

# Imaginative Python

Morgan Visnesky



# What is Art?

## Dictionary

art<sup>1</sup>

/ärt/

*noun*

1. the expression or application of human creative skill and imagination, typically in a visual form such as painting or sculpture, producing works to be appreciated primarily for their beauty or emotional power.

"the art of the Renaissance"

Similar:

fine art

artwork

creative activity

# What is Art?

## Dictionary

**art<sup>1</sup>**

/ärt/

*noun*

1. the expression or application of human creative skill and imagination, typically in a visual form such as painting or sculpture, producing works to be appreciated primarily for their beauty or emotional power.

"the art of the Renaissance"

Similar:

fine art

artwork

creative activity

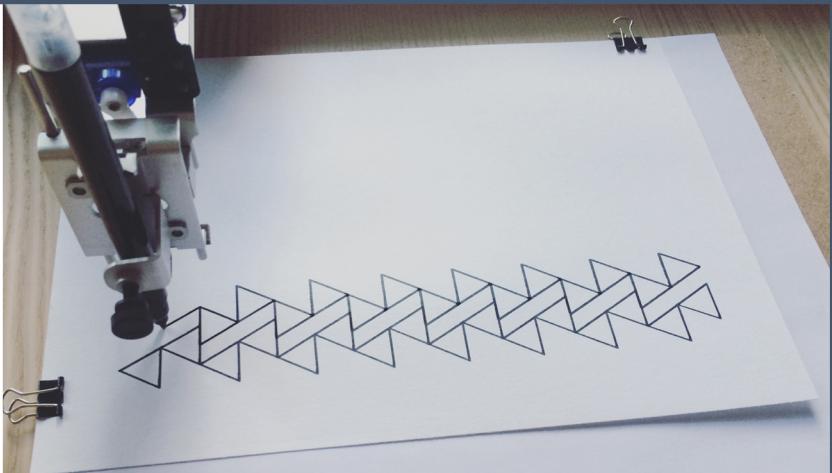
# Process Oriented Art

- “emphasizes or focuses on **processes**, systems, or procedures rather than results or underlying causes” – Oxford Dictionary
- Algorithmic art?

Dictionary

Search for a word  

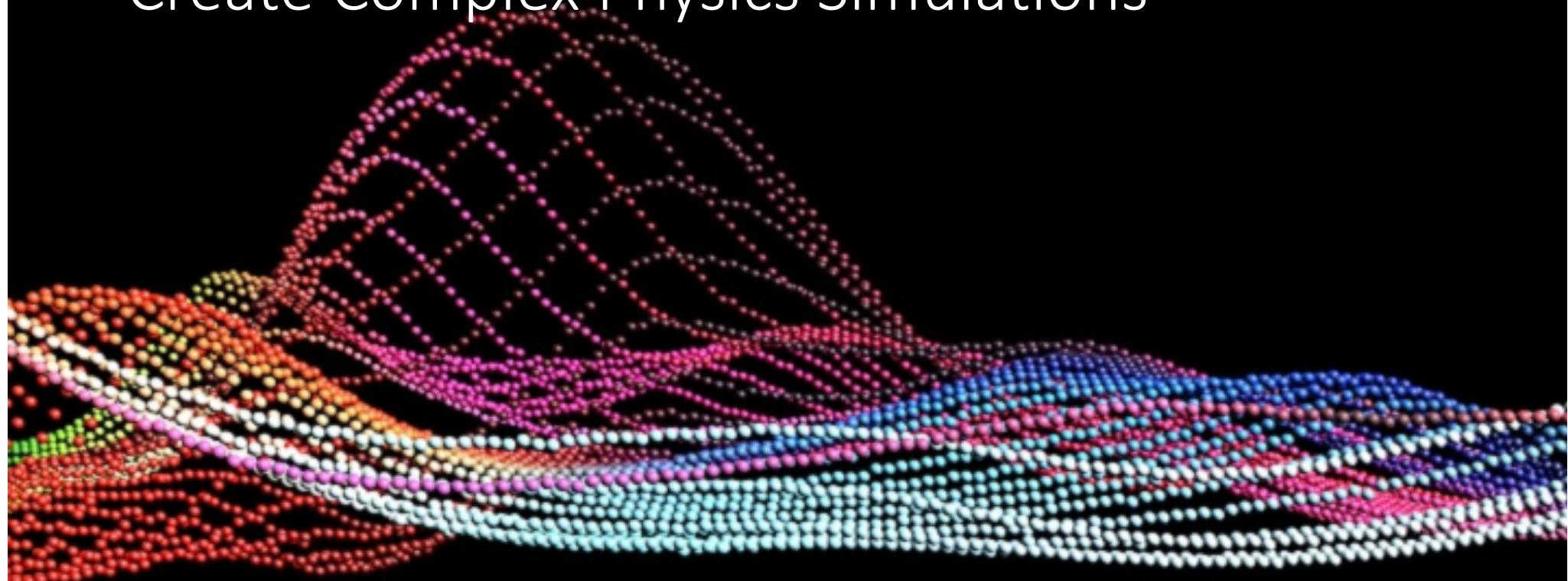
 **al·go·rithm** /'algə,riTHəm/  
*noun*  
a process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer.  
"a basic algorithm for division"



A black and white photograph of a person from the chest up. They are wearing round-rimmed glasses and a dark, possibly black, hoodie. Their head is bowed, and they appear to be looking down at a laptop computer which is partially visible at the bottom of the frame. The background is dark and out of focus.

SO WHATS POSSIBLE?

# Create Complex Physics Simulations



# Creatively Visualizing Data sets



## Small Arms and Ammunition – Imports & Exports

An interactive visualization of government-authorized small arms and ammunition transfers from 1992 to 2010.

UNITED STATES

SEARCH

ABOUT

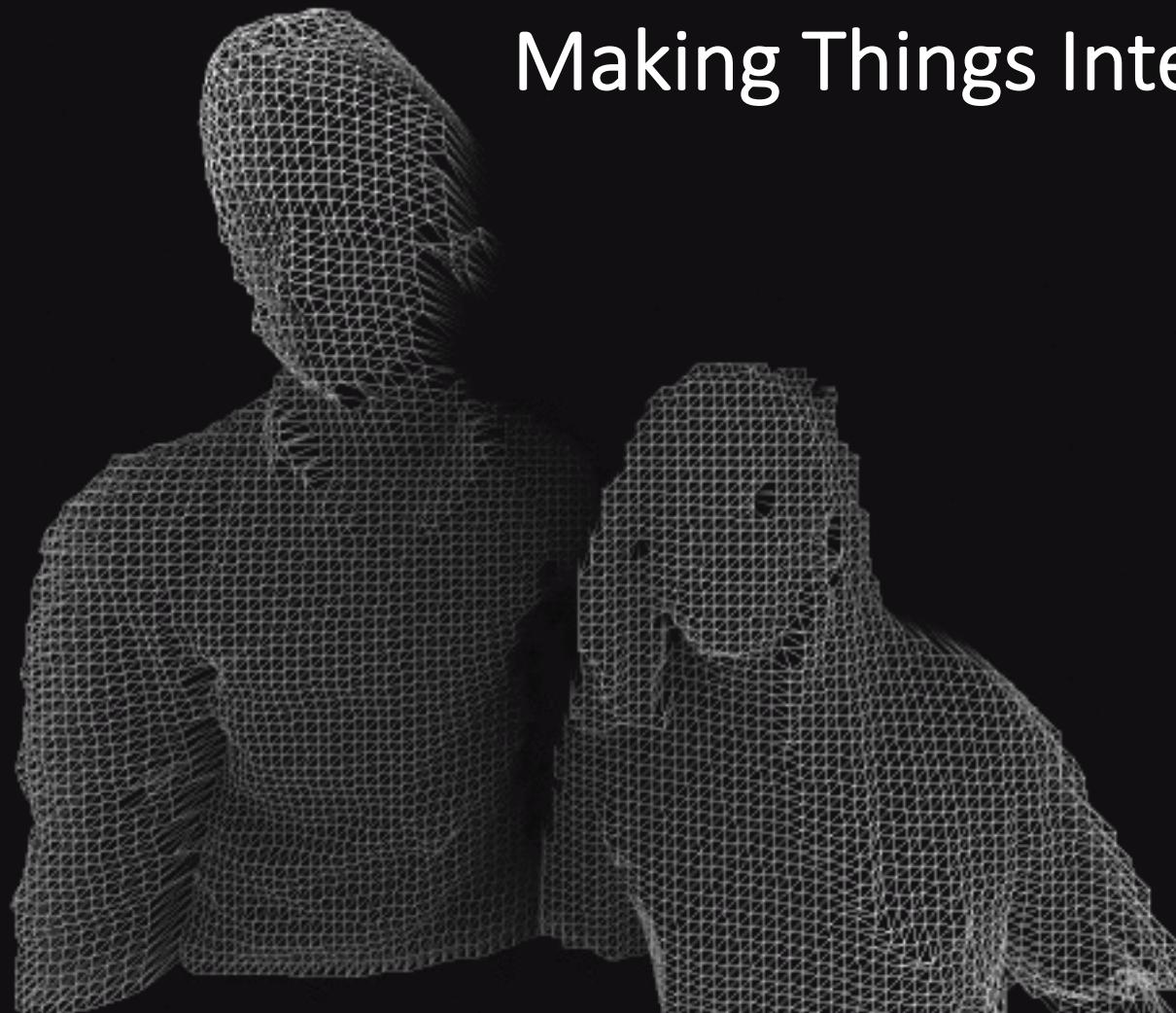
- Small Arms Imports/Exports: an interactive visualization of government-authorized small arms and ammunition transfers from 1992 to 2010. Google idea (2012). ([chromeexperiments.com](http://chromeexperiments.com))



