COP 290 - Assignment 3
Battlestar Galactica

Akshay Kumar Gupta
2013CS50275

Haroun Habeeb
2013CS10225

Barun Patra
2013CS10773

J. Shikhar Murty
2013EE10462

Abstract

The game is a real-time racing cum FPS game. It takes place in space. Each player owns a spaceship and the objective is to be the first to reach the end. Players can also shoot at other players and try and destroy their ships. Each spaceship has different stats like accuracy, speed, ammo etc. The game is designed for a maximum of four human and four AI players.

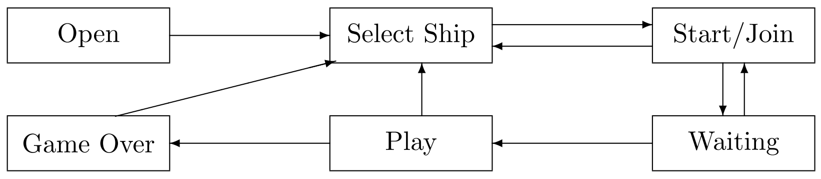

Figure 1: Game Loop

1 Overall Design

2 Network

We will use UDP to handle network connections. It is a completely connected peer-to-peer network.

2.1 Start of game

One user starts the game and the P2P network. This user specifies how many people will be joining the game. A new user can connect to the network by sending his IP to any existing player in the network. As UDP is being used there is no notion of an actual ‘connection’, all each user needs is the IP address of each other player. The existing player will give the IP of the new user to all other users and the IP of all other users to the new user.

2.2 Message Passing

- A user broadcasts his state to all other users at regular time intervals.
- We have a separate thread for listening to incoming messages.
- When a new message is received from another user it is passed on to the rendering thread which renders that user according to his state.
- In case another user sends us a message which drops, the new position of the user is simply extrapolated from his old position, velocity, and acceleration.

2.3 Handling Drops

If we have not received a message from a user for the last T seconds, then we assume that the user has dropped. T is large enough compared to the average time interval between messages for the probability to be very low that the user has not dropped.

3 Graphics

3.1 Model Rendering

- Blender will be used to create models of spaceships, asteroids, debris etc.
- We will load the object files for these models using our own code.
- We will also have our display function for each model.
- However, the bodies actually have bounding boxes. These bounding boxes are what the physics engine will handle.

3.2 User Interface / HUD

We will offer a certain degree of customizability of the UI to the user.

- Select Ship : On this screen, the user sees what ships he can select, we may modify some properties of the ship - such as the droid. he can alter the mouse sensitivity and the keyboard mappings. There'll also be a *Play* button that takes the user to the Start/Join screen.
- Start/Join : This screen allows the user to either

    1. Start: He can start the game with some options, such as the number of players and the number of AI.
    2. Join : He can join a game by providing the IP of any member of a network.

- Waiting : This is the screen shown before the match actually begins. On this screen, the user waits while more people join the match.

- Play : This is what is displayed while the player actually plays the game. The world is rendered with a HUD. The HUD displays properties such as health, ammo.

All of these screens can be implemented using openGL/freeglut/glui/myGUI.

3.3 Camera Positions

According to the ship, various camera positions will be available. For example, the TIE fighter might have only two views - first person and the third person. The X-Wing will have an additional Droid-view. The user can toggle between these views during the game. This can be handled by simply moving the camera of that user.

4 Music

Music will be incorporated in our project using openAL. Every client will have a seperate thread to play the music.

5 Physics and Collision Detection

5.1 Collision Detection and Handling

- Players' ships can collide with other ships and floating space debris. Our code needs to be able to detect and handle such collisions. We will use bounding boxes for detection of collisions.

- Once a collision has been detected, using the point of contact and vectors along collision, we can resolve the equations and assign the bodies their respective velocities.

5.2 Hit Detection

- When a user shoots his weapons, we need to check if the projectile hit any of the players. This again, will be done by using bounding boxes for each ship.

- If a hit has happened, both parties involved will get a visual cue.

6 AI

7 Testing

8 Planned Improvements