Dreamwave

Bringing 3D Visuals to the Web Through Procedural Animation

CS 4361.002 - Computer Graphics

Final Project Report

Parthav Elangovan

PXE230002

1. Problem Summary

Most modern websites still rely heavily on 2D interfaces and flat graphics. While functional, these designs limit the depth, atmosphere, and sense of immersion that users can experience. With advances in web technologies such as WebGL and Three.js, it is now possible to create rich, interactive 3D environments that run directly in a browser without specialized software or downloads.

Dreamwave explores this potential by demonstrating how 3D visuals can dramatically enhance web experiences. The project creates a procedural, cinematic 3D environment inspired by James Bond title sequences, merging imported 3D assets with dynamic, real-time animation, lighting, and post-processing shaders.

The objective is to prove that web-based 3D graphics can be accessible, efficient, and artistically expressive, pushing the boundaries of what web visuals can achieve without requiring heavyweight game engines like Unity or Unreal.

2. Description of Work

2.1 Architecture Overview

The project is structured as a modular scene-based system. A main controller scene (main-bond.js) manages scene transitions, global timing, post-processing, and user controls. Each scene is implemented as a separate ES6 class with its own methods: init(), update(), updateCamera(), reset(), and render(). This architecture allows scenes to be developed, tested, and refined independently while maintaining consistent behavior across the entire experience.

2.2 Scene Implementations

Eight distinct cinematic scenes were created, each with unique visual themes and technical implementations:

1. **Gun Barrel Scene (scene1-gunbarrel.js):** Features a gothic church model and implements cinematic spotlight lighting with three-point illumination (main, rim,

and fill lights), smooth orbital camera movement, and fade-in/fade-out transitions.

2. **Silhouette Scene (scene2-silhouette.js):** A **two**-phase scene transitioning from a vintage car with red/orange lighting to a jet aircraft with blue lighting, featuring a horse statue. Implements dynamic color transitions, floating dice with procedural animation, particle systems, jet engine glow effects, and smooth phase transitions with color grading changes.
3. **Cards & Chips Scene (scene3-cards.js):** A casino-themed scene with a poker table, multiple colored chip stack, falling playing cards and a rolling gold chip that falls over. Features warm spotlight lighting simulating a casino environment with procedural card physics.
4. **Magic Cube Scene (scene3b-cubetest.js):** Showcases a rotating magic cube with purple/violet lighting that explodes into 27 smaller cubes. Implements screen shake effects, explosion flash, and fragment physics with velocity and drag,
5. **MP40 Cinematic Scene (scene4-liquid.js):** Features an MP40 submachine gun with intense red lighting, falling brass bullet casings, and floating dust particles. Multiple red-tinted spotlights create a sinister atmosphere.
6. **Knives Scene (scene5-diamonds.js):** A scene with five knives fanning out from a central pivot point against expanding concentric red rings. The knives animate from closed to fanned position while the background rings pulse and rotate.
7. **Gravestone Scene (scene6-rifle.js):** A graveyard scene with a central gravestone surrounded by scattered smaller tombstones, green-tinted lighting, floating green embers, and slowly descending skulls.

2.3 Technical Implementations

1. **Lighting Systems:** Each scene implements multi-light setups using SpotLight, PointLight, DirectionalLight, and RectAreaLight. Lights feature dynamic intensity modulation using sinusoidal functions to create breathing/pulsing effects. Shadow mapping is enabled with 2048x2048 resolution for realistic shadows.

2. **Camera Systems:** Orbital camera movement with parametric radius, height, and angle variations. Multi-phase camera paths with interpolation between keyframes.
3. **Post-Processing Pipeline:** EffectComposer with RenderPass, UnrealBloomPass (strength: 0.05, radius: 0.1, threshold: 0.98), custom chromatic aberration shader, and FilmPass for film grain effect. This creates the cinematic, stylized look characteristic of Bond title sequences.
4. **Particle Systems:** BufferGeometry-based point clouds for dust particles and embers. Particles animate using procedural noise functions affecting position over time.
5. **Asset Loading:** GLTFLoader for importing 3D models (church, car, jet, horse, cube, gun, bullets, knives, gravestone, skulls). Automatic bounding box calculation and scaling to normalize model sizes. Material property adjustments for metalness and roughness.

