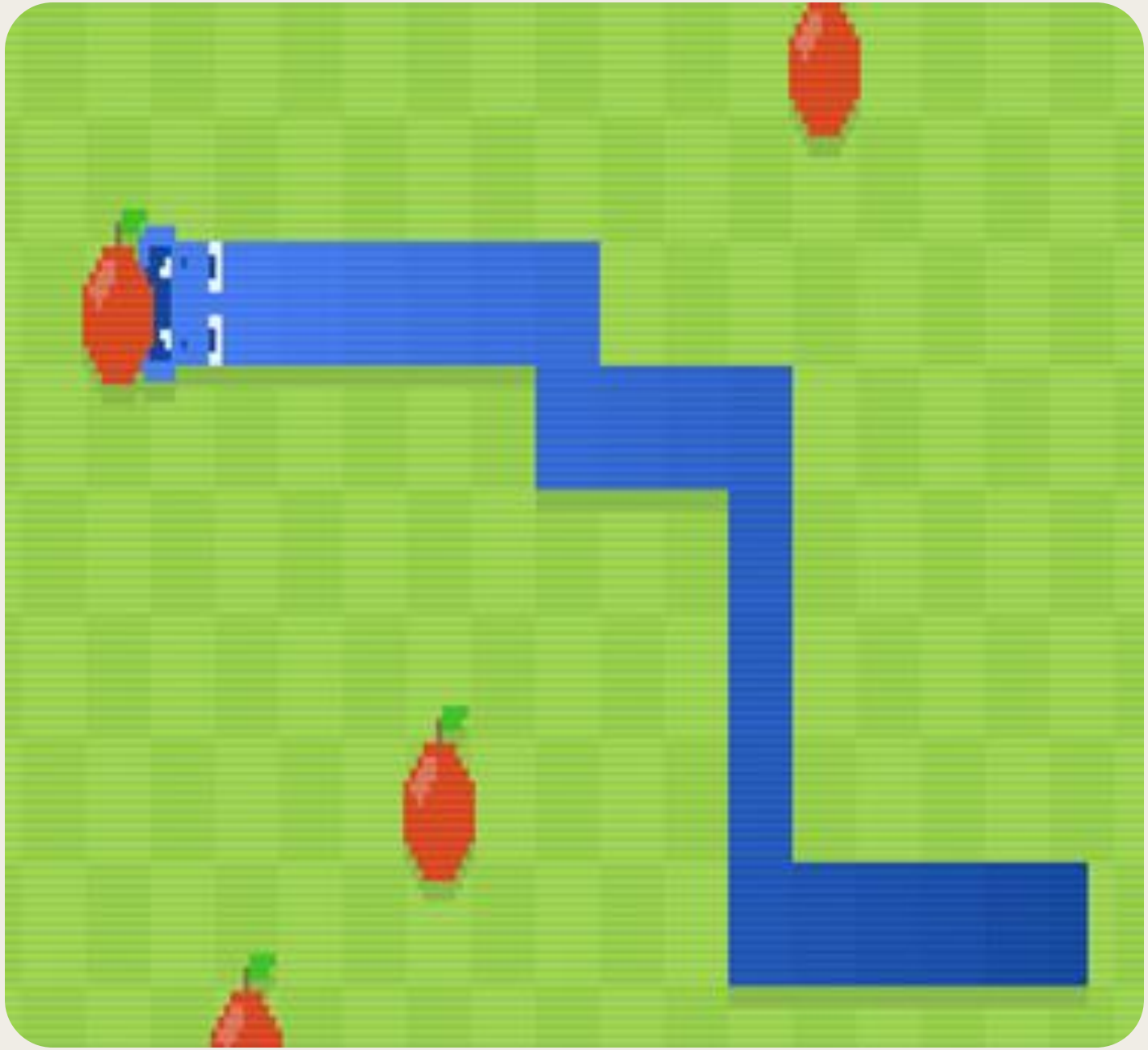
mySnakeGame

Final Project created and presented by Noah Dumas



Game Overview

- Simple Snake game using pygame
- Snake moves on a grid and eats red food blocks
- Each food makes the snake longer and increases the score
- Game ends if the snake hits a wall or its own body

```python
class SnakeGame:
def __init__(self):
    pygame.init()
    self.screen = pygame.display.set_mode((WINDOW_WIDTH, WINDOW_HEIGHT))
    pygame.display.set_caption("Snake Game")
    self.clock = pygame.time.Clock()
    self.font = pygame.font.SysFont("arial", 24)
    self.reset_game()

def reset_game(self):
    start_x = GRID_WIDTH // 2
    start_y = GRID_HEIGHT // 2
    self.snake = [(start_x, start_y)]
    self.direction = (0, 0)
    self.started = False
    self.score = 0
    self.game_over = False
    self.spawn_food()

def spawn_food(self):
    while True:
        food_x = random.randint(0, GRID_WIDTH - 1)
        food_y = random.randint(0, GRID_HEIGHT - 1)
        if (food_x, food_y) not in self.snake:
            self.food = (food_x, food_y)
            break
```

Code Structure and Classes

- All logic is inside one main class SnakeGame
- Constructor __init__ sets up pygame, window, clock, and font
- reset_game initializes snake position, direction, score, and game over flag
- Snake body is stored as a list of grid coordinates (x, y)
- Food is placed with spawn_food at a random empty cell

```python
def handle_input(self):
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            return False

        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_ESCAPE:
                return False

            if not self.game_over:
                if event.key == pygame.K_w and self.direction != (0, 1):
                    self.direction = (0, -1)
                    self.started = True
                elif event.key == pygame.K_s and self.direction != (0, -1):
                    self.direction = (0, 1)
                    self.started = True
                elif event.key == pygame.K_a and self.direction != (1, 0):
                    self.direction = (-1, 0)
                    self.started = True
                elif event.key == pygame.K_d and self.direction != (-1, 0):
                    self.direction = (1, 0)
                    self.started = True

            if self.game_over and event.key == pygame.K_SPACE:
                self.reset_game()
    return True
```

Controls and Player Input

- Uses pygame event loop to read keyboard input
- WASD control movement on the grid
- Direction cannot reverse directly into the opposite direction
- ESC closes the window
- SPACE resets the game after game over

```python
def update_snake(self):
    if self.game_over:
        return
    if not self.started:
        return

    head_x, head_y = self.snake[0]
    dx, dy = self.direction
    new_head_x = head_x + dx
    new_head_y = head_y + dy
    new_head = (new_head_x, new_head_y)

    if (new_head_x < 0 or new_head_x >= GRID_WIDTH or
        new_head_y < 0 or new_head_y >= GRID_HEIGHT):
        self.game_over = True
        return

    if new_head in self.snake:
        self.game_over = True
        return

    self.snake.insert(0, new_head)

    if new_head == self.food:
        self.score += 1
        self.spawn_food()
    else:
        self.snake.pop()
```

Movement, Collisions, Scoring

- update_snake runs every frame to move the snake
- Uses the current direction to compute a new head position
- Checks collisions with walls and with the snake body
- If food is reached, score increases and snake grows
- If no food is reached, tail segment is removed so the snake moves forward

Drawing and Main Loop

- `draw` handles all drawing in one place
- clears the screen
- draws food, snake, and score
- shows game over message
- `run` is the main loop
- reads input
- updates game state
- draws frame
- limits speed with `clock.tick(fps)`

```
def draw(self):
    self.screen.fill(COLOR_BACKGROUND)
    self.draw_grid_cell(self.food, COLOR_FOOD)

    for segment in self.snake:
        self.draw_grid_cell(segment, COLOR_SNAKE)

    score_text = f"Score: {self.score}"
    score_surface = self.font.render(score_text, True, COLOR_TEXT)
    self.screen.blit(score_surface, (10, 10))

    if self.game_over:
        message = "Game Over."
        message_surface = self.font.render(message, True, COLOR_TEXT)
        message_rect = message_surface.get_rect(
            center=(WINDOW_WIDTH // 2, WINDOW_HEIGHT // 2)
        )
        self.screen.blit(message_surface, message_rect)

    pygame.display.flip()

def run(self):
    running = True
    fps = 10

    while running:
        running = self.handle_input()
        self.update_snake()
        self.draw()
        self.clock.tick(fps)
```

DEMO