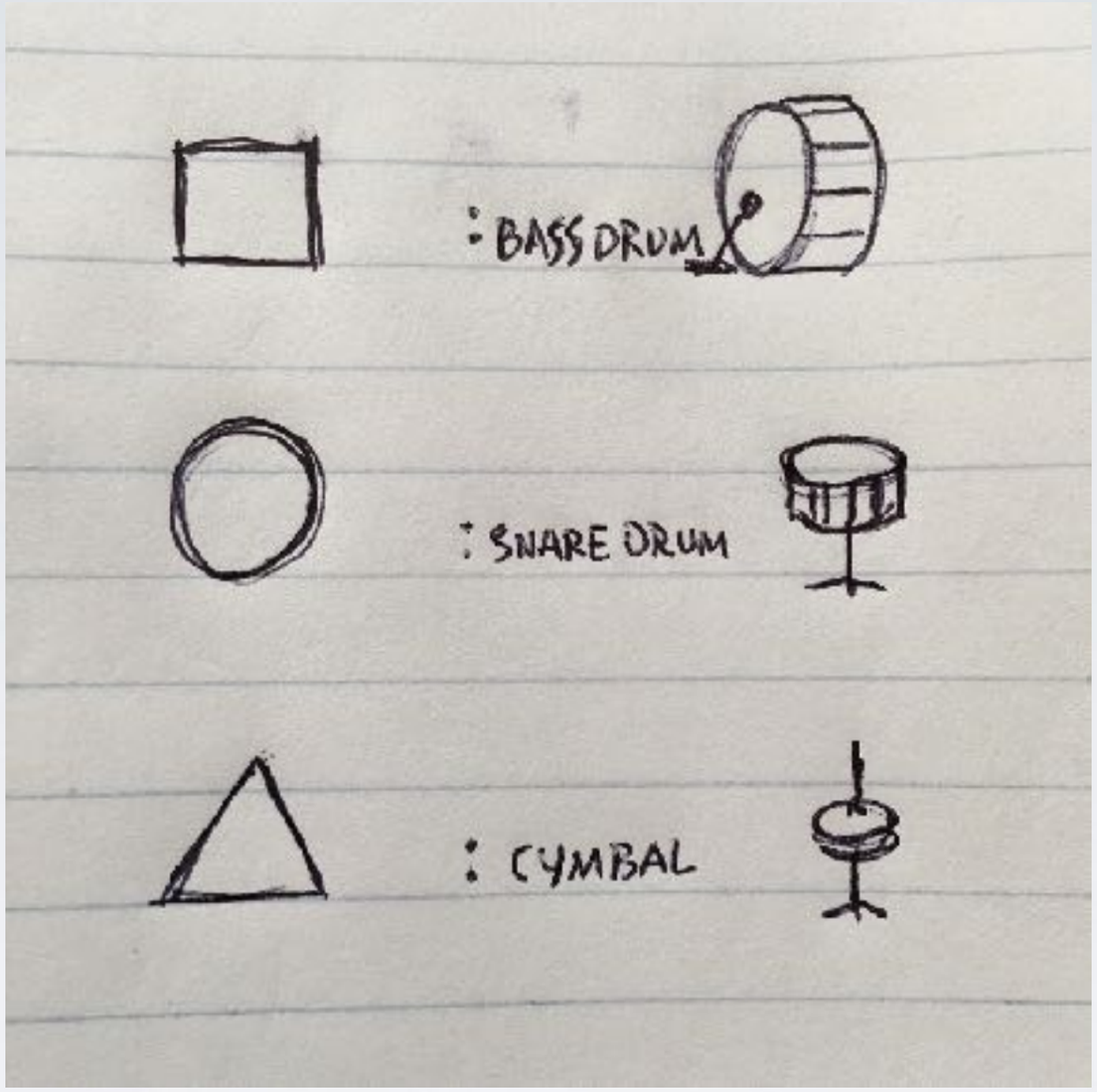
# CAMERA, SHAPES, SOUNDS

ART 185GL .XINGHAN LIU


# SHAPE AND MUSIC

Guitar Hero


GRETSCH

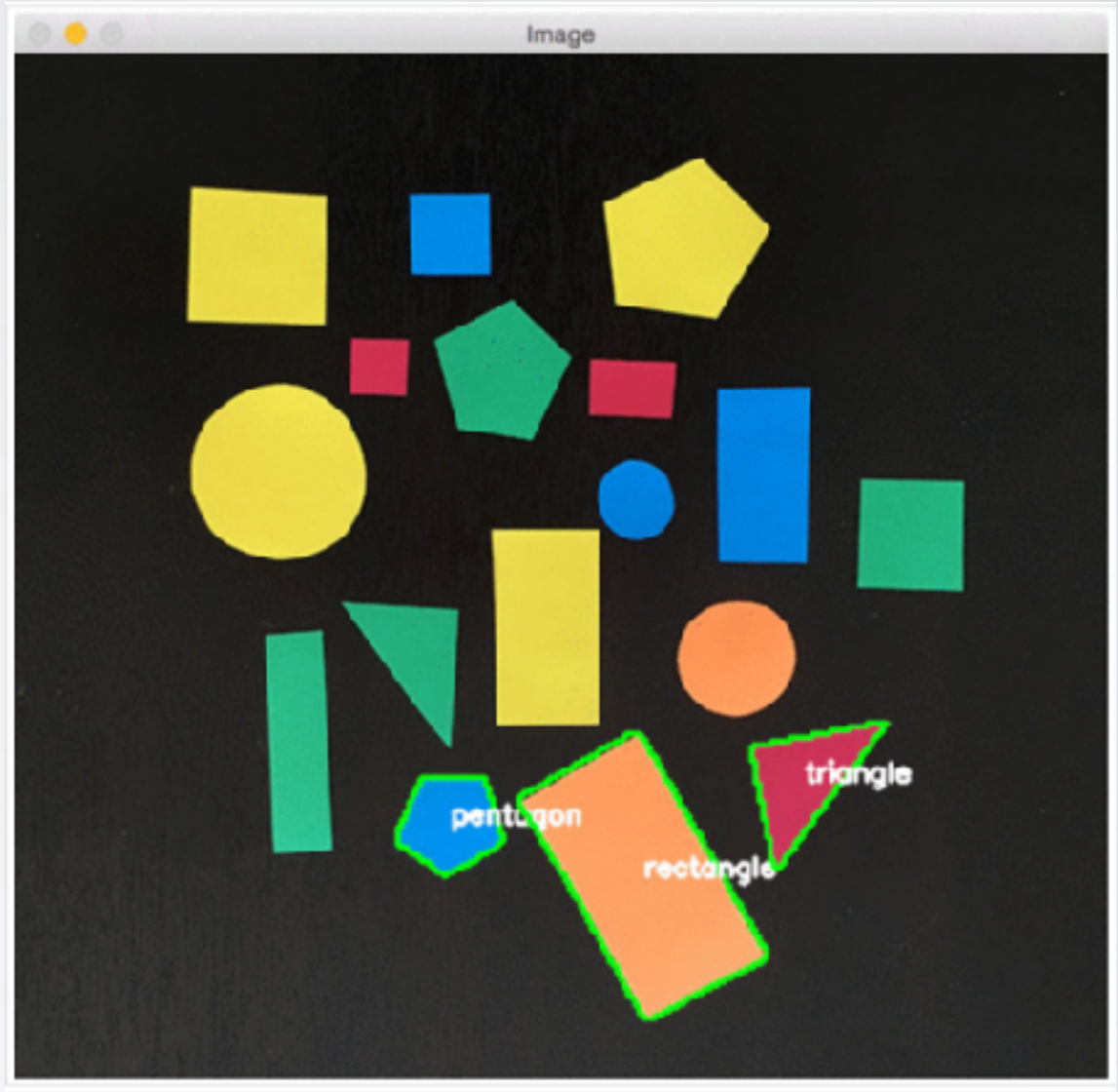



EACH SHAPE REPRESENTS A PART OF THE DRUMS

# CAMERA & SHAPE

# OPENCV SHAPE DETECTION

This is my research



shapedetector.py

```python
# import the necessary packages
import cv2

class ShapeDetector:
    def __init__(self):
        pass

    def detect(self, c):
        # initialize the shape name and approximate the contour
        shape = "unidentified"
        peri = cv2.arcLength(c, True)
        approx = cv2.approxPolyDP(c, 0.04 * peri, True)
```

