TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
A Project Report

On

Ping Pong Game using Pygame

Submitted By:

SUSHAL PANDEY
(HCE081BCT042)

Submitted To:

Department of Electronics and Computer Engineering

Himalaya College of Engineering

Chyasal, Lalitpur

Feb-23,2026

ACKNOWLEDGEMENT

We express our sincere gratitude to all those who supported and guided us throughout the development of this project titled “Ping Pong Game Using Pygame.” The successful completion of this work would not have been possible without their valuable assistance, encouragement, and continuous support.

We would like to extend our heartfelt thanks to our supervisor, Sushant Pandey, whose guidance, expertise, and constructive feedback played a crucial role in shaping the direction and quality of this project. Their insightful suggestions and constant motivation greatly helped us in overcoming challenges during the development process.

We are also thankful to our department and faculty members for providing the necessary resources and learning environment that encouraged creativity and technical growth. Special appreciation goes to our friends and classmates for their meaningful discussions, testing support, and moral encouragement throughout the project.

Although any shortcomings in this project are solely our responsibility, the contributions and support we received have significantly enriched the overall outcome of this work.

Thank you.

SUSHAL PANDEY (HCE081BCT042)

ABSTRACT

This project, titled “Ping Pong Game Using Pygame,” focuses on the design and development of a two-dimensional arcade-style game using the Python programming language and the Pygame library. The objective of this project is to demonstrate the implementation of real-time graphics rendering, collision detection, simple artificial intelligence, and interactive gameplay mechanics within a structured game loop.

The game provides both single-player mode (Player vs AI) and two-player mode, allowing users to experience competitive gameplay. Core features include paddle movement control, ball physics with dynamic collision angles, score tracking, countdown system, pause functionality, and win conditions. To enhance user experience, the game also incorporates visual effects such as animated ball trails, particle effects, screen shake, neon-style background design, and interactive power-ups that temporarily modify gameplay behavior.

This project highlights fundamental concepts of game development, including event handling, object-oriented programming, animation techniques, and timing control using frames per second (FPS). Through this implementation, practical knowledge of computer graphics and interactive system design is achieved. The developed system is efficient, responsive, and demonstrates how Python can be effectively used to build engaging 2D games.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Computer Graphics (CG) focuses on creating and displaying visual content using computers through coordinate systems, rendering, and animation techniques. This report presents a mini project titled “Ping Pong Game Using Pygame,” which demonstrates basic 2D graphics concepts such as object movement, collision detection, real-time rendering, and user interaction. The project implements a classic ping pong game using Python and the Pygame library, applying fundamental computer graphics principles in an interactive environment.

1.1 Background Introduction

Computer graphics play an important role in modern interactive applications, especially in game development. It enables the creation of visual objects, movement, animation, and user interaction within a digital environment. In 2D games, concepts such as coordinate systems, collision detection, and real-time rendering are fundamental. The “Ping Pong Game Using Pygame” project is based on these core principles, applying basic computer graphics techniques to create an interactive and visually engaging game.

1.2 Motivation

The motivation behind developing the “Ping Pong Game Using Pygame” project is to gain practical knowledge of computer graphics and game development concepts. By implementing a simple yet interactive game, this project helps in understanding real-time rendering, object movement, collision detection, and user input handling. It also provides hands-on experience in using Python and the Pygame library to build engaging graphical applications.

1.3 Objectives

The main objectives of the project are listed below:

- Implement a 2D Ping Pong game using Python and the Pygame library.
- Develop real-time ball movement and paddle control using coordinate systems.
- Incorporate single-player (AI) and two-player modes for interactive gameplay.
- Enhance visual experience using animations, particle effects, and dynamic rendering.

- Evaluate game performance based on frame rate (FPS) and responsiveness.

1,4 Scope

This project covers fundamental CG concepts used in 2D rendering:

- Implementation of a 2D Ping Pong game with real-time paddle and ball movement.
- Single-player (AI) and two-player gameplay modes.
- Collision detection between ball, paddles, and screen boundaries.
- Visual enhancements such as animated ball trails, particle effects, and screen shake. Power-up features that temporarily modify gameplay behavior

It does not include online multiplayer functionality, advanced physics engines, complex AI algorithms, or 3D graphics rendering. These features can be considered as future enhancements.

2. LITERATURE REVIEW

2.1 2D Game Rendering and Real-Time Graphics

Real-time 2D games rely on continuous screen updates using a structured game loop. The rendering process involves clearing the screen, updating object positions, handling collisions, and redrawing objects at a stable frame rate. Efficient rendering ensures smooth gameplay and responsiveness.

