Facultad Comunitaria Caacupé

Ingeniería Informática

Trabajo Práctico

Nombre y Apellido: José Renato Villagra Patiño

Profesor: Ing. Ricardo Maidana

Materia: Lenguajes Visuales II

Fecha de entrega: 5/11/2025

Tema: Bizarro_Slayer

Curso: 2° **Semestre**: 4°

AÑO: 2025

2. Introducción

El presente documento describe el desarrollo del proyecto Bizarro Slayer, una aplicación de escritorio realizada en Python, que combina Tkinter y Pygame como entornos gráficos para ofrecer un juego interactivo con interfaz de selección, combate por oleadas y registro de progreso.
El proyecto forma parte del Trabajo Final de la asignatura Lenguajes Visuales II, integrando diseño visual, programación modular y almacenamiento persistente en archivos CSV.

3. Objetivo del Proyecto

Objetivo general:
Desarrollar una aplicación funcional en Python que combine una interfaz gráfica de usuario (GUI) con un entorno de juego 2D, permitiendo gestionar partidas, registrar datos y ofrecer una experiencia interactiva completa.

Objetivos específicos:

Aplicar principios de diseño visual y modularidad en Python.

Crear interfaces intuitivas con Tkinter.

Desarrollar un sistema de juego por oleadas con Pygame.

Implementar guardado de progreso en formato CSV.

Asegurar la interacción fluida entre módulos.

Fomentar la escalabilidad mediante arquitectura por componentes.

---

4. Descripción General del Proyecto

Bizarro Slayer es una aplicación que integra una interfaz de selección de personaje, un menú principal, un sistema de historial y un entorno de juego 2D.
El flujo del programa comienza en el menú principal, pasa por la selección de ventajas y desventajas del jugador, y finalmente inicia una partida controlada con teclado dentro de Pygame.

Módulos principales:

1. `bizarro_slayer_full_menu.py` — Ventana principal y navegación.
2. `bizarro_slayer_historial.py` — Registro y visualización de partidas.
3. `bizarro_slayer_selector.py` — Selección de pros y contras del jugador.
4. `bizarro_slayer_game.py` — Lógica del juego, combate y progresión.

Características técnicas:

- Lenguaje: Python 3.x
- Bibliotecas: Tkinter y Pygame
- Almacenamiento: CSV (`historial.csv`)
- Arquitectura: modular, separando interfaz, lógica y persistencia.
- Estilo visual: paleta oscura (#2c3e50 / #34495e) y sprites 2D dibujados con Pygame.

---

5. Desarrollo del Sistema

5.1 **Menú Principal** (`bizarro_slayer_full_menu.py`)

El menú principal inicia la aplicación, presenta el título y permite elegir entre comenzar una nueva partida, abrir el historial o cerrar el programa.
El diseño usa `ttk.Style` para aplicar fuentes y márgenes personalizados, y centra la ventana automáticamente en pantalla.
Desde este módulo se gestionan las transiciones hacia el selector y el historial mediante `withdraw()` y `deiconify()` para mantener activa una sola ventana principal.

5.2 **Módulo Historial** (`bizarro_slayer_historial.py`)

Gestiona el registro de los jugadores y su visualización.
Lee y escribe los datos del archivo `historial.csv`, mostrando una tabla con `ttk.Treeview`.
Incluye botones para actualizar, limpiar y cerrar.
Además, genera estadísticas básicas (cantidad total de jugadores, máximo puntaje y mayor número de victorias).
También se valida la existencia del archivo con `ensure_save_file()` y se ordenan los datos por puntaje con `load_top_players()`.

5.3 **Módulo Selector** (`bizarro_slayer_selector.py`)

Este módulo representa la interfaz donde el jugador define sus pros y contras antes de comenzar la partida.
Está desarrollado completamente con Tkinter, usando `ttk` para estilos modernos.

Estructura general:

- Clase principal: `SelectorApp`
- Ventana principal con título, campos de nombre, y dos listas de selección (pros / contras).
- Cada atributo se asocia a un diccionario de valores estadísticos que modifican las capacidades del jugador (ataque, vida, velocidad, crítico, evasión, regeneración, etc.).

Lógica funcional:

- Máximo de tres pros y tres contras seleccionables.
- Uso de variables `tk.IntVar()` para controlar los checkboxes.
- Contador dinámico que muestra el número actual de selecciones.
- Validaciones con `messagebox` para avisar si se supera el límite o si no se elige nada.
- Al confirmar, destruye la ventana del selector e importa el módulo de juego (`from bizarro_slayer_game import iniciar_juego_pygame`), pasando el nombre del jugador y las listas seleccionadas.

Rol dentro del sistema:
Es el puente entre el entorno de menús de Tkinter y el entorno interactivo de Pygame. Permite personalizar las estadísticas iniciales y define las ventajas/desventajas que influirán en el desarrollo del juego.

