COP 290 Assignment 3

Ping Pong Game

Aditi
2014CS10205

Ayush Bhardwaj
2014CS10091

Nikhil Gupta
2014CS50462

April 2016

Contents

1 Objectives

Design a desktop app which is:

- Simulation of Multi player and Multilevel Ping Pong game.
- Allows multiple players to play at one time using p2p networking.
- Every client controls its own game state on the basis of information transfer.

2 Overall Design

The game involves each player having a rectangular paddle, using which he has to prevent the balls from colliding with the walls of the walls. Each time the paddle misses a ball, the player loses one life. The player has a limited number of lives and has to score the maximum in them.The game would be controlled via a runnable function which is called again and again till the game ends.All the events of the game would be handled by this function. According to the requirements of the assignment:

- Each peer back-end will be programmed in Java using sockets.
- Doxygen will be used to create HTML documentation of the entire code base.
- The entire code will be split up in multiple files to ensure modularity in code.
- Swing library will be used to display the Graphic User Interface.

3 Graphical User Interface

3.1 Start Screen

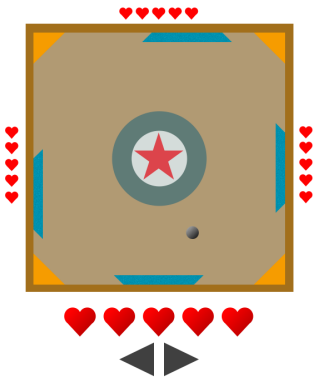
The Start Screen will handle the setting up of the game. It will offer options like Play Against Computer, Play with peers, Number of Players,No of balls,Level of Difficulty. After this screen, another screen to establish connection with other players will appear.

3.2 Main UI

The main user interface displays the board as a JFrame object along with the balls and the paddles at any instant of time. Balls will be modeled as 2-dimensional circles.
The **Swing library** [2] of Java language will be used to render the interface. At every instant of time, data from the object of class Board will be extracted, to get information about all the balls and the paddles to be displayed on screen.

The colors for the balls are randomly chosen, but the radius for all of them is constant for one particular game.Some of the features of UI are:

- Rectangular Board will have **triangular corners**. Balls will be moving on it continuously. Paddles will move according to the player's moves.
- Every user will see his own paddle on the bottom edge of the board.
- Number of Lives remaining will be displayed for each player in the form of hearts.
- **Power Ups(or Downs)** will be random floating objects on the board. They will be randomly generated and will move in random directions.
- There will be suitable functions to render the change in paddle on collection of a Power Up or Down.
- The user controls his paddle position by using the **keyboard** Left/Right keys or by using the **Left/Right buttons** on screen or by dragging the paddle on the window using the **Mouse pointer**.



(a) Splash Screen

4 Sub Components

The following classes would be needed to design an efficient back end.

4.1 Design Of The Ball

Ball class consist of all the information about the balls on board i.e. its velocity, position, radius and color.

Listing 1: Class Parameters for Ball

```
1 class Ball
```

```
{
    private:
        double radius;                  //Radius of the ball
        double center_x;                //X coordinate of the center of ball
        double center_y;                //Y coordinate of the center of ball
        double velocity_x;              //X component of the velocity of ball
        double velocity_y;              //Y component of the velocity of ball
        ArrayList<float> color;         //Color in RGB format of ball
};
```

Functions will be made for accessing all the data parameters of the ball and updating them as desired. Functions for obtaining a random ball based on the desired range will also be created.

4.2 Design of Paddles

The Paddle class would contain all the information regarding one paddle. Basic functions will be there to retrieve all the data related to one paddle. All paddles would have the same color.

Listing 2: Class Parameters for Paddle

```
class Paddle
{
    private:
        double length;                  //Length of paddle
        ArrayList<float> color;         //Color in RGB format of paddle
        double paddle_x;                // x coordinate of paddle center
        double paddle_y;                // y coordinate of paddle center
        int times_ball_missed;          // no of times this paddle has missed ball
        bool isAlive;                   // denotes if player is still alive
};
```

4.3 Design Of The Board

The background screen would be implemented using a Board class. It would consist of all the information about the background i.e. its dimensions, color, number of balls present and a vector containing the information of the balls and similarly for all the alive paddles.