2.2 Collision Detection and Game Physics

Collision detection is fundamental in arcade-style games. In 2D environments, rectangular collision (bounding box method) is commonly used for detecting interactions between objects. Basic physics concepts such as velocity, reflection, and angle variation are applied to simulate realistic ball movement after hitting paddles or boundaries.

2.3 Artificial Intelligence in Simple Games

Basic AI in 2D games often uses rule-based logic. In paddle-based games, the AI typically tracks the ball's vertical position and moves accordingly with a speed limit or dead zone. This approach provides competitive gameplay without requiring complex algorithms.

2.4 Related Tools and Frameworks

Python libraries such as Pygame provide built-in modules for graphics rendering, event handling, sound playback, and timing control. While advanced engines like Unity or Unreal offer more features, Pygame is suitable for educational and small-scale projects due to its simplicity-and-accessibility.

3. METHODOLOGY

3.1 System Overview

The project is organized into main modules:

Game Loop Module: Controls update–render cycle and maintains frame rate (FPS).

• **Rendering Module:** Draws paddles, ball, background, scores, and visual effects.

• **Physics Module:** Handles ball movement, collision detection, and bounce behavior.

• **AI Module:** Controls the computer opponent in single-player mode.

• **Power-Up Module:** Manages spawning, detection, and timed gameplay effects.

• **Input Module:** Handles keyboard interactions for paddle control and menu options.

3.2 Game Logic Foundation

3.2.1 Coordinate System

All objects (paddles and ball) are positioned using a 2D coordinate system. The ball position is updated using velocity components:

$$X=X+VX$$

$$Y= Y+VY$$

3.2.2 Collision Response

When the ball collides with a paddle, its horizontal velocity is reversed, and its vertical velocity is adjusted based on the hit position. This creates different bounce angles and dynamic gameplay.

3.3 Algorithms Used

3.3.1 Paddle–Ball Collision Detection

Rectangular collision detection (bounding box method) is used to check overlap between the ball and paddle rectangles.

3.3.2 AI Movement Algorithm

The AI compares the ball's vertical position with its paddle center. If the difference exceeds a small threshold (dead zone), the paddle moves toward the ball at a fixed speed.

3.3.3 Power-Up Handling

Power-ups are spawned at timed intervals. When the ball collides with a power-up, a temporary effect is activated for a fixed duration.

3.3.4 Implementation Snippet

The following snippet shows the basic idea of updating ball position and handling collision:

- *# Update ball position*
- ball.X += ball.vX
- ball.Y += ball.vY
- *# Collision with paddle*
- If ball_rect.colliderect(paddle):
- ball.vX *= -1

3.4 Screenshots / Figures

Add your actual screenshots to the images/ folder and update filenames below.

3.4.1 Sample Table

Test Case	Resolution	Avg FPS
Normal gameplay	900×600	60
With particle effects	900×600	55
Multi-ball activated	900×600	50

