**INTERIM REPORT SNAKE GAME USING PYGAME**

*Dissertation submitted in fulfilment of the requirements for the Degree of*

**BACHELOR OF TECHNOLOGY**

**in**

**COMPUTER SCIENCE AND ENGINEERING**

**– DATA SCIENCE WITH MACHINE LEARNING**

By

**ADITHYAN M D**

**Registration No: 12316297**

**Section: K23CH**

**Roll No: 05**

**20/11/2024**

GitHub: https://github.com/Adithyan47/Snake-game-project.git

**Supervisor: Dr Aman Kumar**



**School of Computer Science and Engineering**

Lovely Professional University

Phagwara, Punjab (India)

October 2024

# ACKNOWLEDGEMENT

I at this moment declare that the research work reported in the dissertation/dissertation proposal entitled

**"SNAKE GAME USING PYGAME"** in partial fulfillment of the requirement for the award of the Degree for Bachelor of Technology in Computer Science and Engineering at **Lovely Professional University,** Phagwara,

Punjab, is an authentic work carried out under the supervision of my research supervisor **Mr. Aman Kumar**.

I understand that the work presented herewith directly complies with Lovely Professional University’s Policy on plagiarism, intellectual property rights, and the highest standards of moral and ethical conduct. Therefore, to the

best of my knowledge, the content of this dissertation represents an authentic and honest research effort

conducted, in its entirety, by me. I am fully responsible for the contents of my dissertation work.

ADITHYAN M D

REG.NO - 12316297

\*\*Table of Contents\*\*

I. Introduction

* 1.1 Project Overview
* 1.2 Purpose and Significance
* 1.3 Registration Details

II. Objectives and Scope of the Project

* 2.1 Project Objectives
* 2.2 Project Scope

III. Application Tools

* 3.1 Software Applications
* 3.2 Programming Languages

IV. Project Structure of the Student Management System (SMS)

* 4.1 Main Components of the Project
* 4.2 Classes and Their Functions
* 4.3. Interaction Between Components

V. Flow chart

- 5.1 Explanation of the Flowchart

VI. Project Implementation

VII. Code Implementation

VII. Testing and Validation

VIII. Conclusion

IX. References

# I. INTRODUCTION

1.1 Project Overview

The **Snake Game using Pygame** is a Python-based interactive arcade game inspired by the classic Snake game. Developed with the **Pygame library**, this project provides a fun and engaging experience where players control a growing snake that navigates a grid to collect food while avoiding collisions with the walls and itself.

The game features smooth movement mechanics, an intuitive user interface, and dynamic difficulty progression. Players earn points by consuming food, and the game ends when the snake collides with itself or the screen boundary.

With visually appealing graphics and responsive controls, the game showcases the ability to utilize Python's game development capabilities to create an enjoyable experience.

1.2 Purpose and Significance

The primary purpose of this project is to demonstrate the implementation of an interactive game using Python and Pygame, highlighting key concepts of game development such as event handling, animations, collision detection, and score tracking.

This game has educational significance as it introduces learners to the fundamental elements of creating a 2D game, including:

* Rendering graphics and animations.
* Handling user input in real time.
* Implementing dynamic object behaviors and interactions.
* Managing game state transitions (e.g., from gameplay to game-over).

From a recreational perspective, the Snake Game offers an engaging way to challenge a player's reflexes, planning, and multitasking skills. It also provides opportunities for enhancement, such as adding levels, introducing obstacles, or integrating multiplayer functionality, making it a versatile project for further exploration.

1.3 Game Features

1. **Game Controls:**
   * Players can control the snake’s movement using arrow keys (Up, Down, Left, Right) or WASD keys.
   * The snake can only move in one direction at a time, and it cannot reverse directly into its own body.
2. **Game Window:**
   * The game window features a grid-like background representing the playable area.
   * Food items appear randomly within the grid, and the snake must navigate toward them to score points.
3. **Scoring Mechanism:**
   * Players earn points for each food item consumed, with the score displayed on the screen.
   * Each food item increases the length of the snake, making navigation more challenging.
4. **Collision Detection:**
   * The game ends if the snake collides with the walls of the game window or with itself.
5. **Dynamic Difficulty:**
   * The speed of the snake increases slightly after consuming a set number of food items, ensuring the game becomes progressively harder.
6. **Visual and Audio Feedback:**
   * The game includes vibrant graphics, such as a multicolored snake and food items.
   * Simple sound effects play upon eating food and triggering the game-over event.
7. **Game-Over Screen:**
   * When the game ends, a "Game Over" message is displayed, along with the player's final score.
   * Players are given the option to restart the game or quit.

By incorporating these features, the Snake Game provides an engaging and immersive experience while showcasing fundamental game development concepts.

# II. Objectives and Scope of the Project

2.1 Project Objectives

he primary objectives of the Snake Game project are:

1. **To Develop an Interactive Game:**
   * Build a functional version of the classic Snake game, incorporating fundamental gameplay mechanics like movement, food collection, and score tracking.
2. **To Enhance Programming Skills:**
   * Provide hands-on experience in designing and implementing a game using Python programming concepts, such as loops, conditionals, and functions.
3. **To Introduce Game Design Concepts:**
   * Familiarize with key aspects of game development, including game loops, user input handling, and collision detection.
4. **To Create a Scalable Codebase:**
   * Develop a structured, modular, and maintainable codebase that can be extended with additional features in the future.
5. **To Engage Users with Intuitive Gameplay:**
   * Ensure the game provides a user-friendly and engaging experience with smooth controls and dynamic difficulty progression.

