ರೈ ತಾಂತ್ರಿಕ ವಿಶ್ವವಿದ್ಯಾಲಯ

Rai Technology University

(Estd. under Karnataka Act. No. 40 of 2013)

Project Report

On

Sanke Game

Sub:- Machine Learning

Code:- OCS331T

Submitted By:

Name: Tapaswini Shaw, UEN: RTU24101CS024

Name: Hanumakshi , UEN: RTU24101CS006

Name: Suhana , UEN:RTU24101CS010

Program Name: B.TECH CSE AIML

Semester: 3rd Sem

Session: 2025-2026

Submitted To:

Faculty Name: Mr. Kumar Satyam Tanti

College of Engineering & Application

Rai Technology University

Certificate

This is to certify that the project entitled 'Snake Game in Python' has been originally carried out by Tapaswini Shaw, Hanumakshi, Suhana, under the guidance of Mr. Kumar Satyam Tanti in partial fulfillment of the requirements for the degree of Bachelor of Technology at Rai Technology University.

_____ Guide's Signature

_____ Head of Department's Signature

Declaration

We hereby declare that the project report entitled 'Snake Game in Python' submitted to Rai Technology University is our original work and has not been submitted elsewhere for any degree or other purposes.

_____Student's Signature

Acknowledgement

We express our sincere gratitude to Mr. Mr. Kumar Satyam Tanti for his invaluable guidance and support throughout the project. We also thank our department and peers for their encouragement and assistance.

Abstract

The project **“Snake Game in Python”** is a simple yet interactive application developed to demonstrate the use of Python programming concepts and logic building in game development. The game is built using the **Pygame library**, which provides functionalities for graphics, event handling, and game loop execution. The primary objective of the game is to control the snake’s movement using keyboard inputs to collect food items that appear randomly on the screen. Each time the snake eats food, its length increases, and the player’s score is updated. The game ends when the snake collides with the screen boundaries or with its own body.

This project enhances understanding of **fundamental programming concepts** such as loops, conditional statements, functions, event handling, and object-oriented design. Additionally, it introduces essential game development concepts like collision detection, score tracking, and dynamic difficulty management. The Snake Game serves as an engaging way for beginners to strengthen their problem-solving and logical thinking skills while applying Python in a practical scenario.

Table of Contents

GitHub - https://github.com/Hanumakshi/Snake-Game.githttps://github.com/Hanumakshi/Snake-Game.git

Introduction

The Snake Game is one of the most popular classic arcade games that has entertained players for decades. Originally developed for early mobile phones and simple gaming consoles, it is still widely recreated as a beginner-friendly project in programming. The main objective of the game is simple: the player controls a snake that moves continuously across the screen, consuming food items that appear at random positions. Each time the snake eats food, its length increases, making the game progressively more challenging. The game ends when the snake collides with the boundaries of the screen or its own body.

This project, Snake Game in Python, has been developed using the Pygame library, which provides powerful tools for game development, such as graphics rendering, event handling, and sound integration. The project demonstrates how basic Python programming concepts can be applied to build an interactive and engaging application. Through this game, players can enjoy a fun experience, while developers gain practical exposure to important concepts like loops, functions, data structures, event-driven programming, and collision detection.

The purpose of this project is not only to recreate a nostalgic game but also to provide an opportunity for beginners to practice and strengthen their problem-solving and logical thinking skills. Moreover, the implementation of this project helps in understanding how simple ideas can be transformed into functional applications through structured coding.

Objective

The main objectives of developing the **Snake Game using Python** are as follows:

1. **To design a simple and interactive game** that provides an engaging experience for users.
2. **To implement fundamental Python programming concepts** such as loops, conditionals, functions, and data structures in a practical project.
3. **To utilize the Pygame library** for handling graphics, user input, and game loop functionality.
4. **To apply game development concepts** such as collision detection, score tracking, and dynamic difficulty management.
5. **To enhance problem-solving and logical thinking skills** through the implementation of gameplay mechanics.
6. **To understand event-driven programming** by responding to keyboard inputs and updating the game state in real time.
7. **To develop a project that can be extended further**, such as adding levels, sound effects, or customized graphics.

