Project Report
Creating a Spooky Haunted House in 3D

Course Title: Computer Graphics
Course Code: CSE420
Section: 01
Group: 04
Semester: Summer 25

Submitted to:
Dr. Md. Tauhid Bin Iqbal
Assistant Professor, Dept. of CSE
East West University

Submitted by:

Name	ID
Iffat Ara Arin	2021-3-60-286
MD. Ashikul Islam	2022-1-60-048
Farah Diba Islam Rodela	2022-3-60-160

Submission date: 03 August, 2025

Table of Content

1. Introduction

Problem Statement:

Creating a compelling 3D scene is one thing, but making it feel alive and interactive is a much bigger challenge. This project aimed to build more than just a static diorama; the goal was to create a mini haunted world in a web browser that responds to the user. The core problem was integrating various elements—realistic lighting, shadows, animated characters, and sound—into a single, cohesive, and performant experience that feels immersive and engaging.

Motivation and Objectives:

I've always been fascinated by how games and movies build atmosphere. I wanted to see if I could create a similar sense of mood and place using just web technologies. My main objectives were:

- To learn and apply fundamental 3D graphics concepts like lighting, materials, and animation in a practical project.
- To build a scene that is visually interesting from any angle, with attention to textural detail.
- To make it interactive, allowing users to change the scene and soundscape with simple keyboard commands.
- To ensure it runs smoothly in a standard web browser, making 3D graphics accessible without any special software.

2. Literature Review / Background Study

Relevant Concepts and Prior Works:

WebGL is the powerhouse that allows for hardware-accelerated 3D in browsers, but it's notoriously complex to work with directly. This is where Three.js comes in. It's a library that acts as a friendly translator, handling the complex math of WebGL so you can focus on designing your scene. Many fantastic Three.js examples online demonstrate specific techniques: beautiful landscapes, physics simulations, or product visualizers. I drew inspiration from these but noticed that many are either purely visual or demonstrate just one feature in isolation.

Gap in Existing Approaches:

I saw a gap for projects that combine several of these elements into a single, narrative-driven experience. I wanted a scene where the visuals tell a story (a haunted house), the audio enhances the mood (eerie sounds), and the user isn't just a passive viewer but an active participant who can influence the world (spawning bats, toggling sounds). This project is an attempt to bridge that gap and create a holistic, interactive demo rather than a technical proof-of-concept for a single feature.

3. Proposed Method / Approach

System Design / Methodology:

I approached this like building a physical set for a movie. First, I constructed the stage (the scene and floor). Then, I built and placed the main set pieces (the house, graves, bushes). Next, I set up the lighting rig to create the right mood. After that, I brought in the actors (the zombie and bat models) and directed their movements. Finally, I added the sound effects and music and gave the audience (the user) some controls to play with.

Graphics Algorithms and Models Used:

- **Transformations:** Every object's position, rotation, and size in the 3D space is calculated using matrix math (handled by Three.js).
- **Lighting Models:** I used a combination of lighting types: a dim "ambient" light to fake global illumination, a "directional" light like the sun to cast long shadows, and colored "point" lights attached to the ghosts to make them glow.
- **Procedural Animation:** Instead of pre-recorded animations, the ghosts' floating paths are calculated in real-time using Math.sin() and Math.cos() functions, creating smooth, endless wave-like patterns.
- **Shadow Mapping:** The lights calculate what they can see from their perspective, creating a depth map. The renderer then uses these maps to figure out which parts of the scene are in shadow.

Tools and Technologies:

- **Three.js:** The core framework for all 3D rendering.
- **GLTFLoader:** To import the 3D models of the zombie and the bat. GLTF is like the "JPEG for 3D."
- **TextureLoader:** To load all the image-based textures for surfaces like wood, stone, and sand.
- **OrbitControls:** A Three.js add-on that lets users click and drag to rotate, pan, and zoom the camera around the scene, just like in a 3D modeling program.

