TECNOLÓGICO NACIONAL

INATEC

CURSO DE PROGRAMACIÓN EN PYTHON

PROGRAMACIÓN EN PYTHON CON PYGAME

Estudiante: Sidar Fernando Rivas Berrios

Docente: Arnoldo Contreras.

Fecha: 08 de noviembre del 2025.

Contenido

Introducción:

Pygame es una biblioteca de Python diseñada para facilitar la creación de **videojuegos y aplicaciones multimedia interactivas**. Proporciona herramientas para gestionar gráficos, sonidos, animaciones, eventos de teclado, ratón y colisiones entre objetos, todo dentro de un entorno visual dinámico.

Con Pygame es posible crear desde simples animaciones hasta juegos completos en 2D. Su estructura se basa en un bucle principal donde se controlan los eventos del usuario, se actualiza la lógica del juego y se dibujan los elementos en pantalla. Entre sus ventajas destacan la facilidad de uso, su integración con Python y su compatibilidad multiplataforma.

En resumen, Pygame representa una excelente herramienta educativa y práctica para quienes desean adentrarse en el desarrollo de videojuegos utilizando Python.

Desarrollo:

1. Utilizando códigos proporcionados por el docente y mi creatividad, he procedido a la modificación del juego de la bolita y el bate de la siguiente manera:
    a. En primer lugar, al copiar el código proporcionado se presentaron errores como que había que instalar una librería y descargar imágenes acordes a las utilizadas en el código.

        pip install pygame

    b. Las imágenes utilizadas en el programa son las siguientes.
        i. bandera.png

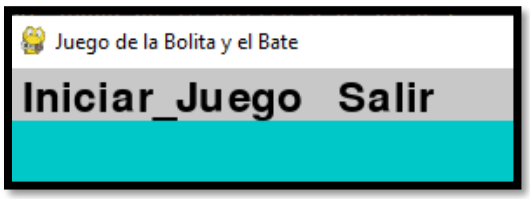
        ii. gion.png

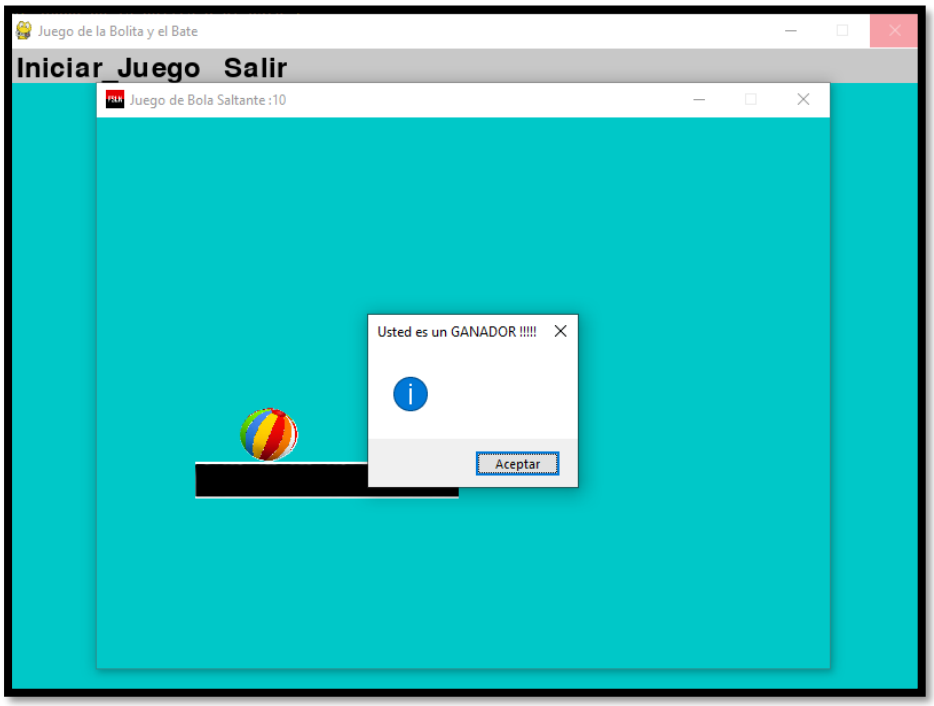
        iii. ball.png

2. Se procedió a modificar el color del fondo de la ventana y se crearon dos menús que dieran acceso a salir de la aplicación y a iniciar el juego nuevamente.



