Universidad Nacional de La Matanza
Florencio Varela 1903 - San Justo - Buenos Aires - Argentina

Departamento de Ingeniería e Investigaciones Tecnológicas

Cátedra de Programación Concurrente

Trabajo Práctico Integrador

Integrantes:

DNI	Apellido	Nombre
43.630.151	Antonioli	Iván Oscar
43.664.669	Di Nicco	Luis Demetrio
41.548.235	Sandoval	Juan Leandro
32.788.835	Tigani	Martin Sebastián
35.277.730	Villca	Luis Alberto

Indice

Enunciado 3
Enlaces 3
Introducción 3
Descripción 3
Descripción general de la solución 3
Características de la implementación 4
Uso de Procesos, Hilos, Sincronización y Comunicación 5
Limitaciones 5
Manual de Usuario 5
Conclusiones 13

Enunciado

Fecha de entrega: 13/11/2024

Forma de entrega: Se deberá generar un informe donde contenga los siguientes puntos

- **Carátula**: Con los integrantes del grupo
- **Link a un repositorio de GitHub**: subir el código fuente, proyecto, etc., así como cualquier otro archivo que crea conveniente.
- **Descripción**: Una breve descripción sobre la solución de software que implementaron; que finalidad tiene, lenguaje utilizado, frameworks, si tiene restricciones de hardware, versión de sistema operativo, de navegador, etc.
- **Manual de Usuario**: Una guía sobre cómo utilizar la solución software, pueden incluir capturas, paso a paso, etc. Puede ser un video también y directamente en el documento agregan el link a donde se encuentra el archivo de video.
- **Conclusiones**: En esta sección se debe describir las dificultades que encontraron al realizar el trabajo práctico

Entregar el informe por plataforma MIeL. Este debe ser en formato .pdf, con nombre TP1_Integrador_NumerodelGrupo.pdf.

Enunciado: Implementar una solución software, a elección del grupo, que integre procesos/hilos que se comuniquen y sincronicen entre sí. Puede apegarse a lo visto en las partes 1, 2 y 3 del TP1, utilizando los lenguajes vistos y utilizando Google Colab o pueden proponer ustedes un lenguaje, framework, etc. A continuación, se citan los trabajos de cursadas previas como para que tener una mejor idea, pero desde ya el límite/alcance y el tema lo define el grupo.

Enlaces

- Link al repositorio de GitHub de la catedra
- Link al repositorio de GitHub local

Introducción

En este trabajo practico integrador, aplicaremos todos los conceptos aprendidos durante la cursada acerca la programación concurrente con el objetivo de desarrollar una solución de software que integre los conocimientos adquiridos sobre procesos, hilos y mecanismos de sincronización y comunicación.

Descripción

Descripción general de la solución

La solución de software desarrollada es un juego multijugador de batalla por turnos inspirado en los combates de los juegos de Pokémon. El objetivo de nuestra solución era crear un juego de combates agiles y entretenidos con una interfaz intuitiva y minimalista pero que, a su vez, requiera cierto grado de estrategia.

Decidimos utilizar una arquitectura cliente-servidor para poder separar de una manera más clara la lógica de las batallas de la lógica de la interfaz de usuario. Cada jugador entra al juego desde su

propio cliente y se conecta al servidor para crear una sesión de batalla. El servidor, además de manejar la lógica y sincronización de la batalla, se encarga de recibir información proveniente de cada jugador (personaje elegido, acción del turno, etc.) y comunicárselo al contrincante. Estas comunicaciones entre cliente-servidor se hacen mediante sockets.

Se juega de a dos personas, en donde cada una puede elegir un personaje a elección. Una vez elegidos los personajes, empieza la batalla. En cada turno, los jugadores pueden realizar una de las siguientes 4 acciones:

- **Atacar**: Su personaje realiza un ataque básico al contrincante.
- **Defenderse**: Su personaje aumenta la armadura, lo que reduce el daño recibido en un 50% para el próximo ataque recibido.
- **Descansar**: Su personaje recupera salud en hasta un 20% de su salud máximo.
- **Concentrarse**: Su personaje recibe una bonificación de ataque del 250% para el próximo ataque.

