Fluxus: Scheme Livecoding

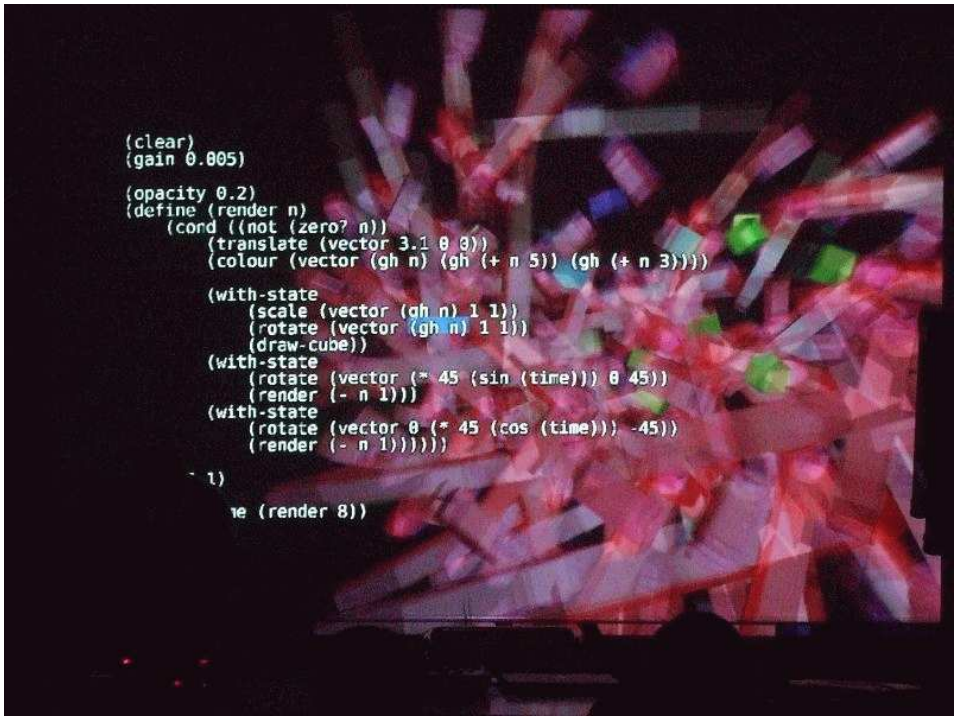
Dave Griffiths

Overview

- Fluxus introduction
- What it's built from
- Why Scheme
- Livecoding
- How it works - some examples
- Functional Reactive Programming
- Gamepad Livecoding

What is fluxus?

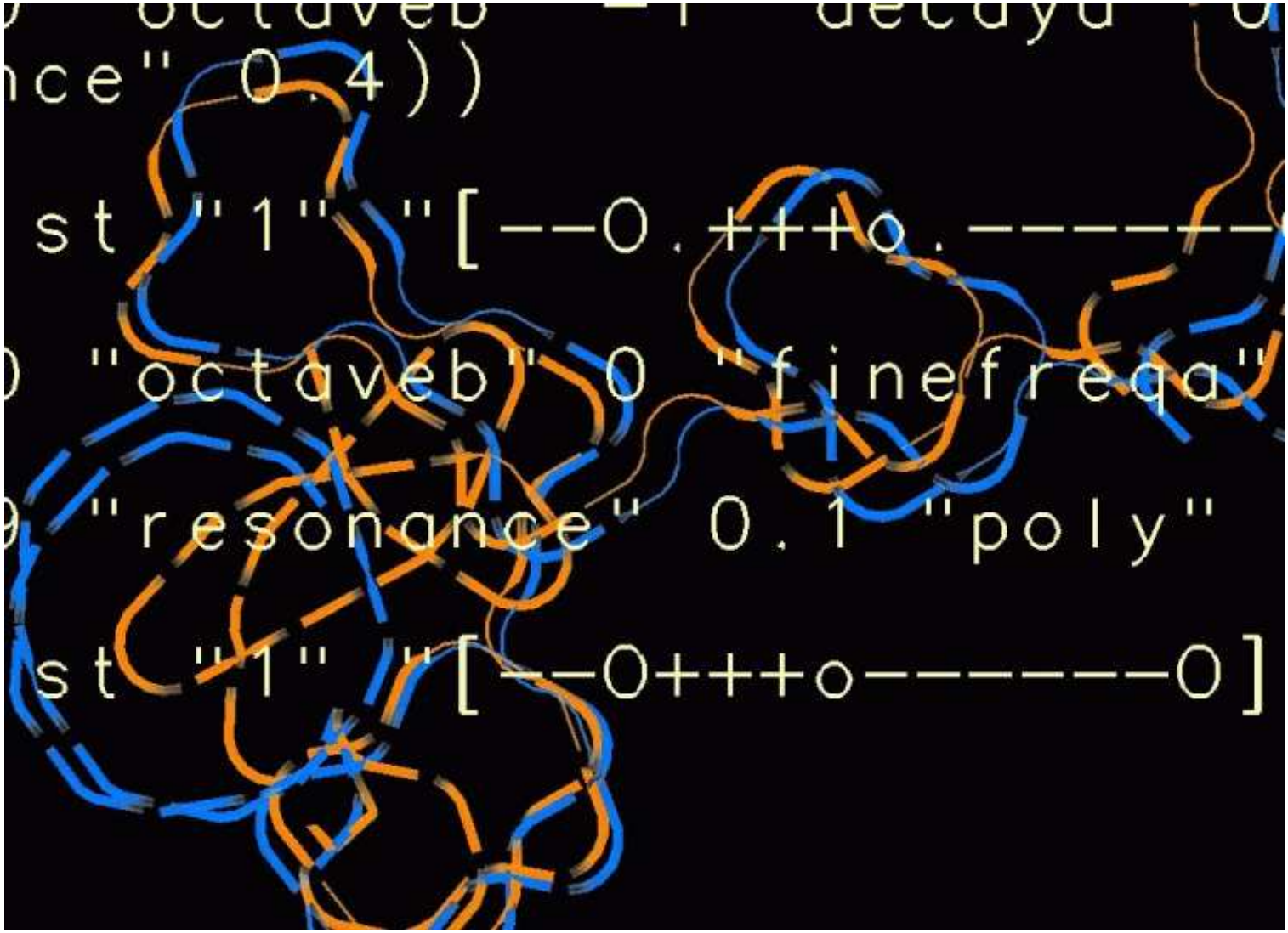
- Framework for various things:
  - Playing/learning about graphics
  - Workshops
  - Performances
  - Art installations
- Game engine at heart...
- With a livecoding editor
- Uses PLT Scheme
- Source released under GPL
- 4 or 5 developers working on it
- Builds on Linux and sometimes OSX



