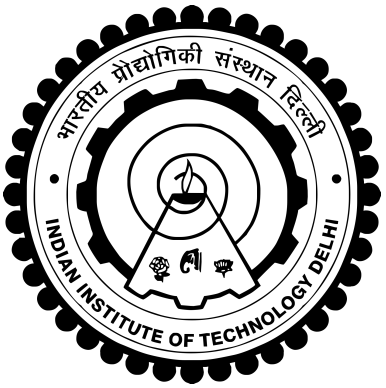

Indian Institute of Technology Delhi

COP290: Assignment 3

Multiplayer Ping-Pong Game

Karan Singh : 2013EEE10465

Shantanu Kumar : 2013EE10798

Surag Nair : 2013EE10504

April 7, 2016

Contents

1 Objective

The objective of this assignment is to design a Ping Pong Multiplayer Game which has the following features:

- Multiplayer on-line game without a central server using completely peer-to-peer networking
- Artificial Intelligence for Computer players with different difficulty levels (easy/medium/hard)
- It is a continuous action game, even after missing the ball it continues to move on the game board
- It will be developed for desktops

The game comprises of a single board with the following basic layout.

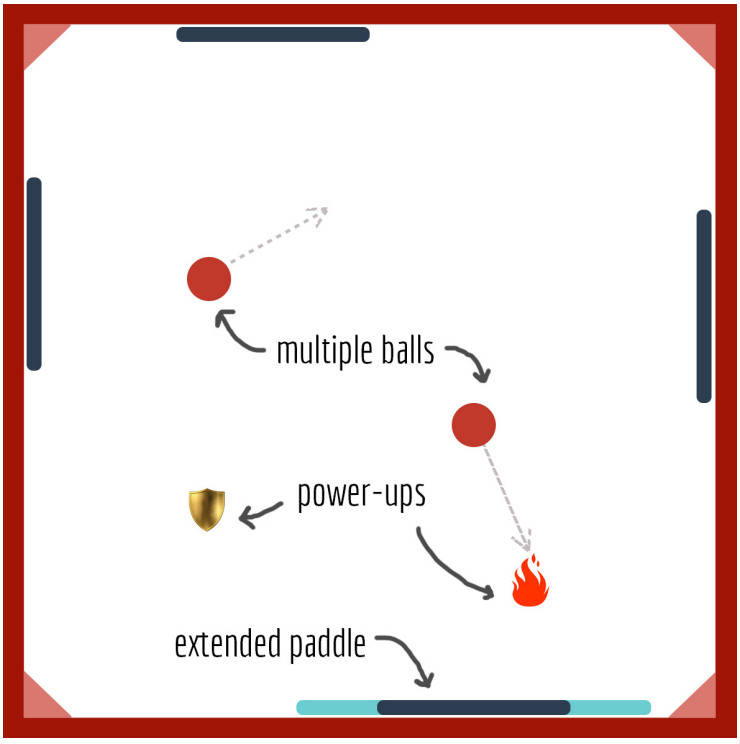

Figure 1: Ping Pong Game Board

2 Game Design Implementation

The Ping-Pong game can have a maximum of 4 players playing at a time. The players will have limited number of lives. Each player has to prevent the ball from striking their wall using a paddle. If the ball strikes their wall, they loose a life. If a player loses all his lives, then he is sent out of the game. The player who stays till the end will be deemed as the winner. The implementation details are:

1. The game will be programmed in Java
2. The GUI will be developed using Java Swing library
3. Socket programming using UDP will be used for communication
4. Different threads will be used to run the network and back-end in parallel

2.1 Game Features

- At the beginning of the game, the host starting the game decides the maximum number of lives for all the players.
- The game can have a maximum of 4 players where each player can be a network player, a computer player or the user themselves.
- The game would contain power ups that randomly pop up on the board during gameplay and can be collected by any player by reflecting the ball into it. These power ups would be the very essence of the game that would make it interesting. The following types of power ups would be available.

  Big Paddle The user who collects this power up gets a bigger paddle uptill the time he collects some other power up or loses a life.

  Extra Life The user who collects this power up gets an extra life added to their existing lives

  Multi Ball If a user collects this power up, the ball splits into 2. Such a power up has an equal impact on all the users.

  Shield This power up enables a shield between the user and his wall for some short time or till the time the user misses a shot and the ball strikes the shield, whichever happens first.

