VIT BHOPAL UNIVERSITY

School of Computing Science and Engineering

Snake Game Project Documentation

Submitted by- Shreya Singh

Registration Number- 25BCY10083

Problem Statement

The aim of this project is to develop a fully functional Snake Game using Python and Pygame, with optional support for an AI agent. The player must control the snake to collect 10 food items within 30 seconds while avoiding collisions with walls and the snake's own body.

Objectives

- Build a user-playable Snake Game using Python and Pygame.
- Implement a timer mechanism for gameplay.
- Generate food dynamically on the grid.
- Display win/lose conditions based on food count and timer.
- Provide optional integration for an AI agent (search-based pathfinding).

Functional Requirements

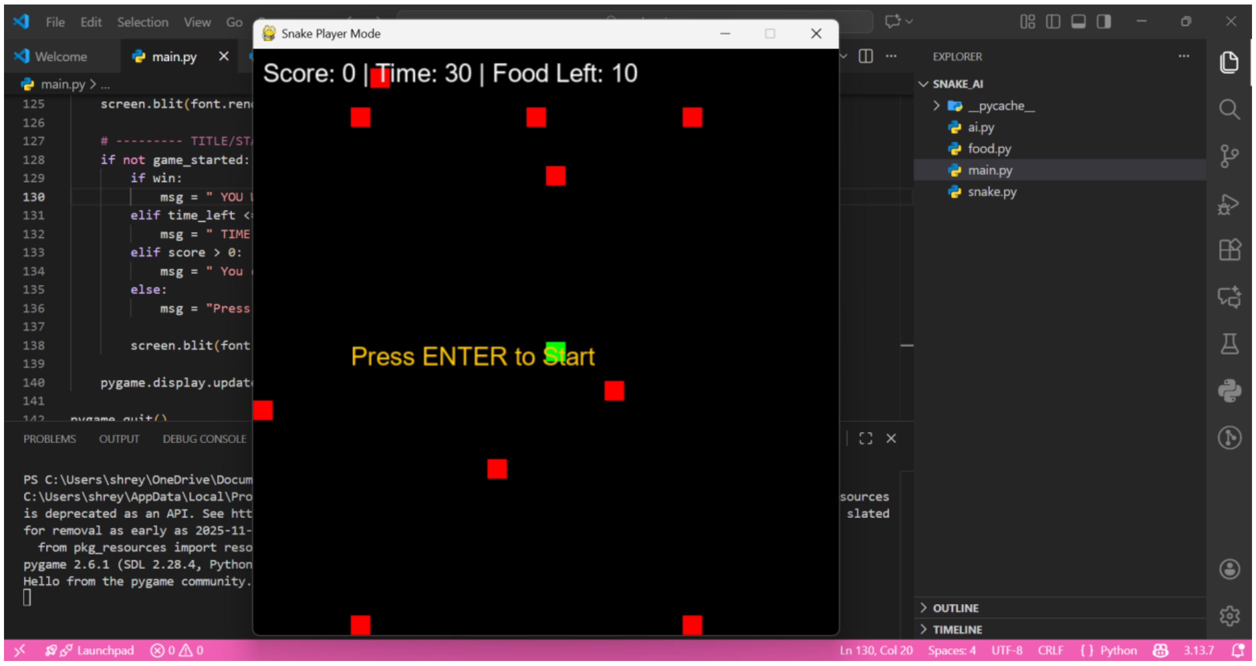
- The system shall display a playable Snake Game window.
- The system shall allow player movement using arrow keys.
- The system shall generate 10 food items per round.

- The system shall maintain a 30-second countdown.
- The system shall detect collisions with walls and the snake body.
- The system shall display win or lose states.
- The system shall allow game restart using the R key.
- Optional: The system may use an AI agent to automate snake movement.

Non-functional Requirements

- **Usability:** Controls should be intuitive and responsive.
- **Performance:** The game should run smoothly at 10 FPS.
- **Maintainability:** The code should be modular (separate Snake, Food, AI, and main logic).
- **Portability:** The project should run on any OS supporting Python and Pygame.