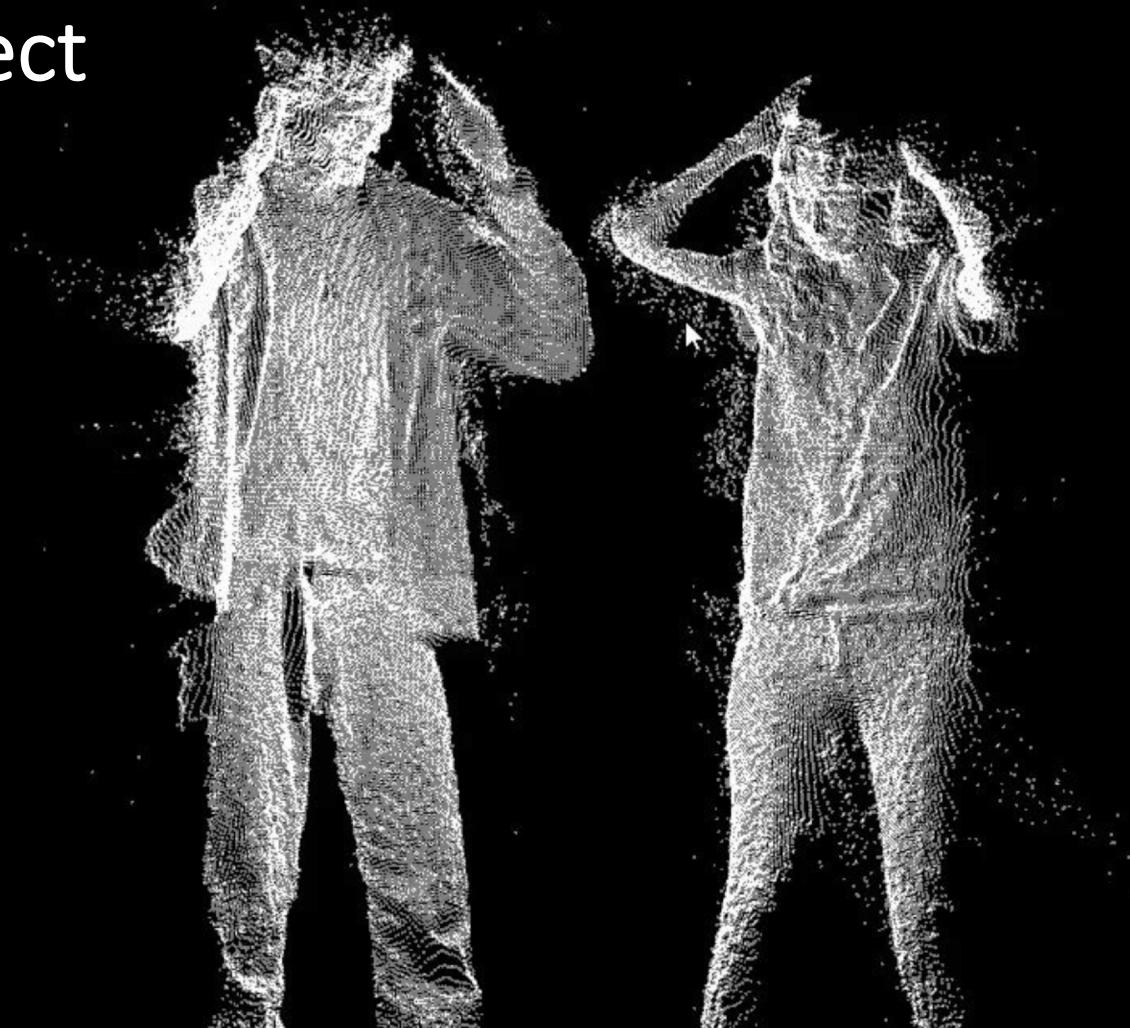
# Concepts that translate

- .OBJ file export/import supported by processing and blender along with a ton of other 3D design software.
  - Can also use these with 3D printing
- Graphics math and programming is really similar across environments.
- Interactivity
- OpenGL / GLSL

# Making Things Interactive



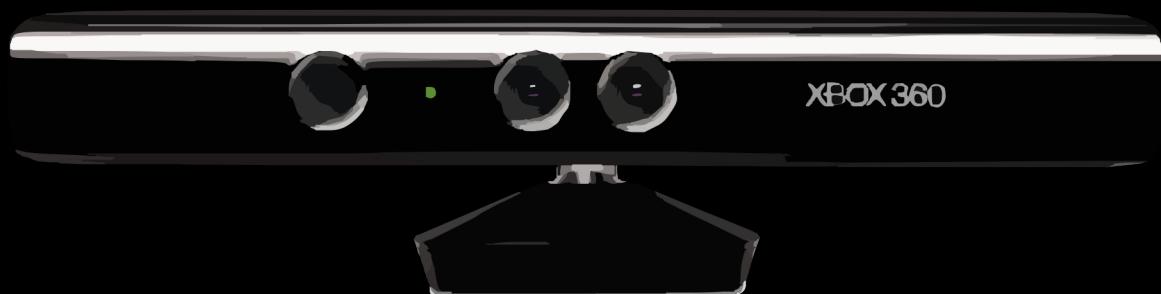
# OpenKinect



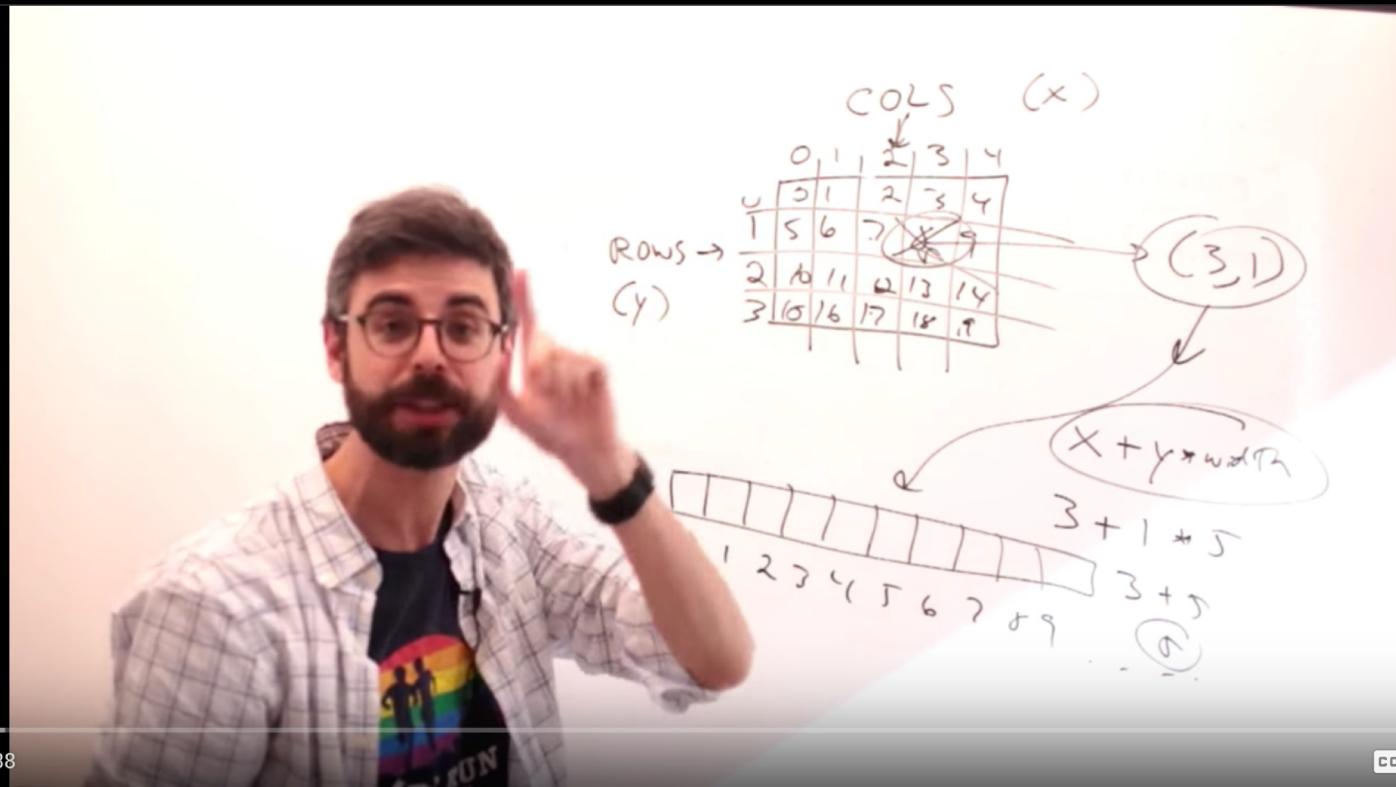
59.808

# Point cloud Data From Xbox Kinect

- Processing.py
- Python
- Blender
- Maya
- TouchDesigner
- OpenFrameworks
- Unity
- Cinema 4D



# Point Cloud Data Explained



▶ ▶ 🔍 1:40 / 16:38

CC HD 🔍 ⏹ ⏷

12.3: Raw Depth Data - Point Clouds and Thresholds - Kinect and Processing Tutorial

