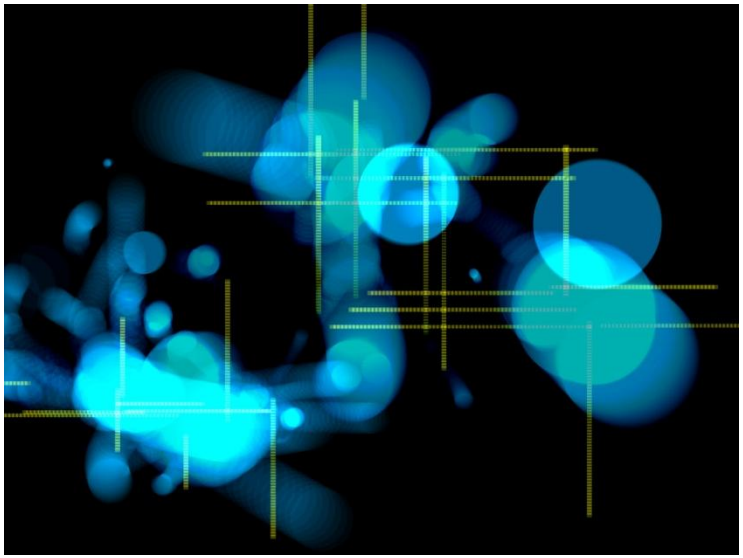
Finger Painting with Planets



Tim Thompson
tjt@nosuch.com

What is it?

- Installation for people to play with
- Generates music and graphics simultaneously
- Controller with buttons, knobs, LCD, multitouch pad
- Fingers on multitouch pad trigger music or graphics
- Graphics motion is simulated gravitational attraction
- Collisions of planets trigger music
- Musical keyboard controls (only) selection of notes

Appearances

- Yuri's Night 2008
- Maker Faire 2008
- Night Light at Climate Theater
- Anon Salon at Climate Theater
- SubZERO street fair, ZERO1 Festival
- Starry Night at Villa Montalvo
- Burning Man 2008

User interface was adjusted/simplified each time

Big Pieces

- KeyKit – input and realtime processing
- Plogue Bidule – VST host for sounds
- Salvation – Freeframe host for visuals
- Planets – Freeframe plugin (embedded Python)
- Cairo – drawing on bitmap (from Python)
- Chipmunk – Physics simulation (from Python)
- OpenCV – raster manipulation (from C)
- OSC – communication between KeyKit and Planets

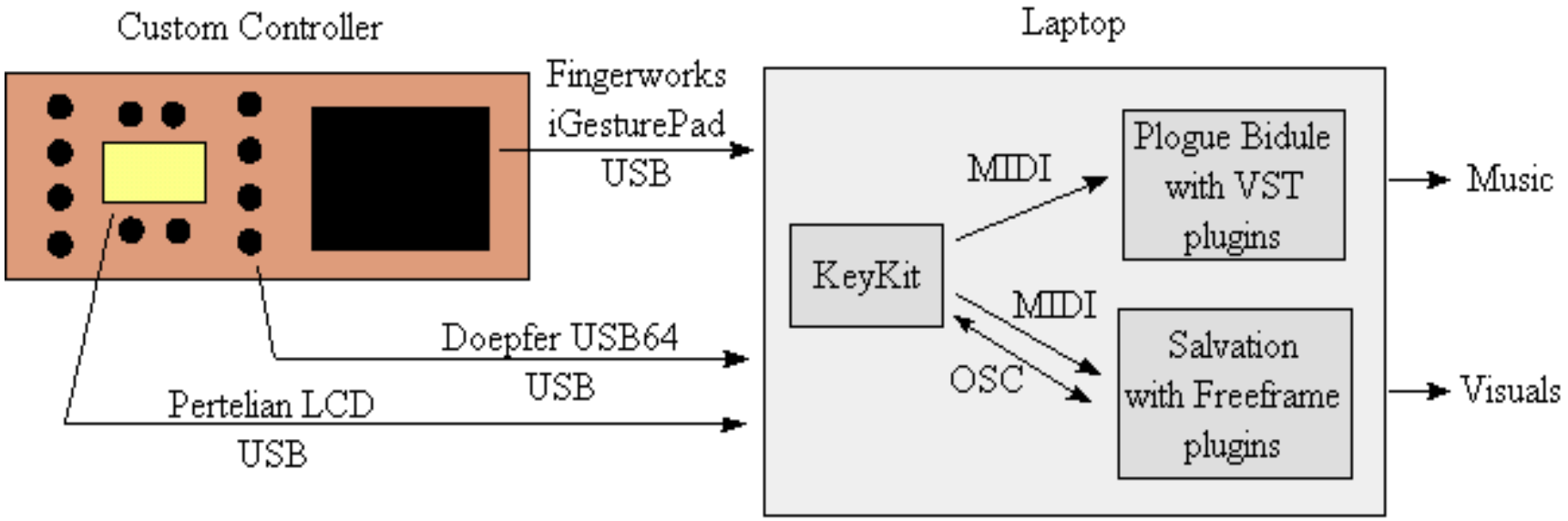
A Mashup without the Web

- Re-using and combining large pieces of software
- Explosion of large pieces of freely available code
- Protocols and mechanisms are relatively standardized
- Good separation of functionality
- APIs are now front and center
- Ease of integration is getting better
- Requires care in selection, one bad apple...