2.4 Challenges Encountered

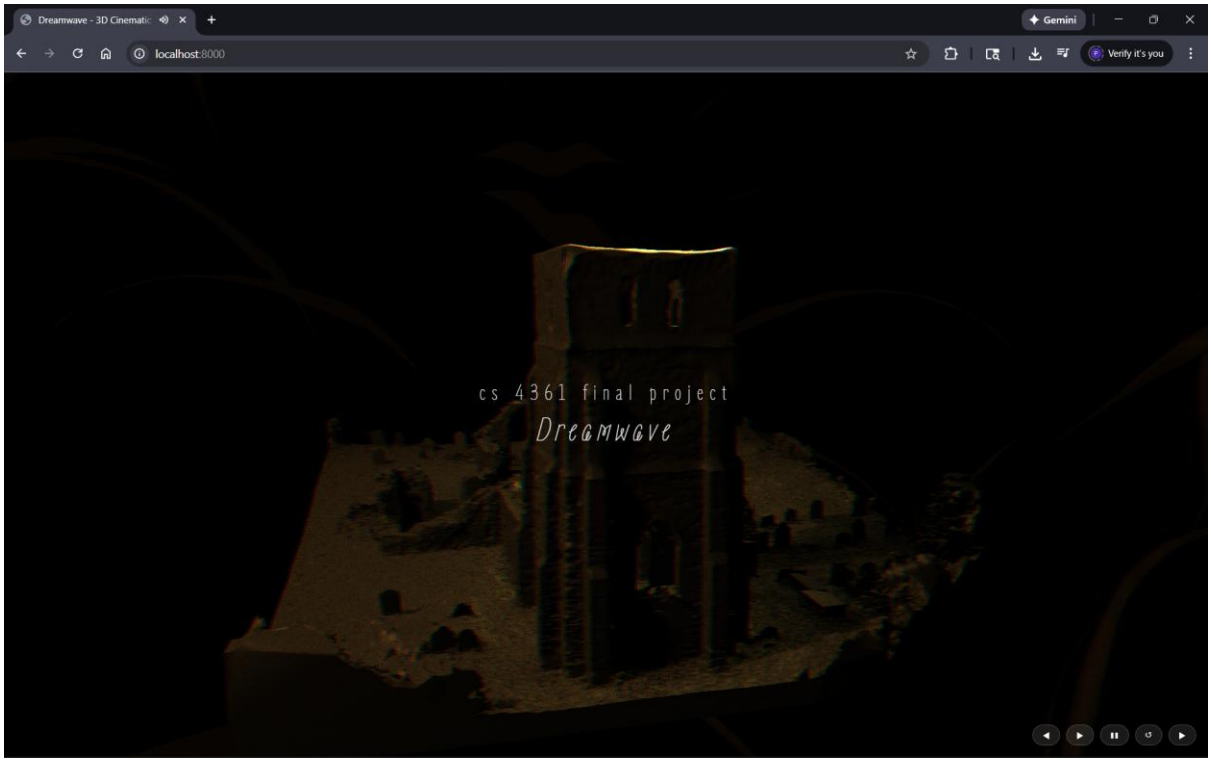
1. **Model Orientation and Scaling:** Different GLTF models came with varying orientations and scales. Solution: Implemented automatic bounding box calculation and normalized scaling to ensure consistent sizing across all assets.
2. **Lighting Balance:** Achieving the right balance between dramatic lighting and model visibility proved challenging. Too dark made models invisible; too bright lost the atmospheric effect. Solution: Iterative adjustment with visible light fixtures (glowing spheres) combined with ambient light as a base.
3. **Smoke/fog:** One of the main challenges was adding fog to the scenes to improve visual appeal. Since Three.js didn't have a built-in fog effect available for my setup, I had to use fog textures on planes instead. This didn't look very good because they were flat images rather than actual particle-based fog.
4. **Performance Optimization:** Multiple particle systems and complex lighting could cause frame drops. Solution: Limited particle counts, used