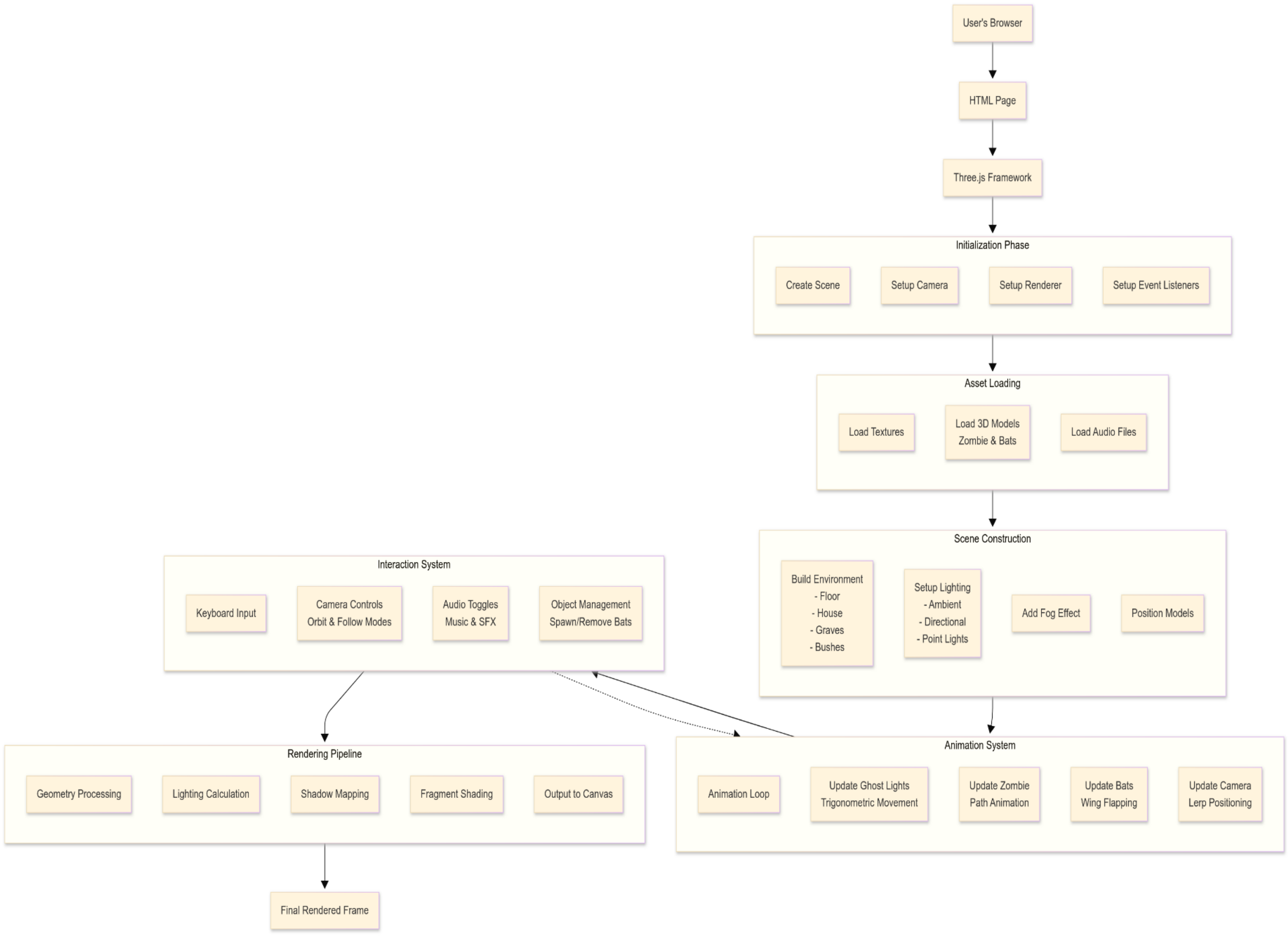
4. Implementation

Description of Project Modules:

- **The Environment:** The floor is a flat plane deformed with a displacement map to make it look uneven. Textures with roughness and ambient occlusion maps make the sand and stones look realistic.
- **The House:** Built from a simple cube for the walls and a cone for the roof, but transformed with detailed textures for broken bricks and slates. The door uses a displacement map to make the wood planks look deeply carved.
- **The Characters:**

    - The **Zombie** is a pre-animated model that walks on a loop. My code controls its direction and when it should turn around.
    - The **Bats** are also pre-animated. My code handles spawning them, placing them randomly in the sky, and playing their flapping wing animation.
- **The Atmosphere:** A faint blue fog is layered over the entire scene to make distant objects fade away, adding a sense of depth and mystery.
- **The Interactivity:** JavaScript "event listeners" wait for the user to press specific keys (like 'B' for bat or 'M' for music) and trigger the corresponding function.

