Activity based

Project Report on

Artificial Intelligence

MINI PROJECT

Submitted to Vishwakarma University, Pune

By

Ojas J Khawas

SRN No : 202100264

Roll No : 10

Div : C

Third Year Engineering

Faculty In charge :- Prof N.Z.Tarapore

Date Of Project :24/04/2024

Department of Computer Engineering

Faculty of Science and Technology

Academic Year

2023-2024 Term-II

Title: Detailed Report on 2-Player Snake Game

1. Introduction:

- The 2-Player Snake Game is a classic arcade-style game implemented using Python and the Pygame library.
- The game allows two players to control separate snakes on the same screen and compete to eat as much food as possible while avoiding collisions with walls and each other.

2. Implementation:

- The game is implemented using object-oriented programming principles in Python.
- Key components of the implementation include:
  - **Snake Class:** Represents a snake in the game. It handles the movement, growth, and rendering of the snake.
  - **Food Class:** Represents food items that snakes can eat to grow. It handles the spawning and rendering of food.
  - **Display Functions:** Functions to display scores and the winner on the screen.
  - **Main Function:** The main game loop where user input, game logic, and rendering are handled.
- Pygame library is utilized for screen management, event handling, and rendering graphics.

3. Game Rules:

- Each player controls a snake using arrow keys (Player 1) and W, A, S, D keys (Player 2).
- Snakes move continuously in their respective directions until a key press changes their direction.
- Snakes grow in length upon eating food, which appears at random locations on the screen.
- The game ends when a snake collides with the wall or itself.
- The player with the highest score (number of foods eaten) wins the game.

4. Features:

- **Two-Player Mode:** Allows two players to compete against each other simultaneously.
- **Scoring System:** Keeps track of the number of foods eaten by each player.
- **Collision Detection:** Detects collisions between snakes, walls, and snake bodies.
- **Graphics and Animation:** Utilizes Pygame to render colorful graphics and provide smooth animations.
- **Winner Announcement:** Displays the winner on the screen for a certain duration after the game ends.

Algorithm of the Code:

- **Initialization:**
    1. Initialize the Pygame library and set up the game screen with the specified dimensions.
    2. Define constants such as block size, frame rate, and colors.
    3. Create instances of the Snake and Food classes for each player.
- **Main Game Loop:**
    1. Enter the main game loop to continuously update the game state and render graphics.
    2. Handle user input to change the direction of each snake when arrow keys or WASD keys are pressed.
    3. Move each snake in its current direction while ensuring it stays within the bounds of the screen.
    4. Check for collisions between each snake and the walls or itself. If a collision occurs, end the game.
    5. Check if each snake has eaten food. If so, increase its score and respawn the food at a random location.
    6. Draw each snake, the food, and display the scores on the screen.
    7. If the game is over, display the winner for 5 seconds before exiting the game loop.
- **Snake Class:**
    1. Initialize each snake with a color and initial position.
    2. Move the snake's head in the current direction and adjust its position to stay within the screen bounds.
    3. Allow the snake to grow by adding a new segment to its body when it eats food.
- **Food Class:**
    1. Initialize food items with a random position within the screen bounds.
    2. Respawn food at a new random location when a snake eats it.
- **Display Functions:**
    1. Render text on the screen to display the scores of each player.
    2. Display the winner on the screen for a specified duration after the game ends.
- **Event Handling:**
    1. Handle quit events to exit the game loop when the user closes the window.
    2. Handle key press events to change the direction of each snake.

Code:

```
import pygame
import random
import time

# Initialize pygame
pygame.init()

# Set up the screen
SCREEN_WIDTH = 600
SCREEN_HEIGHT = 400
BLOCK_SIZE = 20
FPS = 10

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

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

# Snake class
class Snake:
    def __init__(self, color, initial_pos):
        self.color = color
        self.body = [initial_pos]
        self.direction = "RIGHT"

    def move(self):
        head = self.body[0].copy()
        if self.direction == "RIGHT":
            head[0] += BLOCK_SIZE
        elif self.direction == "LEFT":
            head[0] -= BLOCK_SIZE
        elif self.direction == "UP":
            head[1] -= BLOCK_SIZE
        elif self.direction == "DOWN":
            head[1] += BLOCK_SIZE

        # Adjust head position if out of bounds
        head[0] = max(min(head[0], SCREEN_WIDTH - BLOCK_SIZE), 0)
        head[1] = max(min(head[1], SCREEN_HEIGHT - BLOCK_SIZE), 0)

        self.body.insert(0, head)
        self.body.pop()
```

