Task 9

Integrate a WebGL library (e.g., Three.js) to visualize the mesh and the deformed shape on the web interface.

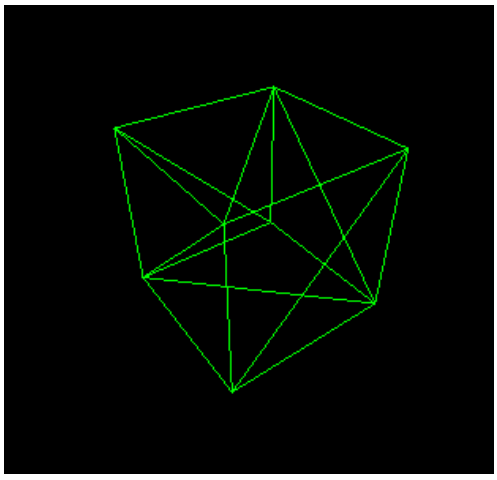
Task Code:

Code for Meshed Body Visualization:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Three.js Mesh Visualization</title>
    <style>
        body { margin: 0; }
        canvas { display: block; }
    </style>
</head>
<body>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/three.js/r128/three.min.js"></script>
    <script>
        var scene = new THREE.Scene();
        var camera = new THREE.PerspectiveCamera(75, window.innerWidth / 
window.innerHeight, 0.1, 1000);
        var renderer = new THREE.WebGLRenderer();
        renderer.setSize(window.innerWidth, window.innerHeight);
        document.body.appendChild(renderer.domElement);
```

```
    // Create a more complex geometry (BoxGeometry)
    var geometry = new THREE.BoxGeometry(2, 2, 2); // Width, Height, Depth
    var material = new THREE.MeshBasicMaterial({ color: 0x00ff00, wireframe: true }); // Apply wireframe rendering
    var mesh = new THREE.Mesh(geometry, material);
    scene.add(mesh);

    camera.position.z = 5;

    function animate() {
      requestAnimationFrame(animate);
      mesh.rotation.x += 0.01;
      mesh.rotation.y += 0.01;
      renderer.render(scene, camera);
    }
    animate();
  </script>
</body>
</html>
```

Output:



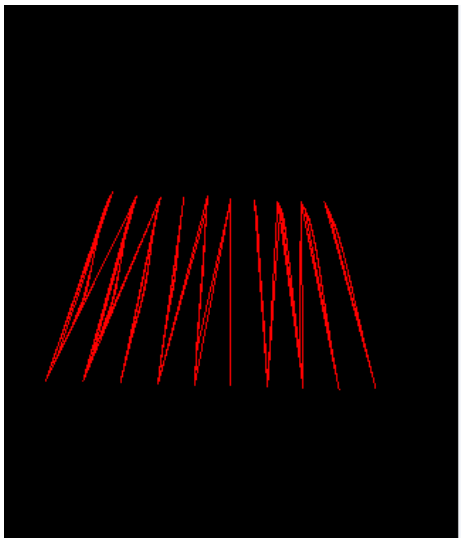
- **Code for Deformed Body Visualization:**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Complex Mesh Deformation with Perlin Noise</title>
    <style>
        body { margin: 0; }
        canvas { display: block; }
    </style>
</head>
<body>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/three.js/r128/three.min.js"></script>
    <script src="https://cdn.jsdelivr.net/gh/josephg/noisejs/perlin.js"></script>
    <script>
        var scene = new THREE.Scene();
        var camera = new THREE.PerspectiveCamera(75, window.innerWidth /
window.innerHeight, 0.1, 1000);
        var renderer = new THREE.WebGLRenderer();
        renderer.setSize(window.innerWidth, window.innerHeight);
        document.body.appendChild(renderer.domElement);


        var gridSize = 10;
        var cubeSize = 1;
        var geometry = new THREE.BufferGeometry();
        var vertices = new Float32Array(gridSize * gridSize * gridSize * 3);
```

```
for (var i = 0, j = 0; i < gridSize; i++) {
    for (var k = 0; k < gridSize; k++) {
        for (var m = 0; m < gridSize; m++) {
            var x = i * cubeSize - gridSize * cubeSize / 2;
            var y = k * cubeSize - gridSize * cubeSize / 2;
            var z = m * cubeSize - gridSize * cubeSize / 2;
            vertices[j++] = x;
            vertices[j++] = y;
            vertices[j++] = z;
        }
    }
}

geometry.setAttribute('position', new THREE.BufferAttribute(vertices, 3));
var material = new THREE.MeshBasicMaterial({ color: 0xFF0000, wireframe: true });
var mesh = new THREE.Mesh(geometry, material);
scene.add(mesh);

camera.position.y = 10;
camera.position.z = 10;
camera.lookAt(new THREE.Vector3(0, 0, 0));
function deformMesh(mesh, noiseStrength) {
    var positions = mesh.geometry.attributes.position;
    var time = performance.now() * 0.001;
    for (var i = 0; i < positions.count; i++) {
        var vertex = new THREE.Vector3()
```

```
  vertex.fromBufferAttribute(positions, i);

            var noiseValue = noise.perlin3(vertex.x * 0.2, vertex.y * 0.2, vertex.z * 0.2 + time) *
noiseStrength;

            vertex.y = noiseValue;

            positions.setXYZ(i, vertex.x, vertex.y, vertex.z);

          }

          positions.needsUpdate = true;

        }

        function animate() {

          requestAnimationFrame(animate);

          deformMesh(mesh, 1);

          renderer.render(scene, camera);

        }

        animate();

    </script>

</body>

</html>
```

Output:



In this code, a grid of cubes is created, and each cube is deformed using Perlin noise.