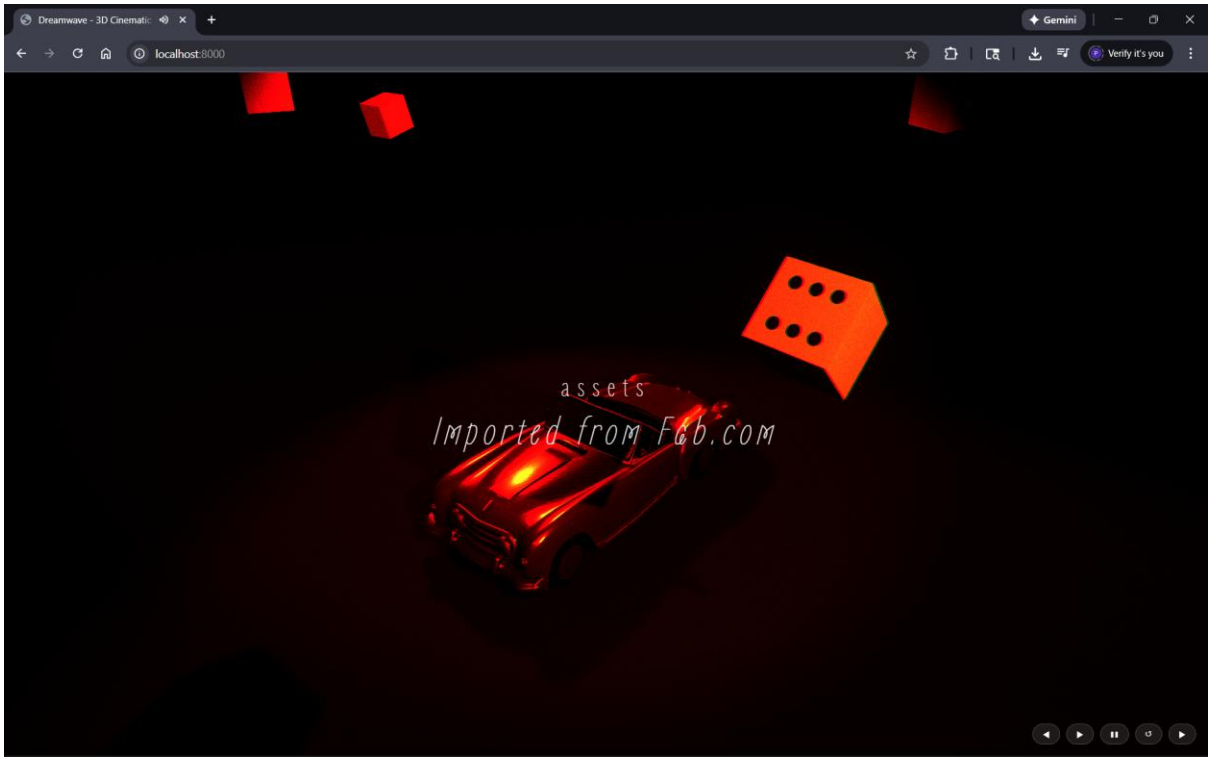
BufferGeometry for efficient updates, and disabled shadows on non-essential objects.