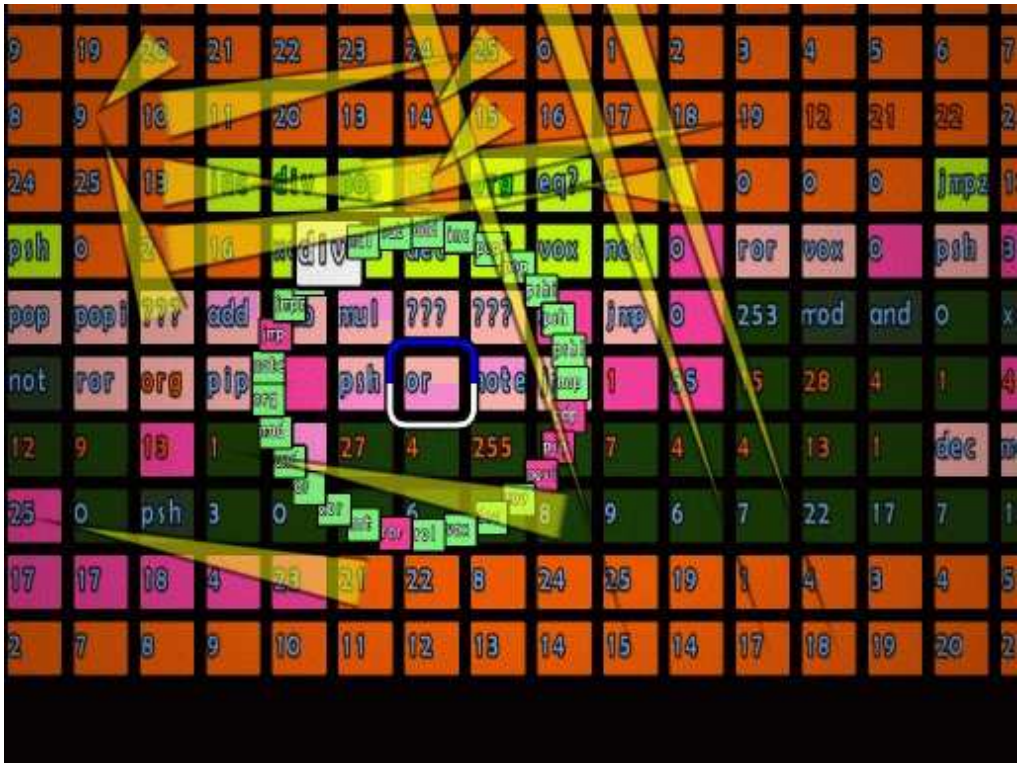
Quick demo

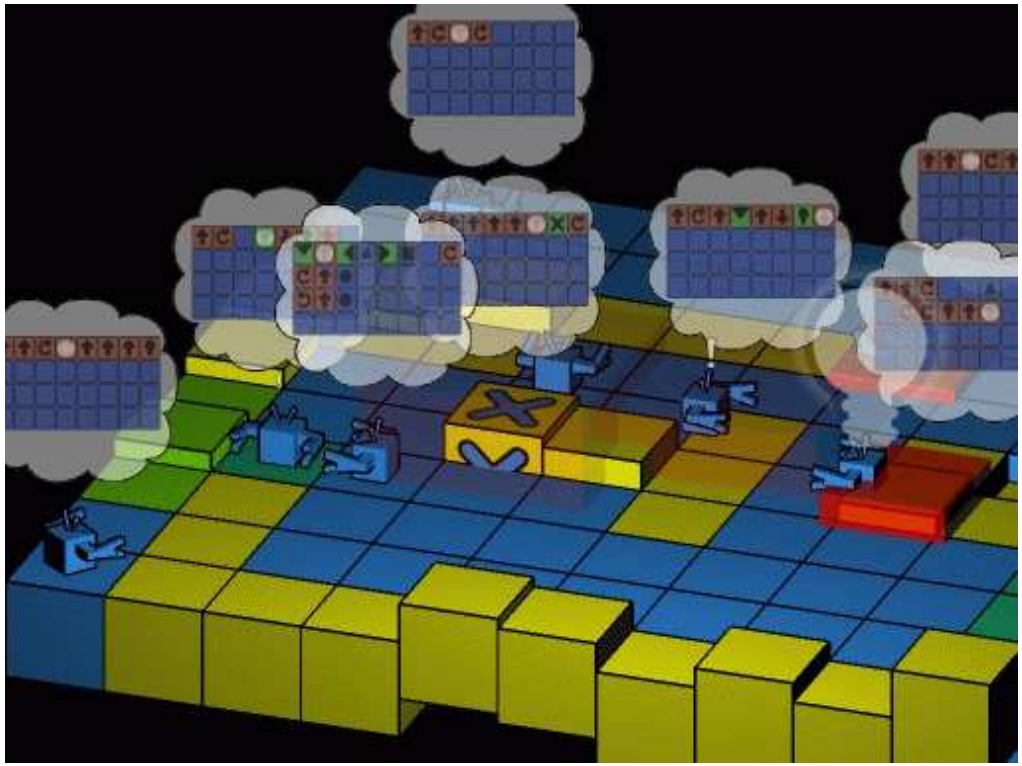
I use fluxus for...