OpenCV shape detection

```python
        # if the shape is a triangle, it will have 3 vertices
        if len(approx) == 3:
            shape = "triangle"

        # if the shape has 4 vertices, it is either a square or
        # a rectangle
        elif len(approx) == 4:
            # compute the bounding box of the contour and use the
            # bounding box to compute the aspect ratio
            (x, y, w, h) = cv2.boundingRect(approx)
            ar = w / float(h)

            # a square will have an aspect ratio that is approximately
            # equal to one, otherwise, the shape is a rectangle
            shape = "square" if ar >= 0.95 and ar <= 1.05 else "rectangle"

        # if the shape is a pentagon, it will have 5 vertices
        elif len(approx) == 5:
            shape = "pentagon"

        # otherwise, we assume the shape is a circle
        else:
            shape = "circle"

        # return the name of the shape
        return shape
```

## Shape detection with OpenCV

Now that our `ShapeDetector` class has been defined, let's create the `detect_shapes.py` driver script:

OpenCV shape detection — Python

```python
# import the necessary packages
from pyimagesearch.shapedetector import ShapeDetector
import argparse
import imutils
import cv2

# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required=True,
    help="path to the input image")
args = vars(ap.parse_args())
```

Next up, let's pre-process our image:

OpenCV shape detection — Python

```python
# load the image and resize it to a smaller factor so that
# the shapes can be approximated better
image = cv2.imread(args["image"])
resized = imutils.resize(image, width=300)
ratio = image.shape[0] / float(resized.shape[0])

# convert the resized image to grayscale, blur it slightly,
# and threshold it
gray = cv2.cvtColor(resized, cv2.COLOR_BGR2GRAY)
blurred = cv2.GaussianBlur(gray, (5, 5), 0)
thresh = cv2.threshold(blurred, 60, 255, cv2.THRESH_BINARY)[1]

# find contours in the thresholded image and initialize the
# shape detector
cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
    cv2.CHAIN_APPROX_SIMPLE)
cnts = cnts[0] if imutils.is_cv2() else cnts[1]
sd = ShapeDetector()
```

The last step is to identify each of the contours:

