**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)*

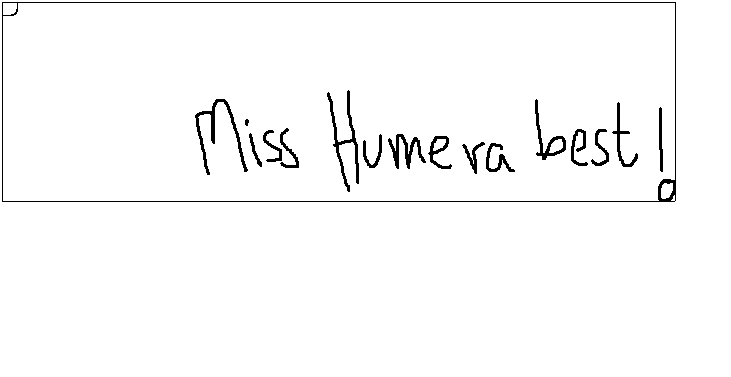
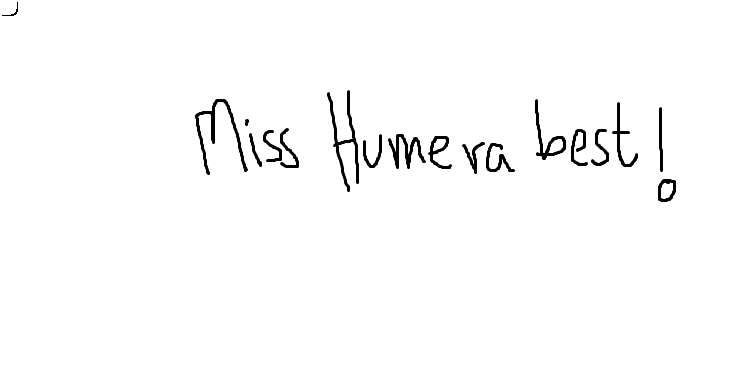
*#print(np\_arr.shape)*

img2 = img2.rotate(270, PIL.Image.Resampling.NEAREST, expand = 1)

img2 = ImageOps.mirror(img2)

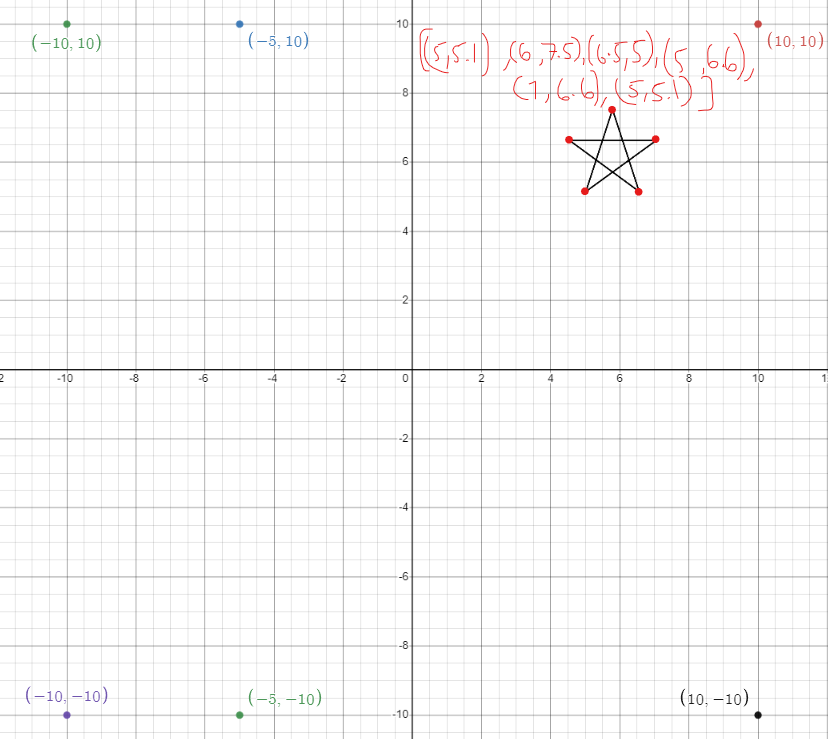
img2.save('binary\_image\_detected.png')