Multiple Languages

- Attempted to avoid it, now embraced
- Each language has pros/cons in:
  - Library availability
  - Device I/O availability
  - Robustness
  - Ease of Development
  - Familiarity
  - Expense

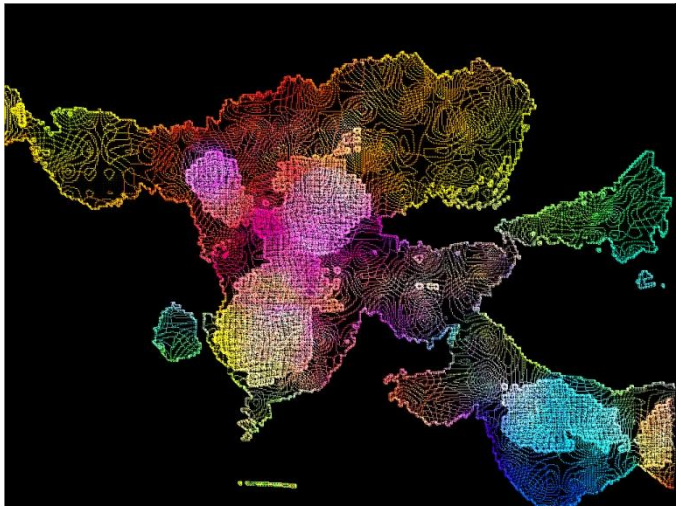
What's connected to What

