ArkaEnergy Assignment

Process for project creation

```
ArkaEnergy Assignment
├── index.html
├── package.json
├── index.css
├── node_modules/
└── source/
    ├── main.js
```

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Arka Assessment</title>
  <link rel="stylesheet" href="index.css">
</head>
<body>
  <div class="topbutton">
    <button id="create">Create</button>
  </div>
  <div class="copybutton">
    <button id="copy">Copy</button>
  </div>
```

```html
  <div class="bottombutton">
    <button id="reset">Reset</button>
  </div>

  <script type="module" src="src/main.js"></script>
</body>
</html>
```

Index.css

```css
.topbutton {
  position: absolute;
  top: 20px;
  left: 50%;
  transform: translateX(-50%);
}
.copybutton {
  position: absolute;
  top: 80px;
  left: 50%;
  transform: translateX(-50%);
}

.bottombutton {
  position: absolute;
  bottom: 20px;
  left: 50%;
  transform: translateX(-50%);
}

.topbutton button {
  font-size: 17px;
  color: rgb(255, 255, 255);
  border-radius: 50%;
  padding: 10px 20px;
  border-color: #42a11ce4;
  box-shadow: 0 8px 16px rgba(224, 240, 101, 0.2);
  background-color: #42a11ce4;
}
.copybutton button {
  font-size: 17px;
  color: rgb(255, 255, 255);
  border-radius: 50%;
  padding: 10px 20px;
  border-color: #42a11ce4;
  box-shadow: 0 8px 16px rgba(224, 240, 101, 0.2);
  background-color: #42a11ce4;
}

.topbutton button:hover {
  padding: 11px 21px;
  border-color: #11391a;
  background-color: #11391a;
  box-shadow: 0px 20px 16px rgba(157, 214, 149, 0.362);
}
.copybutton button:hover {
  padding: 11px 21px;
  border-color: #11391a;
  background-color: #11391a;
  box-shadow: 0px 20px 16px rgba(157, 214, 149, 0.362);
}

.bottombutton button {
  font-size: 17px;
  color: rgb(255, 255, 255);
  border-radius: 50%;
  padding: 10px 20px;
  border-color:  #901010;
  box-shadow: 0px 8px 16px rgba(0, 0, 0, 0.2);
  background-color: #901010;
}

