COLLEGE OF ENGINEERING

Electrical Engineering and Computer Science

A.C.E: ASTRO COMBAT ELITE

Created by Jose Baroza-Martinez, Thomas Evans, Richard Oluyole

Step into a world of dynamic space combat where you command a ship, destroy asteroids and engage in epic space battles that'll have you on the edge of your seat with jaw-dropping graphics and intense multiplayer action. A.C.E is more than just a game. It is a ticket the ultimate cosmic showdown.

MULTIPLAYER: 1V1

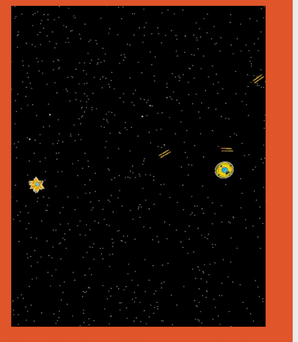
Players step into a high-stakes intense 1v1 showdown, spawning on opposite sides of an arena. Their mission is to destroy the other player for victory. This game mode emphasizes fast-paced, strategic combat where every decision can mean the difference between winning or losing.

FEATURES

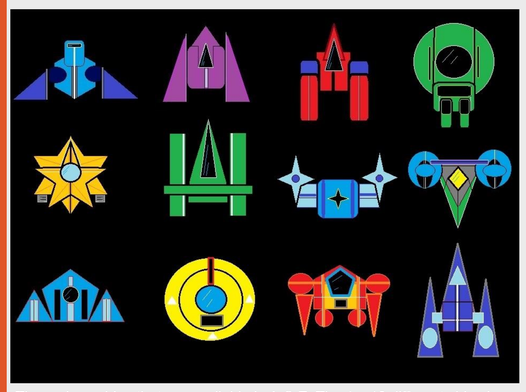
- Authoritative server model, in which the client sends input to the server and the server sends the information back to all the clients ensures fairness.
- Client-side prediction allows for the client to render inputs immediately, while the server processes the inputs and sends back the game state to all the clients which helps to eliminates a visible delay between player actions even with server intervention.
- Entity Interpolation assists with smoothing out the movement of sprites and combat actions.
- Socket.io and Express was used to verify and synchronize game states between multiple clients and the server to allow for fair gameplay.

```
socket.on("input_packet", (inputs) => {
  const roomId = findRoom(socket);
  const thisRoom = gameRooms[roomId];


  if (!thisRoom){
    console.log("Inputs sent to a non-existent/deleted room no inpu
    socket.disconnect();
    return ;
  }
  const player_uuid = thisRoom.id_map[socket.id]
  const player_ship = thisRoom.players[player_uuid];

  if (!player_ship.alive === true){
    console.log("Inputs can not be executed/validated, player is al
  }



  const {W, A, S, D, timestamp, rotation, shooting } = inputs;
  const velocity = 200;
  const strafeVelocity = 100;
  const delta_seconds = 1/60;
```

Example of server receiving player inputs.



Example of a multiplayer space battle



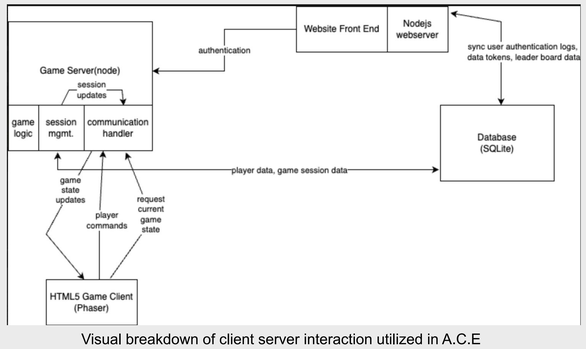
The various ships that are available in A.C.E. They're 12 different ships in total that were designed using inspiration from various sources.