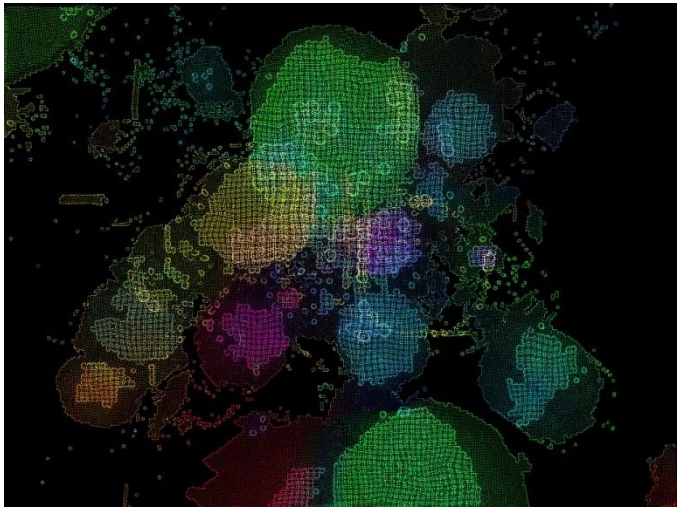


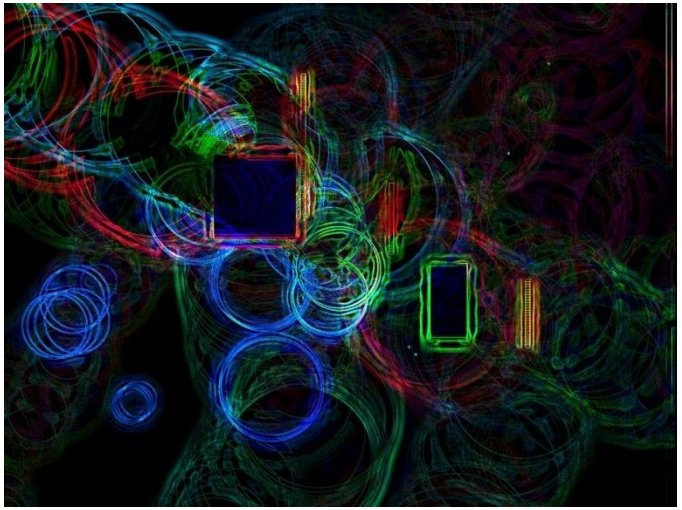
Custom Controller

Controllers In Use

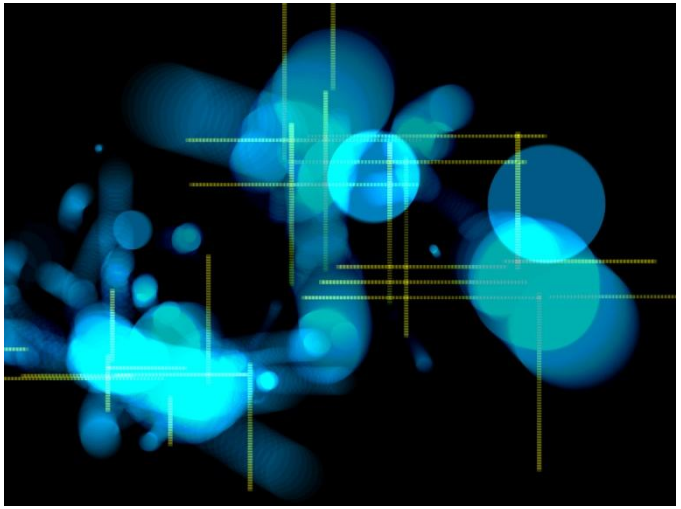
Examples of Visual Output

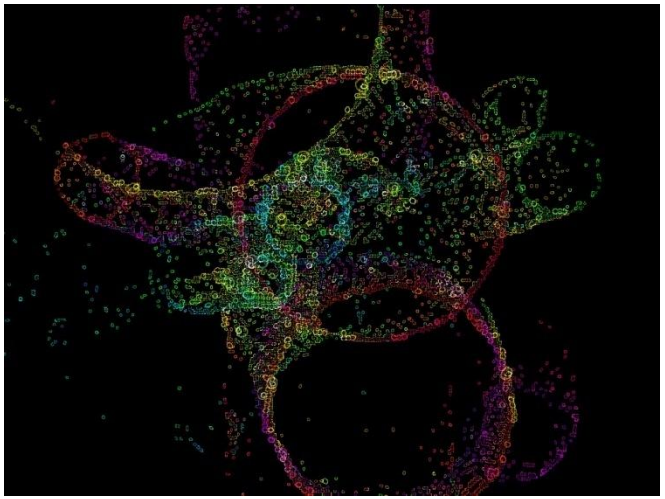


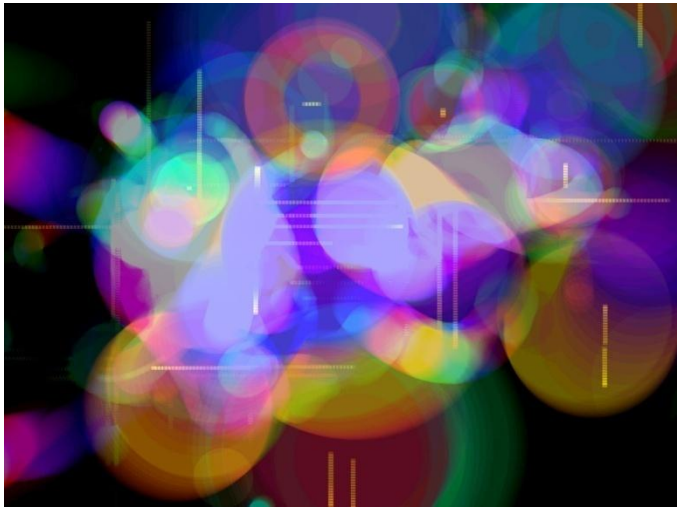


