EXAMEN FINAL OPTATIVA I

PYTHON

DOCUMENTACION DEL CÓDIGO

NOMBRE: FRANCISCO ARIEL GALEANO RUIZ DIAZ

DOCENTE: ING. RICARDO MAIDANA

SEMESTRE: 9NO

CAACUPÉ – PARAGUAY

2024

INDICE

1. Introducción

El código proporcionado implementa un juego clásico de Snake utilizando la biblioteca Pygame en Python. En este juego, el jugador controla una serpiente que debe moverse por la pantalla para recolectar frutas sin chocar contra sí misma o contra los bordes de la pantalla.

1.1 Inicialización y Configuración

```
import pygame
import sys
import random
# Inicializar Pygame
pygame.init()
# Configuración de la pantalla
screen_width = 480
screen_height = 480
cell_size = 20
grid_width = screen_width // cell_size
grid_height = screen_height // cell_size
screen = pygame.display.set_mode((screen_width, screen_height))
pygame.display.set_caption('Snake Game')
```

1.2 Pygame y pantalla:

- Se importa Pygame y se inicializa (pygame.init()).
- Se configura la pantalla del juego con dimensiones específicas (screen_width y screen_height) y el tamaño de las celdas (cell_size).
- Se calcula el número de celdas en el ancho y alto de la pantalla (grid_width y grid_height).

2. Definición de Colores y Direcciones

```
# Colores

WHITE = (255, 255, 255)

BLACK = (0, 0, 0)

GREEN = (0, 255, 0)

RED = (255, 0, 0)

BROWN = (139, 69, 19)

# Direcciones

UP = (0, -1)

DOWN = (0, 1)

LEFT = (-1, 0)

RIGHT = (1, 0)
```

2.1 Colores: Se definen algunos colores en formato RGB que se utilizarán para la pantalla y los elementos del juego.

2.2 Direcciones: Se definen las direcciones posibles para el movimiento de la serpiente, representadas como tuplas de incrementos/decrementos en coordenadas x e y.

2.3 Carga de Imágenes

```
# Cargar las imágenes

snake_head_image = pygame.image.load('gus.png')

snake_head_image = pygame.transform.scale(snake_head_image, (cell_size, cell_size))

fruit_image = pygame.image.load('fruit.png')

fruit_image = pygame.transform.scale(fruit_image, (cell_size, cell_size))
```

2.4 Se cargan y escalan dos imágenes que se utilizarán en el juego:

snake_head_image: Imagen de la cabeza de la serpiente.

fruit_image: Imagen de la fruta que la serpiente debe recolectar.

3. Función para Mostrar Game Over

```
def show_game_over_message():

    font = pygame.font.SysFont(None, 55)

    game_over_text = font.render('Perdiste el juego!', True, RED)

    screen.blit(game_over_text, (screen_width // 4, screen_height
// 2))

    pygame.display.flip()

    pygame.time.wait(6000)  # Espera 6 segundos antes de cerrar.
```

3.1 show_game_over_message():

Esta función muestra un mensaje de "Perdiste el juego!" en rojo centrado en la pantalla cuando el juego termina.

Espera 6 segundos antes de cerrar la ventana del juego

3.2 Clase Snake

```
class Snake:

    def __init__(self):

        self.body = [(screen_width // 2, screen_height // 2)]

        self.length = 1

        self.direction = random.choice([UP, DOWN, LEFT, RIGHT])

        self.score = 0

    def move(self):

        head_x, head_y = self.body[0]

        dir_x, dir_y = self.direction
```

```
        new_head = ((head_x + dir_x * cell_size) % screen_width,
(head_y + dir_y * cell_size) % screen_height)

        if new_head in self.body[1:]:

            return False

        self.body.insert(0, new_head)

        if len(self.body) > self.length:

            self.body.pop()

        return True

    def change_direction(self, direction):

        self.direction = direction

    def collides_with_food(self, food):

        return self.body[0] == food.position

    def eat_food(self):

        self.length += 1

    def check_collision(self):

        return any(self.body[0] == part for part in self.body[1:])

    def draw(self):

        # Rotar la imagen de la cabeza según la dirección

        if self.direction == UP:

            rotated_head                                          =
pygame.transform.rotate(snake_head_image, 90)

        elif self.direction == DOWN:

            rotated_head                                          =
pygame.transform.rotate(snake_head_image, 270)

        elif self.direction == LEFT:

            rotated_head                                          =
pygame.transform.rotate(snake_head_image, 180)
```

```
        elif self.direction == RIGHT:
            rotated_head = snake_head_image
        for i, segment in enumerate(self.body):
            if i == 0:
                screen.blit(rotated_head, segment)
            else:
                pygame.draw.rect(screen, BROWN, (segment[0], segment[1], cell_size, cell_size))
    def reset(self):
        self.body = [(screen_width // 2, screen_height // 2)]
        self.length = 1
        self.direction = random.choice([UP, DOWN, LEFT, RIGHT])
```

