Desarrollo de un Juego de Pong

Niedferson José Vargas Rodriguez, Sara Sofia Delgado Cuevas
Facultad de ingeniería, Universidad Nacional de Colombia
Bogotá, Colombia
Email: nvargasro@unal.edu.co , sdelgadocu@unal.edu.co

***Abstract*—En este artículo se presenta el desarrollo de un juego de Pong en Python utilizando Tkinter para la interfaz gráfica, Pygame para la lógica del juego y Firebase para la gestión de datos en la nube. Se detallan los aspectos técnicos y las herramientas utilizadas, así como los desafíos encontrados y las soluciones implementadas.**

***Index Terms*—Pong, Python, Tkinter, Pygame, Firebase, desarrollo de videojuegos.**

I. Introducción

El desarrollo de videojuegos con Python ha ganado popularidad debido a la accesibilidad del lenguaje y la gran variedad de bibliotecas especializadas disponibles. En este contexto, se diseñó e implementó una versión del clásico Pong con el objetivo de integrar diferentes tecnologías dentro de un mismo proyecto.

Este trabajo busca demostrar cómo es posible combinar herramientas como Tkinter, Pygame y Firebase para crear un videojuego interactivo con almacenamiento de datos en la nube. Tkinter se utilizó para la interfaz gráfica, proporcionando una estructura sencilla pero funcional. Pygame se encargó de la mecánica del juego, asegurando una jugabilidad fluida con detección de colisiones y respuesta de los elementos en pantalla. Finalmente, Firebase permitió almacenar los puntajes de los jugadores en tiempo real, facilitando la gestión de datos a nivel remoto.

El desarrollo de este proyecto se llevó a cabo con el objetivo de explorar el potencial de Python en el ámbito de los videojuegos y evaluar la integración de diferentes tecnologías dentro de un entorno interactivo

II. Metodología

Se utilizó el lenguaje Python y tres bibliotecas principales:

- **Tkinter**: para la creación de la interfaz gráfica.
- **Pygame**: para la mecánica del juego, como el movimiento de la pelota y las colisiones.
- **Firebase**: para almacenar y recuperar puntuaciones en tiempo real.

El desarrollo se llevó a cabo en las siguientes etapas:

1) Diseño de la interfaz con Tkinter.
2) Implementación de la lógica del juego con Pygame.
3) Integración con Firebase para gestionar los puntajes.

III. Estructura del Código y Funcionalidades

El juego de Pong desarrollado se basa en una estructura modular que facilita su mantenimiento y extensión. A continuación, se describen los componentes principales:

A. Módulo de Lógica del Juego (`game_logic.py`)

Este módulo gestiona las puntuaciones altas y las configuraciones de juego, interactuando con Firebase para almacenar y recuperar datos. Las funciones principales incluyen:

- **`update_high_score(new_high_score)`**: Verifica si el nuevo puntaje supera el puntaje más alto registrado. Si es así, actualiza el valor en Firebase.
- **`view_high_score()`**: Recupera el puntaje más alto asociado al perfil actual desde Firebase y lo muestra.
- **`delete_high_score()`**: Elimina el puntaje alto almacenado en Firebase, restableciéndolo a cero.
- **`save_game_configuration(profile, config)`**: Guarda la configuración personalizada de un perfil en Firebase.
- **`load_game_configuration(profile)`**: Recupera la configuración de juego asociada a un perfil desde Firebase.

B. Implementación del Juego con Pygame (`pygame_game.py`)

Este módulo define la lógica del juego Pong, implementando diferentes modos de juego y configuraciones. Los parámetros generales incluyen:

Dimensiones y parámetros básicos

- `WIDTH`, `HEIGHT`: Dimensiones de la pantalla.
- `PAD_WIDTH`, `PAD_HEIGHT`: Tamaño de las paletas.
- `BALL_SIZE`, `BALL_SPEED`, `MAX_BALL_SPEED`: Parámetros relacionados con la pelota.
- `PADDLE_SPEED`: Velocidad de las paletas.

C. Modos de juego

- **`run_pong_classic()`**: Implementa la versión clásica de Pong.
- **`run_pong_speed()`**: Aumenta progresivamente la velocidad de la pelota durante el juego.
- **`run_pong_inverted_controls()`**: Invierte los controles de las paletas periódicamente.
- **`run_pong_custom(config)`**: Permite jugar con configuraciones personalizadas cargadas desde Firebase.