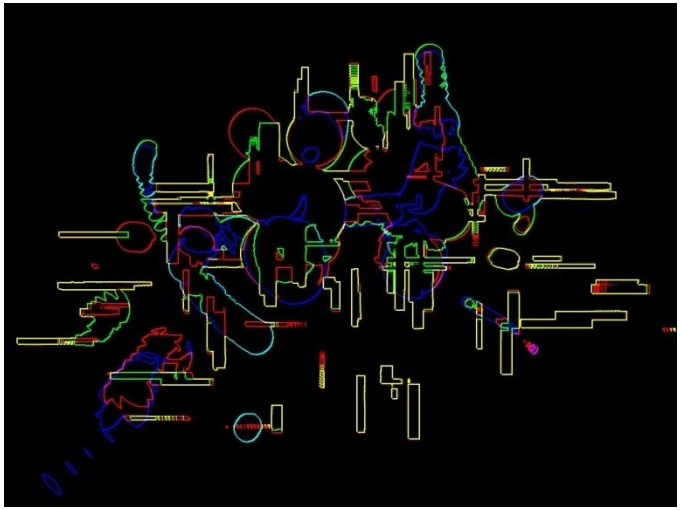


Examples of Visual Output

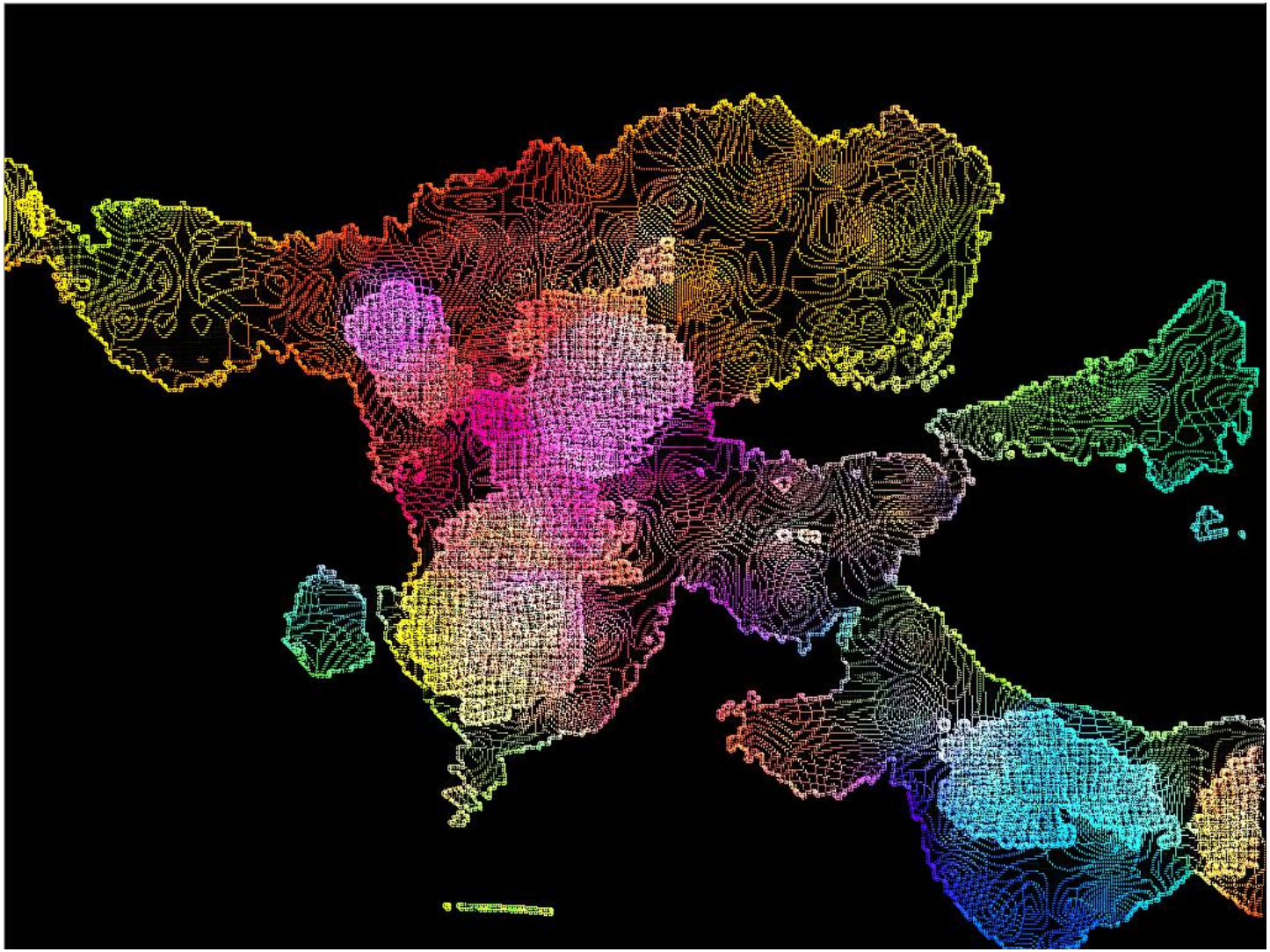






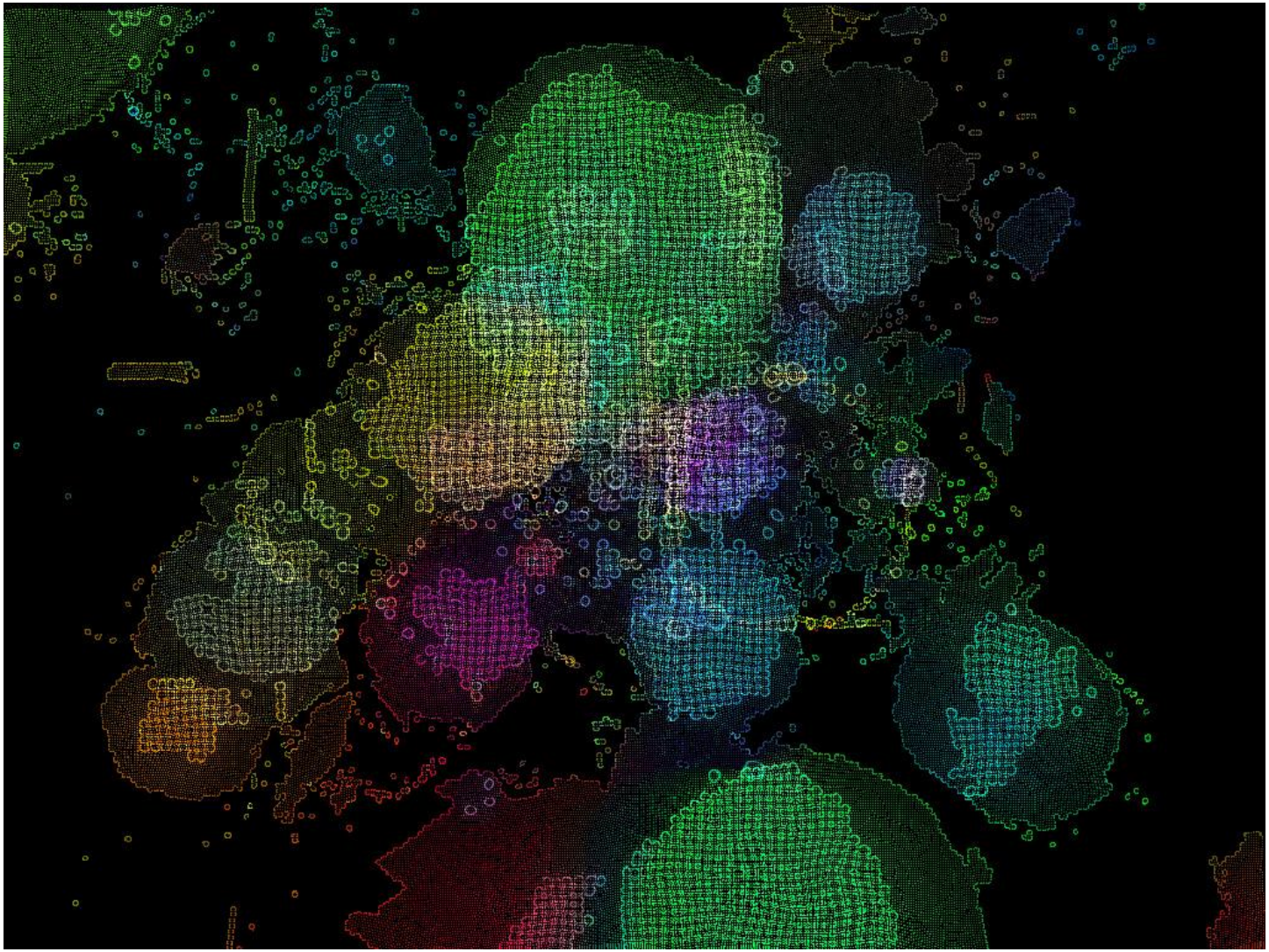


Examples of Visual Output

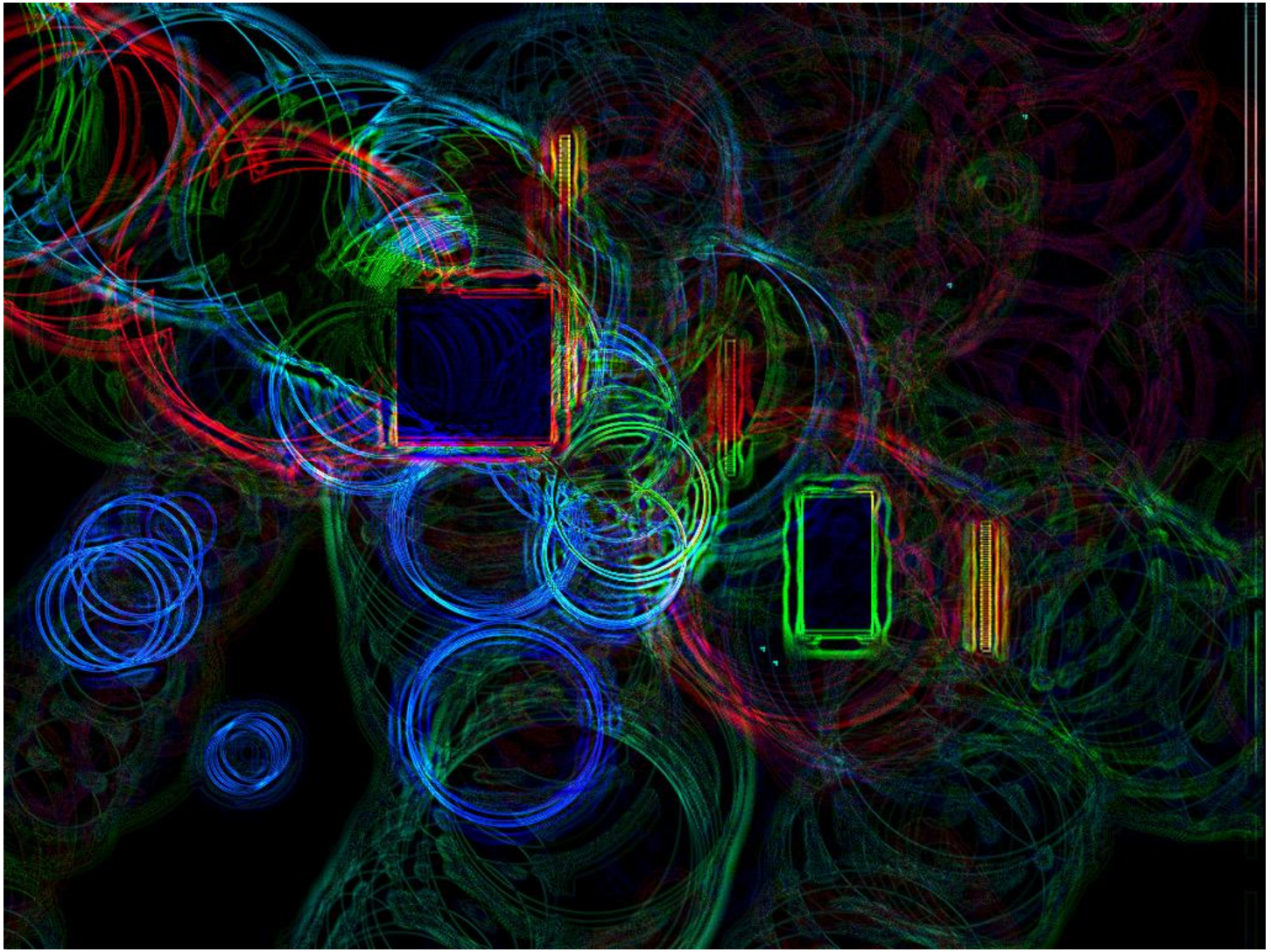


Examples of Visual Output

