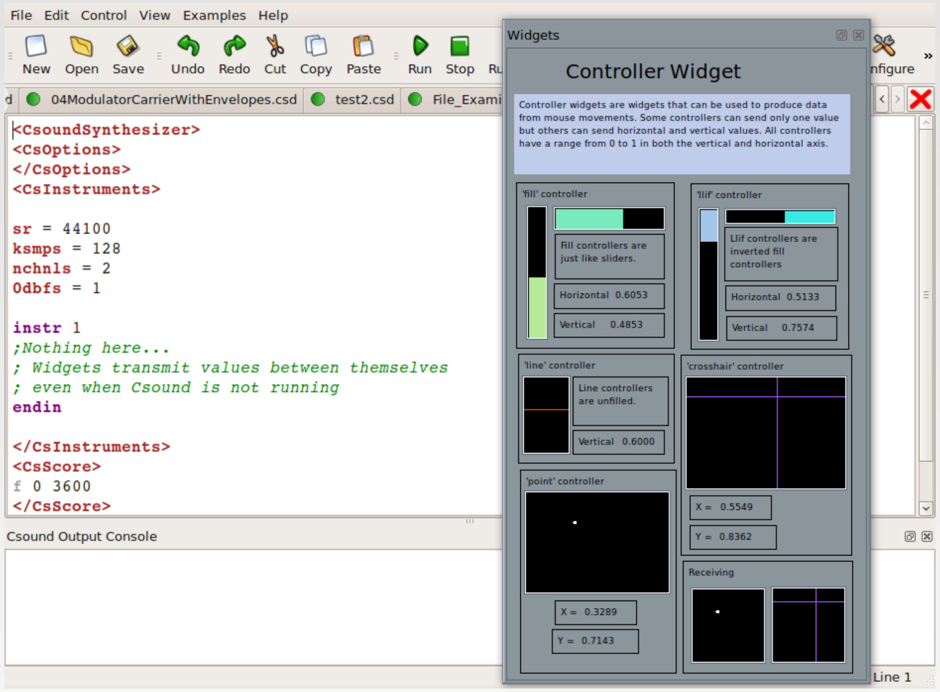
QuteCsound

QuteCsound:
A Csound Front-end

Andrés Cabrera

QuteCsound


File Edit Control View Examples Help
New Open Save Undo Redo Cut Copy Paste Run Stop
04ModulatorCarrierWithEnvelopes.csd
test2.csd
File_Exami
Widgets
Csound Output Console
Line 1

```
<CsoundSynthesizer>
<CsOptions>
</CsOptions>
<CsInstruments>

sr = 44100
ksmps = 128
nchnls = 2
0dbfs = 1

instr 1
;Nothing here...
; Widgets transmit values between themselves
; even when Csound is not running
endin

</CsInstruments>
<CsScore>
f 0 3600
</CsScore>
```

Controller Widget

Controller widgets are widgets that can be used to produce data from mouse movements. Some controllers can send only one value but others can send horizontal and vertical values. All controllers have a range from 0 to 1 in both the vertical and horizontal axis.

'fill' controller

Fill controllers are just like sliders.

Horizontal 0.6053

Vertical 0.4853

'llif' controller

Llif controllers are inverted fill controllers

Horizontal 0.5133

Vertical 0.7574

'line' controller

Line controllers are unfilled.

Vertical 0.6000

'crosshair' controller

X = 0.5549

Y = 0.8362

'point' controller

X = 0.3289

Y = 0.7143

Receiving

- Graphical front-end/editor for Csound
- Language editor
- Realtime widgets
- Integrated help

Goals

- ”Intuitive” but powerful
- Offer functionality from MacCsound (Realtime interactive widgets), and backwards compatibility with it.
- Ease usage and configuration of Csound