Table 1: Performance observation under different gameplay conditions

Figure 1: Main Menu Interface

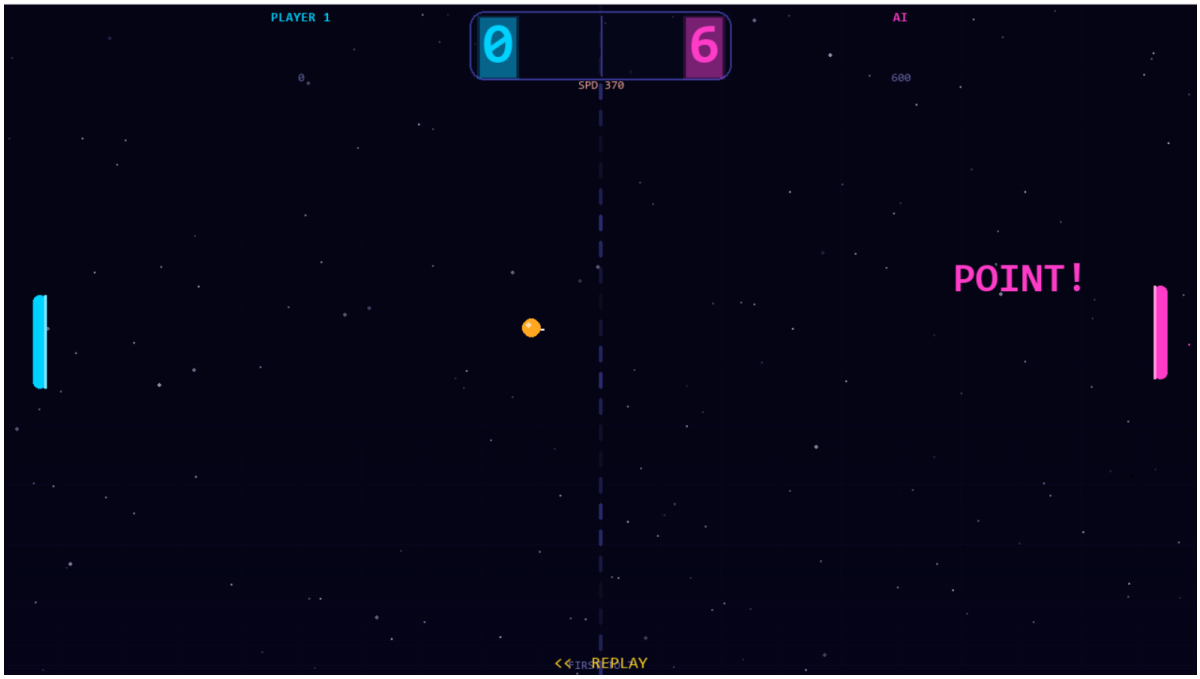

Figure 2: Single Player Mode Gameplay

Figure 3: Two Player Mode Gameplay

4. RESULT AND ANALYSIS

4.1 Correctness of Game Output

The game successfully renders paddles, ball, background, and score elements in real time. Ball movement and paddle interactions behave correctly according to collision detection logic. When the ball hits paddles or boundaries, it reflects realistically with adjusted velocity, ensuring smooth and continuous gameplay.

4.2 Gameplay Behavior

Interactive gameplay demonstrated the following:

- **Single-Player Mode:** The AI paddle tracks the ball position and responds within-a-defined-speed-limit.
- **Two-Player Mode:** Both paddles respond accurately to keyboard input without delay.
- **Power-Ups:** Temporary gameplay modifications such as paddle size increase, speed variation, and multi-ball activation function correctly within their time limits.
- **Game States:** Menu, countdown, play, pause, and game-over transitions operate consistently.

The interaction between game physics and rendering produces stable and engaging user experience.

4.3 Performance Discussion

The game maintains a stable frame rate (approximately 60 FPS) during normal gameplay. When additional visual effects such as particles or multi-ball mode are activated, minor FPS variations may occur due to increased rendering operations. However, the system remains responsive and playable under typical conditions.

4.4 Limitations

- AI behavior is rule-based and can become predictable.
- No online multiplayer or network-based gameplay.
- Physics simulation is simplified and does not include advanced motion dynamics.

5. CONCLUSION AND FUTURE ENHANCEMENT

5.1 Conclusion

This project demonstrates the practical application of 2D computer graphics and interactive game development using Python and Pygame. By implementing real-time rendering, collision detection, AI logic, and visual effects, the project successfully recreates a functional and visually enhanced Ping Pong game. It provides hands-on understanding of core graphics principles within an interactive environment.

5.2 Future Enhancements

•**Improved AI:** Implement adaptive difficulty or predictive ball tracking.
• **Sound and Music Enhancements:** Add dynamic audio effects and background music-controls.
•**Difficulty Levels:** Introduce speed scaling and advanced gameplay modes.
•**Score Saving System:** Add high-score tracking with file storage.
•**Multiplayer Expansion:** Implement online multiplayer functionality.
• **Advanced Graphics:** Add smoother animations and enhanced visual effects.

APPENDICES

Key Controls (Example)

- **W/S:** Move left paddle up / down
- **UP / DOWN Arrow Keys**: Move right paddle up / down (Two Player Mode)
- **1**: Start Single Player Mode (vs AI)
- **2**: Start Two Player Mode
- **P**: Pause / Resume the game
- **ESC**: Exit the game

BIBLIOGRAPHY

[1] M. McShaffry and D. Graham, *Game Coding Complete*, 4th ed., Cengage Learning, 2012.

[2] S. Rabin, *Introduction to Game Development*, 2nd ed., Charles River Media, 2010.

[3] Pygame Documentation, "Pygame Library Reference," Available: https://www.pygame.org/docs/

[4] Python Software Foundation, "Python Documentation," Available: https://docs.python.org/3/

[5] E. Angel and D. Shreiner, *Interactive Computer Graphics: A Top-Down Approach*, 7th ed., Addison-Wesley, 2015.