UNIVERSIDAD NACIONAL DE INGENIERÍA
FACULTAD DE CIENCIAS

PRÁCTICA CALIFICADA 2

AUTORES:

Lezama Verastegui, Dagmar Nicole 20204096K

Perez Cueva, Chandler Steven 20200605H

CURSO:

CC4P1 Pogramación Concurrente

LABORATORIO 02

Lima - Perú

Abstract

Este informe presenta el desarrollo de un juego de disparos multijugador inspirado en el icónico videojuego Battle Chopper. Los jugadores controlan naves espaciales con el objetivo de eliminar enemigos emergentes en pantalla y competir para destruir objetivos finales en diversos escenarios. El juego se desarrolla en Java 11 y hace uso de las bibliotecas badlogic.gdx y JavaFX.fxml para el cliente y el servidor, respectivamente. La arquitectura del juego se basa en un modelo cliente-servidor centralizado, con comunicación a través de sockets en Java. La interfaz gráfica se diseñó utilizando la herramienta Tiled para crear mapas y escenarios, brindando una experiencia visual atractiva. Se han implementado estrategias para garantizar la sincronización de datos y optimizar el rendimiento del juego, permitiendo soportar múltiples jugadores en línea. Este proyecto destaca la importancia de conceptos clave de programación multijugador y ofrece una experiencia de juego emocionante.

Keywords: Juego de disparos, Multijugador, Battle Chopper, Java 11, Sockets Java, Tiled, Rendimiento en línea.

Contents

---

1 Introducción

En el presente informe, se expone el desarrollo de un juego de disparos multijugador basado en el emblemático videojuego Battle Chopper, originalmente lanzado en el año 1987. La premisa fundamental reside en el control de una nave espacial, con el propósito de eliminar a los enemigos que emergen en pantalla. En nuestro proyecto, hemos elevado esta experiencia a un nivel superior, permitiendo que múltiples jugadores compitan entre sí con el fin de destruir los objetivos finales presentes en el escenario.

Este juego ha sido desarrollado utilizando Java 11 como base, y se ha hecho uso de las bibliotecas badlogic.gdx para el cliente y javafx.fxml para el servidor. La arquitectura del juego se divide en dos componentes esenciales: el servidor, encargado de gestionar la conexión entre los jugadores y actualizar el estado del juego, y el cliente, responsable de controlar la nave del jugador y presentar la interfaz gráfica del juego.

Con el objetivo de garantizar una comunicación eficaz y fluida entre los jugadores, hemos implementado una estructura que incluye un campo de puerto para el servidor y dos campos destinados a la dirección IP y el puerto del cliente. Esta elección de diseño nos ha permitido utilizar exclusivamente sockets en Java, asegurando una experiencia de juego robusta y sin interrupciones. Además, en conformidad con las directrices establecidas, hemos optado la concepción de una interfaz gráfica que garantice una experiencia de juego cuasi completa y gratificante.

A lo largo del informe, se proporcionará un análisis del juego, incluyendo una evaluación detallada de su funcionamiento, una demostración de su capacidad de juego en red y una optimización del rendimiento para soportar el mayor número de jugadores en línea posible.

2 Marco Teórico

2.1 Género de Videojuegos de Disparos

Battle Chopper pertenece al género de videojuegos de disparos, un género en el que el jugador asume el control de un personaje solitario, a menudo una nave espacial, helicóptero o avión, que dispara contra hordas de enemigos que aparecen en pantalla. Estos juegos forman un subgénero de los videojuegos de disparos y se caracterizan por la necesidad de que el jugador reaccione rápidamente para tener éxito, esquivando el fuego enemigo mientras combate contra un gran número de adversarios.

2.2 Desarrollo de un Modo Multijugador

Una de las metas principales del proyecto es llevar la experiencia de Battle Chopper al siguiente nivel mediante la implementación de un modo multijugador. En este modo, varios jugadores pueden competir entre sí para destruir objetivos finales que pueden ubicarse en diferentes locaciones del escenario. Para lograrlo, se utiliza Java 11 como mínimo y se definen campos para el puerto en el servidor, así como dos campos para la dirección IP y el puerto en el cliente. La comunicación entre los jugadores se establece exclusivamente a través de sockets en Java, asegurando una experiencia de juego fluida y eficiente.