3. Se muestra un contador que cuando el jugador llega a diez puntos, el juego lo declara como ganador y se detiene el juego.



Como anexo, aunque no se pida, se colocó el código necesario para que cuando se anote un punto suene una campana y si pierde, hay un sonido del famoso juego de pacman de Game Over.

4. Se cambió el ícono de la ventana del juego a nuestra bandera del FSLN.

5. Se muestra una ventana principal con el menú y la ventana propia del juego, donde se hace uso de la librería “Import subprocess” y por medio de subproceso se llama el archivo py correspondiente al juego.

subprocess.run(["python", "Juego1.py"])

Código de los programas del Juego de la Pelota y el Bate:

Juego_bate_bolita.py

```
import pygame
import subprocess

# Inicialización de Pygame
pygame.init()

# Configuración de la pantalla
screen_width = 800
screen_height = 600
screen = pygame.display.set_mode((screen_width, screen_height))
pygame.display.set_caption("Juego de la Bolita y el Bate")


# Colores
WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
GRAY = (200, 200, 200)

# Fuente para el texto
font = pygame.font.Font(None, 36)

# Posición y tamaño de la barra de menú
menu_bar_height = 30
menu_bar_rect = pygame.Rect(0, 0, screen_width, menu_bar_height)

# Opciones de menú
menu_options = ["Iniciar_Juego", "Salir"]
menu_items_pos = []

# Dibujar la barra de menú
pygame.draw.rect(screen, GRAY, menu_bar_rect)

# Dibujar las opciones del menú
x_offset = 10
for i, option in enumerate(menu_options):
    text_surf = font.render(option, True, BLACK)
    text_rect = text_surf.get_rect(topleft=(x_offset, 5)) # Ajuste para la
altura del menú
    screen.blit(text_surf, text_rect)
    menu_items_pos.append((text_rect, option))
    x_offset += text_rect.width + 20 # Añadir espacio entre opciones

def reiniciar():
```

```
    subprocess.run(["python", "Juego1.py"])


# Bucle principal del juego
running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
        if event.type == pygame.MOUSEBUTTONDOWN:
            for rect, option in menu_items_pos:
                if rect.collidepoint(event.pos):
                    #print(f"Se ha hecho clic en: {option}")
                    # Aquí se debe añadir la lógica de cada menú (por
ejemplo, abrir un submenú)
                    if option == "Iniciar_Juego":
                        #print("Se Reinicia el Juego")
                        reiniciar()

                    elif option == "Salir":
                        exit()

    # Lógica de actualización de pantalla (borrar y redibujar para el juego
principal)
    # screen.fill((0, 0, 0)) # Rellena el fondo del juego principal

    # Dibujar la barra de menú (siempre se dibujará encima)
    pygame.draw.rect(screen, GRAY, menu_bar_rect)
    for i, option in enumerate(menu_options):
        text_surf = font.render(option, True, BLACK)
        text_rect = text_surf.get_rect(topleft=(menu_items_pos[i][0].x, 5))
# Mantener la misma posición X
        screen.blit(text_surf, text_rect)

    pygame.display.flip()
    screen.fill((0, 200, 200))


pygame.quit()
```

