Python Snake Game

Muhammed Hanan

Pygame

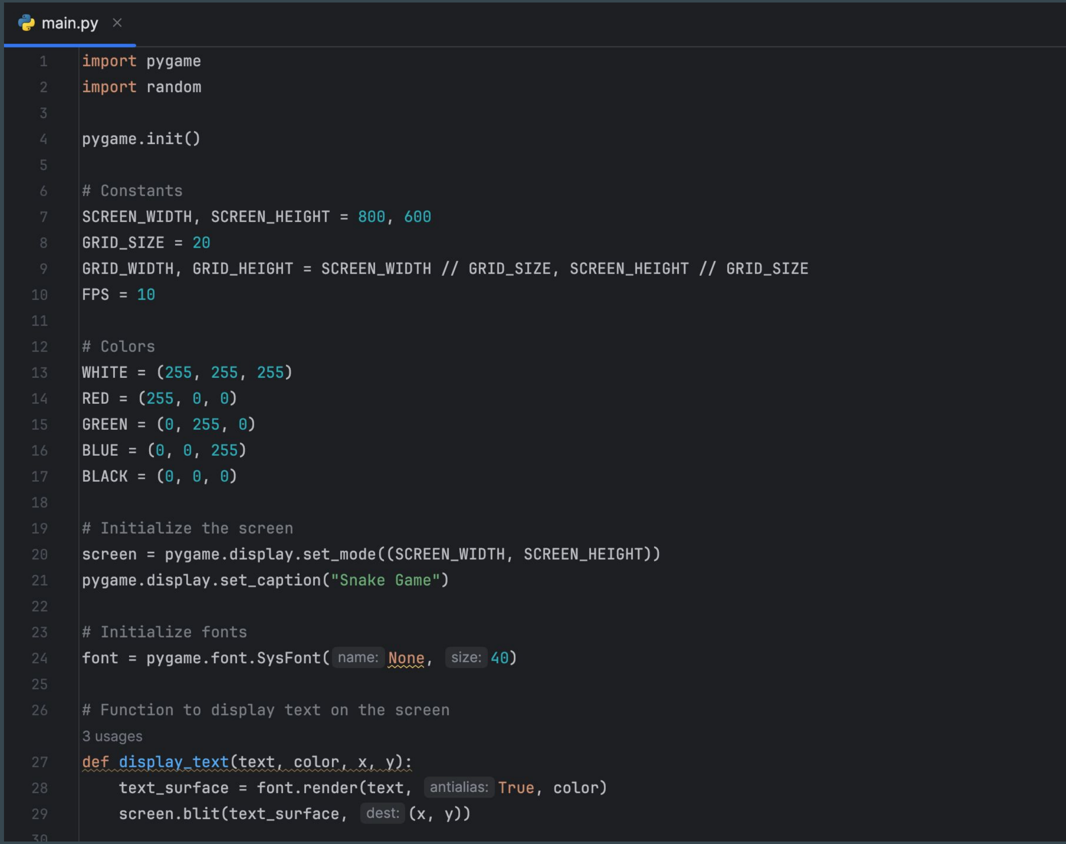
- Pygame is a well-known Python library that is specially created for the purpose of writing video games. This library comes equipped with various features, which facilitate the process of game development.
- Pygame provides developers with the capability integrate graphics, sound, and user input functionality into their games.
- Also used for other multimedia applications.

PyCharm

- The application used to make the snake game is Pycharm.
- PyCharm has been created specifically for the purpose of Python development.
- Has a handful of other coding languages and useful features such as JS and interactive debugging tools.

Game Setup

- Initialize Pygame for game development and create constants such as the screen size and FPS
- Initialize the fonts and create a function to display the text