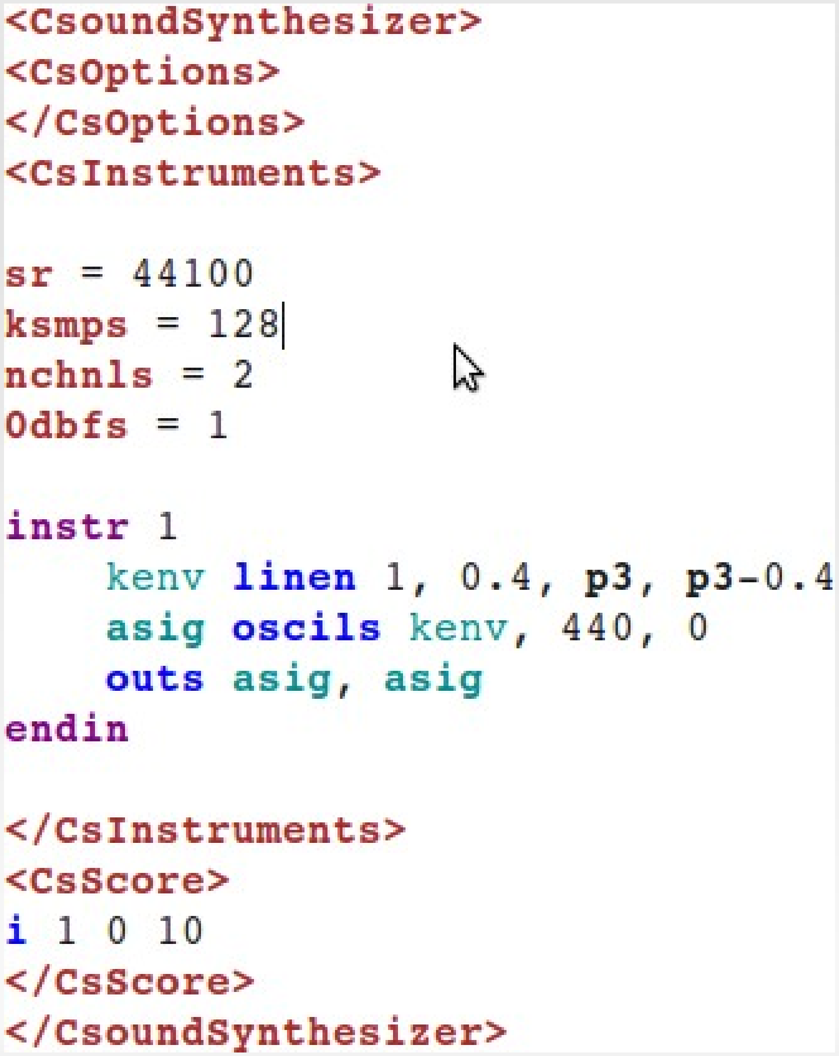
QuteCsound

- Requires Qt (from Nokia), libsndfile and Csound.
- Crossplatform
  - Windows, OS X, Linux, Solaris
- Uses Csound internally (Csound API)
- Open source (GPL or LGPL)
- In english, spanish, french, portuguese, italian and turkish
- Having its second birthday this May (2010)

Csound

- Programming language for music and sound
- First version 1984/85
- Descended from older MusicN systems
- Processing loop at a set control rate with audio signals being vectors
- Has an API in C, C++, Python, Java, Lua which allows embedding the Csound engine inside applications

