Maze Runner – A 3D Interactive Maze Game

Course: CSC-317 – Computer Graphics

Instructor: *Dr. Muhammad Hataba*

Team Members:

- *Omar Mohamed Salah 320230162*
- *Mohamed Emad Mohamed 320230165*

1. Tools & Frameworks Used

The following tools and technologies were used in developing the game:

- **Programming Language:** JavaScript (ES6)
- **Graphics Library:** Three.js
- **Rendering Technology:** WebGL
- **Development Environment:** Visual Studio Code
- **Browser:** Google Chrome
- **Version Control:** Git & GitHub
- **Assets:** Custom primitive geometries and free sound effects

Three.js was chosen because it provides a high-level abstraction over WebGL while still exposing core computer graphics concepts such as transformations, cameras, lighting, and real-time rendering.

2. Game Overview & Story / Objective

Maze Runner is a simple 3D third-person exploration game where the player navigates through a maze environment.

Objective:

- Collect all coins inside the maze.
- Find the exit cube.
- Reach the exit in the shortest possible time.

The game ends when the player successfully reaches the exit. The final completion time and score are displayed on screen.

3. Graphics Techniques Used

3.1 Lighting

The game uses two types of lighting:

- **Ambient Light:** Provides uniform lighting for the entire scene.
- **Directional Light:** Simulates sunlight and adds realistic shading and depth.

This combination improves visibility while maintaining realism.

3.2 Camera Transformations

A dynamic third-person camera is implemented:

- The camera follows the player smoothly.
- Camera rotation is controlled using mouse movement.
- The camera uses **Perspective Projection**.

Transformations used:

- Translation
- Rotation
- View and Projection matrices (handled internally by Three.js)

3.3 Texture Mapping

Although no external textures are used, different **materials and colors** are applied to objects:

- Player: distinct color for visibility
- Walls: solid color to define maze structure
- Coins: golden color
- Exit: emissive green material for clarity

3.4 Shaders

The default **Phong shading model** provided by Three.js is used through MeshStandardMaterial.
This enables:

- Diffuse lighting
- Specular highlights
- Realistic surface shading

3.5 Animation

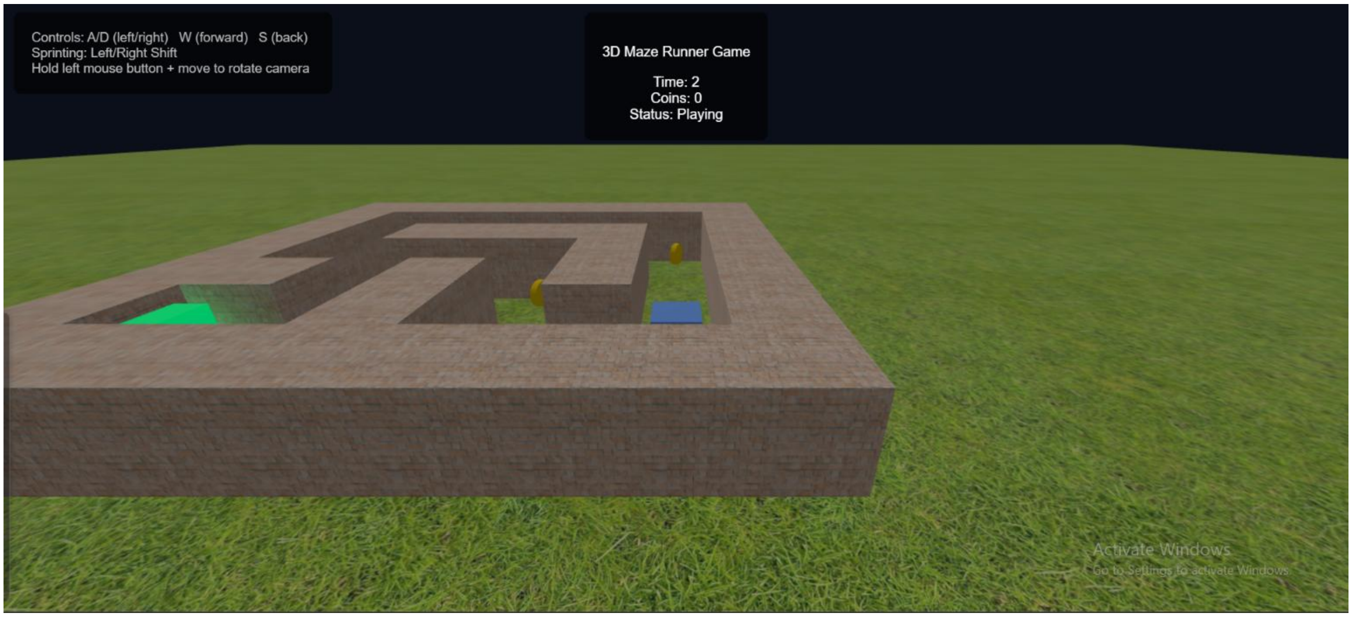
Animations implemented include:

- Rotating coins
- Smooth camera movement
- Player movement
- Sound effects triggered by events
- Exit glowing effect

These animations run in real time using the animation loop.

