Final Project
Design Document

By Leo Politis for CSC221 Final Project

Introduction

Project Functionality

The game I am creating using Python is a classic Snake Game. The game is simple: you are a snake and collect "food" items to grow bigger. The snake can only move in a straight line, not diagonal. Food will spawn randomly subsequently to being collected. Only the head of the snake can consume the food. If the snake runs into the wall or into itself, the game will end. The game technically cannot run forever. This is because once you get to a certain score, the snake will get too long and will eventually run into itself. The player will use arrow keys (and WASD) to move across the grid. If the player loses, and wants to restart, they should press the spacebar or the R key.

Design Process

For the design, I went with a basic snake game design. I made the snake green, and the "food" red to represent apples. I originally intended to make the background black, but I wanted to make it look nicer. I ended up making a tile background with alternating colors of green. The reason why I did this is because I'm a little new to Python and wanted to make the game uncomplicated and like the original snake game. I also wanted the game to be played with WASD in addition to arrow keys, so I added that as well. To restart the game after dying, the user has to press R or the Spacebar. I also didn't like how everything was blocky, so I made the "apples" rounded, and the snake pixels slightly round to differentiate them when they are connected to each other.

I had trouble getting the game to be fullscreen or resizable. I wanted to make the window fullscreen or resizable because it was only a 10x10 grid and appears on the screen very small. What I ended up doing was increasing the GRID_SIZE to 20, so it is more user-friendly and is bigger. Another thing that needed to be improved was the input response and the fact that some inputs were not detected, which made a noticeable difference in the gameplay. To solve this, I used else-if statements to handle the inputs and make the movement feel less delayed. Even after changing this, there were still inputs going unnoticed/unaccepted. After some research, I came to the conclusion that it was a problem with the clock.tick because the FPS was too low to notice quick inputs. Although, changing this made a slight difference. When trying to implement a restart feature, the game was accepting any keypress to restart the game which can be annoying for the user/player. After a little bit of tweaking, I had to separate the input handling from the restart statement, write an if statement regarding the R key, and write an identical else-if statement for the Spacebar key.

The parts I enjoyed while making my final project is applying the changes to the code and seeing it in action. After many error messages, watching the game start up correctly was very satisfying. Playing the game myself was also pretty fun and worthwhile.

Project Development

Pseudocode

```
Start game

Initialize window and grid size (at 20x20, cell size=50)
Initialize snake at center
Initialize direction to none
Initialize food at a random location that is not on the snake
Initialize score to equal 0
Set game_over to false
```

```
Define block drawing
Define game_over screen
Define on-screen score
Define background

WHILE game is running AND not game_over:
  user input
  FOR each event in event queue:
          IF event is QUIT:
                  Set running to false
                  Exit game
          IF event is KEYDOWN:
                  IF NOT game_over:
                          IF UP or W is pressed and direction is NOT DOWN:
                          Set direction to UP
                          ELSE IF DOWN or S is pressed and direction is NOT UP:
                          Set direction to DOWN
                          ELSE IF LEFT or A is pressed and direction is NOT RIGHT:
                          Set direction to LEFT
                          ELSE IF RIGHT or D is pressed and direction is NOT LEFT:
                          Set direction to RIGHT
                  ELSE:
                          IF R is pressed:
                                  Snake goes back to center
                                  Direction is none
                                  Food is at random location NOT on snake
                                  Score = 0
                                  Game_over is set to false
                  ELSE-IF:
                          IF SPACE is pressed:
                                  Snake goes back to center
                                  Direction is none
                                  Food is at random location NOT on snake
                                  Score = 0
                                  Game_over is set to false
```

```
IF NOT game_over:
    IF UP key pressed and NOT moving DOWN:
        Set direction to UP
    IF DOWN key pressed and NOT moving UP:
        Set direction to DOWN
    IF LEFT key pressed and NOT moving RIGHT:
        Set direction to LEFT
    IF RIGHT key pressed and NOT moving LEFT:
        Set direction to RIGHT
    IF W key pressed and NOT moving DOWN:
        Set direction to UP
    IF S key pressed and NOT moving UP:
        Set direction to DOWN
    IF A key pressed and NOT moving RIGHT:
        Set direction to LEFT
    IF D key pressed and NOT moving LEFT:
        Set direction to RIGHT

IF direction is NOT none:
    Calculate new head position based on direction

Check for collision:
    IF new head is outside grid OR new head is in snake body:
        Set game_over to true

    ELSE: insert new head at beginning of snake

IF new head is on food:
    Increase score by 1
    Place new food at random location not on the snake

ELSE: remove last pixel of snake (seen as pop tail/snake pop)

Fill screen with black
Draw background
Draw snake segments
Draw food blocks
Draw score on screen

IF game_over is true:
        Display Game Over Message
        Display final score
```

Flowchart

Text flowchart:
Start
|
Initialize variables (such as the snake, food, color, and movement direction)
|
Define functions (such as drawing blocks, game_over, and score)
|
Start game loop
|
Handle player input (from arrow keys/WASD) - snake direction will be updated
|
Movement of snake - calculate new head position
|
Check collision - game will end if snake collides with wall or itself
|
Place new head & check if food is eaten - food spawns randomly, away from snake
|
Snake growth - when food is eaten, snake grows, as the tail doesn’t pop
|
Draw entire game - (fills colors for objects, draws score, ect)
|
Game over - shows game over message

Requirements

- [x] ~~Board Size / Play Area~~
  Has to be at least 10x10 grid. **Made a 20x20 grid**
- [x] ~~Snake Movement~~
  Should go up, down, left or right. **Uses arrow keys + WASD to move**
- [x] ~~Snake Growth~~
  Should grow by one unit for every point. **Inserts new head and snake tail doesn't pop, hence making the snake grow**
- [x] ~~Food Generation~~
  Food should randomly spawn. **Food generates randomly after being eaten**
- [x] ~~Collision Detection~~
  Game should end if the snake collides with anything. **Game will end if snake collides with itself or goes outside the grid**
- [x] ~~Game Over and Score~~
  Should be a game over screen with score. **Game over screen shows, and displays final score. Added live score keeper in the top left corner.**
- [x] ~~Restart Option~~
  Should be a restart option after dying. **Game is available to restart after dying, and pressing Spacebar or R will prompt it to restart.**