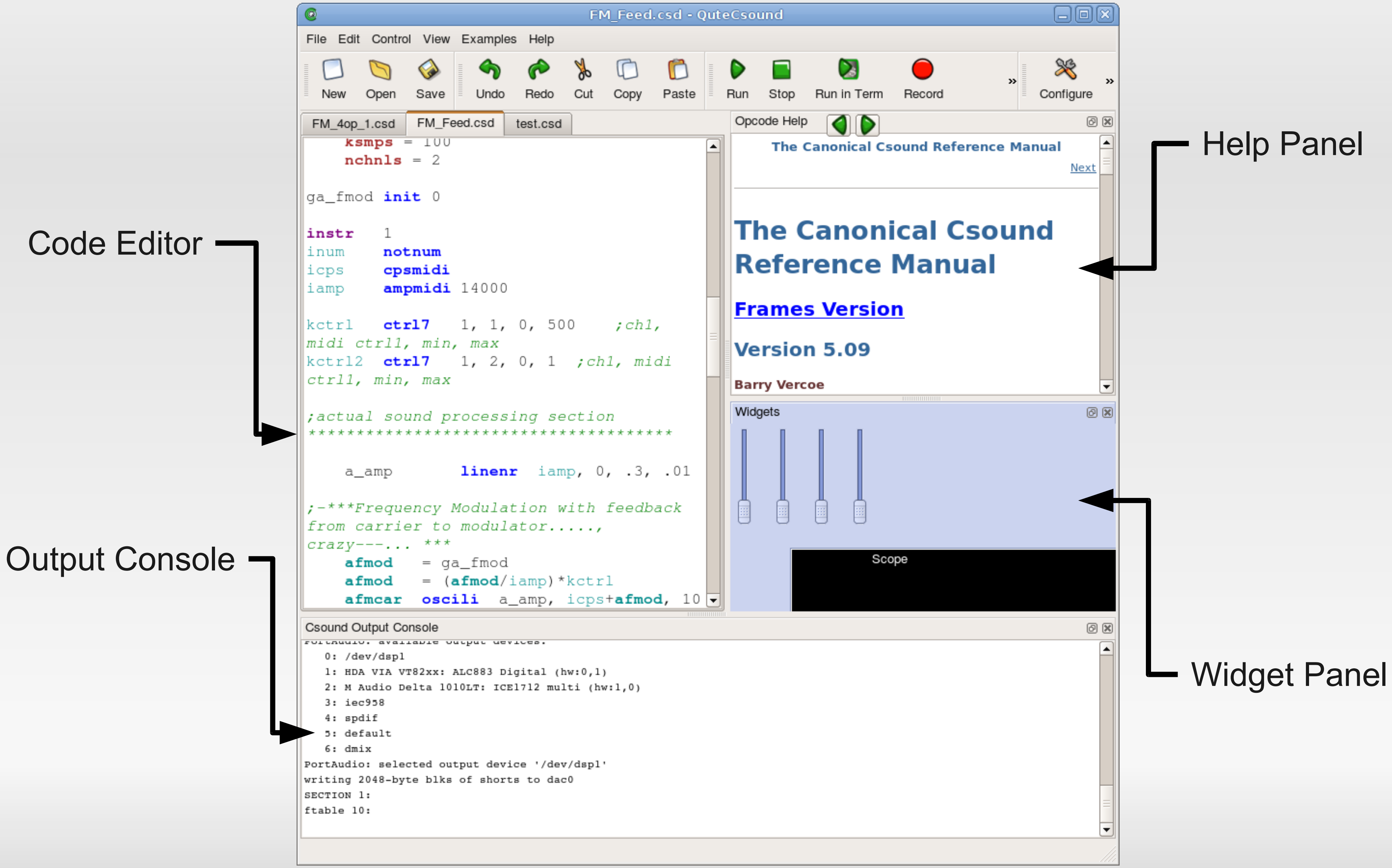
Code editor



```
<CsoundSynthesizer>
<CsOptions>
</CsOptions>
<CsInstruments>

sr = 44100
ksmps = 128
nchnls = 2
0dbfs = 1

instr 1
    kenv linen 1, 0.4, p3, p3-0.4
    asig oscils kenv, 440, 0
    outs asig, asig
endin

</CsInstruments>
<CsScore>
i 1 0 10
</CsScore>
</CsoundSynthesizer>
```

- Syntax highlighting
- Auto completion
- Python IDE as well
- Hides sections that are handled by other parts of QuteCsound (Widgets, Live Events)
- Code Inspector