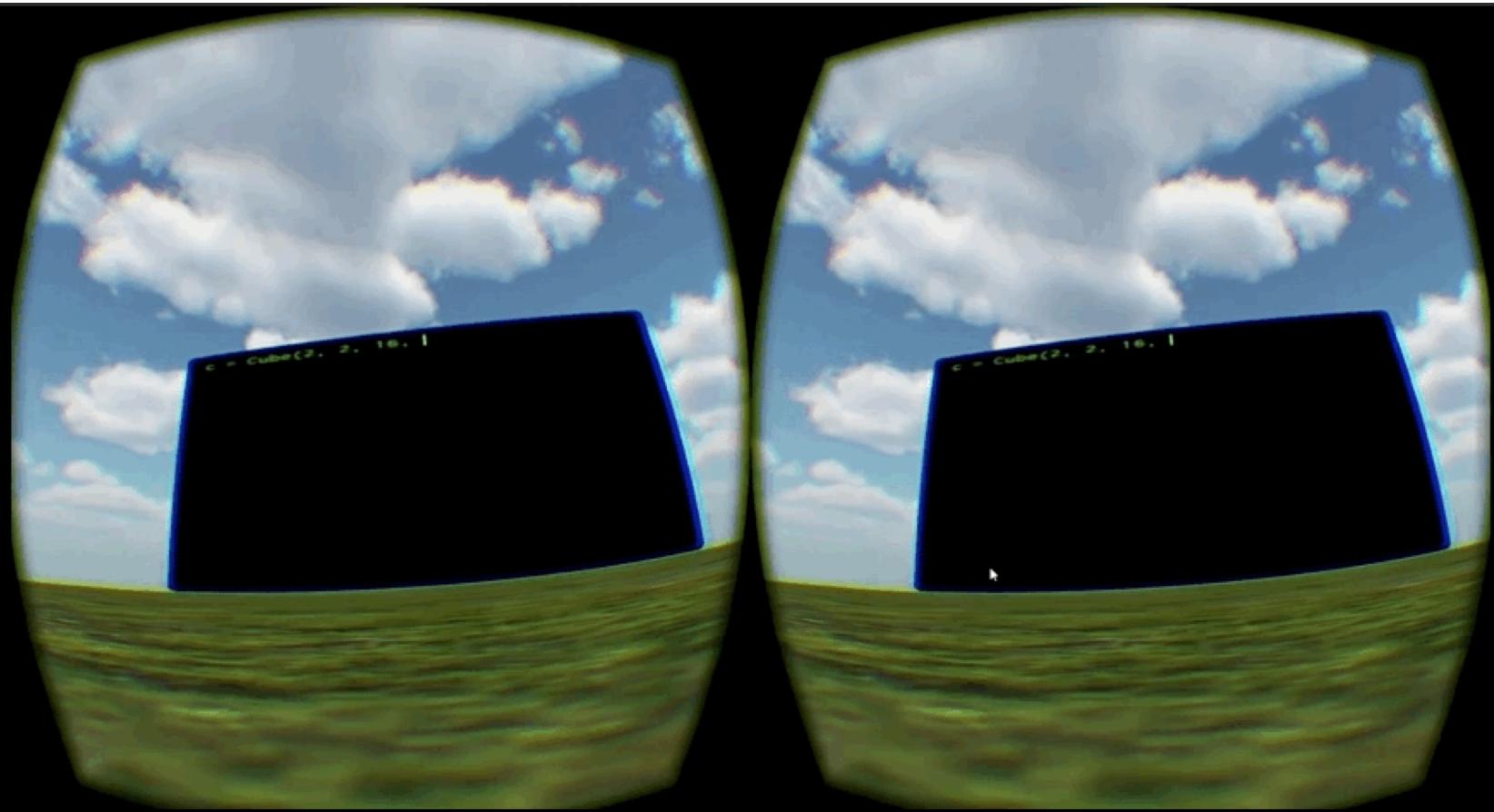


# VR with python

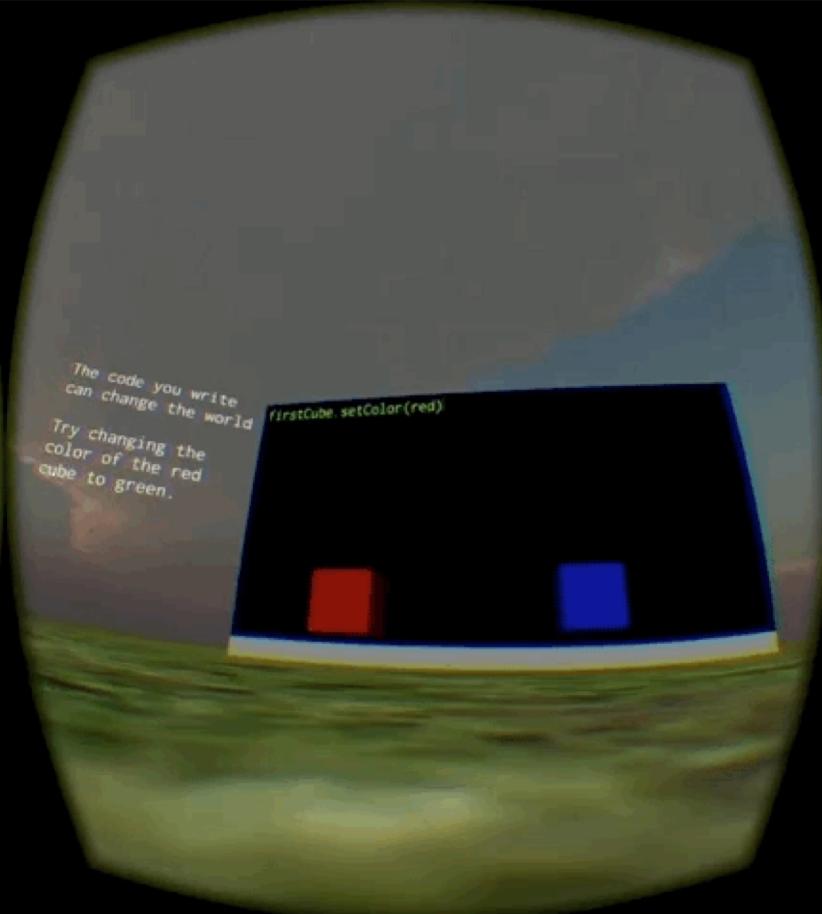
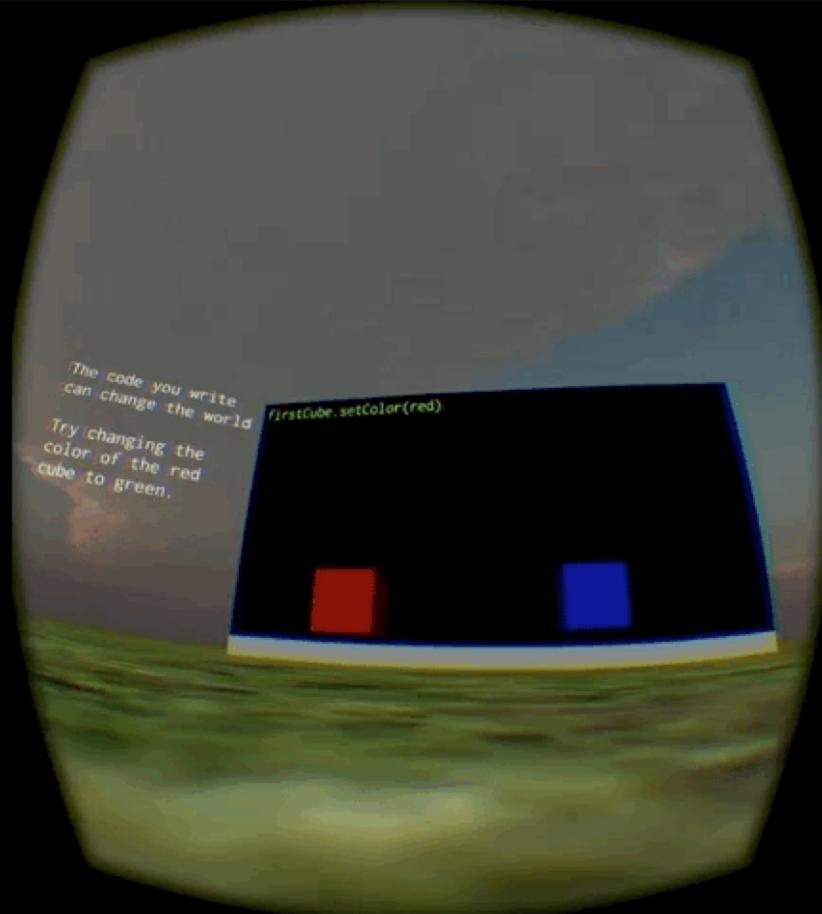
- Blender
- Maya
- OpenGL

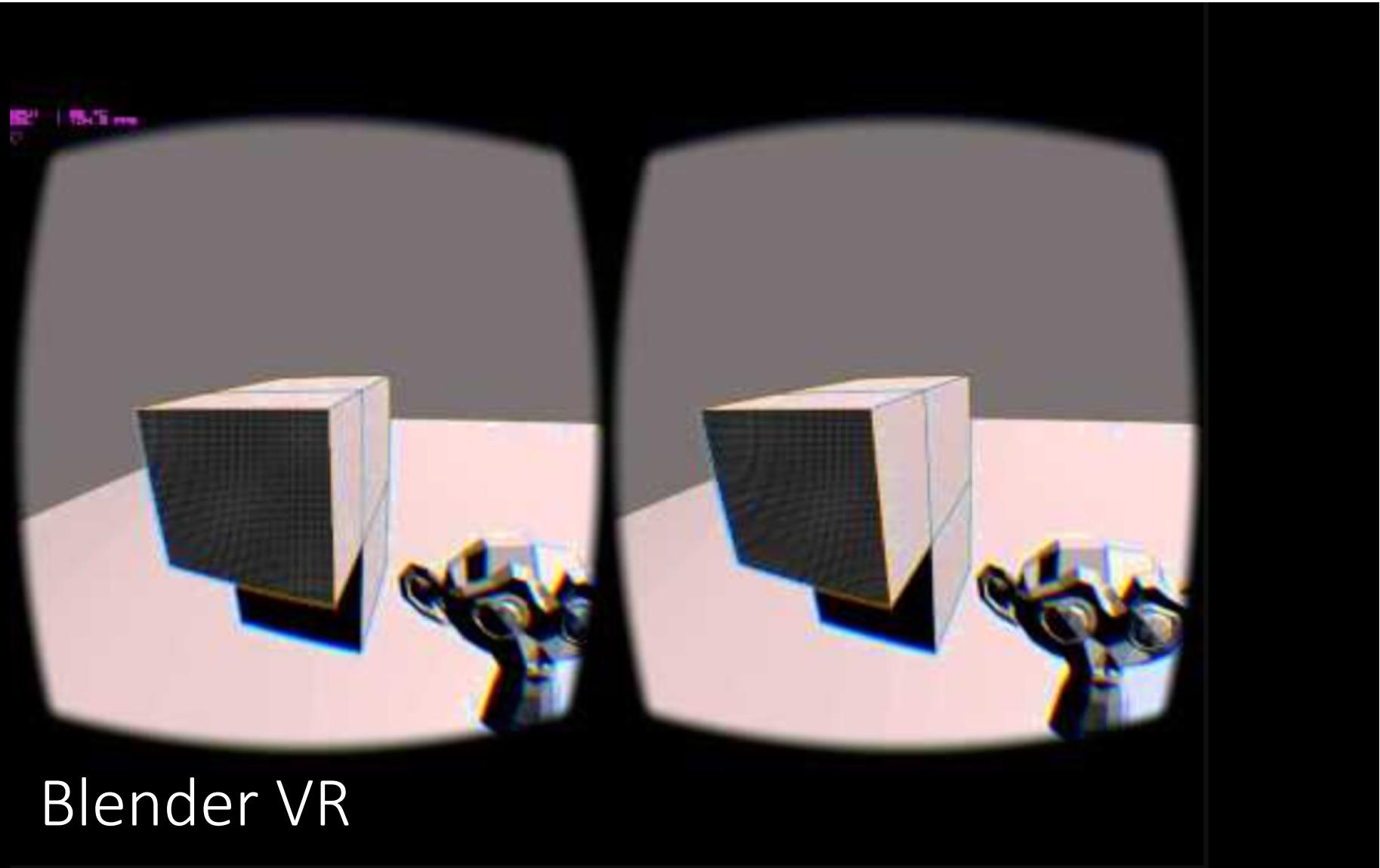
## NOT IN PYTHON

- Processing.vr for android
  - Uses Google VR
  - \*this only supports Java ATM\*
- Unity, uses C# (I frequently use python to send control signals to this)

