Pontificia Universidad Católica del Ecuador
Seréis mis testigos
ESMERALDAS

Tema

Tetris

Asignatura

Programación estructurada y funcional

Presenta

Elias Lozada

Fecha

19/01/2026

Paradigma Utilizado

En el desarrollo de este juego de Tetris se utilizó principalmente la Programación Orientada a Objetos.

Pieza: Representa cada pieza de Tetris, con atributos como su forma, color y posición, y métodos para rotar y moverse.

Juego: Maneja la lógica del juego, incluyendo la detección de colisiones, fijación de piezas y descenso automático.

La elección de POO se justificó porque:

Permite organizar el código de manera modular, separando claramente la lógica de cada pieza de la lógica general del juego.

Facilita la reutilización de código, por ejemplo, para crear nuevas piezas o modificar la lógica de movimiento sin afectar otras partes del juego.

Hace más sencilla la lectura y el mantenimiento, al agrupar comportamiento y atributos relacionados en objetos.

Ejecución del código

Instrucciones:

Python 3.8 o superior

Necesitas tener turtle y pygame

¿Cómo se ejecuta?

Coloca el archivo del juego (por ejemplo: tetris.py) en una carpeta.

En la **misma carpeta**, asegúrate de tener:

- La música y la imagen
- musica.mp3
- fondo.gif

Código

- def iniciar_musica():
- def detener_musica():

Se usa pygame.mixer para reproducir música de fondo.

La música se inicia al comenzar una partida y se detiene al perder o volver al menú.

Class Juego:

- Controla la lógica principal del juego:
- colision(): detecta choques con bordes o piezas
- fijar(): fija la pieza al tablero
- bajar(): hace caer la pieza
- mover(dx): mueve la pieza horizontalmente
- caer_directo(): baja la pieza instantáneamente

Tablero y Puntaje

tablero = [[0 for _ in range(COLUMNAS)] for _ in range(FILAS)]

- El tablero es una matriz 2D.
- Cada línea completa otorga **100 puntos**.
- El puntaje se muestra durante el juego y en **GAME OVER**.

Dibujo con Turtle

Funciones principales:

- dibujar_menu() → Muestra el menú con fondo e instrucciones
- dibujar_juego() → Dibuja el tablero, piezas y puntaje
- dibujar_game_over() → Muestra mensaje y puntaje final
- dibujar_celda() → Dibuja cada bloque individual

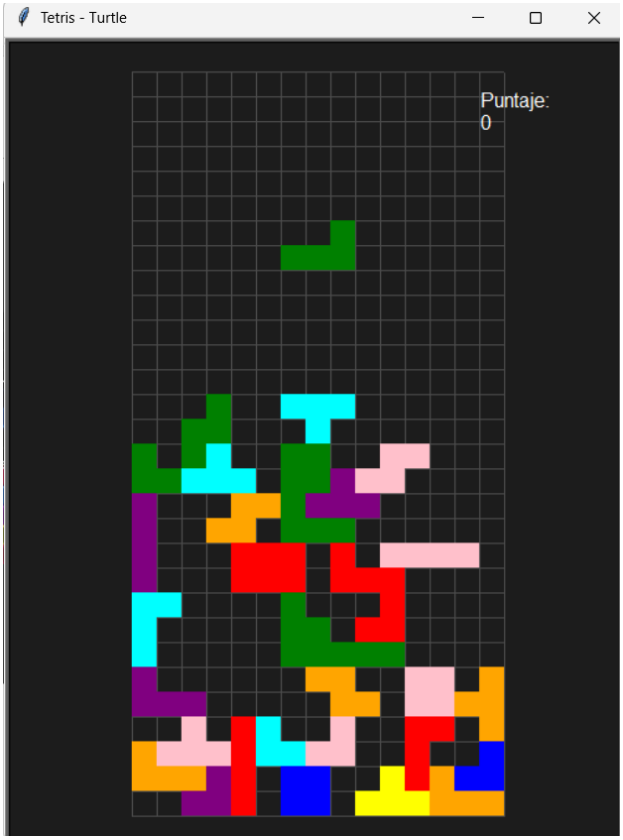
Estados del Juego

estado = "MENU" | "JUGANDO" | "GAME_OVER"

El juego cambia de estado según la situación:

- **MENU**: selección de modo
- **JUGANDO**: partida activa
- **GAME_OVER**: fin de partida y muestra puntaje


Tetris - Turtle
TETRIS
1 - MODO ARCADE
2 - JUGAR POR NIVELES
CONTROLES
← → : Mover ↑ : Rotar
↓ : Bajar ESPACIO : Caer directo


