Instituto Tecnológico de Costa Rica

Área de Ingeniería en Computadores

Lenguajes, Compiladores e Intérpretes (CE 3104)

Tarea III– SpaceInvaders

Integrantes:

Meibel Ceciliano Picado, carné 2020023333

Kevin Lobo Juárez, carné 2020087823

Nicol Otárola Porras, carné 2021117282

Profesor:

Marco Rivera Meneses

Fecha de entrega:

21 de abril

I Semestre, 2023

Tabla de contenidos

Descripción de la utilización de las estructuras de datos desarrolladas

Listas enlazadas simples

Para el proyecto se usaron listas enlazadas simples como estructuras de datos en las que cada elemento está conectado a través de un puntero al siguiente elemento. Fueron utilizadas principalmente para la manipulación de los enemigos y los proyectiles, tanto de los meteoritos

(balas de los invasores) así como las balas lanzadas por la nave. De esta manera fue posible mostrar los sprites en pantalla y manejar sus movimientos.

Cada elemento invasor como cangrejo, calamar y pulpo es un nodo de la lista enlazada, cada uno con su propia estructura “struct” y se relacionan directamente con punteros que manejan las direcciones en memoria de los siguientes nodos en la lista.

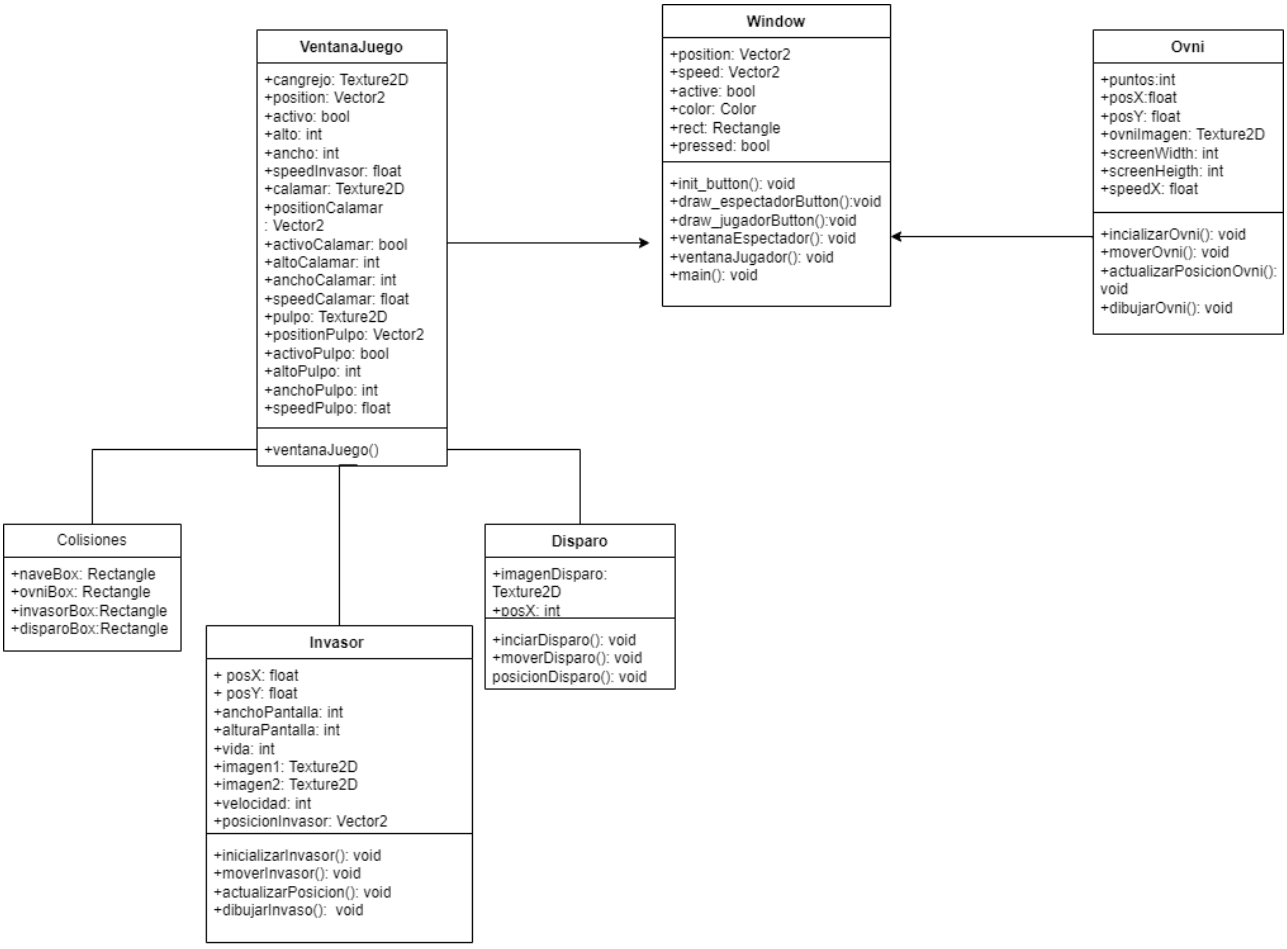
Las listas enlazadas fueron convenientes para la ordenación de los datos, además de que cuenta con la ventaja que permiten agregar y eliminar elementos de la lista con mayor eficiencia que las estructuras de datos estáticas como los arrays.