OpenCV shape detection — Python

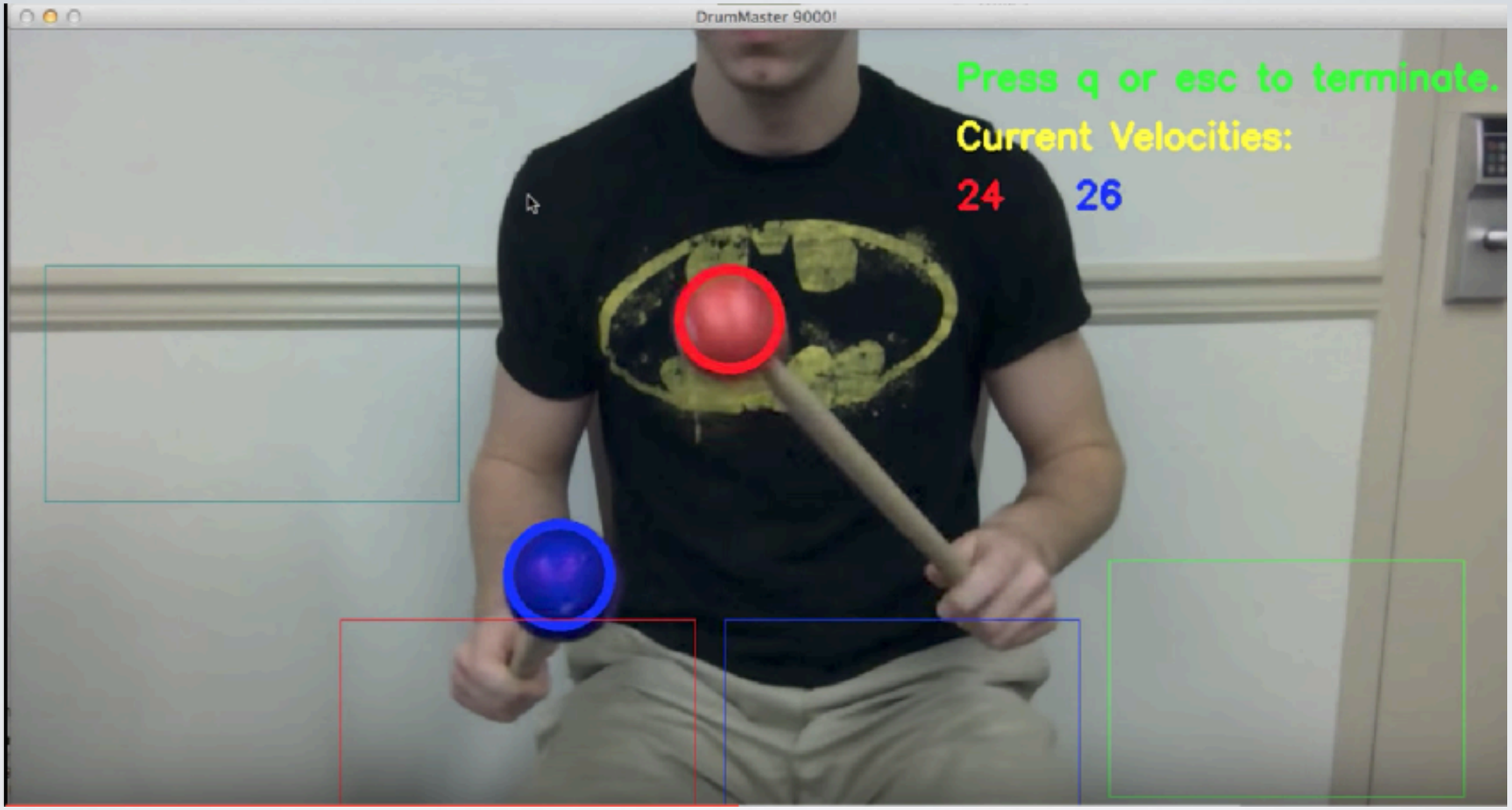
```python
# loop over the contours
for c in cnts:
    # compute the center of the contour, then detect the name of the
    # shape using only the contour
    M = cv2.moments(c)
    cX = int((M["m10"] / M["m00"]) * ratio)
    cY = int((M["m01"] / M["m00"]) * ratio)
    shape = sd.detect(c)

    # multiply the contour (x, y)-coordinates by the resize ratio,
    # then draw the contours and the name of the shape on the image
    c = c.astype("float")
    c *= ratio
    c = c.astype("int")
    cv2.drawContours(image, [c], -1, (0, 255, 0), 2)
    cv2.putText(image, shape, (cX, cY), cv2.FONT_HERSHEY_SIMPLEX,
        0.5, (255, 255, 255), 2)

    # show the output image
    cv2.imshow("Image", image)
    cv2.waitKey(0)
```

- Define the pyimagesearch module first. Inside this module we have shapedetector.py which will store the implementation of the ShapeDetector class.

- Finally we have the detect_shapes.py driver script that we'll use to load an image from disk, analyze it for shapes, and then perform shape detection and identification via the ShapeDetector class.