Tetris - Turtle
Puntaje:
0


Tetris - Turtle
GAME OVER
Puntaje final: 0
Presiona ENTER para volver al menú

Código Completo:

```
import turtle
import random
import time
import pygame

# ---------------- MÚSICA ----------------
pygame.mixer.init()

def iniciar_musica():
    pygame.mixer.music.load("musica.mp3")
    pygame.mixer.music.set_volume(0.4)
    pygame.mixer.music.play(-1)

def detener_musica():
    pygame.mixer.music.stop()

# ---------------- CONFIGURACIÓN ----------------
ANCHO, ALTO = 300, 600
TAM = 20
COLUMNAS = ANCHO // TAM
FILAS = ALTO // TAM

VELOCIDAD_BASE = 0.45
MAX_NIVELES = 5
TIEMPO_POR_NIVEL = 25

estado = "MENU"
modo = None
nivel = 1
velocidad_caida = VELOCIDAD_BASE
tiempo_inicio = 0
ultimo_tiempo = time.time()

puntaje = 0
tablero = [[0 for _ in range(COLUMNAS)] for _ in range(FILAS)]

# ---------------- PANTALLA ----------------
pantalla = turtle.Screen()
pantalla.setup(500, 650)
pantalla.bgcolor("#1c1c1c")
pantalla.title("Tetris - Turtle")
pantalla.tracer(0)
pantalla.addshape("fondo_menu.gif")
```

```
# ---------------- TORTUGAS ----------------
dib = turtle.Turtle()
dib.hideturtle()
dib.penup()
dib.speed(0)

fondo = turtle.Turtle()
fondo.hideturtle()
fondo.penup()


FORMAS = [
    # Tetrominós clásicos
    [[1, 1, 1, 1]],

    [[1, 1],
     [1, 1]],

    [[0, 1, 0],
     [1, 1, 1]],

    [[1, 0, 0],
     [1, 1, 1]],

    [[0, 0, 1],
     [1, 1, 1]],

    [[0, 1, 1],
     [1, 1, 0]],

    [[1, 1, 0],
     [0, 1, 1]],

    # Figuras simples adicionales
    [[1, 1, 1]],

    [[1, 0],
     [1, 1]],

    [[0, 1],
     [1, 1]],

    [[1, 1, 1],
     [0, 1, 0]],
```

```
    [[1, 1, 1],
     [1, 1, 1]]
]




COLORES = [
    "cyan", "yellow", "purple",
    "blue", "orange", "green",
    "red", "pink"
]


class Pieza:
    def __init__(self):
        self.forma = random.choice(FORMAS)
        self.color = random.choice(COLORES)
        self.x = COLUMNAS // 2 - 1
        self.y = FILAS - 1

    def rotar(self):
        self.forma = [list(f) for f in zip(*self.forma[::-1])]

class Juego:
    def __init__(self):
        self.pieza = Pieza()

    def colision(self, dx=0, dy=0):
        for y, fila in enumerate(self.pieza.forma):
            for x, celda in enumerate(fila):
                if celda:
                    nx = self.pieza.x + x + dx
                    ny = self.pieza.y - y + dy
                    if nx < 0 or nx >= COLUMNAS or ny < 0:
                        return True
                    if ny < FILAS and tablero[ny][nx]:
                        return True
        return False

    def fijar(self):
        global estado
        for y, fila in enumerate(self.pieza.forma):
            for x, celda in enumerate(fila):
```

```
                if celda:
                    tablero[self.pieza.y - y][self.pieza.x + x] =
self.pieza.color

        limpiar_lineas()
        self.pieza = Pieza()

        if self.colision():
            estado = "GAME_OVER"
            detener_musica()

    def bajar(self):
        if not self.colision(dy=-1):
            self.pieza.y -= 1
        else:
            self.fijar()

    def mover(self, dx):
        if not self.colision(dx=dx):
            self.pieza.x += dx

    def rotar(self):
        copia = self.pieza.forma
        self.pieza.rotar()
        if self.colision():
            self.pieza.forma = copia

    def caer_directo(self):
        while not self.colision(dy=-1):
            self.pieza.y -= 1
        self.fijar()

# --------------- LÓGICA ---------------
def limpiar_lineas():
    global tablero, puntaje
    nuevas = []
    eliminadas = 0

    for fila in tablero:
        if all(fila):
            eliminadas += 1
        else:
            nuevas.append(fila)

    for _ in range(eliminadas):
```

```
        nuevas.append([0] * COLUMNAS)

    tablero = nuevas
    puntaje += eliminadas * 100

# ---------------- DIBUJO ----------------
def dibujar_celda(x, y, color):
    dib.goto(x * TAM - ANCHO // 2, y * TAM - ALTO // 2)
    dib.color(color)
    dib.begin_fill()
    for _ in range(4):
        dib.forward(TAM)
        dib.left(90)
    dib.end_fill()

def dibujar_cuadricula():
    dib.color("gray30")
    for x in range(COLUMNAS + 1):
        dib.goto(x * TAM - ANCHO // 2, -ALTO // 2)
        dib.pendown()
        dib.goto(x * TAM - ANCHO // 2, ALTO // 2)
        dib.penup()
    for y in range(FILAS + 1):