Architecture of the system



Rendering Pipeline / Workflow:

1. **Setup:** The browser loads the HTML, CSS, and JavaScript files.
2. **Initialization:** Three.js sets up a WebGL renderer, a scene, and a camera.
3. **Loading:** All textures, models, and audio files are loaded asynchronously.
4. **Scene Building:** All objects are created, configured with materials, and added to the scene. Lights are created and configured to cast shadows.
5. **The Loop:** A function called tick() runs ~60 times per second (requestAnimationFrame). Every frame, it:
    - Updates the positions of the ghosts and the zombie.
    - Updates the camera if it's in "follow" mode.
    - Advances the animation mixers for the bat wings and zombie walk cycle.
    - Tells the WebGL renderer to draw the entire scene from the camera's perspective, calculating lighting and shadows in the process.

5. Results and Discussion

Visual Outputs:

The final scene is a cohesive and moody environment. The house looks weathered, the sand looks rough, and the ghost lights cast an eerie, colored glow on the graves and walls. The zombie shuffles relentlessly, and the bats flutter in place when spawned. The fog ties everything together, making the scene feel isolated.

User Interaction: (keyboard key)

M-> Play or pause the background music

P-> Play or pause Zombie sound

B-> Spawn Bat and R-> Remove Bat

C-> Toggle Camera Follow

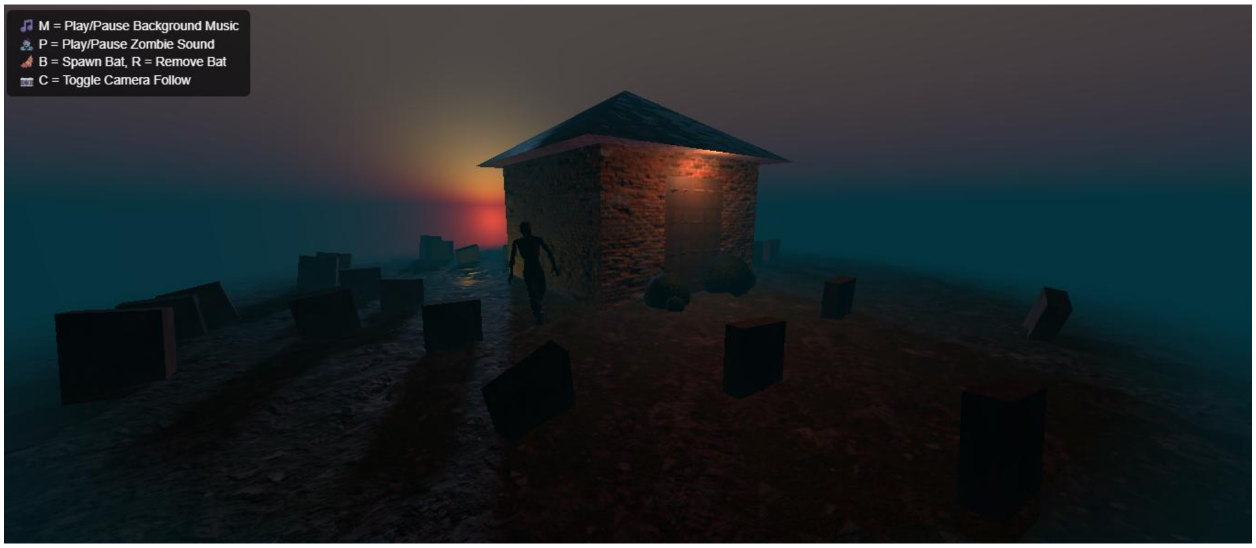

M = Play/Pause Background Music
P = Play/Pause Zombie Sound
B = Spawn Bat, R = Remove Bat
C = Toggle Camera Follow