Gana el jugador que logre derrotar al personaje del jugador enemigo. Luego de finalizada la batalla los jugadores son llevados al menú de selección de personajes en caso de que quieran volver a jugar nuevamente.

Características de la implementación

Para desarrollar el juego decidimos utilizar Python como lenguaje de programación debido a la versatilidad y facilidades que ofrece la gran cantidad de librerías existente. Principalmente, nos utilizamos las siguientes librerías:

- Pygame: Para todo lo relacionado con el aspecto visual del juego.
- Threading: Para la generación, comunicación (sockets) y sincronización (mutex y barreras) de los hilos generados.

Para conseguir los sprites y las estadísticas de los personajes utilizamos la API de Pokemon (https://pokeapi.co/). Utilizamos un script para bajar toda la información y almacenarla en las carpetas:

- Compartido/characters para almacenar las estadísticas e información general de los personajes.

Luego de tener almacenada la información, utilizamos otro script para descargar las URL de los sprites ya que en el JSON descargado venia el enlace de las imágenes/sprites.

- Carpeta Cliente/sprites almacenamos los sprites de cada uno de los personajes.

Uso de Procesos, Hilos, Sincronización y Comunicación

Parta la comunicación entre los clientes y el servidor utilizamos sockets.

Cada cliente y el servidor corren en procesos separados e independiente. El servidor para manejar las conexiones con cada uno de los clientes genera hilos para atender las peticiones recibidas.

Dentro del servidor, para manejar los problemas de sincronización entre hilos se utilizan mutex para evitar que los hilos que se comunican con cada cliente no accedan simultáneamente a la región critica. También, utilizamos barreras para estandarizar el tiempo de respuesta para cada uno de los hilos.

Utilizamos sockets para comunicar los procesos e hilos para manejar cada cliente del servidor. En el caso de los sockets implementamos send() y recv() para comunicar al cliente con el servidor, enviar índice del personaje elegido, y para notificarlo de los ataques recibidos, entre otros. Esto además de permitir la comunicación entre cliente y servidor, nos permite sincronizar de alguna manera los turnos durante la batalla.

Limitaciones

Debido al uso de la librería Pygame, estos son los requisitos mínimos:

- Sistema operativo
  - Windows 7 o superior.
  - macOS 10.13 (High Sierra) o superior.
  - Distribuciones de Linux lanzadas después de 2016.
- RAM: 512 MB como mínimo, 1 GB o superior es lo recomendable.
- Procesador de 2 GHz o superior
- Es compatible con Python 3.7 o superior.

Manual de Usuario

Forma de uso:

Para poder jugar se deben seguir los siguientes pasos:

1. Primero, se debe ejecutar el script del servidor en una terminal para que quede a la espera de las solicitudes de los clientes.
2. Luego, se debe ejecutar en otras terminales los scripts de los clientes. Es muy importante ejecutar antes el servidor, ya que de otra manera los clientes no van a poder comunicarse con el servidor y por lo tanto no va a poder realizarse la comunicación entre los clientes
3. Una vez que un cliente selecciono un personaje, este deberá esperar a que el siguiente cliente seleccione el suyo para continuar con la escena de batalla.
4. Se inicia la batalla y cada usuario o cliente puede elegir entre 4 acciones a realizar (atacar, defenderse, curarse o concentrarse), según su turno
5. Cuando se envía una acción de la batalla, el servidor recibe la acción y se la envía al cliente para informar por la interfaz un mensaje de la acción que recibió.

Al abrir el juego se mostrará el siguiente menú de inicio

Imagen 1: Menú de inicio del juego

Al presionar jugar, se mostrará el menú de selección de personajes. Ahí el jugador debe hacer clic o presionar Enter en el personaje que desea utilizar. Se pueden elegir entre una variedad de **200** personajes únicos.