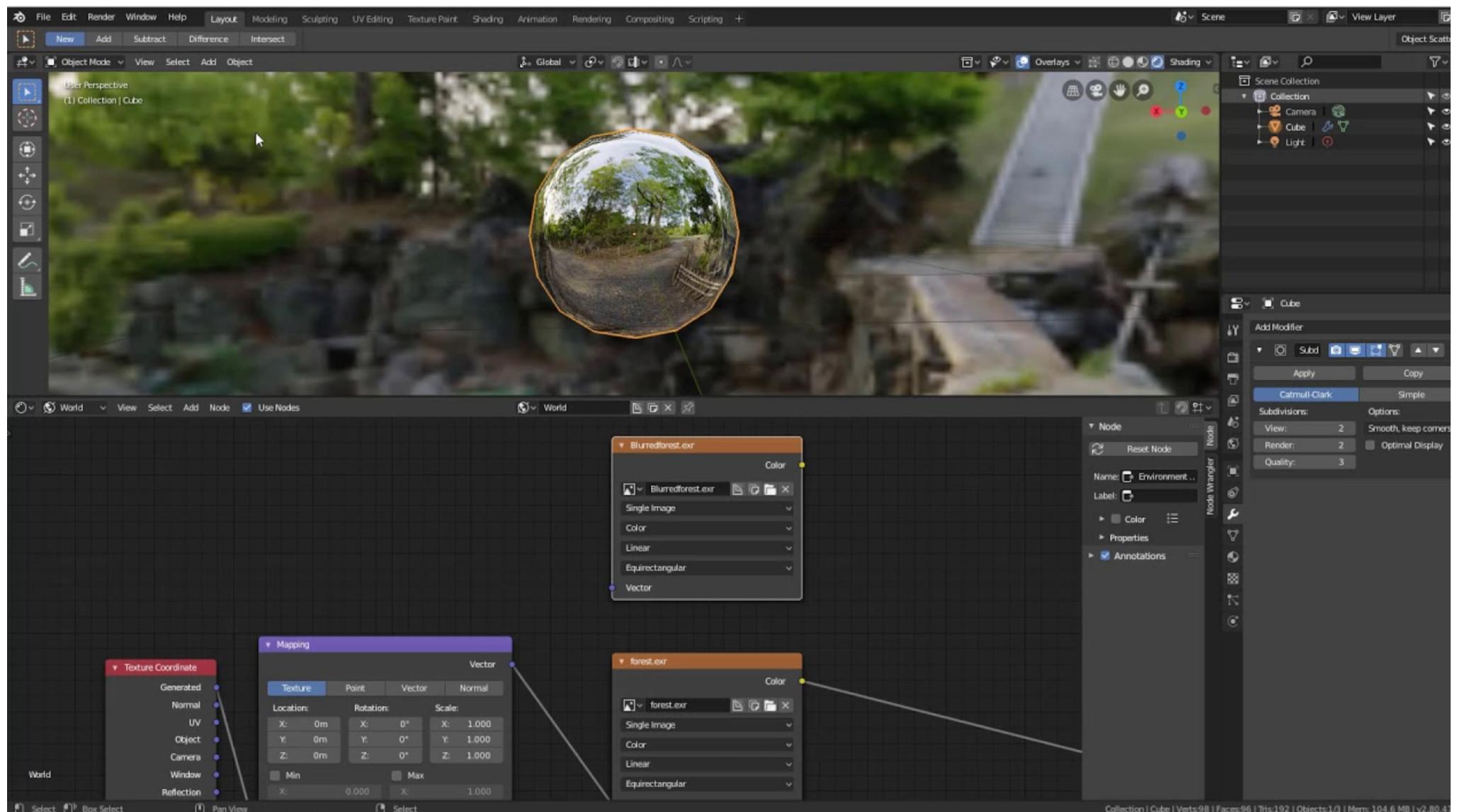


Rachel Friedman: Full python unity implementation, able to see variable of an object by looking at it.





Blender VR

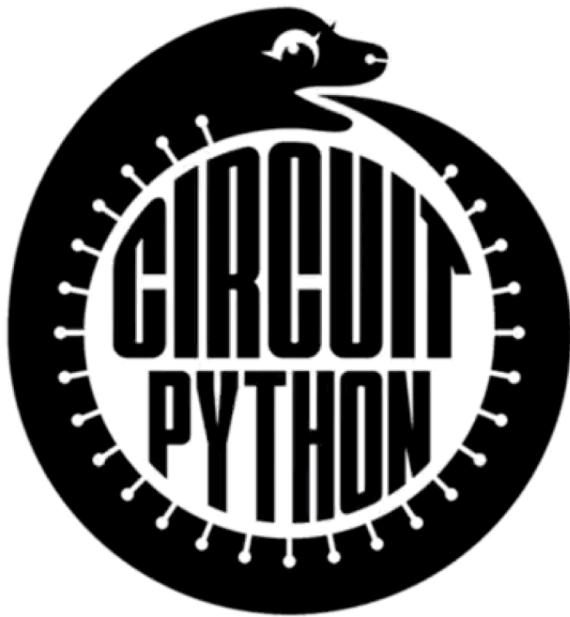


# Interactive: Bot art

- Input data into your project from bot interactivity.
  - Webserver that outputs art based on:
    - user input
    - Many different types of twitter data
- Use tools like:
  - Natural language processing & sentiment analysis
  - Social media and other web API's
    - Twitter
    - Slack
    - Robinhood (Stock Market Data)
    - OpenWeatherMap

# Interactive: CircuitPython

## CIRCUITPYTHON



CircuitPython is Adafruit's branch of MicroPython designed to simplify experimentation and education on low-cost microcontrollers. No compiler, linker or IDE required! Now you can code directly on the device itself. CircuitPython makes it easier than ever to get prototyping by requiring *no upfront desktop software downloads* - open up `main.py` in any text editor and type away.

**It is great for beginners:** With CircuitPython you can write clean and simple Python code to control hardware instead of having to use complex low-level languages like Arduino, C or C++. The simplicity of the Python programming language makes CircuitPython an excellent choice for beginners who are new to programming and hardware.

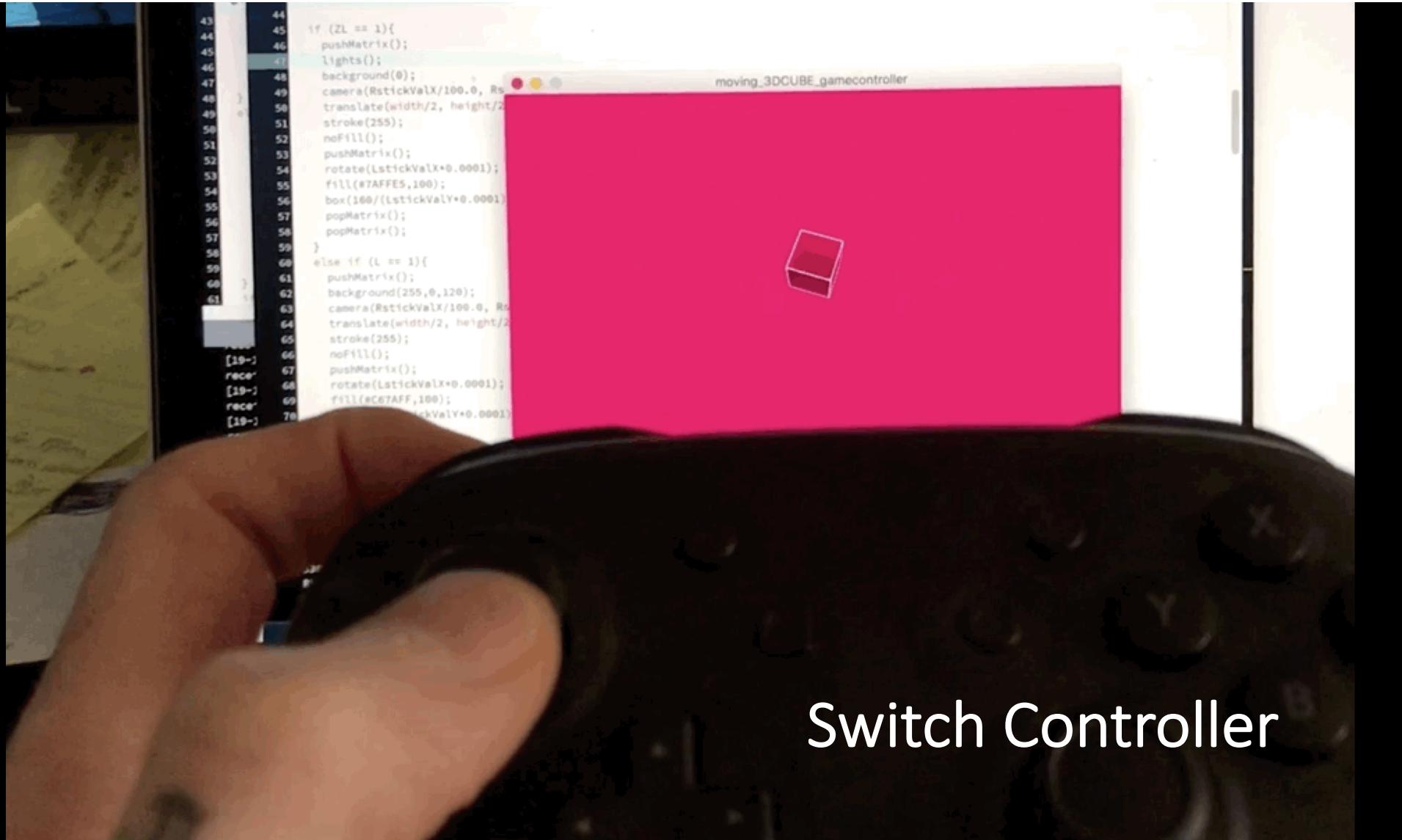
**Experts love it too:** skip the annoyances of long-compile cycles, pointers and memory management - get right to the programming you want to do. CircuitPython is also quite full-featured and supports all of Python's syntax (Python version 3.4) and implements some useful parts of the Python standard library so even seasoned Python veterans will find CircuitPython familiar and fun to use.

Pick up a Circuit Playground Express if you want everything-and-the-kitchen-sink-too, Gemma M0 for small wearables, Trinket M0 for compact breadboard-friendly designs, Metro M0 Express for shield-compatibility or Feather M0 Express to take advantage of the Feather system

# Raspberry pi

- Python libraries to leverage GPIO pins for electronics projects
- Runs processing and Blender
- Easy device to use for artistic installations
- Can update and interact with wirelessly





Switch Controller

# Game Controller Setup



Game Controller

Connects to pi  
via bluetooth



Raspberry pi  
Running evdev script

Sends converted  
messages as UDP



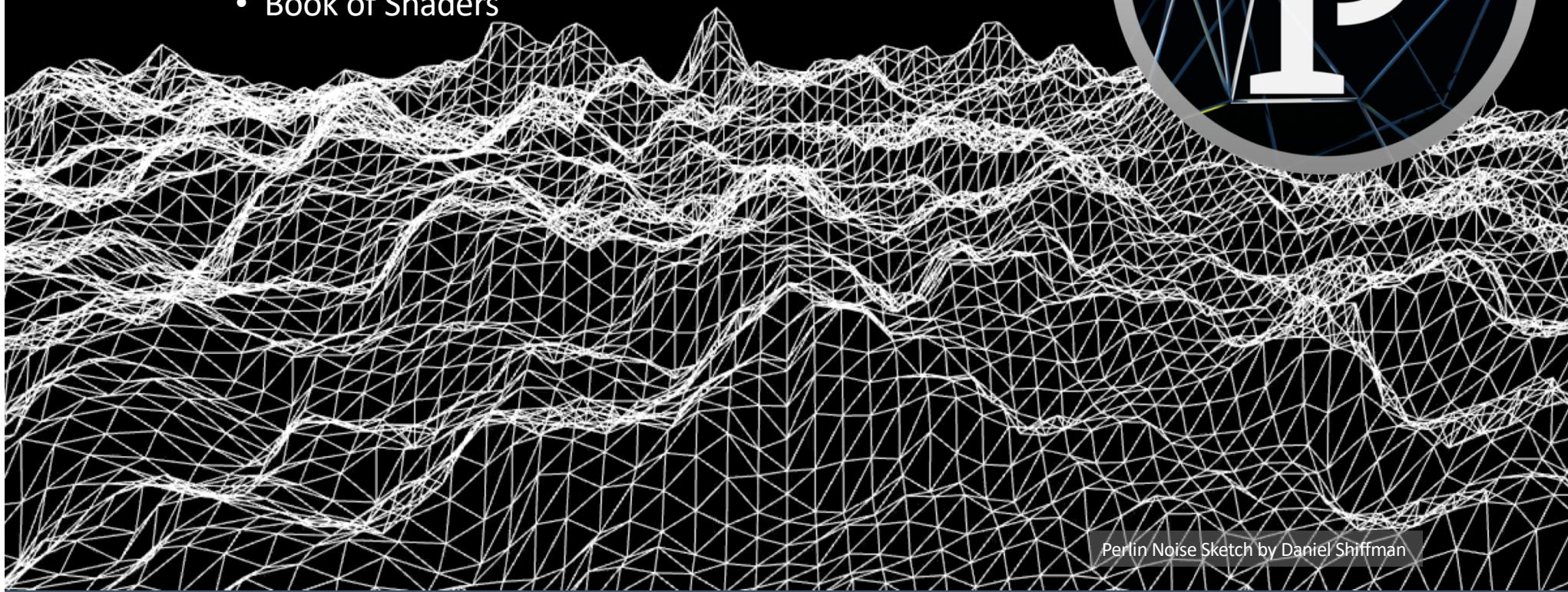
Macbook Pro  
Running processing

Parses bluetooth  
messages with  
evdev & performs  
conversion

Performs logic  
based on UDP

## processing.py

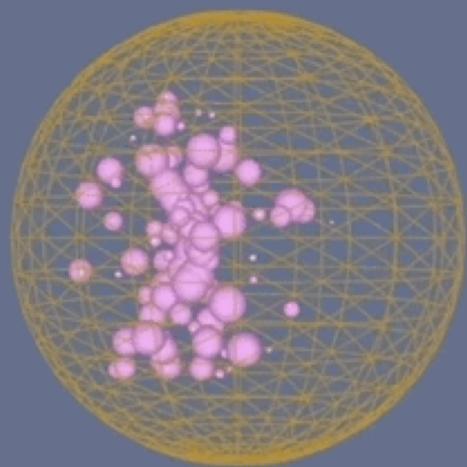
- Python mode available (some limitations).
- Daniel Shiffman youtube tutorials
- Can create OpenGL wrapper to host GLSL shaders
  - Book of Shaders