2.2 Project Scope

The scope of the project includes the following aspects:

1. **Core Gameplay Implementation:**
   * A player-controlled snake that moves across the screen, grows in size when consuming food, and ends the game upon collision with walls or itself.
2. **Basic Game Interface:**
   * A visually appealing game window displaying the snake, food, and current score.
3. **Scoring and Feedback System:**
   * Dynamic score updates as the player progresses and clear feedback upon game-over scenarios.
4. **Platform Independence:**
   * The game is designed to be compatible with multiple platforms, provided Python and the required libraries are installed.
5. **Future Extensions:**
   * The modular design will allow additional features, such as power-ups, obstacle introduction, or customizable difficulty levels, in later versions.
6. **Limitations:**
   * The project is restricted to single-player gameplay and 2D graphics, focusing on simplicity and core functionality over advanced features like 3D visuals or multiplayer modes.

# III. Application Tools

3.1 Software Applications

The development of the Snake Game using Pygame involves the use of several software tools and libraries to ensure efficient game development and an interactive user experience. Below is an elaboration of the tools used:

**1. Python Programming Language (Version 3.x)**

* **Role:** The core programming language used to develop the Snake Game.
* **Advantages:**
  + Python's simplicity and readability make it an excellent choice for beginners and experienced developers alike.
  + Provides extensive libraries for game development, such as Pygame, which simplifies handling graphics, sound, and user input.

**2. Pygame Library**

* **Role:** The primary library used for game development. It provides modules to manage graphics, input handling, sound effects, and other game features.
* **Advantages:**
  + Enables smooth rendering of 2D graphics with minimal effort.
  + Facilitates the creation of game loops, event handling, and collision detection.
  + Includes support for sound effects, making the game more immersive.

**3. Integrated Development Environment (IDE): Visual Studio Code (VS Code)**

* **Role:** A versatile text editor and IDE for writing, debugging, and testing Python code.
* **Advantages:**
  + Syntax highlighting, code linting, and auto-completion features help improve code quality.
  + Integrated terminal for running Python scripts directly within the editor.
  + Extensions like "Python" and "Pylance" enhance development workflows.

**4. Git and GitHub**

* **Role:** Tools for version control and collaborative development.
* **Advantages:**
  + Git helps track changes to the codebase, ensuring the ability to roll back to previous versions if needed.
  + GitHub provides a platform for sharing and storing the code repository, facilitating collaborative enhancements and reviews.

**5. Graphics and Sound Tools**

* **Graphics:**
  + Simple graphical assets for the snake and food can be created using basic tools like Microsoft Paint, Canva, or online icon generators.
  + Pygame also allows direct drawing of shapes, such as rectangles and circles, eliminating the need for external graphics tools.
* **Sound Effects:**
  + Tools like Audacity or online sound effect libraries (e.g., FreeSound, Zapsplat) can provide audio clips for actions like food collection and game-over sounds.

3.2 Programming Languages of the Project

The development of the Snake Game relies primarily on Python as the programming language. Here’s an elaboration of its role and relevance in the project:

**1. Python (Primary Language)**

* **Role:** Python is the primary programming language used to design and implement the Snake Game. It serves as the foundation for all game logic, graphics, and interactivity.
* **Features and Benefits for the Project:**
  + **Ease of Learning and Use:** Python’s syntax is intuitive and beginner-friendly, making it ideal for developing small-to-medium-sized projects like the Snake Game.
  + **Rich Libraries:** Python offers a variety of libraries, such as Pygame, which simplifies game development by handling essential tasks like rendering graphics, managing input, and playing sounds.
  + **Platform Independence:** Python code can run on multiple platforms (Windows, macOS, Linux) with minimal changes, making the game universally accessible.
  + **Extensibility:** Python’s modular nature allows for easy expansion of game features, such as adding levels, new mechanics, or high-score tracking.

**2. Python's Libraries and Modules**

* While Python itself is the main language, its functionality is extended through libraries. The **Pygame** library is central to the Snake Game project.
  + **Pygame Features:**
    - Facilitates rendering of 2D graphics like the snake and food.
    - Handles real-time keyboard input for player control.
    - Provides built-in support for sounds and music to enhance the gaming experience.
  + **Additional Modules Used:**
    - **random:** Generates random positions for the snake’s food.
    - **time:** Handles timing and delays, which are essential for smooth animations.
    - **sys:** Manages program exit and other system-level operations.

# lV . Project Structure of Snake Game

The **Snake Game** using Pygame is organized into several components, classes, and functions, each responsible for managing specific aspects of the game. The architecture of the game ensures modularity, maintainability, and scalability.