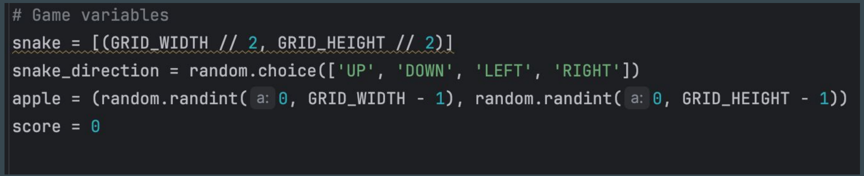


main.py

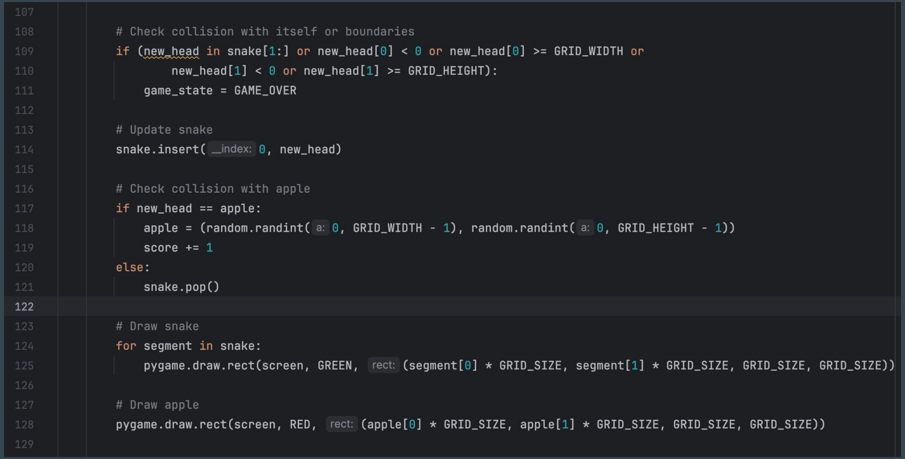
```python
import pygame
import random

pygame.init()

# Constants
SCREEN_WIDTH, SCREEN_HEIGHT = 800, 600
GRID_SIZE = 20
GRID_WIDTH, GRID_HEIGHT = SCREEN_WIDTH // GRID_SIZE, SCREEN_HEIGHT // GRID_SIZE
FPS = 10

# Colors
WHITE = (255, 255, 255)
RED = (255, 0, 0)
GREEN = (0, 255, 0)
BLUE = (0, 0, 255)
BLACK = (0, 0, 0)

# Initialize the screen
screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
pygame.display.set_caption("Snake Game")

# Initialize fonts
font = pygame.font.SysFont( name: None,  size: 40)

# Function to display text on the screen
3 usages
def display_text(text, color, x, y):
    text_surface = font.render(text,  antialias: True, color)
    screen.blit(text_surface,  dest: (x, y))
```

Snake and Apple

- First declare snake, apple and score variables
- The snake is a list of x, y coordinates on the grid.
- The initial position of the snake is set at the center of the grid (GRID_WIDTH/ 2, GRID_HEIGHT/ 2).
- The snake's movement is determined by snake_direction, which can be up, down, left, or right.
- The snake's head is updated according to its direction, and its body grows as it moves and consumes apples.
- The apple is a randomly generated x, y coordinates on the grid.
- When the snake head's position matches the apple's position, the apple is eaten.
- If snake collides with the grid or itself than the game is over



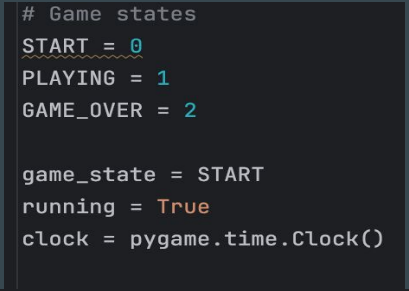
```
# Game variables
snake = [(GRID_WIDTH // 2, GRID_HEIGHT // 2)]
snake_direction = random.choice(['UP', 'DOWN', 'LEFT', 'RIGHT'])
apple = (random.randint( a: 0, GRID_WIDTH - 1), random.randint( a: 0, GRID_HEIGHT - 1))
score = 0
```



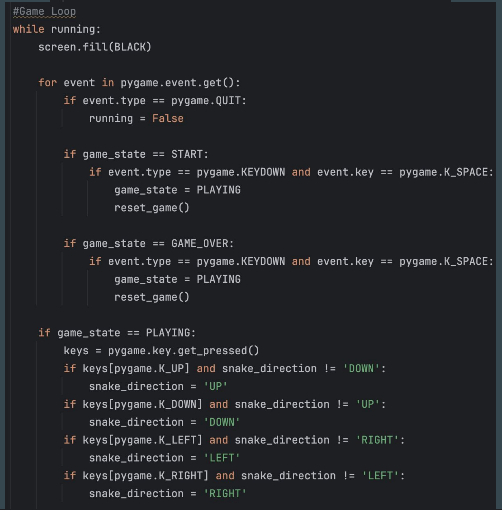
```

# Check collision with itself or boundaries
if (new_head in snake[1:] or new_head[0] < 0 or new_head[0] >= GRID_WIDTH or
        new_head[1] < 0 or new_head[1] >= GRID_HEIGHT):
    game_state = GAME_OVER

# Update snake
snake.insert( __index: 0, new_head)

# Check collision with apple
if new_head == apple:
    apple = (random.randint( a: 0, GRID_WIDTH - 1), random.randint( a: 0, GRID_HEIGHT - 1))
    score += 1
else:
    snake.pop()

# Draw snake
for segment in snake:
    pygame.draw.rect(screen, GREEN,  rect: (segment[0] * GRID_SIZE, segment[1] * GRID_SIZE, GRID_SIZE, GRID_SIZE))

# Draw apple
pygame.draw.rect(screen, RED,  rect: (apple[0] * GRID_SIZE, apple[1] * GRID_SIZE, GRID_SIZE, GRID_SIZE))

