Deliverable Session 04 Micro Game (Optional) – Snake Game with Sense HAT

Rafael Antonio Echevarria Silva

En esta sesión crearemos un sencillo juego de “Snake” sobre la matriz LED de Sense HAT, controlado por la inclinación del dispositivo y con generación de comida aleatoria.

Ex. 1: Inicialización y configuración de colores

Comenzamos importando las librerías necesarias y creando el objeto **SenseHat**:

```
from sense_hat import SenseHat
from time import sleep
import random

sense = SenseHat()
```

A continuación definimos los colores que utilizaremos:

- g = (0, 255, 0): verde (color de la serpiente).
- r = (255, 0, 0): rojo (color de la comida).

También inicializamos la dirección por defecto **("right")**, la posición inicial de la serpiente **(serpi)** y el contador de puntos **(contador = 0)**.

Ex. 2: Generación aleatoria de comida

Para colocar la comida en una posición no ocupada por la serpiente, definimos la función **food_spawn()**:

```
def food_spawn():
    while True:
        x, y = random.randint(1, 6), random.randint(1, 6)
        if (x, y) not in serpi:
            return (x, y)
```

- Genera coordenadas **(x, y)** aleatorias en el rango [1, 6] (dejando un borde libre).
- Comprueba que no coincidan con ningún segmento de **serpi**.
- Devuelve la posición válida de la comida.

La posición inicial de la comida se obtiene así:

```
food = food_spawn()
```

Ex. 3: Dibujo de la serpiente y la comida

Cada iteración refrescamos la pantalla y pintamos la serpiente y la comida:

```
def draw():
    sense.clear()
    for segment in serpi:
        sense.set_pixel(segment[0], segment[1], g)
    sense.set_pixel(food[0], food[1], r)
```

- **sense.clear()**: limpia todos los píxeles.
- Recorremos cada **segment** de la lista serpi y lo pintamos de verde.
- Finalmente, pintamos la posición de **food** en rojo.

Ex. 4: Lectura del acelerómetro para cambiar la dirección

Usamos el acelerómetro interno para leer la inclinación y actualizar **direction**:

```
def update_direction():
    global direction
    accel = sense.get_accelerometer_raw()
    x, y = accel['x'], accel['y']

    if abs(x) > abs(y):
        if x > 0.2 and direction != "left":
            direction = "right"
        elif x < -0.2 and direction != "right":
            direction = "left"
    else:
        if y > 0.2 and direction != "up":
            direction = "down"
        elif y < -0.2 and direction != "down":
            direction = "up"
```

- Comparamos valores absolutos de *x* e *y* para saber si el movimiento es horizontal o vertical.
- Sólo cambiamos la dirección si no es la opuesta a la actual (evitando que la serpiente se choque contra sí misma inmediatamente).

Ex. 5: Movimiento y detección de colisiones

Calculamos la nueva cabeza de la serpiente según **direction** y comprobamos colisiones:

```
def mov_serpi():
    cabeza_x, cabeza_y = serpi[0]
    if direction == "up":
        cabeza_y -= 1
    elif direction == "down":
```

```
        cabeza_y += 1
    elif direction == "left":
        cabeza_x -= 1
    elif direction == "right":
        cabeza_x += 1

    if not (0 <= cabeza_x < 8 and 0 <= cabeza_y < 8):
        dead(True)
    return (cabeza_x, cabeza_y)
```

- Movemos la cabeza un píxel en la dirección indicada.
- Si sale del rango **[0,7]** en cualquier eje, se llama a **dead(True)** para finalizar el juego.
- En el bucle principal, tras obtener ****nueva_cabeza**, comprobamos también si colisiona con el propio cuerpo:

```
if nueva_cabeza in serpi:
    dead(True)
serpi.insert(0, nueva_cabeza)
```

Ex. 6: Comer, puntuar y final del juego

Tras mover la cabeza ejecutamos los siguientes pasos:

1. Comer comida:

```
if nueva_cabeza == food:
    food = food_spawn()
    contador += 1
else:
    serpi.pop()
```

Si la cabeza alcanza la posición de **food**, generamos nueva comida y aumentamos contador. Si no, eliminamos el último segmento para mantener longitud constante.

2. Dibujar y esperar:

```
draw()
sleep(0.6)
```

3. Game Over: La función **dead(muerte)** muestra el mensaje en pantalla y sale del programa:

```
def dead(muerte):
    if muerte:
        sense.show_message(
```

```
            f"Game Over Puntos: {contador}",
            text_colour=(255, 0, 0)
        )
        sleep(2)
        exit()
```

Con esto, cada ciclo de juego actualiza la dirección, mueve la serpiente, gestiona la comida, redibuja la pantalla y controla el final de la partida.