Snake Game - Design Document (CSC221 Final Project)

Author: Drexel Zybko

Date: April 20, 2025

Course: CSC221 - Introduction to Problem Solving and Programming

Project Title: Classic Snake Game Using PyGame

# 1. Project Overview

This project is a modern recreation of the classic Snake game using Python and the PyGame library. The objective is to build a game where the player controls a snake that moves around a grid-based screen, consumes randomly appearing food, and grows in length. The game ends when the snake collides with the boundaries of the screen or with itself.

The purpose of this project is to demonstrate proficiency in:  
- Designing and implementing a program using Python  
- Applying key concepts from the course including control structures, loops, functions, and libraries  
- Using PyGame for event handling and rendering  
- Using GitHub for code collaboration and version control

# 2. Game Mechanics and Features

• Grid Size: The play area will be a 600x600 pixel window, divided into 20x20 pixel tiles (30x30 grid)

• Snake Movement: The snake moves continuously in one of four directions: up, down, left, right

• User Input: Direction is controlled using arrow keys

• Food Generation: Food appears randomly and respawns after being eaten

• Growth Mechanism: Snake grows one segment each time it eats food

• Collision Detection: Game ends when the snake hits the wall or its own body

• Score Tracking: Score increases by 1 for each food item eaten and is displayed on screen

• Restart Option: Game can be restarted after game over by pressing a key (e.g., R key)

# 3. Tools & Libraries

- pygame: For window management, drawing shapes, handling input, and game loop

- random: To randomly place food items

# 4. Program Structure

The game will be built using a modular structure. Each core component of the game will be implemented as a function or class to improve clarity and reusability.

• main(): Initializes PyGame and runs the main game loop

• draw\_snake(surface, body): Renders the snake body on the screen

• move\_snake(direction, body): Updates the snake's position based on the direction input

• generate\_food(snake): Returns a new food location that does not overlap the snake

• check\_collision(snake): Determines if the snake has hit the wall or itself

• display\_score(score): Displays current score on the game window

• restart\_game(): Resets the game state after game over

# 5. Game Loop Overview

The game loop will execute approximately 10–15 times per second. Each cycle will follow this sequence:  
1. Event Handling: Detect key presses (arrow keys or R for restart)  
2. Update Logic: Move the snake, check collisions, update score, generate food  
3. Render Frame: Clear the screen, draw food and snake, update display  
4. Delay for Frame Rate: Use pygame.time.Clock().tick(10)

# 6. Algorithmic Concepts (from Chapter 5)

- Control Structures: if, else, elif for handling movement and collision outcomes  
- Loops: while loop for the game loop, for loop for drawing each segment of the snake  
- Lists: Used to store coordinates of each segment of the snake  
- Tuples: Used to represent 2D coordinates (x, y)  
- Functions: Decomposed game logic into modular, testable components  
- Event Handling: pygame.KEYDOWN events for movement and restarting the game

# 7. Data Structures

snake = [(x1, y1), (x2, y2), ...] # List of tuples representing snake body parts  
direction = 'RIGHT' # String representing current movement direction  
food\_pos = (x, y) # Tuple for food coordinates  
score = 0 # Integer score value

# 8. Expected Challenges & Solutions

• Snake collision with itself: Check if head coordinate overlaps with any other segment in the body

• Snake hitting the wall: Compare snake head position with screen boundaries

• Smooth directional change: Prevent reversing direction directly (e.g., can't go LEFT after RIGHT)

• Clean restart: Clear snake list, reset variables, and reinitialize food and direction

# 9. Development Timeline

• Apr 14–20: Set up window, movement, and food logic

• Apr 21–27: Collision detection, scoring, game over

• Apr 28–May 5: Add restart feature, polish visuals

• May 6: Final testing, upload to GitHub, rehearse presentation

# 10. GitHub Repository

[GitHub link here] – Will be added after uploading files

Repository will contain:  
- All Python files needed for the game  
- This Design Document  
- README with instructions to run the game