        dib.goto(-ANCHO // 2, y * TAM - ALTO // 2)
        dib.pendown()
        dib.goto(ANCHO // 2, y * TAM - ALTO // 2)
        dib.penup()

def dibujar_menu():
    dib.clear()
    fondo.clear()

    fondo.shape("fondo_menu.gif")
    fondo.goto(0, 0)
    fondo.stamp()



    dib.color("white")
    dib.goto(0, 150)
    dib.write("TETRIS", align="center", font=("Arial", 32, "bold"))

    dib.goto(0, 90)
    dib.write("1 - MODO ARCADE", align="center", font=("Arial", 16))
```

```
    dib.goto(0, 60)
    dib.write("2 - JUGAR POR NIVELES", align="center", font=("Arial", 16))

    dib.goto(0, 10)
    dib.write("CONTROLES", align="center", font=("Arial", 18, "bold"))

    dib.goto(0, -20)
    dib.write("← → : Mover   ↑ : Rotar", align="center", font=("Arial", 14))
    dib.goto(0, -45)
    dib.write("↓ : Bajar   ESPACIO : Caer directo", align="center",
font=("Arial", 14))

    pantalla.update()

def dibujar_juego():
    dib.clear()
    dibujar_cuadricula()

    for y in range(FILAS):
        for x in range(COLUMNAS):
            if tablero[y][x]:
                dibujar_celda(x, y, tablero[y][x])

    for y, fila in enumerate(juego.pieza.forma):
        for x, celda in enumerate(fila):
            if celda:
                dibujar_celda(juego.pieza.x + x, juego.pieza.y - y,
juego.pieza.color)

    dib.goto(160, 250)
    dib.color("white")
    dib.write(f"Puntaje:\n{puntaje}", align="center", font=("Arial", 12))

    if modo == "NIVEL":
        tiempo_restante = max(0, TIEMPO_POR_NIVEL - int(time.time() -
tiempo_inicio))
        dib.goto(160, 210)
        dib.write(f"Nivel: {nivel}", align="center", font=("Arial", 12))
        dib.goto(160, 180)
        dib.write(f"Tiempo: {tiempo_restante}s", align="center",
font=("Arial", 12))

    pantalla.update()

def dibujar_game_over():
```

```
    dib.clear()

    dib.color("red")
    dib.goto(0, 60)
    dib.write("GAME OVER", align="center", font=("Arial", 28, "bold"))

    dib.color("white")
    dib.goto(0, 10)
    dib.write(f"Puntaje final: {puntaje}", align="center", font=("Arial",
16, "bold"))

    dib.goto(0, -40)
    dib.write("Presiona ENTER para volver al menú",
              align="center", font=("Arial", 14))

    pantalla.update()


# ---------------- CONTROLES ----------------
def iniciar_arcade():
    global estado, modo, tablero, puntaje, juego, ultimo_tiempo
    fondo.clear()
    tablero = [[0 for _ in range(COLUMNAS)] for _ in range(FILAS)]
    puntaje = 0
    juego = Juego()
    ultimo_tiempo = time.time()
    modo = "ARCADE"
    estado = "JUGANDO"
    iniciar_musica()

def iniciar_niveles():
    global estado, modo, nivel, velocidad_caida, tiempo_inicio
    global tablero, puntaje, juego, ultimo_tiempo
    fondo.clear()
    tablero = [[0 for _ in range(COLUMNAS)] for _ in range(FILAS)]
    puntaje = 0
    juego = Juego()
    ultimo_tiempo = time.time()
    nivel = 1
    velocidad_caida = VELOCIDAD_BASE
    tiempo_inicio = time.time()
    modo = "NIVEL"
    estado = "JUGANDO"
    iniciar_musica()
```

```
def volver_menu():
    global estado, modo, juego
    detener_musica()
    modo = None
    estado = "MENU"
    juego = Juego()

pantalla.listen()
pantalla.onkeypress(iniciar_arcade, "1")
pantalla.onkeypress(iniciar_niveles, "2")
pantalla.onkeypress(volver_menu, "Return")
pantalla.onkeypress(lambda: juego.mover(-1), "Left")
pantalla.onkeypress(lambda: juego.mover(1), "Right")
pantalla.onkeypress(lambda: juego.bajar(), "Down")
pantalla.onkeypress(lambda: juego.rotar(), "Up")
pantalla.onkeypress(lambda: juego.caer_directo(), "space")

# ---------------- LOOP ----------------
juego = Juego()

def loop():
    global ultimo_tiempo, tiempo_inicio, nivel, velocidad_caida, estado
    ahora = time.time()

    if estado == "MENU":
        dibujar_menu()

    elif estado == "JUGANDO":
        if ahora - ultimo_tiempo > velocidad_caida:
            juego.bajar()
            ultimo_tiempo = ahora

        if modo == "NIVEL" and ahora - tiempo_inicio >= TIEMPO_POR_NIVEL:
            if nivel < MAX_NIVELES:
                nivel += 1
                velocidad_caida *= 0.85
                tiempo_inicio = time.time()
            else:
                volver_menu()

        dibujar_juego()

    elif estado == "GAME_OVER":
        dibujar_game_over()
```

```
    pantalla.ontimer(loop, 16)

loop()
pantalla.mainloop()
```