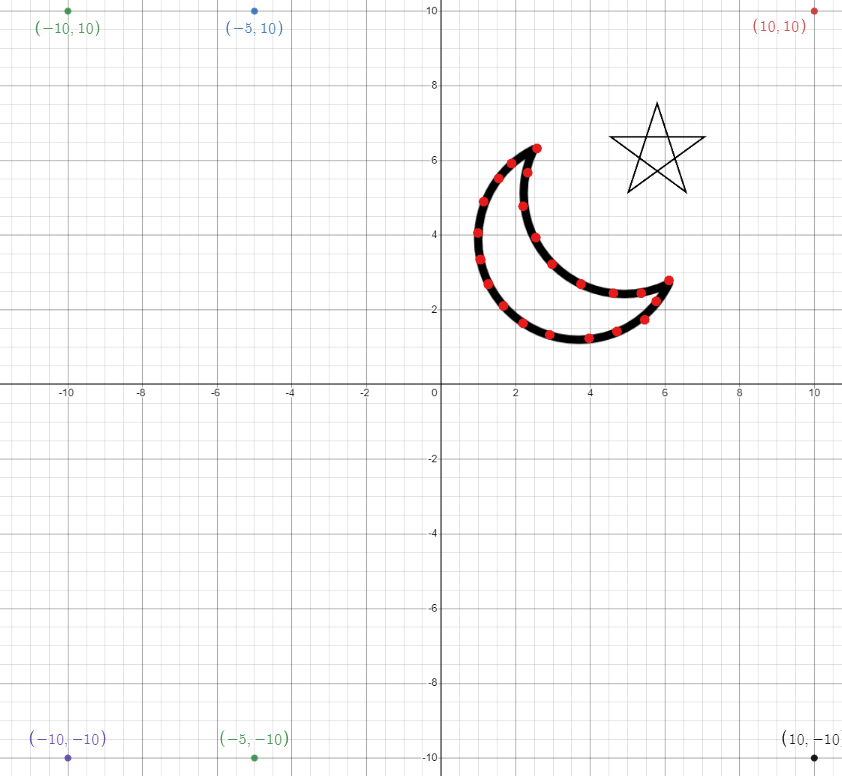
img2.show()



**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);*

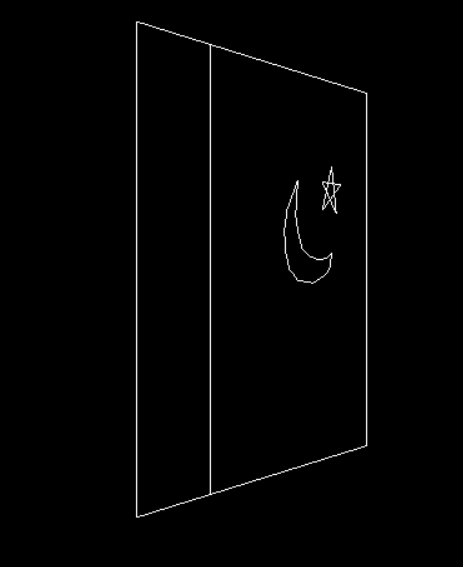
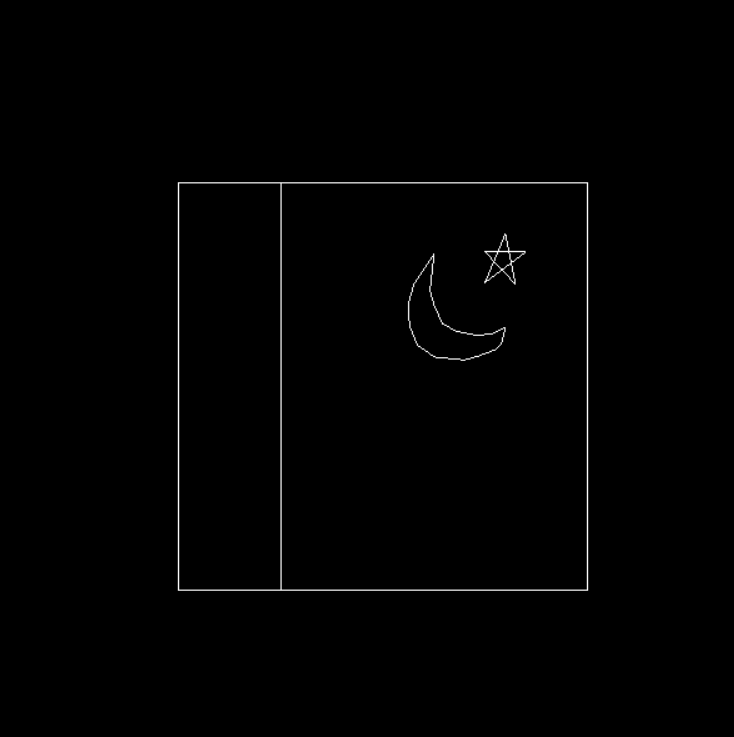
    renderer.render(scene, camera);

