CHANDIGARH UNIVERSITY

(Department of Computer Applications)

A MINI PROJECT REPORT ON

SNAKE GAME WITH OBSTACLES
(Using Python and Pygame)

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF COMPUTER APPLICATIONS (AI & ML)

By

Rabinder kumar

UID: 25MCI10073

Section: 25MAM 2-A

Under the Guidance of

MRS. SARABJEET KAUR

Abstract:

This project titled “Snake Game with Obstacles” is a simple yet engaging Python-based mini-project that utilizes the Pygame library for game development. The game allows the player to control a snake to eat food items and increase its length while avoiding collisions with walls, stones, and itself. The primary objective is to achieve the highest possible score before a collision occurs.

The project demonstrates the use of fundamental programming concepts such as loops, conditionals, event handling, and object collision detection. It also provides a foundation for understanding how real-time game mechanics are implemented using Python. This project serves as an excellent example for beginners learning about Python programming and graphical user interfaces. It provides an engaging platform to understand key programming concepts such as loops, conditionals, data structures, and modular coding practices. The inclusion of walls and obstacles increases the difficulty level, offering a more challenging and dynamic game-play experience compared to the traditional snake game.

The implementation follows a modular design that separates game logic, rendering, and event handling into distinct components, ensuring clarity and maintainability of the code. The program also features a responsive user interface, real-time score updates, and dynamic speed adjustment as the game progresses.

Overall, this project serves as an excellent example of applying Python programming and Pygame to design a simple yet complete game application. It not only enhances logical thinking and problem-solving abilities but also provides a foundation for developing more advanced games and graphical applications in the future.

Chapter 1 : Introduction

The Snake Game is one of the most classic and well-known video games, originally popularized on early mobile phones and computers. The objective of the game is simple yet engaging — the player controls a snake that moves around the screen, eating food and growing in length. The challenge increases as the snake becomes longer and the player must avoid colliding with the walls, obstacles, or itself.

This project, "Snake Game with Obstacles using Python", is a modern version of the traditional Snake game, developed using the Pygame library. The game introduces additional elements such as stone obstacles and boundary walls, which make gameplay more challenging and exciting. The player earns points by eating food, and the speed of the snake increases as the score goes up, testing the player's reflexes and concentration.

1.1 Background

- The Snake Game is one of the oldest and most popular arcade games, originally appearing on early mobile phones and computer systems.
- The concept involves controlling a moving snake that eats food to grow longer while avoiding collisions with the walls, obstacles, and itself.
- This project recreates the classic snake game using Python and the Pygame library, integrating modern design elements and additional challenges like obstacles.
- It serves as an excellent way to learn fundamental programming, logic building, and game development skills.

1.2 Purpose

- To design and develop an interactive 2D game using Python programming.

- To apply the concepts of loops, conditional statements, functions, and graphics handling.
- To enhance logical thinking and problem-solving skills through game mechanics.
- To provide a fun, engaging, and educational experience while learning programming concepts.

1.3 Objectives

- Develop a fully functional snake game where the player controls a snake to eat food and avoid collisions.
- Implement obstacles and boundary walls for increased difficulty.
- Display real-time score updates on the screen.
- Create a game-over screen and allow players to restart or quit.
- Implement smooth snake movement and realistic collision detection using Pygame Rects.
- Ensure the game runs efficiently with adjustable speed and difficulty levels.

1.4 Scope

- The project focuses on 2D game development using Python and Pygame.
- It demonstrates how to create games with simple logic, animation, and user input handling.
- The game is designed for single-player use, controlled via keyboard arrow keys.

- It can be further expanded with new features like sound effects, multiple levels, or score saving.

- Suitable for beginners and students learning Python and basic game programming concepts.

1.5 Significance of the project

- Understanding Game Development Concepts:

This project helps in understanding the fundamentals of game development using Python and the Pygame library. It provides practical experience with loops, event handling, collision detection, and screen updates — the core building blocks of any interactive game.

- Application of Programming Logic:

The Snake Game demonstrates how simple logic and algorithms can create interactive entertainment. It emphasizes concepts such as condition checking, coordinate systems, data structures (lists), and functions in a real-world context.

- Enhancing Problem-Solving Skills:

During development, one has to manage movement, boundary collisions, food placement, and obstacle avoidance logically. This improves logical thinking and debugging skills essential for programmers.