Cada modo de juego incluye un bucle principal (`game_loop()`) que gestiona eventos de teclado, dibuja los elementos del juego en pantalla y controla las colisiones y puntuaciones.

D. Interfaz de Usuario con Tkinter (`tkinter_ui.py`)

Este módulo maneja la interfaz gráfica del usuario, permitiendo la selección de perfiles y configuraciones de juego. Sus características principales son:

Inicialización y navegación

- **`start_tkinter_ui()`**: Inicializa la ventana principal de la interfaz.
- **`show_game_modes()`**, **`show_options()`**, **`exit_game()`**: Gestionan la navegación entre las opciones de la interfaz.

Inicio de modos de juego

- **`start_game_classic()`**, **`start_game_speed()`**, **`start_game_inverted_controls()`**, **`start_game_custom()`**: Inician los diferentes modos de juego desde la interfaz.

Configuración personalizada

- **`configure_general()`**, **`configure_classic()`**, **`configure_speed()`**, **`configure_inverted_controls()`**, **`configure_custom()`**: Permiten personalizar las configuraciones del juego según el perfil seleccionado.

Gestión de perfiles

- **`select_profile()`**: Permite seleccionar un perfil y cargar su configuración desde Firebase.

E. Integración con Firebase

El juego utiliza Firebase como backend para almacenar y recuperar datos en tiempo real. Las principales interacciones incluyen:

Autenticación

- **`credentials.Certificate("credencial.json")`**: Autentica la aplicación con Firebase utilizando un archivo de credenciales.

Gestión de datos

- **`db.reference()`**: Proporciona acceso a la base de datos en tiempo real para leer y escribir datos.
- **`save_game_configuration(profile, config)`**: Guarda configuraciones personalizadas de un perfil en Firebase.
- **`load_game_configuration(profile)`**: Recupera configuraciones guardadas para un perfil específico.

Aun así resulta necesario que antes de iniciar el juego, se cree una clave privada en Firebase, pues la credencial se requiere para poder iniciar sesión en el juego, esta ultima se agrega en la carpeta donde se encuntra el juego.

IV. Resultados

El juego desarrollado permite a dos jugadores competir en tiempo real en una experiencia interactiva y dinámica. Gracias a la combinación de Tkinter y Pygame, se logró una jugabilidad fluida con respuesta rápida a las interacciones de los jugadores. La implementación de Firebase permitió registrar y recuperar puntajes en la nube, ofreciendo un sistema de almacenamiento confiable y accesible desde diferentes dispositivos.

Entre los principales logros del proyecto, destacan:

- **Interfaz gráfica funcional**: Se desarrolló un entorno visual simple pero intuitivo, que facilita la interacción de los jugadores.
- **Jugabilidad optimizada**: Se logró una experiencia fluida con detección de colisiones precisa y tiempos de respuesta adecuados.
- **Gestión de datos en la nube**: La integración con Firebase permitió almacenar puntajes de manera eficiente, asegurando su persistencia y disponibilidad y disponibilidad en diferentes sesiones de juego.

Uno de los principales desafíos encontrados fue la sincronización entre Pygame y Tkinter, debido a sus diferentes enfoques en la gestión de eventos y gráficos. Se resolvió mediante el uso de hilos, lo que permitío ejecutar ambas tecnologías de manera simultánea sin afectar el rendimiento.

V. Conclusiones

Este proyecto demostró la viabilidad de combinar diferentes herramientas de Python para el desarrollo de un videojuego interactivo. La integración de Tkinter y Pygame permitió una experiencia de usuario fluida y accesible, mientras que Firebase aportó una solución eficiente para la gestión de datos en la nube.

Se evidenció que Python es una opción versátil para el desarrollo de videojuegos, incluso en proyectos que requieren la combinación de múltiples bibliotecas. Para futuras mejoras, se propone la implementación de inteligencia artificial para crear un oponente controlado por computadora, así como la optimización del juego para su compatibilidad con dispositivos móviles.

Este trabajo representa una exploración inicial en la integración de herramientas de desarrollo de videojuegos en Python y abre la posibilidad de expandir sus funcionalidades en proyectos más complejos.

References

[1] GitHub Documentation. Disponible en: https://github.com/Nied-vr/Pong-

[2] Firebase Documentation. Disponible en: https://pong-3e97f-default-rtdb.firebaseio.com/