Live coding graphics, using live audio input

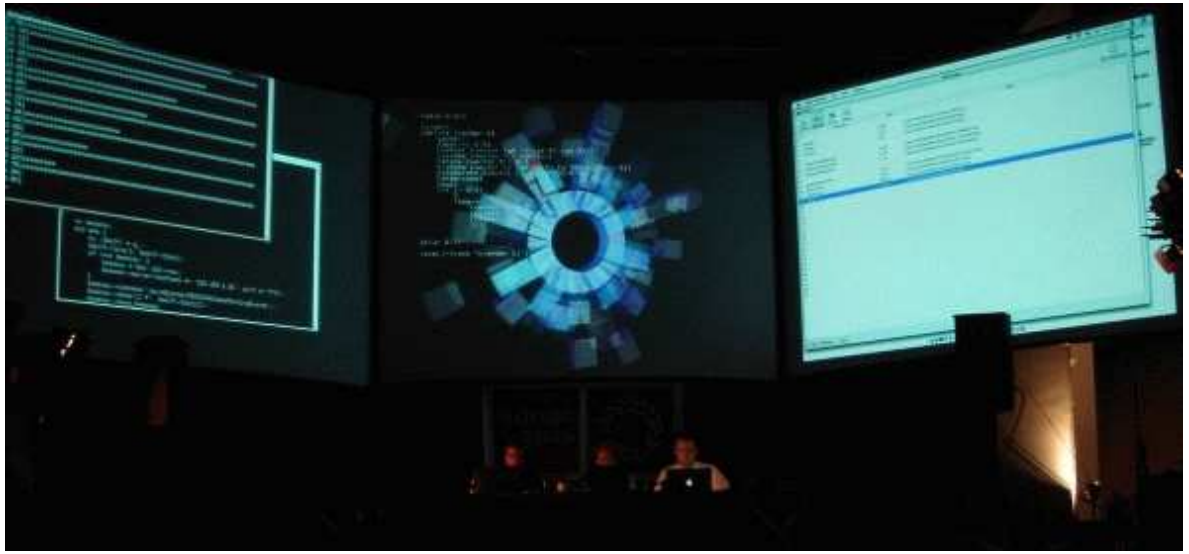


Live coding graphics and audio at the same time





As a framework for developing new livecoding languages



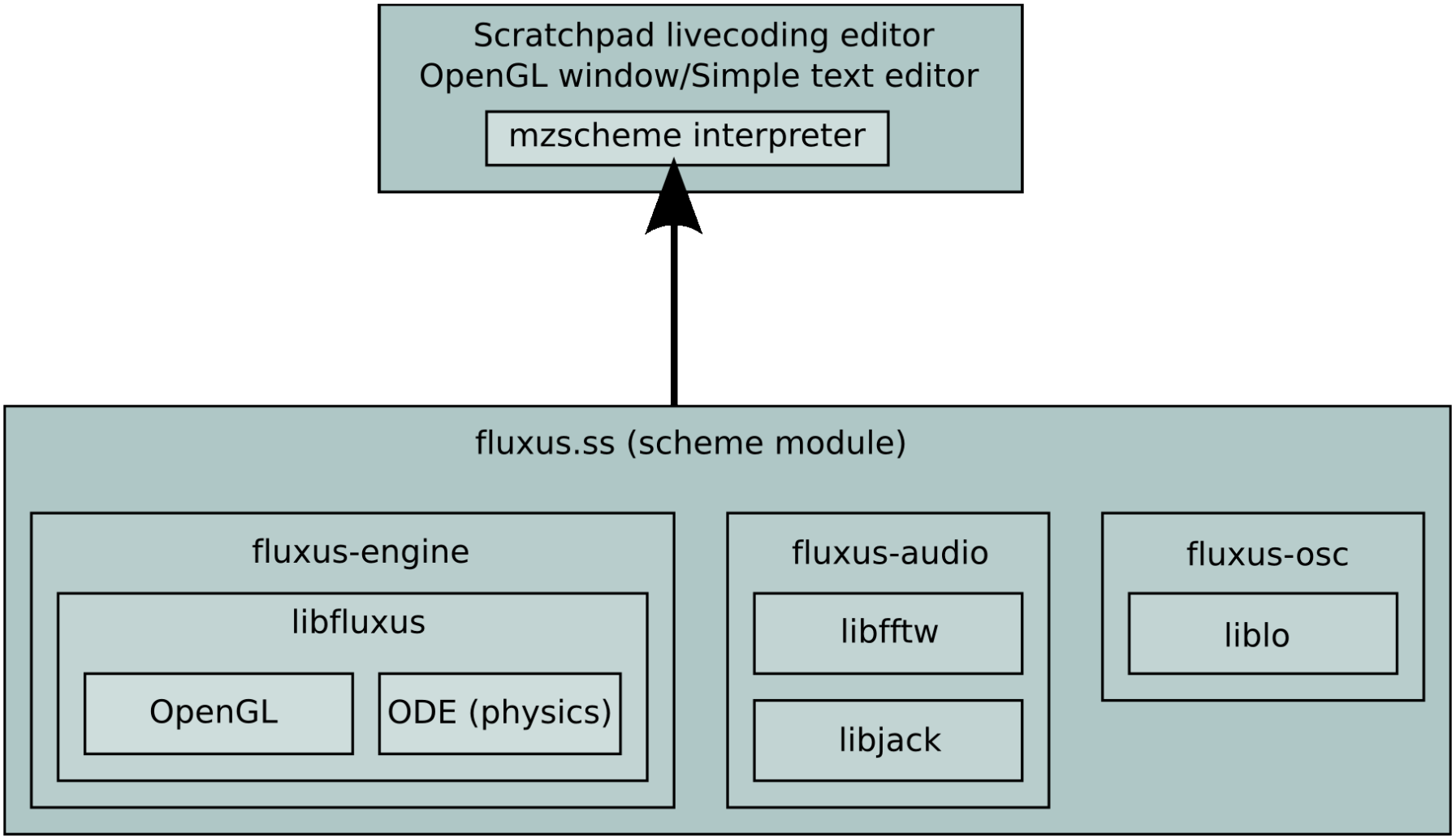
slub

What's inside?