Methodology

The methodology outlines the **step-by-step plan** used to develop the Snake Game in Python:

1. **Problem Analysis**
    - Understand the rules of the classic Snake Game.
    - Identify key challenges: snake movement, collision detection, food generation, and scoring.
2. **Technology Selection**
    - **Programming Language**: Python (for simplicity and readability).
    - **Library**: Pygame (for graphics, event handling, and game loop control).
3. **Game Design**
    - Define the snake as a series of connected blocks.
    - Design food placement logic to appear at random coordinates.
    - Establish game rules: game over occurs on collision with walls or snake's own body.
    - Implement scoring and display.
4. **Development Plan**
    - Initialize Pygame and set up the game window.
    - Create the snake object and define movement controls.
    - Implement food generation and collision detection.
    - Develop the main game loop to update the game state continuously.
    - Test and debug to ensure smooth gameplay.

Implementation

The implementation describes **how the methodology was executed in code**:

1. **Setting Up the Environment**
    - Imported pygame, random, and time.
    - Initialized Pygame, created the game window, set colors, fonts, and clock speed.
2. **Snake Creation**
    - Snake represented as a list of coordinates (blocks).
    - Defined initial position, length, and movement direction.
    - Used pygame.draw.rect() to render the snake on the screen.
3. **Food Generation**
    - Food placed at random positions using random.randint().
    - Checked to prevent food from spawning on the snake's body.
4. **User Input Handling**
    - Captured keyboard events for arrow keys.
    - Updated snake's movement direction based on input.
    - Prevented the snake from reversing direction immediately.
5. **Game Loop Execution**
    - Continuous loop to:
        - Move the snake.
        - Detect collisions with walls or itself.
        - Detect when the snake eats food.
        - Update and display the score.
        - Refresh the screen at each iteration.

6. **Collision Detection**
    - Walls: checked if snake coordinates exceed window boundaries.
    - Self-collision: checked if the snake's head overlaps its body segments.
    - Food: checked if the snake's head overlaps food coordinates.
7. **Score Display and Game Over**
    - Score continuously displayed using Pygame font rendering.
    - On collision, displayed a **"Game Over"** message along with the final score.
8. **Final Outcome**
    - A fully functional Snake Game with smooth controls, dynamic growth, and real-time scoring.
    - Engaging gameplay that increases in difficulty as the snake grows.

Code Explanation

The Snake Game is implemented in Python using the **Pygame library**. The code is structured using an **object-oriented approach** with a SnakeGame class, making it modular and easy to manage.

1. Libraries and Initialization

- **pygame**: Handles graphics, user input, and the main game loop.
- **random**: Generates food at random positions on the grid.
- **sys**: Allows proper exit of the program.
- pygame.init() initializes all Pygame modules.

2. Game Constants

- WIDTH and HEIGHT: Define the game window dimensions.
- GRID_SIZE: Size of each grid cell for the snake and food.
- GRID_W and GRID_H: Number of horizontal and vertical cells.
- Colors are defined using RGB tuples (BLACK, WHITE, GREEN, RED, YELLOW).
- Directions are mapped to (x, y) movements using the DIRS dictionary.

3. Class SnakeGame

The SnakeGame class encapsulates all functionality of the game:

a) Initialization (__init__)

- Sets up the game window, clock, and font.
- Defines game states including levels (Slow, Fast), snake position, initial direction, food position, score, and flags for game_over and menu.

b) Food Generation (new_food)

- Randomly generates a new food position within the grid.
- Ensures food does not appear on the snake's body.

c) Event Handling (handle_events)

- Captures keyboard inputs and window close events.
- **Menu navigation**:
  - 1 selects Slow, 2 selects Fast.
  - ESC returns to the menu or exits the game.
- **Snake movement**: Arrow keys control the snake's direction, with a restriction to prevent reversing into itself.
- **Game over handling**: SPACE returns to the menu.

d) Reset Game (reset_game)

- Resets snake to initial position, direction, score, food, and game-over state.
- Called when a new game starts from the menu.

e) Updating the Game (update)

- Updates the snake's position by adding a new head based on the current direction.
- Checks **food collision**: If the snake eats food, the score increases by 10, and new food is generated; otherwise, the tail is removed to maintain length.
- Checks **collision with walls or self**: Sets game_over = True if collision occurs.

f) Drawing the Game (draw)

- Clears the screen and draws the current state.
- **Menu screen**: Shows title and level selection options.
- **Game Over screen**: Displays "GAME OVER" message, final score, and restart instructions.
- **Game screen**:
  - Draws snake using circles (pygame.draw.circle()), with a slightly different color for the body segments.
  - Adds simple eyes on the snake head according to direction.

