Aim Lab

Kawin Tiyawattanaroj and Mikako Inaba

Abstract

The goal of this project was to create a first person shooter (fps) game to allow players to practice aiming and learn to navigate scenes using commonly used controls. The game tracks precision, accuracy, and time, different measures of performance. The objective of the game is to maximize precision and accuracy while also minimizing time. As in many fps games, players can move through the scene using *WASD* keys and look using the mouse. The targets generate in random locations to allow players to get comfortable with these controls.

Introduction

Goal

Our project, Aim Lab, was to implement an existing game with the same name: aimlab.gg. Specifically, we hoped to implement the core technical features of the game, including rendering, moving, shooting and scoreboards. We hope that our demo gives users the experience of playing the actual game.

Previous Work

The idea of our project was drawn directly from a game called aimlab.gg. aimlab.gg is very popular amongst both professional gamers and regular players as it primarily helps them with aiming targets and transferring shots between multiple targets. The original version of aimlab has multiple levels and multiple scenarios. For example, there are scenarios where targets move around in different directions or players need to direct their characters through obstacles. We were also inspired by a three.js example, which is also an fps shooting game.

Methodology

Rendering the scene and lighting

The way we created our scene and lighting is pretty straightforward. For the scene, we used a combination of PlaneGeometry and BoxGeometry, which we used in the class assignments, to create a plane, walls, and boxes in the scene. For lighting, we used THREE.spotlight and THREE.HemisphereLight. The challenge that we encountered in

this part is to find a good position to place the light and the direction that the light should point to. This youtube video was a great source for learning how to use lighting in THREE.js: https://www.youtube.com/watch?v=T6PhV4Hz0u4.

Player movement

Player movement is controlled by *WASD* (for movement). This is accomplished through event handlers that mark keys as pressed in a JavaScript object. Upon each render loop, if a given movement key is pressed then a force is applied to the character's physical body in the appropriate direction. We also accounted for multiple movement keys being pressed at once (for example, "W" and "A"), and scaled the force accordingly.

In addition, like other fps games, we also created camera.js which gave us the property of first person perspective. Players can move the cursor around to look around the map. Combining this with *WASD* controllers, we achieve a decent controller for our game.

Note that we studied many sources in order to deliver our controllers. Primarily, we studied code from a previous COS426 project Portal 0.5 by Allen Dai and Edward Yang on how to implement *WASD* controllers. Upon talking to Edward, he also suggested that we look at a Threejs example which they used to base the fps controls off of: https://threejs.org/examples/?q=fp#games_fps. We studied a few youtube videos as well (cited below).

Generating targets and collision detection

All the targets in the game were generated using the three.js SphereGeometry as the geometry and the three.js MeshPhongMaterial as the material. Each round consists of five different targets – one big target and four smaller targets. The big target has a radius four times that of the smaller targets. At the beginning of each round, the first of the five targets is generated at a random position within the scene. The first target is always the big target. When this target is shot, the four smaller targets are generated about the center of the original target. Once all the smaller targets are hit, the round ends.

To detect collisions, the three.js Raycaster was used with the normalized coordinates of the mouse and the camera as the origin of the ray. When a collision was detected, the click was accompanied with a sound. The distance between the camera and the intersection calculated by the Raycaster was used in measuring precision.

Tracking statistics

To track the precision, the score was incremented depending on how close the click was to the center of the target. The maximum score per target was set such that the maximum score attainable across all the rounds in the game was 100. If the click was within half the radius of the target, the score was incremented by the maximum score. Otherwise, the player received only half of the maximum score.

To calculate whether the click was within half the radius of the target, some vector algebra was performed. The angle between the vector from the center of the target to the camera and the vector from the center of the target to the intersection (calculated via the Raycaster as previously described) was found. This angle was used to determine how the score should be updated.

The accuracy was measured as the ratio of the number of clicks that hit a target to the number of total clicks made by the player. To help players improve precision and accuracy, the cursor was replaced with a crosshair. As another measure of performance, the total time per game was also tracked. Both the score (our measure of precision) and the accuracy is displayed during game play, and all three measures of performance are displayed at the end of all the rounds.

Results

The main goal of our project was to create a game to help players improve their aim and practice navigating scenes using controls commonly used in fps games. Therefore, we measured success by having our peers play the game. Tracking precision and time and replacing the cursor with a crosshair are examples of features that we added in response to feedback. Additionally, Mikako did not have any experience with fps games and initially had difficulty navigating the scene. However, she became significantly better at moving to find targets throughout the course of building the game.

We also measured success with respect to completion of the MVP and stretch goals we initially proposed. We completed all of our MVP features: generating random targets, detecting collisions, tracking score, and rendering the scene and lighting. We were able to get started on our stretch goals, including tracking accuracy, precision, and time, adding some obstacles, and splitting a big target into smaller targets. We also added other features such as the sound effect when a target is hit and the crosshair to help players aim better as well as accepting player input to choose the number of rounds and replay the game.

Overall, measured by peer feedback and completion of our MVP and stretch goals, we were able to fulfill our initial goal.

Discussion

Overall, we would consider our project a success. First and foremost, we implemented the core functionality of the original game, which includes moving and looking around with *WASD* controller and cursor, shooting targets, and keeping track of scores and statistics. Per our project advisor Henry, he found that our project served all the purposes of the game.

Nonetheless, we believe that our project could be improved much better, if time allowed. One setback of our project is that we primarily focused on the functionality of the game rather than the aesthetics: we agree that both the menu page and the game itself can look nicer. We also didn't have time to create multiple levels for the game. We could've added more features, such as moving targets, enemies shooting at the player, and more complex scenes.

Conclusion

We learned a lot more than we could have thought. Compared to the assignments where we were given directions and all the materials needed for implementation, this project posed many challenges. For example, when we created the scene, we got stuck with a dark scene for a while because we didn't implement the lighting properly. Thus, we weren't able to see anything that we created. This project also taught us the challenge of looking up resources and understanding other people's work. As easy as it sounds, incorporating existing resources into our project was difficult, as we often had completely different settings and implementations.

Contributions

Kawin focused on rendering the scene and lighting and player movement, while Mikako mainly worked on generating the targets and collision detection and tracking statistics.

Works Cited

https://aimlab.gg/

https://threejs.org/examples/?q=fp#games_fps.

https://threejs.org/docs/#api/en/geometries/SphereGeometry

https://threejs.org/docs/#api/en/materials/MeshPhongMaterial

https://threejs.org/docs/#api/en/objects/Mesh

https://threejs.org/docs/#api/en/core/Raycaster

https://threejs.org/docs/#api/en/math/Vector3

https://www.youtube.com/watch?v=oqKzxPMLWxo&t=225s

https://www.youtube.com/watch?v=UuNPHOJ_V5o

https://www.youtube.com/watch?v=T6PhV4Hz0u4

Sound and images:

https://www.youtube.com/watch?v=dSLAziydY8I

https://thenounproject.com/icon/crosshair-simple-283034/

https://www.pngplay.com/image/95109

https://tinyurl.com/59kpk883