2.3 Conceptos de Programación Multijugador

El desarrollo de un modo multijugador en el proyecto implica la comprensión y aplicación de varios conceptos clave en programación multijugador. Estos conceptos son esenciales para permitir que múltiples jugadores interactúen en tiempo real en un entorno de juego compartido. Algunos de estos conceptos incluyen:

2.3.1 Comunicación Cliente-Servidor

En un entorno multijugador, se utiliza una arquitectura cliente-servidor. El servidor actúa como una entidad central que coordina la comunicación y el estado del juego entre todos los clientes. Los clientes se conectan al servidor mediante sockets en Java, que proporcionan una forma estándar y versátil de enviar y recibir datos a través de la red.

2.3.2 Sincronización de Datos

La sincronización de datos es fundamental para garantizar que todos los jugadores vean una representación coherente del juego. Esto implica la transmisión de datos actualizados sobre la posición de los jugadores, objetos del juego y otros elementos relevantes a través de los sockets.

2.3.3 Latencia y Retardo de Red

La latencia y el retardo de red son desafíos comunes en los juegos multijugador. La latencia se refiere al tiempo que tarda un paquete de datos en viajar desde un cliente al servidor y viceversa. El retardo de red puede causar problemas de sincronización y percepción del tiempo en el juego, lo que requiere estrategias de mitigación.

2.3.4 Modelo de Autoridad

En un juego multijugador, es necesario determinar cuál de las dos entidades (cliente o servidor) tiene la autoridad sobre ciertos aspectos del juego, como la validación de disparos o la detección de colisiones. Esto evita trampas y asegura la integridad del juego.

2.3.5 Escalabilidad

La capacidad de admitir un número variable de jugadores es un aspecto importante en la programación multijugador. Es necesario diseñar la arquitectura del servidor y la gestión de recursos de manera que el juego pueda funcionar sin problemas con diferentes cantidades de jugadores.

2.4 Sockets en Java

En el desarrollo de este proyecto, se hace uso de sockets en Java para la comunicación entre el servidor y los clientes. Un socket es una abstracción de comunicación que permite que los programas se comuniquen a través de una red, como Internet. Los sockets proporcionan una forma estándar y versátil de enviar y recibir datos entre aplicaciones en diferentes dispositivos. En este contexto, los sockets en Java permiten que el servidor y los clientes se conecten, envíen y reciban datos, facilitando la interacción entre los jugadores en un entorno multijugador.

2.5 Interfaz Gráfica: Uso de Tiled

Para el diseño y desarrollo de la interfaz gráfica del juego, se hace uso de la herramienta Tiled. Tiled es una aplicación que permite la creación de mapas y escenarios de manera eficiente y versátil. Permite a los desarrolladores diseñar niveles, ubicar objetos, definir colisiones y generar mapas que luego se integran fácilmente en el juego. Esta herramienta resulta esencial para la representación visual de los escenarios y la disposición de elementos que enriquecen la experiencia de juego.