Fig01: First Loaded Environment

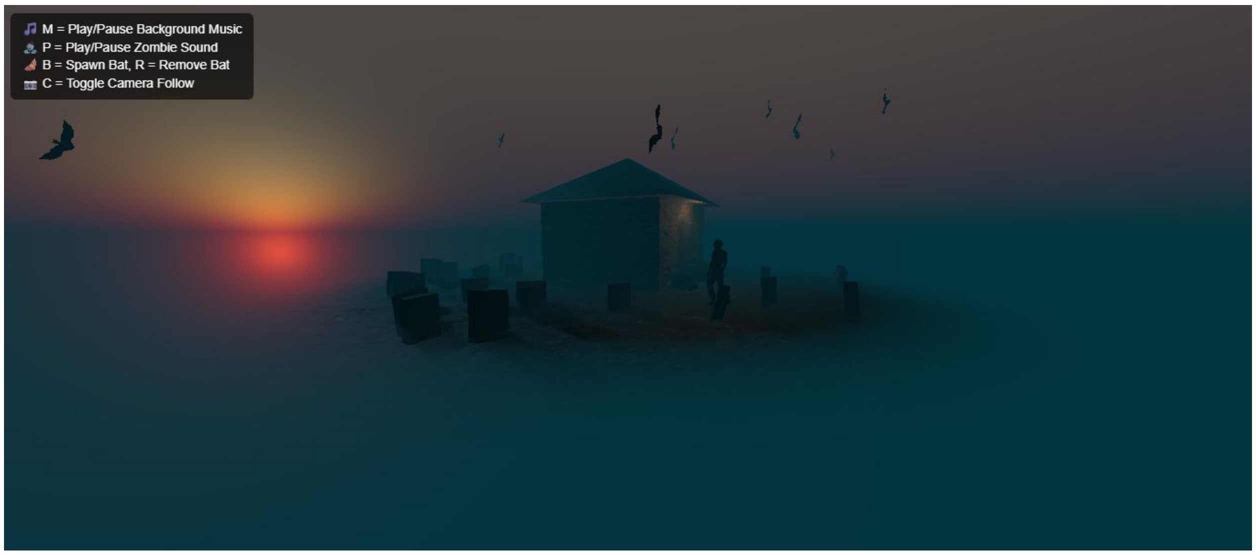

M = Play/Pause Background Music
P = Play/Pause Zombie Sound
B = Spawn Bat, R = Remove Bat
C = Toggle Camera Follow

Fig02: Bat Spawn using keyboard key “b”

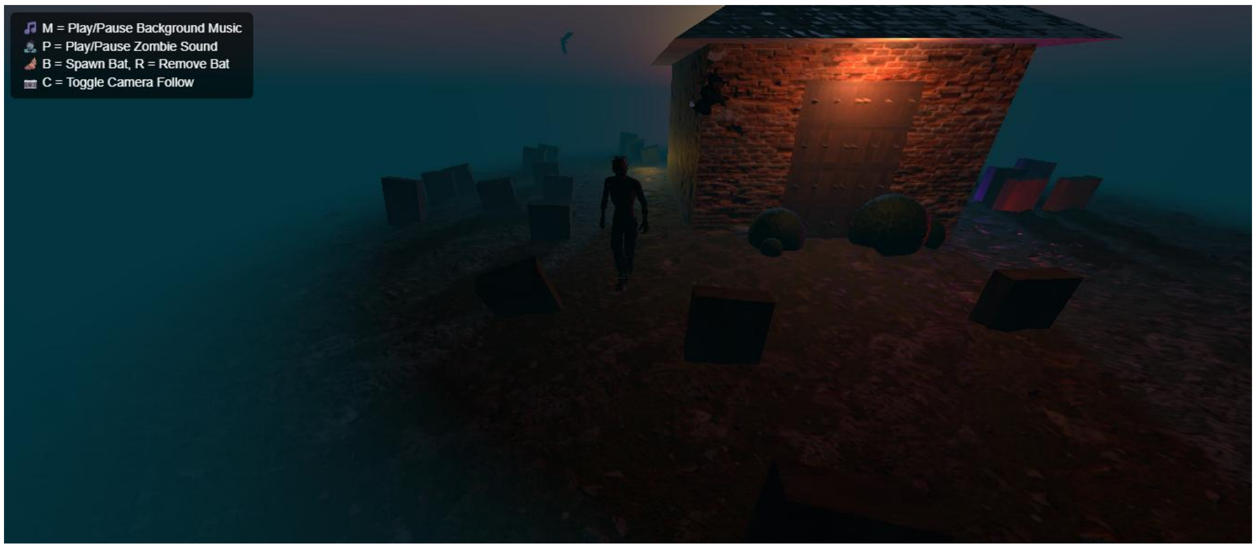
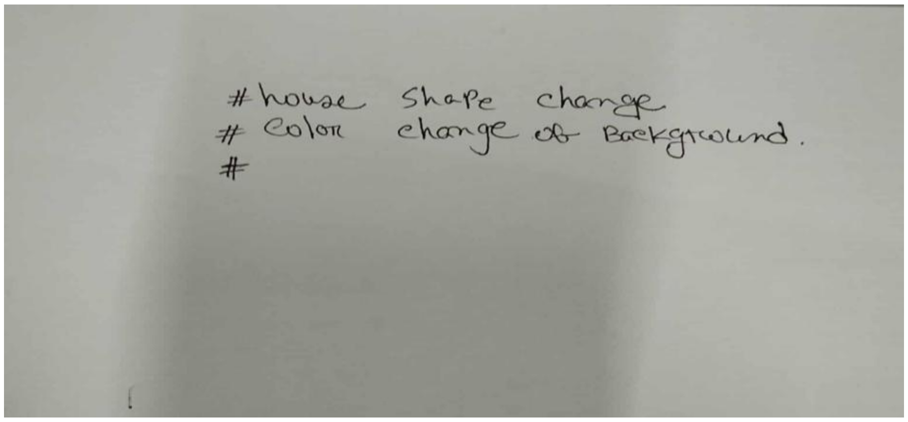