.bottombutton button:hover {
  padding: 11px 21px;
  border-color: #3d0909 ;
  background-color: #3d0909;
  box-shadow: 0 20px 16px rgba(217, 154, 153, 0.362);
}
```

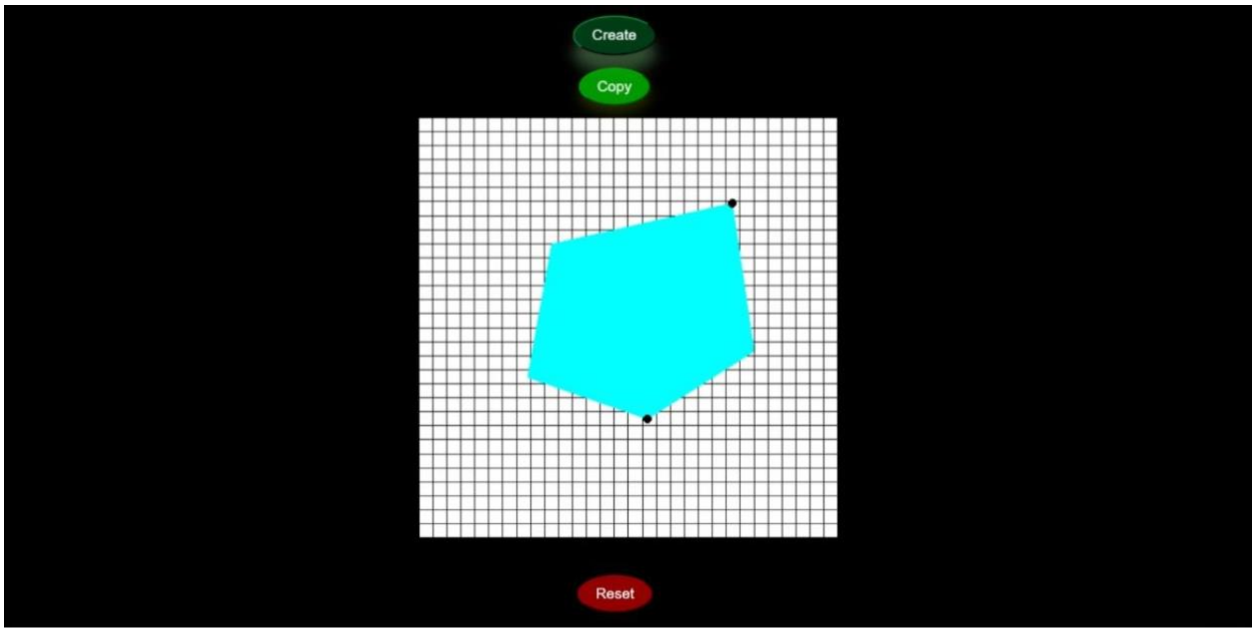
Main.js

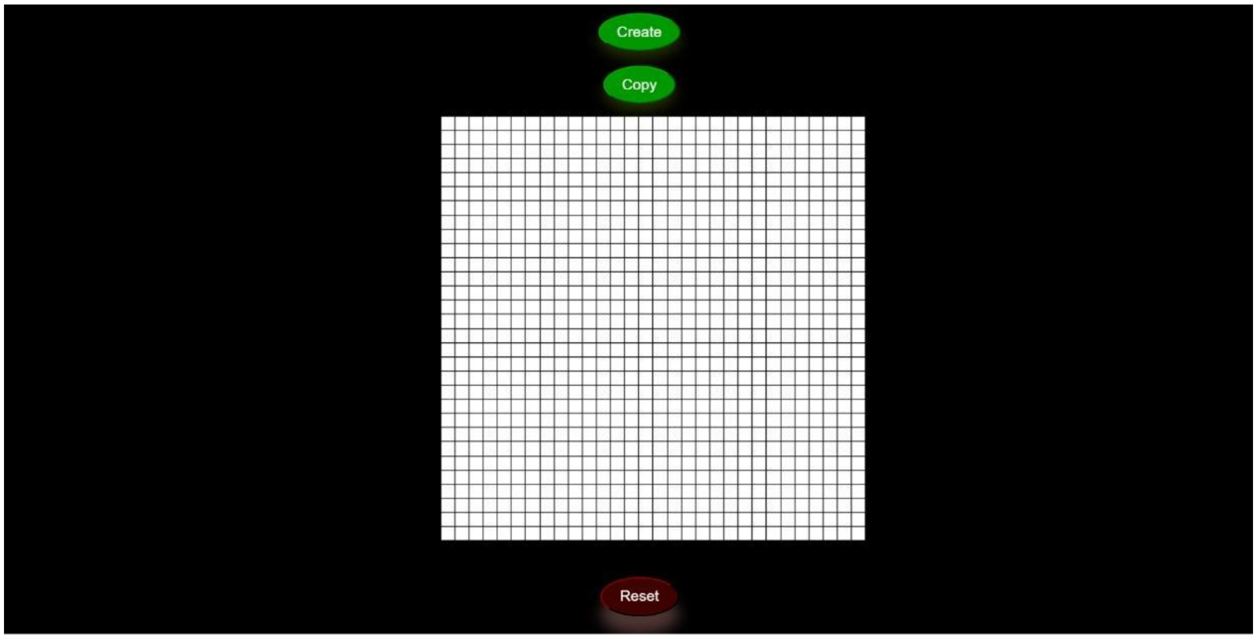
```
import * as THREE from
'https://cdn.jsdelivr.net/npm/three@0.166.1/build/three.module.js';

//crating scene to place object,cameras

const scene = new THREE.Scene();