- Introduction to Object-Oriented Game Design:

Even though this project uses functions, it lays the foundation for moving toward object-oriented designs — where each game component (snake, food, obstacle) can be treated as an independent object.

- Foundation for Advanced Game Projects:

By completing this project, learners gain confidence to explore advanced topics such as animation, sound integration, artificial intelligence (AI) for opponents, and smooth motion mechanics.

Chapter 2 : System Analysis

The Snake Game with Obstacles is designed to enhance the traditional snake gameplay by adding new features such as randomly generated obstacles, wall boundaries, and real-time score updates. The system analyzes user inputs to control the snake's movement while continuously checking for collisions with walls, stones, or itself. Developed using Python and Pygame, the game ensures smooth performance, interactive graphics, and dynamic difficulty. This system provides an engaging experience and demonstrates core programming concepts like event handling, loops, and collision detection.

2.1 Problem statement:

The traditional Snake Game is one of the most popular arcade games, where the player controls a moving snake that grows longer after eating food. However, most classic versions lack environmental challenges.This project aims to enhance the classic Snake Game by introducing obstacles (stones) **and** wall boundaries that increase difficulty and engagement.The main problem addressed is:

"How to design an interactive, visually appealing, and logic-based game in Python that challenges the player while maintaining simplicity and smooth gameplay?"

The goal is to develop a 2D grid-based Snake Game using Pygame that allows real-time control, random obstacle generation, and adaptive difficulty through speed variation.

2.2 Requirement Analysis:

- **Game Initialization:**

The system should initialize a game window with fixed dimensions, set up walls around the boundary, and generate random obstacles (stones).

- **Snake Movement:**

The player should be able to control the snake's direction using the arrow keys (Up, Down, Left, Right) for smooth real-time movement.

- **Food Generation and Scoring:**

The game must randomly place food items on the screen, avoiding collision with obstacles or the snake's body.
Each time the snake eats food, its length increases and the score updates.

- **Collision Detection:**

The system should detect collisions between the snake and walls, obstacles, or itself, triggering the game-over event.

- **Game Over and Restart:**

When a collision occurs, the system must display a "Game Over" screen showing the final score and provide options to restart or quit.

- **Performance and Responsiveness:**

The game should maintain smooth animation and quick response to user input using Pygame's timing and event handling mechanisms.

Chapter 3 : System Overview

The system architecture of the Snake Game with Obstacles is based on a modular, event-driven design using the game loop mechanism.
It follows a three-layer structure consisting of:

1. Input Layer:
   Handles player inputs via keyboard events (arrow keys, Enter, Escape).
   These inputs determine the snake's direction and control game restarts or exits.
2. Processing Layer (Logic Layer):
   Responsible for the main game logic, including:
   - Snake movement and growth
   - Random generation of food and obstacles
   - Collision detection with walls, stones, or the snake's body
   - Score calculation and speed adjustment
3. Output Layer (Display Layer):
   Uses the Pygame rendering system to update the game screen in real time, showing:
   - The snake
   - Food items
   - Walls and obstacles
   - Scoreboard and game-over messages

This layered approach ensures smooth interaction between input, logic, and output, making the system modular and easy to enhance.

3.1 System Workflow:

1. The game initializes the Pygame window and sets up all graphical elements.
2. The main game loop runs continuously, handling user inputs and updating object positions.
3. After each move:
   - The system checks for collisions.
   - If food is eaten, the snake grows, score increases, and new food is generated.
   - If a collision occurs with walls, stones, or the snake's body, the game ends.
4. The display refreshes every frame using pygame.display.update().

3.2 Technology Used

- Programming Language: Python 3.x
- Library: Pygame (used for graphics, event handling, and real-time game control)
- Additional Modules: random and sys
- Programming Paradigm: Event-driven and procedural programming
- Development Environment: Any Python IDE such as PyCharm, VS Code, or IDLE
- Operating System: Compatible with Windows, Linux, and macOS
- Hardware Requirement: Basic system with at least 2 GB RAM and Python installed

Chapter 4: Implementation

Overview:

The implementation phase converts the system design into an executable Python program using the Pygame library.
This phase focuses on writing code modules that handle graphics rendering, event handling, collision detection, and score management.
The game is implemented as a real-time, event-driven application where all actions occur within a continuous game loop.