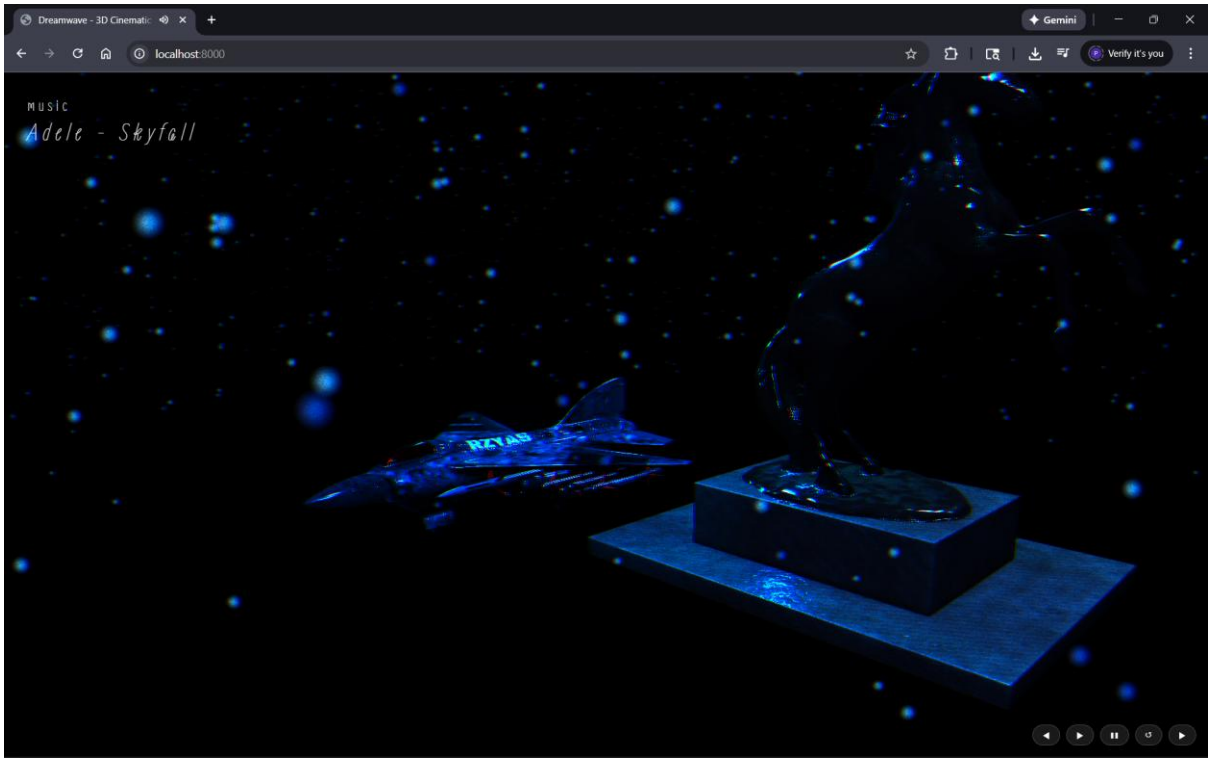
3. Results

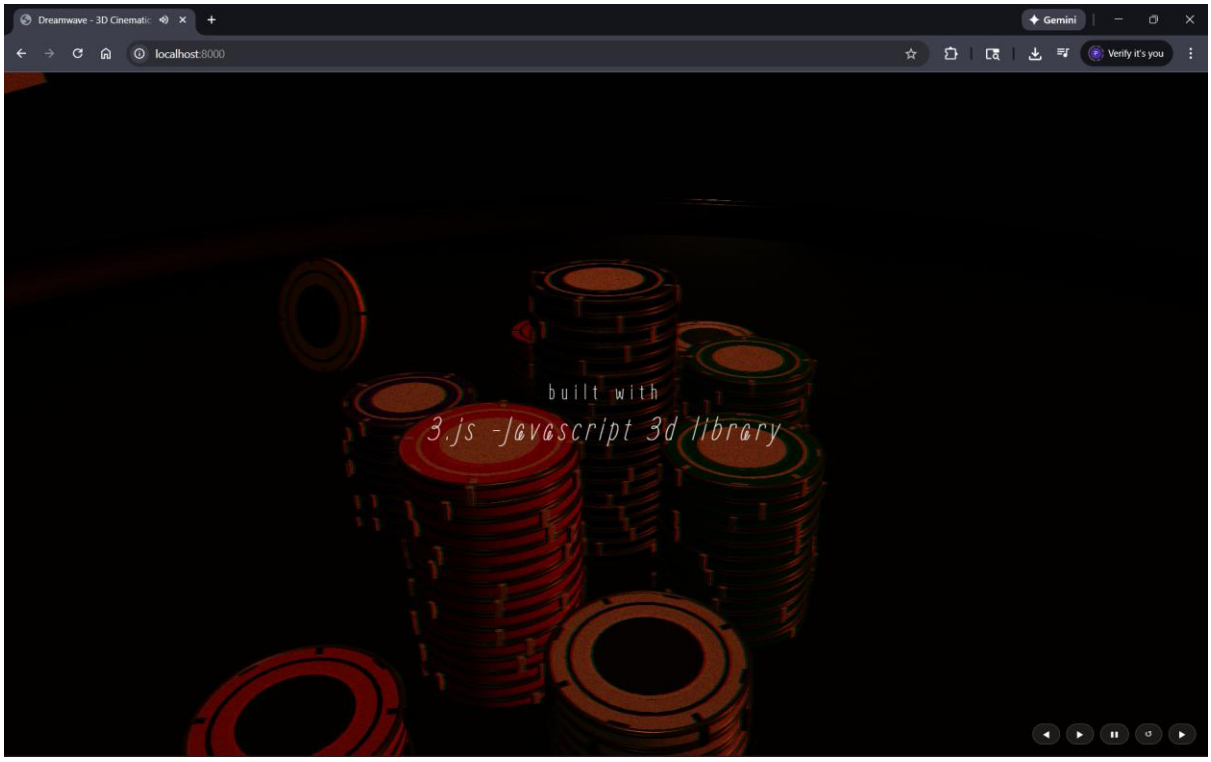
The final deliverable is a fully functional, browser-based 3D cinematic experience running approximately 96 seconds (8 scenes × 12 seconds each). The experience includes:

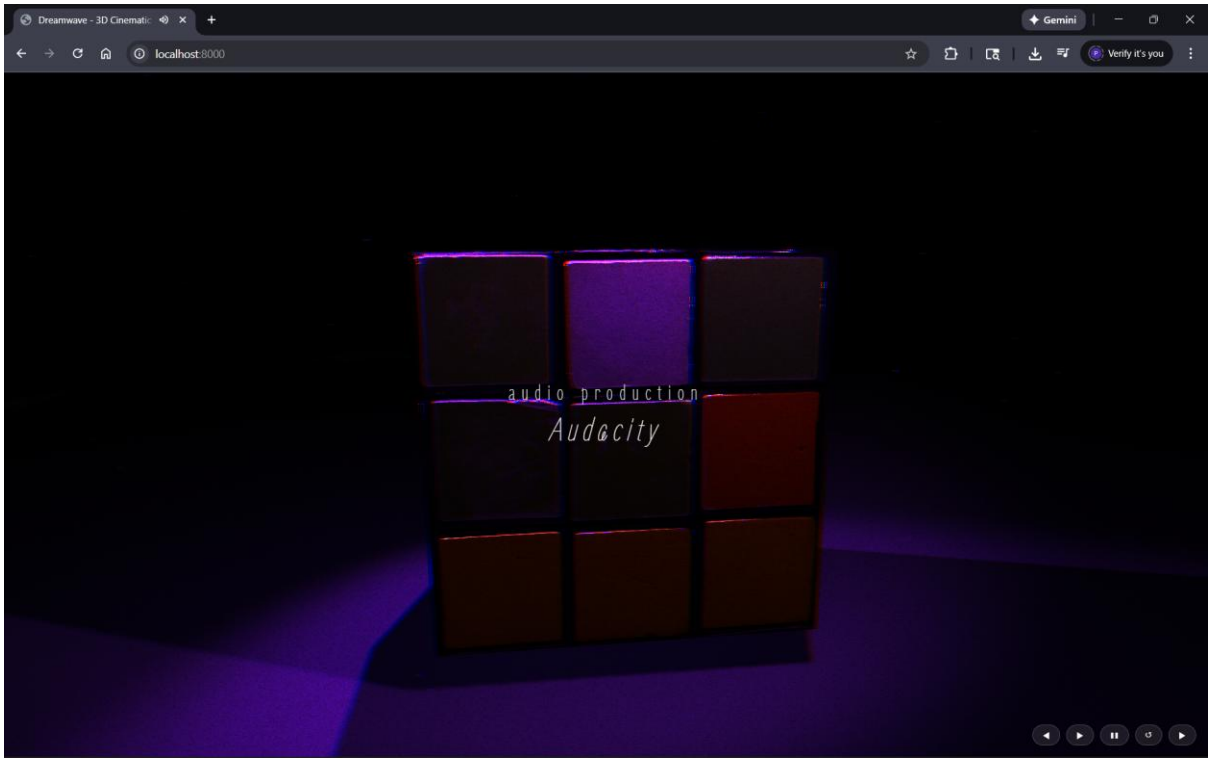
- Eight distinct cinematic scenes with unique visual themes and lighting
- Smooth transitions between scenes with fade effects
- Post-processing effects including bloom, chromatic aberration, and film grain
- Synchronized background music (Adele - Skyfall)
- Text overlays displaying credits and scene information
- Start screen and end screen with replay functionality
- Custom styling with the Zen Loop font for a cohesive aesthetic