System Architecture Diagram



```
    screen.blit(font.ren

    # --------- TITLE/ST
    if not game_started:
        if win:
            msg = " YOU
        elif time_left <
            msg = " TIME
        elif score > 0:
            msg = " You
        else:
            msg = "Press

        screen.blit(font

    pygame.display.updat

pygame.quit()
```

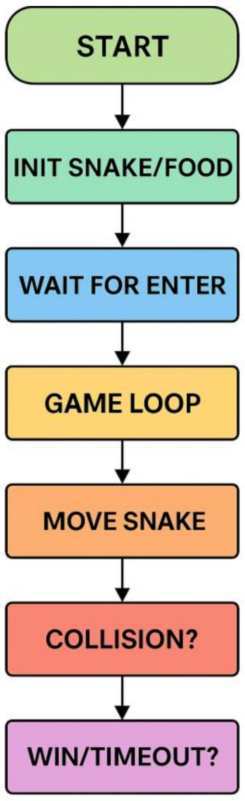
```
PS C:\Users\shrey\OneDrive\Docum
C:\Users\shrey\AppData\Local\Pro
is deprecated as an API. See htt                                    sources
for removal as early as 2025-11-                                     slated
  from pkg_resources import reso
pygame 2.6.1 (SDL 2.28.4, Python
Hello from the pygame community.
```

Description:

- `main.py` controls game flow.

- `snake.py` manages snake movement, growth, and collisions.
- `food.py` handles random food spawning.
- `ai.py` (optional) computes the next move using search strategies.
- Pygame renders window, graphics, and handles input.

Process Flow / Workflow Diagram



General Flow:

1. Initialize game and variables.
2. Start game loop.
3. Capture user input.
4. Update snake position.
5. Check collisions.
6. Check food consumption.
7. Update timer.

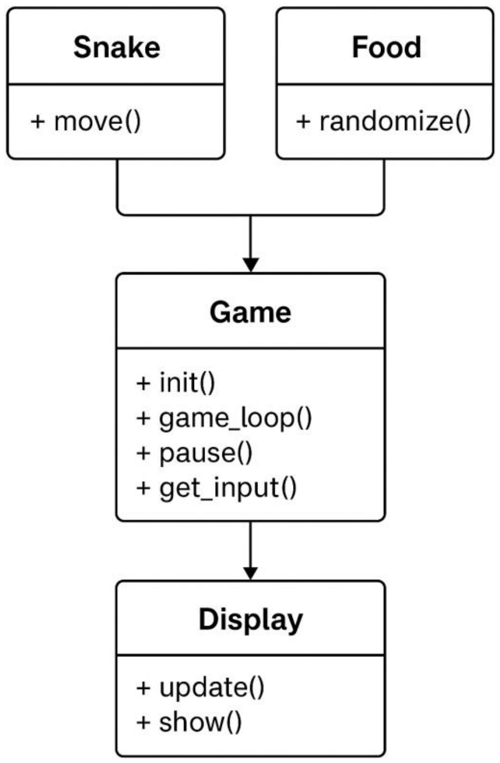
8. Display game state.
9. End game if win/lose conditions met.

UML Diagrams

Use Case Diagram

```
+---------------------------+
|        User              |
+------------+--------------+
             |
             | Plays game
             v
+---------------------------+
|       Snake Game         |
|---------------------------|
| - Start Game             |
| - Take Input (W/A/S/D)   |
| - Move Snake             |
| - Detect Collision       |
| - Generate Food          |
| - Update Score           |
+---------------------------+
```

