TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

A Project Report

On

Snake Game using python

Submitted By:

Aditya Sharma (HCE081BEI004)

Submitted To:

Department of Electronics and Computer Engineering

Himalaya College of Engineering

Chyasal,Lalitpur

Feb-22,2026

ACKNOWLEDGEMENT

I express my sincere gratitude to all those who have supported and guided us throughout the development of this project on Snake Game using Python (Pygame). This endeavor would not have been possible without their valuable contributions, encouragement, and technical assistance.

The authors extend their heartfelt thanks to our supervisor, whose expertise, constructive feedback, and continuous guidance played a vital role in shaping the structure and functionality of this game. Their suggestions helped us understand important programming concepts such as game loops, collision detection, image handling, and sound integration in Pygame.

I would also like to thank our friends and colleagues for their meaningful discussions, debugging assistance, and moral support during the development process. Their collaborative input greatly enhanced the quality and efficiency of this project.

Additionally, I acknowledge the developers of Python and the Pygame library, whose powerful tools and documentation made it possible to create an interactive 2D game with graphics and sound features.

This project would not have been successfully completed without the collective efforts and support of these individuals and resources. Although any shortcomings in this project are solely our responsibility, their guidance and encouragement have significantly enriched its outcome.

Thank you.

Aditya Sharma (HCE081BEI004)

ABSTRACT

This project focuses on the development of a 2D Snake Game using Python and the Pygame library. The game features a player-controlled snake that moves within a grid-based environment, randomly spawning apples, sound effects for eating and crashing, a scoring system, and a restart mechanism. The primary objective of the project is to create an engaging and interactive game while demonstrating fundamental concepts of game development such as real-time user input handling, collision detection with walls and the snake's own body, coordinate-based movement, image rendering, and audio integration. As the player collects apples, the snake grows longer, increasing the difficulty level and enhancing the overall gaming experience. This project serves as a practical implementation of 2D game design principles using Python and Pygame.

Table of Contents

List of Figures

1. INTRODUCTION

Game development is an important application of computer graphics and programming that focuses on creating interactive visual experiences using computers. Modern 2D games rely on coordinate systems, image rendering, real-time user input handling, and collision detection to simulate engaging gameplay. This report presents a mini project titled "2D Snake Game using Python and Pygame", which demonstrates fundamental concepts of 2D game design such as grid-based movement, object positioning, collision detection, sound integration, and score management.

1.1 Background Introduction

At a fundamental level, a 2D game operates through a continuous game loop that processes user input, updates object positions, checks for collisions, and renders updated graphics on the screen. Even in simple games like Snake, essential programming concepts are applied, including coordinate systems, frame rate control, event handling, image scaling, and audio playback. The Pygame library provides tools for rendering images, detecting keyboard events, and managing multimedia components efficiently, enabling the creation of interactive graphical applications.

1.2 Motivation

Many beginners learn programming through theoretical examples but struggle to understand how logic translates into real-time visual interaction. This project is motivated by the need to bridge that gap by developing a simple yet interactive game from scratch. By building the Snake Game, we gain practical experience in handling real-time input, implementing movement mechanics, managing dynamic data structures, detecting collisions, and integrating sound effects. The project strengthens understanding of fundamental programming and game development principles in an engaging and hands-on manner.

1.3 Objectives

The main objectives of the project are listed below:

- Develop a fully functional 2D Snake Game using Python and the Pygame library.
- Implement real-time snake movement using keyboard controls within a grid-based coordinate system.
- Design and integrate collision detection mechanisms for walls and self-collision

scenarios.

- Implement a scoring system that increases as the snake consumes apples and display it dynamically on the screen.

1.4 Scope

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

- Develop a simple and interactive 2D Snake Game using Python and Pygame.
- Provide real-time keyboard-based control for snake movement.
- Implement collision detection with walls and self-body.
- Generate apples at random grid positions.
- Maintain and display a dynamic scoring system.

It does not include a full 3D pipeline, advanced shading (Phong/PBR), or GPU shader programming. Those are suggested as future enhancements.