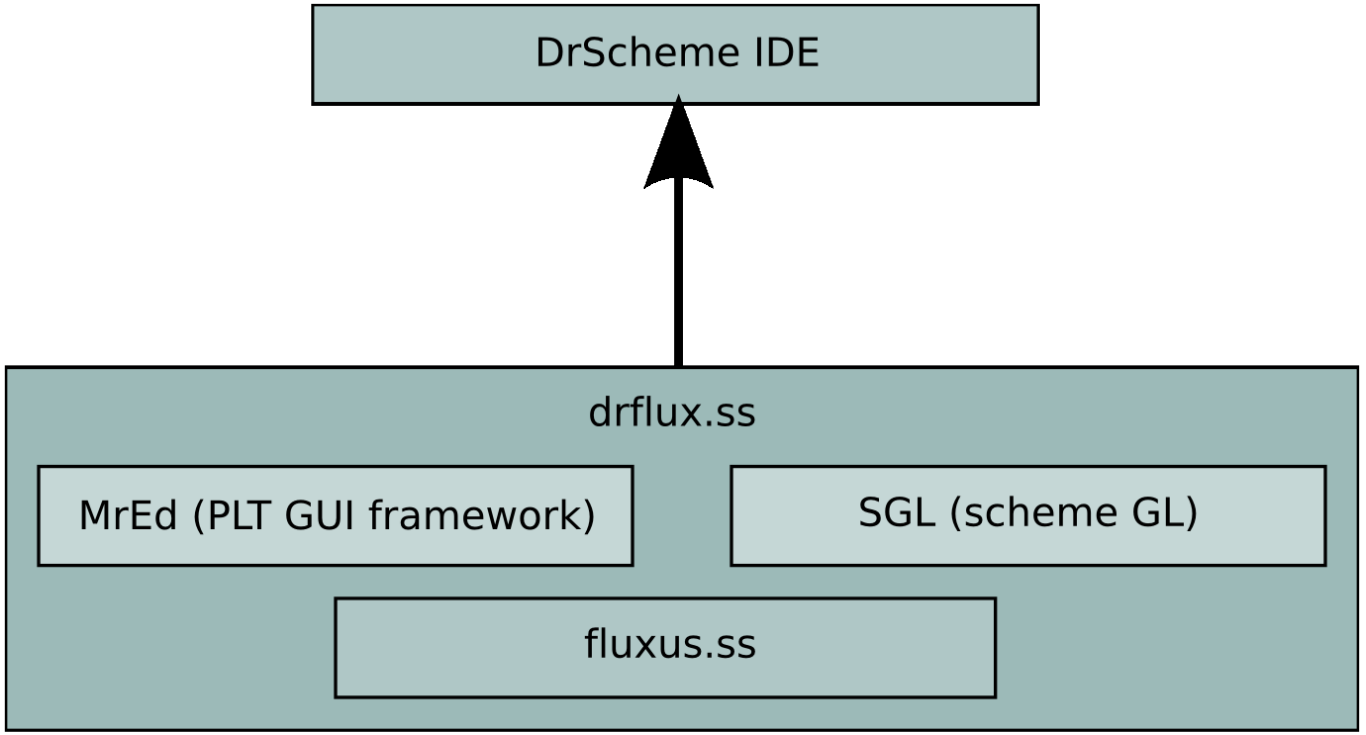
Boring Feature List

- Immediate mode drawing
- Scenegraph
- Primitives
  - Polys
  - Particles
  - NURBS patches
  - Blobbies (implicit surfaces)
  - Pixels (procedural texture access)
- Unified access to primitive data (vertex arrays, texture data)
- More advanced stuff
  - GLSL Hardware shading
  - ODE physics
  - Shadows
  - Skinning/Skeletons
- Audio synthesis