DESCRIPTION

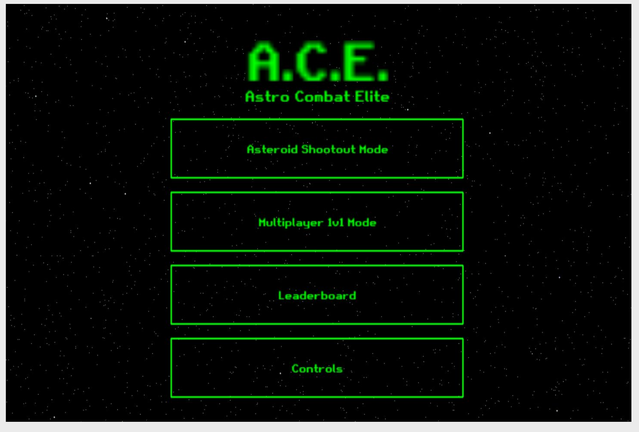
A.C.E is a fast-paced space combat game that offers single and multiplayer game modes. Players have the option to choose from 12 uniquely designed ship models. In ACE, each player starts each match with a fully armed spaceship, ready to unleash a barrage of lasers. Players are able to control their ships utilizing their mouse and keyboard allowing for precise maneuvering and enhanced targeting to improve the overall gameplay experience.

CLIENT-SERVER INTERACTION

The architecture for the game is broken down into client, server and database.



Visual breakdown of client server interaction utilized in A.C.E



A.C.E's home screen with main menu

On the left are some examples of various animations implemented into A.C.E. Phaser utilizes spritesheets.

IMPLEMENTATION

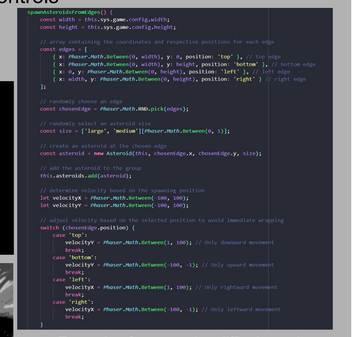
- PhaserJS served as the game engine for developing ACE and bringing the game to life
- The game backend was written in Node.js
- Socket.io and Express were utilized to allow for bidirectional communication between the client and server.
- The game has two separate built-from-the-ground-up custom made game modes.
- The game features custom artwork and sounds with the ship designs drawing inspiration from Star Trek, Star Wars, Shovel knight and other various games and movies.

SINGLEPLAYER: ASTEROID COMBAT

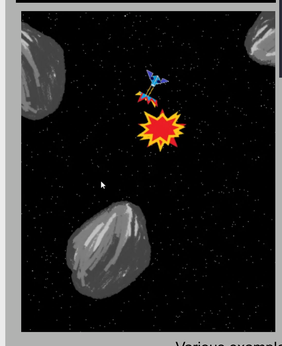
Players find themselves thrust into the heart of a perilous asteroid belt, and their mission is to eliminate as many asteroids as possible while dodging asteroids to keep their ship alive.

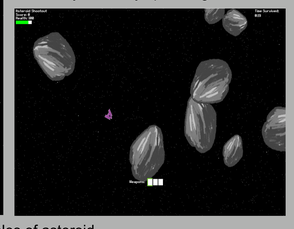
FEATURES

- Progressive difficulty and dynamically spawning asteroids.
- Timed and scored matches to create a competitive atmosphere.
- Responsive dynamic controls

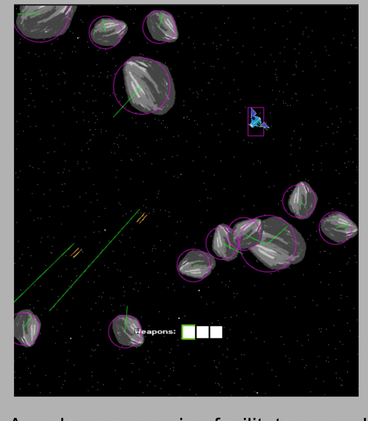


Example of progressive difficulty and dynamically spawning asteroids





Various examples of asteroid shootout gameplay.



The Phaser Arcade game engine facilitates complex physics for movement and collisions, and provides a robust debugging capacity as depicted in the screenshot above.

Oregon State University