juego.py

```
from tkinter import messagebox
import sys
import pygame
pygame.mixer.init()


pygame.init()

ventana = pygame.display.set_mode((640, 480))
font = pygame.font.Font(None, 36) # Fuente por defecto y tamaño
icono = pygame.image.load("bandera.png")
pygame.display.set_icon(icono)
pygame.display.set_caption("Juego de Bola Saltante")
campana = pygame.mixer.Sound('campana.mp3')
perder = pygame.mixer.Sound("perder.mp3")

ball = pygame.image.load("ball.png")
ballrect = ball.get_rect()
speed = [4, 4]
ballrect.move_ip(0, 0)
# Crea el objeto bate, y obtengo su rectángulo
bate = pygame.image.load("guion.png")
baterect = bate.get_rect()
# Pongo el bate en la parte inferior de la pantalla
baterect.move_ip(200, 300)

#Comenzar el juego
contador = 0
perdedor = 10
jugando = True
while jugando:

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            jugando = False

    # Compruebo si se ha pulsado alguna tecla
    keys = pygame.key.get_pressed()
    if keys[pygame.K_LEFT]:
        baterect = baterect.move(-3, 0)
    if keys[pygame.K_RIGHT]:
        baterect = baterect.move(3, 0)

    # Compruebo si hay colisión
```

```
    if baterect.colliderect(ballrect) and ballrect.colliderect(baterect):
        speed[1] = -speed[1]
        #Acá es la colisión
        campana.play()
        contador = contador + 1
        pygame.display.set_caption("Juego de Bola Saltante" + " :" +
str(contador))
        if contador == 10:
            #Usted ha ganado
            messagebox.showinfo("Usted es un GANADOR !!!!!")
            sys.exit()

    #La Pelota no tocó el bate o pasó de su posición inferior
    if not ballrect.colliderect(baterect) and ballrect.top >
baterect.bottom:
        print("❌ La pelota pasó el bate")
        #Usted ha Perdido
        perder.play()
        messagebox.showinfo(f"Usted ha Perdido !!!!!")
        sys.exit()

    ballrect = ballrect.move(speed)
    if ballrect.left < 0 or ballrect.right > ventana.get_width():
        #Si la bola pegó en los bordes izquierdo o derecho de la ventana
        speed[0] = -speed[0]
    if ballrect.top < 0 or ballrect.bottom > ventana.get_height():
        #Si la bola pegó en los bordes superior o inferior de la ventana
        speed[1] = -speed[1]

    ventana.fill((0, 200, 200))
    ventana.blit(ball, ballrect)

    # Dibujo el bate
    ventana.blit(bate, baterect)
    pygame.display.flip()
    pygame.time.Clock().tick(60)


pygame.quit()
```

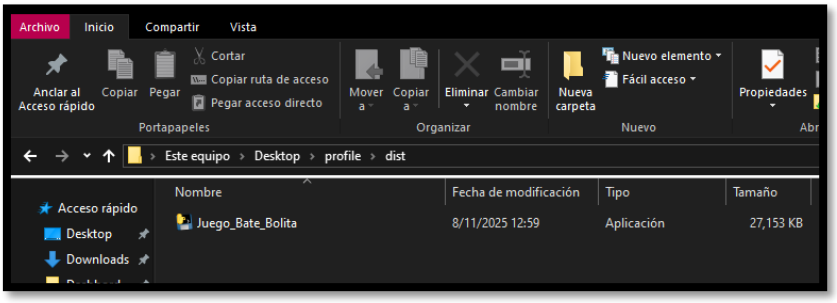
Creación de ejecutable:

Se procedió a la creación del archivo ejecutable, a partir de los fuentes “.py”, de la siguiente manera:

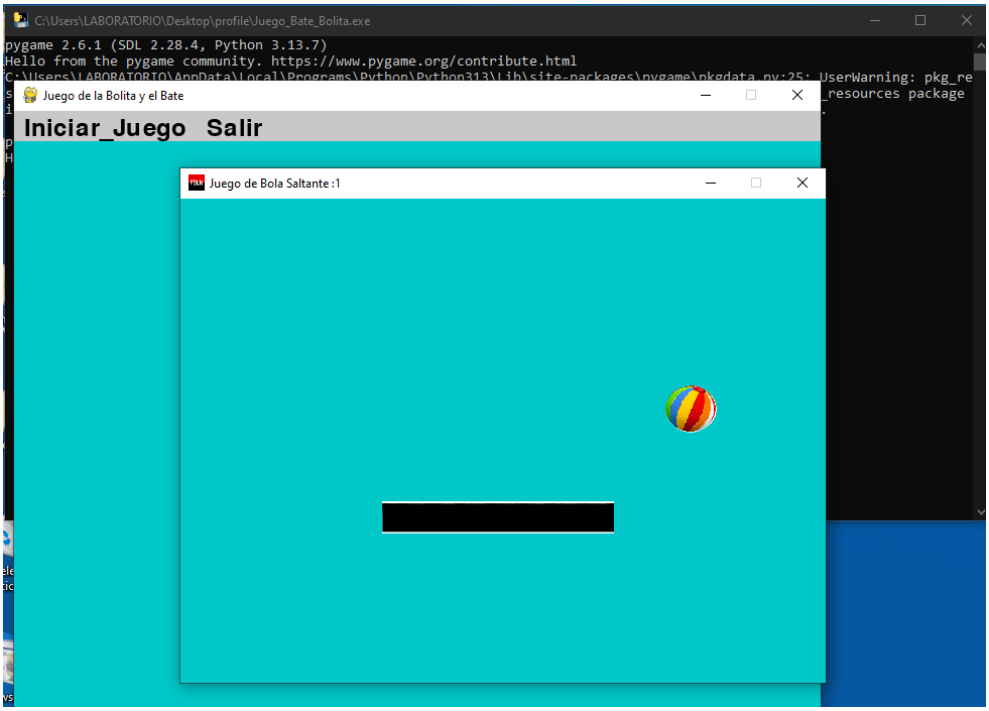
1. **Se utilizó la siguiente línea de código:**
   pyinstaller --onefile Juego_Bate_Bolita.py

2. **Se generaron dos carpetas:**
   - **dist.**
   - **build.**

3. **Estando en la carpeta dist, el ejecutable:**



Ejecución del Ejecutable:



En el caso del juego del ahorcado, se ejecutó código proporcionado por el docente basado en la impresión en consola y se adaptó utilizando la librería tkinter para modográfico.

Código del Ahorcado con tkinter:

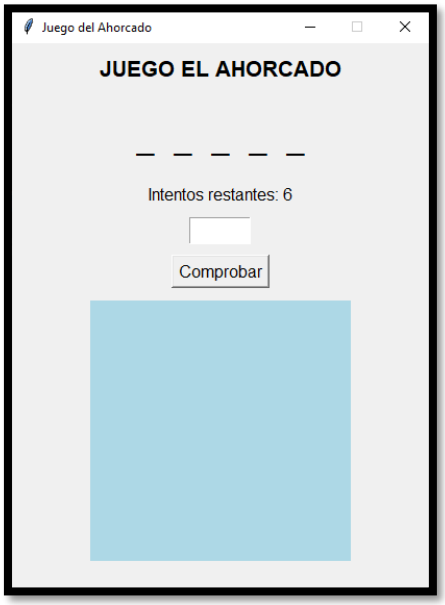
```
import tkinter as tk
from tkinter import messagebox
import random

# --- Palabras posibles ---
PALABRAS = ["SIDAR", "AHORCADO", "PYTHON", "JUEGO", "LEON", "NICARAGUA",
"INATEC"]

# --- Configuración del juego ---
palabra = random.choice(PALABRAS)
letras_adivinadas = []
intentos_restantes = 6

# --- Crear ventana principal ---
ventana = tk.Tk()
ventana.title("Juego del Ahorcado")
ventana.geometry("400x550")
ventana.resizable(False, False)

# --- Funciones del juego ---
def actualizar_palabra():
    """Muestra las letras adivinadas y oculta las no descubiertas."""
    palabra_mostrada = ""
    for letra in palabra:
        if letra in letras_adivinadas:
            palabra_mostrada += letra + " "
        else:
            palabra_mostrada += "_ "
    lbl_palabra.config(text=palabra_mostrada.strip())

def comprobar_letra():
    """Verifica si la letra ingresada está en la palabra."""
    global intentos_restantes

    letra = entrada_letra.get().upper()
    entrada_letra.delete(0, tk.END)

    # Validación
    if not letra.isalpha() or len(letra) != 1:
```

```
        messagebox.showwarning("Advertencia", "Introduce una sola letra
válida.")
        return

    if letra in letras_adivinadas:
        messagebox.showinfo("Aviso", "Ya adivinaste esa letra.")
        return

    letras_adivinadas.append(letra)

    if letra in palabra:
        actualizar_palabra()
        if all(l in letras_adivinadas for l in palabra):
            messagebox.showinfo("🎉 ¡Ganaste!", f"Adivinaste la palabra:
{palabra}")
            reiniciar_juego()
    else:
        intentos_restantes -= 1
        lbl_intentos.config(text=f"Intentos restantes:
{intentos_restantes}")
        dibujar_ahorcado()
        if intentos_restantes == 0:
            messagebox.showerror("💀 Fin del Juego", f"La palabra era:
{palabra}")
            reiniciar_juego()

def reiniciar_juego():
    """Reinicia el juego con una nueva palabra."""
    global palabra, letras_adivinadas, intentos_restantes
    palabra = random.choice(PALABRAS)
    letras_adivinadas = []
    intentos_restantes = 6
    lbl_intentos.config(text=f"Intentos restantes: {intentos_restantes}")
    canvas.delete("all")
    actualizar_palabra()

def dibujar_ahorcado():
    """Dibuja el ahorcado según los intentos restantes."""
    if intentos_restantes == 5:
        canvas.create_line(20, 230, 180, 230, width=3)  # base
    elif intentos_restantes == 4:
        canvas.create_line(100, 230, 100, 50, width=3)  # poste
    elif intentos_restantes == 3:
        canvas.create_line(100, 50, 180, 50, width=3)   # barra superior
        canvas.create_line(180, 50, 180, 70, width=3)   # cuerda
```