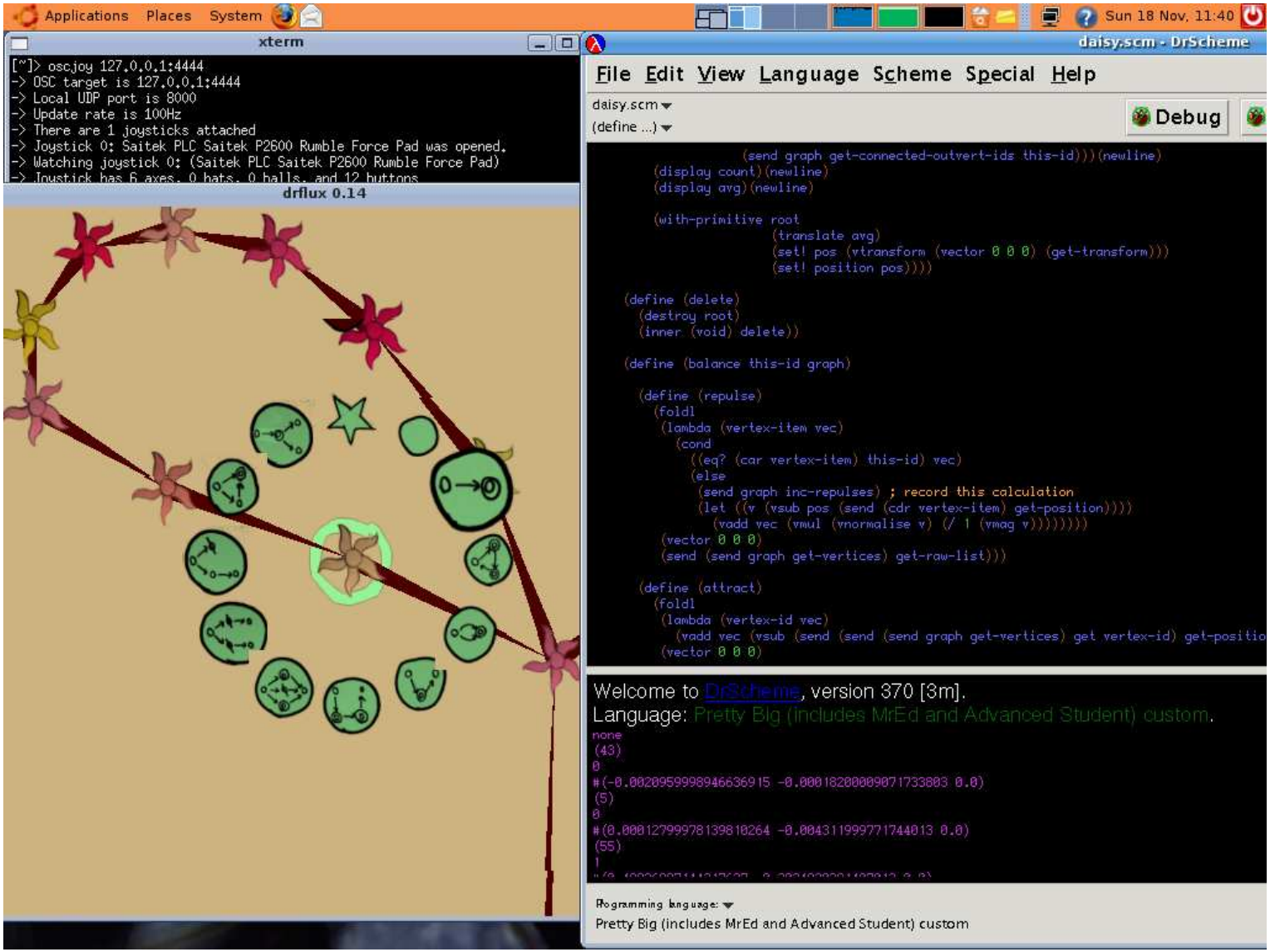
Architecture



DrScheme integration



DrScheme integration



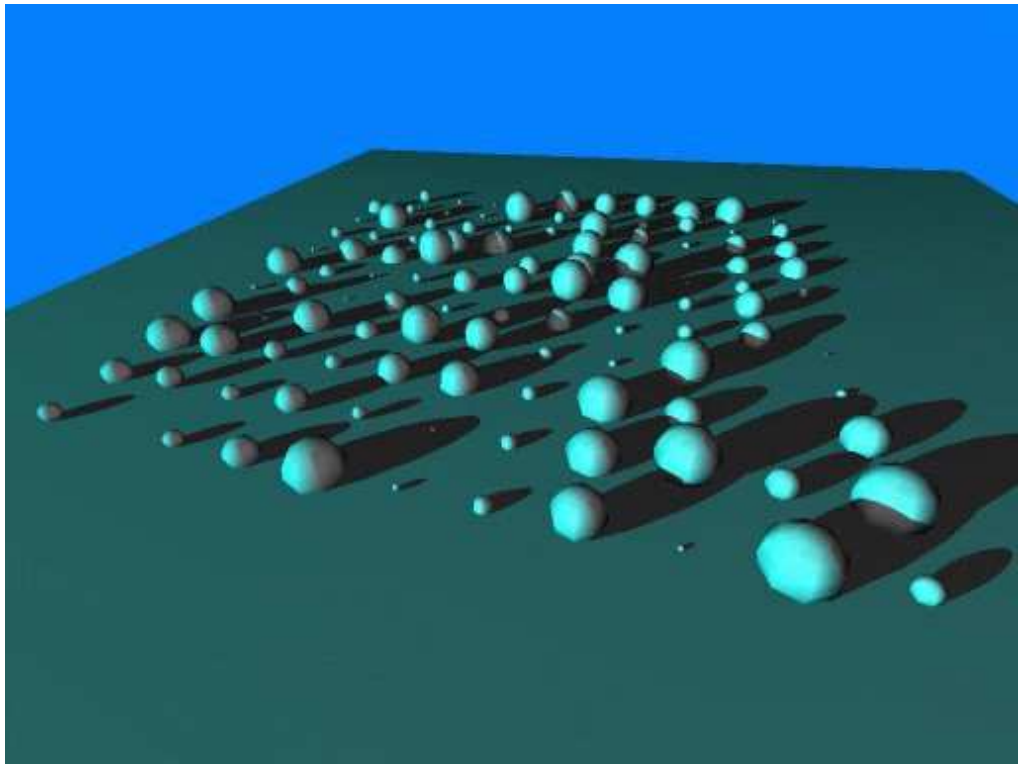
DrScheme integration

- Much better editor
- Debugger
- Profiler
- Syntax highlighting
- Less control over OpenGL
  - No GLSL shader support
  - No stencilmap shadows
  - Slower
- Not suitable for livecoding performance

Why Scheme?

Fluxus philosophy

- 'Creative' code in Scheme
- OpenGL grunt work in C++



Scheme for creative code

- Fast feedback (use an interpreter)
- Expressive power
- Few keypresses needed to get interesting results
  - Functional roots
  - Use of macros to shape the language
  - Dynamic typing
- Lots of interesting research going on with Scheme

Livecoding

Livecoding

- Performance programming
- Mainly a musical field
- Reaction against the normal laptop performance
- Showing the audience what you're doing

TOPLAP

- Formed Feburary 2004 in a smokey Hamburg bar
- Now grown to 100's of livecoders
- Role is to promote live coding as a unique art form

TOPLAP MANEFESTO

We demand:

- Give us access to the performer's mind, to the whole human instrument.
- Obscurantism is dangerous. Show us your screens.
- Programs are instruments that can change themselves.
- The program is to be transcended - Artificial language is the way.
- Code should be seen as well as heard, underlying algorithms viewed as well as their visual outcome.
- Live coding is not about tools. Algorithms are thoughts. Chainsaws are tools. That's why algorithms are sometimes harder to notice than chainsaws.