**4.1 Main Components of the Project**

1. **Snake Entity (Snake Class):**
   * **Purpose:** Represents the snake in the game.
   * **Responsibilities:** Manages the snake's size, position, movement, and collision detection.
2. **Game Logic (SnakeGame Class):**
   * **Purpose:** Handles the core mechanics of the game.
   * **Responsibilities:** Controls the game loop, manages user input, updates game states, and checks for win or loss conditions.
3. **Graphical User Interface (GUI) (Pygame):**
   * **Purpose:** Provides the visual representation of the game for the player.
   * **Responsibilities:** Renders the game environment, including the snake, food, score, and game over screen.
4. **Storage (Optional for High Scores):**
   * **Purpose:** Stores game data like high scores.
   * **Responsibilities:** Uses in-memory storage (e.g., Python dictionaries) in the initial version. Future versions could implement file storage for persistent high-score tracking.

4.2 Classes and Their Functions

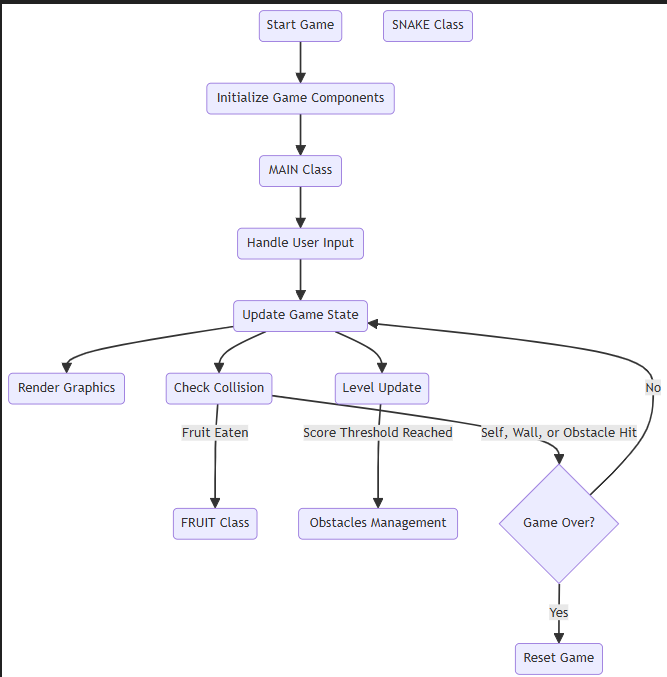
1. **SnakeClass**  
   The Snake class is the core element of the game, representing the snake's attributes and behaviour.
   * **Attributes:**
     + body: A list representing the segments of the snake. Each segment is defined by its position (x, y).
     + direction: Indicates the current direction of movement (e.g., "UP", "DOWN", "LEFT", "RIGHT").
     + color: Defines the color of the snake's body.
   * **Methods:**
     + \_\_init\_\_(self, initial\_position, initial\_length): Constructor to initialize the snake's position, length, and direction.
     + move(self): Updates the snake's position based on its current direction.
     + grow(self): Adds a new segment to the snake's body when it eats food.
     + check\_collision(self): Checks if the snake has collided with itself or the boundaries of the game window.
2. **SnakeGameClass**  
   The SnakeGame class manages the overall gameplay and coordinates interactions between the snake and other elements.
   * **Attributes:**
     + snake: An instance of the Snake class representing the player’s snake.
     + food\_position: A tuple representing the position of the food.
     + score: Tracks the player’s score.
     + window\_size: The dimensions of the game window.
   * **Methods:**
     + start\_game(self): Initializes the game window and starts the main game loop.
     + spawn\_food(self): Randomly generates a new food position on the grid.
     + update\_game\_state(self): Updates the snake’s position, checks for collisions, and handles scoring.
     + display\_game(self): Renders the snake, food, and score on the game window.
     + end\_game(self): Ends the game and displays a "Game Over" screen.
3. **GUI**  
   The GUI is built using Pygame to create a visually appealing and interactive game interface.
   * **Main Components of the GUI:**
     + **Game Window:** The primary canvas for the snake game.
     + **Score Display:** Shows the player’s current score.
     + **Game Over Screen:** Displays when the game ends, with options to restart or quit.
   * **Main Functions:**
     + render\_snake(self): Draws the snake on the screen.
     + render\_food(self): Draws the food on the screen.
     + render\_score(self): Displays the current score on the screen.
     + handle\_input(self): Captures player input to change the snake’s direction.

4.3. Interaction Between Components

1. **Moving the Snake:**
   * The player uses arrow keys to change the snake’s direction.
   * The GUI captures this input and updates the snake’s direction attribute.
   * The SnakeGame class calls the move() method to reposition the snake based on the direction.
2. **Eating Food:**
   * When the snake's head position matches the food position, the grow() method is called to extend the snake.
   * The game spawns new food using the spawn\_food() method.
   * The score is incremented.
3. **Collision Detection:**
   * The game continuously checks for collisions with boundaries or the snake's own body.
   * If a collision occurs, the end\_game() method is triggered, and a "Game Over" screen is displayed.
4. **Updating the Game:**
   * The game loop updates the game state, including the snake's movement, food rendering, and collision detection.
5. **Displaying the Game:**
   * The GUI uses the Pygame library to render the snake, food, score, and other visual elements on the screen.

**V. Flowchart or Algorithm of the Project**

Below is a simplified flowchart that illustrates the process flow of the priority scheduling system.

