Online Gaming Technologies

Project 1

Cliona Hayden

C00225401

Contents

Game loop

Setup winsock

Setup game entities

Setup game states

Send movement data

Send colour data

Player movement

Collisions

Initiate game over

Screen wrap

# Features

## Set up Game loop

**Description:** The process of setting up the base project and implementing a basic game loop incorporating SFML and Winsock.

**Conditions of satisfaction:**

1. Set up the project file so it includes SFML, Winsock and any other necessary libraries.
2. Implement a basic game loop into the project.

## Setup Winsock

**Description:** Setup a basic Winsock architecture in the project, including relevant ports, connections etc.

**Conditions of satisfaction:**

1. Implement basic Winsock architecture into the host and client projects, each incorporating any data about connections, ports etc.

## Setup Game Entities

**Description:** Set up three players in the project, one being the player and the other two being the opposing players. Each entity will be displayed as a randomly coloured circle. The player will have the ability to move using the arrow keys and their updated data will be sent to the server to be distributed among the rest of the players.

**Conditions of satisfaction:**

1. The player and enemy entities will be set up with a circle graphic, position, radius, and colour.
2. Only the player can move using the arrow keys.
3. The player’s data will be sent to the server and updated.

## Setup Game States

**Description:** Set up game states for different phases of the game, for example: initialisation, set-up, gameplay, end game. These game states will be updated as needed and used for the sending of different packet types.

**Conditions of satisfaction:**

1. Set up an enum class which includes states for any relevant game states.
2. Use these states to update the game and send packets accordingly.

## Send Movement Data

**Description:** Send any entity positions and radii to the server as needed. Then use this data to update the relevant positions of each entity across all servers and clients.

**Conditions of satisfaction:**

1. Make a packet type and struct to contain data to send.
2. Send the data to the server and process it accordingly.
3. Update server entity positions.
4. Send the data from the server to all other clients.
5. Update client entity positions.

## Send Colour Data

**Description:** Send any entity colours server as needed. Then use this data to update the relevant colours of each entity across all servers and clients.

**Conditions of satisfaction:**

1. Make a packet type and struct to contain data to send.
2. Send the data to the server and process it accordingly.
3. Update server entity colours.
4. Send the data from the server to all other clients.
5. Update client entity colours.

## Player Movement

**Description:** The player can move their entity using the arrow keys. Update the player entity’s position based on these inputs.

**Conditions of satisfaction:**

1. The game received key input.
2. The player’s position is updated based on input.

## Collisions

**Description:** When either of the other entities collide with the blue entity the game ends. All collisions are done in the server based on the positions the clients passed to it. If a collision is detected, set the game to over and relay this information to all the clients on the server.

**Conditions of satisfaction:**

1. Check circle to circle collision with the blue entity and all other players.
2. If a collision is detected end the game.
3. Send a packet to all clients stating that the game has ended.

## Initiate Game Over

**Description:** If a game over is detected, switch to a game over screen.

**Conditions of satisfaction:**

1. If a client receives a packed stating that the server has detected collision, switch to a game over screen and end the game.
2. If a server detects a game over, switch to a game over screen and send a packet stating this to all clients connected to the server.

## Screen Wrap

**Description:** If a game entity reaches the edge of the window, they wrap around to the other side of the window.

**Conditions of satisfaction:**

1. If a game entity’s position goes out of the bounds of the screen, move their position to the opposite side of the window.
2. Update their positions on the other clients and server accordingly.