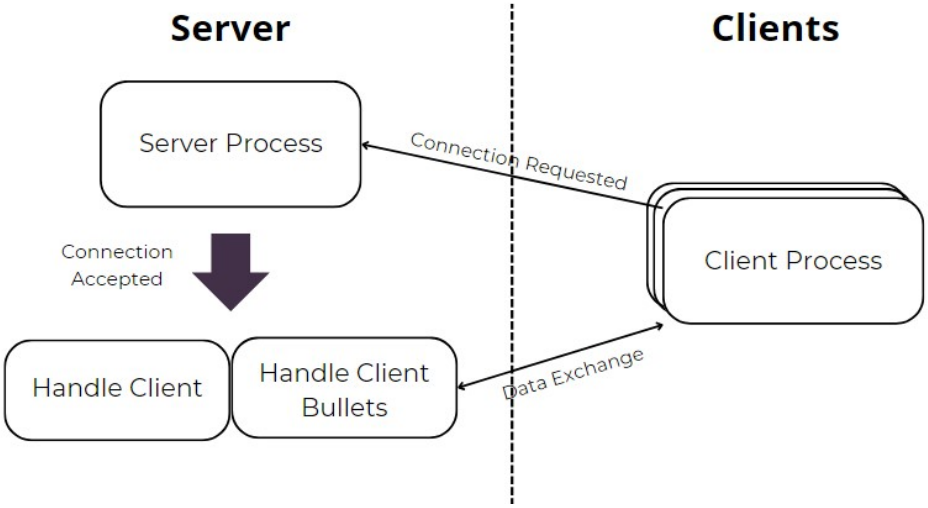
2.6 Bibliotecas Utilizadas

Para el desarrollo del juego, se emplean dos bibliotecas específicas. Badlogic.gdx se utiliza para el cliente y JavaFX.fxml para el servidor. Badlogic.gdx es una biblioteca de código abierto que proporciona herramientas y recursos para la creación de juegos en 2D y 3D, lo que la convierte en una elección sólida para desarrollar la parte del cliente del juego. JavaFX.fxml, por otro lado, es una tecnología que facilita la creación de interfaces gráficas de usuario en aplicaciones Java. Esto permite diseñar una interfaz gráfica que brinde una experiencia visual atractiva y funcional a los jugadores.

3 Metodología

3.1 Diseño de la Arquitectura

Antes de comenzar con la implementación, diseñamos la arquitectura general del juego que engloba la definición de nuestra estructura cliente-servidor.
Decidimos utilizar una arquitectura de servidor centralizado para garantizar la sincronización entre los jugadores.
A continuación, se muestra un gráfico que representa nuestra estructura:



La lógica del juego se desarrolló siguiendo una arquitectura cliente-servidor centralizada. En el lado del servidor, se implementó un bucle de juego que gestionaba la física del mundo, las interacciones entre los jugadores y los enemigos, y la actualización constante del estado del juego. Cada cliente, por su parte, tenía su propia instancia del juego que representaba su estado local. La comunicación entre el servidor y los clientes se realizó mediante sockets en Java, lo que permitió la transmisión eficiente de datos de estado y eventos del juego. La sincronización de datos se logró enviando paquetes de actualización periódicos desde el servidor a los clientes y procesando las acciones de los jugadores en el servidor. Además, se implementó un sistema de autoridad para garantizar la coherencia del juego, y se optimizó el rendimiento para admitir el mayor número posible de jugadores en línea simultáneamente. Este enfoque permitió crear un juego multijugador dinámico y envolvente basado en el clásico Battle Chopper como indicaban las directrices del proyecto en un inicio.

4 Resultados y Discusiones

Los detalles de las computadores que se utilizaron para poner a prueba el Juego respectivo se encuentran a continuación: Computadora 1

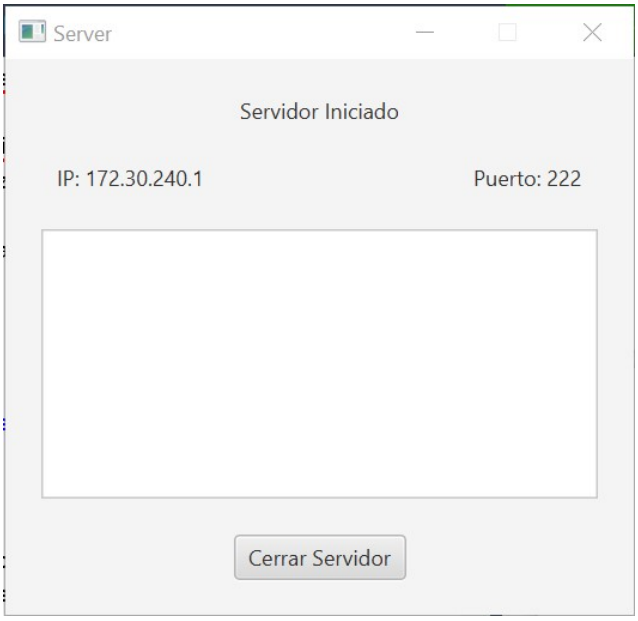
- Procesador: AMD Ryzen 9 4900HS with Radeon Graphics, 3000 MHz, 8 procesadores principales, 16 procesadores lógicos.
- Unidad de Estado Sólido: 1024 GB
- RAM: 16 GB a 3200 MHz.