M = Play/Pause Background Music
P = Play/Pause Zombie Sound
B = Spawn Bat, R = Remove Bat
C = Toggle Camera Follow

Fig03: Toggle camera to follow the character

Performance Analysis:

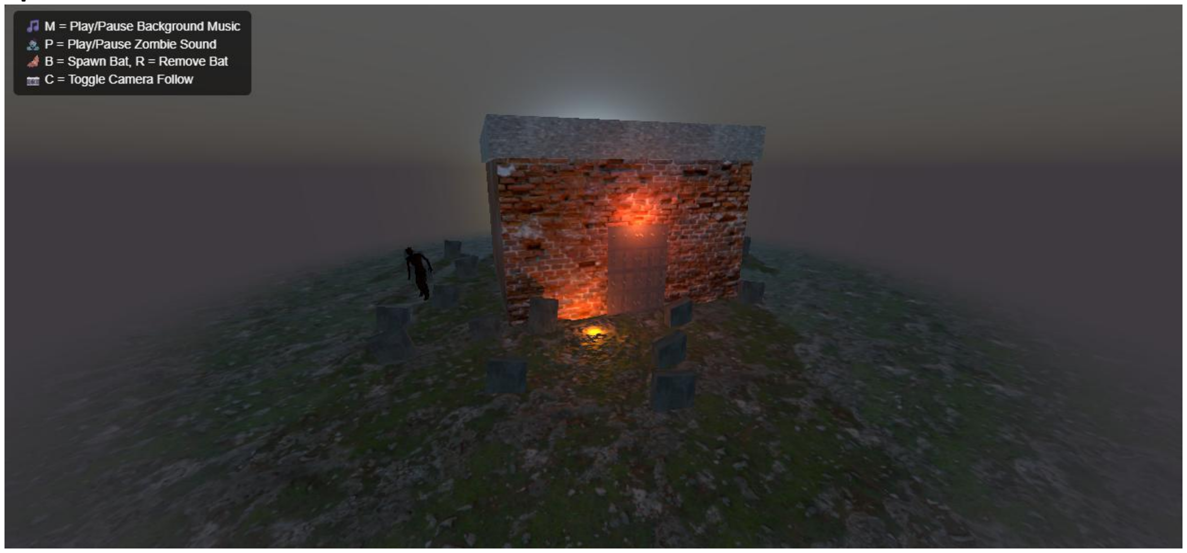
The scene performs very well on modern desktop computers and laptops, maintaining a smooth 60 frames per second. The main performance cost comes from calculating shadows. To keep things fast, I used lower-resolution shadow maps (256x256 pixels instead of 1024x1024). This is a common trade-off—slightly rougher shadows for a much smoother framerate.