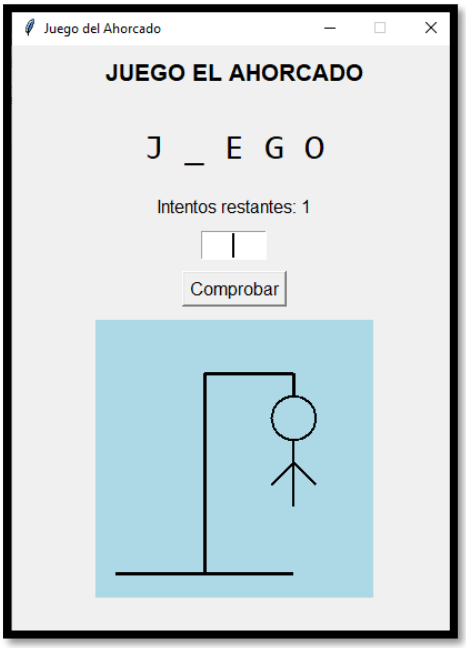
```
    elif intentos_restantes == 2:
        canvas.create_oval(160, 70, 200, 110, width=2)  # cabeza
    elif intentos_restantes == 1:
        canvas.create_line(180, 110, 180, 170, width=2)  # cuerpo
        canvas.create_line(180, 130, 160, 150, width=2)  # brazo izq
        canvas.create_line(180, 130, 200, 150, width=2)  # brazo der
    elif intentos_restantes == 0:
        canvas.create_line(180, 170, 160, 200, width=2)  # pierna izq
        canvas.create_line(180, 170, 200, 200, width=2)  # pierna der

# --- Controles ---
lbl_titulo = tk.Label(ventana, text="JUEGO EL AHORCADO", font=("Arial", 16,
"bold"))
lbl_titulo.pack(pady=10)

lbl_palabra = tk.Label(ventana, text="", font=("Consolas", 24))
lbl_palabra.pack(pady=20)

lbl_intentos = tk.Label(ventana, text=f"Intentos restantes:
{intentos_restantes}", font=("Arial", 12))
lbl_intentos.pack()

entrada_letra = tk.Entry(ventana, font=("Arial", 14), width=5,
justify="center")
entrada_letra.pack(pady=10)

btn_comprobar = tk.Button(ventana, text="Comprobar", font=("Arial", 12),
command=comprobar_letra)
btn_comprobar.pack()

canvas = tk.Canvas(ventana, width=250, height=250, bg="lightblue")
canvas.pack(pady=10)

# Inicializar la palabra oculta
actualizar_palabra()

# --- Ejecutar el juego ---
ventana.mainloop()
```

Se elige modo ramdom una palabra y se le proporcionan al usuario seis opciones de error, en cuanto a la adivinanza de letra que se corresponde a la palabra oculta.