Computadora 2

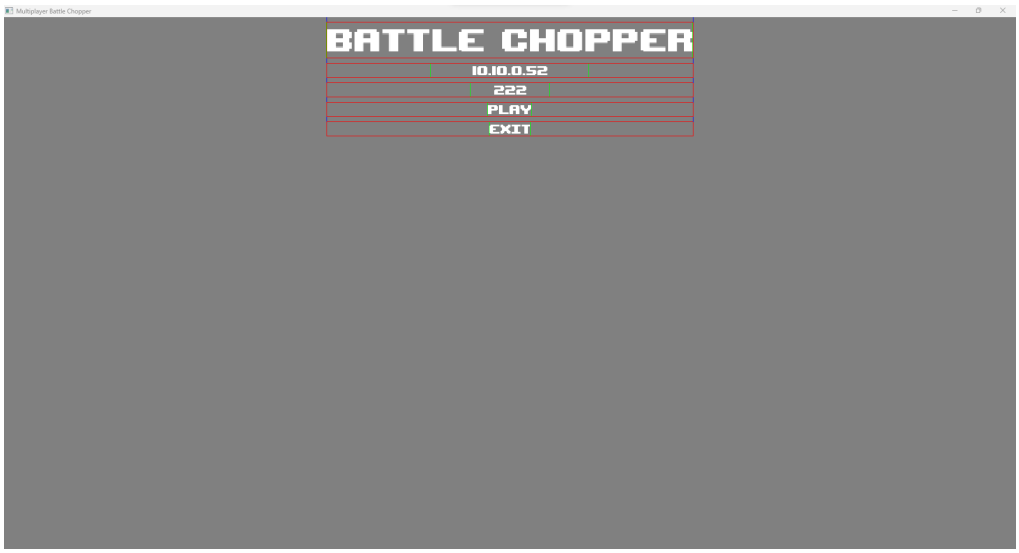
- Procesador: Procesador AMD Ryzen 7 7735HS with Radeon Graphics, 3201 Mhz, 8 procesadores principales, 16 procesadores lógicos
- Unidad de Estado Sólido: 512 GB
- RAM: 16 GB a 4800 MHz.

Al inicializar la aplicación para el servidor, tenemos la pantalla siguiente:

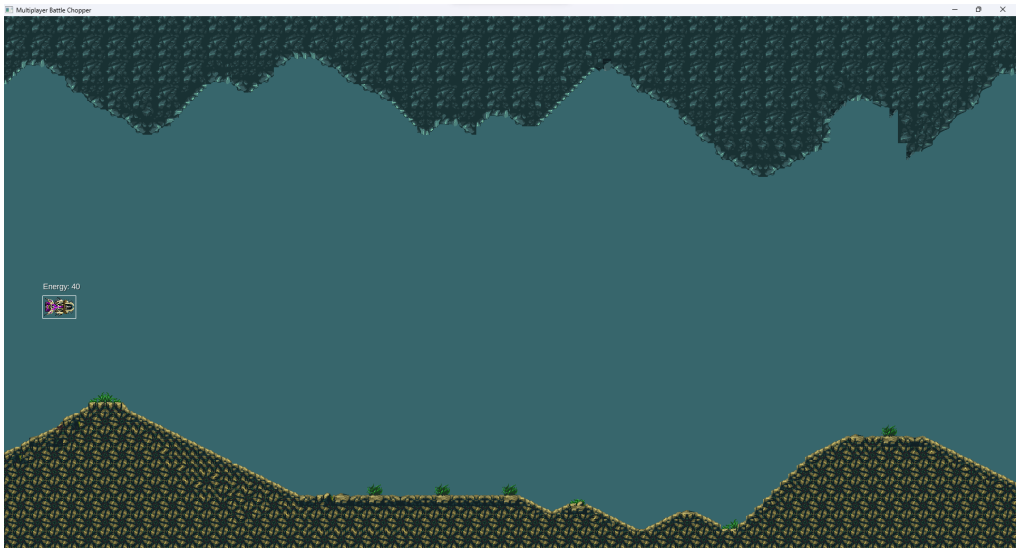
El apartado para el puerto solo acepta números, a continuación se muestra la ventana siguiente al hacer click en el botón Iniciar Servidor:



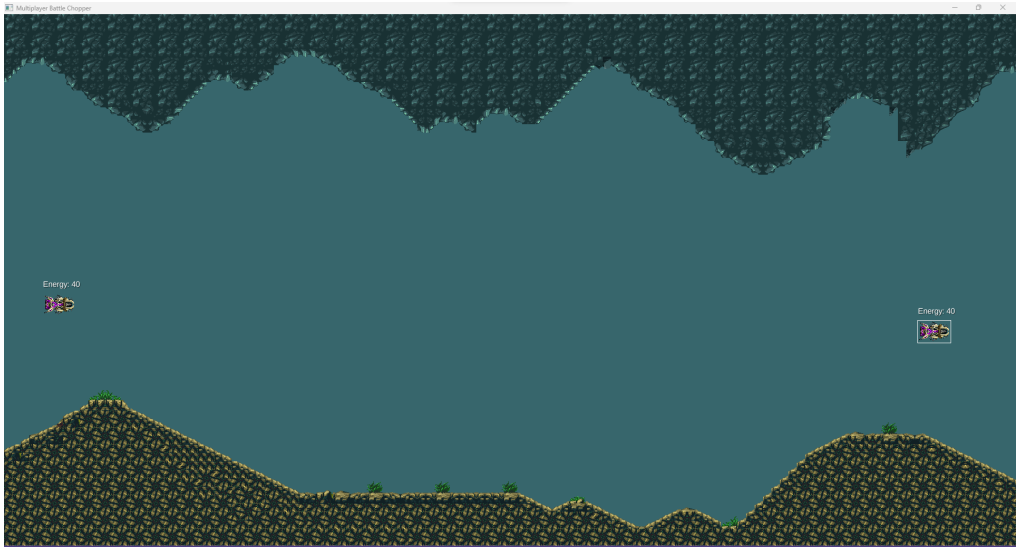
El servidor se encuentra esperando una conexión, así que se muestra la ejecución inicial de un cliente:



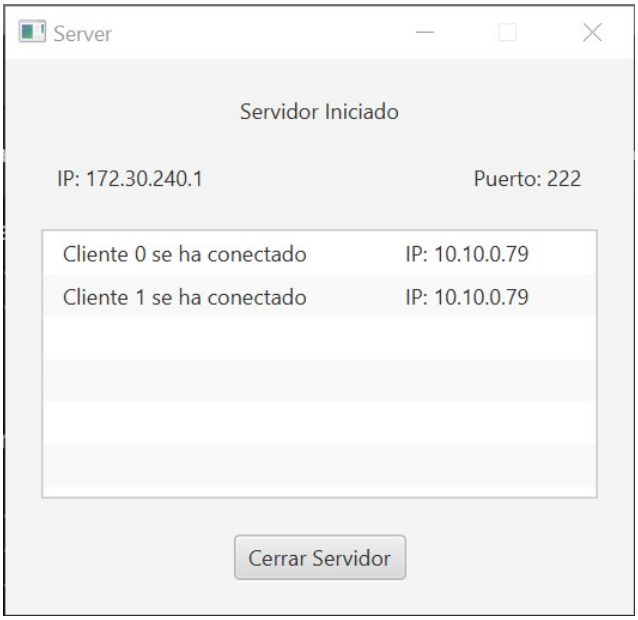
Al hacer click en play, se obtiene lo siguiente:



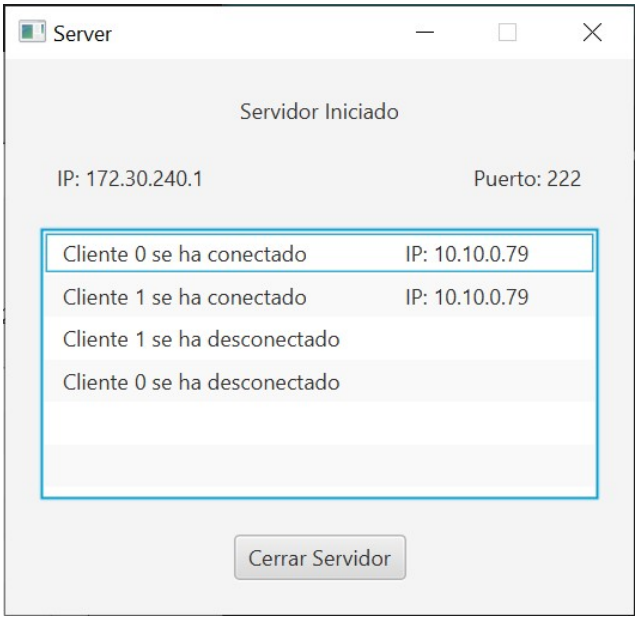
Para probar la conexión multijugador agregamos un cliente más y obtenemos:



Así el servidor se actualiza de la siguiente manera:



Y cuando ambos se desconectan, también se muestran los eventos en la ventana del servidor:



5 Conclusiones

El juego logró capturar la esencia y la diversión del juego original Battle Chopper. Los jugadores pudieron disfrutar de emocionantes batallas espaciales y competir por la destrucción de objetivos finales en variados escenarios.

La implementación técnica del juego resultó ser sólida. Se diseñó una arquitectura cliente-servidor centralizada que permitió la comunicación fluida entre los jugadores a través de sockets en Java. Además, la utilización de Tiled y JavaFX.fxml para la creación de la interfaz gráfica proporcionó una experiencia visual atractiva.

Durante las pruebas, se identificaron problemas de escalabilidad cuando había 4 o 3 jugadores en el juego. Estos problemas se tradujeron en errores notables y retrasos en la experiencia del usuario. Este hallazgo resalta la importancia de abordar la escalabilidad desde el inicio de proyectos similares.

El proyecto proporcionó valiosas lecciones sobre la programación multijugador y el rendimiento en juegos en

línea. Se reconoció la necesidad de implementar estrategias más avanzadas para gestionar múltiples jugadores y mitigar problemas de latencia y retardo de red.
Para futuros proyectos, se considera esencial abordar y resolver los problemas de escalabilidad de manera más efectiva. Además, se explorarán tecnologías alternativas y estrategias de autoridad más avanzadas para mejorar el rendimiento en juegos multijugador.

6 Anexo Código

El código realizado se encuentra adjuntado en el archivo .zip.

---

7 Referencias

Baydal Cardona, M. E. (2017). Programación de clientes TCP en java.

Villalobos, J., Hernandez, U., del silicon power Armor, R., Reseña, U. S. B., & Marvel, S. P. (2011). Sockets en Java (cliente y servidor). Codigoprogramación. com, 7.

Brennan, Ciarán (10 March 1988). "Slots of Fun". Your Sinclair. No. 28 (April 1988). pp. 78–9.

https://www.genbeta.com/desarrollo/tiled-map-editor-el-editor-de-mapas-libre

https://libgdx.com/dev/

https://docs.oracle.com/javase/8/javafx/api/javafx/fxml/doc-files/introduction$_t o_f xml.html$