```

Game States

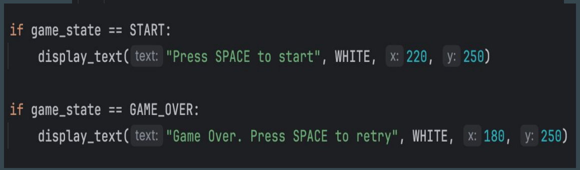
- The 3 game states are start, playing and game over and the clock handles the FPS of the game
- While running is true, the loop starts, which handles all the events of the game, from when start to end.
- In the loop, the snake's controls are assigned as well.
- Lastly, the menu options to start the game which starts the loop and retry transitions back to playing by calling reset_game to restart the game.



```python
# Game states
START = 0
PLAYING = 1
GAME_OVER = 2

game_state = START
running = True
clock = pygame.time.Clock()
```



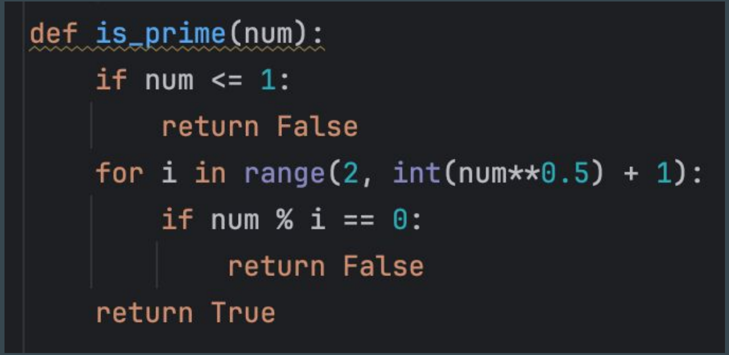
```python
#Game Loop
while running:
    screen.fill(BLACK)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

        if game_state == START:
            if event.type == pygame.KEYDOWN and event.key == pygame.K_SPACE:
                game_state = PLAYING
                reset_game()

        if game_state == GAME_OVER:
            if event.type == pygame.KEYDOWN and event.key == pygame.K_SPACE:
                game_state = PLAYING
                reset_game()

    if game_state == PLAYING:
        keys = pygame.key.get_pressed()
        if keys[pygame.K_UP] and snake_direction != 'DOWN':
            snake_direction = 'UP'
        if keys[pygame.K_DOWN] and snake_direction != 'UP':
            snake_direction = 'DOWN'
        if keys[pygame.K_LEFT] and snake_direction != 'RIGHT':
            snake_direction = 'LEFT'
        if keys[pygame.K_RIGHT] and snake_direction != 'LEFT':
            snake_direction = 'RIGHT'
```



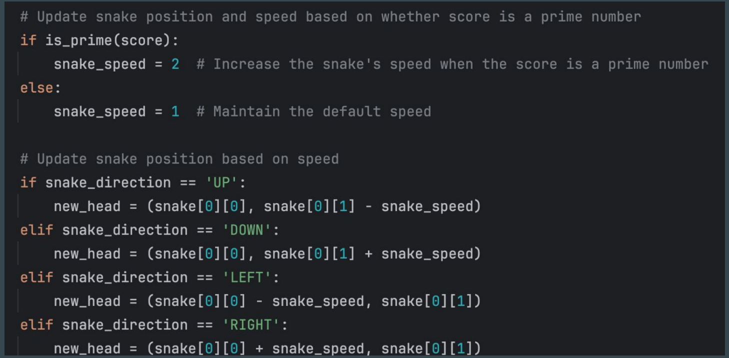
```python
if game_state == START:
    display_text( text: "Press SPACE to start", WHITE,  x: 220,  y: 250)

if game_state == GAME_OVER:
    display_text( text: "Game Over. Press SPACE to retry", WHITE,  x: 180,  y: 250)
```

Number Theory Applications

- Prime numbers are natural numbers greater than 1 that have only two divisors: 1 and the number itself and are the building blocks of integers.
- The first function validates if a number is prime.
- Second function checks if the score is prime and based on that adjusts the speed of the snake in the game
- Last function determines new head position based on snake direction.
- Adjusts new head coordinates according to the current direction and snake speed.



```python
def is_prime(num):
    if num <= 1:
        return False
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False
    return True
```



```python
# Update snake position and speed based on whether score is a prime number
if is_prime(score):
    snake_speed = 2  # Increase the snake's speed when the score is a prime number
else:
    snake_speed = 1  # Maintain the default speed

# Update snake position based on speed
if snake_direction == 'UP':
    new_head = (snake[0][0], snake[0][1] - snake_speed)
elif snake_direction == 'DOWN':
    new_head = (snake[0][0], snake[0][1] + snake_speed)
elif snake_direction == 'LEFT':
    new_head = (snake[0][0] - snake_speed, snake[0][1])
elif snake_direction == 'RIGHT':
    new_head = (snake[0][0] + snake_speed, snake[0][1])
```

Improvements and Final Thoughts

- Learning and implementing the foundations of pygame while also implementing number theory concepts was definitely a unique challenge.
- Possible improvements for this project would be to implement it with 3d graphics, possibly with the Ursina Engine, which is another game development library for python but has more support for 3d graphical rendering or other libraries such as PyOpenGL or Panda3D.
- Another possible improvement could be to add textures and sprites for the snake, apple and background