LABYRINTH NAVIGATOR

Shadow Passage

A First-Person 3D Horror Maze Game

Built with Three.js

Computer Graphics Project Report

Team Members:

Ahmed Abdelkader - 320230035
Eyad Mostafa - 320230028
Omar Momtaz - 320230071

Egypt – Japan University of science and technology
(E-JUST)

CSIT

CSC-317 - Computer Graphics & Visualization

December 2025

1. Game Title & Team Information

1.1 Game Title

Labyrinth Navigator: Shadow Passage

A first-person horror maze game that challenges players to escape from three progressively terrifying levels while being hunted by an AI-controlled entity inspired by SCP-173 and the Weeping Angels from Doctor Who.

1.2 Team Members & Roles

Name	Student ID	Primary Responsibilities
Ahmed Abdelkader	320230035	Textures, Presentation & Full Development of Level 2
Eyad Mostafa	320230028	Project Base Structure and Game Logic, and Full Development of Level 1
Omar Momtaz	320230071	Theme, Audio/Sound Effects, Navigation Menu, Report and Full Development of Level 3

2. Tools & Frameworks Used

2.1 Core Technologies

- **Three.js (r128)** - 3D graphics library built on top of WebGL for rendering the game world, handling meshes, materials, lighting, and camera systems
- **JavaScript ES6+** - Modern JavaScript with module system for code organization and maintainability
- **HTML5 Canvas & WebGL 2.0** - Rendering context providing hardware-accelerated 3D graphics
- **Web Audio API** - For 3D spatial audio and dynamic sound management

2.2 Browser APIs Utilized

- **Pointer Lock API** - Captures mouse movement for first-person camera control
- **requestAnimationFrame** - Provides 60 FPS game loop synchronized with browser refresh rate
- **Performance API** - Delta time calculations for frame-rate independent movement

2.3 Development Tools

- **VS Code** - Primary code editor with Live Server extension for local testing
- **Chrome DevTools** - Debugging, performance profiling, and WebGL inspection
- **Git & GitHub** - Version control and collaborative development

3. Game Overview & Story

3.1 Game Concept

Labyrinth Navigator is an immersive first-person horror maze game where players must navigate through three progressively challenging levels while racing against time. The game combines traditional maze navigation with psychological horror elements, culminating in a unique AI-controlled entity system inspired by SCP-173.

3.2 Story & Setting

You awaken in a dark, ancient labyrinth with no memory of how you arrived. The only way out is forward—through three increasingly sinister levels, each more twisted than the last. In the depths of the final level lurks the Trapped Soul, a malevolent entity that was once a previous victim who failed to escape. Now it hunts you, moving only when you're not looking. Your flashlight is your only defense, but you must keep moving to find the mystical crystal key and reach the exit portal before time runs out.

3.3 Core Objectives

Primary Goal:

- Find the green crystal key hidden in each level
- Reach the orange exit portal after collecting the key
- Complete the level before the timer reaches zero

Level 3 Additional Objective:

- Survive the Trapped Soul entity by periodically looking back (L key) to freeze it in your flashlight beam
- Manage the entity timer that resets when the entity is caught in your view

3.4 Level Progression

Level 1: Shadow Passage (Difficulty: Easy, Time: 60s)
A beginner-friendly maze with simple corridors and basic navigation challenges. Players learn the core mechanics of movement, key collection, and time management.

Level 2: The Haunted Halls (Difficulty: Medium, Time: 60s)
A complex labyrinth with dead ends, multiple moving obstacles, and non-linear pathfinding. The atmosphere intensifies with darker lighting and more oppressive audio.

Level 3: Trapped Soul (Difficulty: Hard, Time: 120s)
The ultimate challenge featuring all horror mechanics: pulsing flashlight, false echo audio, movement speed reduction, and the AI-controlled Trapped Soul entity that hunts the player. Success requires strategic planning, quick reflexes, and nerve management.

4. Graphics Techniques Used

4.1 Lighting System

4.1.1 Ambient Lighting

A low-intensity ambient light (intensity: 0.1) provides minimal global illumination to prevent complete darkness while maintaining the horror atmosphere. This creates a baseline visibility that emphasizes the importance of the flashlight.