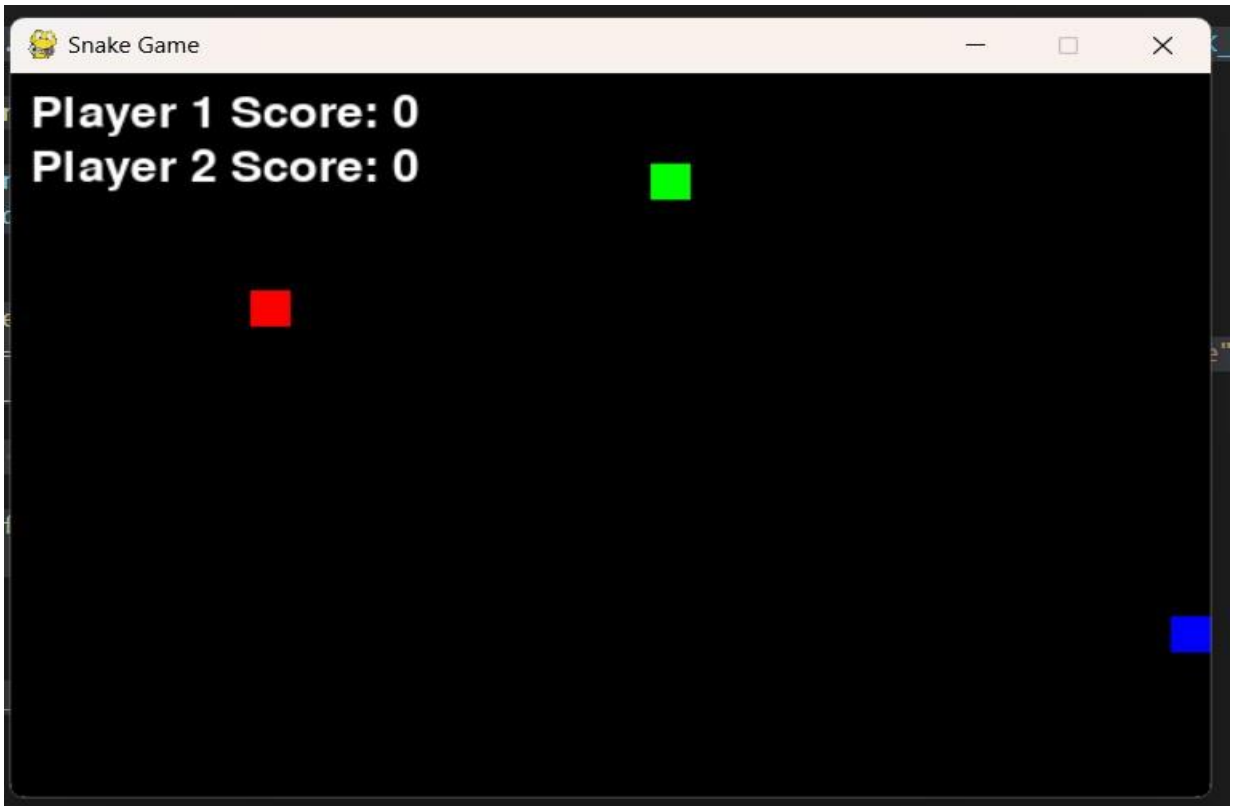
```
    def grow(self):
        self.body.append(self.body[-1])

# Food class
class Food:
    def __init__(self):
        self.position = [random.randrange(1, (SCREEN_WIDTH // BLOCK_SIZE)) * BLOCK_SIZE,
                         random.randrange(1, (SCREEN_HEIGHT // BLOCK_SIZE)) * BLOCK_SIZE]
        self.color = RED

    def spawn_food(self):
        self.position = [random.randrange(1, (SCREEN_WIDTH // BLOCK_SIZE)) * BLOCK_SIZE,
                         random.randrange(1, (SCREEN_HEIGHT // BLOCK_SIZE)) * BLOCK_SIZE]

# Function to draw snake
def draw_snake(snake):
    for block in snake.body:
        pygame.draw.rect(screen, snake.color, pygame.Rect(block[0], block[1], BLOCK_SIZE,
BLOCK_SIZE))

# Function to check for collision with wall or itself
def check_collision(snake):
    if snake.body[0][0] < 0 or snake.body[0][0] >= SCREEN_WIDTH or snake.body[0][1] < 0
or snake.body[0][1] >= SCREEN_HEIGHT:
        return True
    for block in snake.body[1:]:
        if snake.body[0] == block:
            return True
    return False

# Function to check if snake has eaten food
def check_eat_food(snake, food):
    if snake.body[0] == food.position:
        snake.grow()
        food.spawn_food()
        return True
    return False

# Function to display scores
def  display_scores(score1,  score2):
    font = pygame.font.Font(None, 36)
    score_text1 = font.render(f"Player 1 Score: {score1}", True, WHITE)
    score_text2 = font.render(f"Player 2 Score: {score2}", True, WHITE)
    screen.blit(score_text1, (10, 10))
    screen.blit(score_text2, (10, 40))

# Function to display winner
def display_winner(winner):
```

```
    font = pygame.font.Font(None, 50)
    winner_text = font.render(f"The winner is {winner}!", True, WHITE)
    screen.blit(winner_text, (SCREEN_WIDTH // 2 - 150, SCREEN_HEIGHT // 2))

# Main function
def main():
    snake1 = Snake(GREEN, [100, 50])
    snake2 = Snake(BLUE, [400, 300])
    food = Food()

    score1 = 0
    score2 = 0

    clock = pygame.time.Clock()
    running = True
    game_over = False
    game_end_time = None

    while running:
        screen.fill(BLACK)

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                running = False
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_UP and snake1.direction != "DOWN":
                    snake1.direction = "UP"
                elif event.key == pygame.K_DOWN and snake1.direction != "UP":
                    snake1.direction = "DOWN"
                elif event.key == pygame.K_LEFT and snake1.direction != "RIGHT":
                    snake1.direction = "LEFT"
                elif event.key == pygame.K_RIGHT and snake1.direction != "LEFT":
