COLOR DETECTOR (Works for Binary)

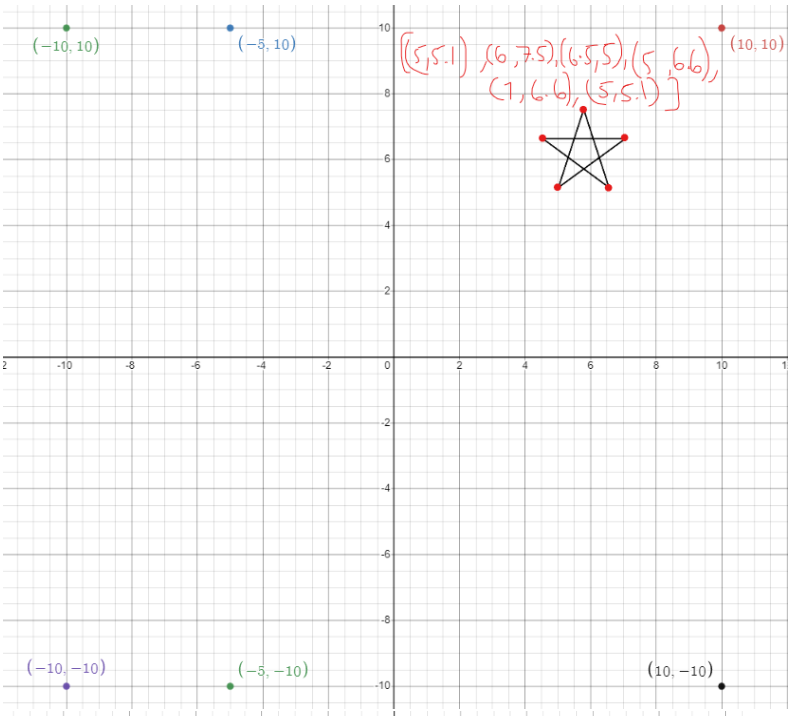
```
import PIL
from PIL import Image, ImageGrab, ImageOps
import turtle
import numpy
import os


filename = r"C:\Users\izzug\Desktop\Docs\UNI\assingments and labs\Sem - 6\CG\lab2\binary_image.png"
img = Image.open(filename) # image reader
img = img.convert('1')  # coverts rgb -> single digit representation
pixel_size = img.size # saves (width, height)

x = []
y = []

w = []
h = []

for width in range(pixel_size[0]):
    for height in range(pixel_size[1]):
        pixel_color = img.load() #pixel access
        w.append(pixel_color[width, height]) # gets color for each pixel
    h.append(w)
    w = []



# extracting black pixel position
for i in range(pixel_size[0]):
    for j in range(pixel_size[1]):
        if h[i][j] == 0:
            x.append(j)
            y.append(i)

# corners

# top right
A = (max(x), min(y))
# bottom left
B = (min(x), max(y))
# top left
```

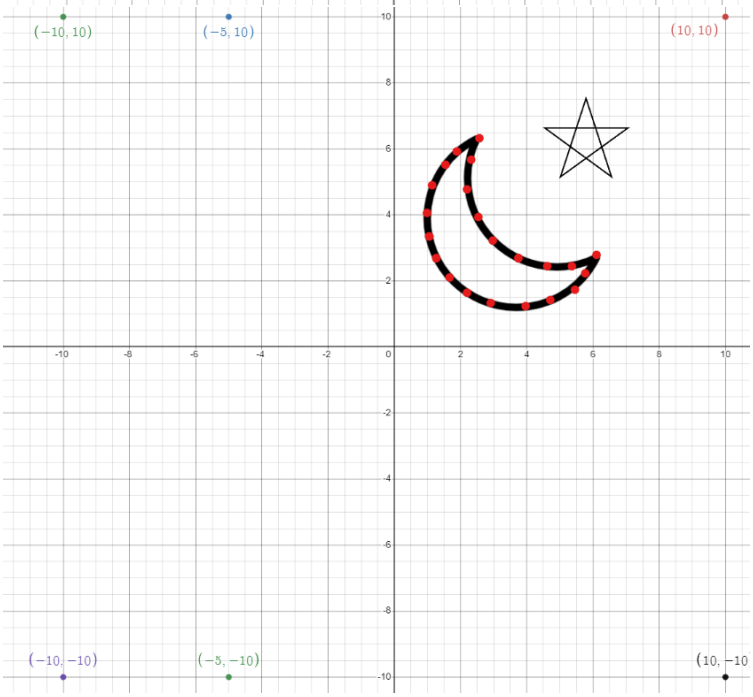
```
C = (min(x), min(y))
# bottom right
D = (max(x), max(y))

print(A, B, C, D)

# drawing lines (bit flip)
for i in range(C[1], B[1]):
    h[i][C[0]] = 0
for i in range(C[0], A[0]):
    h[C[1]][i] = 0
for i in range(A[1], D[1]):
    h[i][A[0]] = 0
for i in range(B[0], D[0]):
    h[B[1]][i] = 0

# #  bitmap for visualize
# for i in h:
#     print(i)

# converting 225 and 0 into True and False for np operations
for width in range(pixel_size[0]):
    for height in range(pixel_size[1]):
        if h[width][height] > 0:
            h[width][height] = True
        else:
           h[width][height] = False

np_arr = numpy.array(h)
img2 = Image.fromarray(np_arr)

# directly converts into numpy array
# npimg = numpy.array(img)
# print(npimg)
#print(np_arr.shape)

img2 = img2.rotate(270, PIL.Image.Resampling.NEAREST, expand = 1)
img2 = ImageOps.mirror(img2)
img2.save('binary_image_detected.png')
img2.show()
```

Miss Humera best!