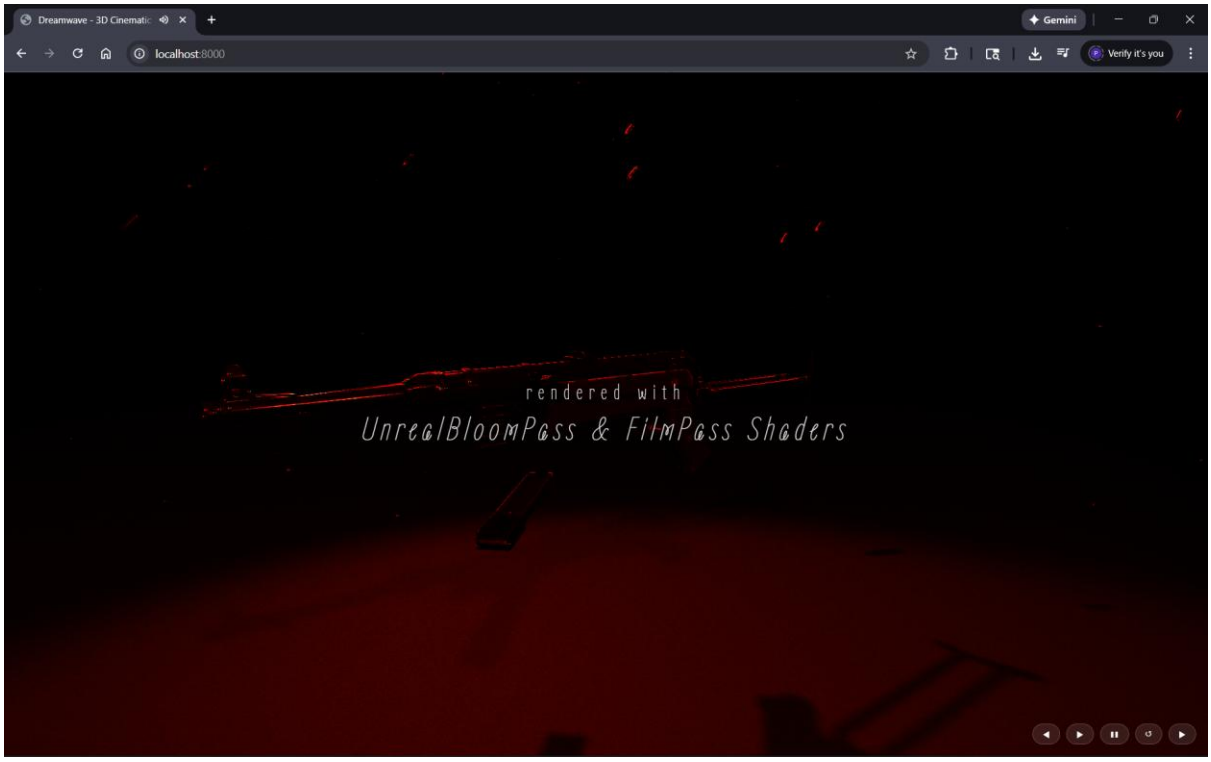


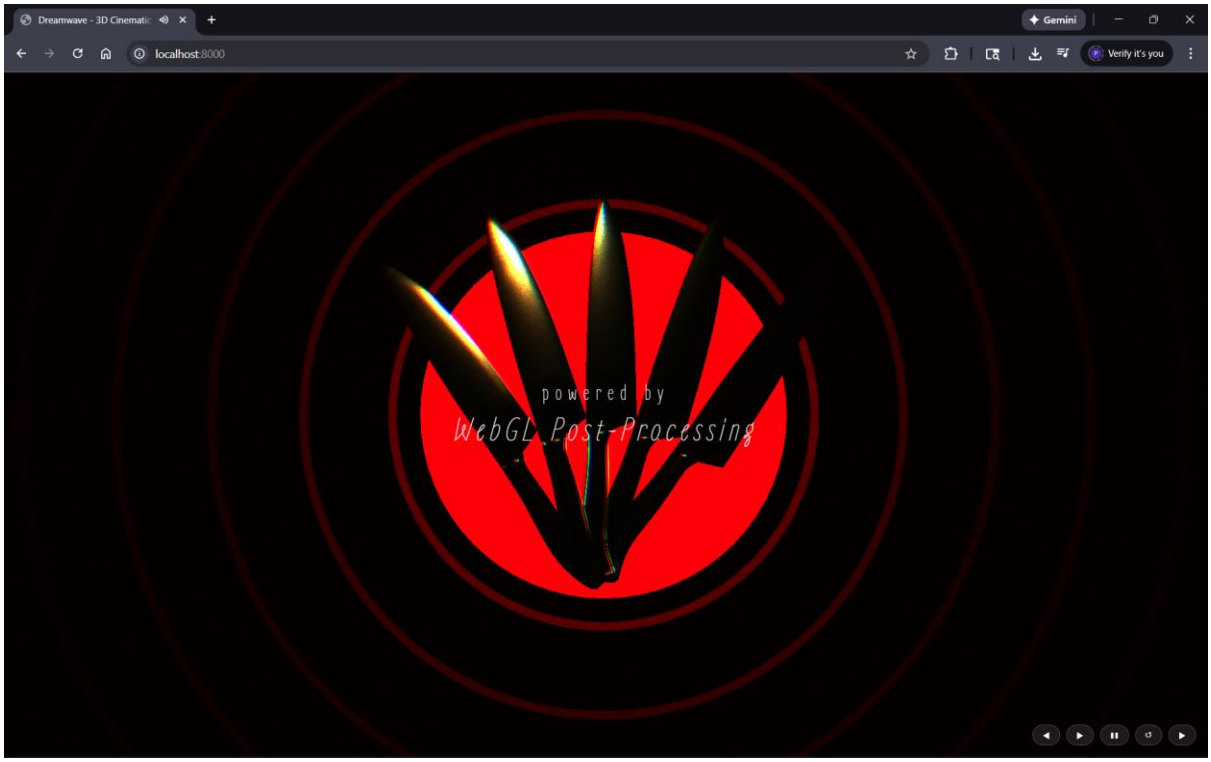

Dreamwave - 3D Cinematic
Gemini
localhost:8000
Verify it's you
assets
Imported from Fab.com