Perlin Noise Sketch by Daniel Shiffman

## Things that wont work in processing.py:

- Numpy, scipy, and scikit-learn will not work in Jython, so will also be unavailable in processing.py.
- Libraries that depend on anything listed above will not work
  - \*\* A lot of these DO work in blender\*\*



sketch\_3DCircle attractor.py mover.py

Python ▾

```
3 def __init__(self):
4     self.position = PVector(0, 0)
5     self.mass = 20
6     self.g = 0.4;
7
8     def attract(self, m):
9         force = PVector.sub(self.position, m.position)
10        d = force.mag()
11        d = constrain(d, 5.0, 25.0)
12
13        force.normalize();
14        strength = (self.g * self.mass * m.mass) / float(d * d)
15        force.mult(strength)
16
17        return force
18
19    def display(self):
20        stroke(199, 145, 44)
21        noFill()
22        pushMatrix()
23        translate(self.position.x, self.position.y, self.position.z)
24        sphere(self.mass*8)
25        popMatrix()
```

The image shows two side-by-side code editors in the Processing IDE. The left editor is set to 'Java' mode and contains the Java code for the 'cube\_of\_cubes\_offset' sketch. The right editor is set to 'Python' mode and contains the corresponding Python conversion for the same sketch. Both code snippets are identical, demonstrating how to create a 3D cube structure using nested loops and matrix operations.

```
cube_of_cubes_offset
```

```
int OFF_MAX = 200;  
void setup() {  
    size(900,720,P3D);  
}  
  
void draw() {  
    background(0);  
    translate(width/2, height/2, -OFF_MAX);  
    rotateX(frameCount * .01);  
  
    for (int x = -OFF_MAX; x <= OFF_MAX; x += 50){  
        for (int y = -OFF_MAX; y <= OFF_MAX; y += 50){  
            for (int z = -OFF_MAX; z <= OFF_MAX; z += 50){  
  
                pushMatrix();  
                translate(x,y,z);  
                fill(colorFromOffset(x), colorFromOffset(y), colorFromOffset(z));  
                box(30);  
                popMatrix();  
            }  
        }  
    }  
  
    int colorFromOffset(int offset){  
        return (int) ((offset + OFF_MAX)/(2.8 * OFF_MAX) * 255);  
    }  
}
```

```
pythonboxofcubes
```

```
# Python conversion 'box of cubes sketch'  
offset = 200  
  
def setup():  
    size(900,720,P3D)  
  
def draw():  
    background(0)  
    translate(width/2, height/2, -offset)  
    rotateX(frameCount * .01)  
    for x in range(-offset,offset,50):  
        for y in range(-offset,offset,50):  
            for z in range(-offset,offset,50):  
                pushMatrix()  
                translate(x,y,z)  
                fill(colorFromOffset(x),colorFromOffset(y),colorFromOffset(z))  
                box(30)  
                popMatrix()  
  
def colorFromOffset(offset_c):  
    return ((offset_c + offset)/(2.8*offset)* 255)
```

Easy to convert existing examples and code from java into python.

```

cube_of_cubes_offset | ▾
1 int OFF_MAX = 200;
2
3 void setup() {
4   size(900,720,P3D);
5 }
6
7 void draw() {
8   background(0);
9   translate(width/2, height/2, -OFF_MAX);
10  rotateX(frameCount * .01);
11
12  for (int x = -OFF_MAX; x <= OFF_MAX; x += 50){
13    for (int y = -OFF_MAX; y <= OFF_MAX; y += 50){
14      for (int z = -OFF_MAX; z <= OFF_MAX; z += 50){
15
16        pushMatrix();
17        translate(x,y,z);
18        fill(colorFromOffset(x), colorFromOffset(y), colorFromOffset(z));
19        box(30);
20        popMatrix();
21      }
22    }
23  }
24}
25
26 int colorFromOffset(int offset){
27   return (int) ((offset + OFF_MAX)/(2.8 * OFF_MAX) * 255);
28}
29
30
```

Java

```

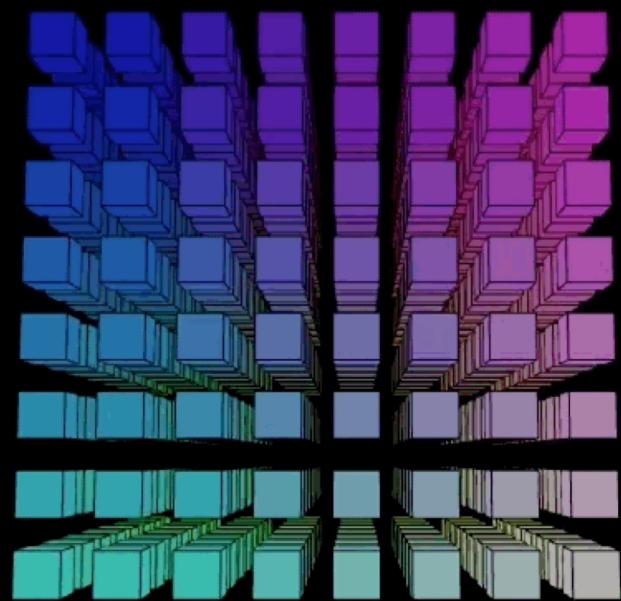
pythonboxofcubes | ▾
1 # Python conversion 'box of cubes sketch'
2
3 offset = 200
4
5 def setup():
6   size(900,720,P3D)
7
8 def draw():
9   background(0)
10  translate(width/2, height/2, -offset)
11  rotateX(frameCount * .01)
12  for x in range(-offset,offset,50):
13    for y in range(-offset,offset,50):
14      for z in range(-offset,offset,50):
15
16        pushMatrix()
17        translate(x,y,z)
18        fill(colorFromOffset(x),colorFromOffset(y),colorFromOffset(z))
19        box(30)
20        popMatrix()
21
22 def colorFromOffset(offset_c):
23   return ((offset_c + offset)/(2.8*offset)* 255)
24
25
26
```

Python

Done saving.

Easy to create java-like for loops using range function.

Both sketches have  
the same output.