Imagen 2: Menú de selección de personaje

Una vez seleccionado el personaje, el jugador debe esperar en la pantalla de selección de personaje hasta que su contrincante seleccione un personaje y de esa manera pueda comenzar el combate

Cuando ambos jugadores han elegido sus personajes, se mostrará la interfaz de la batalla. El personaje que aparece en la parte inferior izquierda de la pantalla corresponde al personaje aliado, mientras que el personaje en la parte superior derecha de la pantalla corresponde al personaje enemigo. En la parte superior izquierda de la pantalla se muestra la vida de cada uno de los personajes.

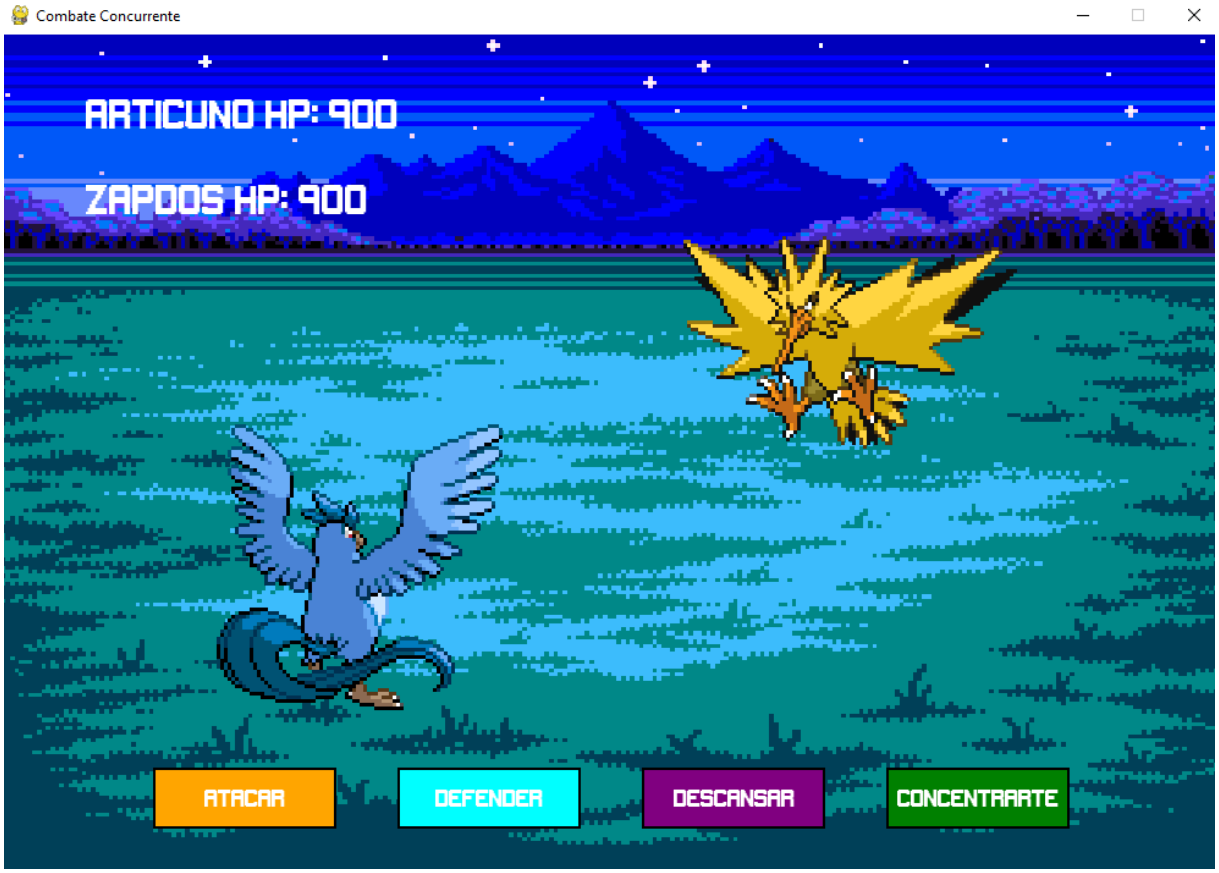
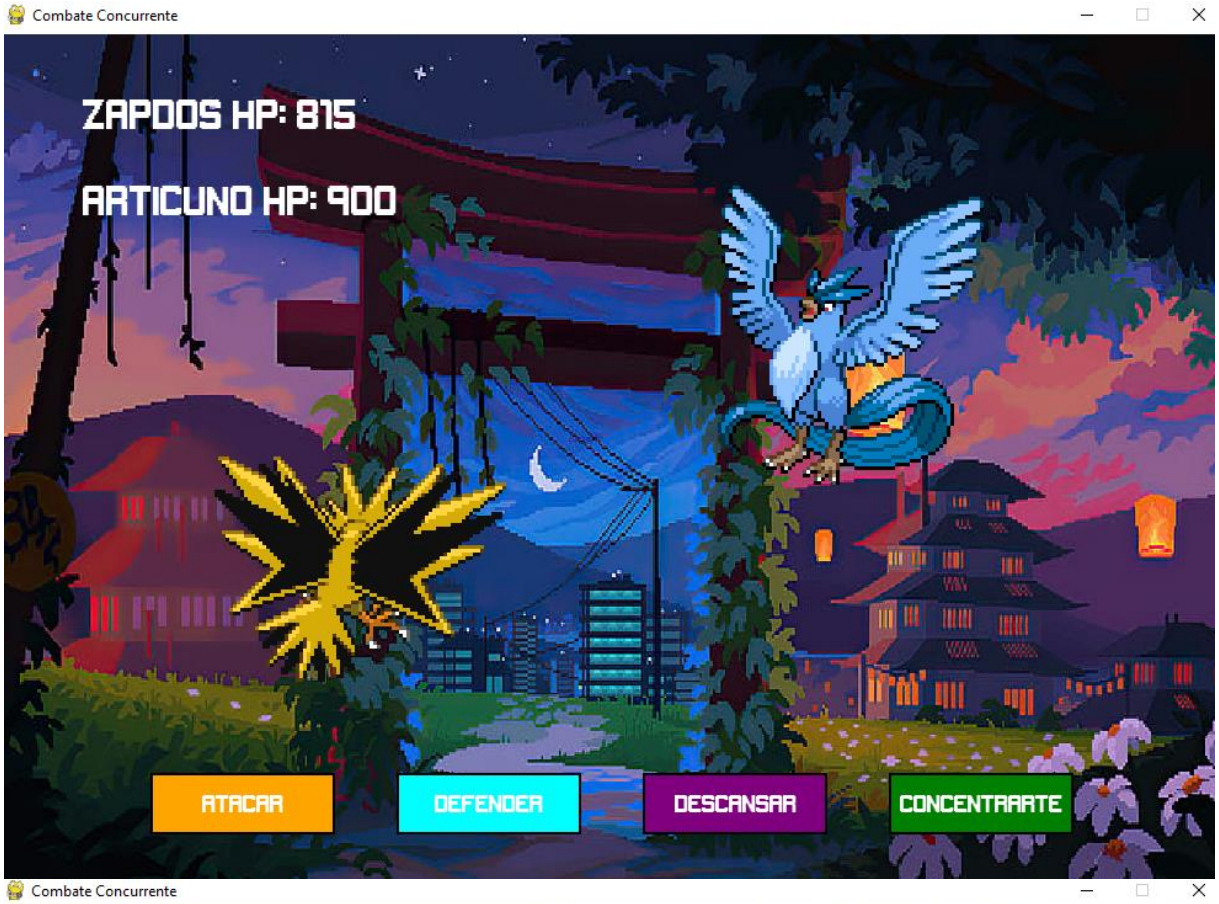


Imagen 4: Interfaz de batalla

En el turno del jugador, se muestran cuatro botones correspondientes a las acciones disponibles que se pueden realizar. Se debe presionar el botón de la acción elegida para terminar su turno.