We recognise continuums of interaction and profundity, but prefer:

- Insight into algorithms
- The skillful extemporisation of algorithm as an expressive/impressive display of mental dexterity
- No backup (minidisc, DVD, safety net computer)

We acknowledge that:

- It is not necessary for a lay audience to understand the code to appreciate it, much as it is not necessary to know how to play guitar in order to appreciate watching a guitar performance.
- Live coding may be accompanied by an impressive display of manual dexterity and the glorification of the typing interface.
- Performance involves continuums of interaction, covering perhaps the scope of controls with respect to the parameter space of the artwork, or gestural content, particularly directness of expressive detail. Whilst the traditional haptic rate timing deviations of expressivity in instrumental music are not approximated in code, why repeat the past? No doubt the writing of code and expression of thought will develop its own nuances and customs.

Livecoding & Fluxus

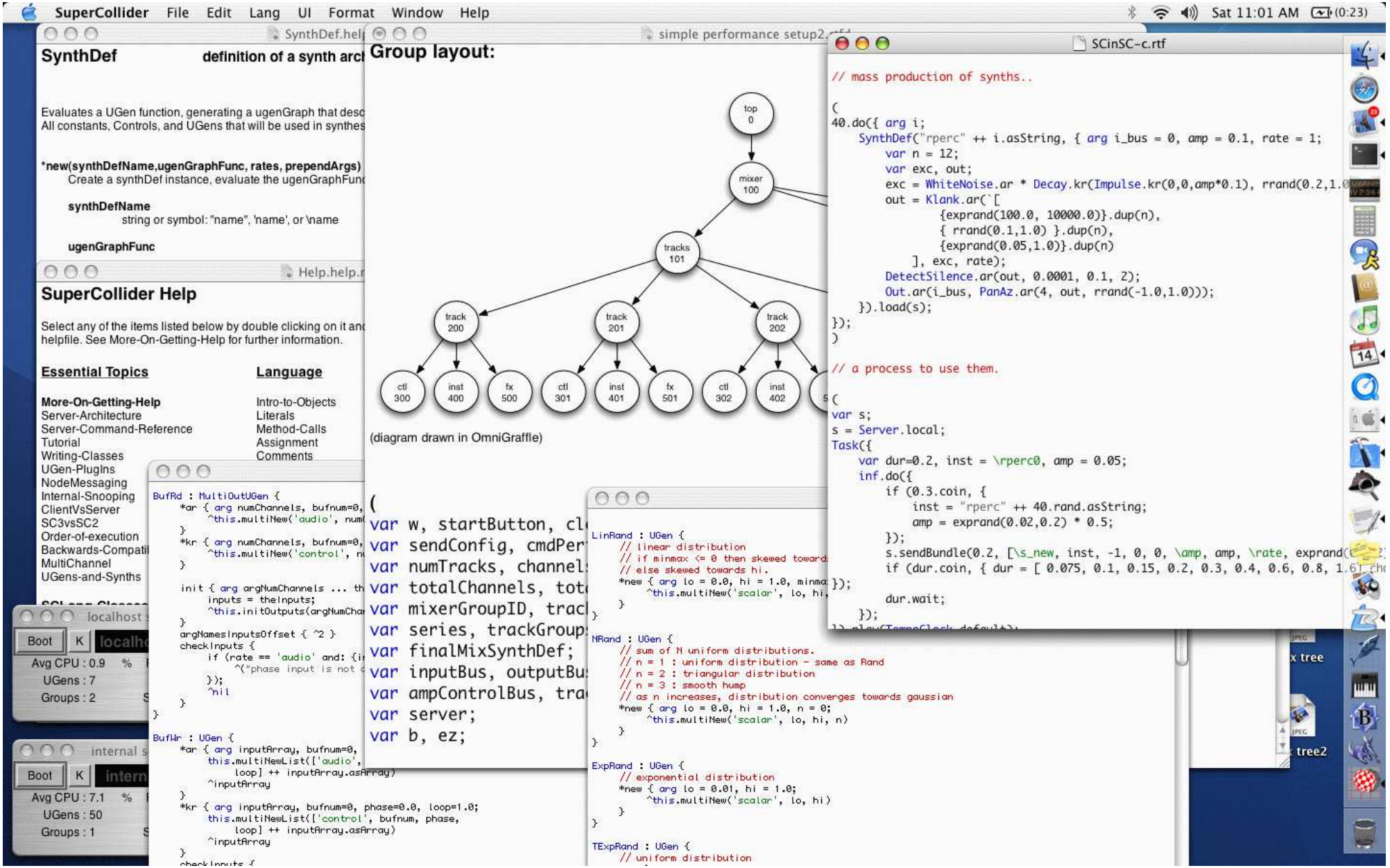
- Fluxus is part of the livecoding movement
- People using it for performance ('no copy paste' from Budapest)
- Fluxus/Supercollider Workshop at the LOSS Livecoding festival in Sheffield
- The movement has greatly influenced fluxus development

