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 ping pong game
- Allows multiple players to play at one time using p2p networking.

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:

1. The server side will be programmed in Java using sockets.
2. Volley will be used to send requests and receive responses.
3. Doxygen will be used to create HTML documentation of the entire code base.
4. The entire code will be split up in multiple files to ensure modularity in code.
5. Swing library will be used to display the Graphic User Interface.

3 User Interface

3.1 Graphical User Interface

The user interface displays the board along with the balls and the paddles at any instant of time. Balls will be modelled as 2-dimensional circles.
The Swing library 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.
There are various floating objects displayed on the screen, for special power ups and downs. They may be of following types:

4 Sub Components

4.1 Design Of The Ball

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

Listing 1: Class Parameters for Ball

```
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 stored.

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.

Listing 3: Class Parameters for Board

```
class Board
{
    private:
        float dimension_x;                      //x dimension of board
        float dimension_y;                      //y dimension of board
        int number_balls;                       //number of balls on board
        ArrayList<Ball> vector_of_balls;   //vector of balls on board
        int number_paddles;                     // number of paddles, i.e. players
        ArrayList<Paddle> vector_of_paddles; // vector of paddles on board
        ArrayList<bool> dead_alive;             // array for whether player is alive
};
```

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.

4.4 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. A suitable coefficient of restitution would be used.
- **Collision of Ball and Boundary:** Basic collision equation will be used. A suitable coefficient of restitution would be used.
- **Collision between balls:** Basic equations for head on collision would be used to calculate the next velocities of the balls.

4.5 Computer Player Algorithm

The computer player can have variable speed. But the range of speed varies on the basis of level of difficulty.

- Initially computer player tries to align its paddle center with the center of the closest ball moving towards the paddle.

- 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.

4.6 p2p Networking

We would use socket programming 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:

- **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.

1. 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 .
2. 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 and is consistent throughout the game.
3.

5 Thread Interactions

We would need multiple threads to handle different components of the game separately. Following is the list of various 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.

6 Testing Of Components

6.1 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.

6.2 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.

6.3 Overall p2p Testing

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

7 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 circular.
- Players can play with more than one balls also.

8 Future Endeavors

9 Source Code

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

References