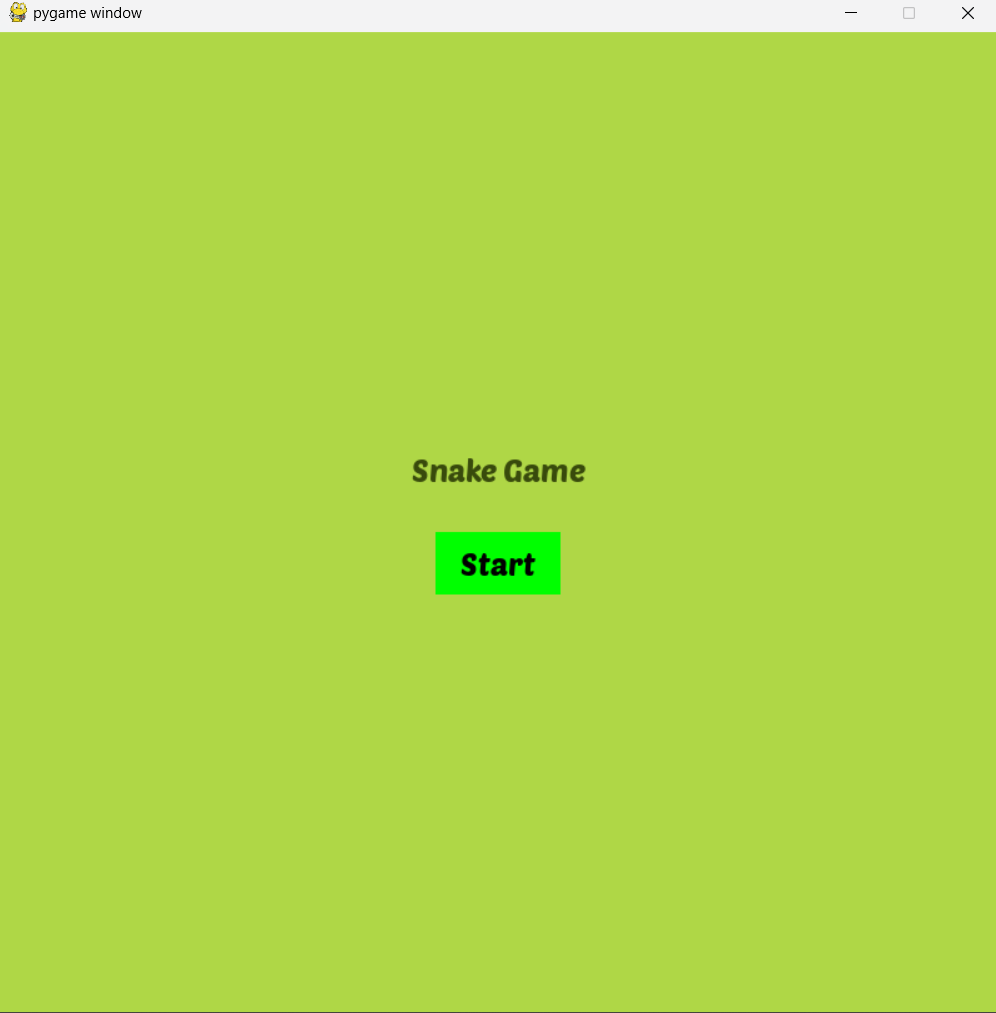


5.1 Explanation of the Flowchart

* Start **Game:** Game begins.
* Initialize **Game:** Set up snake, food, and score.
* Move **Snake:** The snake moves in the current direction.
* Eat **Food:** Checks if the snake eats food; if yes, it grows.
* Collision: Checks for wall or self-collision.
* If a collision occurs, the game ends.
* Repeat**:** The game loops back to move the snake if there's no collision.