Some other livecoding systems

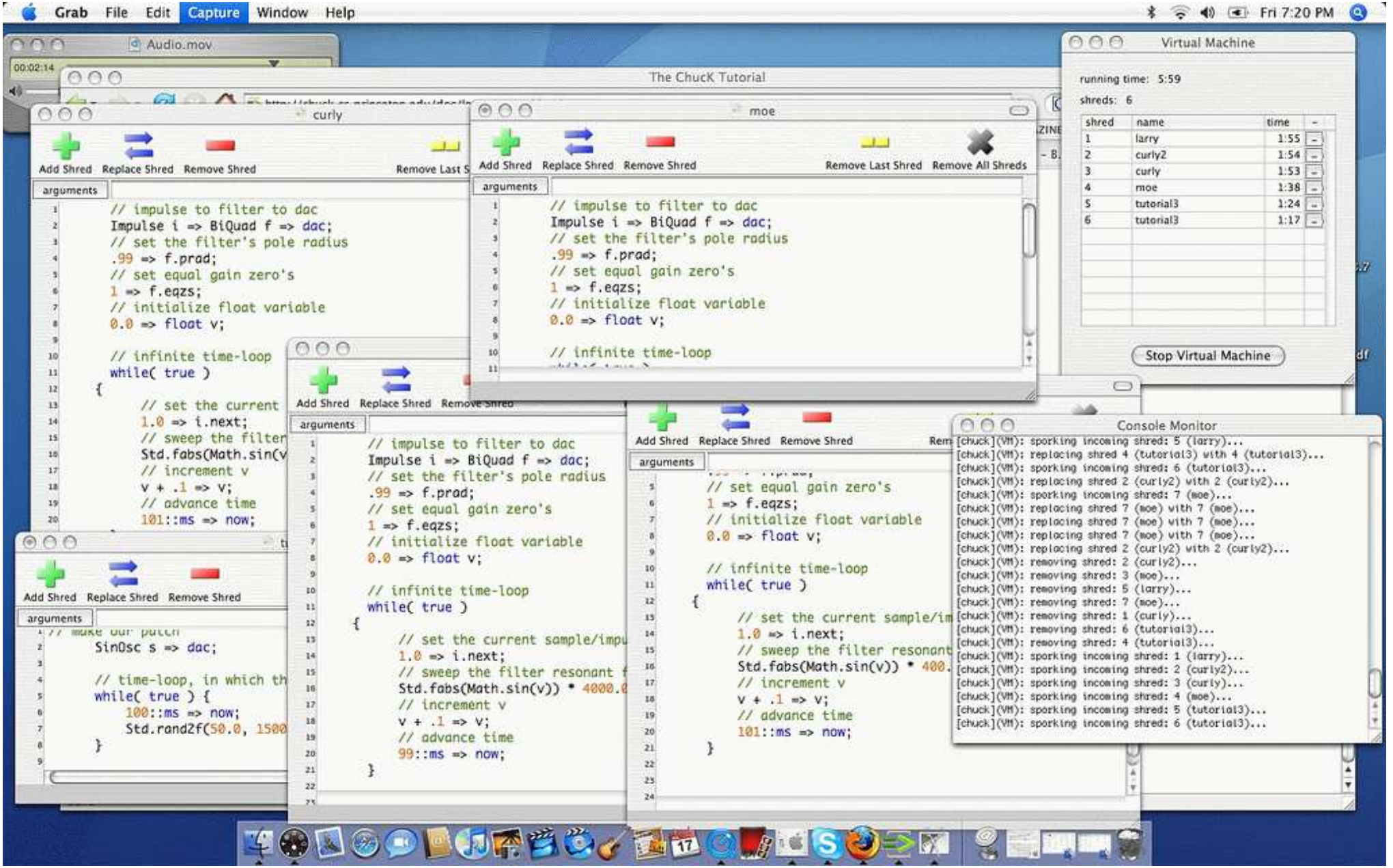
Impromptu


Impromptu Edit Host Window Help
OpenGL Window
tas harpsichord

SuperCollider



ChucK


Grab File Edit Capture Window Help
Fri 7:20 PM
Audio.mov
The ChucK Tutorial
curly
moe
Virtual Machine
running time: 5:59
shreds: 6
shred name time
1 larry 1:55
2 curly2 1:54
3 curly 1:53
4 moe 1:38
5 tutorial3 1:24
6 tutorial3 1:17
Stop Virtual Machine
Add Shred
Replace Shred
Remove Shred
Remove Last Shred
Remove All Shreds
arguments
// impulse to filter to dac
Impulse i => BiQuad f => dac;
// set the filter's pole radius
.99 => f.prad;
// set equal gain zero's
1 => f.eqzs;
// initialize float variable
0.0 => float v;
// infinite time-loop
while(true)
{
// set the current sample/impu
1.0 => i.next;
// sweep the filter resonant
Std.fabs(Math.sin(v)) * 4000.0
// increment v
v + .1 => v;
// advance time
99::ms => now;
}
SinOsc s => dac;
// time-loop, in which th
while(true) {
100::ms => now;
Std.rand2f(50.0, 1500
}
Console Monitor
[chuck](VM): sporking incoming shred: 5 (larry)...
[chuck](VM): replacing shred 4 (tutorial3) with 4 (tutorial3)...
[chuck](VM): sporking incoming shred: 6 (tutorial3)...
[chuck](VM): replacing shred 2 (curly2) with 2 (curly2)...
[chuck](VM): sporking incoming shred: 7 (moe)...
[chuck](VM): replacing shred 7 (moe) with 7 (moe)...
[chuck](VM): replacing shred 7 (moe) with 7 (moe)...
[chuck](VM): replacing shred 7 (moe) with 7 (moe)...
[chuck](VM): replacing shred 2 (curly2) with 2 (curly2)...
[chuck](VM): removing shred: 2 (curly2)...
[chuck](VM): removing shred: 3 (moe)...
[chuck](VM): removing shred: 5 (larry)...
[chuck](VM): removing shred: 7 (moe)...
[chuck](VM): removing shred: 1 (curly)...
[chuck](VM): removing shred: 6 (tutorial3)...
[chuck](VM): removing shred: 4 (tutorial3)...
[chuck](VM): sporking incoming shred: 1 (larry)...
[chuck](VM): sporking incoming shred: 2 (curly2)...
[chuck](VM): sporking incoming shred: 3 (curly)...
[chuck](VM): sporking incoming shred: 4 (moe)...
[chuck](VM): sporking incoming shred: 5 (tutorial3)...
[chuck](VM): sporking incoming shred: 6 (tutorial3)...

