```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>3D Polygon Creator</title>
  <style>
    body {
      font-family: Arial, sans-serif;
    }
    #buttons {
      position: absolute;
      bottom: 0;
      left: 50%;
      transform: translateX(-50%);
      background-color: #fff;
      padding: 10px;
      border: 1px solid #ddd;
      box-shadow: 0px 0px 10px rgba(0,0,0,0.2);
    }
    #buttons button {
      margin-right: 10px;
    }
    #canvas {
      width: 100%;
      height: 100vh;
      display: block;
    }
  </style>
</head>
<body>
```

```
<div id="buttons">
  <button id="complete-button">Complete Polygon</button>
  <button id="copy-button">Copy Polygon</button>
  <button id="reset-button">Reset</button>
</div>
<canvas id="canvas"></canvas>
<script src="(link unavailable)"></script>
<script>
  // GroundPlane class
  class GroundPlane {
    constructor(scene) {
      this.plane = new THREE.Mesh(
        new THREE.PlaneGeometry(100, 100),
        new THREE.MeshBasicMaterial({ color: 0xffffff })
      );
      this.plane.receiveShadow = true;
      scene.add(this.plane);
    }
  }

  // Polygon class
  class Polygon {
    constructor(scene, color) {
      this.vertices = [];
      this.color = color;
      this.polygon = null;
      this.edgeLines = null;
    }

    addVertex(x, y) {
      this.vertices.push(new THREE.Vector3(x, y, 0));
```

```
  }

  completePolygon() {
    const shape = new THREE.Shape(this.vertices);
    this.polygon = new THREE.Mesh(
      new THREE.ShapeGeometry(shape),
      new THREE.MeshBasicMaterial({ color: this.color })
    );
    this.edgeLines = new THREE.LineSegments(
      new THREE.EdgeBufferGeometry(shape),
      new THREE.LineBasicMaterial({ color: 0x000000 })
    );
    scene.add(this.polygon);
    scene.add(this.edgeLines);
  }

  copy() {
    const copiedPolygon = new Polygon(scene, this.color);
    copiedPolygon.vertices = [...this.vertices];
    copiedPolygon.completePolygon();
    return copiedPolygon;
  }
}

// PolygonManager class
class PolygonManager {
  constructor(scene) {
    this.polygons = [];
  }

  createPolygon(color) {
```

```
    const polygon = new Polygon(scene, color);
    this.polygons.push(polygon);
    return polygon;
  }

  copyPolygon(polygon) {
    const copiedPolygon = polygon.copy();
    this.polygons.push(copiedPolygon);
    return copiedPolygon;
  }

  reset() {
    this.polygons.forEach((polygon) => {
      scene.remove(polygon.polygon);
      scene.remove(polygon.edgeLines);
    });
    this.polygons = [];
  }
}

// UI class
class UI {
  constructor() {
    this.completeButton = document.getElementById("complete-button");
    this.copyButton = document.getElementById("copy-button");
    this.resetButton = document.getElementById("reset-button");
  }

  addEventListeners(polygonManager) {
    this.completeButton.addEventListener("click", () => {
      // Complete polygon logic
```

```
    });
    this.copyButton.addEventListener("click", () => {
      polygonManager.copyPolygon(currentPolygon);
    });
    this.resetButton.addEventListener("click", () => {
      polygonManager.reset();
    });
  }
}

// Main class
class Main {
  constructor() {
    this.scene = new THREE.Scene();
    this.camera = new THREE.PerspectiveCamera(
      75,
      window.innerWidth / window.innerHeight,
      0.1,
      1000
    );
    this.renderer = new THREE.WebGLRenderer({
      canvas: document.getElementById("canvas"),
    });
    this.groundPlane = new GroundPlane(this.scene);
    this.polygonManager = new PolygonManager(this.scene);
    this.ui = new UI();
    this.ui.addEventListeners(this.polygonManager);
  }

  animate() {
    requestAnimationFrame(() => {
```

```
        this.renderer.render(this.scene, this.camera);
        this.animate();
      });
    }
  }

  // Initialize
  const main = new Main();
  main.animate();
 </script>
</body>
</html>
```