Yeah, ChucK It! Dynamic, Controllable Interface Mapping

Ge Wang gewang@cs.princeton.edu
Ananya Misra amisra@cs.princeton.edu
Ajay Kapur ajay@ece.uvic.ca
Perry R. Cook prc@cs.princeton.edu

Department of Computer Science (also Music)
Princeton University, Princeton, NJ, U.S.A.

Music Intelligence and Sound Technology Interdisciplinary Center
University of Victoria, Victoria, BC, Canada

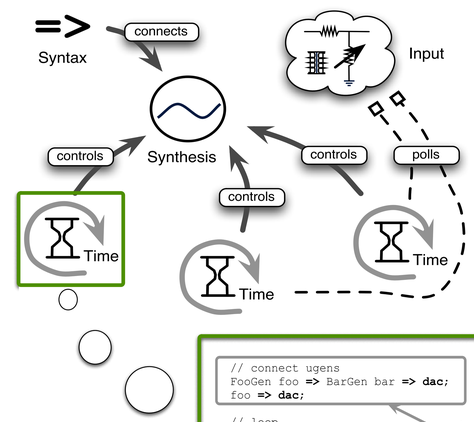
http://chuck.cs.princeton.edu/

1 ChucK What?

Strongly-timed and concurrent programming language
High-level rapid prototyping + low-level control

- Define control rate for each sensor independently, at any granularity
- Map many sensors concurrently
- Create and modify patches dynamically
- Quickly experiment using on-the-fly programming
- Free!

2 Programming Model



Concurrent processes control audio synthesis over time.

```
// connect ugens
FooGen foo => BarGen bar => dac;
foo => dac;

// loop
while( ... )
{
    ...
    xyz() => foo.abc;
    ...
    ...
    D1 +=> now;
    ...
    ...
    D2 +=> now;
}
...
D3 +=> now;
...
```

dynamically create and connect unit generators

assert control on UGen foo

statements to advance time may be placed anywhere

3 Basic Mapping

- MIDI
- OSC
- Events
- Triggers

```
// infinite time-loop
while( true )
{
  // event (instead of explicitly advancing time)
  midi.event( msg ) => now;

  // at this point, assume event occurred
  // handle msg as desired…
}
```

event-driven

```
// the patch: sine wave to reverberator to dac
sinosc s => JCRev r => dac;
// initial settings
.2 => r.mix;

 // declare a MIDI message to be filled in
 MidiMsg msg;

 // infinite time-loop
 while( true )
 {
    // advance time (by 5 milliseconds)
    5::ms => now;

    // poll MIDI - (could be any event)
    while( midi.in( msg ) )
    {
       // if the message was a note on
       if( msg.type == midi.NOTE_ON )
       {
          // convert to frequency, and set
          std.mtof( msg.data1 ) => s.sfreq;
          // set gain of sinosc
          msg.data2 / 128.0 => s.gain;
       }
       else if( msg.type == midi.KNOB )
       {
          // set dry/wet mix and gain of reverb
          midi.data1 / 128.0 => float mix => r.mix;
          .25 + .25 * (1.0-mix) => r.gain;
       }
    }
 }
```

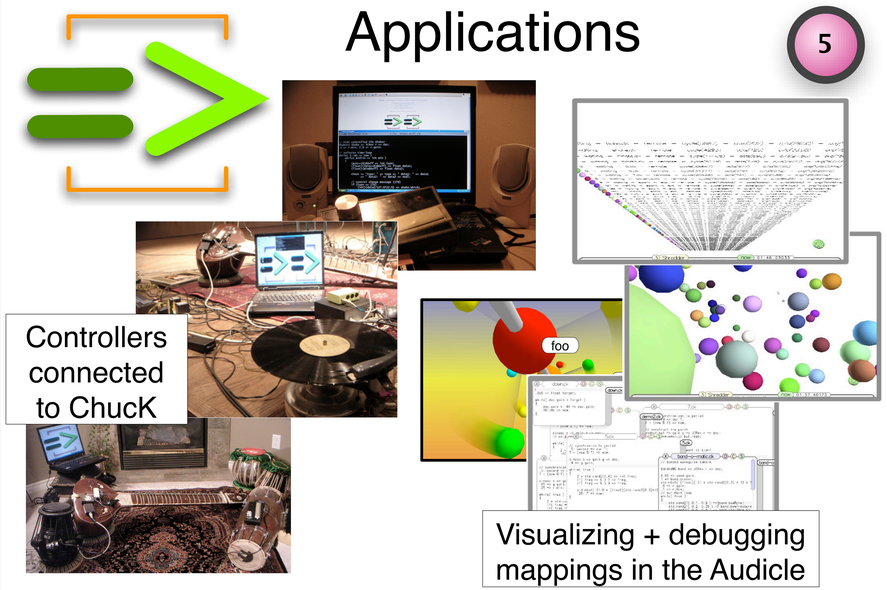
polling

4 Advanced Features

OK

Dynamic Patching: connect/disconnect unit generators at runtime
Dynamic Voice Allocation: interactively instantiate entire patches; useful for polyphony
On-the-fly Controller Mapping: edit/swap mappings sample-synchronously during runtime
Precise Recording/Playback: sample-accurate recording and playback of gestural data; record an entire performance, playback while tweaking parameters in a precisely-timed manner.

5 Applications



Controllers connected to ChucK

Visualizing + debugging mappings in the Audicle