4.1.2 Dynamic Flashlight (SpotLight)

The player's primary light source is a THREE.SpotLight with the following properties:

- **Base intensity:** 2.5 (increases visibility within cone)
- **Distance:** 50 units (effective range of illumination)
- **Angle:** 27° cone (Math.PI × 0.15 radians)
- **Penumbra:** 0.5 (soft edge falloff)
- **Position:** Attached to camera, moves with player view

4.1.3 Breathing Light Effect (Level 3)

Level 3 features a unique breathing effect that creates psychological tension through flashlight intensity modulation:

- **Algorithm:** Sine wave modulation of intensity and distance
- **Intensity range:** 1.0 to 4.0 (pulsing from dim to bright)
- **Frequency:** 2.5 Hz (simulates rapid heartbeat)
- **Distance variation:** ±10 units (enhances perceived instability)
- **Effect:** Creates anxiety as shadows appear to move on walls, tricking peripheral vision

4.2 Camera System & Transformations

4.2.1 First-Person Perspective Camera

The game uses a THREE.PerspectiveCamera with the following configuration:

- **Field of View:** 75° (provides immersive peripheral vision)
- **Aspect Ratio:** Dynamic (matches window dimensions)
- **Near Clipping Plane:** 0.1 units (prevents close object clipping)
- **Far Clipping Plane:** 1000 units (renders distant maze structures)

4.2.2 Camera Hierarchy & Parent-Child Relationships

The camera is attached as a child of an invisible player container object (THREE.Object3D). This hierarchy enables:

- **Position inheritance:** Camera moves with player object
- **Independent rotations:** Player rotates on Y-axis (horizontal looking), camera rotates on X-axis (vertical looking/pitch)
- **Eye-level positioning:** Camera offset at Y = 1.8 units above player ground position

4.2.3 Mouse Look Implementation

Mouse movement is captured via Pointer Lock API and translated into camera rotations:

- **Horizontal rotation:** Modifies player.rotation.y (yaw)

- **Vertical rotation:** Modifies camera.rotation.x (pitch) with clamping
- **Pitch limits:** ±90° (prevents camera flipping over)
- **Sensitivity:** 0.002 multiplier for smooth, controllable movement

4.2.4 L Key Look-Back Transformation (Level 3)

When the player presses the L key, the following transformation sequence occurs:

- **Store current rotation:** originalYRotation = player.rotation.y
- **Apply 180° rotation:** player.rotation.y += Math.PI
- **On E key release:** player.rotation.y = originalYRotation (instant return)

4.3 Texture Mapping

4.3.1 Wall Textures

Maze walls use brick texture with repeating pattern:

- **Wrapping mode:** THREE.RepeatWrapping (tiles seamlessly)
- **Repeat factor:** 2×1 (twice horizontal, once vertical)
- **Material:** THREE.MeshPhongMaterial with diffuse texture map

4.3.2 Floor Textures

Ground plane uses tiled stone texture:

- **Wrapping mode:** THREE.RepeatWrapping
- **Repeat factor:** 8×8 (creates detailed tiling across large floor)
- **Rendering:** Double-sided (visible from both above and below)

4.3.3 Entity Sprite (Level 3)

The Trapped Soul entity uses a 2D sprite billboard that always faces the camera:

- **Type:** THREE.Sprite with texture material
- **Billboard behavior:** Automatically rotates to face camera
- **Transparency:** Semi-transparent (opacity: 0.9) for ghostly effect
- **Scale:** Width 4 units × Height 6 units (imposing presence)

4.4 Materials & Shading

4.4.1 Phong Shading Model

All geometry uses THREE.MeshPhongMaterial for realistic lighting calculations:

- **Diffuse reflection:** Based on surface normal and light direction
- **Specular highlights:** Shininess values create reflective surfaces
- **Texture integration:** Diffuse map modulates base color

4.4.2 Emissive Materials

Key objects use emissive materials for self-illumination:

- **Key Crystal:** Green emissive (0x00ff00) with intensity 0.8
- **Exit Portal:** Orange emissive (0xdd5500) with intensity 0.6, transparent (opacity 0.8)

4.5 Shadow Rendering

4.5.1 Shadow Mapping Configuration

Real-time shadows enhance depth perception and horror atmosphere:

- **Algorithm:** PCFSoftShadowMap (Percentage Closer Filtering)
- **Shadow map resolution:** 1024×1024 pixels
- **Light shadow camera:** Near plane 0.5, far plane 10 (close-range shadows only)

4.5.2 Shadow Casting & Receiving

- **Cast shadows:** Walls, obstacles, key crystal
- **Receive shadows:** Floor plane, walls (creates depth)

4.6 Animations

4.6.1 Key Crystal Rotation

The key crystal continuously rotates on the Y-axis at 1 radian/second, making it easily identifiable from a distance.