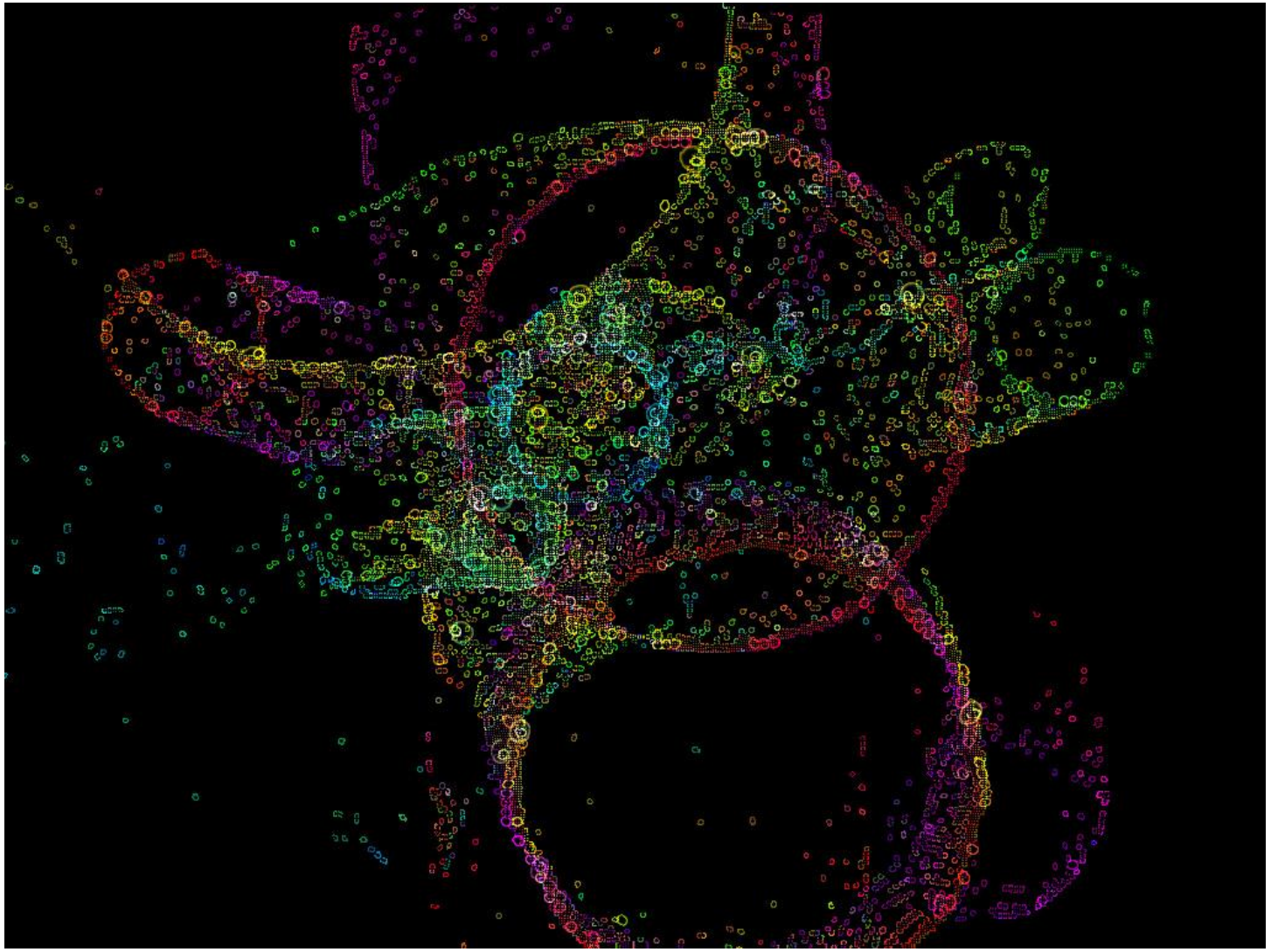
Examples of Visual Output



Examples of Visual Output



Examples of Visual Output



User Interface

- 4 buttons, 8 knobs, LCD, multitouch pad
- Knob style: endless rotation vs. absolute position ?
  - Hardware availability influenced the choice - absolute
- LCD used to 'label' the 8 knobs
- First iteration: many pages of parameters
- Second iteration: 2 pages of parameters (graphics & music)
- Third iteration: 1 page of parameters
- Expert mode allows access to all parameters

