CS324 Coursework Report

¡Your Name¿ ¡Student ID¿

January 11, 2026

1 How to Run

1.1 Environment

- Tested browser: **Google Chrome** (department Linux machines).
- The game must be served via a local web server (ES modules / asset loading may fail under `file://`).

1.2 Start a local server

Open a terminal, go to the **Code** directory, and run any local web server. For example:

- `python3 -m http.server 8000`

You may use any other server available on the department machines; the only requirement is that it serves the project folder over HTTP.

1.3 Launch the game

1. Open Chrome and navigate to:

   ```
   http://localhost:8000/
   ```

2. From the directory listing, open the game entry:
   - Navigate to `Game/` and open the entry page (e.g., `index.html` / `Game.html`).
   - If your packaged structure uses a different entry, follow the same path (e.g., `Solution/Game/...`).
3. You will see the main menu page. Click [**START / PLAY / xxxx**] to enter the game.

1.4 Controls (as shown in the in-game instructions)

- Mouse: look around (FPS pointer-lock style).
- **F**: toggle flashlight.
- **4**: switch to grenade and throw (grenade provides a red point light + emissive material).
- Other controls (movement / weapon switching / firing) are described in the menu instruction section.

If the mouse cursor is locked, press **Esc** to release the pointer lock and return to menu interaction.

2 Game Overview

This project is a first-person shooter (FPS). The player explores a level and eliminates zombies using multiple weapons. The win condition is achieving **30 zombie kills**, after which the objective is considered completed.

The gameplay loop is:

1. Spawn into the selected environment.
2. Use FPS controls to aim and attack zombies.
3. Track kill progress and reach 30 kills to complete the objective.

3 Implementation Details

3.1 Levels (two distinct environments)

The game includes two uniquely designed environments:

- **Level 1**: the primary gameplay map.
- **Shooting Range**: a separate training/testing scene with a distinctly different layout and visual appearance.

Both environments are designed to look **distinctively different**, and each level uses **distinct textures**, satisfying the "two unique levels + distinct texture per level" requirement.

3.2 Lighting (ambient + multiple non-ambient sources per level)

Lighting is implemented with:

- **Ambient light** in the scene to provide baseline illumination.
- A **sun light** in both Level 1 and the Shooting Range (implemented as a directional light source).

In addition, the player can introduce independent dynamic light sources:

- **Flashlight** (toggle with **F**): implemented as a **SpotLight** that follows the player's camera position and view direction, producing a focused beam effect suitable for FPS navigation.
- **Grenade light** (weapon switch with **4**, then throw): the grenade acts as a moving object with a **red PointLight** attached. The grenade material also uses **red emissive** properties so it visibly glows, reinforcing the light source both physically (light) and visually (material emission).

This setup ensures each level includes ambient light plus at least two distinct non-ambient light sources (sun + flashlight / grenade), meeting the light source requirement.

3.3 Camera / View Control

The game uses an FPS camera controlled by the mouse (pointer lock). A **pitch (vertical look) clamp** is applied to prevent unnatural rotation and keep the world orientation stable in the player's view.

Because this is an action game with a mouse-controllable viewpoint, it satisfies the camera requirement as specified.

4 References

List all external code, libraries beyond lab materials, models, textures, and tutorials you used. Include URLs and clearly indicate what each source was used for. Also ensure attribution appears in the code where relevant.

- Three.js (if applicable): `https://threejs.org/` — rendering framework.
- ¡Asset/Texture/Model source¿ — ¡what it was used for¿.
- ¡Tutorial/Code reference¿ — ¡what it was used for¿.