  Fast Ball This ability can prove to be a problem for everyone as it would increase the speed of the ball for some given time.

2.2 Physics of the Game

The exact physics of the game is subject to experimenting and actually choosing what works best, but we have listed the basic ideas that we are going try.

- Different positions of the paddle reflect the balls at different angles. The center of the paddle reflects it straight up whereas as we move closer to the ends, the reflection becomes greater.
- The ball moves at constant speed except for when the Fast Ball power up is taken by some user. We will also experiment with increasing the ball speed with time and see what makes for a better gaming experience.
- In case of multiple balls (due to the Multi Ball power up), different balls can collide with each other on the board and get reflected at the correct angles according to the laws of physics without losing any velocity.
- Reflections off the (literal) corners of the wall for each player need to be handled correctly. We will design the corners to be smooth and declare some part of it as No Man's Land, so that no one loses a life when the ball strikes it.
- Other corner cases include collisions of the ball with the sides of the paddle, especially after striking the same paddle's wall. This is a special case of collision since the ball will interact with both the user's ball and paddle. It will be handled to send the ball in the appropriate direction.

3 Networking

We will use User Datagram Protocol (UDP) for game networking. This protocol is used to send and receive packets directly. It is built on top of IP and we can send a packet to a destination IP address and port which will get passed from computer to computer until it arrives at the destination computer or is list along the way. On receiver side, we listen on a specific port and when the packet arrives from any computer, we get notified of the address and the port of the computer which sent the packet, the packet size and can read the packet data. It is unreliable since some of the packets are lost but it doesn't have much effect on the game since the game state changes rapidly and resending the lost packet will be irrelevant as it would contain the old information.

3.1 Basic Network Layout

Our network will have the following integral features:

- Each player will have two basic threads
- One thread to receive data from the other players
- Other thread to send the data to other players.

UDP is a connectionless protocol so each time we send a packet we would specify the destination address. A single UDP socket can be used to send packets to any number of different IP addresses. At the receiving end the packets sent to a particular socket IP address and port are placed in a queue. Since UDP is connectionless, packets may arrive from any number of different computers. Each time we receive a packet we retrieve information about the IP address and port of the sender.

3.2 Packet Details

Each data packet will contain a time stamp, the operation type/code, and the operation content varying with the operation type. The messages exchanged in the game will be described as objects, all containing a mandatory property called operation code. The operation code is used to identify the message type and indicate the properties of the message object. Since the game is dynamic and the states change rapidly large number of packets will be shared every second for smooth running of the game. The sent packets will be received by all the other players and the game state will get updated as the time proceeds. We will be sending around 30 messages every second and hence the use of UDP is justified. The structure of the packets would be as follows,

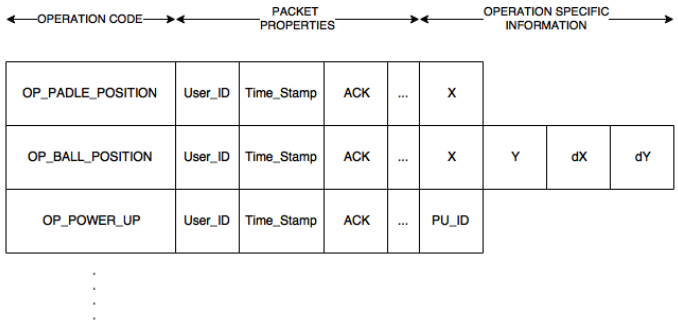

Figure 2: Data Packet Structure

The time-stamp property of the packets also works as the packet ID. It will be used to evaluate if the received packet from a user is the latest one they sent. The ACK property of the data packet gives the ID of the last packet received by the Sender that was sent by the Receiver, as an acknowledgment tag.

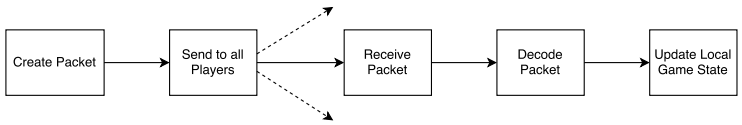

Figure 3: Event Flow of Data Packet

3.3 Game State Consistency

If the game exclusively moves entities based on network updates, any lost or delayed message will cause the entity to "teleport" from one point to another. In order to ensure that same game state is seen by all the players and all player movements are smooth for everyone, we would try to employ the following techniques,

Ball Position The ball is simulated independently on each network machine, because its movement is dictacted by the same rules everywhere, except in the case of collisions with a paddle or a wall. Whenever the ball collides with a player's paddle or wall, the new heading of the ball is best dictated by his simulation. Thus in such an event, the player (with whose wall or paddle the ball collides) sends the new position and direction of the ball to all other users so as to make sure all simulations are synced.