Steps in Implementation

1. Importing Required Libraries
   The modules pygame, random, and sys are imported.
   - pygame handles all graphical and event-based operations.
   - random generates random positions for food and obstacles.
   - sys enables clean program termination.
2. Initialization of Pygame
   Pygame is initialized using pygame.init(), and the game window is created with dimensions 800×600 pixels.
   The game clock and font style are also initialized in this stage.
3. Defining Colors and Game Variables
   Standard RGB color values are defined for all graphical elements such as the snake, food, stones, and walls.
   Other variables like CELL_SIZE, WIDTH, and HEIGHT are set for screen layout and movement calculations.
4. Creating Game Elements
   - Snake: Represented as a list of rectangular segments.
   - Food: A single block generated randomly within the screen boundaries.
   - Walls: Gray rectangles placed around the border of the window.
   - Obstacles (Stones): Brown blocks randomly generated in the central area.

5. Game Loop Implementation
   The main logic is executed inside an infinite loop that continuously:
   - Checks for keyboard input.
   - Updates the snake's position and direction.
   - Detects collisions with food, walls, obstacles, or the snake's own body.
   - Updates the score and refreshes the display each frame.
6. Collision Detection and Scoring
   Collision detection is done using Pygame's Rect objects.
   - When the snake's head overlaps the food rectangle, the score increases, and the snake grows longer.
   - When a collision occurs with a wall, obstacle, or its own body, the game ends.
7. Game Over Handling
   A separate function displays the GAME OVER screen, showing the player's final score.
   The user can press Enter to restart or Esc to quit the game.
8. Restart and Exit Control
   The restart feature re-initializes all variables and restarts the game_loop() function, while the exit option calls sys.exit() to terminate the program.

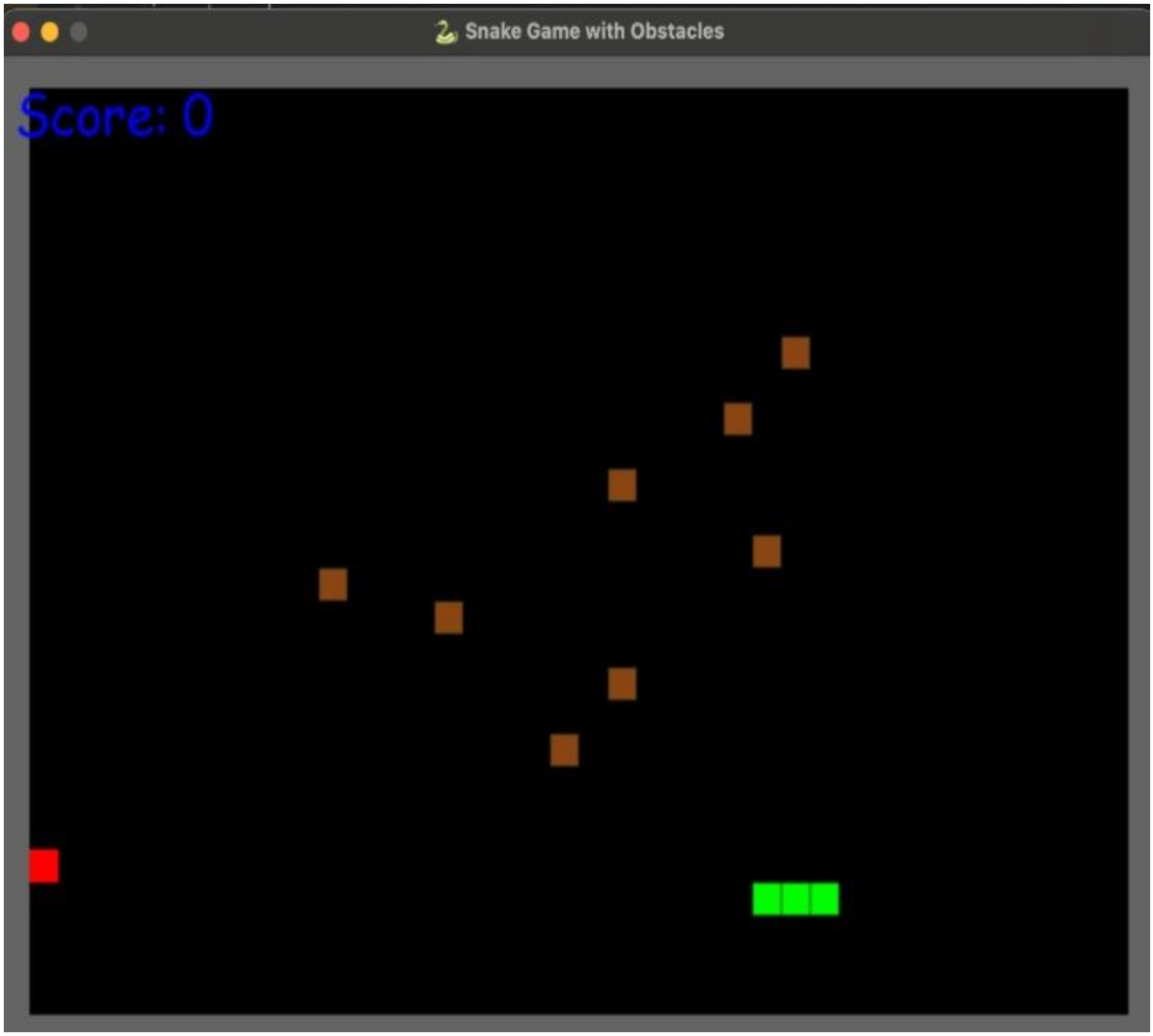
Output Description:

When the game runs:

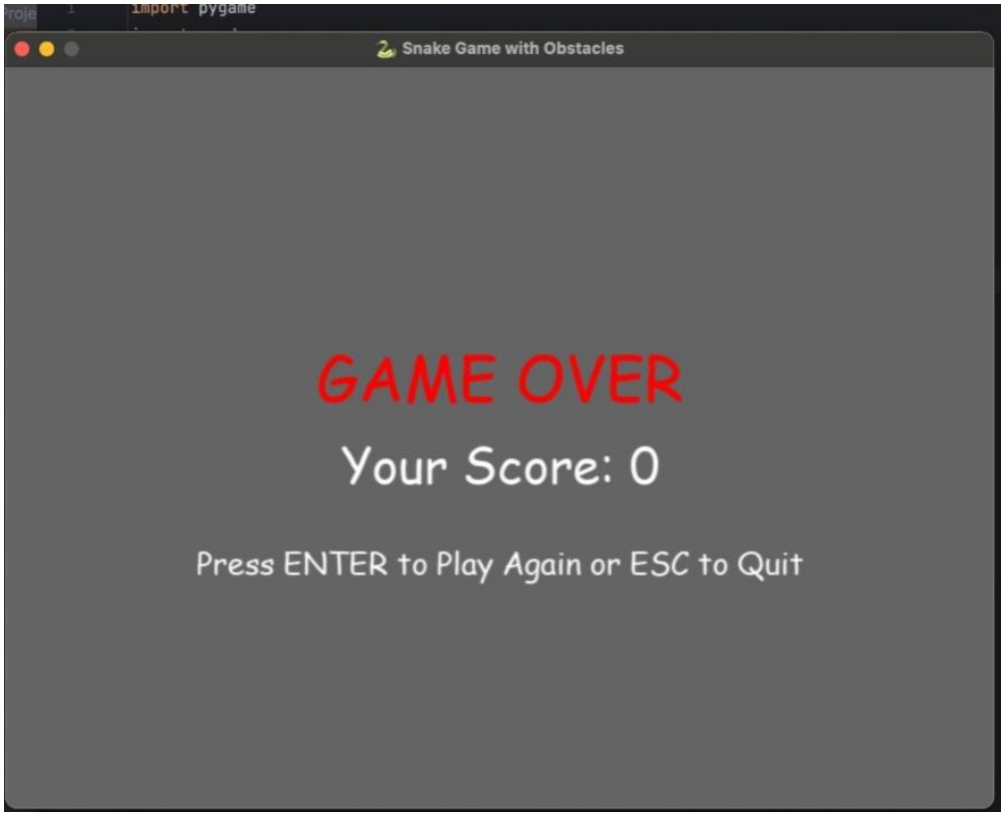
- A green snake appears on the screen with red food blocks and brown obstacles.
- The score updates each time food is eaten.
- The game ends if the snake collides with a wall, obstacle, or itself.
- A final score and restart or exit options are displayed on the Game Over screen.

