Final Project Design Document
By Drexel Zybko for CSC221 Final Project

Introduction

I created a classic Snake game using Python and the Pygame library. In this game, I control a snake that moves around a 2D grid, trying to eat food that appears randomly on the screen. Every time the snake eats, it grows longer and my score increases. The game ends if the snake runs into itself or the wall. I added beautiful background music to make the experience more immersive, and included a startling and funny sound effect that really cuts into the serenity of the gameplay.

Project Functionality

- The snake moves in a grid of 25x25 tiles.
- Movement is controlled using the arrow keys.
- Food appears in a random spot on the grid that is not already occupied by the snake.
- Eating food increases the length of the snake and the score by 1.
- If the snake runs into itself or the edge of the grid, the game ends.
- Background music plays during the game.
- A separate sound plays when the player loses.
- The score is displayed in real-time, and a custom game over screen shows the final score.

Design decisions included:

- Making the snake only grow when it eats food with its head (not other body parts).
- Food never spawns inside the snake's body.
- The game does not go on forever—it ends on collision.

- A silly game message appears at the end along with a funny sound effect.

Design Process

I started by setting up the basic grid and the snake. Once movement was working, I added food placement and collision detection. After that, I implemented score tracking and the game over logic. One of the biggest challenges was making sure the snake couldn't turn directly back into itself, which I fixed by checking the opposite direction. I also added music to make it more engaging and had fun testing different sounds. One hiccup was getting `.mp3` files to work properly. I had to make sure the mixer was initialized correctly and that the files were in the right format. The project was fun and helped me get more comfortable with game logic and event handling in Python.

Project Development

This project was developed in Python using the Pygame library. It was run and tested locally in a Python environment with the necessary music and sound files included in the same directory. I followed a simple but iterative approach: get one thing working, test it, and build from there. I also used `pygame.mixer` to add music and sound effects.

Pseudocode

```
Initialize Pygame
Set up screen, colors, constants
Load background music and game over sound

Define a function to draw text on screen
Define a function to place food randomly (not inside the snake)

Start main game loop:
    - Handle player input for direction
    - Move snake in current direction
    - Check for collisions with self or wall
    - If collision: play game over sound, return score
    - If food eaten: increase score, grow snake, place new food
    - Else: move snake by removing tail
    - Draw snake, food, and score on screen
```

After game ends:
- Show game over message and final score
- Wait for player to press R (restart) or Q (quit)

Flowchart

```
[Start]
  ↓
[Initialize Game]
  ↓
[Enter Game Loop]
  ↓
[Handle Input]
  ↓
[Update Snake Position]
  ↓
[Check for Collision] -- Yes --> [Play Game Over Sound] --> [Game Over Screen] -->
[Restart/Quit]
  |
  No
  |
[Check if Food is Eaten] -- Yes --> [Grow Snake and Add Score]
  |
  No
  |
[Move Snake Normally]
  ↓
[Draw Everything]
  ↓
[Loop Back to Handle Input]
```

UML Diagram

(Since this project is procedural and not object-oriented, a traditional UML class diagram isn't necessary. If desired, I could show a simple diagram showing function relationships.)

Requirements

- Includes user interaction (e.g., command-line inputs, GUI, keyboard movement, etc.).

• Performs some processing (e.g., conditional logic, loops, function calls, math operations).

• Generates cleanly formatted output based on input/processing.

• Implements at least one custom function/method/module.

• Use an imported Python library.

Key Features and Functional Requirements:

1. Board Size / Play Area:

• The game should be played on a window containing at least a 10-by-10 tile set.

2. Snake Movement:

• The snake should move continuously in one of four directions (up, down, left, right).

• The player should control the direction using the arrow keys.

3. Snake Growth:

• Every time the snake eats a food item, it should grow longer by one unit.

4. Food Generation:

• Food should appear randomly on the game screen after being eaten by the snake.

5. Collision Detection:

• The game should end if the snake collides with:

o The walls of the screen.

o Its own body.

6. Game Over and Score:

• Display a Game Over message when the snake dies.

• Display the player's score, which is based on how many food items the

snake has eaten.

7. Restart Option:

• There should be an option to replay the game after a game over.