2. LITERATURE REVIEW

2.1 2D Game Development Concepts

2D games are built upon fundamental programming concepts such as coordinate systems, object movement, collision detection, and real-time rendering. In grid-based games like Snake, object positions are updated incrementally within a continuous game loop. Efficient data structures such as lists are commonly used to manage dynamic objects like the snake's body segments. Collision detection techniques are essential to determine interactions between the snake, walls, and food items.

2.2 Game Loops and Event Handling

A core component of any interactive game is the game loop, which continuously processes user input, updates the game state, checks for collisions, and renders updated visuals to the screen. Event-driven programming enables real-time keyboard input handling, allowing smooth directional control. Frame rate management (FPS control) ensures consistent gameplay speed across different systems.

2.3 Multimedia Integration in 2D Games

Modern 2D games integrate images, sound effects, and text rendering to enhance user experience. Libraries such as Pygame provide built-in functions for loading images, scaling sprites, playing audio files, and displaying dynamic text such as scores and game-over messages. Sound effects, such as eating and crash sounds, improve feedback and interactivity.

2.4 Related Tools and Frameworks

Game development can be implemented using various tools and frameworks. Python-based libraries such as Pygame are widely used for educational and beginner-level projects due to their simplicity and accessibility. While advanced engines like Unity or Unreal Engine support complex 3D rendering and physics systems, lightweight frameworks like Pygame are ideal for learning fundamental game development principles and understanding core programming logic.

3. METHODOLOGY

3.1 System Overview

The Snake Game project is organized into the following main modules:

- **Game Initialization Module:** Initializes the Pygame environment, sets up the display window, loads images and sound files, and configures frame rate control.
- **Snake Module:** Manages the snake's position using a dynamic list structure, updates movement based on keyboard input, and handles snake growth when food is consumed.
- **Apple Module:** Generates apples at random grid-aligned positions while ensuring they do not overlap with the snake's body.
- **Collision Detection Module:** Detects wall collisions and self-collisions, triggering game-over conditions and sound effects accordingly.
- **Rendering Module:**Continuously updates the screen by drawing the background, snake, apple, score, and game-over messages.
- **Interaction Module:**Handles keyboard inputs for direction control, restart functionality, and quitting the game.

3.2 Logical Foundation

3.2.1 Grid-Based Coordinate System

The game operates on a 2D coordinate system divided into equal grid blocks. Each object's position is defined using (x, y) coordinates aligned to a fixed grid size. Movement is implemented by incrementing or decrementing the x or y coordinate by a constant value (GRID SIZE), ensuring consistent block-based motion.

Snake movement logic:

- Moving Right $\rightarrow$ x = x + GRID SIZE
- Moving Left $\rightarrow$ x = x $-$ GRID SIZE.
- Moving Up $\rightarrow$ y = y $-$ GRID SIZE
- Moving Down $\rightarrow$ y = y + GRID SIZE

3.2.2 Game Loop and State Management

The core of the game is a continuous game loop that performs the following steps:

- Process user input (keyboard events).
- Update snake position based on current direction.
- Check for collisions (wall or self-collision).
- Check for apple consumption.
- Update score and snake length if needed.
- Render updated graphics on the screen.
- Control execution speed using FPS regulation.

Control execution speed using FPS regulation.

3.3 Dynamic Data Structure

The snake's body is stored as a list of coordinate pairs:

snake pros = [[x1][y1],[x2][y2],...]

This dynamic update mechanism enables smooth movement and growth behavior.