Matriz

Por otro lado, se utilizaron matrices como estructura de datos para almacenar elementos en una disposición de filas y columnas, con esto se pudieron colocar en posiciones mas acertadas los sprites en la pantalla, como el caso de los bunkers, el movimiento del ovni que contiene posiciones aleatorios en filas, la nave y los invasores. Algo positivo de esto es que se pueden contener datos de diferentes tipos ya sea enteros o flotantes, en Raylib muchas funciones están diseñadas para trabajar con matrices principalmente para representar las coordenadas, la rotación y la escala de los objetos en pantalla.

Descripción detallada de los algoritmos desarrollados

Lógica del juego en C



La clase "window" genera una ventana con dos botones: uno para jugar y otro para observar. Si se presiona el botón "jugar", se abre una ventana donde el jugador puede controlar una nave y disparar a los invasores que aparecen en la pantalla. Si se presiona el botón "observar", se abre una ventana donde se puede ver la partida, pero sin la posibilidad de interactuar con ella.

Tiene métodos para crear la interfaz gráfica y manejar los eventos del teclado y del mouse. Además, para dibujar los botones, mover la nave y detectar si se presiona la tecla de espacio para disparar. También se utilizan estructuras para definir la posición, velocidad y estado de los proyectiles, así como para almacenar información sobre los botones (su posición, color y si están presionados o no).

La clase "Ovni" tiene atributos como la posición (posX y posY), una imagen del ovni y una velocidad de movimiento (speedX). También tiene funciones como "inicializarOvni" para inicializar el ovni en una posición y puntos específicos, "moverOvni" para mover el ovni en la pantalla y "actualizarPosicionOvni" para verificar si el ovni ha llegado al borde derecho de la pantalla y reiniciarlo en una posición aleatoria en la pantalla.

De la clase "ventanaJuego", en el programa se definen tres tipos de invasores: Cangrejo, calamar y pulpo, cada uno con sus propios atributos como textura, posición y velocidad.
El programa inicializa la ventana y carga las texturas necesarias para el juego, incluidas las imágenes de la nave espacial y las balas. Luego inicializa las estructuras de invasor para cada tipo de invasor, configurando sus atributos como posición y velocidad. El programa también define la posición inicial y la velocidad de la nave espacial y la bala, y establece el estado de la bala en inactivo. También establece una variable booleana para realizar un seguimiento de la dirección en la que se mueven los invasores e inicializa las variables para realizar un seguimiento de la cantidad de puntos que ha ganado el jugador.

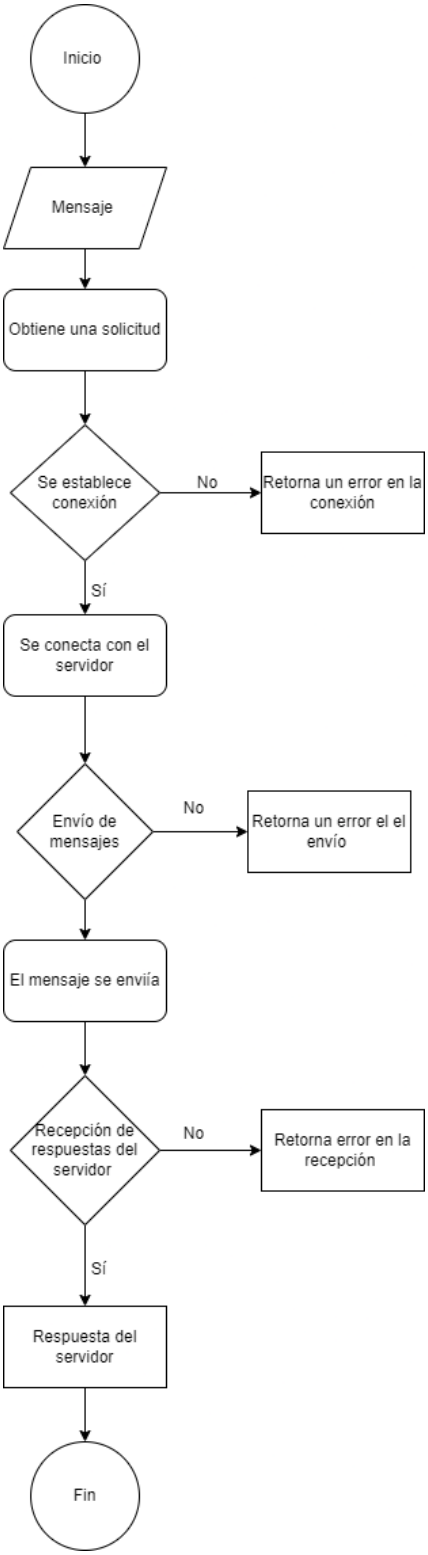
La clase Colisiones tiene una serie de atributos que representan las áreas de colisión de los elementos del juego. Estas áreas son objetos Rectangle que se utilizan para detectar colisiones entre los diferentes elementos. No tiene métodos propios, ya que se utiliza simplemente como un conjunto de atributos compartidos entre diferentes objetos que necesitan detectar colisiones.