3.3 Clase Snake:

Representa la serpiente en el juego.

__init__(): Inicializa la serpiente en el centro de la pantalla con una longitud inicial de 1 y una dirección aleatoria.

move(): Mueve la serpiente según su dirección actual y verifica si hay colisión con sí misma.

change_direction(): Cambia la dirección de la serpiente según la tecla presionada por el jugador.

collides_with_food(): Verifica si la cabeza de la serpiente colisiona con la posición de la fruta.

eat_food(): Incrementa la longitud de la serpiente cuando come una fruta.

check_collision(): Verifica si la serpiente colisiona consigo misma.

draw(): Dibuja la serpiente en la pantalla, rotando la imagen de la cabeza según su dirección.

reset(): Reinicia los atributos de la serpiente cuando el juego termina o se reinicia.

3.4 Clase Food

```
class Food:

    def __init__(self):

        self.position = (random.randint(0, grid_width - 1) *
cell_size, random.randint(0, grid_height - 1) * cell_size)

    def draw(self):

        screen.blit(fruit_image, self.position)

    def randomize_position(self):

        self.position = (random.randint(0, grid_width - 1) *
cell_size, random.randint(0, grid_height - 1) * cell_size)
```

3.5 Clase Food:

Representa la fruta en el juego que la serpiente debe recolectar.

__init__(): Inicializa la fruta en una posición aleatoria dentro de los límites de la pantalla.

draw(): Dibuja la fruta en la posición actual en la pantalla.

randomize_position(): Cambia aleatoriamente la posición de la fruta cuando la serpiente la recolecta.

3.6 Función principal main()

```
def main():

    clock = pygame.time.Clock()

    snake = Snake()

    food = Food()

    game_over = False

    while True:

        for event in pygame.event.get():

            if event.type == pygame.QUIT:

                pygame.quit()
```

```
                sys.exit()

            if event.type == pygame.KEYDOWN:

                if event.key == pygame.K_UP and snake.direction
!= DOWN:

                    snake.change_direction(UP)

                elif    event.key    ==    pygame.K_DOWN    and
snake.direction != UP:

                    snake.change_direction(DOWN)

                elif    event.key    ==    pygame.K_LEFT    and
snake.direction != RIGHT:

                    snake.change_direction(LEFT)

                elif    event.key    ==    pygame.K_RIGHT    and
snake.direction != LEFT:

                    snake.change_direction(RIGHT)

        if not game_over:

            if snake.move():

                if snake.collides_with_food(food):

                    snake.eat_food()

                    food.randomize_position()

                if snake.check_collision():

                    game_over = True

                    show_game_over_message()

                    # Reiniciar el juego después de mostrar el
mensaje

                    snake.reset()

                    game_over = False

            else:
```

```
                game_over = True
                show_game_over_message()
                # Reiniciar el juego después de mostrar el mensaje
                snake.reset()
                game_over = False
        # Dibujar todo
        screen.fill(BLACK)
        snake.draw()
        food.draw()
        pygame.display.flip()
        clock.tick(10)  # Velocidad de actualización
```

3.7 Función main():

Esta es la función principal que controla el flujo del juego.

Se crea un objeto clock para controlar la velocidad de actualización de la pantalla.

