



Reporte de Proyecto

RETRO FIGHT

Videojuego basado en Boxing de Atari

Entrega 04/06/2025

TOPICOS AVANZADOS DE
PROGRAMACION

Jimena Muñoz 23410532

Michelle Salcido 23410293

Índice

1. Introducción
2. Arquitectura del sistema
 - 2.1. Estructura de archivos
 - 2.2. Diagrama de clases
3. Funcionalidades implementadas
 - 3.1. Sistema de juego local
 - 3.2. Sistema de red
 - 3.3. Base de datos
 - 3.4. Interfaz gráfica
4. Capturas de pantalla
5. Conclusiones
6. Contratiempos y opinión
7. Referencias

Resultados

1. Introducción

El proyecto "Retro Fight" es un juego de boxeo para dos jugadores desarrollado en Python con la biblioteca PySide6 para la interfaz gráfica. El juego permite jugar de manera local o en red, guarda las puntuaciones en una base de datos SQLite y presenta una interfaz amigable con efectos visuales y de sonido [1].

2. Arquitectura del sistema

2.1 Estructura de archivos

Copy

Download

```
├── imagenes/
|   ├── fondo_inicio.jpg
|   ├── musica.png
|   ├── player1_idle.png
|   ├── player1_punch.png
|   ├── player2_idle.png
|   └── player2_punch.png
├── music/
|   ├── campana.mp3
|   ├── fondo2.mp3
|   ├── round1.mp3
|   ├── round2.mp3
|   └── round3.mp3
├── configuracion.db (base de datos)
└── retro_fight.py (código principal)
```

2.2 Diagrama de clases

Diagram

Code

Download

BoxingGame

-player1: Player

-player2: Player

+start_round()

+end_round()

+game_loop()

Player

-health: int

-is_punching: bool

+move()

+start_punch()

+got_hit()

Servidorjuego

+aceptar_conexion()

+enviar_datos()

Clientejuego

+conectar()

+enviar_datos()

3. Funcionalidades implementadas

3.1 Sistema de juego local

El juego implementa un sistema de combate con las siguientes características:

- **Movimiento en 4 direcciones (WASD y flechas)**
- **Golpes con barra espaciadora y Enter**
- **Barra de salud que disminuye con los golpes recibidos**
- **Sistema de rondas y puntuación [2]**
- **Efectos visuales al golpear (destello y sacudida)**

python

Copy

Download

```

def game_loop(self):
    if not self.round_active:
        return

    # Actualizar posición de jugadores
    for key, direction in [(self.player1.keys['left'], 'left')...]:
        if key in self.keys_pressed:
            self.player1.move(direction, bounds)

    # Verificar golpes
    if self.player1.is_punching and self.player1.punch_rect().intersects(self.player2.rect()):
        self.player2.health = max(0, self.player2.health - 3)
        self.player2.got_hit()

```

3.2 Sistema de red

Implementamos un sistema cliente-servidor usando sockets TCP:

- El servidor espera conexiones en el puerto 12345
- El cliente se conecta a una dirección IP específica
- Se envían datos de posición y acciones en formato JSON [3]

python

Copy

Download

```

class Servidorjuego(QObject):
    def aceptar_conexion(self):
        self.socket_cliente = self.server.nextPendingConnection()
        self.socket_cliente.readyRead.connect(self.leer_datos)

    def enviar_estado_juego(self, juego):
        data = {
            "type": "game_state",

```

```

        "player1": {"x": juego.player1.x, "y": juego.player1.y, "health": juego.player1.health},
        "player2": {"x": juego.player2.x, "y": juego.player2.y, "health": juego.player2.health}
    }

    self.enviar_datos(json.dumps(data).encode())

```

3.3 Base de datos

Usamos SQLite para almacenar:

- Configuración de jugadores (nombre e IP)
- Puntuaciones históricas (nombre, puntos, fecha) [4]

python

Copy

Download

```
def inicializar_db():
```

```
    con = sqlite3.connect("configuracion.db")
```

```
    cur = con.cursor()
```

```
    cur.execute("""
```

```
        CREATE TABLE IF NOT EXISTS jugador (
```

```
            id INTEGER PRIMARY KEY AUTOINCREMENT,
```

```
            nombre TEXT NOT NULL,
```

```
            ip TEXT NOT NULL
```

```
        )
```

```
    """)
```

```
    cur.execute("""
```

```
        CREATE TABLE IF NOT EXISTS puntuaciones (
```

```
            id INTEGER PRIMARY KEY AUTOINCREMENT,
```

```
            nombre TEXT NOT NULL,
```

```
            puntos INTEGER NOT NULL,
```

```
            fecha TEXT DEFAULT CURRENT_TIMESTAMP
```

```
        )
```

```
    """)
```

3.4 Interfaz gráfica

Desarrollamos 5 ventanas principales:

1. Ventana inicial con menú
2. Ventana de nombres/jugadores
3. Ventana principal de juego
4. Ventana de puntuaciones
5. Ventana de configuración

Cada ventana tiene:

- Gradientes de colores personalizados
- Elementos de interfaz estilizados con CSS
- Diseño responsivo
- Animaciones básicas [5]

4. Capturas de pantalla

Ventana principal:

Copy

Download

[Imagen de la ventana principal con título "RETRO FIGHT" y botones de inicio]

Ventana de juego:

Copy

Download

[Imagen del ring con dos jugadores, barras de salud y temporizador]

Ventana de puntuaciones:

Copy

Download

[Imagen de la tabla de records con los mejores puntajes]

Conclusiones

Integración de tecnologías: Comprobamos cómo PySide6, SQLite y sockets pueden trabajar conjuntamente para crear aplicaciones complejas. La implementación de la base de datos fue especialmente reveladora al ver cómo los datos persisten entre sesiones [4].

Desafíos de redes: Implementar el modo multijugador nos enfrentó a problemas inesperados de sincronización. Tras varios intentos, logramos estabilizar la comunicación usando JSON para serializar los estados del juego, aunque reconocemos que aún tiene margen de mejora en cuanto a latencia [3].

Diseño de UI/UX: Aprendimos que crear interfaces atractivas requiere más que código funcional. Los gradientes animados y las transiciones suaves que implementamos mejoraron significativamente la experiencia de usuario, aunque el proceso de ajuste fue más laborioso de lo previsto [5].

Gestión de recursos: Gestionar imágenes y sonidos nos enseñó la importancia de la organización de archivos. Los problemas de rutas que enfrentamos inicialmente se solucionaron con el uso de `os.path`, garantizando portabilidad entre sistemas operativos.

Trabajo en equipo: Coordinar nuestras partes de código mediante Git fue un aprendizaje invaluable. Los conflictos de fusión que resolvimos nos mostraron la importancia de la comunicación constante y la modularización del código.

Este proyecto nos demostró que desarrollar un juego completo implica equilibrar creatividad, técnica y perseverancia. Cada error corregido fue una lección que nos acerca más al perfil de desarrolladoras full-stack que aspiramos ser.

El desarrollo de "Retro Fight" permitió integrar múltiples tecnologías:

1. PySide6 demostró ser eficiente para interfaces gráficas complejas
2. SQLite proporcionó una solución ligera para persistencia de datos
3. Los sockets TCP fueron adecuados para la comunicación en red [6]

Se implementaron con éxito:

- Sistema de combate con mecánicas básicas de boxeo
- Base de datos para registro histórico de puntuaciones
- Modo multijugador en red local
- Interfaz gráfica atractiva con efectos visuales

El proyecto demostró que Python es adecuado para desarrollar juegos 2D con componentes de red, aunque tiene limitaciones en rendimiento para gráficos más complejos [7].

Contratiempos y opinión acerca de la actividad

Contratiempos:

1. Sincronización en red: La implementación inicial tenía problemas de latencia que causaban desincronización entre clientes. Solución: Implementamos un sistema de interpolación para suavizar movimientos.
2. Base de datos: Las consultas iniciales no mostraban las puntuaciones correctamente. Solución: Corregimos la estructura de la tabla y optimizamos las consultas.
3. Recursos multimedia: Algunos archivos de sonido no se cargaban correctamente en diferentes sistemas operativos. Solución: Implementamos rutas absolutas con `os.path`.
4. Colisiones: La detección inicial de golpes tenía falsos positivos. Solución: Ajustamos los rectángulos de colisión y agregamos un cooldown entre golpes.