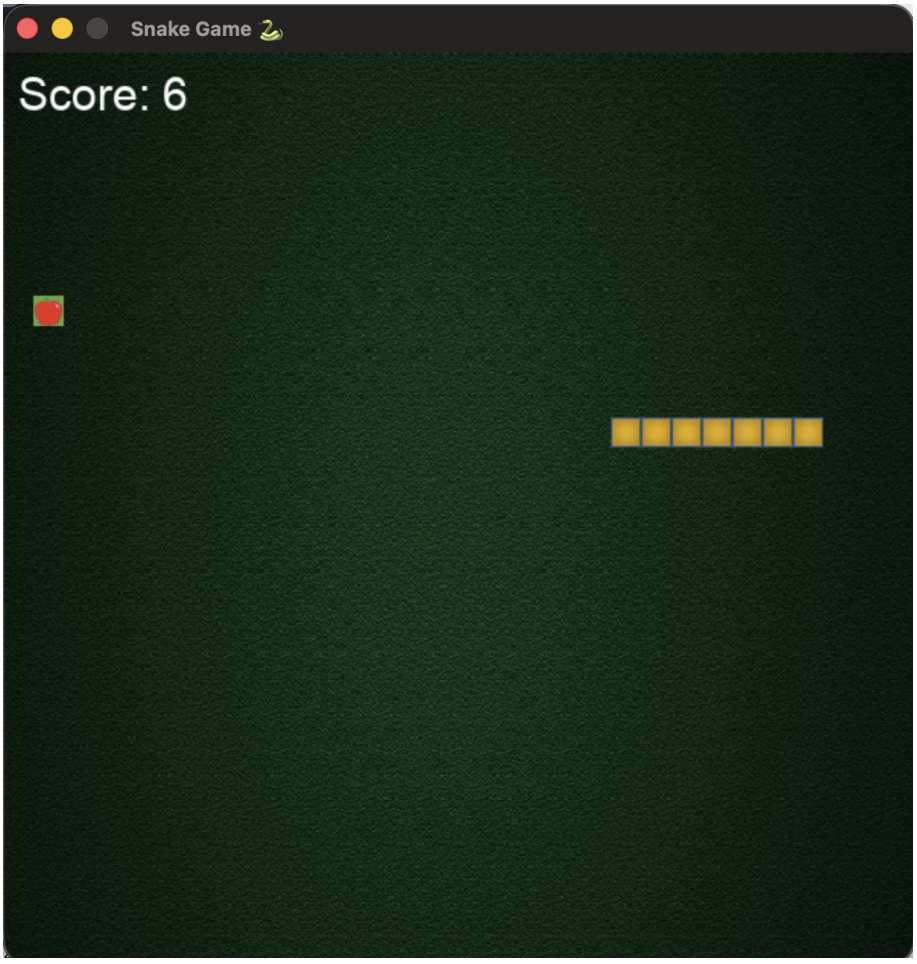

Figure 3.1: Working of Game

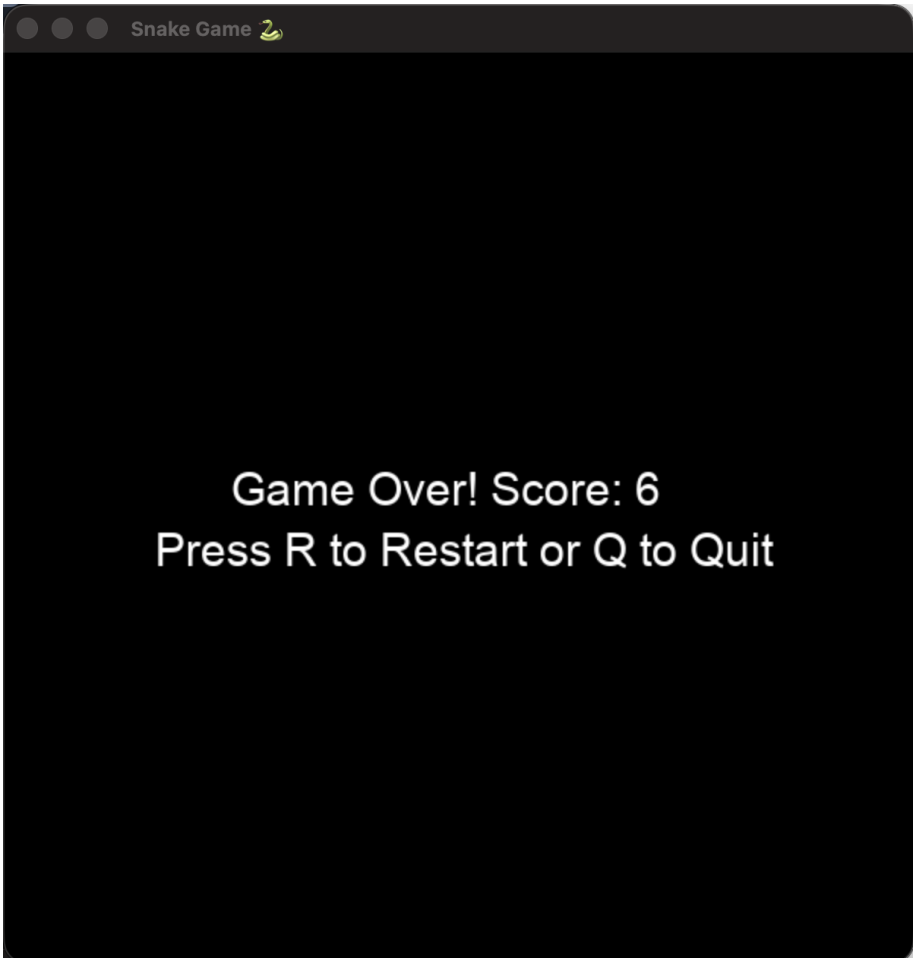

Figure 3.2: Screen after game ends

4. RESULT AND ANALYSIS

4.1 Correctness of Game Rendering

The Snake Game correctly renders all visual elements, including the background, snake, apples, and score display. The snake and apple images are displayed at the correct grid-aligned positions, and movement appears smooth due to proper frame rate control. Collision detection with walls and self-body works as expected, triggering game-over conditions reliably.

4.2 Snake Movement and Gameplay Behavior

- **Snake Movement:** The snake moves uniformly along the grid based on keyboard input without overlapping or skipping blocks.
- **Apple Consumption:** Apples appear at random positions within the grid and do not overlap with the snake's body. Eating an apple increases the snake's length and updates the score.
- **Collision Detection:** Hitting a wall or colliding with the snake's own body triggers the crash sound and ends the game.
- **Restart Functionality:** The game can be restarted using the "R" key, resetting the snake, apple, and score properly.

The game demonstrates consistent behavior for all interactions, showing that the logic for movement, growth, and collisions is reliable.

4.3 Performance Discussion

The game maintains smooth performance at 7–10 FPS. Since the game uses a small grid and simple image rendering, CPU usage is minimal. Increasing FPS improves responsiveness but may require higher frame rate management to keep gameplay consistent. Sound playback does not affect performance significantly.

4.4 Limitations