Combate Concurrente
ZAPDOS HP: 815
ARTICUNO HP: 900
ATACAR
DEFENDER
DESCANSAR
CONCENTRARTE

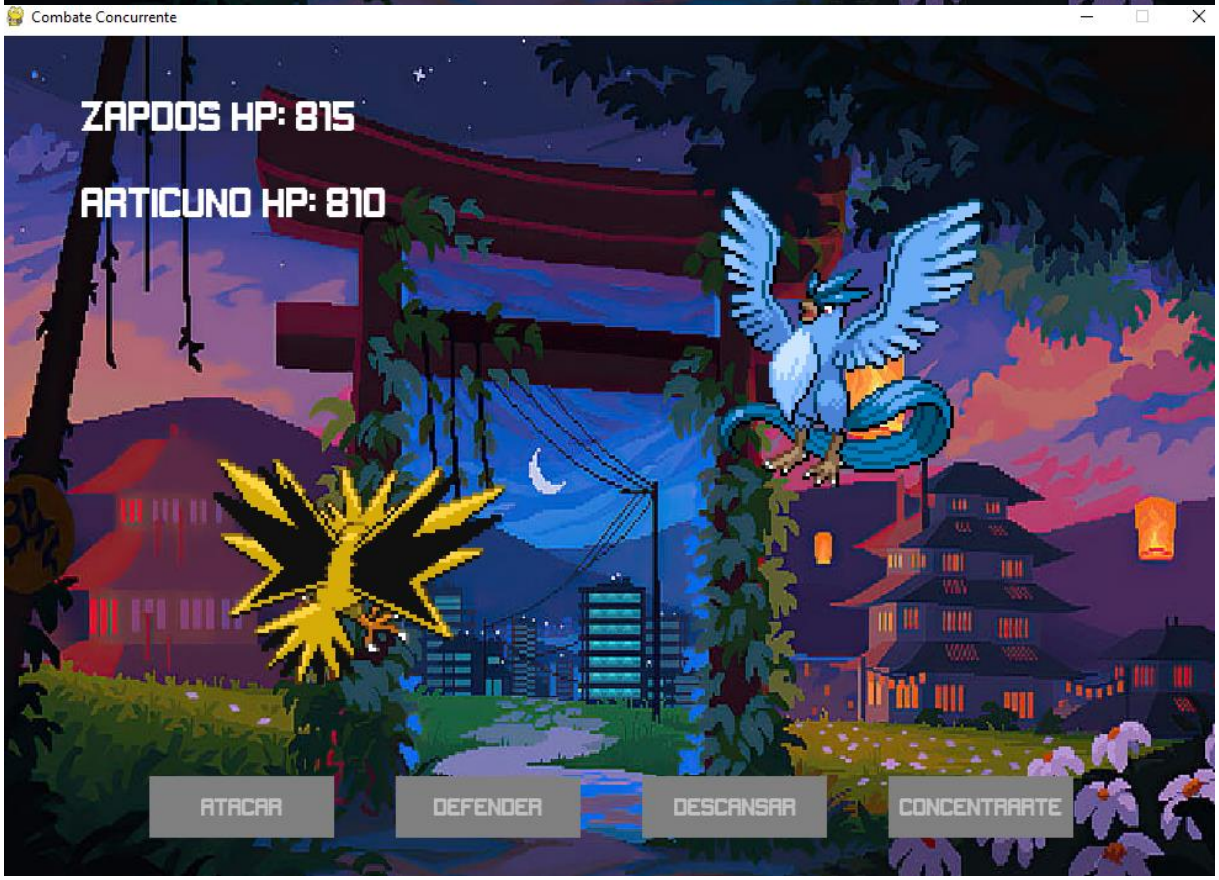

Combate Concurrente
ZAPDOS HP: 815
ARTICUNO HP: 810
ATACAR
DEFENDER
DESCANSAR
CONCENTRARTE

Imagen 5: Botones de Acción

Al elegir una acción, finaliza el turno del jugador actual y pasa el turno al jugador oponente. Se le indicará al contrincante cuál fue la acción elegida mediante un mensaje en pantalla.

