Project 1: Shooting Game Report

[Link to the program: 2D Shooting Game]

Name: João Silva Martins

Email: jrd4@hi.is

1. Introduction

This report outlines the implementation of a 2D shooting game using WebGL. The game, titled "2D Shooting Game", challenges players to shoot flying birds using a movable gun at the bottom of the screen. This project demonstrates the application of WebGL for 2D graphics programming and game development.

2. Implementation Features

2.1 Basic Game Mechanics

- **Gun Control**: The player controls a gun at the bottom of the screen, moving it horizontally using the mouse.
- **Shooting Mechanism**: Players can shoot by pressing the 'W' key, firing bullets upwards.
- **Bird Movement**: Birds fly across the screen at varying speeds and heights.
- **Collision Detection**: The game accurately detects collisions between bullets and birds.
- **Scoring System**: Players score points by successfully shooting birds, with the game ending upon reaching 5 points. Once it ends it shows a “Good Job!!” message.

2.2 Advanced Features

- **Multiple Birds**: The game spawns and manages up to 5 birds simultaneously on the screen. The birds initially spawn at a random height between 0.4 and 0.8 and at a random speed between 0.003 and 0.009 canvas width per refresh rate. Once a bird disappears from the canvas it automatically spawns again with a random height and speed.
- **Multiple Shots**: Players can fire up to 5 shots at a time, allowing for more dynamic gameplay. Once the shot hits the target or disappears the frame, the counter decreases and it allows for another shot to be taken.
- **Visual Feedback**: A counter at the top of the screen displays the current score using blue rectangles from 0 (0 birds hit) up to 5 (5 birds hit).

3. Technical Details

3.1 Rendering and Animation

- The game utilizes WebGL for rendering, with separate buffers for the gun, birds, shots, and score counters.
- Animation is achieved using `requestAnimationFrame` for smooth gameplay.
- The game employs vertex and fragment shaders for rendering game objects.

3.2 Game Logic

- Bird spawning is managed dynamically, ensuring a constant presence of up to 5 birds.
- Each bird has individual properties including position, speed, and active status.
- Collision detection is implemented using bounding box calculations for accurate hit registration.
- The game state is continuously updated, managing active/inactive status of birds and shots.

5. Challenges and Solutions

- **Rendering Multiple Objects:** Managing the rendering of multiple animations was tricky. In the beginning some animations were “leaking” to other objects. An example was while rendering the gunshot animation where the gun was shooting itself instead of the actual shot. This was solved by sending the coordinates of each object through the same uniform variable.
- **Having Multiple Counters in the Image:** Having more than one counter in the image while keeping updating for the next counter was not straightforward. The solution involved using a `birdShotCounter` to track the total score and a `counterIndex` to represent the number of counters displayed. When `birdShotCounter` exceeds `counterIndex`, the latter increments, triggering the display of a new counter. Counters are rendered in a loop, with positions calculated using the formula `-0.83 + 0.4 * (i)` for the x-coordinate, ensuring even spacing across the top of the screen.
- **Collision Detection**: Implementing accurate collision detection ended up being a bit more challenging than expected. This was solved by defining clear bounding boxes for birds and shots and checking for overlaps.
- **Game Balance**: Adjusting bird speeds and spawn rates to create a challenging but fair game required iterative testing and refinement.

6. Conclusion

This project successfully implements a 2D shooting game using WebGL, incorporating all basic requirements and advanced features. The game demonstrates effective use of WebGL for 2D graphics, animation, and game logic implementation. Future improvements could include more varied bird types, power-ups, and level progression.