Listing 3: Class Parameters for Board

```
class Board
{
    private:
        float dimension_x;                  //x dimension of board
        float dimension_y;                  //y dimension of board
```

```
6           int number_balls;                       //number of balls on board
7           ArrayList<Ball> vector_of_balls;        //vector of balls on board
8           int number_paddles;                     // number of paddles, i.e. players
9           ArrayList<Paddle> vector_of_paddles;    // vector of paddles on board
10          ArrayList<bool> dead_alive; // array for whether player is alive/dead
11  };
```

Functions for accessing and updating the parameters of the board will be created. Apart from this, functions to add or remove particular balls will be provided.

5 Physics

The basic law of physics will hold throughout the game.We will be handling the following events:

- **Motion of the Ball:** Instantaneous velocities of the ball would be maintained in X and Y directions.Next positions would be calculated simply by adding these to the current X and Y positions.
- **Motion of the Paddle:** Movement of paddle can be controlled via Mouse or the Keyboard. In case of Mouse, current position of the mouse will control the movement of paddle. While in case of Keyboard, Arrow Keys will control the paddle movements.
- **Collision of Ball and Paddle:** Collision occurs whenever the distance between the center of ball and paddle boundary is less than the ball's radius.Basic collision equation will be used while also considering the speed of the paddle at the time of collision. A suitable coefficient of restitution would be used.
- **Collision of Ball and Boundary:** The detection of this collision is also similar to the one listed above. Basic collision equation will be used. A suitable coefficient of restitution would be used.
- **Collision between balls:** Basic equations for head on collision between two solid objects would be used to calculate the next velocities of both the balls.
- **Corner Case:** As design of all boundaries(of paddles, board) are of triangular shape, Corner Cases reduce to a simple collision between the ball and straight boundaries. The boundary with which the client first detects the collision is considered for the calculation.
- **Our Equations**
  1. $$V_{y2} = U_{x1}(\frac{\epsilon+1}{2})cos\theta sin\theta + U_{y1}(\frac{\epsilon+1}{2})sin^2\theta + U_{x2}((\frac{1-\epsilon}{2})cos\theta sin\theta - sin\theta cos\theta) + U_{y2}((\frac{1-\epsilon}{2})sin^2\theta + c$$

2.

$$V_{x2} = U_{x1}(\frac{\epsilon+1}{2})cos^2\theta + U_{y1}(\frac{\epsilon+1}{2})sin\theta cos\theta + U_{x2}((\frac{1-\epsilon}{2})cos^2\theta + sin^2\theta) + U_{y2}((\frac{1-\epsilon}{2})sin\theta cos\theta - cos\theta$$

3.

$$V_{y1} = U_{x1}((\frac{1-\epsilon}{2})cos\theta sin\theta - cos\theta sin\theta) + U_{y1}((\frac{1-\epsilon}{2})sin^2\theta + cos^2\theta) + U_{x2}(\frac{\epsilon+1}{2})cos\theta sin\theta + //U_{y2}(\frac{1-}{2}$$

4.

$$V_{x1} = U_{x1}((\frac{1-\epsilon}{2})cos^2\theta - sin^2\theta) + U_{y1}((\frac{1-\epsilon}{2})cos\theta sin\theta - cos\theta sin\theta) + U_{x2}(\frac{\epsilon+1}{2})cos^2\theta + U_{y2}(\frac{1-\epsilon}{2})sin$$

6 Computer Player Algorithm

The computer player can have variable speed. But the range of speed varies on the basis of level of difficulty.Algorithm used by computer would be based on the following facts:

- Initially computer player tries to align its paddle center with the center of the closest ball moving towards it.
- When the ball is about to reach a wall, the computer player first calculates according to the assumption of a static paddle and moves.
- After the collision has occurred, the computer again moves accordingly.
- Another event will be choosing between ball catching versus catching power up objects. The decision will basically be based upon the distance of the power up object and the expected distance of the ball from the paddle after collision.