Se crean instancias de la serpiente (snake) y la fruta (food).

El juego se ejecuta en un bucle principal (while True), donde se manejan los eventos de Pygame (como cerrar la ventana o presionar teclas).

Dependiendo de si el juego está en curso o ha terminado (game_over), se mueve la serpiente, se maneja la recolección de frutas, se verifica colisiones y se muestra el mensaje de Game Over.

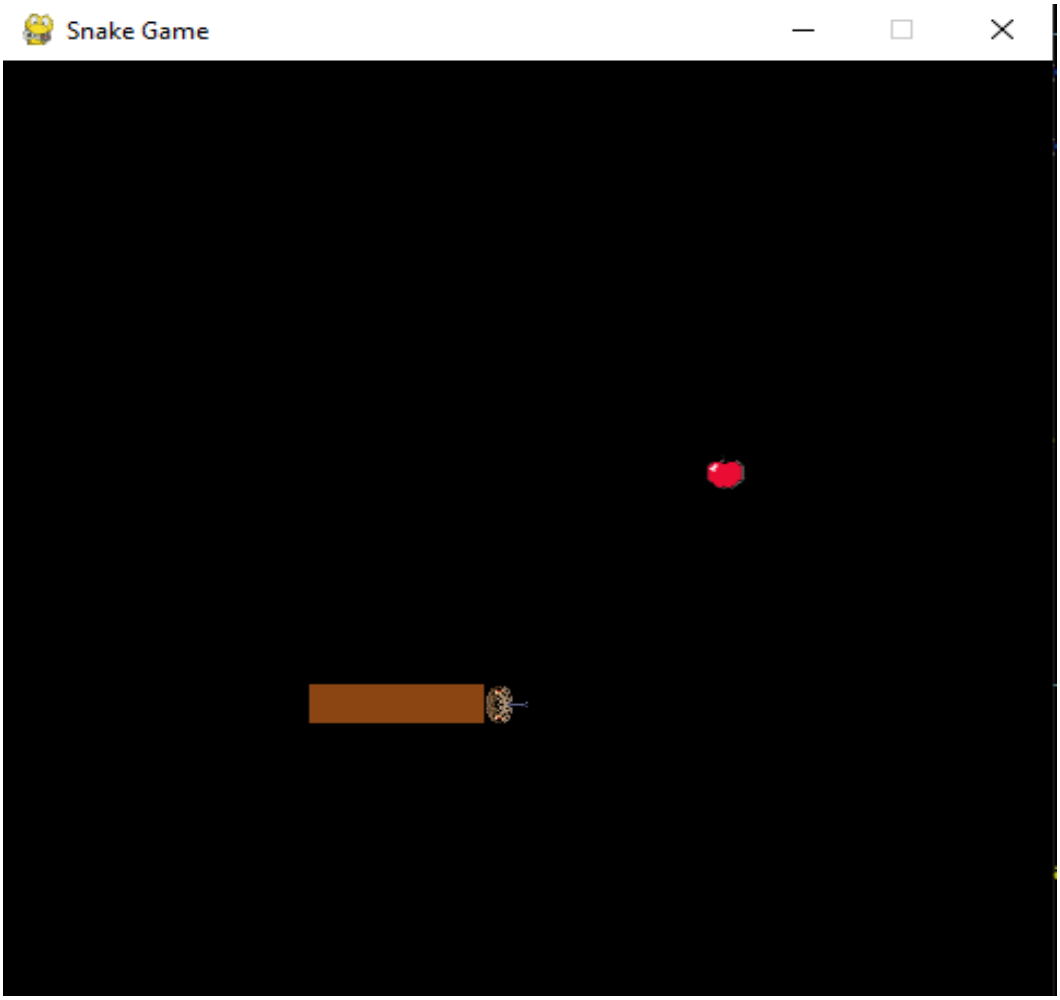
Se actualiza y dibuja todo en la pantalla (screen.fill(BLACK), snake.draw(), food.draw()), y se llama a pygame.display.flip() para actualizar la pantalla.