Code Pieces

- Languages
  - KeyKit
  - C/C++
  - Python
- Toolkits
  - Chipmunk (called from Python)
  - OpenCV (called from C/C++)
  - Cairo (called from Python)

Interface Pieces

- Standards
  - MIDI
  - Freeframe
  - OSC
- Hardware
  - Fingerworks iGesture multitouch pad (USB)
  - Doepfer USB64 MIDI control board (USB)
  - Pertelian LCD (USB)

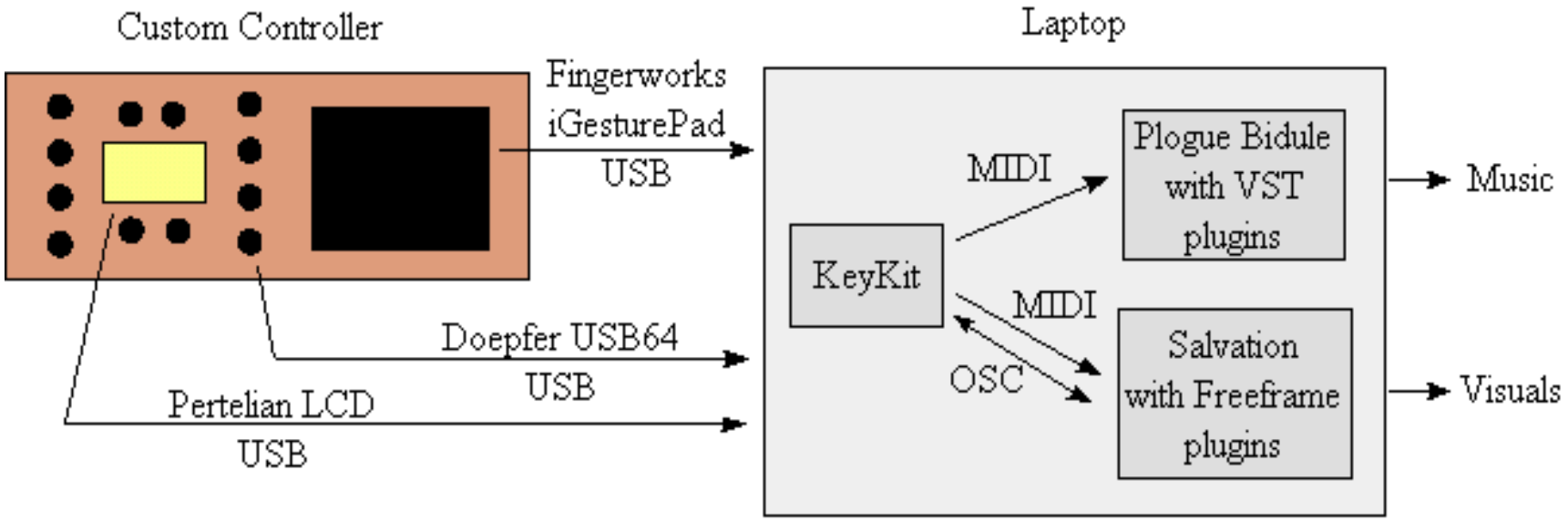
Application Pieces

- Applications
  - KeyKit
  - Salvation
  - Plogue Bidule
- Plugins
  - VST soft synths
  - Freeframe video processors
  - Planets plugin

My Pieces

- Decisions on what software and protocols to use
- KeyKit code for input processing and LCD control
- C and Python code in “Planets” Freeframe plugin
  - OpenCV for bitmap formatting/fading (from C)
  - OSC for 2-way communication with KeyKit (from C and Python)
  - Chipmunk for physics (from Python)
  - Cairo for 2d graphics (from Python)

What's connected to What



Event Routing

- Finger events are detected by Keykit, and either trigger sounds directly or get passed to Planets plugin via OSC, instantiating planets in Python code
- Python code simulates physics/gravity, moving planets
- When planets collide, a visual 'tracer' is generated (horizontal/vertical lines), and OSC is sent back to KeyKit to trigger a sound
- KeyKit sends MIDI to Plogue/VSTs to make sounds
- Knob and button movements are read by KeyKit as MIDI and processed by control logic in Keykit, occasionally sending OSC to Planets plugin to control its parameters
- KeyKit manipulates LCD display as knobs/buttons are used
- Music keyboard sends MIDI to KeyKit, for setting scales