//creating camera

const camera = new THREE.PerspectiveCamera(75, window.innerWidth /
window.innerHeight, 0.1, 1000);

camera.position.z = 10;


//renderer to display the scene onto a html canvas

const renderer = new THREE.WebGLRenderer();

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

document.body.appendChild(renderer.domElement);


//creating a white plane om the screen

const pGeometry = new THREE.PlaneGeometry(10, 10);

const pMaterial = new THREE.MeshBasicMaterial({ color: 0xffffff, side: THREE.DoubleSide
});

const plane = new THREE.Mesh(pGeometry, pMaterial);

scene.add(plane);



//creating grids on the plane
```

```
const grids = new THREE.GridHelper(10, 30, 0x000000, 0x000000);

//to keep the grids 90 degrees rotated
grids.rotation.x = Math.PI / 2;
scene.add(grpolygonl

//to store the vertices that we draw using mouse click
let vertices = [];
//to store the polygon
let polygons = [];
//to store the edges of the polygon
let edges = [];

document.addEventListener('click', MouseClick, false);

function MouseClick(event) {
  const mouse = new THREE.Vector2(
    (event.clientX / window.innerWidth) * 2 - 1,
    -(event.clientY / window.innerHeight) * 2 + 1
  );

  const raycaster = new THREE.Raycaster();
  raycaster.setFromCamera(mouse, camera);
  const intersects = raycaster.intersectObject(plane);

  if (intersects.length > 0) {
        const point = intersects[0].point;
        vertices.push(new THREE.Vector3(point.x, point.y, 0));
        creatingvertexmarks(point.x, point.y);
        //to keep the create button visible
        document.getElementById('create').style.display = 'block';
    }
}
//creates the vertexes on the plane
function creatingvertexmarks(x, y) {
    const geometry = new THREE.SphereGeometry(0.1, 32, 32);
    const material = new THREE.MeshBasicMaterial({ color: 0x000000 });
    const sphere = new THREE.Mesh(geometry, material);
    sphere.position.set(x, y, 0);
    scene.add(sphere);
}
const create = document.getElementById('create');
create.onclick = createPolygon;

function createPolygon() {
    //creating geometrical space for polygon the we draw
    const shape = new THREE.Shape();
    vertices.forEach((vertex, index) => {
        if (index === 0) {
            shape.moveTo(vertex.x, vertex.y);
        } else {
            shape.lineTo(vertex.x, vertex.y);
```

```
      }
   });
   shape.lineTo(vertices[0].x, vertices[0].y);
   const geometry = new THREE.ShapeGeometry(shape);
   const material = new THREE.MeshBasicMaterial({ color: 0x00ffff, side:
THREE.DoubleSide });
   const polygon = new THREE.Mesh(geometry, material);
   scene.add(polygon);

   const edgeGeometry = new THREE.EdgesGeometry(geometry);
   const edgeMaterial = new THREE.LineBasicMaterial({ color: 0x00ffff });
   const edge = new THREE.LineSegments(edgeGeometry, edgeMaterial);
   scene.add(edge);

  //storing edges of polygon
   polygons.push({ polygon, edge });
   vertices = [];
   clearingvertex();
}


//function to clear vertex
function clearingvertex() {
   scene.children.forEach(child => {
      if (child instanceof THREE.Mesh && child.geometry instanceof
THREE.SphereGeometry) {
         scene.remove(child);
```

```
        }
    });
}
const reset = document.getElementById('reset');
reset.onclick = resetplane;
function resetplane() {
    polygons.forEach(({ polygon, edge }) => {
        scene.remove(polygon);
        scene.remove(edge);
    });
    polygons = [];
    edges = [];
    vertices = [];
    clearingvertex();
    document.getElementById('create').style.display = 'block';
}

function animate() {
    requestAnimationFrame(animate);
    renderer.render(scene, camera);
}
animate();
```


Create
Copy
Reset


Create
Copy
Reset