Class Diagram

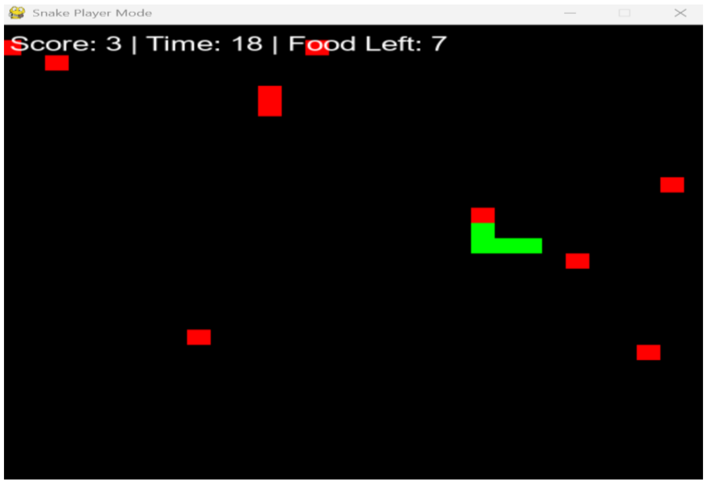


Sequence of the game

```
User -> Game : Start()
Game -> Snake : Initialize snake
Game -> Food : Spawn food
Loop Every Frame
    User -> Game : Input Direction
    Game -> Snake : Move()
    Snake -> Game : Collision status
    Game -> Food : Check eaten?
    Food -> Game : Respawn if needed
    Game -> User : Render frame
End Loop
```

