Creativity Through Interaction: 10 projects using open-source software

Justin Yang
2014-09-04

Contents

PROJECT 1 - MOUSE THEREMIN

1 PROJECT1 - MOUSE THEREMIN

```
With a few lines of code, we build a pretty useful and infinitely expandable
digital instrument. All of the ingredients for digital performance are here,
instrument, interface and score.  It'll introduce us to basics concepts in
coding, synthesis, SuperCollider, Processing, and give us a basic template
where the sky's the limit.
```

1.1 Start At The End

EXPLORING IN THIS SECTION

```
—> Following Instructions
—> Wrist & Finger Dexterity
—> Mouse Accuracy & Speed
—> Digital Performance
```

Run the Code: Copy the SC code below into SuperCollider, or you can download it from here:

https://github.com/elosine/Mouse_Theremin/archive/master.zip

1. Run it as indicated.
2. See if you can figure out what to do without any further guidance.
3. Did you play a tune? Which one? My favourite is the Godfather theme.

```
//PLACE MOUSE CURSOR ON EACH OF THE NUMBERS IN ORDER AND HIT CMD-RETURN

(
// 1):
// Initialize
var currMidiNum, noteNames, octaves, screenRes, yres;
~pitchLo = \a0;
~pitchHi = \c8;

//MIDI NOTE DICTIONARY
////Allows you to input a symbol and get out the midi note number
////Use only these symbols:
//// \c, \cs, \d, \eb, \e, \f, \fs, \g, \gs, \a, \bb, \b

~mnd = ();
currMidiNum = 0;
noteNames = [\c, \cs, \d, \eb, \e, \f, \fs, \g, \gs, \a, \bb, \b];
octaves = [-1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
octaves.do{ |it ix|
        noteNames.do{ |it2 ix2|
                ~mnd.put((it2 ++ it).asSymbol, currMidiNum);
                currMidiNum = currMidiNum+1;
        }
};
```

```
//FUNCTION TO LOOK UP Y-LOCATION FOR A PITCH

screenRes = "system_profiler SPDisplaysDataType | grep Resolution".
    unixCmdGetStdOut.findRegexp("(?<!@ )[0-9]{3,}").collect({|item| item
    [1].asInteger}).clump(2);
~yres = screenRes[0][1];

~midiToY = {
        arg midinote;
        ~mnd[midinote].linlin(~mnd[~pitchLo].asFloat, ~mnd[~pitchHi].
            asFloat, ~yres, 0.0).round;
};

//SYNTHDEF
SynthDef(\mouseTheremin, {
        arg pitchLo = 30, pitchHi = 80, pitchres=1.0;
        var osc, freq, env, envgate, pitch, amp;
        amp = MouseX.kr(-40, 10).dbamp;
        pitch = MouseY.kr(minval:pitchHi, maxval:pitchLo, lag:0.0001).round
            (pitchres);
        envgate = MouseButton.kr(lag:0.0001);
        env = EnvGen.ar( envelope:Env.asr(releaseTime:0.05), gate:envgate )
            ;
        freq = pitch.midicps;
        osc = SinOsc.ar( freq );
        Out.ar(0, osc*env*amp);
}).add;

) ////////////////////////////////////////////////////////////

(
// 2):
// Boot Server
s.boot;
)

(
// 3):
// Start Synth
~mt = Synth(\mouseTheremin, [\pitchLo, ~mnd[~pitchLo], \pitchHi, ~mnd[
    ~pitchHi]]);
)


// TO STOP SYNTH OR USE CMD-.
~mt.free;



//UTILITIES

//EXAMPLE OF MIDI NOTE TO Y COORDINATE TO LOOK UP VALUES FOR YOUR
    PROCESSING PATCH:
////Use only these symbols:
//// \c, \cs, \d, \eb, \e, \f, \fs, \g, \gs, \a, \bb, \b
```

```
~midiToY.value(\g5);
~midiToY.value(\fs4);
//Look @ Post Window


//USE THESE TO 'PLAY ALONG' AND THEN FIND MIDI PITCH
~mspoll = {MouseY.kr(~mnd[~pitchHi], ~mnd[~pitchLo]).round.poll}.play;
~mnd.findKeyForValue(66);

~mspoll.free;

/*
//GANGSTER'S PARADISE, MAIN RIFF
84 - C6 - 298
83 - B5 - 310
79 - G5 - 360
*/
```

1.2 Make A Simple Score

EXPLORING IN THIS SECTION
—> Basic Drawing w/ Processing
—> Using SuperCollider Lang as a Calculating Environment

Let's make a simple graphic score that will help us play the riff from Coolio/Stevie Wonder's 'Gangster's Paradise'. Copy the following code into Processing or download from here (use 'Coolio1):

https://github.com/elosine/coolio/archive/master.zip

```
int topmargincompinsation = -45;
void setup(){
  size(1920, 1080);
  smooth();
}

void draw(){

  stroke(0);
  line(0, 298+topmargincompinsation , width, 298+topmargincompinsation);
  line(0, 310+topmargincompinsation , width, 310+topmargincompinsation);
  line(0, 360+topmargincompinsation , width, 360+topmargincompinsation);

}
```

If you select 'Run' from the Sketch menu in the Processing IDE, or press CMD-R your screen should fill with a grey canvas with lines running through. Now if you point to the lines with the mouse, you might be playing the pitches from 'Gangster's Paradise'. However, you may have a different screen resolution so let's take this opportunity to learn some Processing basics and make a score fit to your screen.

First, adjust the size parameters in your code to the resolution of your screen. Simply change the arguments in the size function to the width and height of your screen. CMD-R to run again and you should have a canvas the size of your screen:

```
  void setup(){
    size(1920, 1080);
  smooth();
}
```

Example1: Change Canvas Size

Now, we may have to adjust the location of the lines on your score. I've written a couple of utilities to help with this. We don't need to worry about how they work right now, though I find it is always helpful to try to figure mysterious code out. Go back to your SC code and find the bit of code that looks like this:

```
//EXAMPLE OF MIDI NOTE TO Y COORDINATE TO LOOK UP VALUES FOR YOUR
    PROCESSING PATCH:
////Use only these symbols:
//// \c, \cs, \d, \eb, \e, \f, \fs, \g, \gs, \a, \bb, \b
        ~midiToY.value(\g5);
        ~midiToY.value(\fs4);
//Look @ Post Window
```

We are actually just interested in the function: midiToY.value(5). At the bottom of the SC code you'll find the notes to our riff: c6, b5, g5. Now just fill these notes into the function, place your cursor on the line, and press CMD-RETURN, and look at your post window and you should get a number, these will be the y coordinates that correspond to the pitches for your mouse theremin.

```
        ~midiToY.value(\c6);
        ~midiToY.value(\b5);
        ~midiToY.value(\g5);
```

Once you get the proper coordinates, go back to your Processing code and fill in the coordinates. Where it says 298, fill in the number you got for c6, b5 goes in 310, and g5 replaces 360:

```
  line(0, 298+topmargincompinsation , width, 298+topmargincompinsation);
  line(0, 310+topmargincompinsation , width, 310+topmargincompinsation);
  line(0, 360+topmargincompinsation , width, 360+topmargincompinsation);
```

`http://www.processing.org`