**VI. Project Implementation**

* Start button



* Level 1

A screenshot of a computer

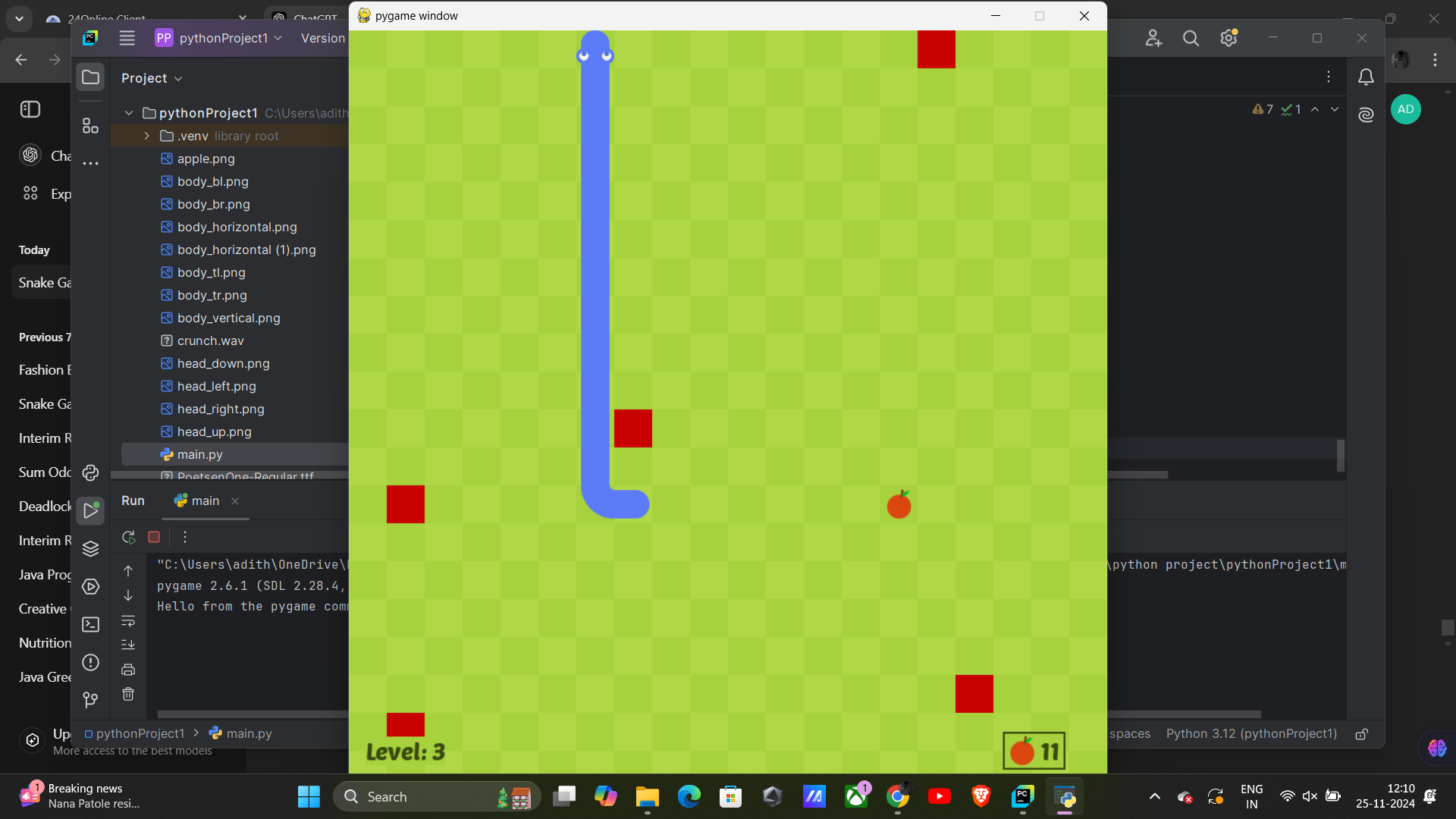
Description automatically generated

* Level 2

A screenshot of a computer screen

Description automatically generated

* Level 3



* Level 4

A screenshot of a computer

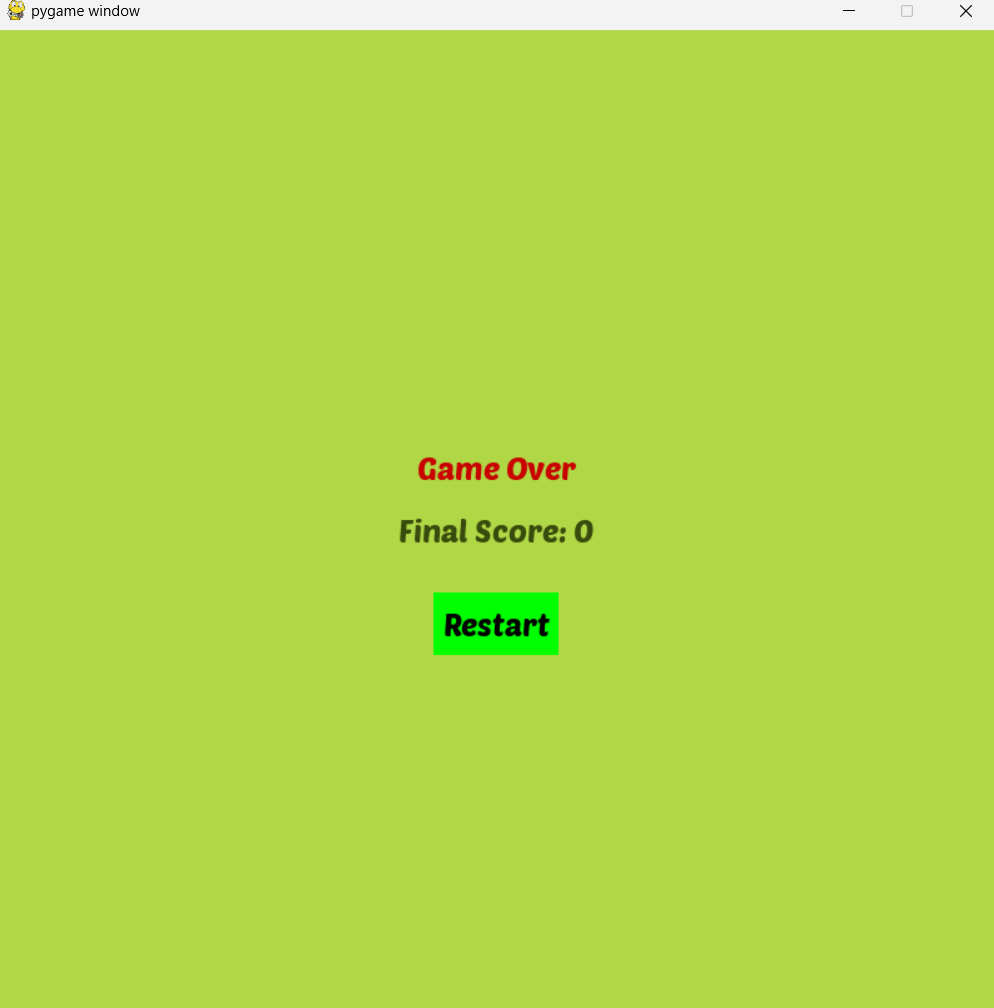
Description automatically generated

* Level 5

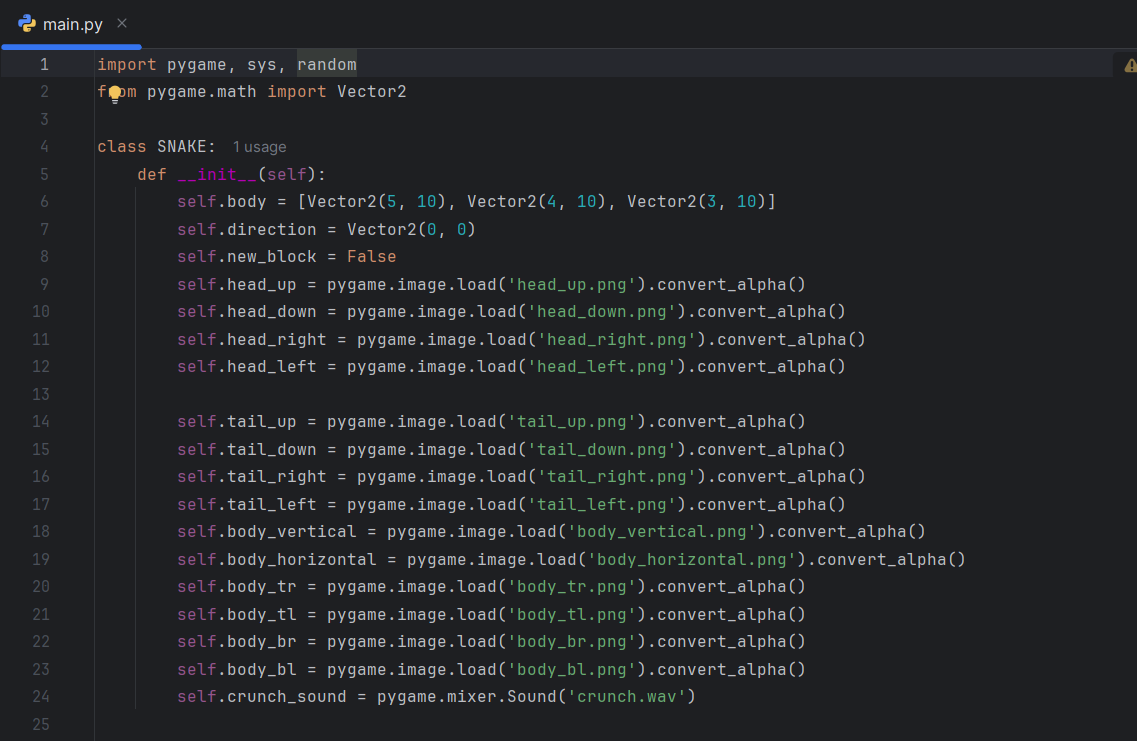
A screenshot of a computer screen

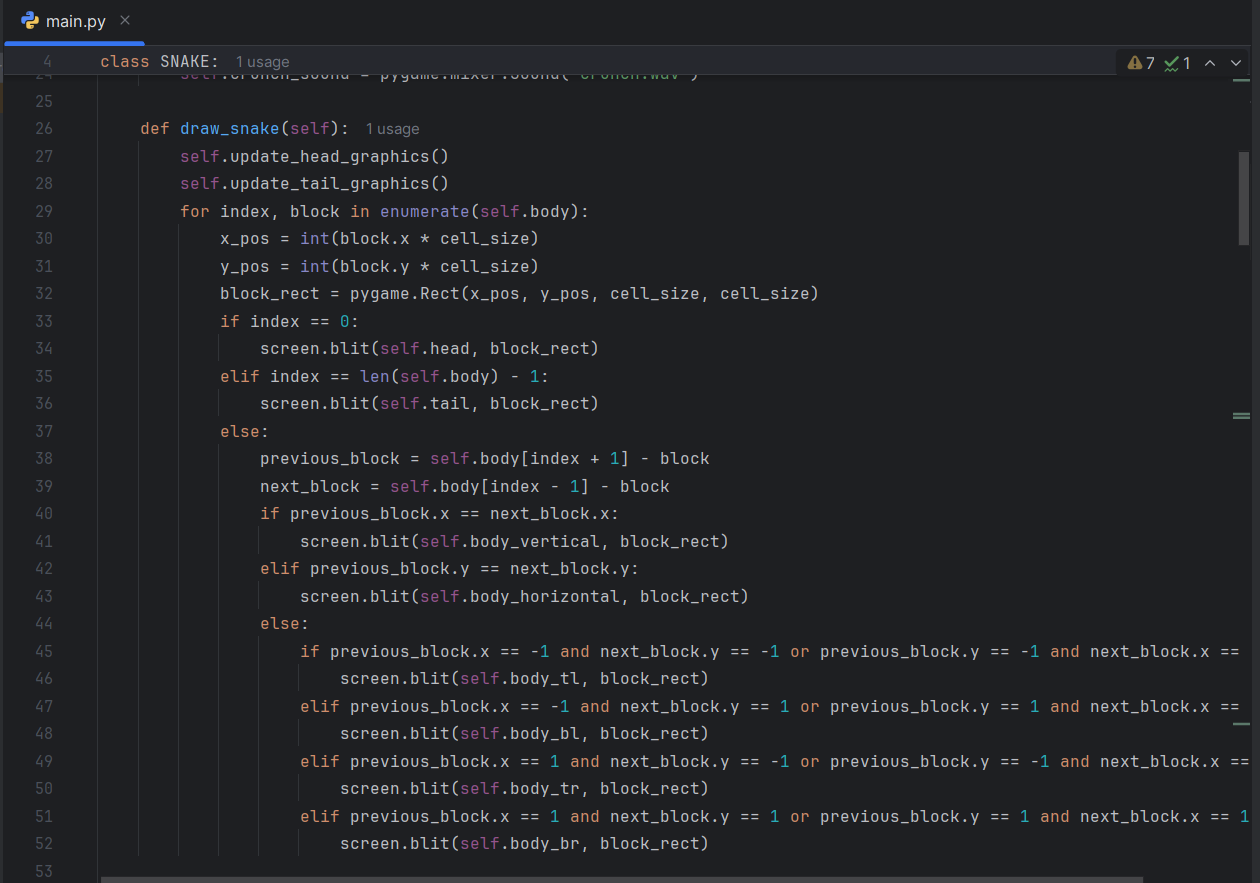
Description automatically generated

* Restart button



**VII. Code Implementation**

****

****

**VIII. Testing and Validation**

The “ConvertEase” project underwent testing to ensure its functionality, accuracy, and reliability. Different testing methods were employed at various stages of development to validate the application’s performance and user experience. Below is a detailed account of the testing process:

1.**Unit testing:**

Unit testing ensures individual components of your Snake game function as expected. These tests check the behavior of the Snake, Fruit, and other game logic independently.

| **Test Case ID** | **Test Description** | **Input** | **Expected Output** | **Actual Output** | **Status** |
| --- | --- | --- | --- | --- | --- |