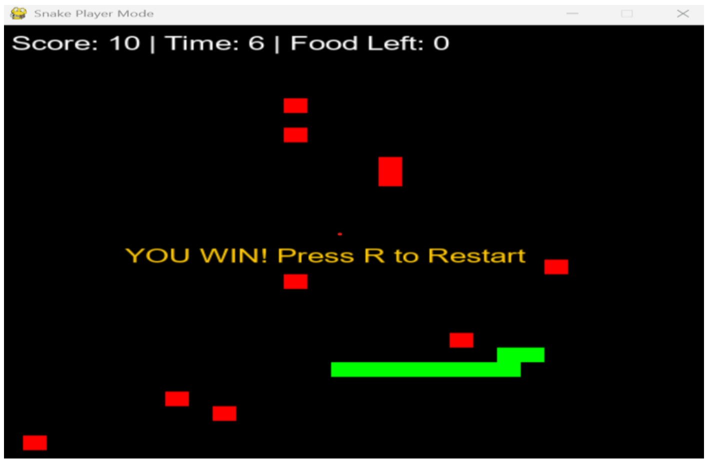
Gameplay Images

Running game-



Score: 3 | Time: 18 | Food Left: 7

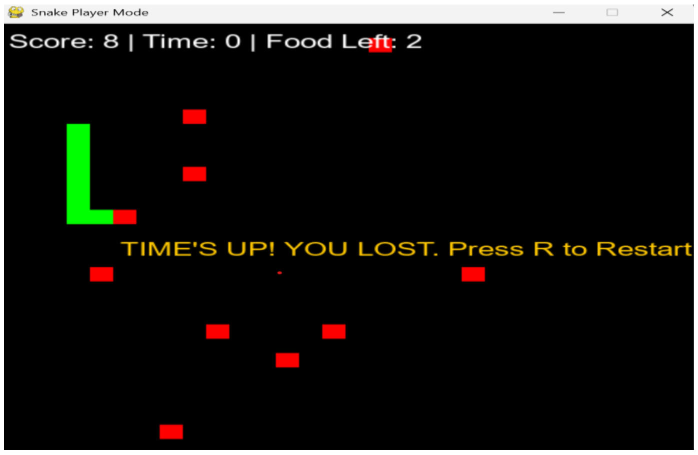
Winning the game-



Score: 10 | Time: 6 | Food Left: 0

YOU WIN! Press R to Restart

Losing the game-

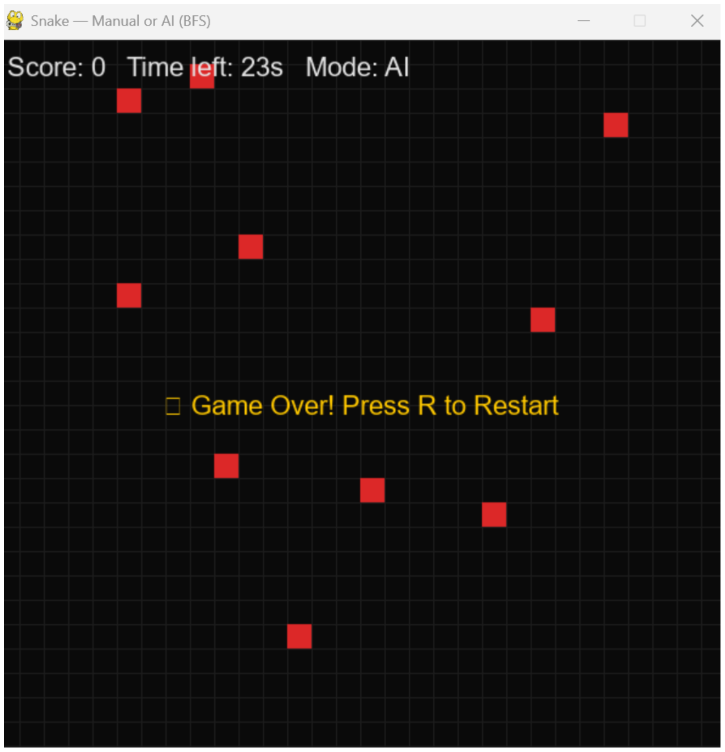


Score: 8 | Time: 0 | Food Left: 2

TIME'S UP! YOU LOST. Press R to Restart

AI Agent Search Strategy Explanation

The AI agent explores the grid using:

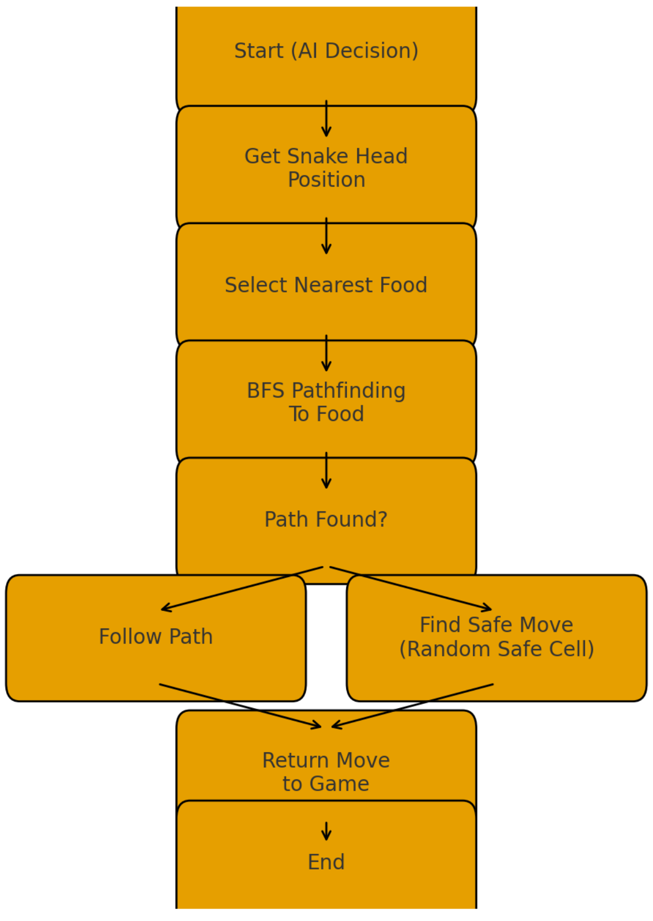

Snake — Manual or AI (BFS)
Score: 0 Time left: 23s Mode: AI
Game Over! Press R to Restart

1. Breadth-First Search (BFS)
- Guarantees shortest path to food.
- Explores all neighbors level by level.
- Works well in open areas.

2. Depth-First Search (DFS)
- Explores deep paths first.
- Less optimal for shortest path.
- Can be used for exploration-heavy modes.

3. A* Search
- Uses heuristics like Manhattan Distance.
- Optimal and fast.
- Best suited for grid environments.

Flowchart of AI Agent



Conclusion

This Snake Game project successfully demonstrates the design and development of an interactive gameplay system using Python and Pygame. The game integrates essential software engineering components such as modular code structure, object-oriented design, UML diagrams, workflow processes, and system architecture planning. Through the implementation of a timed challenge mode, multiple food generation, and smooth gameplay mechanics, the project showcases effective problem-solving and algorithmic thinking.

The documentation provides a complete overview of requirements, design models, and logical flow, making the system easy to understand, maintain, and extend. Overall, the project fulfills its objectives of creating a functional, user-controlled Snake game while adhering to structured software engineering principles.