ChucK Lesson 7

24 May 2012

This lesson discusses Events and how ChucK responds to external inputs.

ChucK (like Max/MSP, Supercollider, and most other musical computer languages, has two faces.

- The **logical** face of ChucK interprets the instructions in your code.
- The **audio** face of ChucK generates a continuous stream of audio samples.

Here's the thing: ChucK can only do *one or the other* at once! When it's calculating audio, it's not listening to your instructions; and it is forced to cram all of your instructions in a tiny tiny period of time between samples.

1 When is now?

We've been using **now** to pass time, but we haven't really explored how it works. The instruction **10::second =¿ now;** is a built-in shorthand for *"Schedule an event 10 seconds in the future, and then wait until that event happens."* This is a scheduler event, but ChucK allows us to create other types of events as well. An input from a MIDI keyboard, or a keypress from your computer, or a message sent over the internet can be events too.

2 HID Events

HID stands for Human Interaction Device, and it is a blanket term for any of the standard methods we use to interact with our computer: keyboard, mouse/touchpad, joystick, etc.

Here is the code for using the computer keyboard to control ChucK:

```
Hid myHid; // the object to receive the HID data
HidMsg msg; // the data from myHid will get stored in a HidMsg
myHid.openKeyboard (0); // the zero indicates device #0;
        // you may have multiple devices,
        // and you may need to try 1, or 2, etc
while (true) // repeat forever
{
 myHid => now; // wait for keyboard data
 while (hi.recv (msg)) // receive key message
 {
  if (msg.isButtonDown()) <<< "down:", msg.which >>>; // which key
   else <<< "up:", msg.ascii >>>; // ascii is the numeric code
 }
}
```

The ChucK Manual has good examples on using MIDI events as well.

3 Open Sound Control

Open Sound Control, or OSC, is a way to send messages between ChucK and Max, or lots of other programs. You can even send messages to other computers connected on the same wireless network.

OSC is sort of like MIDI, but with a lot of advantages:

- It's faster
- It's wireless capable
- It's capable of carry different kinds of messages

To send an OSC message, you need to know a few things:

1. The IP address of the target computer. (In most cases, you're sending data to yourself, so you use the address "localhost").
2. A port number. This is a number you pick that will be a unique pathway for your data. You can safely choose any number between 10000 and 65535.
3. A message ID or tag. These start with a slash and are descriptive: /chuck or /notedata
4. The data types you're sending. Are you sending two floats? Two ints and a string?
5. And finally, of course, your data!

oscsend: localhost 6449 /demo ff 20.5 0.33

4 Receiving OSC in ChucK

```
OscRecv osc;
6449=> osc.port;
osc.listen();
osc.event("/demo, ff") @=> OscEvent myOSC;

while (true)
{
  myOSC => now;
  while ( myOSC.nextMsg() != 0 ) // looks for multiple msgs
  {
   myOSC.getFloat() => float val1;
   myOSC.getFloat() => float val2;
  }
  // here, do something with the data you received
}
```

Try the webcam example on Blackboard.