Project Report

Project Name : A.C.E - Astro Combat Elite

(HTML 5 Multiplayer 2D Space Arcade Game)

CS 467 Spring 2024

Thomas Evans, Jose Baroza-Martinez, Richard Oluyole

Table of contents

Installation Instructions - How-to-play guide

The best way to run A.C.E is by utilizing Visual Studio Code. The shortcuts used in these instructions are based on the most recent version of Visual Studio Code. To start A.C.E, open the terminal and use CTRL+SHIFT+5 to split the terminal. In the first terminal, ensure that it is in A.C.E's root folder (./ACE-Astro-Combat-Elite) then run the command npm run install-all. This will ensure that all the dependencies are properly installed into the root package.json file and in the backend package.json file. Once the packages have been successfully installed, run the command npm run dev.The game will indicate in the console that it is running on localhost:8080. This indicates that the front end for A.C.E is properly running. Next, in the second terminal run the command npm run backend. A message should appear and will indicate that the logs are being saved into a .txt file(located within the log folder). This means that A.C.E's back is now connected and will allow A.C.E's multiplayer option to function correctly.

In order to play the multiplayer aspect of A.C.E, open up two web browsers and navigate to localhost:8080 in both browsers. Select the multiplayer option in both browsers. Next join one of the available rooms and select the same room number for both browsers.

Good luck and have fun!

Controls

A.C.E utilizes the keyboard and mouse.

Github clone link:

https://github.com/Evanstho/ACE-Astro-Combat-Elite.git

References

Book

[1] P. X. Altadill and R. Davey, *Phaser By Example*, 2024.

[2] J. Glazer and Sanjay Madhav, *Multiplayer game programming: architecting networked games*. New York: Addison-Wesley, 2016.

Website

[1] “Rotating a Sprite Towards a Pointer,” *phaser.discourse.group*, [Online]. Available: https://phaser.discourse.group/t/rotating-a-sprite-towards-the-pointer/921. [Accessed: Apr. 30, 2024].

[2] J. Capellan, “Sprite Angle towards pointer,” [Online]. Available: https://codepen.io/jjcapellan/pen/bzVWWW/. [Accessed: Apr. 30, 2024].

[3] phaser js, “examples/public/src/game objects/graphics/health bars demo.js at master · phaserjs/examples,” GitHub, 2016. [Online]. Available:

[4] https://github.com/phaserjs/examples/blob/master/public/src/game%20objects/graphics/health%20bars%20demo.js [Accessed Apr. 25, 2024].

[5] G. Gambetta, “Fast paced Multiplayer,” [Online]. Available: ”https://www.gabrielgambetta.com/client-server-game-architecture.html. [Accessed: Jun. 5, 2024]

Introduction

Astro Combat Elite is an HTML5 multiplayer 2D space arcade game where players can engage in decisive 1v1 spaceship battles or try to survive in a perilous asteroid field as long as possible. The game offers an immersive experience featuring fast-paced combat, various ship models, and precise controls using mouse and keyboard controls. The game leverages modern web technologies and a robust server-client architecture to ensure a fair, synchronized, and fluid gameplay experience with an authoritative server model and client-side prediction.

User Perspective

From the user's perspective, users are greeted by a simple, attention grabbing main menu where A.C.E. provides the option for controls, leaderboard and two main game modes: the multiplayer one-versus-one mode and the single-player Asteroid Shootout mode. In the multiplayer mode, players engage each other in an intense space battle starting on opposite sides of the arena. The goal of this game mode is for one player to eliminate the other player. The single-player mode offers a challenging and engaging experience where players must destroy dynamically spawning asteroids while trying to survive the ever-increasing difficulty level.

Development

Our development process generally followed the initial project plan, and the overall feel of A.C.E remained as we envisioned. We envisioned an 8-bit space shooter that was dynamic, quick and easy to learn. The player's controls are multidirectional and allow for both keyboard and mouse inputs. Players have the ability to pick up power-ups and execute maneuvers(strafing). Despite these similarities to the original plan, there were some deviations.

Initially, we planned to have the users be greeted by a dynamically scrolling leaderboard that showed the top players in A.C.E. We planned to implement the scrolling leaderboard feature using a MySQL database. However, due to time constraints and configuration issues we were limited to only building the framework for the leaderboard. The main menu was still developed as originally planned and kept simple with straightforward options where the leaderboard was one of the selectable options.

Another deviation from the plan was that we wanted users to be able to log into A.C.E. The plan was to get A.C.E loaded onto Google App Engine for deployment where we could then implement a login system through the google cloud services, but Google App Engine remained non-functional due to configuration issues with the app.yaml file. Instead, we focused on refining the core game mechanics leveraging the Phaser.js Physics.Arcade game framework and ensuring smooth client-server communication using Socket.io and Express.

The main game portion of A.C.E compares well to what we initially envisioned despite some minor differences. A.C.E includes both player vs player and player vs environment gameplay which compares to what we initially planned for. The only difference was that both of these aspects were planned to be included into a single gamemode, however, these two types of gameplay were split into two separate modes in the actual implementation of the game.

The next deviation from our initial plan was that we had planned to have 5 players per multiplayer server and an objective-based style game that included AI-controlled enemies. We deviated from this plan to implement two separate game modes that seemed to more appropriately fit the timeline in which we were given. The two game modes are directed towards two separate audiences, which gives both casual and hardcore players the opportunity to enjoy the game. This compares to our initial goal of wanting to make a game that appeals to a large audience of players and provides depth.

A.C.E was built from the ground up with the idea that we wanted to create a fun unique take on a space combat game that was quick and easy to pick and play but was also appealing to a large audience of people.

Technologies

Phaser.js was the backbone and centerpiece to the A.C.E project. Phaser.js enabled the creation of various scenes and gave access to various built-in functions to simplify the development process. In both game modes, Phaser.js facilitated the rendering and animation of object sprites and all visual/audio assets. In the single player mode, Phaser's Physics.Arcade system was additionally utilized for collision detection, defining world boundaries, and implementation of the fluid movement physics.

Vite is a build tool and development server used for all frontend testing with the A.C.E project.

Socket.io and its implementation of websockets were used for bi-directional communication between the client and the server during a multiplayer match. The ability to declare socket event handlers with string names that match on both sides of the socket greatly helped to organize the code for multiplayer. Express.js was used as the game server from which running multiplayer games were managed over a websocket connection.

Conclusion

With Astro Combat Elite(A.C.E), we have achieved our primary goal of creating an engaging and competitive space arcade game. While the gameplay mechanics are not as dynamic as we had originally planned, we still believe that A.C.E. demonstrates our ability to develop a complex real-time multiplayer game using modern web technologies. The game showcases a low-latency, fast paced space shooter where players get the opportunity to either compete against other players or the environment.