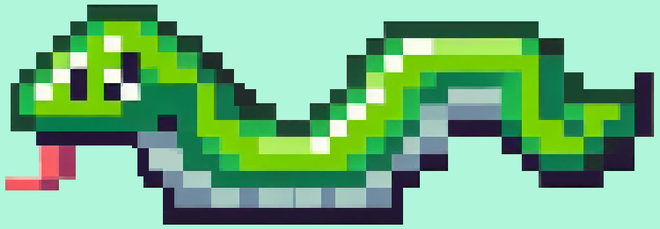
UIDE
Powered by
Arizona State University

JUEGO DE LA SERPIENTE

Snaqui

Explicación del Código Final



Por Lorenzo Cumbal.

```
#Importación de módulos necesarios
import pygame
import random
import sys
from enum import Enum
```

```
# Inicialización de Pygame y configuración inicial
pygame.init()

# Configuración usando un diccionario
CONFIG = {
  "ANCHO": 600,
  "ALTO": 600,
  "TAMANO_CELDA": 30,
  "COLORES": {
    "NEGRO": (0, 0, 0),
    "BLANCO": (255, 255, 255),
    "VERDE": (0, 255, 0),
    "ROJO": (255, 0, 0),
    "AZUL": (0, 0, 255),
    "GRIS": (200, 200, 200)
  },
  "FILAS": 600 // 30, # Calculado como ALTO // TAMANO_CELDA
  "COLUMNAS": 600 // 30 # Calculado como ANCHO // TAMANO_CELDA
}
```

```
# Paleta de colores (acceso desde el diccionario)
COLORES = CONFIG["COLORES"]

# Estados del juego usando Enum
class EstadoJuego(Enum):
  MENU = 1
  PLAYING = 2
  GAME_OVER = 3

# Configuración de ventana inicial
ventana = pygame.display.set_mode((CONFIG["ANCHO"], CONFIG["ALTO"]))
pygame.display.set_caption("SNAQUI")
reloj = pygame.time.Clock()
```

```
# Clase Boton
class Boton:
  def __init__(self, x, y, ancho, alto, texto, color):
    self.rect = pygame.Rect(x, y, ancho, alto)
    self.texto = texto
    self.color_actual = color
    self.fuente = pygame.font.SysFont('Arial', 30)

  def dibujar(self, superficie):
    pygame.draw.rect(superficie, self.color_actual, self.rect)
    texto_surf = self.fuente.render(self.texto, True, COLORES["BLANCO"])
    texto_rect = texto_surf.get_rect(center=self.rect.center)
    superficie.blit(texto_surf, texto_rect)
```

```
# Función para dibujar texto
def dibujar_texto(texto, tamaño, color, x, y):
  fuente = pygame.font.SysFont('Arial', tamaño)
  texto_surf = fuente.render(texto, True, color)
  texto_rect = texto_surf.get_rect(center=(x, y))
  ventana.blit(texto_surf, texto_rect)

# Función para dibujar botones
def dibujar_botones(botones, ventana):
  for boton in botones:
    boton.dibujar(ventana)

# Función para salir del juego
def salir():
  pygame.quit()
  sys.exit()
```

```
# Función para mostrar el menú principal
def mostrar_menu(estado_juego):
  ventana.fill(COLORES["AZUL"])
  dibujar_texto("SNAQUI", 50, COLORES["BLANCO"], CONFIG["ANCHO"] // 2,
CONFIG["ALTO"] // 4)

  # Lista de botones con diccionarios
  botones = [
    Boton(CONFIG["ANCHO"] // 2 - 100, CONFIG["ALTO"] // 2 - 50, 200, 50, "JUGAR",
COLORES["VERDE"]),

    Boton(CONFIG["ANCHO"] // 2 - 100, CONFIG["ALTO"] // 2 + 50, 200, 50, "SALIR",
COLORES["ROJO"])
  ]
  dibujar_botones(botones, ventana)
```

```
for event in pygame.event.get(): # Leer eventos
  if event.type == pygame.QUIT: # Si se pulsa el boton x de la ventana
    salir() # Salir del juego
  if event.type == pygame.MOUSEBUTTONDOWN:
    for boton in botones:
      if boton.rect.collidepoint(event.pos):
        if boton.texto == "JUGAR":
          return EstadoJuego.PLAYING
        elif boton.texto == "SALIR":
          salir()

pygame.display.update()
return estado_juego
```

```
# Función para mostrar Game Over
def mostrar_game_over(puntuacion):
  ventana.fill(COLORES["NEGRO"])
  dibujar_texto("GAME OVER", 50, COLORES["ROJO"], CONFIG["ANCHO"]//2,
CONFIG["ALTO"]//4)
  dibujar_texto(f"Puntuación: {puntuacion}", 30, COLORES["BLANCO"],