    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

**CIRCLE USING PARAMETRIC EQUATION (make it on a 2d plane in a 3d environment)**

import \* as THREE from "three"

import fs from "fs"

*// 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({

    canvas:document.querySelector(".three")

});     *// links to the canvas in html*

renderer.setSize(window.innerWidth, window.innerHeight);

*//document.body.appendChild(renderer.domElement); //creates new canvas*

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

camera.lookAt(0, 0, 0);

*//material*

const material = new THREE.LineBasicMaterial({ color: 0xFFD700 });

*// circle*

var c\_points = [];

var r = 20;  *//radius of the circle*

const tmax = 2 \* Math.PI;

const tmin = 0;

const total\_points = 40;

console.log("|   "+"i"+"   |      "+"t"+"     |       "+"x"+"     |      "+"y"+"      |")

for(let i = 0; i < total\_points + 1; i++){

    let t = ((tmax - tmin)/total\_points) \* i

    let x = r \* Math.cos(t)

    let y = r \* Math.sin(t)

    c\_points.push(new THREE.Vector3(x, y, 0))

    console.log("|   "+i+"   |    "+t.toFixed(2)+"   |    "+x.toFixed(2)+"   |    "+y.toFixed(2)+"   |")

}

const geometry = new THREE.BufferGeometry().setFromPoints(c\_points);

const circle = new THREE.Line(geometry, material);

scene.add(circle);

*// plane*

const geometry2 = new THREE.PlaneBufferGeometry(40, 40, 2, 2);

const material2 = new THREE.MeshBasicMaterial({

    color: 0x808080,

    side: THREE.DoubleSide,

    wireframe:true

} );

const plane = new THREE.Mesh(geometry2, material2);

scene.add(plane);

function animate() {

    let x = plane.rotation.x += 0.01;

    let y = plane.rotation.y = 0;

    let z = plane.rotation.z = 0;

    plane.scale.set(1, 1, 0)

    circle.scale.set( 0.5, 0.5, 0)

    if (x != circle.rotation.x){

        circle.rotation.x = x;

    }

    if (y != circle.rotation.y){

        circle.rotation.y = y;

    }

    if (z != circle.rotation.z){

        circle.rotation.z = z;

    }

    if (circle.scale.x > plane.scale.x){

        circle.scale.x = plane.scale.x;

    }

    if (circle.scale.y > plane.scale.y){

        circle.scale.y = plane.scale.y;