                    snake1.direction = "RIGHT"

                if event.key == pygame.K_w and snake2.direction != "DOWN":
                    snake2.direction = "UP"
                elif event.key == pygame.K_s and snake2.direction != "UP":
                    snake2.direction = "DOWN"
                elif event.key == pygame.K_a and snake2.direction != "RIGHT":
                    snake2.direction = "LEFT"
                elif event.key == pygame.K_d and snake2.direction != "LEFT":
                    snake2.direction = "RIGHT"

        if not game_over:
            snake1.move()
            snake2.move()

            if check_collision(snake1) or check_collision(snake2):
                game_over = True
```

```
            if check_eat_food(snake1, food):
                score1 += 1
            if check_eat_food(snake2, food):
                score2 += 1

            draw_snake(snake1)
            draw_snake(snake2)

            pygame.draw.rect(screen, food.color, pygame.Rect(food.position[0],
food.position[1], BLOCK_SIZE, BLOCK_SIZE))

            display_scores(score1, score2)

            if game_over:
                game_end_time = time.time()

        else:
            if time.time() - game_end_time < 5:
                winner = "Player 1" if score1 > score2 else "Player 2" if score2 > score1
else "No one"
                display_winner(winner)
            else:
                running = False

        pygame.display.flip()
        clock.tick(FPS)

    pygame.quit()

if __name__ == "__main__":
    main()
```

OUTPUT

:


Snake Game
Player 1 Score: 0
Player 2 Score: 0


Snake Game
The winner is Player 1!

5. Future Enhancements:

- **Sound Effects:** Add sound effects for snake movement, food consumption, and game over events.
- **Difficulty Levels:** Implement multiple difficulty levels with varying game speed and obstacle density.
- **Customization Options:** Allow players to customize snake colors, game backgrounds, and other visual elements.
- **Network Multiplayer:** Enable multiplayer functionality over a network to allow players to compete remotely.
- **AI Opponents:** Implement AI-controlled snakes for single-player mode, providing a challenging experience.

6. Conclusion:

- The 2-Player Snake Game provides an engaging and competitive gaming experience for two players.
- Its simple yet addictive gameplay, combined with colorful graphics and smooth animations, makes it suitable for players of all ages.
- The game can serve as a fun leisure activity or a learning tool for programming and game development enthusiasts.

7. References:

- Pygame Documentation: https://www.pygame.org/docs/
- Python Documentation: https://docs.python.org/