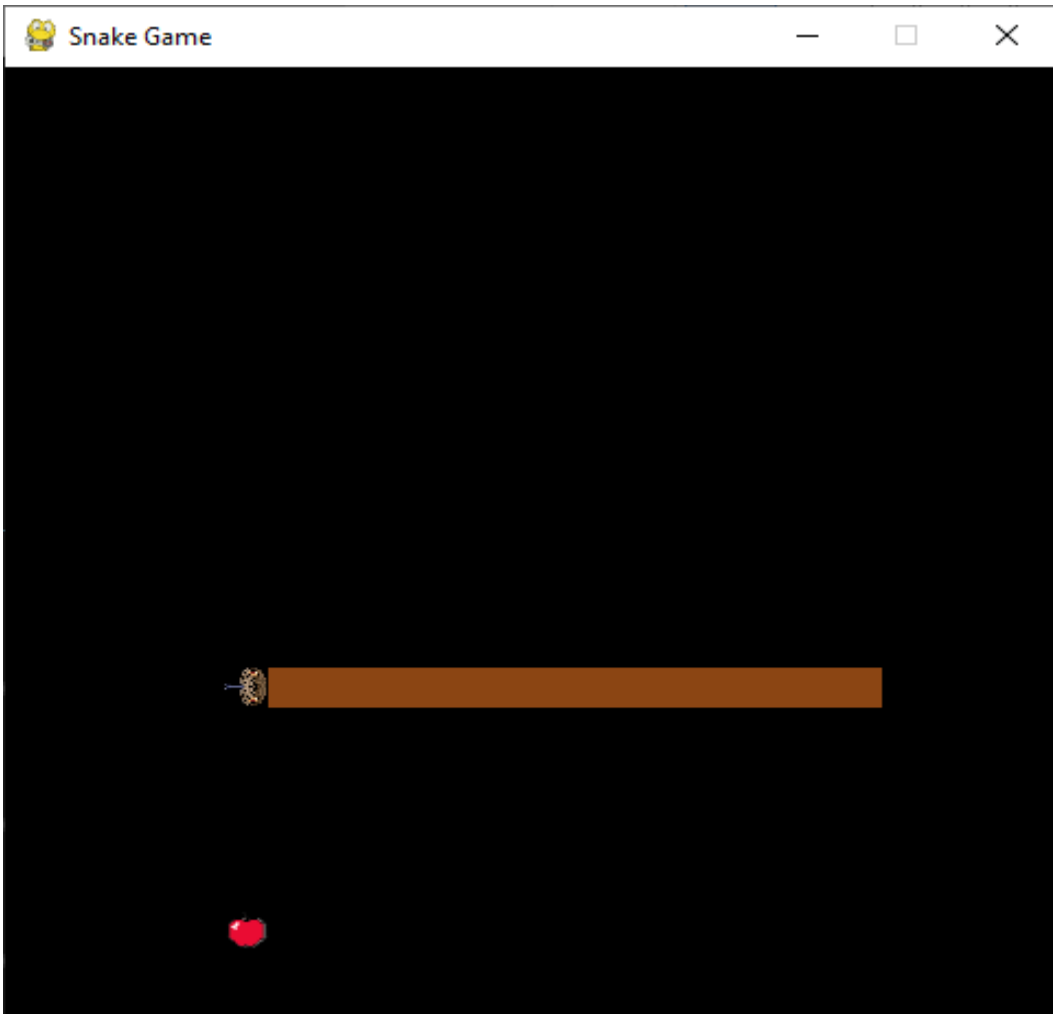
clock.tick(10) establece la velocidad de actualización a 10 fotogramas por segundo.

4. Conclusión.

El juego de Snake implementado en este código utiliza Pygame para manejar la interfaz gráfica y la lógica del juego. Permite al jugador controlar una serpiente que se mueve por la pantalla para recolectar frutas mientras evita colisiones consigo misma y con los bordes de la pantalla. El código está estructurado en clases (Snake y Food) para encapsular la funcionalidad relacionada con la serpiente y la fruta, lo que facilita la modularidad y el mantenimiento del código.

5. ANEXO




Snake Game