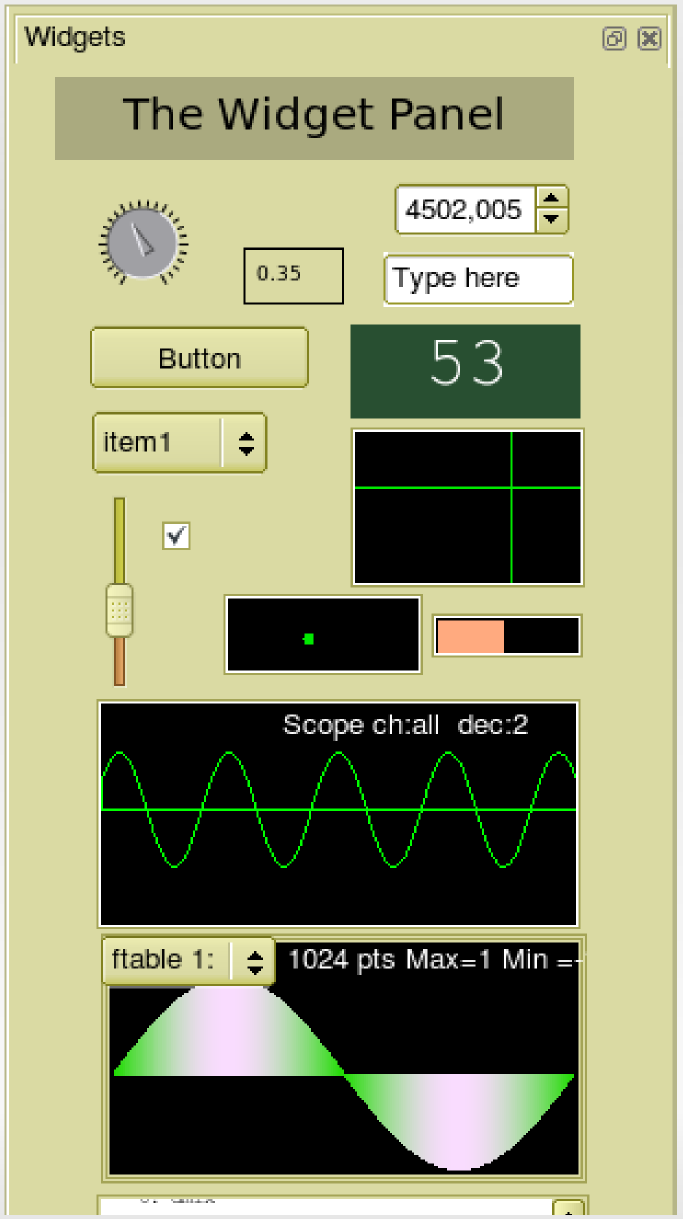
Interface


FM_Feed.csd - QuteCsound
File Edit Control View Examples Help
New Open Save Undo Redo Cut Copy Paste Run Stop Run in Term Record Configure
FM_4op_1.csd FM_Feed.csd test.csd
Opcode Help
The Canonical Csound Reference Manual
Next
The Canonical Csound Reference Manual
Frames Version
Version 5.09
Barry Vercoe
Widgets
Scope
Csound Output Console
Help Panel
Code Editor
Output Console
Widget Panel

```
    ksmps  = 100
    nchnls  = 2

ga_fmod init 0

instr    1
inum     notnum
icps     cpsmidi
iamp     ampmidi 14000

kctrl    ctrl7    1, 1, 0, 500    ;ch1,
midi ctrl1, min, max
kctrl2   ctrl7    1, 2, 0, 1  ;ch1, midi
ctrl1, min, max

;actual sound processing section
**************************************

    a_amp        linenr  iamp, 0, .3, .01

;-***Frequency Modulation with feedback
from carrier to modulator.....,
crazy---...  ***
    afmod    = ga_fmod
    afmod    = (afmod/iamp)*kctrl
    afmcar   oscili   a_amp, icps+afmod, 10
```

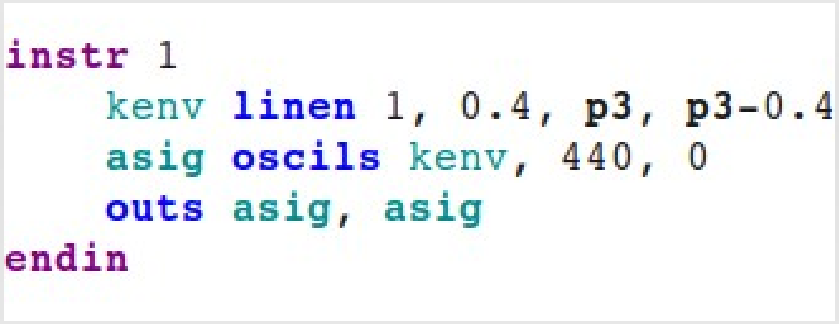
```
PortAudio: available output devices:
   0: /dev/dsp1
   1: HDA VIA VT82xx: ALC883 Digital (hw:0,1)
   2: M Audio Delta 1010LT: ICE1712 multi (hw:1,0)
   3: iec958
   4: spdif
   5: default
   6: dmix
PortAudio: selected output device '/dev/dsp1'
writing 2048-byte blks of shorts to dac0
SECTION 1:
ftable 10:
```