  - Draws the food as a red circle.
  - Displays UI elements: current level and score.
- Updates the display using pygame.display.flip().

g) Running the Game (run)

- Main loop that continues while running is True.
- Calls handle_events(), update(), and draw() in each iteration.
- Controls the **FPS (frames per second)** based on the selected level (Slow = 5 FPS, Fast = 12 FPS) to adjust snake speed.
- Exits cleanly with pygame.quit() and sys.exit() when the game ends.

4. Object-Oriented Design and Modularity

- The use of a class ensures that all game components (snake, food, menu, score) are encapsulated.
- Functions like new_food(), handle_events(), update(), and draw() separate different responsibilities, making the code easier to maintain and extend.

5. Enhancements in Gameplay

- Two difficulty levels adjust the snake's speed.
- Realistic circular snake with eyes for a more engaging look.
- Smooth menu navigation and game restart options.
- Dynamic growth of the snake and real-time score tracking.

Results and Discussion

The implementation of the Snake Game in Python resulted in a fully functional and interactive application that meets the objectives set out in the methodology. The following outcomes were observed:

1. Smooth and Responsive Gameplay

- The snake responds accurately to keyboard inputs (arrow keys).
- Movement is fluid, and the snake grows correctly when food is consumed.
- Different difficulty levels (Slow and Fast) adjust the speed, providing a customizable challenge.

2. Score Tracking and User Interface

- The score updates in real-time, incrementing by 10 points each time food is eaten.
- UI elements such as current level and score are displayed clearly on the game screen.
- Menu and Game Over screens are intuitive and easy to navigate.

3. Collision Detection

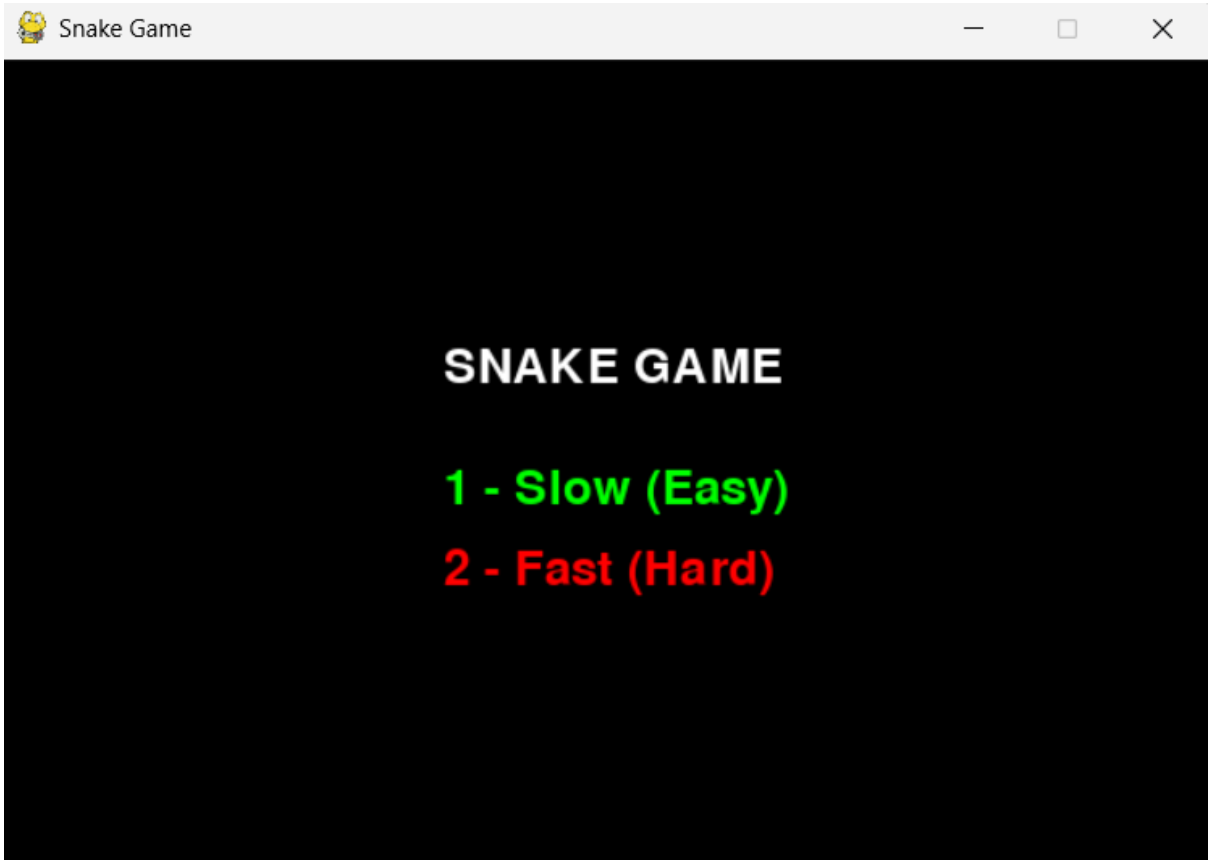
- The game correctly detects collisions with the walls and the snake itself.
- On collision, the game displays a “Game Over” message along with the final score.
- Food is generated randomly in valid positions, ensuring fair gameplay.