- No advanced animations; snake body movement is block-based and not interpolated.
- Apple spawn is random; it could occasionally be challenging if placed near edges.
- No AI or automatic gameplay; the game relies entirely on user input.
- Sound and graphics are simple; there is no background music or visual effects beyond images and sounds.

5. CONCLUSION AND FUTURE ENHANCEMENT

5.1 Conclusion

This project demonstrates the fundamental concepts of 2D game development using Python and Pygame. By implementing a grid-based Snake Game, it illustrates real-time movement, collision detection, dynamic object management, and multimedia integration such as images and sound effects. The interactive gameplay helps visualize how coordinate-based positioning, frame rate control, and event handling come together to create a responsive and engaging game experience. The project reinforces practical programming skills and provides a solid foundation for learning more advanced game development concepts

5.2 Future Enhancements

- **Improved Graphics:** Add animations for snake movement and apple consumption to make the game visually smoother.
- **Dynamic Difficulty:** Gradually increase game speed as the player's score rises for added challenge.
- **Score Saving and Leaderboards:** Implement high score tracking and display to enhance replay value.
- **AI or Obstacles:** Introduce automated snakes, moving obstacles, or multiple levels to increase complexity.

A. APPENDICES

A.1 Key Controls

- **↑/↓/←/→:** Move the snake up, down, left, and right
- **R:** Restart the game after Game Over
- **Q:** Quit the game
- **Esc:** Close the game window