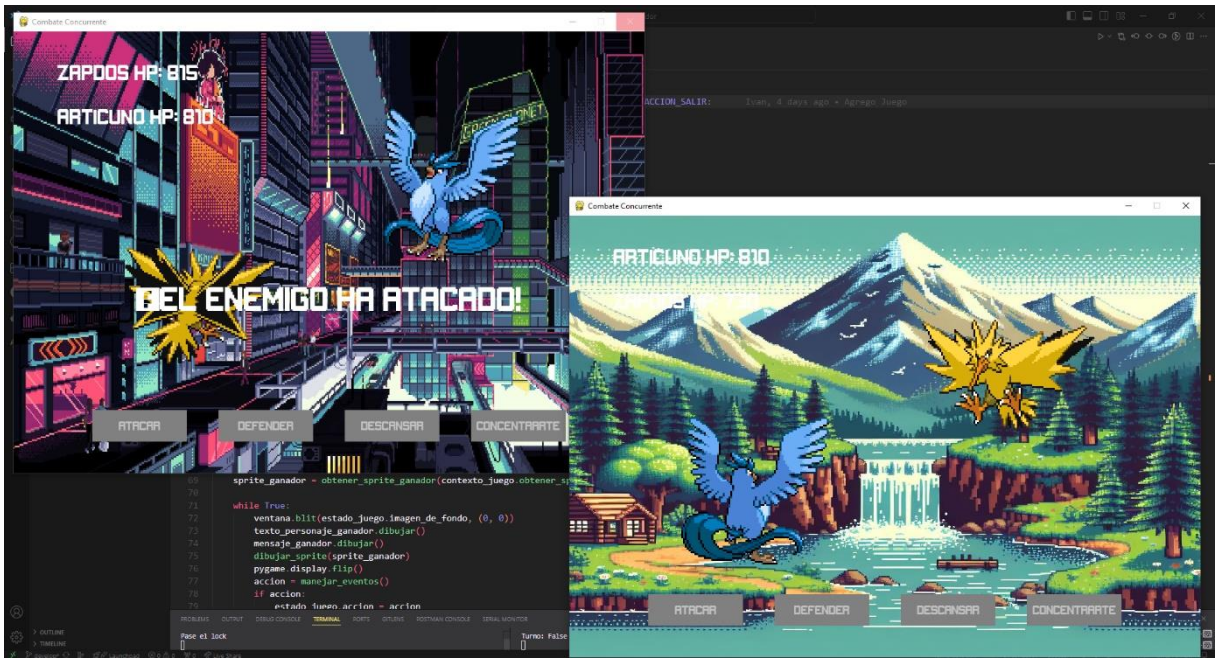


Imagen 6: Mensaje informando acción del contrincante

La batalla finaliza cuando uno de los personajes es derrotado, es decir, su vida llega a cero. Al finalizar la batalla se muestra un mensaje por pantalla indicando el nombre del personaje ganador.

Imagen 7: Mensaje informando resultado de la batalla

Una vez finalizada la batalla, se vuelve a mostrar el menú de selección de personajes en caso de que los jugadores deseen seguir jugando. Para salir del juego, se debe cerrar la ventana. Aparecerá un mensaje preguntando si se desea salir del juego.

Imagen 8: Mensaje confirmación de salida

Conclusiones

En conclusión, este trabajo practico no solo nos ayudó a consolidar los conceptos vistos en la primera parte de la materia, sino que además nos incentivó a investigar y profundizar nuestros conocimientos en estos temas y en nuevas tecnologías como pygame. La libertad de poder elegir el alcance de la aplicación fue un factor fundamental que mantuvo nuestra motivación a lo largo de todo el trabajo practico.