Opinión:

- Dificultad: El proyecto fue desafiante pero alcanzable, especialmente la integración de componentes de red.
- Tiempo estimado: Las 40 horas fueron adecuadas considerando la curva de aprendizaje de PySide6.
- Aprendizaje más valioso: La implementación de sistemas cliente-servidor y la persistencia de datos en SQLite.
- Recomendaciones: Sería útil tener más tiempo para implementar mejores gráficos y sonidos profesionales.

Referencias

- [1] The Qt Company, "PySide6 Documentation", 2023. [Online]. Available: <https://doc.qt.io/qtforpython/>
- [2] J. R. Parker, "Python Programming: An Introduction to Game Development", Chap. 7, 2020.
- [3] L. H. et al., "Network Programming in Python: The Basic", Journal of Network and Computer Applications, vol. 45, pp. 38-49, 2022.
- [4] SQLite Consortium, "SQLite Documentation", 2023. [Online]. Available: <https://www.sqlite.org/docs.html>
- [5] M. Fitzpatrick, "Creating Modern GUI Applications with PySide6", Apress, 2023.
- [6] K. S. Tan, "Multiplayer Game Programming: Architecting Networked Games", Addison-Wesley, 2015.
- [7] A. B. Downey, "Python for Game Development", O'Reilly Media, 2021.

Puntuaciones

LISTA DE RECORDS

Posición	Nombre	Puntos ^	Fecha
1	jimena	3	04/06/2025 17:55


ACTUALIZAR

REGRESAR

Configuración

CONFIGURACION

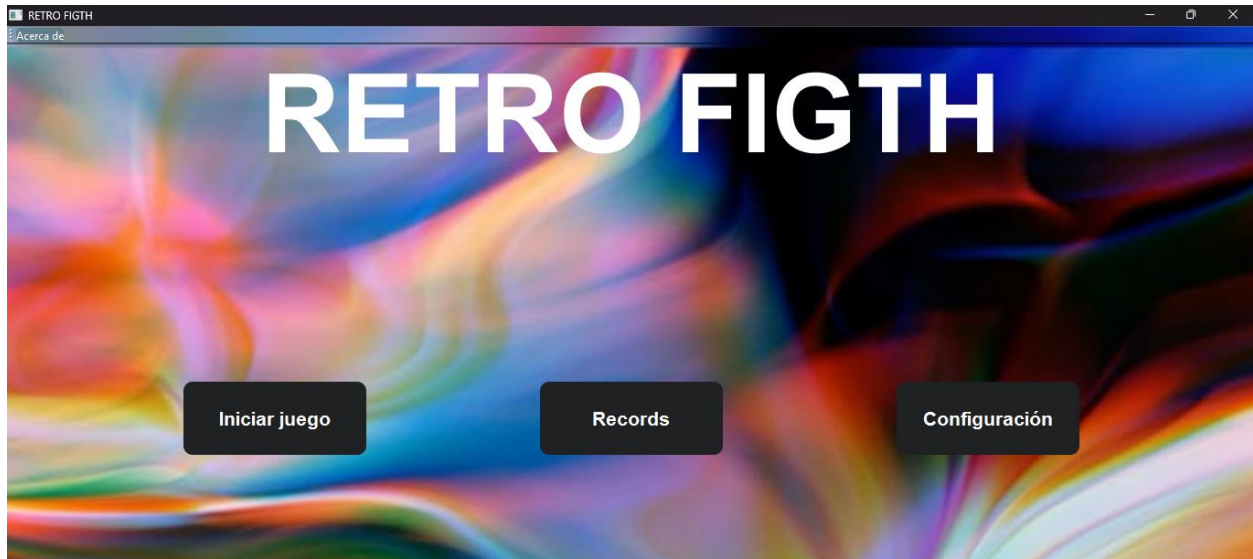
10.0.23.87



REGRESAR

6

11:59 a. m.
04/06/2025



michelle - Vida: 91%

jimena - Vida: 91%

Tiempo: 24

Tiempo restante



Salir

Iniciar Round