4. Screenshots of the Game

1. Full maze view

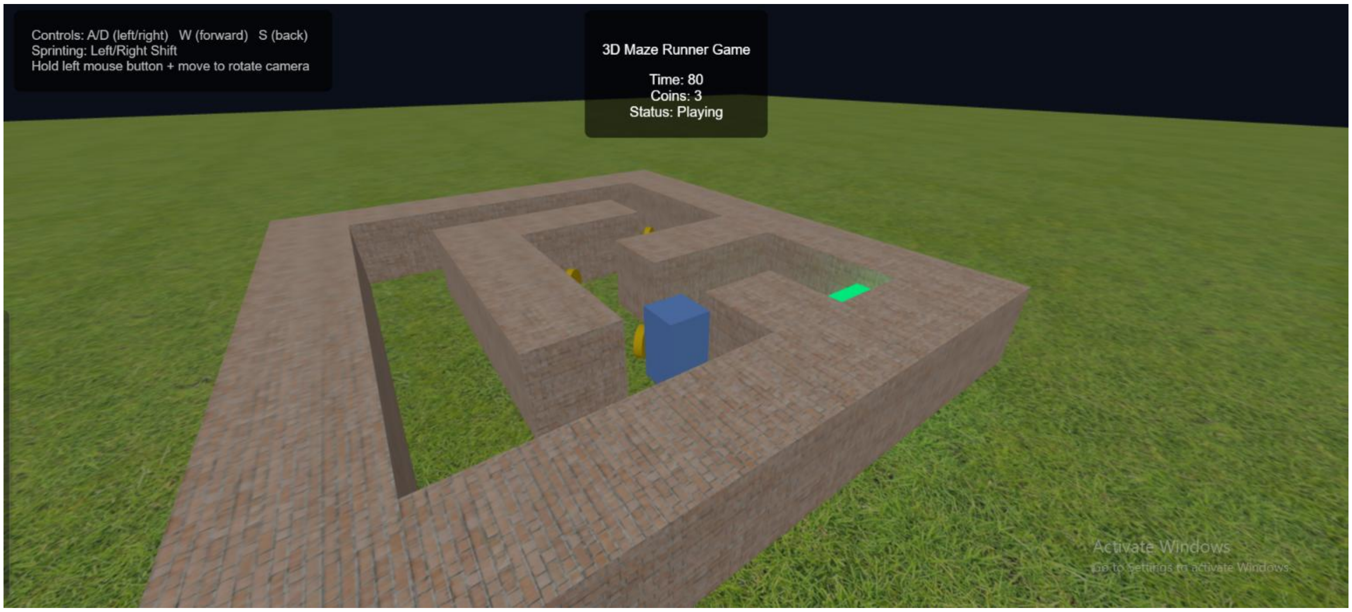


Controls: A/D (left/right) W (forward) S (back)
Sprinting: Left/Right Shift
Hold left mouse button + move to rotate camera