Comparative results with alternative approaches



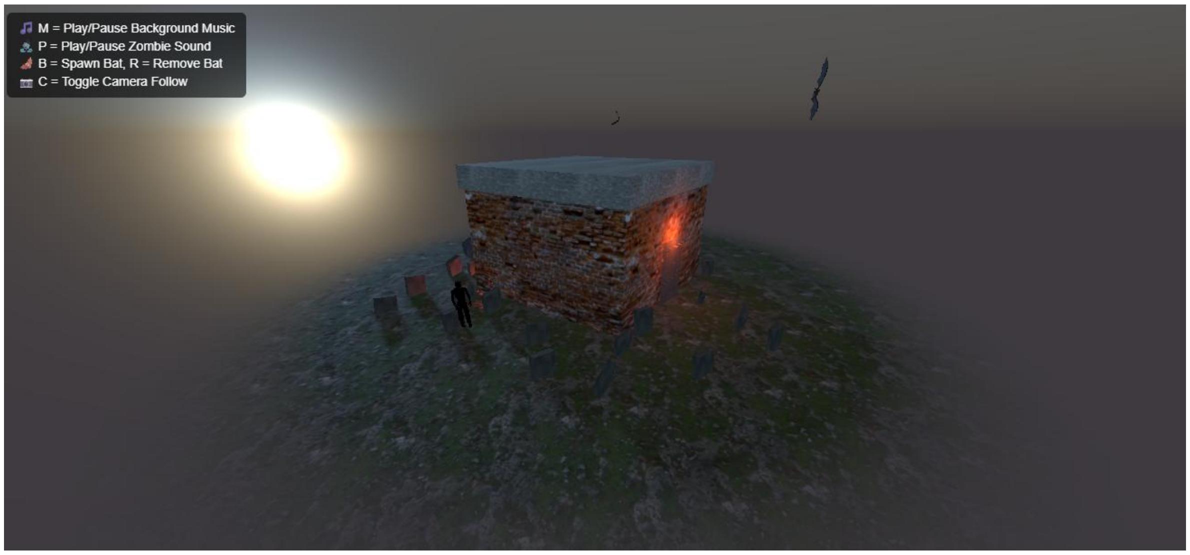
house Shape change
Color change of Background.
#

Output:



M = Play/Pause Background Music
P = Play/Pause Zombie Sound
B = Spawn Bat, R = Remove Bat
C = Toggle Camera Follow

Fig05: Home Shape change and Scale



M = Play/Pause Background Music
P = Play/Pause Zombie Sound
B = Spawn Bat, R = Remove Bat
C = Toggle Camera Follow

Fig06: Color Change of background and sun

Key Findings:

- **Three.js is Incredibly Powerful:** It truly democratizes 3D for the web, making complex techniques accessible.
- **Textures Make all the Difference:** A simple shape with a good texture can look incredibly detailed and realistic.
- **Sound is 50% of the Experience:** The atmosphere completely changes when you toggle the music and sound effects on and off. It's not an optional extra; it's essential for immersion.
- **Interactivity Creates Engagement:** Giving the user even simple controls makes them an active participant and greatly increases the time they'll spend exploring the scene.

6. Conclusion and Future Work

Summary of Contributions:

This project successfully created an interactive and atmospheric 3D experience that runs in a web browser. It serves as a practical demonstration of how to combine modeling, texturing, lighting, animation, and audio using the Three.js framework. It moves beyond a static display to become a dynamic world that the user can influence.

Limitations:

- The shadows are somewhat low-resolution and can appear jagged.
- The zombie's movement is very basic (just walking back and forth).
- The project is designed for a keyboard and doesn't work on touchscreens.
- There's no collision detection—bats and the zombie can clip through other objects.

Potential Future Improvements:

- **Mobile Support:** Redesign the interface for touch with on-screen buttons.
- **Improved Animations:** Add a turning animation for the zombie instead of it instantly flipping around.
- **More Interaction:** Let the user open the door, turn lights on/off, or have the zombie react to the player's presence.
- **Visual Enhancements:** Add post-processing effects like a subtle film grain, color grading, or antialiasing to make the image even smoother.
- **A Simple Story:** Add a text narrative or objectives to give the user a reason to explore further.

Appendix

Source Code:

https://github.com/ASHIK27445/Haunted-House-with-ThreeJS

Live Demo:

https://haunted-gamma.vercel.app/?fbclid=IwY2xjawMiTPFleHRuA2FlbQIxMQABHiiJsLqsxS-TdzX3A_NceCeXtNNVuyghW3wANGu9UF4RjgiOxsaNgMb7Dtwi_aem_-gmE7NaV8FMAMCvqiEVazw