4. Player Engagement

- The game provides a progressively challenging experience as the snake grows longer.
- Simple eye graphics on the snake’s head enhance visual appeal.
- The menu allows players to select difficulty levels, restart the game, or return to the menu.

Discussion and Observations

- The project demonstrates the practical application of Python programming concepts, such as loops, lists, functions, and event-driven programming.
- The use of Pygame simplifies handling graphics, user input, and game timing.
- Object-oriented design via the SnakeGame class makes the code modular, readable, and maintainable.
- During testing, minor issues such as initial food placement on the snake were resolved with collision checks, improving robustness.
- Overall, the project provides both educational value and entertainment, highlighting how basic programming concepts can create engaging interactive applications.



Results:

The Snake Game successfully demonstrates the integration of Python programming logic with graphical and interactive elements. It achieves smooth gameplay, accurate collision detection, dynamic score tracking, and a visually appealing interface, fulfilling the objectives of the live project.

Conclusion

The **Snake Game in Python** is a successful implementation of a classic arcade game using object-oriented programming and the Pygame library. The project achieves its primary objectives by providing:

- **Interactive Gameplay:** The snake moves smoothly, responds accurately to user inputs, and grows dynamically as it consumes food.
- **Real-Time Score Tracking:** The player's score updates instantly, and levels allow for adjustable difficulty.
- **Collision Detection:** The game ends reliably when the snake collides with the walls or itself.
- **Modular Code Structure:** The use of a SnakeGame class and modular functions makes the code easy to read, maintain, and extend.

The project demonstrates practical application of Python concepts such as loops, lists, functions, event handling, and object-oriented design, while also offering an engaging and enjoyable gaming experience.

Future Scope

The project can be enhanced in several ways to increase complexity, visual appeal, and replayability:

1. **Multiple Levels:** Introduce new levels with obstacles or larger maps to increase challenge.
2. **Enhanced Graphics and Animations:** Improve the snake and food graphics, add animations, or use sprites for a richer visual experience.
3. **Sound Effects and Music:** Add background music and sound effects for eating food or colliding, enhancing user engagement.
4. **High Score Tracking:** Implement persistent high-score storage to encourage repeated gameplay.
5. **Power-Ups and Special Items:** Add items that temporarily increase speed, shrink the snake, or give bonus points.
6. **Multiplayer Mode:** Allow two players to control separate snakes in the same arena.
7. **AI Opponents:** Introduce computer-controlled snakes to compete against the player.

In summary, this project serves as a foundational example of game development using Python and Pygame, providing scope for continuous learning, creativity, and technical enhancements.