FindContours | Processing 3.5.3

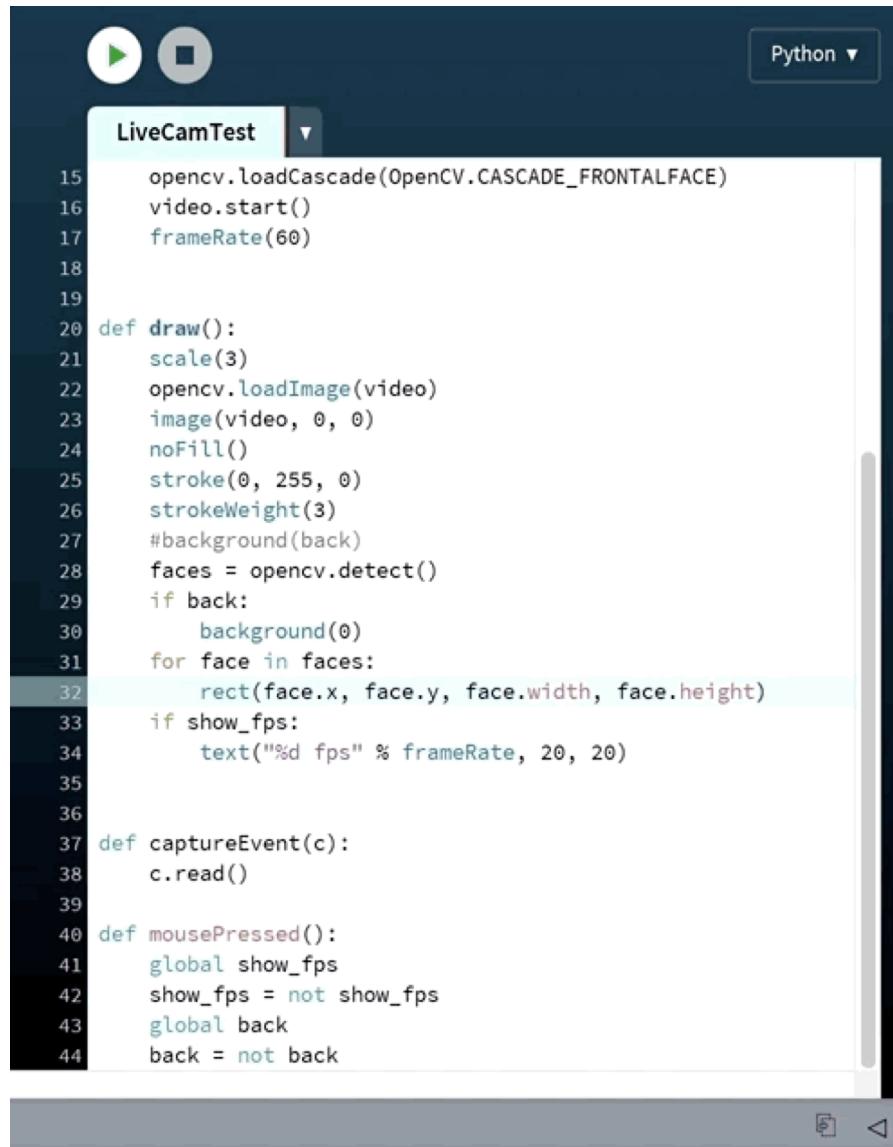
FindContours

OpenCV: Contour Detection

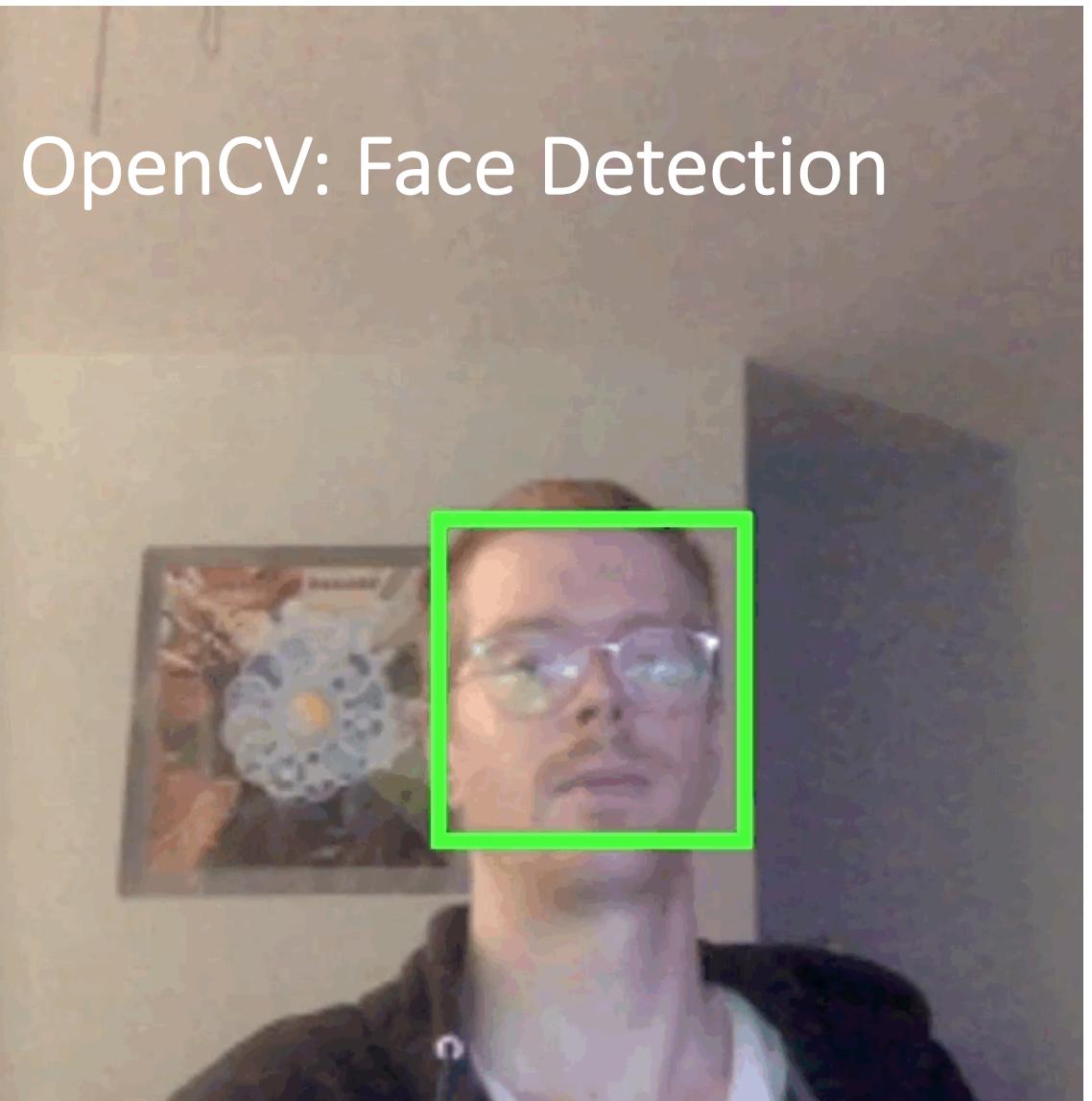
The image shows a screenshot of the Processing IDE. On the left, the code for 'FindContours' is displayed:

```
1 add_library('opencv_processing')
2
3 src = loadImage("test.jpg")
4 size(480,480, P2D)
5 opencv = OpenCV(this, src)
6 opencv.gray()
7 opencv.threshold(70)
8 dst = opencv.getOutput()
9 contours = opencv.findContours()
10 print "found %d contours" % contours.size()
11
12 scale(0.5)
13 image(src, 0, 0)
14 image(dst, src.width, 0)
15 noFill()
16 strokeWeight(3)
17
18 for contour in contours:
19     stroke(0, 255, 0)
20     contour.draw()
21
22     stroke(255, 0, 0)
23     with beginShape():
24         for point in contour.getPolygonApproximation().getPoints():
25             vertex(point.x, point.y)
26
27
```

On the right, the result of the code execution is shown. It displays a portrait of a man with red and green outlines highlighting the detected contours. A smaller inset image in the top-left corner shows a hand holding a smartphone displaying a similar contour-detection result.



```
LiveCamTest
15 opencv.loadCascade(OpenCV.CASCADE_FRONTALFACE)
16 video.start()
17 frameRate(60)
18
19
20 def draw():
21     scale(3)
22     opencv.loadImage(video)
23     image(video, 0, 0)
24     noFill()
25     stroke(0, 255, 0)
26     strokeWeight(3)
27     #background(back)
28     faces = opencv.detect()
29     if back:
30         background(0)
31     for face in faces:
32         rect(face.x, face.y, face.width, face.height)
33     if show_fps:
34         text("%d fps" % frameRate, 20, 20)
35
36
37 def captureEvent(c):
38     c.read()
39
40 def mousePressed():
41     global show_fps
42     show_fps = not show_fps
43     global back
44     back = not back
```



# Blender

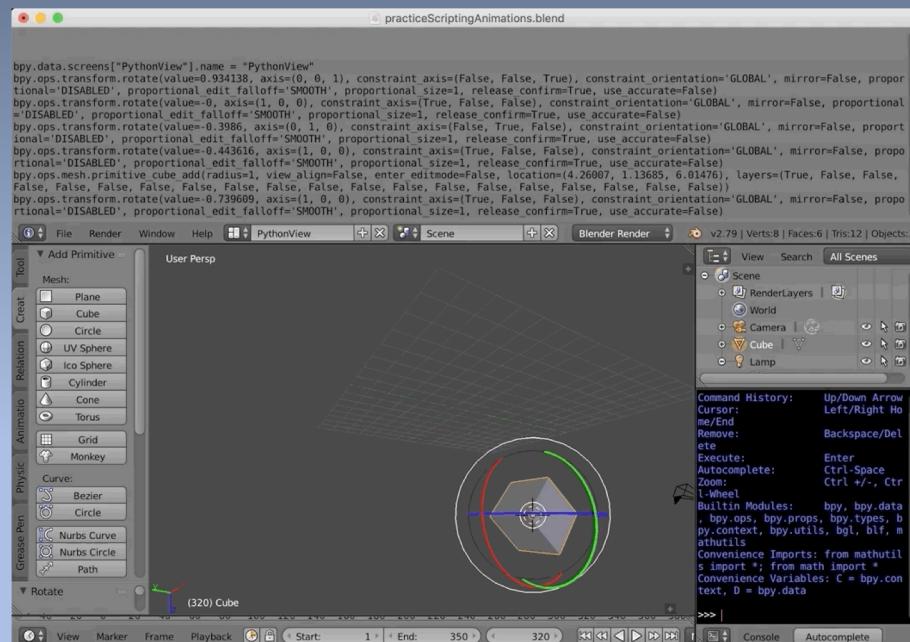
- Open source 3D design suite
- Jeremy Berheandt “Creative Coding in Blender: A Primer”
- “Code-Generator” feature
- Python API for BlenderVR



# Motion Capture in Blender with OpenCV

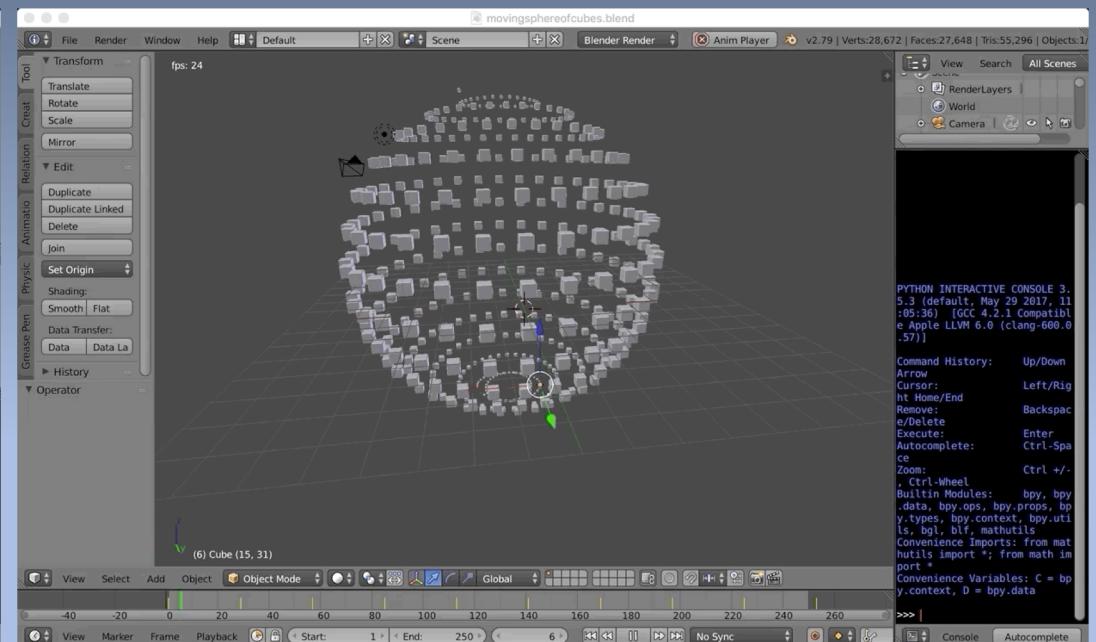


# What's possible in blender?



A screenshot of the Blender Python Viewport. The top bar shows the file name "practiceScriptingAnimations.blend". The main area displays a 3D scene with a single cube object selected. A circular path is drawn on the ground plane around the cube. The left side of the interface shows the "Tool Shelf" with various creation tools like Plane, Cube, Circle, etc. The right side shows the "Properties Shelf" with "Scene" settings. A Python console window at the bottom contains the following code:

```
 bpy.data.scenes["PythonView"].name = "PythonView"
 bpy.ops.transform.rotate(value=0.92418, axis=(0, 0, 1), constraint_axis=(False, False, True), constraint_orientation='GLOBAL', mirror=False, proportional='DISABLED', proportional_edit_falloff='SMOOTH', proportional_size=1, release_confirm=True, use_accurate=False)
 bpy.ops.transform.rotate(value=0, axis=(1, 0, 0), constraint_axis=(True, False, False), constraint_orientation='GLOBAL', mirror=False, proportional='DISABLED', proportional_edit_falloff='SMOOTH', proportional_size=1, release_confirm=True, use_accurate=False)
 bpy.ops.transform.rotate(value=-0.3986, axis=(0, 1, 0), constraint_axis=(False, True, False), constraint_orientation='GLOBAL', mirror=False, proportional='DISABLED', proportional_edit_falloff='SMOOTH', proportional_size=1, release_confirm=True, use_accurate=False)
 bpy.ops.transform.rotate(value=0.443616, axis=(1, 0, 0), constraint_axis=(True, False, False), constraint_orientation='GLOBAL', mirror=False, proportional='DISABLED', proportional_edit_falloff='SMOOTH', proportional_size=1, release_confirm=True, use_accurate=False)
 bpy.ops.mesh.primitive_cube_add(radius=1, view_align=False, enter_editmode=False, location=(4.26007, 1.13685, 6.01476), layers=(True, False, False))
 bpy.ops.transform.rotate(value=-0.739609, axis=(1, 0, 0), constraint_axis=(True, False, False), constraint_orientation='GLOBAL', mirror=False, proportional='DISABLED', proportional_edit_falloff='SMOOTH', proportional_size=1, release_confirm=True, use_accurate=False)
```



# Creative Hacking & Making Connections

- Embrace programming in general not just programming in python, most important concepts are similar or the same in many programming languages, just small changes in syntax.