Fluxus code

Scene description

- Describing 3D scene and behaviours
- Simple example:

```
(define (draw-my-scene)
        (draw-cube))
(every-frame (draw-my-scene))
```

Scene description

- It's essentially a state machine:

```
(translate (vector 0 1 0))
(scale (vector 0.1 0.1 2))
(rotate (vector 45 0 0))
(colour (vector 1 0 0))
(texture (load-texture "brick.png"))
(shader "blinn.vert.glsl" "blinn.frag.glsl")
(draw-cube)
...
(draw-sphere)
(draw-torus)
```

Example Demos

Mutable state

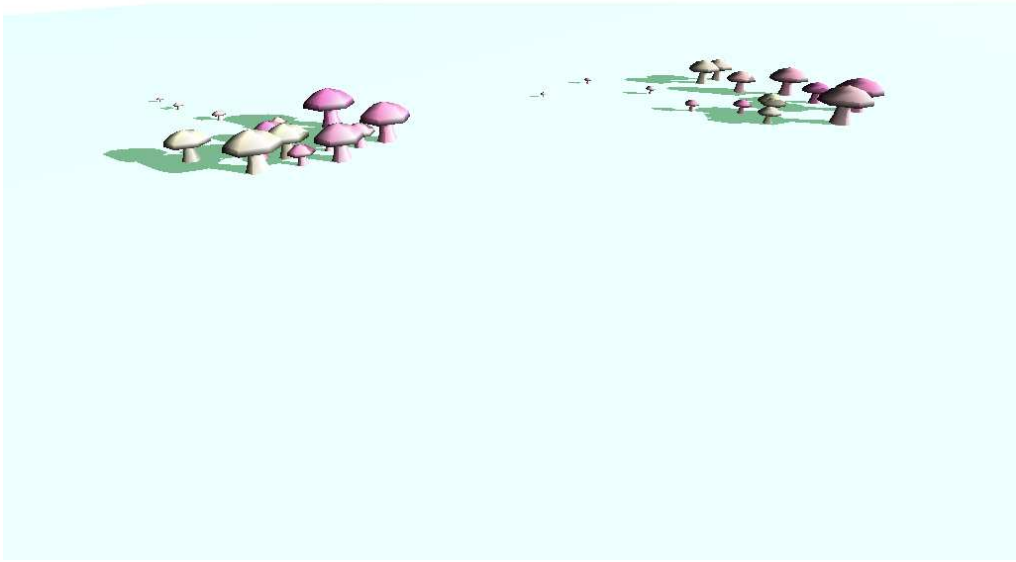
- Everything so far relies on mutable state
- Some of this is for performance
- Conventional way of using scene graphs
  - Init step: build scene graph
  - Every frame step: update scene graph
- Complexity starts to bite with big scenes
- Is there a better way?

Functional Reactive Programming

- Applying declarative programming for reactive systems
- First class behaviours and events
- Examples:
  - Fran - Functional reactive animation (Haskell)
  - Yampa - Haskell
  - Flapjax - Javascript
  - FrTime - PLT Scheme
- Avoids the complexity of state machines for behaviour
- More scalable
- Starting to be looked at here and there for games

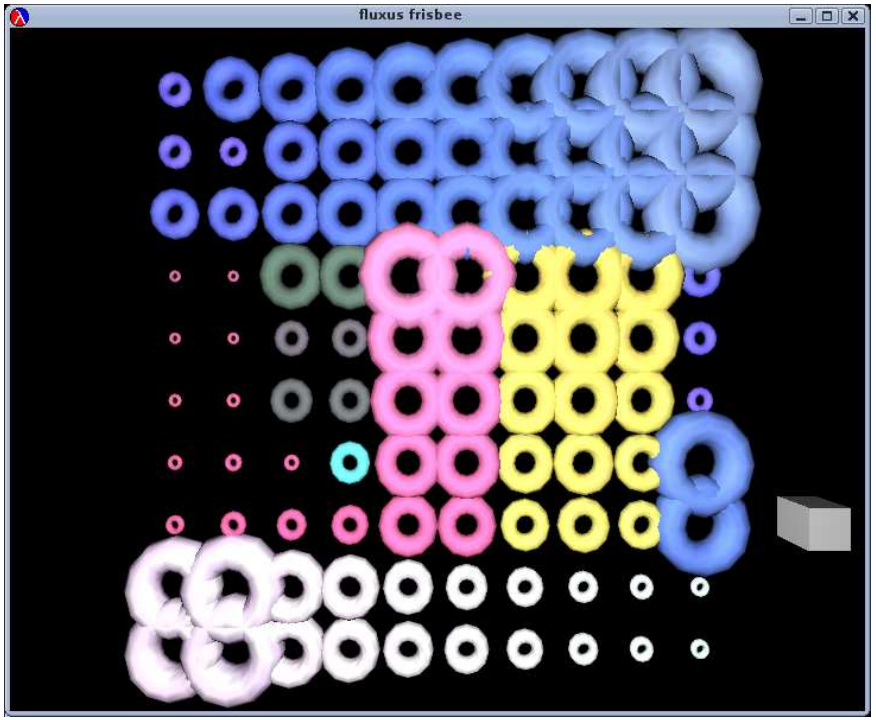
Frisbee

- FRP in fluxus
- Based on FrTime - by Gregory Cooper
- Trying to build a game engine on top of FrTime
- Could be a more expressive language for fluxus
- Easier to teach game programming
- Increase what is possible while livecoding



Frisbee disclaimers

- Frisbee is very much work in progress
- FrTime is still undergoing optimisation work
- A few glitches still
- We're working on it...



Frisbee Demo

Gamepad Livecoding

Gamepad Livecoding

- Make watching live coding a bit more accessible
- Live coding doesn't have to be about text editors
- Live coding doesn't have to be hard
- Making fun, simplified languages