Starpilot

Final Project Proposal

Group Members
Andre Haas

Description
In this PC game powered by WebGL and Three.js, the player pilots their spacecraft into battles against AI-controlled adversaries. Both the player and the adversaries will be able to pilot different spacecraft in the game levels. Below are the core mechanics for this game idea.

Intuitive Controls with Six Degrees of Freedom

The player will be able to precisely manipulate the ship's movement:

- Roll, pitch, and yaw input, affected by moment of inertia
- Thrust and reverse thrust input, affected by mass
- Positional and angular velocities will decay back to zero, to help make the controls more intuitive
- Particle trails from the ship engines will help give the player a sense of speed

Spacecraft Variety

The game will feature a variety of spacecraft, custom-modeled in Blender. Each spacecraft will have different handling and combat characteristics:

- Fighter ship with small mass and agile movement. Armed with a forward-facing rapid-fire blaster.
- Medium-sized bomber equipped with homing missiles.
- Capital ship with very large mass. Equipped with mouse-aimed heavy cannons.

Object-Rich Environment

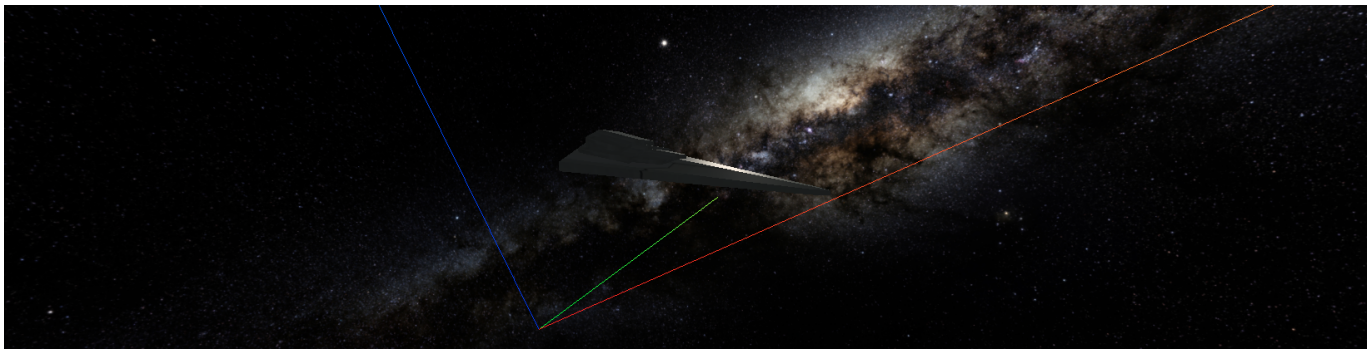
The environment will contain various stationary objects, to help the player have a frame of reference:

- Asteroids
- Space stations
- Planets

AI-Controlled Adversaries

The adversaries will fly spacecraft with the same capabilities as the player's spacecraft. To create challenging scenarios the AI will utilize the following techniques:

- Finite-state machine to determine when the ship will attack, flee, or perform other actions
- Proximity detection to avoid collisions
- Anticipation of the future position of other ships, so that it can lead its attacks



Early development screenshot of a spaceship next to a set of axes.

Technical Aspects

Since Three.js is simply a lightweight 3D graphics library that uses WebGL, the implementation of this game will require many low-level components. Below are some examples:

Graphics

Various details will require the implementation of graphical effects beyond simple 3D models:

- Particles for ship engine trails and weapons
- Beams and projectiles for weapons

Physics System

The physics system will need to track the positional and angular momentum of the spacecrafts, and manipulate those values based on the pilot's inputs. Additionally, the physics system will need to detect collisions between all objects.

Combat System

A module will need to create and track all of the projectiles and do collision detection for them. Some of the weapons may be homing-missiles, which require additional steering logic. The combat system will also need to integrate with the controls to allow for mouse-aimed weapons.

Course Material Integration

- **Particle systems** for weapons and ship effects
- **Physics** of the spacecrafts
- **Simulation** of adversarial pilots
- **Rigid bodies** of spacecrafts modeled in Blender