4.6.2 Moving Obstacles

Obstacles use sinusoidal oscillation:

- **Movement pattern:** offset = sin(time × 1.5) × 3 units
- **Axis:** Z-axis (back and forth)
- **Collision update:** Bounding boxes recalculated each frame

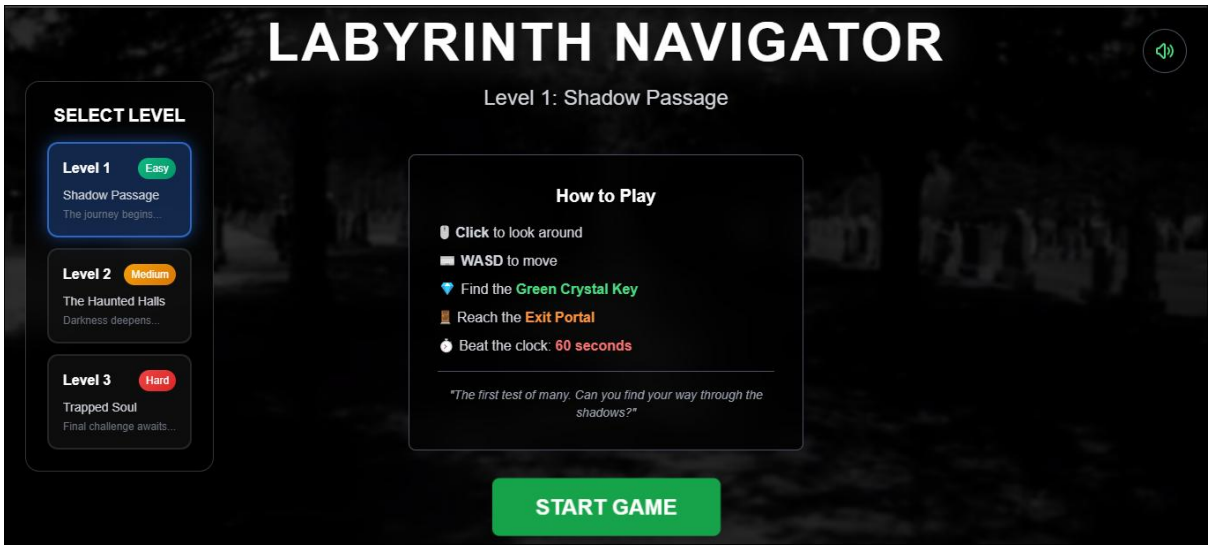
4.6.3 Entity Movement Animation (Level 3)

The entity uses frame-rate independent movement:

- **Speed:** 2 units/second (constant when not frozen)
- **Direction:** Normalized vector from entity to player
- **Freeze condition:** Movement stops when caught in player's flashlight view cone