Interpolation Using interpolation, the movement of all entities is locally interpolated from one point to the other, both received through network updates, so that the the entity moves smoothly between these points

Extrapolation By extrapolation, we can try to predict the expected state of the entity assuming that it would not deviate a lot from its current heading and velocity. We can follow this expected movement till a new network update arrives, resulting in a smooth movement of the entities.

3.4 Disconnection Detection & Handling

- If the message from a player is not received for the last T seconds, then we would assume that the user has been dropped from the game.
- Time T would be set to a large enough value, compared to the average time interval between the messages.
- In the event that a player is dropped from the game, then the host shall start an AI player to take his place in the game, so that the game keeps running smoothly for everyone else.
- We would setup a player order list in the beggining of the game with the host as the first player. In case that the host is dropped from the game, the next player in the list shall become the host and start handling all the AI players in the game from their machine.

4 Interaction between Components

A game state is maintained on each device. The game state is constantly updated by network and user updates. This update is independent of the type of input- user input or network input. The networking is completely independent of the GUI, which will run on a separate thread and keep refreshing itself based on the game state.

5 Computer Player

We will incorporate two types of AIs- rule based and a trained AI:

- **Rule-based AI**: We will write a few rules (e.g. following the x coordinate of a ball) and the difficulty can be varied by tweaking the rules (e.g. maximum speed with which the AI can follow the ball will be small for an easy AI).
- **Trained AI**: We will use Reinforcement Learning to train this AI. We hope that it will help capture strategies beyond simple rules, especially in the cases of multiple balls and power-ups. The Markov Decision Process will take the state, available actions and a reward function. The reward function will be straightforward- a penalty for missing the ball and a reward for making the next player miss the ball, and no reward otherwise. The difficulty level will be proportional to the time for which it is trained.

All the Computer players will run on the host machine. As mentioned earlier, in case the host is dropped from the game, the next player is made the host and the Computer players will be initiated on their machine. Since the AI player does not depend on the histrory of the gameplay and only on the current state, we can initialise it on the new machine in the middle of the game in the event of host drop.

All the Computer players will run on separate threads on the host machine. Since the game is completely decoupled from the multiplayer aspect, all players, be it the Current User or the AI players or the Network players, will be handled in the same way, simply as being able to provide some input to alter the current state of the game. The Computer players will look at the current state of the game and produce action commands for moving their respective paddles. At the same time, they will also send their positions as data packets to the Network players, if any.

6 Testing of Components

6.1 Graphics

- We will implement the various components of the game (board, paddle(s), ball(s), etc.) incrementally, visually testing their correctness.
- We will check that all mouse or keyboard controls work properly

6.2 Physics

- We will carry out extensive unit testing to check that collision detection with the wall and between the balls and collision handling works correctly
- We will check that calculation of new position and velocity of ball and paddle based on player state is correct

- Extensive testing of the corner cases will be done by deliberately creating the corner test cases

6.3 Networking

- At first we will test the basic data transfer. We will create some test messages and send them from one player and check the percentage of message drops at receiver's end.
- We will test the case of network disconnection and shifting of player into AI mode when it happens.

6.4 AI

- We will check the difficulty levels of the AI players against the real players.
- This will be through a manual inspection against the different difficulty levels to judge the level and quality of the rule-based and trained AIs.

7 Extra Features

- Extra Power Ups. We are think of extra power ups to implement but they would need to be tested to decide on their feasibility and if they are interesting enough.

  Fire Ball Round This power up would start a 10 second round where the ball becomes a fire ball. Here if the ball touches a user's paddle, their paddle would become smaller by some percentage. When the round is complete all paddles would return to their normal size.

  Sticky Paddle This would give the ability to a user to stick the ball to his paddle when it touches it. The user can then strategically shoot the ball in the direction he wants.

- Sound Effects. We plan to add sound effects to our game later in order to make the experience more engaging.

– * * * –