3D Maze Runner Game

Time: 2
Coins: 0
Status: Playing

2. Player collecting a coin

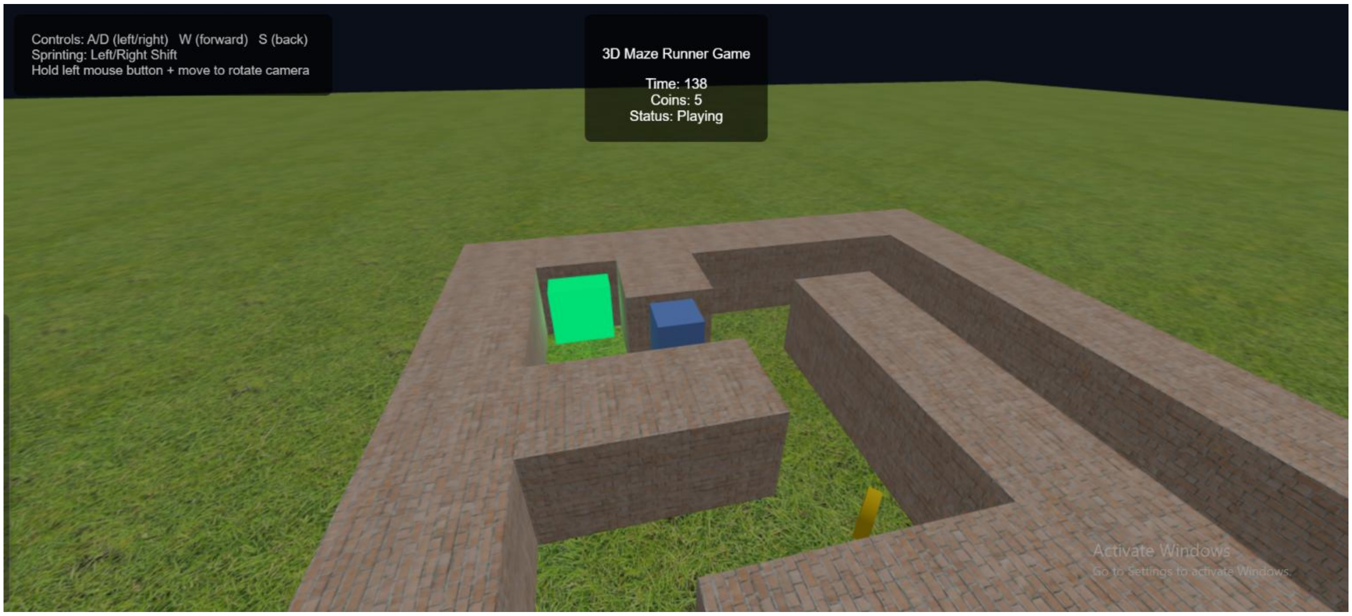


Controls: A/D (left/right) W (forward) S (back)
Sprinting: Left/Right Shift
Hold left mouse button + move to rotate camera

3D Maze Runner Game

Time: 80
Coins: 3
Status: Playing

3. Exit cube

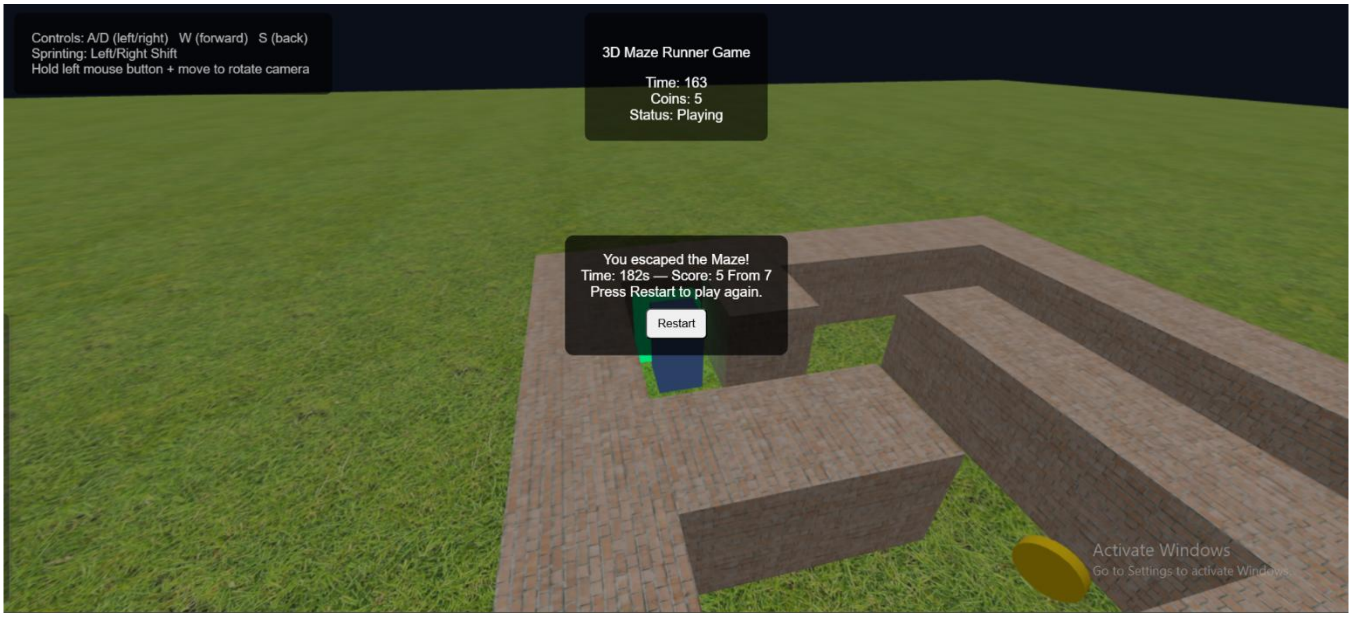


Controls: A/D (left/right) W (forward) S (back)
Sprinting: Left/Right Shift
Hold left mouse button + move to rotate camera

3D Maze Runner Game

Time: 138
Coins: 5
Status: Playing

4. Win screen with timer and score



Controls: A/D (left/right) W (forward) S (back)
Sprinting: Left/Right Shift
Hold left mouse button + move to rotate camera

3D Maze Runner Game

Time: 163
Coins: 5
Status: Playing

You escaped the Maze!
Time: 182s — Score: 5 From 7
Press Restart to play again.

5. How to Run the Game

1. Download or clone the GitHub repository.
2. Ensure all files are in the same directory.
3. Install a simple static server (if not available): ***npm install -g http-server***.
4. Run the server in the project directory: ***http-server*** (default port 8080).
5. Open http://localhost:8080 in a modern web browser.
6. Controls:
   - **W / A / S / D or Arrow Keys:** Move player
   - **Mouse Drag:** Rotate camera
   - **Left / Right Shift:** Player sprint
   - **Restart Button:** Restart game

No additional installation is required.

---

7. Division of Work Among Team Members

Team Member	Responsibilities
Omar	Game logic, player movement, collision detection
Mohamed	Camera system, maze design, sound integration, UI

8. Demo Video Link

Link: Demo Video Link

---

9. GitHub Repository Link

Link: GitHub Repository Link