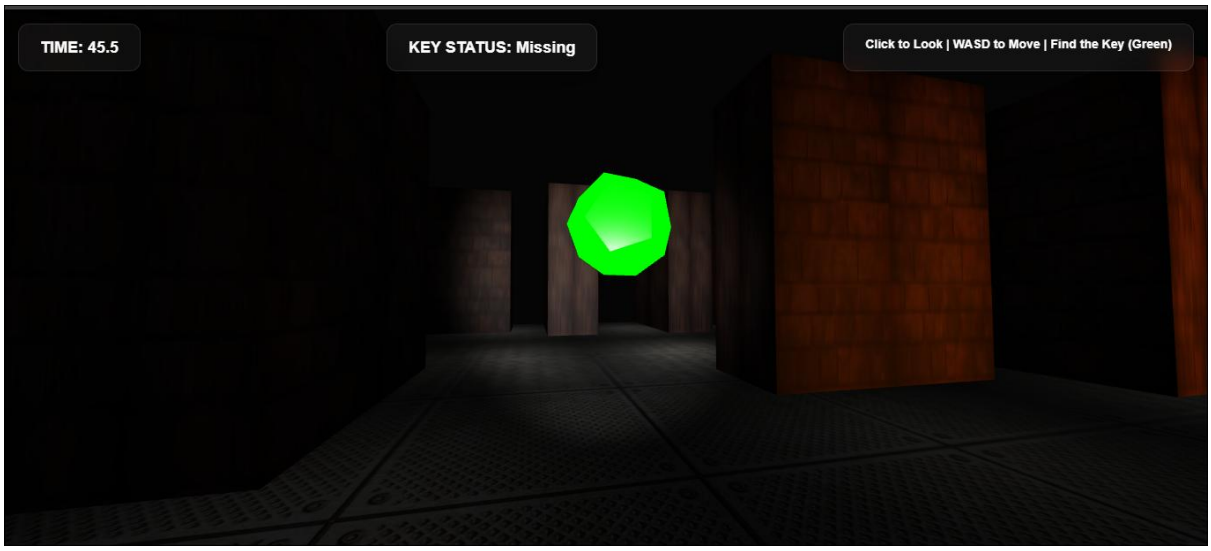
5. Game Screenshots

5.1 Start Screen & Level Selection Menu

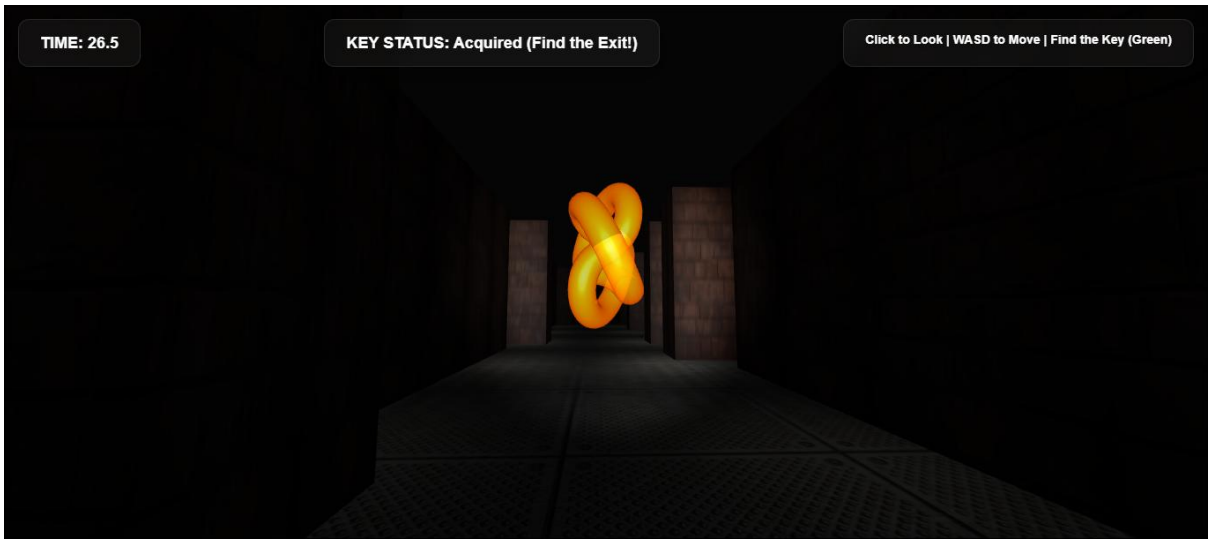


5.2 Level 1 Gameplay

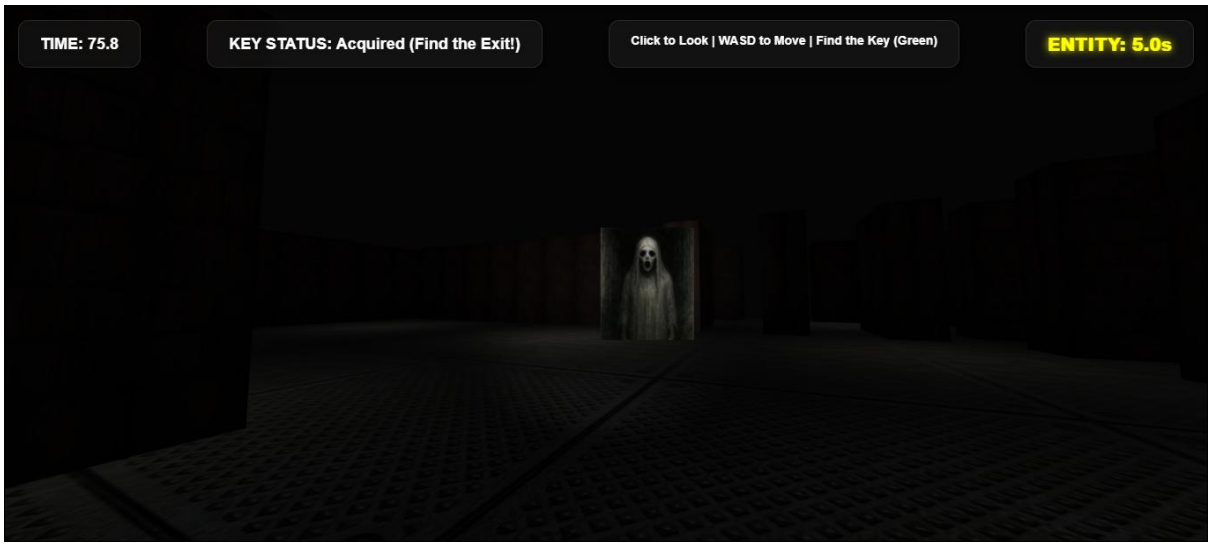
5.3 Key Crystal Collection



5.4 Exit Portal

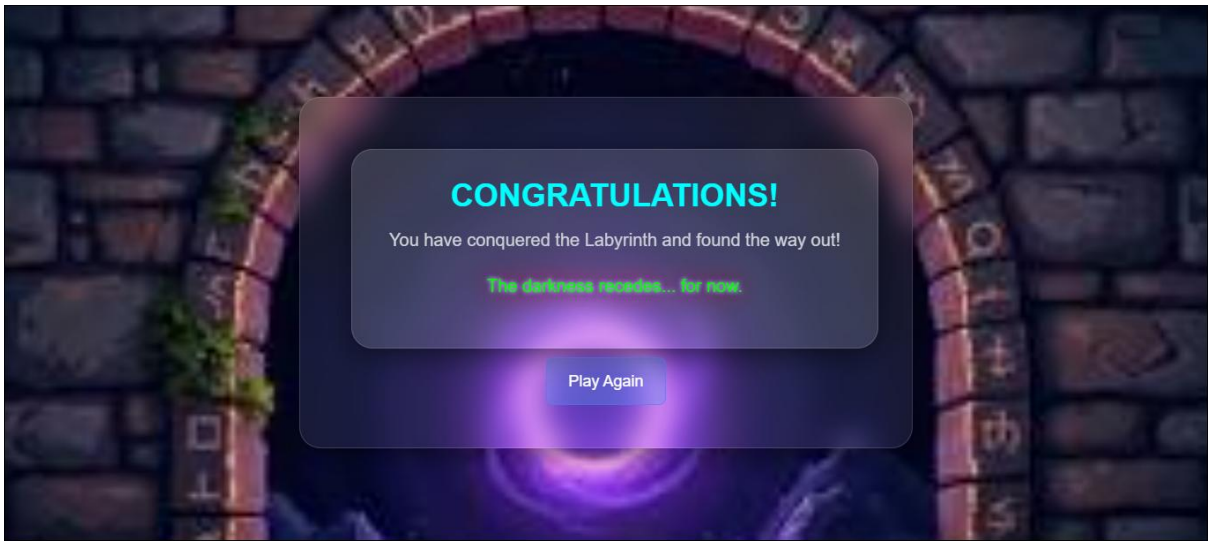


5.5 Level 3: The Entity



5.6 Level Transition Screen

5.7 Victory Screen



6. How to Run the Game

6.1 System Requirements

6.1.1 Minimum Requirements

- **Browser:** Modern browser with WebGL 2.0 support (Chrome 56+, Firefox 51+, Edge 79+, Safari 15+)
- **Graphics:** WebGL-compatible GPU
- **Display:** 1280×720 minimum resolution
- **Input:** Mouse and keyboard
- **Audio:** Headphones or speakers (recommended for spatial audio)

6.2 Installation & Setup

6.2.1 GitHub Repository Access

Link: *https://github.com/EyadMostafa/labyrinth-navigator/tree/main*
Step 1: Clone or download the repository:

```
git clone https://github.com/EyadMostafa/labyrinth-navigator.git
```

Step 2: Navigate to the project directory:

```
cd labyrinth-navigator
```

Step 3: Install dependencies:

```
npm install
```

6.2.2 Running the Game Locally

The game requires a local web server due to ES6 module imports and CORS policies. Choose one of the following methods:

Method 1: Python HTTP Server

```
python -m http.server 8000
```

Then open browser to: `http://localhost:8000`

Method 2: Node.js HTTP Server

```
npx http-server -p 8000
```

Then open browser to: `http://localhost:8000`

Method 3: VS Code Live Server Extension

- Install 'Live Server' extension in VS Code
- Right-click `index.html` → 'Open with Live Server'

6.3 Game Controls

6.3.1 Movement Controls

- **W** - Move forward
- **A** - Move left (strafe)
- **S** - Move backward
- **D** - Move right (strafe)

6.3.2 Camera Controls

- **Mouse Movement** - Look around (horizontal and vertical)
- **Click to Start** - Activate pointer lock for mouse look

6.3.3 Level 3 Special Controls

- **E Key (Hold)** - Turn 180° to look behind and freeze the entity in your flashlight
- **E Key (Release)** - Return to forward-facing view

6.4 Gameplay Tips

- Listen carefully to spatial audio cues to locate the key crystal
- Use your flashlight to explore dark corners thoroughly
- In Level 3, regularly press L to check if the entity is behind you
- Keep the entity timer above 1 second for safety
- Plan your route to the exit before collecting the crystal in Level 3

END OF REPORT

Thank you for reviewing our project!