**Final Assignment  
Cube Dodge 3D**

Rishi Gandhi

Student ID: 202014908

Department of Computer Science, Faculty of Science

Memorial University of Newfoundland

COMP 4302: 3D Computer Graphics

Oscar Meruvia-Pastor

March 27, 2024

**Option Chosen:**

I decided to implement a 3D version of a basic arcade game with retro-style elements, incorporating personalized features specific to the members of the group.

**Description of Implementation Features and Characteristics:**

1. Three.js Integration: Utilizes the Three.js library for 3D object creation, rendering, and scene management.
2. Player Cube: A controllable cube serves as the player character, with movement along the x and z axes and a distinctive texture.
3. Environmental Interaction: Player cube interacts with ground and enemy cubes, adjusting velocity and applying gravity upon collision.
4. Orbit Controls: Allows easy manipulation of the camera through mouse input for orbiting and zooming.
5. Particle Effects: Particles orbit around the player cube, enhancing visual appeal.
6. Dynamic Enemy Spawning: Enemies spawn dynamically and move towards the player cube.
7. Lighting: Scene includes directional and ambient lighting, with adjustable properties such as visibility, position, color, and intensity.
8. User Interaction: Keyboard events control player movement and trigger actions like toggling light visibility, changing light position, and camera movement.
9. GUI Controls: Utilizes dat.gui library for a graphical interface to adjust light and camera settings.
10. Modular Design: Organized into classes and functions for improved readability and maintainability.
11. Animation Loop: animate() function oversees continuous scene updates and user input.

**User Manual:**

Installation:

To run the code, follow these steps:

1. Clone the Repository:
   1. git clone <https://github.com/iamrishigandhi/COMP-4302-Final-Assignment>
2. Install Dependencies:
   1. Node.js, three.js and npm is required to run the program.
   2. Node.js can be downloaded from here: <https://nodejs.org/en/download>
   3. To download the latest version of npm, on the command line, run the following command:
      1. npm install -g npm
   4. Ensure you have Node.js and npm installed on your system. Then, install the dependencies using:
      1. npm install --save three
      2. npm install --save-dev vite
      3. npm install dat.gui
      4. npx vite

Interacting with the Program:

1. Movement Controls: Use `W`, `A`, `S`, `D` keys to move the player cube.
2. Toggle Light Visibility: Press `I` key to toggle directional light visibility.
3. Adjust Light Properties:
   1. Toggle light visibility using GUI toggle.
   2. Change light color randomly by pressing `C` key.
   3. Adjust light color using GUI sliders.
   4. Move light position with `Z`, `X`, `V`, `B`, `N`, `M` keys or GUI sliders.
4. Camera Navigation:
   1. Use OrbitControls for mouse-based movement.
   2. Keyboard inputs (`F`, `G`, `O`, `J`, `K`, `L`) move the camera in various directions:
   3. `F`: Move left
   4. `G`: Move right
   5. `O`: Move up
   6. `J`: Move down
   7. `K`: Move forward
   8. `L`: Move backward.
5. Texture Usage: Textures are applied to player cube, enemies, particles, and ground.
6. Particle System: Particles move in circular motion around the player cube.
7. Collision Detection: Game stops if player cube collides with an enemy.

**Video Submission:**

The YouTube video demonstrating the implementation can be accessed here.

**References & Sources:**

ChatGPT - <https://chat.openai.com/>

three.js - <https://threejs.org/docs>

dat.gui - <https://github.com/dataarts/dat.gui>

Code Reference - <https://github.com/chriscourses/threejs-game>

Orbit Controls - <https://threejs.org/docs/#examples/en/controls/OrbitControls>

Textures - <https://www.freepik.com/>

I have written all the code by myself and used the following sources above as part of libraries, and to include code for accessing textures and GUI, among other features. I made all the models all by myself, and the textures were open-source images sourced from the link above.

**Special Features for Marker Evaluation:**

1. Particle Effects and Collision Detection:
   1. To test particle effects and collision detection, the marker should observe the behavior of particles orbiting around the player cube.
   2. Intentionally collide the player cube with an enemy cube to ensure the game stops as expected upon collision.
2. Texture Application:
   1. Verify that textures are properly applied to the player cube, enemies, particles, and ground, ensuring visual appeal and coherence.
3. GUI Controls:
   1. Evaluate the effectiveness of GUI controls for adjusting light properties and camera settings.
   2. Ensure GUI sliders and toggles provide intuitive control over relevant parameters.
4. Overall Gameplay and Immersion:
   1. Evaluate the overall gameplay experience, considering the interaction between player cube, enemies, environmental elements, and visual/audio feedback.

**A screenshot of a video game

Description automatically generatedApplication Screenshots:**

**A video game screen shot

Description automatically generatedA screenshot of a video game

Description automatically generated**