5.4 **Módulo de Juego** (`bizarro_slayer_game.py`)

Es el núcleo de la aplicación y contiene toda la lógica jugable en Pygame, incluyendo movimiento, combate, oleadas de enemigos, progresión del jugador y guardado automático.

Principales componentes:

- **Configuración y constantes:**
  Inicialización de Pygame, tamaño de pantalla, FPS y colores base.
- **Sprites dibujados por código:**
  El jugador y los enemigos se generan con primitivas gráficas (`pygame.draw.rect`, `pygame.draw.circle`), sin imágenes externas.
  Hay tres tipos de enemigos: *Rana Prisma*, *Molusco Óseo* y *Gusano de Reloj*, cada uno con sus propios atributos y colores.
- **Clases principales:**
  - **`Boton`**: gestiona botones visuales durante las pantallas de pausa y victoria.
  - **`SistemaOleadas`**: controla la cantidad de enemigos por oleada, el temporizador entre ellas y la dificultad creciente.
  - **`Player`**: representa al jugador, con atributos personalizables según los pros y contras seleccionados.
    Incluye estadísticas (HP, ataque, velocidad, regeneración, evasión, oro, nivel) y métodos para moverse, atacar, curarse y subir de nivel.
  - **`Monster`**: controla los enemigos, su inteligencia básica, movimiento hacia el jugador, ataques y muerte.
    Cada monstruo escala sus estadísticas con el número de oleada.

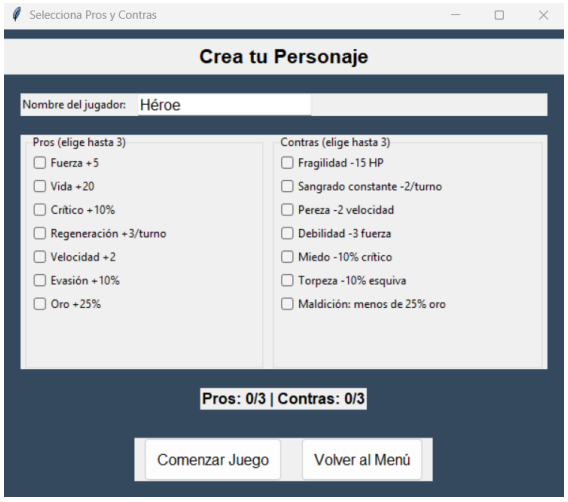
- **Funciones complementarias:**
  - `generar_monstruos_oleada()`: crea una lista de enemigos aleatorios por oleada.
  - `mostrar_pantalla_pausa()` y `mostrar_pantalla_victoria()`: muestran interfaces dentro del juego sin salir del entorno.
  - `save_player()`: registra automáticamente el progreso en el archivo `historial.csv`.
- **Lógica principal (`iniciar_juego_pygame`):**
  - Inicializa pantalla, jugador y oleadas.
  - Administra el bucle de eventos, movimiento, colisiones y ataques.
  - Controla el sistema de pausa y los estados "explorar", "batalla" y "victoria".
  - Gestiona subidas de nivel, experiencia, oro y guardado al salir.

Interacción general:
El jugador se mueve con WASD o flechas, ataca con E, se cura con H (con tiempo de espera), y puede pausar con ESC o P.
El sistema de oleadas incrementa progresivamente la dificultad, y cada victoria muestra una pantalla de estadísticas antes de continuar o salir.

6. Capturas de Pantalla

Menú Principal

Selector de Pros y Contras



Escena del juego (Pygame) mostrando combate y barras de vida

Archivo `historial.csv` con datos registrados.

7. Organización del Código

El proyecto completo queda organizado del siguiente modo:

```
Examen de LV II/
_pycache_
bizarro_slayer_full_menu.py
bizarro_slayer_historial.py
bizarro_slayer_selector.py
bizarro_slayer_game.py
historial.csv
```

Esta estructura modular favorece la mantenibilidad, la comprensión del flujo entre interfaces y la evaluación independiente de cada componente.

---

8. Gestión de Datos

El archivo historial.csv almacena la información persistente de los jugadores con las columnas:
Jugador | Puntos | Victorias | Pros | Contras
Los pros y contras se guardan separados por “;” para su posterior lectura desde el historial.

9. Conclusión

El desarrollo de Bizarro Slayer permitió integrar de manera efectiva Tkinter (para la interfaz de usuario) y Pygame (para la lógica de juego en tiempo real).
El sistema demuestra competencias en programación modular, eventos, control de estados, persistencia de datos y diseño visual.
Cada módulo cumple una función específica dentro del flujo de ejecución, manteniendo cohesión y claridad estructural.
El resultado es un proyecto completo, funcional y ampliable, adecuado como entrega final de la asignatura Lenguajes Visuales II.

Enlaces de Github y Trello

https://github.com/Renato8bit/proyecto_final_LV_II

https://trello.com/invite/b/68f8c4cff3e5ad930fd51ae2/ATTI2819dedfe996986afd4d8ae4b3724a3e2C570940/proyecto-final-lenguajes-visuales