CONFIG["ANCHO"]//2, CONFIG["ALTO"]//3)

  # Lista de botones con diccionarios
  botones = [
  Boton(CONFIG["ANCHO"]//2 - 100, CONFIG["ALTO"]//2 - 50, 200, 50, "Reiniciar",
COLORES["VERDE"]),
  Boton(CONFIG["ANCHO"]//2 - 100, CONFIG["ALTO"]//2 + 50, 200, 50, "Menú
Principal", COLORES["AZUL"]),
  Boton(CONFIG["ANCHO"]//2 - 100, CONFIG["ALTO"]//2 + 150, 200, 50, "Salir",
COLORES["ROJO"])
]
  dibujar_botones(botones, ventana)
```

```
for event in pygame.event.get(): # Leer eventos
  if event.type == pygame.QUIT: # Si se pulsa el boton x de la ventana
    salir() # Salir del juego
  if event.type == pygame.MOUSEBUTTONDOWN:
    for boton in botones:
      if boton.rect.collidepoint(event.pos):
        if boton.texto == "Reiniciar":
          return EstadoJuego.PLAYING
        elif boton.texto == "Menú Principal":
          return EstadoJuego.MENU
        elif boton.texto == "Salir":
          salir()

pygame.display.update()
return EstadoJuego.GAME_OVER
```

```
# Función para mostrar menú de pausa
def mostrar_pausa():
  while True:
    ventana.fill(COLORES["NEGRO"])
    dibujar_texto("PAUSA", 50, COLORES["BLANCO"], CONFIG["ANCHO"]//2,
CONFIG["ALTO"]//4)

    # Lista de botones con diccionarios
    botones = [
      Boton(CONFIG["ANCHO"]//2 - 100, CONFIG["ALTO"]//2 - 95, 200, 50, "Reanudar",
COLORES["VERDE"]),
      Boton(CONFIG["ANCHO"]//2 - 100, CONFIG["ALTO"]//2 - 10, 200, 50, "Reiniciar",
COLORES["GRIS"]),
      Boton(CONFIG["ANCHO"]//2 - 100, CONFIG["ALTO"]//2 + 70, 200, 50, "Menú
Principal", COLORES["AZUL"]),
      Boton(CONFIG["ANCHO"]//2 - 100, CONFIG["ALTO"]//2 + 150, 200, 50, "Salir",
COLORES["ROJO"])
    ]
    dibujar_botones(botones, ventana)
```

```
    for event in pygame.event.get(): # Leer eventos
      if event.type == pygame.QUIT: # Si se pulsa el boton x de la
ventana
        salir() # Salir del juego
      if event.type == pygame.MOUSEBUTTONDOWN:
        for boton in botones:
          if boton.rect.collidepoint(event.pos):
                if boton.texto == "Reanudar":
                  pygame.time.wait(1)
                  ventana.fill(COLORES["NEGRO"])
                  dibujar_texto("Espera 3 segundos", 20,
COLORES["BLANCO"], CONFIG["ANCHO"]//2, CONFIG["ALTO"]//2)
                pygame.display.update()
                pygame.time.wait(3000)
                return EstadoJuego.PLAYING
            elif boton.texto == "Reiniciar":
                return 'restart'
            elif boton.texto == "Menú Principal":
                return EstadoJuego.MENU
            elif boton.texto == "Salir":
                salir()

    pygame.display.update()
    reloj.tick(1)
```

```
# Función principal del juego
def juego():
  snake = [(4, 4)]
  comida = (random.randint(0, CONFIG["COLUMNAS"]-1), random.randint(0, CONFIG["FILAS"]-1))
  direccion = None
  puntuacion = 0
  velocidad = 8
  estado_juego = EstadoJuego.PLAYING
  game_started = False

# Diccionario de direcciones
  direcciones = {
    "derecha": (1, 0),
    "izquierda": (-1, 0),
    "arriba": (0, -1),
    "abajo": (0, 1)
  }
```

```
# Bucle principal del juego
while estado_juego == EstadoJuego.PLAYING:
    ventana.fill(COLORES["NEGRO"]) # Limpiar pantalla

    for event in pygame.event.get():

      if event.type == pygame.QUIT:
        salir()
      if event.type == pygame.KEYDOWN:
        if event.key == pygame.K_ESCAPE:
          action = mostrar_pausa()
          if action == 'restart':
            snake = [(4,4)]
            comida = (random.randint(0, CONFIG["COLUMNAS"]-1),
random.randint(0, CONFIG["FILAS"]-1))
            direccion = None
            puntuacion = 0
            velocidad = 8
            game_started = False
            continue
```

```
else:
  # Movimiento
  if not game_started and event.key in (pygame.K_RIGHT, pygame.K_d, pygame.K_LEFT,
                    pygame.K_a, pygame.K_UP, pygame.K_w,
                    pygame.K_DOWN, pygame.K_s):
    game_started = True
    direccion = "derecha" if event.key in (pygame.K_RIGHT, pygame.K_d) else \
          "izquierda" if event.key in (pygame.K_LEFT, pygame.K_a) else \
          "arriba" if event.key in (pygame.K_UP, pygame.K_w) else "abajo"
  elif game_started== True:
    nueva_dir = None
    if event.key in (pygame.K_RIGHT, pygame.K_d) and direccion != "izquierda":
      nueva_dir = "derecha"
    elif event.key in (pygame.K_LEFT, pygame.K_a) and direccion != "derecha":
      nueva_dir = "izquierda"
    elif event.key in (pygame.K_UP, pygame.K_w) and direccion != "abajo":
      nueva_dir = "arriba"
    elif event.key in (pygame.K_DOWN, pygame.K_s) and direccion != "arriba":
      nueva_dir = "abajo"
    if nueva_dir: # Comprueba si la nueva_dir cambia
      direccion = nueva_dir
```

```
if direccion: # Comprueba si la dirección cambia
    cabeza = snake[0]
    dx, dy = direcciones[direccion] # Uso del diccionario
    nueva_cabeza = (cabeza[0] + dx, cabeza[1] + dy)
    #Verificacion de colisiones
    if (nueva_cabeza in snake or
      not (0 <= nueva_cabeza[0] < CONFIG["COLUMNAS"]) or
      not (0 <= nueva_cabeza[1] < CONFIG["FILAS"])):
      estado_juego = EstadoJuego.GAME_OVER
      break
    #Actualizacion de serpiente y comida
    if nueva_cabeza != comida:
      snake = [nueva_cabeza] + snake[:-1]
    elif nueva_cabeza == comida:
      snake = [nueva_cabeza] + snake
      puntuacion += 1
      velocidad += 0.5
      comida = (random.randint(0, CONFIG["COLUMNAS"]-1),
            random.randint(0, CONFIG["FILAS"]-1))
```

```
# Dibujado
for segmento in snake:
  pygame.draw.rect(ventana, COLORES["VERDE"],
        (segmento[0]*CONFIG["TAMANO_CELDA"], segmento[1]*CONFIG["TAMANO_CELDA"],
         CONFIG["TAMANO_CELDA"], CONFIG["TAMANO_CELDA"]))
  pygame.draw.rect(ventana, COLORES["ROJO"],
       (comida[0]*CONFIG["TAMANO_CELDA"], comida[1]*CONFIG["TAMANO_CELDA"],
        CONFIG["TAMANO_CELDA"], CONFIG["TAMANO_CELDA"]))

  dibujar_texto(f"Puntuación: {puntuacion}", 20, COLORES["BLANCO"], CONFIG["ANCHO"]//2, 20)

  if not game_started:
    dibujar_texto("Presione una tecla de dirección para comenzar",
        20, COLORES["BLANCO"], CONFIG["ANCHO"]//2, CONFIG["ALTO"]//2)

  pygame.display.update()
  reloj.tick(velocidad)

return (estado_juego, puntuacion)
```

```
# Función principal
def main():
  estado_juego = EstadoJuego.MENU
  while True:
    if estado_juego == EstadoJuego.MENU:
      estado_juego = mostrar_menu(estado_juego)
    elif estado_juego == EstadoJuego.PLAYING:
      next_state, puntuacion = juego()
      estado_juego = next_state
    elif estado_juego == EstadoJuego.GAME_OVER:
      estado_juego = mostrar_game_over(puntuacion)



  main()
```