# FoxDot

- Created and maintained by Ryan Kirkbride, Leeds UK
- Tool for Live coding audio patterns
- \*\* Would be a useful tool to send control messages to something like processing, or another visual platform you'd like to use.
- What is live coding?
- Using core python to do strange things in FoxDot

# Foxdot Performance

The screenshot shows the SuperCollider IDE interface. On the left, the code editor displays the file `startup.scd` with various SuperCollider code snippets. In the center, there is a Python terminal window titled "C:\Python27\python.exe" showing the command `>>> from env import *`. To the right, a Tkinter window titled "74 tk" is open. The right-hand panel of the IDE displays the output of the SuperCollider server, which includes error messages about duplicate nodes, device options, and server initialization details.

```
startup.scd (C:/Python27/FoxDot Code) - SuperCollider IDE
File Session Edit View Language Help
File startup.scd
260 env = EnvGen.ar(Env([0,amp / 16,0], [sus, 0]), doneAction:2);
261 Out.ar(0,Pan2.ar( osc * env , pan));
262 } ).add;
263
264
265 SynthDef.new(\speed,
266 {
267 arg amp=0,sus=1,pan=0,freq=0,pos=0, verb=0.25, stretch=1, degree=0, dur=1, shift=0;
268 var buf, env, rate, out, start;
269
270 rate = 1.0 / sus;
271
272 buf = Buffer.read(s,thisProcess.nowExecutingPath.dirname +/- "samples/ryan/speech.wav";
273 env = EnvGen.ar(Env([0,1,0],[0,sus.abs,0]),doneAction:2);
274
275 // Work out where to start playing
276 start = BufSampleRate.kr(buf.bufnum) * stretch;
277
278 shift = shift * start;
279
280 pos = pos % (BufDur.kr(buf.bufnum) / stretch) * dur;
281
282 start = start * pos;
283
284 out = PlayBuf.ar(1,buf.bufnum, BufRateScale.kr(buf.bufnum), startPos: start + shift) * e
285
286 // Change pitch
287 out = PitchShift.ar(out, 0.1, 1 + (1/8 * (degree)), 0, 0.004);
288
289 out = FreeVerb.ar(out, verb);
290
291 Out.ar(0,Pan2.ar( out , pan));
292 } ).add;
293
294
295 SynthDef.new(\tone,
296 {
297 arg amp=0,sus=1,pan=0,freq=0,rate=1,depth=0.04,delay=0.01,onset=0,rateVar=0,depthVar=0,i
298 var osc, env, out;
299
300 osc = SinOsc.ar(Vibrato.kr(freq, rate:rate - 1, depth:depth, delay:delay, onset:onset, r
301 env = EnvGen.ar(Env(levels:[0,1,1,0]*(amp/2),times:[1/8,7/8,1/8]*sus),doneAction:2);
302
303 out = (osc * env);
304
305 Out.ar(0,Pan2.ar( out , pan));
306 } ).add;
307
308
309
310
311
312
313
314
315
```

C:\Python27\python.exe

```
>>> from env import *
```

74 tk

```
Post window
FAILURE IN SERVER /s_new duplicate node
ID
/quit sent
booting 57110
RESULT = 0

Device options:
- MME : Microsoft Sound Mapper -
Input (Device #0 with 2 ins 0 outs)
- MME : Mic in at rear panel (Pink)
(Re (device #1 with 2 ins 0 outs)
- MME : Stereo Mix (Realtek High
Defini (device #2 with 2 ins 0 outs)
- MME : Microphone (2- USB Audio
CODEC (device #3 with 2 ins 0 outs)
- MME : Microsoft Sound Mapper -
Output (device #4 with 0 ins 2 outs)
- MME : Speakers (2- USB Audio CODEC )
(device #5 with 0 ins 2 outs)
- MME : Headphones (Realtek High
Defini (device #6 with 0 ins 2 outs)
- MME : Realtek Digital Output
(Realtek (device #7 with 0 ins 2 outs)
- MME : Realtek Digital
Output(Optical) (device #8 with 0 ins 2 outs)

Booting with:
In: MME : Mic in at rear panel (Pink)
(Re
Out: MME : Speakers (2- USB Audio
CODEC )
Sample rate: 44100.000
Latency (in/out): 0.013 / 0.091 sec
SC_AudioDriver: sample rate =
44100.000000, driver's block size = 64
SuperCollider 3 server ready.
Receiving notification messages from
server localhost
Shared memory server interface
initialized
a SynthDef
```

Interpreter: Active Server: 0.29% 0.32% 0u 0s 2g 78d

EN 1642  
25/08/2015



## Local artists:

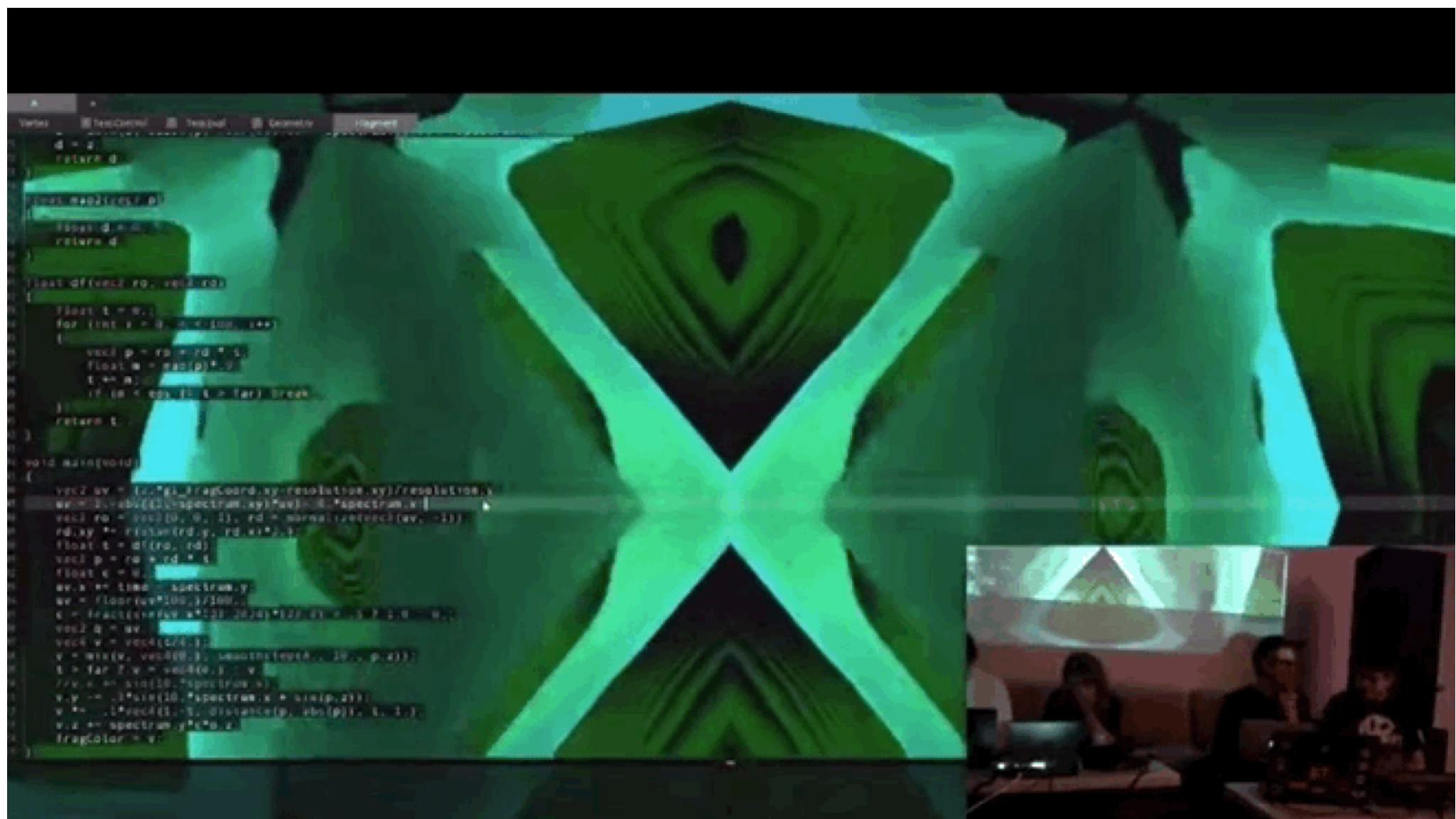
**Danielle Rager**  
**Kevin Bednar**  
**Lukas Herman**  
**Char Stiles**  
**Austin Marcus**

**Chris Palmer**  
**Samir Gangwani**  
**Slow Danger**  
**Adrienne Cassel**  
**Projectile Objects**

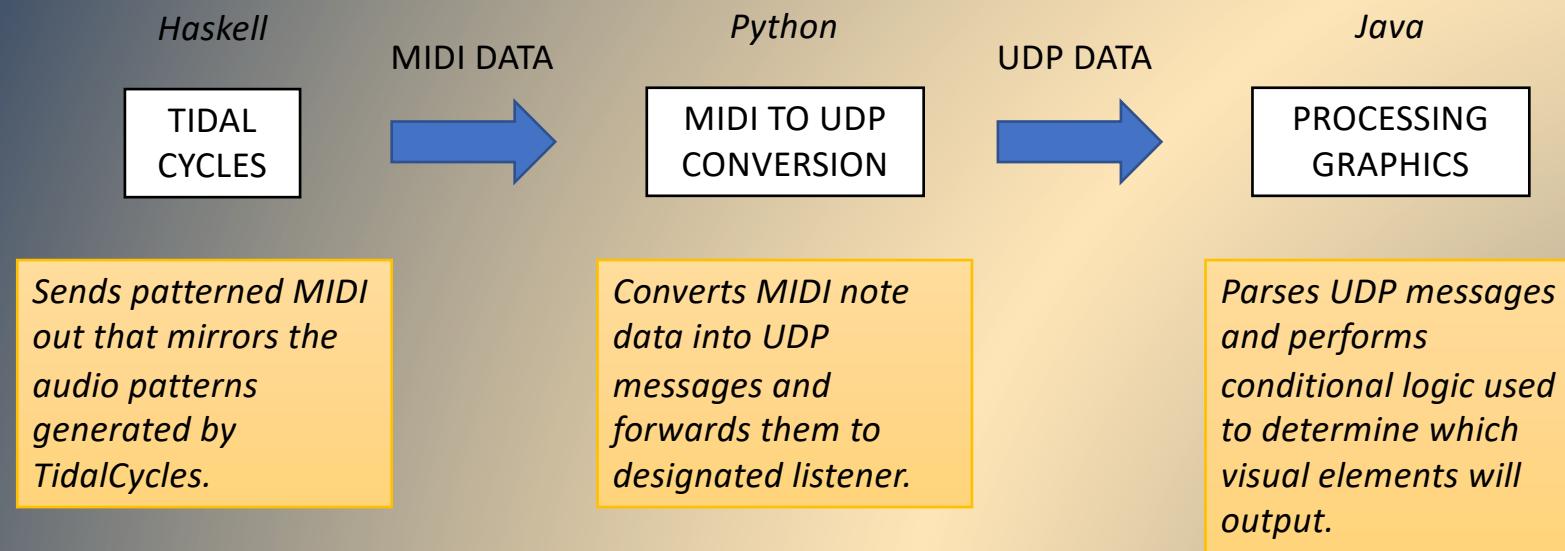
# SLOWDANGER

- Frequent collaboration with Projectile Objects & Char Stiles to bring a technology element to their live performance.
- Blog write-up's available here: <https://projectileobjects.com/>





