MUS 4711 –
Interactive Computer Music

Course Lecture Notes

MODULE 3: Interactivity

PART 12: HID and OSC

Objects – [hi], [udpreceive], [mira.status], [mira.frame], [textedit], [OSC-route] (CNMAT external)

HI Devices – This is specific to the Sn30pro+ Controller from 8bitdo.com – pretty great device (especially for Smash Brothers…). [hi] allows the input from human interface devices – a VERY large category that can encompass joysticks, drawing tablets, foot pedals for racing games or flight simulators, etc. Send [hi] the (menu) message connected to a [umenu] to get a list of compatible devices attached to your computer. Set a polling interval, and it outputs the value of the button selected and its value as a list. This lets us use route to select things like directional values, or specific buttons, before sending them to [sel] to test for an on/off toggle, or a range of values in the case of analog sticks/triggers. Note that there is a [gate] in place to determine which set of [route] and [sel] objects are used. This is because the same joystick will have VASTLY different values on Mac and PC.

To get the values for a HID easily, go to the HI Tester under the Extras menu, select a device, turn on polling, and start hitting buttons. Again, be sure to keep in mind that Mac and PC will have different values, so you should get both if you want your patch to be available to both.

Analysis of the *GrainSieve* instrument from *Particle Forge* and the *Learn to Play* patch

Touch OSC – TouchOSC is a super useful iOS and Android application that creates a virtual OSC (Open Sound Control – like MIDI except faster, way higher resolution floats normalized 0 – 1, and it can be controlled over a local network or the internet) faderboard, keyboard, toggles, etc. It will also route your phone's accelerometer data to MaxMSP as a bonus. It's in the AppStore and on Google Play for a whopping $5, and the layout editor, manual, and MIDI bridge plugin for your computer are available at hexler.net.

To use TouchOSC, you need to enter the IP address of your target computer into the TouchOSC app. To get that IP address, use the [p getIP] subpatch – it uses [mira.status] from the Miraweb library to get the computer's IP and output it through [sprint], the C language print command, which formats the message. This is then output of the subpatch to the [textedit] – a container for text. Get the IP, and enter that in TouchOSC.

TouchOSC always sends OSC on port 8000, so in MaxMSP, we need to set it up to listen on that port. [udpreceive] takes in data over a network. Since we're on the same network, we don't need to specify a host (IP address), but we do need to input the port. Just set it up as [udpreceive 8000], and we should be getting some data from the app. I'm using the Mix2 layout, only the first page though. So, we'll use the CNMAT library's super useful [OSC-route] to route the OSC messages. It's just like [route], but it supports characters like /, multiple depth levels, and can handle wild cards. The first of these routes by /1 /2 /3, which corresponds the pages in the layout. The second [OSC-route] is used to route the faders to [gain~], top rotaries to filter cutoff, middle rotaries to filter gain, and the bottom rotaries to Q. The final horizontal fader controls panning via the [cosPan] abstraction.

All values from OSC are floats normalized from 0 – 1, so the [scale] objects are needed to translate them to the range needed by [filtergraph~].

Envelope Tracking – Here's a handy little abstraction from the Cycling '74 forums by Steven Miller (no, not the Nazi…). It takes an audio signal and uses [abs~] to get rid of negative values. Then it uses [slide~] to filter the signal and track its amplitude. Since [slide~] takes samples, and not MS, the [mstoamps~] object is used to convert *at the current sample rate* – really good feature as that can change! Finally, the [patcherargs] object autoloads initial values for an abstraction that load with the patch. Good way to set initial values you'll be using a lot.

Max Puzzle 8 – Time for a challenge! Using a game pad and OSC, create a patch that plays a random note when a button is pressed on the game pad and passes the audio to an [amxd~] device that changes it. The [amxd~] should be controlled by OSC messages from TouchOSC or JockeyOSC. In either case, include the OSC layout that you're using in the folder you upload.