Processing

- KeyKit has a MIDI looper, optionally enabled by a knob
- Generated MIDI is periodically transposed, in a cycle
- Planet motion is controlled by gravity and inertia in Python, invoked every frame (15 per second or so) from within a Freeframe plugin running inside Salvation
- Visual are generated by a serial chain of 3 Freeframe plugins:
  - Planets plugin does the initial drawing/movement, controlled by OSC from KeyKit
  - 2 other Freeframe plugins are controlled by MIDI
  - MIDI is sent from KeyKit to Salvation in order to select which 2 specific Freeframe plugins (from a set of several dozen) are used, and to control their parameters

Using Python from a Freeframe plugin

- “Planets” is the freeframe plugin, written in C/C++
- When it first initializes, the freeframe plugin:
  - Instantiates python
  - Recompiles/reloads nosuch.particles python module (so python code can be changed without restarting freeframe host)
  - Calls python to instantiate cairo surface and context
  - Retrieves a C-accessible surface and creates a C-accessible OpenCV image (IplImage)

Using Python from a Freeframe plugin

- On every video frame, the freeframe plugin:
  - Calls python to advance time, update planet positions and movement, and draw current planets onto the cairo surface
  - After returning to C code, the cairo surface is added to an accumulation buffer/image which is then progressively faded, using OpenCV routines.
- On every received OSC message, the freeframe plugin:
  - Sometimes adjusts parameters in the C code, but usually...
  - Forwards the OSC message intact to python to process it - changing parameter values, creating new planets, clearing, etc.

Other details

- Used PyBufferProcs to get access to PyCairo image surface
- Need to convert RGBA (from cairo) to RGB (for freeframe):
  - cvCvtColor(cairoimage1,accum1,CV_RGBA2RGB);

Python access to Chipmunk physics

```
import pymunk._chipmunk as cp
cp.cpInitChipmunk()
space = cp.cpSpaceNew()
moment = cp.cpMomentForCircle(mass, inr, outr, cpvzero())
body = cp.cpBodyNew(mass, moment)
cp.cpBodyResetForces(body)
cp.cpSpaceAddBody(space, body)
s = cp.cpCircleShapeNew(body, radius, cpvzero())
cp.cpSpaceAddShape(space, s)
cp.cpBodyApplyForce(body, force, cpvzero())
cp.cpSpaceStep(space, deltatime)
```

Python access to Cairo drawing

```
import cairo
s = cairo.ImageSurface(cairo.FORMAT_ARGB32,width,height)
c = cairo.Context(s)
c.set_source_rgba(r, g, b, a)
c.set_operator(cairo.OPERATOR_SOURCE)
c.set_line_width(width)
c.scale(sx, sy)
c.translate(tx, ty)
c.move_to(x,y)
c.curve_to(x1,y1, x2,y2, x3,y3)
c.rel_line_to(x,y)
c.fill()
c.stroke()
```

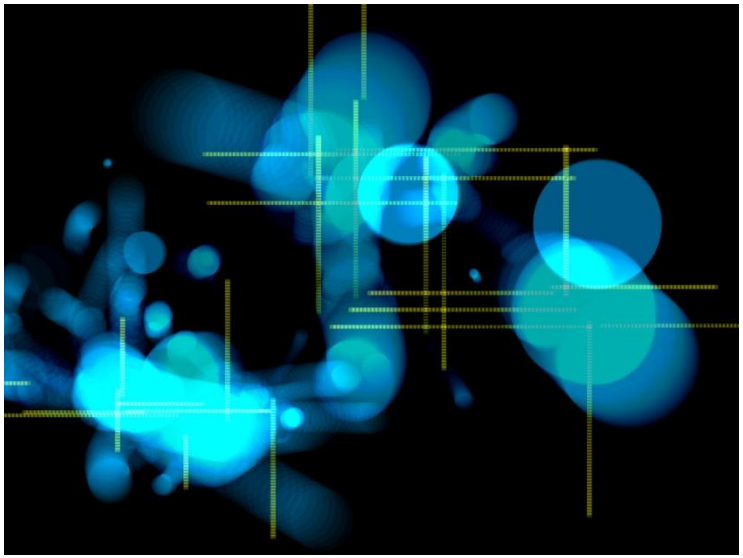
In Hindsight, the Good Things

- Python integration with low-level code works well, is robust, and has good syntax/error reporting
- Ability to change Python code without restarting the larger application is very nice
- Bitmap manipulation with multiple toolkits can work
- OSC is a nice simple API mechanism, and a good "side-channel" for controlling Freeframe plugins
- Local sockets for inter-app API invocation good for:
  - Flexibility in choice of languages and applications
  - Portability, Firewalling, Robustness
  - Separating device I/O from graphics/audio output

In Hindsight, the Bad Things

- Devices and drivers are often the weak link
  - Things that work in isolation may not work simultaneously
  - The more devices you have, the more problems you have
  - Always try to have a quick way of resetting/restoring things that is controllable from the primary interface
- Absolute knob style is a pain
- Version control is a pain with so many pieces
- Giving it to other people is difficult
- OS dependencies

Finger Painting with Planets



Tim Thompson
tjt@nosuch.com