# SOUND

Pygame

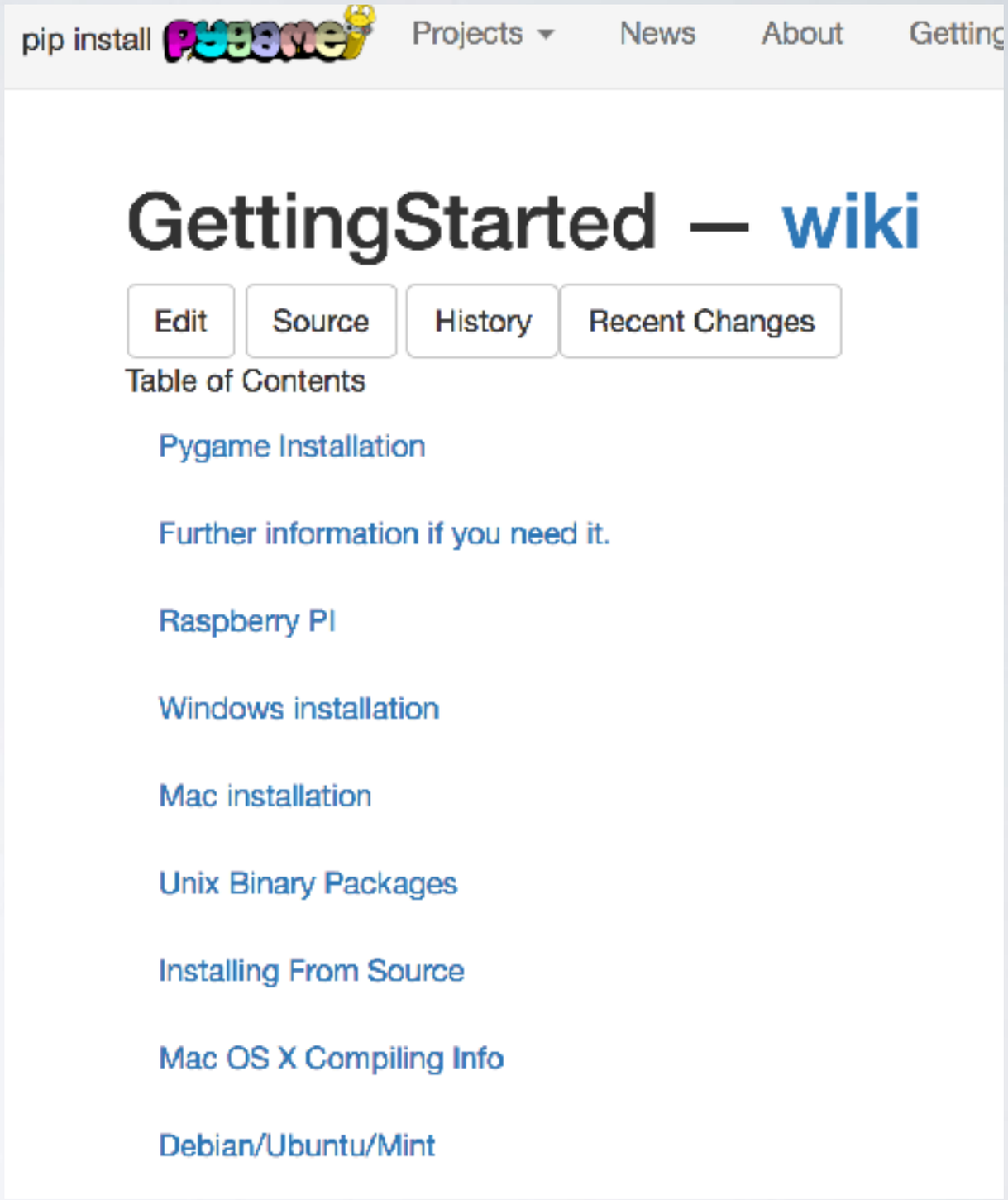


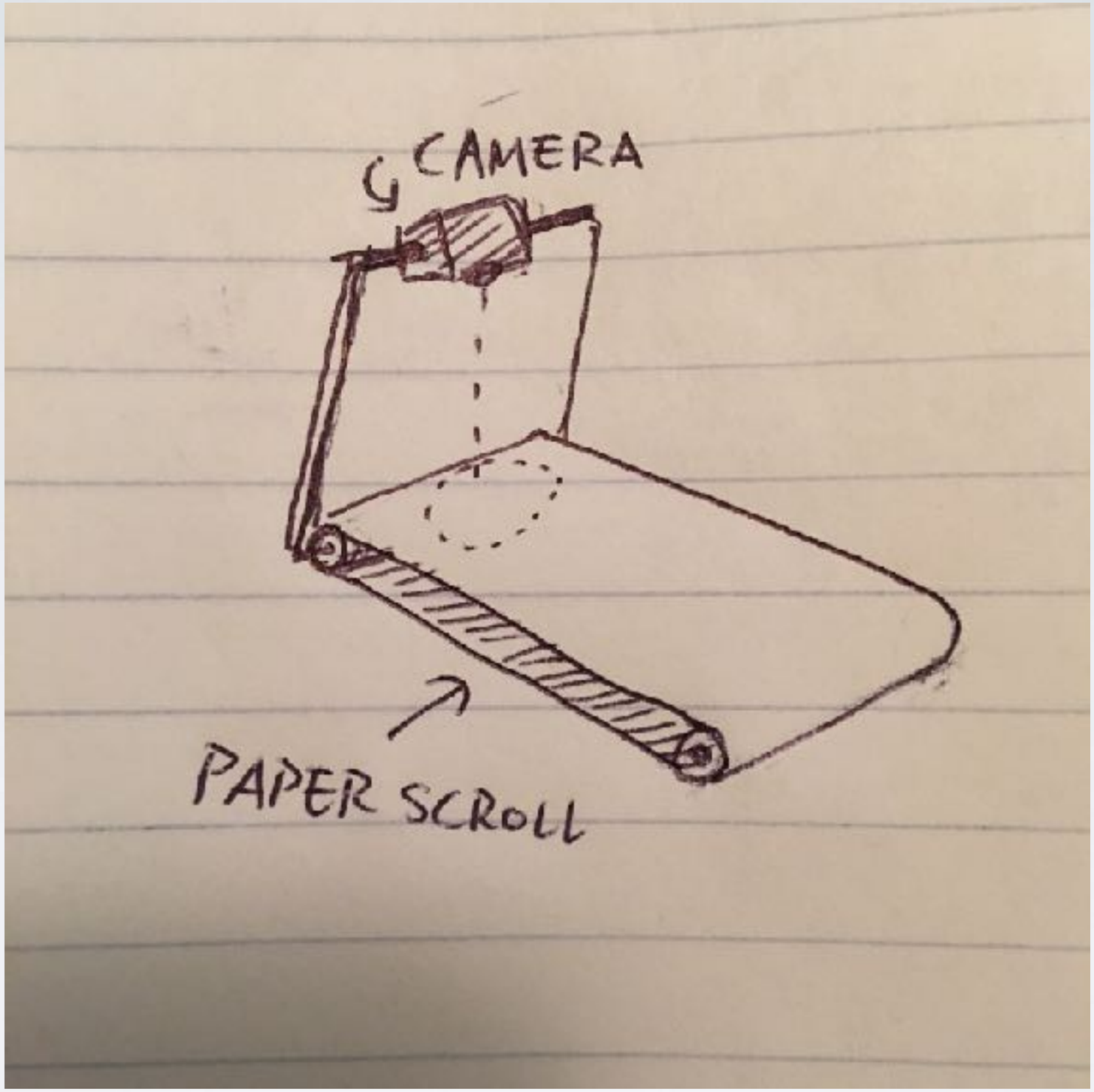

# AIR DRUMS

https://www.youtube.com/watch?v=MAnWwxTjL3k

# PYGAME

A third party library for python dedicated to easy game development.




CAMERA
PAPER SCROLL

# PROJECT USAGE

- An art installation that offer an interactive experience with audiences.
- A tool for musician to write music.

# THANK YOU!