Project Report

Instructor: Shafique Rehman

Project Title: Classic Nokia Snake Game

Project Team:

Jawwad Shahzad (21k3461)
Haris Munshi (21k3357)

Introduction:

The Snake Game project was undertaken as part of our coursework under the guidance of our instructor, Shafique Rehman. The goal was to create a functional and engaging version of the classic Snake Game using Python and the Pygame library. This report provides a detailed overview of the project, including implementation details, challenges faced, and potential future improvements.

Project Overview:

The Snake Game involves controlling a snake that moves around a grid, eating apples that appear randomly. Each apple eaten makes the snake longer. The game ends if the snake collides with itself or the grid boundaries. The project was implemented using Python and Pygame for graphical rendering.

Implementation Details:

Code Structure

The project consists of three main Python files:

> settings.py: Contains configuration settings for the game, such as dimensions, colors, game settings, and helper functions.
> snake.py: Defines the Snake and Square classes, responsible for the snake's movement, apple generation, collision detection, and pathfinding algorithms.
> play.py: Initializes the game environment, handles user input, and updates the game state.

Settings and Configurations

In settings.py, various configurations are defined:

> Dimensions: The game surface is set to 1000x1000 pixels with a grid of 50 rows. Each square in the grid is 20x20 pixels with a 2-pixel gap.
> Colors: Various colors are defined for the game elements, including the snake, apple, grid, and background.
> Game Settings: Frame rate (FPS), initial snake length, waiting time after winning, and maximum moves without eating are set.

Snake and Square Classes

In snake.py, the Square class represents each segment of the snake and the apple. The Snake class manages the snake's behavior, including movement, growth, collision detection, and pathfinding.

Square Class:

> Initialization: Sets the position, surface, and direction of the square.
Draw Method: Renders the square on the game surface.
> Move Method: Updates the square's position based on its direction.
> Collision Detection: Checks if the square is hitting the wall.

Snake Class:

> Initialization: Sets up the initial state of the snake, including its starting position, direction, and apple generation.
> Draw Method: Renders the snake and apple on the game surface.
> Movement and Direction: Updates the snake's position and handles direction changes based on user input.
> Collision Detection: Checks if the snake collides with itself or the grid boundaries.
> Pathfinding (BFS): Implements the Breadth First Search algorithm to find the shortest path to the apple.

Game Loop and Event Handling

> In play.py, the game loop is managed, which includes initializing Pygame, setting up the game window, and continuously updating the game state.

The game loop handles:

> Screen Drawing: Fills the background and draws the grid.
> Event Handling: Captures user input to control the snake's direction.
> Game State Update: Updates the snake's state, including movement and collision detection..

Challenges Faced:

Enhanced Graphics

Adding animations, sound effects, and better visual feedback can enhance the gaming experience. Smooth transitions and more engaging graphics will make the game more enjoyable for players.

Additional Features

Introducing power-ups, multiple levels, or different game modes can add depth to the gameplay. For example, adding obstacles that the snake must avoid or special items that grant temporary abilities can make the game more challenging and interesting.

Improved AI and Pathfinding

Enhancing the pathfinding algorithm to handle more complex scenarios and adding AI-controlled snakes that the player can compete against can increase the game's complexity and replay value. Implementing more sophisticated AI behavior will make the game more challenging for players.

User Interface

Creating a more polished user interface with menus, scoreboards, and settings can improve the overall user experience. Allowing players to customize game settings, such as grid size and snake speed, can make the game more accessible and enjoyable.

Conclusion:

In conclusion, the Snake Game project was a valuable learning experience that allowed us to apply our programming skills to develop a classic arcade game.