Adobe® LuaSynth

Audio Scripting with Lua

Celso Aguiar

Alex Mohr (presenter)

Adobe Systems

Audition

- **Plug-ins == DLLs**
  - Supports VST plug-in standard
  - 1 plug-in == 1 signal processing algorithm
- **Limitation: # of plug-ins shipped w/ product**
  - Engineers: C/C++ AND signal processing
  - No common definition of what an oscillator means
  - Not pushing the VST forefront
  - Focus on C/C++ community

Audition + LuaSynth

- **Another plug-in**
  - VST standard
  - 1 plug-in can be many signal processing algorithms
- **Limitation: # of Lua scripts you can write**
  - Engineers: Lua, signal processing not critical
  - Same oscillator used through-out
  - Pushes the VST to the forefront
  - Focus on script authors community

What is LuaSynth

- **Sound/Music processing APIs in Lua**
  - CLM C libs (CCRMA/Stanford package)
  - Freeverb (Jezar’s studio quality reverberation)
  - STK (Open source, physical modeling)
  - C Sound
  - You name it!
- **Lua VST GUI API**
- **Console App + VST Plug-in** (POCs)

sample Lua plugin-script

```
pitch1=param(0)*4000;    pitch2=param(1)*4000
amp = .2;    dur = 8.0;     maxIndex = 8;     srate=44100
ends = dur * srate

o1 = Osc(pitch1);     o2 = Osc(pitch2)
e1 = Env({0, 1, .2, .3, 1, 0}, amp, 0, 200, ends)
e2 = Env({0, 1, .1, .1, 1, 0}, .2, 0, 100, ends)
c1  = Comb(.312, 1155)
c2  = Comb(.212, 733)
F = FreeVerb(.8, 1, 1.5, 1.0, .2, param(2))

function processIt(nSamples)
        for i = 0, nSamples-1, 1 do
             s = env(e1) * osc(o2, maxIndex * param(3) * env(e2) * osc(o1, 0.0))
             lo,ro = freeVerb(F, s, 0, .5)
             z1 = comb(c1, lo)
             z2 = comb(c1, ro)
             output(i, z1, z2)
        end
end
```

Some Numbers

- **Complex algorithms in real-time**
  - Around 50 osc + FreVerb on a 1.8GHz PC
- **Lua versus Native C implementation**
  - ~1.7x slower
- **toLua binding only outside sample loop**
  - ~3-5 times slower

Better by Adobe.™