Computer Graphics Project: Ping Pong Game

Instructor: Dr. Hossein Mahvash Mohammadi

Teaching Assistant: Mohammad Javadian

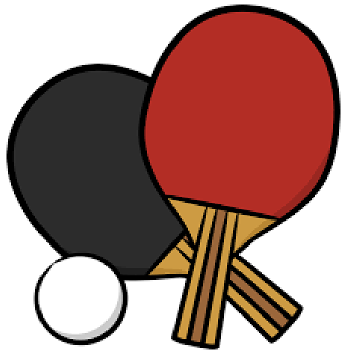
Student: Mohammad Amin Saberi

Student ID: 993623026

University: University of Isfahan

Faculty: Computer Engineering

Date: May 2023



Tables of Contents

Introduction

Ping Pong is a simple yet engaging game where two players use paddles and a ball to score points. The objective is to hit the ball toward the opponent's paddle to earn points by making the opponent miss. Due to its simplicity and entertainment value, it remains one of the most popular games worldwide.

Functions

To implement the Ping Pong game, the OpenGL and GLUT libraries are utilized. The main functions include:

display()

This function is responsible for rendering the current game state. It draws players, the ball, the central line, and the players' scores. Using OpenGL functions such as glBegin and glEnd, simple shapes like rectangles, circles, and lines are rendered. Player paddles are represented as rectangles, and the ball as a circle. The central line is a simple line. Scores are displayed using glutBitmapCharacter.

update()

This function updates the game state for each frame. It calculates the ball's position and checks for collisions with walls and paddles based on the rules of Ping Pong. If the ball hits the top or bottom walls, its vertical velocity reverses. Similarly, when it hits a paddle, its horizontal velocity reverses. If the ball bypasses a paddle, the opposing player earns a point, and the ball resets to the center.

keyboard()

Handles user inputs during the game. This function processes key presses for moving the paddles up and down. Player paddles move with arrow keys, where the up arrow moves the paddle up and the down arrow moves it down.

main()

The program's entry point. This function initializes the GLUT window, sets display modes, configures the window size and title, and registers the display, update, and keyboard functions. Finally, it enters the game loop using glutMainLoop.

Additional Helper Functions:

- **drawRect()**: Draws a rectangle based on the starting position, width, and height.
- **drawCircle()**: Draws a circle using its center and radius.

- **drawScore()**: Renders the players' scores by converting them into strings and displaying them using glutBitmapCharacter.
- **resetGame()**: Resets the game by setting players' scores to zero and reinitializing the ball and paddle positions.
- **updateBall()**: Updates the ball's position based on its velocity and collisions. If a player scores, their score increments, and the game state may transition to "game over" if a winning score is reached.
- **mouse()**: Detects mouse clicks on buttons like "Exit" and "Help" to end the game or display instructions.

The game loop continuously calls the display and update functions, processes keyboard inputs, and persists until the user exits the game.

Variables

The game uses several variables to manage its state:

1. **player1_pos, player2_pos**: Track the positions of Player 1 and Player 2's paddles.
2. **ball_pos, ball_vel**: Track the ball's position and velocity.
3. **score1, score2**: Record scores for Player 1 and Player 2.
4. **WINDOW_WIDTH, WINDOW_HEIGHT**: Define the game window's dimensions.
5. **player1_up, player1_down**: Detect keyboard inputs for Player 1 (up and down movements).
6. **player2_up, player2_down**: Detect keyboard inputs for Player 2.
7. **BALL_RADIUS**: Defines the ball's radius for collision detection.
8. **PAD_WIDTH, PAD_HEIGHT**: Define paddle dimensions.
9. **HALF_PAD_WIDTH, HALF_PAD_HEIGHT**: Store half the width and height of paddles for collision calculations.
10. **LEFT, RIGHT**: x-coordinates of the left and right walls for collision detection.
11. **TOP, BOTTOM**: y-coordinates of the top and bottom walls for collision detection.

Algorithms

The game leverages simple algorithms to handle core functionalities:

- **Ball Movement and Collision Detection**: The ball's velocity is adjusted based on collisions with walls and paddles. For example, a collision with the top wall reverses the vertical velocity, while a collision with a paddle reverses the horizontal velocity. If a scoring condition is met, the score is updated, and the ball resets to the center.
- **Paddle Movement**: Paddle positions are updated based on keyboard inputs. The keyboardDown function maps keys (e.g., W and S for Player 1, arrow keys for Player 2) to paddle movements.
- **Game Reset**: The resetGame() function reinitializes scores and positions, enabling a fresh game state.

Shapes like paddles, the ball, and the central line are rendered using drawRect() and drawCircle(). The game is programmed in C++ and uses OpenGL for graphics rendering. The project applies basic graphics programming techniques, making it easy to understand and implement.

Conclusion

This Ping Pong game demonstrates the use of OpenGL and basic C++ programming for creating interactive graphical applications. By combining algorithms for collision detection, input handling, and rendering, the game provides a simple yet engaging experience. It serves as a foundational project for understanding computer graphics and game development.