**Team 5 Project Writeup**

PingPong Game

Maryanne Choeun, Joseph Heitman, Tan Nguyen, Griffin Ryan

**Introduction**

This project is a take on the classic Pong game, but in single-player. In the initial project brainstorm, we had difficulty coming up with a single cohesive idea for a project. We agreed that, whatever project we work on, it should have a GUI. In the end, we decided on something relatively simple: a version of Pong.

Coordinates of the Player object and Ball object are both updated in between frames, and collision detection is used to determine if the Ball object is successfully hit. A significant difference between the classic Pong game is that the player the racket by moving it horizontally across the bottom of the window.

Each time the ball hits the racket - the player will receive one point. The game ends when the player’s racket misses the ball.



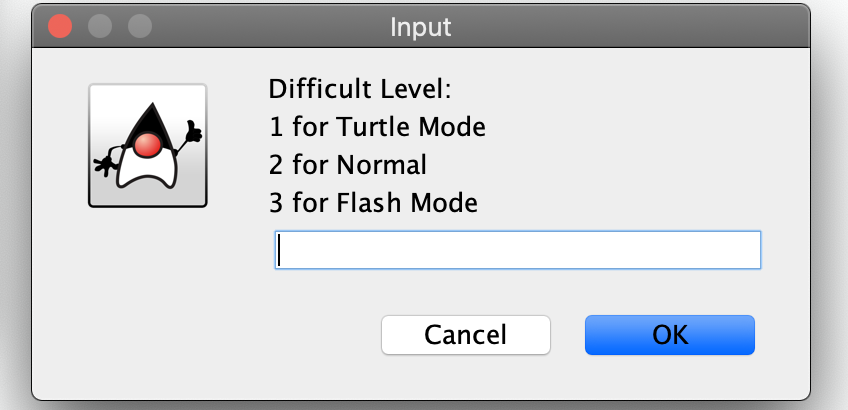
*Figure 1 - PingPong Game Application*

When launching the game, the application asks the user to input a mode to play the game in. There is Turtle Mode, Normal Mode, and Flash Mode. This will set the speed of the game.

There are five classes in total for the project. PingPong.java, PongPanel.java, BouncingBall.java, Player.java and Sprite.java are all used to display the game. The main method can be found in PingPong.java.

**PingPong Class**

The PingPong class is the base class of our project. It is the entryway for the PingPong game application, and includes the main method for executing. Before starting the game, the player is asked to choose a mode to play.



*Figure 2 - Difficulty Option*

As a stylistic touch, our team has chosen to introduce the game by opening a window displaying our team name, shown on this document in Figure 3 and Figure 4. Rather than display it in plain text, the sprites from image.png are used to display the words “Team 5” in a small window.



*Figure 3 - Sprite Welcome Screen*

**PongPanel Class**

The PongPanel class is the master class for the Pong game. This class generates the specific BouncingBall and Player racket used to play the game, and reads the player’s inputs to move the racket. This class also increments the player’s score if it detects a racket hit. If this class detects the ball has hit the bottom of the panel, it gives a “Game Over” message depending on the player’s score.

PongPanel.java is responsible for drawing the game window, updating each frame, and the objects in the game.

The PongPanel class uses collision detection between the Player and BouncingBall objects. The collision detection algorithm calculates the new coordinates for the Player object and the Ball object for each frame.

The collision detection also increments the player’s score if it detects a racket hit. If the player misses the ball, the PongPanel class generates a “Game Over!” window, and the game closes.

**BouncingBall Class**

The BouncingBall class is responsible for the ball object drawn on screen, and controls the speed, size, and color of the ball. When the ball object needs to be drawn to screen again, the red value increases by one. Out of a scale 256, the red value will start to decrease when at maximum value. It is a simple way to create some flashiness for the ball between frames.



*Figure 4 - PingPong Game Application*

While currently constant, our team hopes to enable the speed of this ball to change. We have also tinkered with the idea of changing the color of the ball in a more dynamic fashion than simply blue to pink and back.

**Player Class**

The Player class contains the size and color of the player-controlled racket. This racket can move back and forth horizontally, but not vertically. In future versions, this racket may be more dynamic, with elements such as changing speed or size.

**Sprite Class**

The Sprite class is responsible for loading sprites from a file to draw the opening window. BufferedImage and ImageIO are used to import the image and draw text using sprites displaying “Team 5” on application launch.



*images.png*

**Analysis**

Accomplished:

Thus far, our team has succeeded in creating the playing field, enabling the bouncing ball, ensuring the ball bounces properly when it hits the sides of the playing field or the player’s racket, enabling player control of the racket, and incrementing the score whenever the ball hits the player’s racket. The ball also pulses in color over time.

Intending to accomplish:

To improve, we have considered implementing a bit more complexity to the game in a few ways. Enabling power-ups could change the size or speed of the player’s racket, and perhaps the bouncing ball. We would also like to make the color of the bouncing ball change whenever the ball hits the racket, and by adding a small sound and/or visual effect for each impact. This would make playing our game a bit more dynamic and exciting.