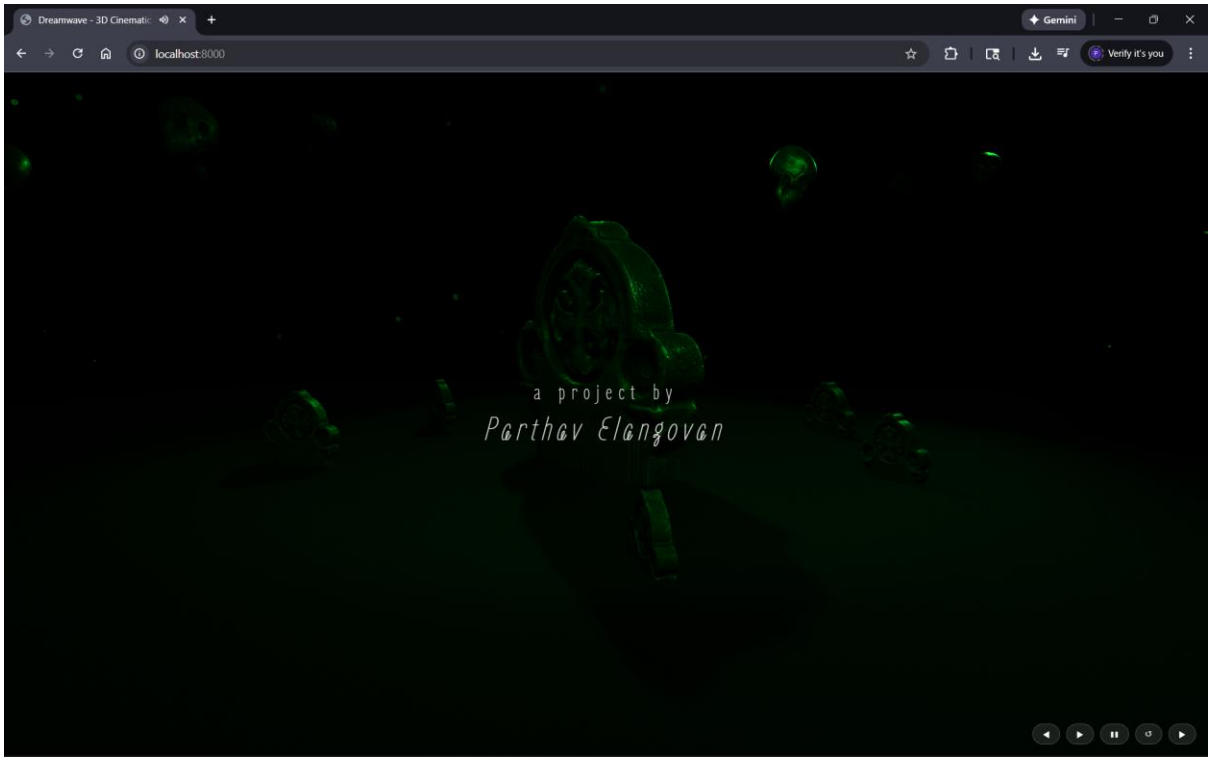

Dreamwave - 3D Cinematic
Gemini
localhost:8000
Verify it's you
music
Adele - Skyfall
RZYAB


Dreamwave - 3D Cinematic
Gemini
localhost:8000
Verify it's you
built with
3.js -Javascript 3d library


Dreamwave - 3D Cinematic
Gemini
localhost:8000
Verify it's you
audio production
Audacity


Dreamwave - 3D Cinematic
Gemini
localhost:8000
Verify it's you
rendered with
UnrealBloomPass & FilmPass Shaders


Dreamwave - 3D Cinematic
Gemini
localhost:8000
Verify it's you
powered by
WebGL Post-Processing



3.1 Video Demonstration

A video demonstration of the project is available at: https://youtu.be/uwju0Ecvmlo

3.2 How to Run the Project

To compile and run the project:

- Extract all files to a directory, maintaining the folder structure (scenes/, assets/)
- Start a local web server (e.g., 'python -m http.server 8000' or use VS Code Live Server)
- Navigate to localhost:8000 (or appropriate port) in a web browser
- Click 'START' to begin the experience. Let the scenes play out and if need be the controls at the bottom can be used to go to the next scene or replay the current one.

4. Analysis of Work

The project successfully met and exceeded the original proposal goals:

Original Goal	Achievement
60-90 second 3D sequence	96-second experience with 8 scenes (Exceeded)
James Bond-inspired visuals	Achieved with silhouettes, dramatic lighting, and stylized scenes
Import 3D assets (GLTF)	10+ models imported from Fab.com
Cinematic camera movement	Smooth orbital cameras with easing in all scenes
Post-processing effects (bloom, vignette)	Bloom, chromatic aberration, and film grain implemented
Audio sync (optional)	Background music (Skyfall) with playback controls

Technologies and Tools

- **Three.js:** Core 3D rendering library
- **WebGL:** Hardware-accelerated graphics API
- **GLTF/GLB:** 3D model format for asset import
- **Fab.com:** Source for 3D model assets
- **Audacity:** Audio editing for background music
- **VS Code:** Development environment
- **Google Fonts (Zen Loop):** Typography for UI