```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>2D Polygon Creator</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <div id="canvas-container"></div>
  <button id="complete-btn">Complete</button>
  <button id="copy-btn">Copy</button>
  <button id="reset-btn">Reset</button>
  <!-- Include Three.js from CDN -->
  <script src="https://cdnjs.cloudflare.com/ajax/libs/three.js/r128/three.min.js"></script>
  <script src="script.js"></script>
</body>
</html>
```

```
const scene = new THREE.Scene();
const camera = new THREE.OrthographicCamera(window.innerWidth / -2, window.innerWidth / 2, window.innerHeight / 2, window.innerHeight / -2, 1, 1000);
const renderer = new THREE.WebGLRenderer({ antialias: true });

renderer.setSize(window.innerWidth, window.innerHeight);
document.getElementById('canvas-container').appendChild(renderer.domElement);

camera.position.z = 5;

// Create the ground plane
const planeGeometry = new THREE.PlaneGeometry(window.innerWidth, window.innerHeight);
const planeMaterial = new THREE.MeshBasicMaterial({ color: 0xffffff });
```

```
const plane = new THREE.Mesh(planeGeometry, planeMaterial);
scene.add(plane);

// Add grid lines
const gridHelper = new THREE.GridHelper(window.innerWidth, 50, 0x000000, 0x000000);
scene.add(gridHelper);

// Render the scene
function animate() {
  requestAnimationFrame(animate);
  renderer.render(scene, camera);
}
animate();

// Handle window resize
window.addEventListener('resize', () => {
  renderer.setSize(window.innerWidth, window.innerHeight);
  camera.left = window.innerWidth / -2;
  camera.right = window.innerWidth / 2;
  camera.top = window.innerHeight / 2;
  camera.bottom = window.innerHeight / -2;
  camera.updateProjectionMatrix();
});

// Polygon class and other functionalities can be added here...
class Polygon {
  constructor() {
    this.vertices = [];
    this.polygonMesh = null;
    this.lineMesh = null;
  }
```

```
addVertex(x, y) {
    this.vertices.push(new THREE.Vector3(x, y, 0));
    if (this.vertices.length > 1) {
        this.drawEdges();
    }
}

complete() {
    if (this.vertices.length > 2) {
        const shape = new THREE.Shape(this.vertices);
        const geometry = new THREE.ShapeGeometry(shape);
        const material = new THREE.MeshBasicMaterial({ color: 0xff0000 });
        this.polygonMesh = new THREE.Mesh(geometry, material);
        scene.add(this.polygonMesh);
    }
    this.clearLines();
}

drawEdges() {
    if (this.lineMesh) {
        scene.remove(this.lineMesh);
    }

    const points = this.vertices.map(v => new THREE.Vector3(v.x, v.y, v.z));
    const geometry = new THREE.BufferGeometry().setFromPoints(points);
    const material = new THREE.LineBasicMaterial({ color: 0x0000ff });
    this.lineMesh = new THREE.Line(geometry, material);
    scene.add(this.lineMesh);
}
```

```
    clearLines() {
        if (this.lineMesh) {
            scene.remove(this.lineMesh);
            this.lineMesh = null;
        }
    }
}

let currentPolygon = new Polygon();
// Handle mouse click to add vertices
document.addEventListener('mousedown', (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;
        currentPolygon.addVertex(point.x, point.y);
    }
});
document.getElementById('complete-btn').addEventListener('click', () => {
    currentPolygon.complete();
    currentPolygon = new Polygon(); // Start a new polygon
});
let copiedPolygon = null;
```

```
document.getElementById('copy-btn').addEventListener('click', () => {
    if (currentPolygon.vertices.length > 2 && currentPolygon.polygonMesh) {
        copiedPolygon = new Polygon();
        copiedPolygon.vertices = [...currentPolygon.vertices];

        const shape = new THREE.Shape(copiedPolygon.vertices);
        const geometry = new THREE.ShapeGeometry(shape);
        const material = new THREE.MeshBasicMaterial({ color: 0x00ff00 });
        copiedPolygon.polygonMesh = new THREE.Mesh(geometry, material);
        scene.add(copiedPolygon.polygonMesh);

        document.addEventListener('mousemove', moveCopiedPolygon);
        document.addEventListener('mousedown', placeCopiedPolygon);
    }
});

function moveCopiedPolygon(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 && copiedPolygon.polygonMesh) {
        const point = intersects[0].point;
        copiedPolygon.polygonMesh.position.set(point.x, point.y, 0);
    }
}
```

```
function placeCopiedPolygon(event) {
    document.removeEventListener('mousemove', moveCopiedPolygon);
    document.removeEventListener('mousedown', placeCopiedPolygon);
    copiedPolygon = null;
}
document.getElementById('reset-btn').addEventListener('click', () => {
    scene.children = scene.children.filter(child => child !== plane && child !== gridHelper);
    currentPolygon = new Polygon();
    copiedPolygon = null;
});
```