    }

*//circle.position.set(80, 0 ,0);*

*//circle.scale.set(15, 15, 0);*

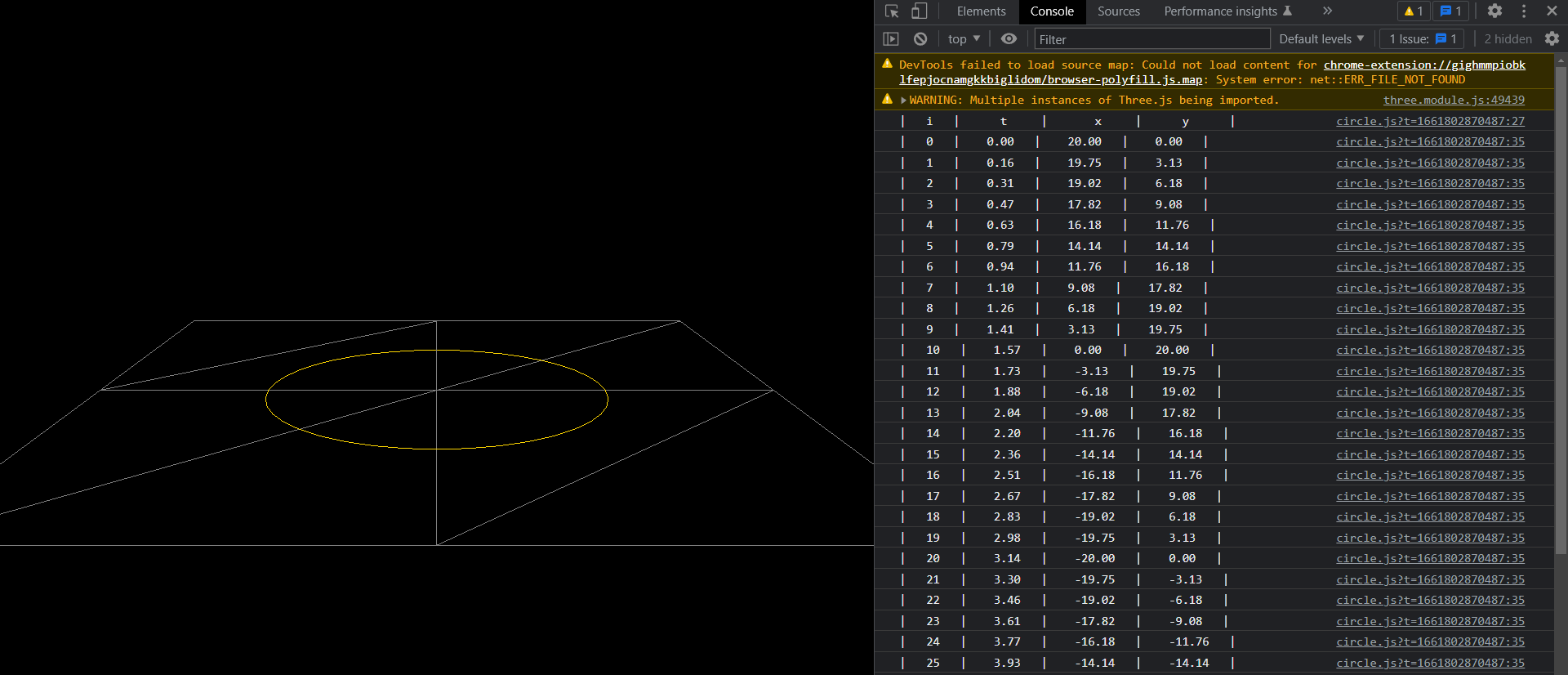
*//circle.rotation.y += 0.01;*

    renderer.render(scene, camera);

    window.requestAnimationFrame(animate);

};

animate();



**CREATE A SKYBOX USING A CUBE**

import \* as THREE from "three"

*// npm run dev for local server*

*// ctrl + C to close local server*

let scene, camera, renderer;

function init() {

  scene = new THREE.Scene();

  camera = new THREE.PerspectiveCamera(55,window.innerWidth/window.innerHeight,45,30000);

  camera.position.set(200,150,0);

  renderer = new THREE.WebGLRenderer({

    antialias:true,

    canvas:document.querySelector(".three")

  });

  renderer.setSize(window.innerWidth,window.innerHeight);

  document.body.appendChild(renderer.domElement);

  let materialArray = [];

  let texture\_ft = new THREE.TextureLoader().load("material/Daylight Box\_Front.bmp");

  let texture\_bk = new THREE.TextureLoader().load("material/Daylight Box\_Back.bmp");

  let texture\_up = new THREE.TextureLoader().load("material/Daylight Box\_Top.bmp");

  let texture\_dn = new THREE.TextureLoader().load("material/Daylight Box\_Bottom.bmp");

  let texture\_rt = new THREE.TextureLoader().load("material/Daylight Box\_Right.bmp");

  let texture\_lf = new THREE.TextureLoader().load("material/Daylight Box\_Left.bmp");

  materialArray.push(new THREE.MeshBasicMaterial( { map: texture\_ft }));

  materialArray.push(new THREE.MeshBasicMaterial( { map: texture\_bk }));

  materialArray.push(new THREE.MeshBasicMaterial( { map: texture\_up }));

  materialArray.push(new THREE.MeshBasicMaterial( { map: texture\_dn }));

  materialArray.push(new THREE.MeshBasicMaterial( { map: texture\_rt }));

  materialArray.push(new THREE.MeshBasicMaterial( { map: texture\_lf }));

  for (let i = 0; i < 6; i++)

     materialArray[i].side = THREE.BackSide;

  var skyboxGeo = new THREE.BoxGeometry(1000, 1000, 1000);

  var skybox = new THREE.Mesh( skyboxGeo, materialArray );

  scene.add( skybox );

}

init();

function animate() {

    camera.rotation.y += 0.001;

  renderer.render(scene,camera);

  requestAnimationFrame(animate);

}

animate();

