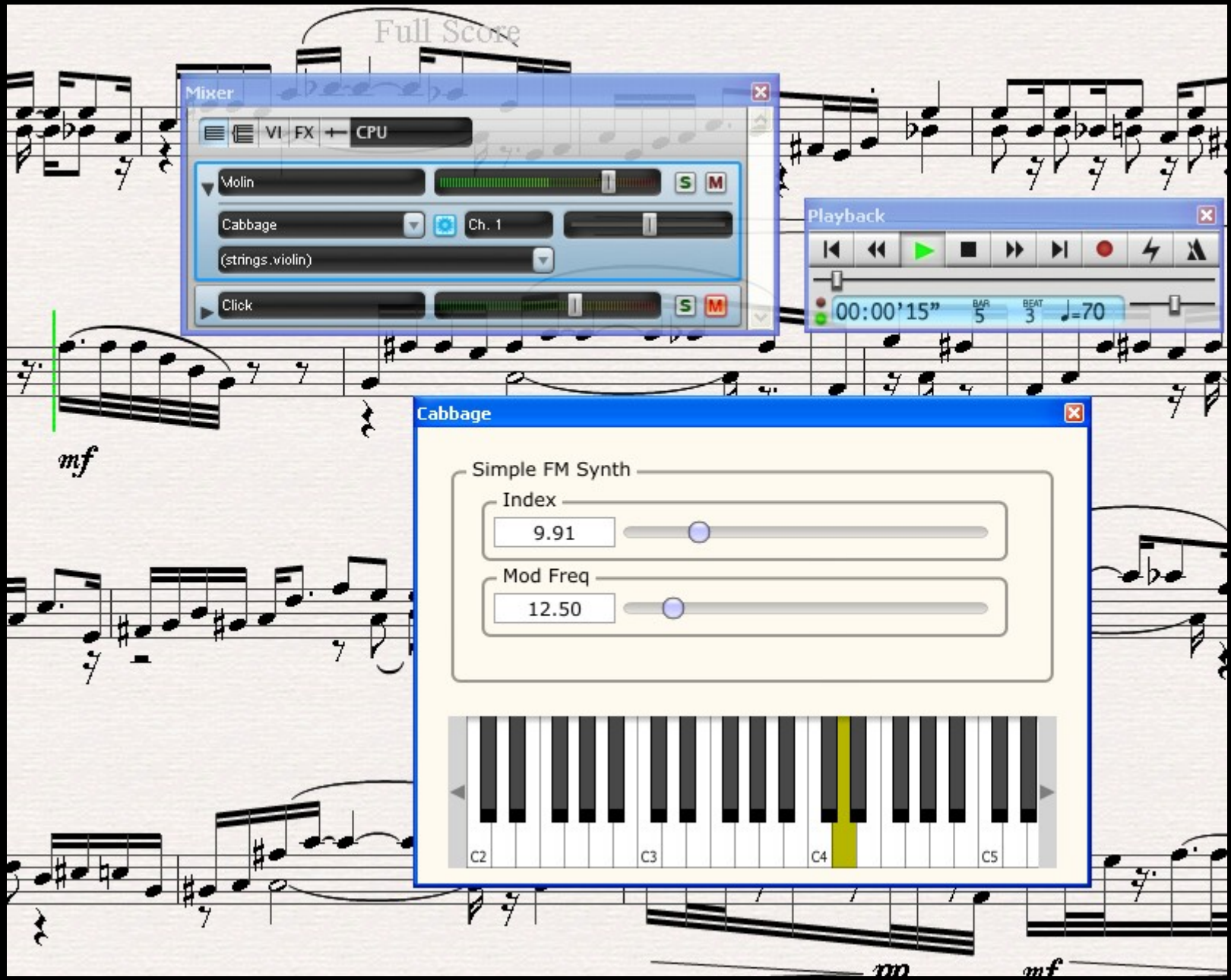
Things to note about certain DAWs

- Some hosts load plugins dynamically whilst others don't!

- Some hosts such as Live will treat a VSTi the same as a VST. So you need only export as a VST, even if technically your plugin is a synth.

- Other hosts such as Sibelius insist on VSTi.

Cabbage running in Sibelius



Examples...

Thanks to.....

I wish to express a sincere thank you to everyone on the Csound, Renoise and Juce mailing lists, in particular Mike Goggins for his work on CsoundVST and Steven Yi for his Virtual MIDI controller for Csound.

Finally I wish to thank Stefano Bonetti, the author of WinXound for his patience and expertise over the last few months. His help has been invaluable to this project.