7 p2p Networking

We would use socket programming [1] to implement p2p networking model. Each client has (n - 1) sockets, one for interaction with all the other n-1 players. Each client sends the following information to its peers:

- **Loss of Life:** Boolean value denoting whether the player has missed the ball and lost a life..
- **Current data of paddle:** Position of paddle is sent, other clients adjust the position of the paddle on their board accordingly.
- **Collision with paddle:** Boolean value denoting collision along with the new position and velocity of ball would be sent.
- **Collection of PowerUps** An integer mapped to a specific power up would be sent.Client will render the UI accordingly.

7.1 Event Flow

The steps involved in flow of each network message from Peer 1 to Peer 2 are:

1. Peer 1 initiates the process of sending the message
2. Peer 1 tries to establish the connection with the peer
3. Peer 2 accepts the connection and starts the handler thread
4. Peer 1 sends the message
5. Peer 2 receives the message and send backs the acknowledgment back to peer 1 and closes the connection.

The flow is similar for any Sender and Receiver pair.

7.2 Event Handling

1. **Missing the ball:** Player looses a life if the ball misses the paddle and touches the wall.It involves updating of the number of remaining lives of the player.
2. **Disconnection:** If a peer takes more than an specified time to answer, it is disconnected. In case a particular client is disconnected from the network, a pop up would be displayed to the other players, and according to their decision, a computer player will be added or the game would continue with the player removed.
3. **Updating Position:** Every client adjusts the paddle positions of other client, according to the data received. Next position of the ball is calculated individually by the clients or is updated in case of collision with any other peer. All the data is consistent throughout the game for all the players.
4. **Collection of Power Up:**Communicated via the network message and a suitable method is called to update the UI.

8 Thread Interactions

We would need multiple threads [3] to handle different components of the game separately. There would be separate runnable() functions for the following list of threads to be used:

- **Update UI** : This thread continuously updates the positions of the balls and the paddles being displayed to the user, on the basis of the information received from the network. This is done after every fixed interval of time.

- **One Thread Per Socket** : There would be n-1 threads, each controlling communication with one of the other clients. Each thread is responsible for making the connection via sockets using unique ports and sending and receiving information about the states of the paddles over network.

The thread maintaining UI will call the update() function only after receiving the updated information from the other n-1 threads.

9 Testing Of Components

9.1 Single peer testing

- Unit testing will be used to check if the socket connections are transferring information correctly.
- For each peer to peer socket, stress testing will be done via python or bash scripts to verify that the socket connections perform as expected in various situations.

9.2 Computer Player Testing

We will play against the computer algorithm both on single PC as well as in a network of less than four players.

9.3 Overall p2p Testing

We will use the app on our desktops once it is ready to identify and squash any remaining bugs.

10 Extra Features

- Every player gets an option of choosing the orientation of the board visible to him, according to his comfort.
- Special items would be floating around on the board, which on being captured, will provide the player with special powers.
- The special powers include : extra speed of paddle, more length of paddle, slower ball close to the player's paddle and extra points.
- Some danger items would also be floating, which on being captured, would either slow down the paddle, or reduce its length or make the ball faster in the player's region.
- The game would have multiple levels that would differ by increasing ball speed and decreasing paddle length.
- While considering collision, frictional drag will also be considered.

- Corners of the board and the paddles will be triangular.
- Players can play with more than one balls also.

11 Future Endeavors

- The 2D game can be translated into 3D using a 3D rendering software to give an in-depth gaming experience.

12 Source Code

The source code of the project is maintained in the following repository:
https://github.com/aditi741997/COP290_PingPong.git

References

[1] Socket programming in java. http://www.tutorialspoint.com/java/java_networking.htm.

[2] Swing library for ui in java. http://docs.oracle.com/javase/tutorial/uiswing/painting/index.html.

[3] Threads in java. http://math.hws.edu/javanotes/c12/s3.html.