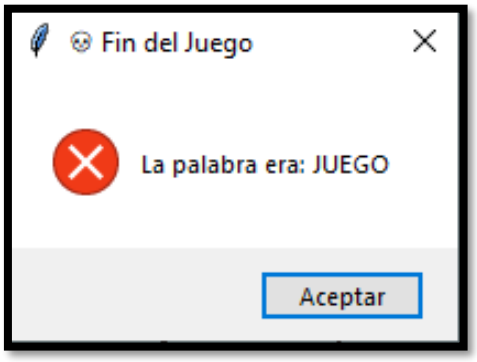
Ejecución del Ahorcado:



En la pantalla anterior se muestran en forma de guiones bajos la cantidad de letras que componen la palabra escondida, el usuario debe de ingresar la letra que considere en el cuadro de texto y luego dar al botón comprobar, si la letra está dentro de la palabra se colocará en uno de los guiones, sino está se irá formando el ahorcado.

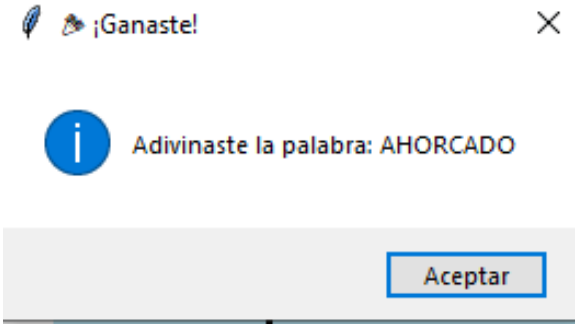


Si no se tiene éxito se completa la figura de ahorcado y se muestra un mensaje al usuario donde se notifica la palabra que estaba escondida.



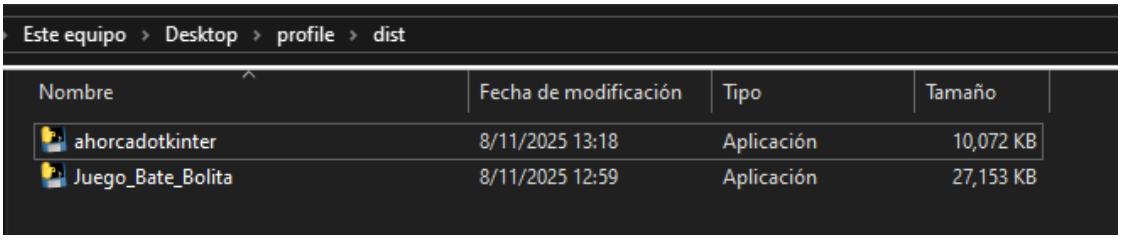
Se reinicia el juego automáticamente.

Y si se tiene éxito se muestra la siguiente pantalla:



De igual manera se realizó el correspondiente ejecutable:

pyinstaller --onefile ahorcadotkinter.py



Conclusión:

La elaboración del juego de **bolita y bate con Pygame** permitió aplicar de manera práctica los fundamentos de la programación gráfica y el control de eventos en Python.

A través del desarrollo se comprendió cómo gestionar el movimiento de objetos, detectar colisiones, controlar la interacción del usuario y mantener la actualización constante de la pantalla dentro de un bucle principal de juego.

Este proyecto demuestra que Pygame es una herramienta accesible y poderosa para iniciarse en el desarrollo de videojuegos, ya que combina la lógica de programación con la creatividad visual. Además, fomenta el razonamiento lógico y la resolución de problemas al integrar conceptos de física básica, control de flujo y diseño interactivo.