Widget Panel

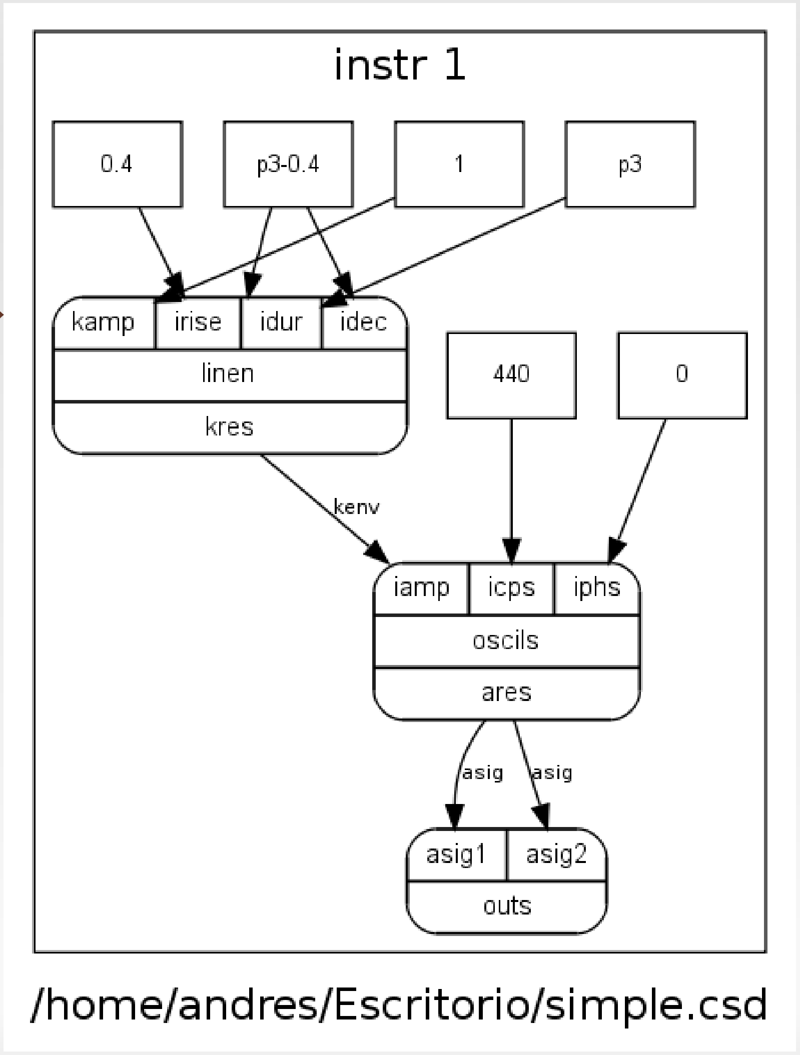


- Realtime parameter control
- Display information from Csound
- ”Soft-synth” designer
- Widgets saved in text file, but hidden from user.