POINT AND LINE (Pakistan Flag)

Mockups



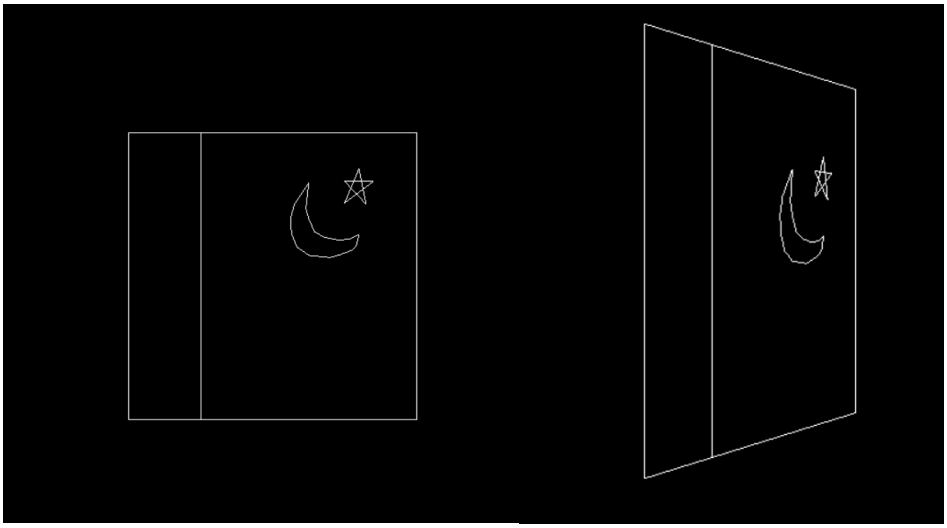


```
import * as THREE from "three"
// npm run dev for local server
// ctrl + C to close local server

const scene = new THREE.Scene();
const camera = new THREE.PerspectiveCamera(50, window.innerWidth /
window.innerHeight, 1, 500);
const renderer = new THREE.WebGLRenderer();
renderer.setSize(window.innerWidth, window.innerHeight);
document.body.appendChild(renderer.domElement);

camera.position.set(0, 0, 50);
camera.lookAt(0, 0, 0);


//material
const material = new THREE.LineBasicMaterial({ color: 0xffffff });

// flag layout
const l_points = [];
l_points.push(new THREE.Vector3(10, 10, 0));
l_points.push(new THREE.Vector3(-10, 10, 0));
l_points.push(new THREE.Vector3(-10, -10, 0));
l_points.push(new THREE.Vector3(10, -10, 0));
l_points.push(new THREE.Vector3(10, 10, 0));
l_points.push(new THREE.Vector3(-5, 10, 0));
l_points.push(new THREE.Vector3(-5, -10, 0));

const geometry = new THREE.BufferGeometry().setFromPoints(l_points);
const line = new THREE.Line(geometry, material);


// star
const s_points = [];
s_points.push(new THREE.Vector3(5, 5.1, 0));
s_points.push(new THREE.Vector3(6, 7.5, 0));
s_points.push(new THREE.Vector3(6.5, 5, 0));
s_points.push(new THREE.Vector3(5, 6.6, 0));
s_points.push(new THREE.Vector3(7, 6.6, 0));
s_points.push(new THREE.Vector3(5, 5.1, 0));

const geometry2 = new THREE.BufferGeometry().setFromPoints(s_points);
const star = new THREE.Line(geometry2, material);
```

```
//crescent
const c_points = [];
c_points.push(new THREE.Vector3(2.5, 6.5, 0));
c_points.push(new THREE.Vector3(2.4, 5.6, 0));
c_points.push(new THREE.Vector3(2.3, 4.7, 0));
c_points.push(new THREE.Vector3(2.5, 4, 0));

c_points.push(new THREE.Vector3(2.9, 3.1, 0));
c_points.push(new THREE.Vector3(3.6, 2.7, 0));
c_points.push(new THREE.Vector3(4.6, 2.5, 0));
c_points.push(new THREE.Vector3(5.4, 2.6, 0));
c_points.push(new THREE.Vector3(6, 2.9, 0));

c_points.push(new THREE.Vector3(5.8, 2.1, 0));
c_points.push(new THREE.Vector3(5.5, 1.8, 0));
c_points.push(new THREE.Vector3(4.7, 1.5, 0));
c_points.push(new THREE.Vector3(4, 1.3, 0));
c_points.push(new THREE.Vector3(2.5, 1.45, 0));
c_points.push(new THREE.Vector3(2.2, 1.68, 0));
c_points.push(new THREE.Vector3(1.7, 2, 0));

c_points.push(new THREE.Vector3(1.3, 3, 0));
c_points.push(new THREE.Vector3(1.24, 4, 0));
c_points.push(new THREE.Vector3(1.5, 5, 0));
c_points.push(new THREE.Vector3(2.5, 6.5, 0));



const geometry3 = new THREE.BufferGeometry().setFromPoints(c_points);
const crescent = new THREE.Line(geometry3, material);

scene.add(line);
scene.add(star);
scene.add(crescent);
renderer.render(scene, camera);


function animate() {

    // line.rotation.y += 0.01;
    // star.rotation.y += 0.01;
    // crescent.rotation.y += 0.01;
```

```
    // star.position.setY(-2);
    // crescent.position.setY(-2);

    renderer.render(scene, camera);
    window.requestAnimationFrame(animate);
};

animate();
```

changed position, rotation and scale



POINT AND LINE (rain scene)

Inspired by a star animation, changed the object-type and a few tweaks to make it feel like real rain instead of stars in space. Has a bottom to top POV. (3D)

Better illustrated through a video XD

SOURCE: https://codepen.io/GraemeFulton/pen/BNyQMM