| **UT01** | Validate snake movement | Move the snake in different directions | Snake moves correctly on the grid | Correct snake movement in all directions | Passed |
| **UT02** | Test snake body length after eating fruit | Snake eats a fruit (fruit randomly placed) | Snake length increases by 1 | Snake length correctly increases | Passed |
| **UT03** | Test snake collision with itself | Move snake into its own body | Game ends or resets | Collision detection works as expected | Passed |
| **UT04** | Validate fruit spawning | Fruit is eaten by snake | New fruit appears in a random position | Fruit spawns in a valid position | Passed |
| **UT05** | Test game over logic | Snake hits the wall or itself | Game ends | Game over condition triggers | Passed |
| **UT06** | Test level progression | Snake eats enough fruit | Level increases | Level increases after 5 fruits are eaten | Passed |
| **UT07** | Test obstacle placement | Snake collides with obstacles | Game ends | Obstacle collision works as expected | Passed |
| **UT08** | Validate game reset functionality | Game over condition occurs | Snake resets to initial position, level resets | Game resets correctly | Passed |

2.Integration Testing

Integration testing focuses on the interaction between different components of the game, such as the snake, fruit, obstacles, and game mechanics. These tests ensure that these components work together smoothly.

| **Test Case ID** | **Test Description** | **Input** | **Expected Output** | **Actual Output** | **Status** |
| --- | --- | --- | --- | --- | --- |
| **IT01** | Test snake movement and fruit interaction | Snake moves towards fruit and eats it | Fruit is eaten, snake length increases | Fruit is eaten correctly | Passed |
| **IT02** | Verify snake, fruit, and obstacle integration | Snake eats fruit and avoids obstacle | Snake grows, level progresses, and no collision | Integration works smoothly | Passed |
| **IT03** | Test level progression and obstacle addition | Snake eats fruit as the level progresses | Obstacles appear and game continues | Obstacles added at higher levels | Passed |
| **IT04** | Verify game-over conditions with obstacles | Snake collides with an obstacle | Game ends | Game correctly ends when hitting obstacle | Passed |
| **IT05** | Test snake's movement across levels | Snake reaches a new level | New level starts with additional obstacles | New level functionality works | Passed |

**3.System Testing:**

System testing checks the overall performance of the Snake game, ensuring that all modules, such as the game mechanics, graphics, sound, and user interface, work together as intended.

