Computer Graphics Lab 11: Three.js

Lab Instruction: Three.js - Shooting Asteroids Game

Objective

By the end of this lab, students will:

- Gain hands-on experience with **Three.js** for 3D graphics rendering.
- Learn to create **basic game objects** (spaceship, bullets, asteroids).
- Implement **game physics** (movement, shooting mechanics, and collision detection).
- Add **shadows and lighting** for a realistic effect.
- Implement **lives and game over** logic.

Prerequisites

- Basic knowledge of **JavaScript**.
- Understanding of **Three.js basics** (scene, camera, renderer, meshes, materials).
- A web browser that supports **WebGL**.

Lab Steps

Step 1: Setting Up the Environment

Create an **HTML file** and include the Three.js library:
<script src="https://cdnjs.cloudflare.com/ajax/libs/three.js/r128/three.min.js"></script>

1. Define **scene, camera, and renderer**.
2. Append the renderer to the document body.

Step 2: Creating the Spaceship

- Use a **ConeGeometry** to represent the spaceship.
- Set its initial position below the screen.

- Add it to the scene.

```
let spaceshipGeometry = new THREE.ConeGeometry(0.5, 1, 8);
let spaceshipMaterial = new THREE.MeshStandardMaterial({ color: 0x00ff00 });
let spaceship = new THREE.Mesh(spaceshipGeometry, spaceshipMaterial);
spaceship.position.set(0, -4, 0);
scene.add(spaceship);
```

Step 3: Implementing Movement

- Capture **keyboard events** for left and right movement.
- Restrict movement within screen boundaries.

```
window.addEventListener("keydown", (event) => {
  if (event.key === "ArrowLeft") moveLeft = true;
  if (event.key === "ArrowRight") moveRight = true;
  if (event.key === " ") shoot = true;
});
```

Step 4: Shooting Bullets

- When the space key is pressed, **bullets** are created and move upwards.
- Remove bullets when they leave the screen.

```
function shootBullet() {
  let bulletGeometry = new THREE.CylinderGeometry(0.1, 0.1, 1, 8);
  let bulletMaterial = new THREE.MeshStandardMaterial({ color: 0xffff00 });
  let bullet = new THREE.Mesh(bulletGeometry, bulletMaterial);
  bullet.position.set(spaceship.position.x, spaceship.position.y + 0.5, 0);
  scene.add(bullet);
  bullets.push(bullet);
}
```

Step 5: Spawning Asteroids

- Periodically generate asteroids at random positions above the screen.
- Move them downward.
- Remove them when they move off-screen.

```
function spawnAsteroids() {
  setInterval(() => {
    let asteroidGeometry = new THREE.SphereGeometry(0.5, 8, 8);
```

```
    let asteroidMaterial = new THREE.MeshStandardMaterial({ color: 0xff0000 });
    let asteroid = new THREE.Mesh(asteroidGeometry, asteroidMaterial);
    asteroid.position.set((Math.random() - 0.5) * 10, 5, 0);
    scene.add(asteroid);
    asteroids.push(asteroid);
  }, 1000);
}
```

Step 6: Collision Detection

- Check if bullets collide with asteroids.
- Remove the asteroid and bullet when they collide.

```
bullets.forEach((bullet, bulletIndex) => {
  asteroids.forEach((asteroid, asteroidIndex) => {
    if (bullet.position.distanceTo(asteroid.position) < 0.5) {
      scene.remove(bullet);
      scene.remove(asteroid);
      bullets.splice(bulletIndex, 1);
      asteroids.splice(asteroidIndex, 1);
    }
  });
});
```

Challenge: Enhance the Game

Objective: Add advanced game mechanics

✅ Implement Lives System

- The spaceship starts with **3 lives**.
- If an asteroid hits the spaceship, a **life is lost**.
- If all lives are lost, display **"Game Over"** and stop the game.

✅ Display Lives on Screen

- Create an **HTML element** to show the remaining lives.
- Update it when lives decrease.

✅ Enable Shadows

- Enable shadows in **renderer, lights, and objects**.

✅ Add Lighting

- Use a **Directional Light** for a natural effect.
- Add an **Ambient Light** to brighten the scene.

✅ Create a Ground to Receive Shadows

- Add a **plane geometry** at the bottom of the scene.