# Personal Configuration



# Multiple Listeners

Haskell

TIDAL CYCLES

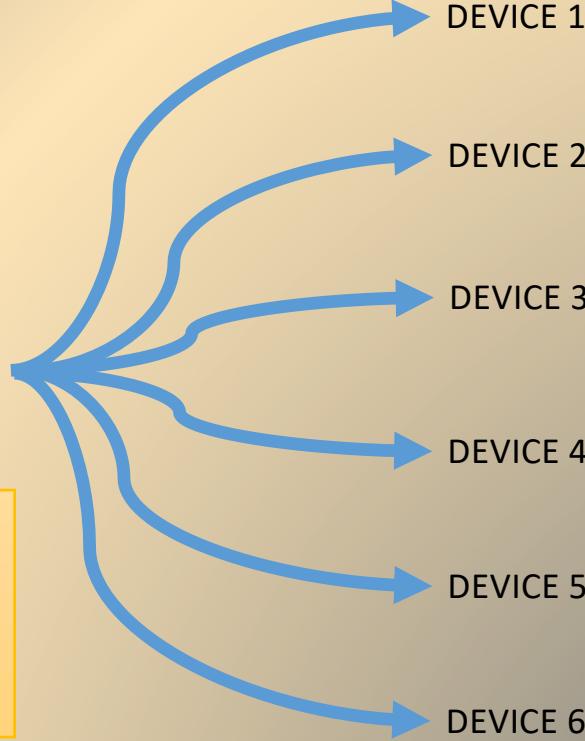


*Sends patterned MIDI out that mirrors the audio patterns generated by TidalCycles.*

Python

MIDI TO UDP CONVERSION

*Converts MIDI note data into UDP messages and forwards them to designated listener.*



PROCESSING GRAPHICS

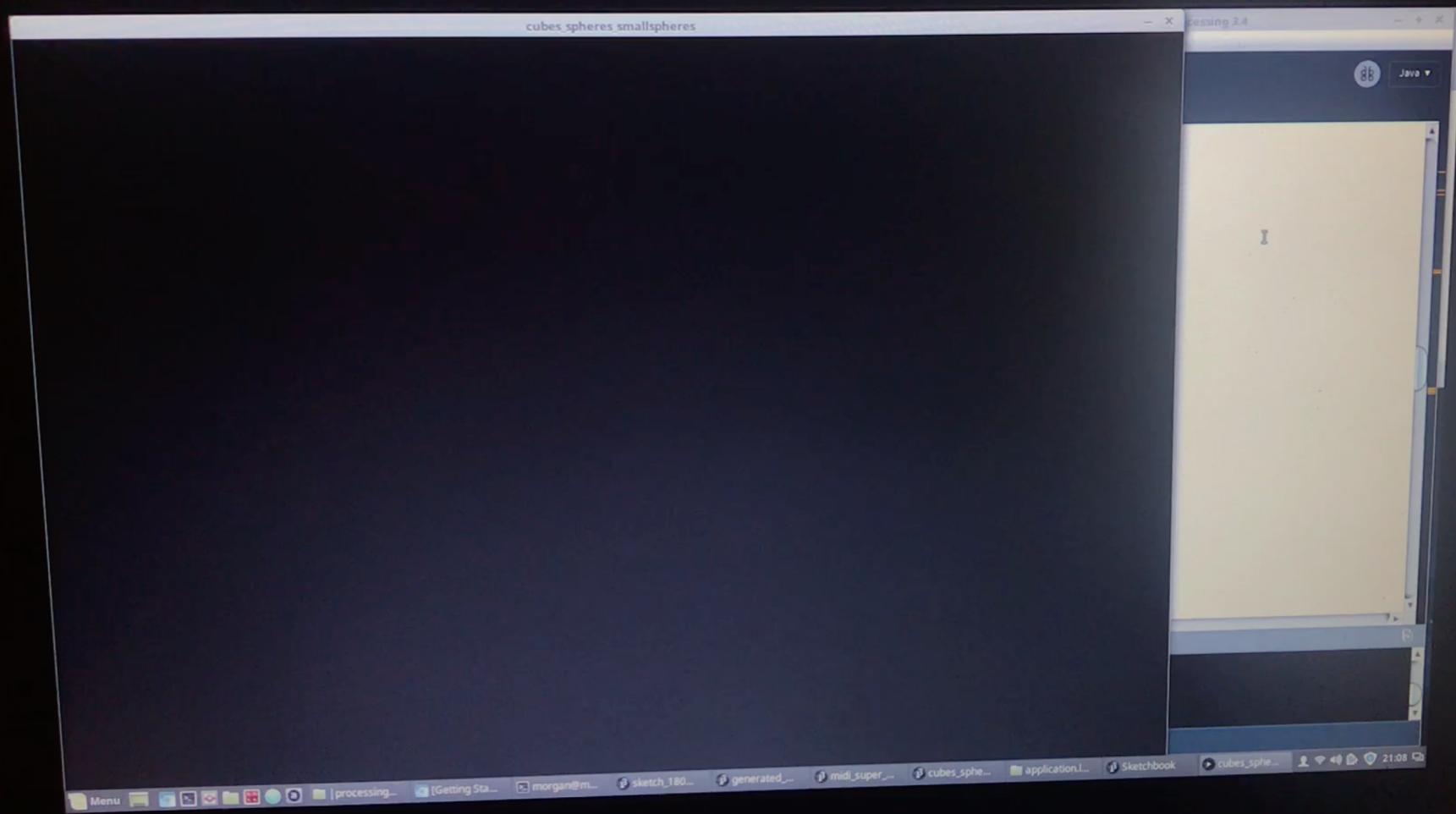
PROCESSING GRAPHICS

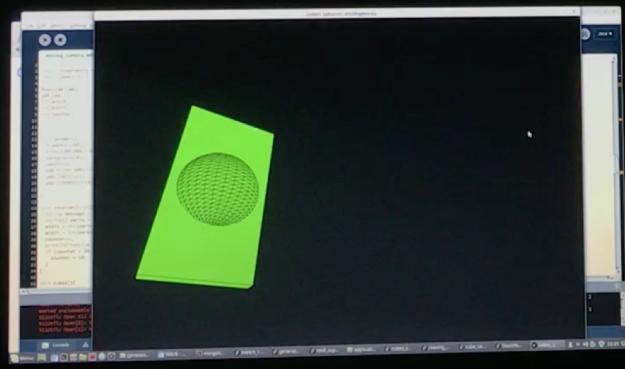
PROCESSING GRAPHICS

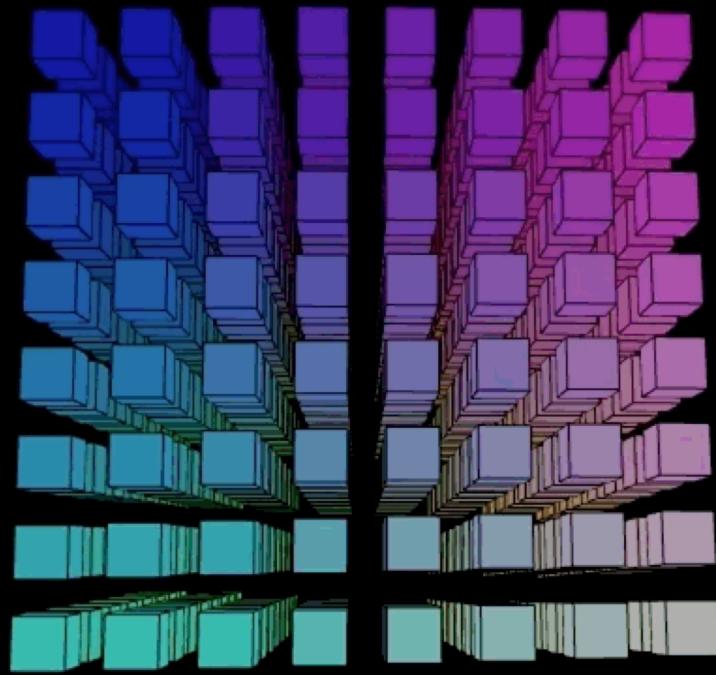
PROCESSING GRAPHICS

PROCESSING GRAPHICS

PROCESSING GRAPHICS







**MORE RESOURCES:**  
[https://github.com/vism2889/imaginative\\_python](https://github.com/vism2889/imaginative_python)

# Meet-up: Generative Art and Adversarial Networks

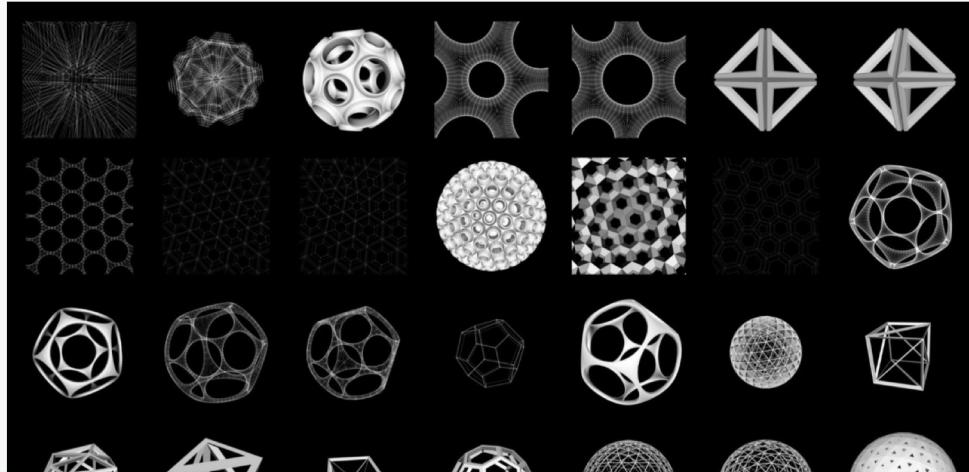
Thursday, November 21, 2019

## Generative Session



Hosted by [Matt Turnshek](#) and [Hazem](#)  
From [Generative Art and Adversarial Networks](#)  
Public group [?](#)

Share



Thursday, November 21, 2019  
6:30 PM to 9:30 PM  
[Add to calendar](#)



STACK  
5740 Baum Blvd. 3rd floor ·  
Pittsburgh, PA

How to find us  
After entering the building, head  
down the hall and then up the long  
staircase to the very top.

[https://github.com/vism2889/imaginative python](https://github.com/vism2889/imaginative_python)