Code graph

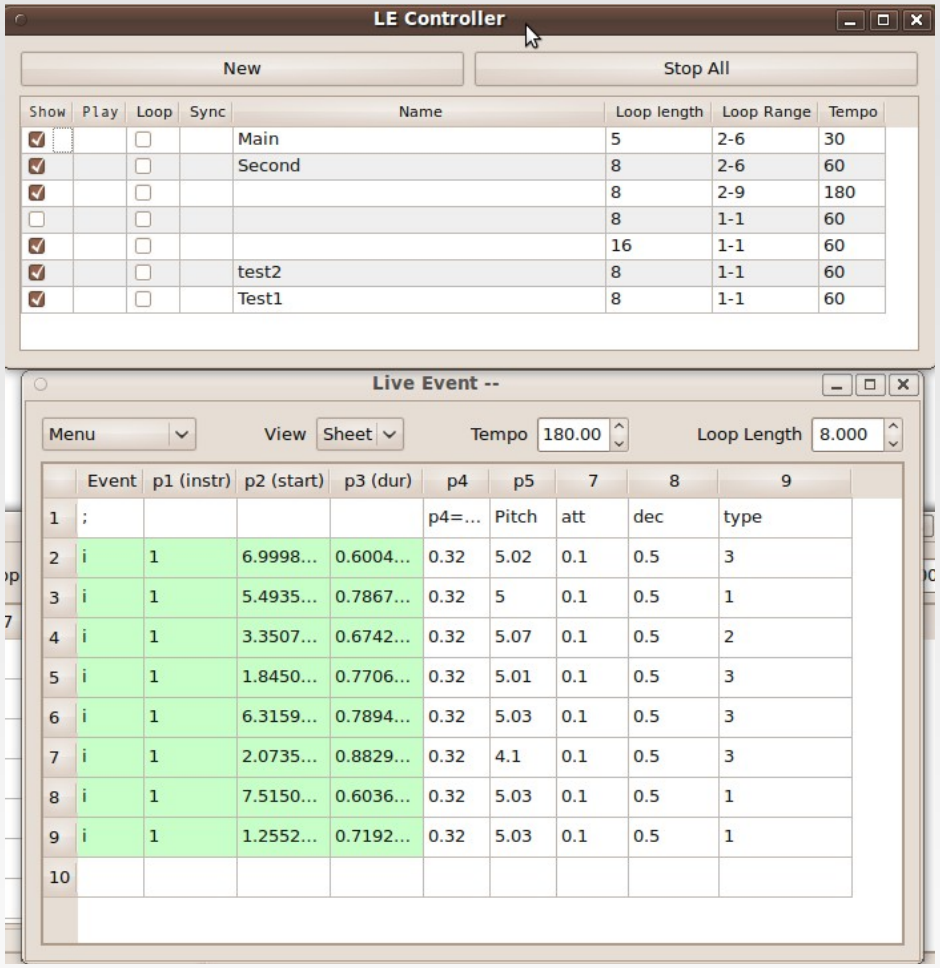


```
instr 1
    kenv linen 1, 0.4, p3, p3-0.4
    asig oscils kenv, 440, 0
    outs asig, asig
endin
```



- Automatic generation of graph for any code using graphviz
- Good for simple instruments

Live Event Panel



Show	Play	Loop	Sync	Name	Loop length	Loop Range	Tempo
✓		☐		Main	5	2-6	30
✓		☐		Second	8	2-6	60
✓		☐			8	2-9	180
☐		☐			8	1-1	60
✓		☐			16	1-1	60
✓		☐		test2	8	1-1	60
✓		☐		Test1	8	1-1	60

View: Sheet Tempo: 180.00 Loop Length: 8.000

	Event	p1 (instr)	p2 (start)	p3 (dur)	p4	p5	7	8	9
1	;				p4=...	Pitch	att	dec	type
2	i	1	6.9998...	0.6004...	0.32	5.02	0.1	0.5	3
3	i	1	5.4935...	0.7867...	0.32	5	0.1	0.5	1
4	i	1	3.3507...	0.6742...	0.32	5.07	0.1	0.5	2
5	i	1	1.8450...	0.7706...	0.32	5.01	0.1	0.5	3
6	i	1	6.3159...	0.7894...	0.32	5.03	0.1	0.5	3
7	i	1	2.0735...	0.8829...	0.32	4.1	0.1	0.5	3
8	i	1	7.5150...	0.6036...	0.32	5.03	0.1	0.5	1
9	i	1	1.2552...	0.7192...	0.32	5.03	0.1	0.5	1
10									

- Spreadsheet style editing and processing of score events
- Simple transformation functions
- Simple python API for generation and transformation of events

Future

- Refine Python scripting API for realtime interaction with widgets and live score coding (some work done, but still some to do)
- Export to standalone application and plugin (VST,LV2?) (can currently do LADSPA via csLADSPA)

Demo

Demo

Questions?