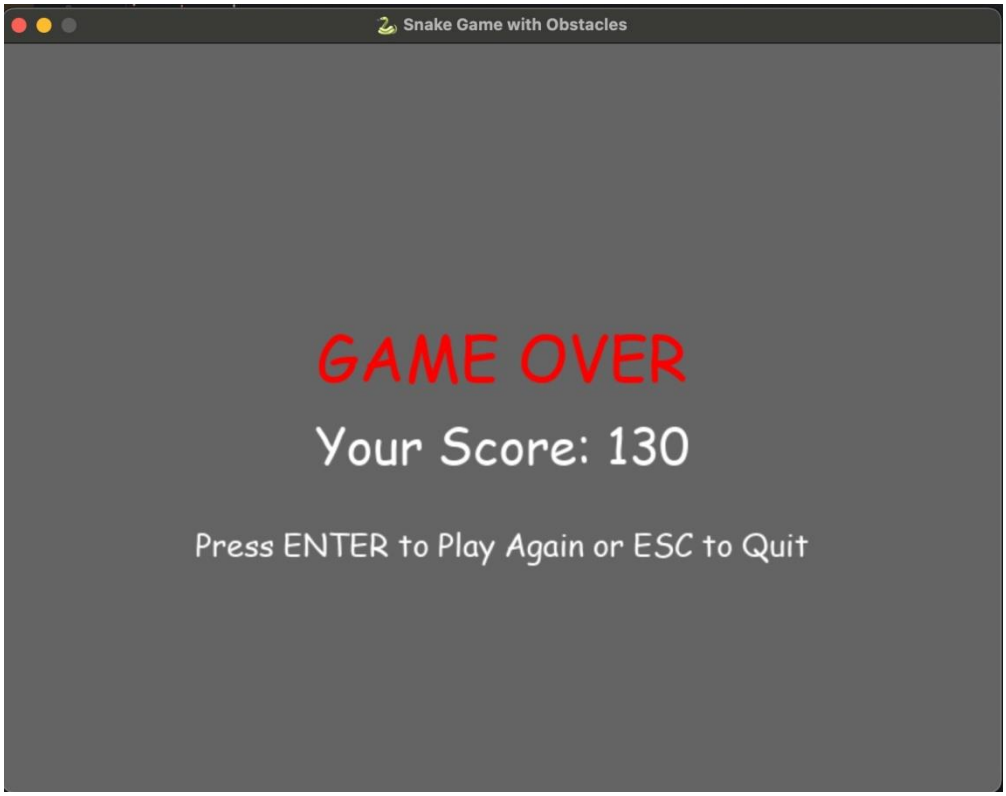
Chapter 5: Output of Designed Project

5.1 Gameplay Screen



5.2 Game over Screen


Snake Game with Obstacles
GAME OVER
Your Score: 0
Press ENTER to Play Again or ESC to Quit


Snake Game with Obstacles
GAME OVER
Your Score: 130
Press ENTER to Play Again or ESC to Quit

Chapter 6 : Conclusion

The Snake Game with Obstacles project successfully demonstrates the use of Python and the Pygame library to create an interactive and enjoyable gaming application. It provides an engaging experience where the player controls the snake to collect food, avoid obstacles, and achieve a higher score. The project highlights key programming concepts such as event handling, collision detection, loops, and real-time graphics rendering. It also improves logical thinking and problem-solving skills through gameplay development. Overall, the project achieves its objective of building a simple yet challenging game that showcases creativity and technical understanding of Python-based game development.

Chapter 7 : Future Scopes

- **Level-Based Gameplay:**
Additional levels with increasing difficulty, faster speed, and more obstacles can be introduced to make the game more challenging.

- **Sound Effects and Background Music:**
Adding sound effects for movement, eating food, and game-over events can enhance the overall gaming experience.

- **High Score System:**
A database or file-based system can be implemented to store and display the highest scores achieved by players.

- **Custom Themes and Skins:**
Players could choose from different snake and background themes to personalize their gameplay experience.

- **Multiplayer Mode:**
Introducing a two-player mode can make the game more interactive and competitive.

- **Mobile and Web Version:**

The project can be extended into a mobile or browser-based game using frameworks like Kivy.

Chapter 8 : References

1. Python Software Foundation. Python Documentation. Available at: https://docs.python.org/3/
2. Pygame Community. Pygame Documentation. Available at: https://www.pygame.org/docs/
3. TutorialsPoint. Python Game Programming Using Pygame. Available at: https://www.tutorialspoint.com/python_pygame/
4. GeeksforGeeks. Creating a Simple Snake Game using Pygame. Available at: https://www.geeksforgeeks.org/snake-game-in-python-using-pygame/
5. W3Schools. Python Basics and Modules. Available at: https://www.w3schools.com/python/