| **Test Case ID** | **Test Description** | **Input** | **Expected Output** | **Actual Output** | **Status** |
| --- | --- | --- | --- | --- | --- |
| **ST01** | Verify game start and snake movement | Start the game and move the snake | Snake moves correctly on the screen | Snake movement works as expected | Passed |
| **ST02** | Test snake length increase after eating fruit | Snake eats multiple fruits | Snake length increases as expected | Length increase works correctly | Passed |
| **ST03** | Validate collision detection with walls | Snake moves into wall | Game ends or resets | Collision detection with wall is accurate | Passed |
| **ST04** | Verify game over after snake collision | Snake collides with its own body | Game ends | Collision with itself triggers game over | Passed |
| **ST05** | Test obstacle generation in later levels | Snake reaches higher levels | Obstacles appear and affect game play | Obstacles appear correctly and affect game play | Passed |
| **ST06** | Validate level progression and difficulty | Snake eats fruit and advances levels | Game adds obstacles and increases difficulty | Level progression and difficulty increase as expected | Passed |
| **ST07** | Test graphics rendering under normal gameplay | Run the game with smooth graphics | Snake and background render correctly | Graphics render smoothly | Passed |
| **ST08** | Validate sound effects during gameplay | Snake eats fruit or collides | Correct sound effects play | Sound effects work as expected | Passed |
| **ST09** | Test UI elements (score, level) | Play the game and eat fruits | Score and level display correctly | UI updates with correct score/level | Passed |
| **ST10** | Test game reset after game over | Game ends and is reset | Game restarts with initial snake and fruit | Reset works as expected | Passed |

**4. Validation of Results:**

Validation of results ensures that the game behaves as expected in terms of snake movement, collision detection, fruit interaction, and level progression.

| **Test Case ID** | **Test Description** | **Input** | **Expected Output** | **Actual Output** | **Status** |
| --- | --- | --- | --- | --- | --- |
| **VR01** | Validate fruit spawning and interaction | Snake eats fruit | New fruit spawns in a valid random position | Fruit spawns correctly after being eaten | Passed |
| **VR02** | Verify snake collision with obstacles | Snake collides with obstacle | Game ends | Collision detection with obstacles works correctly | Passed |
| **VR03** | Test level progression after eating fruits | Snake eats fruit | Level increases after every 5 fruits | Level progression works as expected | Passed |
| **VR04** | Validate game-over conditions after self-collision | Snake collides with itself | Game ends | Self-collision results in game over | Passed |

**IX. Conclusion**

In this project, we successfully developed a fully functional Snake game using Pygame, a popular Python library for building 2D games. The project provided valuable insights into the core concepts of game development, including the game loop, event handling, object-oriented programming, and collision detection. By implementing the game mechanics, we were able to design a dynamic environment where the snake moves, grows by eating food, and interacts with boundaries and its own body, creating the challenge and excitement characteristic of the classic game.

The development process involved creating the game's interface, handling real-time user input (such as arrow keys for controlling the snake), and implementing features like score tracking, game-over conditions, and speed progression. We also made use of Pygame’s powerful graphical capabilities to render the snake, food, and game borders effectively, ensuring a smooth visual experience for the player. Through managing game states such as "playing," "game over," and "paused," we developed a robust framework for controlling the flow of the game.

Additionally, we learned how to design modular, reusable code by utilizing functions and object-oriented principles. This helped in organizing the various components of the game, such as the snake, food, and the game window, making it easier to maintain and modify in the future. Optimizing the game for smooth performance, handling edge cases, and troubleshooting bugs also enhanced our problem-solving skills.

In conclusion, developing the Snake game with Pygame not only provided a fun and engaging way to learn about game development, but it also served as an excellent introduction to more complex programming concepts. The project allowed us to apply fundamental principles of programming, while gaining practical experience in a game development context. This knowledge lays a solid foundation for future projects, whether they involve game creation, interactive applications, or deeper dives into graphics programming. Ultimately, the success of this project demonstrates the power of Pygame as a tool for creating engaging and interactive experiences, and it opens up numerous possibilities for more advanced game development endeavors.

**X. References**

**YouTube Tutorials**

1. **Tech With Tim – "How to Make a Snake Game in Python"**

* A step-by-step tutorial on building a Snake game in Python using Pygame.
* [Watch on YouTube](https://www.youtube.com/watch?v=QfRzXx9ZCKA)

1. **freeCodeCamp – "Snake Game in Python with Pygame"**

* An in-depth guide to creating the Snake game with Pygame.
* [Watch on YouTube](https://www.youtube.com/watch?v=8F1Go0HjEyk)

1. **Clear Code – "Python Pygame Snake Game Tutorial"**

* Focuses on game logic and implementation in Pygame.
* [Watch on YouTube](https://www.youtube.com/watch?v=OGg3gUB4bp4)

**GitHub Repositories**

1. **Snake Game by Abhishek Agarrwal**

* A complete Pygame-based Snake game project.
* [Visit the GitHub Repo](https://github.com/abhishek321/snake-game-python-pygame)

1. **Python Snake Game by RoboSquad**

* A fully functional Snake game with graphics and sounds.
* [Visit the GitHub Repo](https://github.com/robo-squad/python-snake-game)

These resources will help you get started quickly with the Snake game using Pygame.