La clase Disparo representa un disparo en el juego. Tiene atributos que representan la posición y la imagen del disparo. Tiene tres métodos, uno para inicializar el objeto, otro para mover el disparo en el juego y otro para posicionar el disparo en función de la posición actual de la nave. El método "posicionaDisparo" utiliza un objeto de la clase Nave para determinar la posición inicial del disparo en el juego.

La clase Invasor tiene una serie de atributos que representan las propiedades de un invasor en el juego. Los métodos proporcionan la funcionalidad para inicializar, mover, actualizar la posición y dibujar al invasor en la pantalla.

Servidores

Para el cliente:

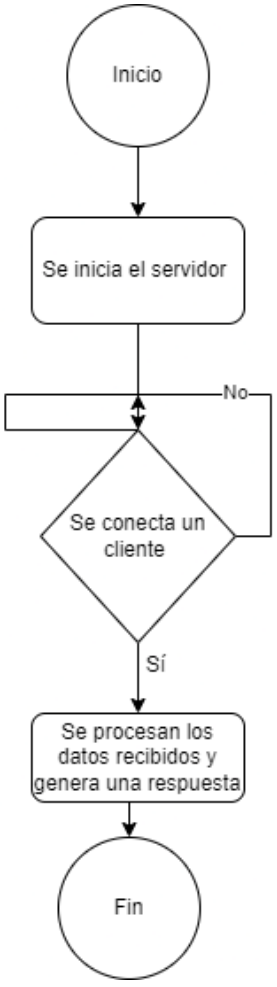


El cliente recibe una entrada, en este caso un mensaje conformado por una cadena de caracteres que el usuario ingresa por medio de la entrada estándar (stdin). Este mensaje se envía al servidor a través del socket.

Posteriormente, se da el proceso de validar condiciones, la primera es verificar si se establece conexión, si esta condición se cumple si la conexión con el servidor se ha establecido correctamente. Si la conexión falla, el programa muestra un mensaje de error y termina.

Seguidamente está el envío de mensaje, esta condición se cumple si el mensaje se ha enviado correctamente al servidor a través del socket. Si el envío falla, el programa muestra un mensaje de error y termina. Por último, recepción de respuesta del servidor se cumple si se ha recibido una respuesta del servidor después de enviar el mensaje. Si la recepción falla, el programa muestra un mensaje de error y termina.

Para el servidor:



El servidor se inicializa en el puerto 8080 y queda atento a los clientes que se vayan a conectar, es por lo que, tiene un bucle infinito que espera la conexión de clientes, si se establece la conexión con un cliente, se procesan los datos recibidos, se genera la respuesta y se envía, luego cierra la conexión y repite el proceso.

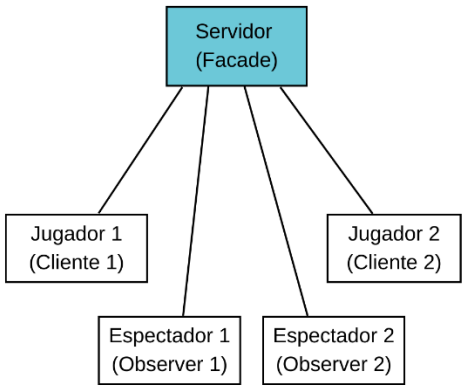
La única manera que el servidor se cierre es que ocurra un error o que se de interrupción de manera manual.

Problemas sin solución:

Implementación de sockets, funcionan correctamente, pero no se logra agregar completamente al juego, es decir, los sockets se conectan entre sí, pero la lógica desarrollada en C no esta enlazada con el cliente, por lo que el servidor no tiene conocimiento de lo que está ocurriendo, solo sabe que hay una conexión activa.

Patrones observer, no se logró implementar en los sockets, ya que no se pudo comunicar los cambios hechos en la ventanaJuego.c, por lo que el servidor no puede pasar los datos a la ventada espectador, asimismo, la ventana de espectador no pudo ser implementada y era parte del observer.

Por la forma en que se sugiere el juego, lo ideal es utilizar el patron facade ya que se utiliza para proporcionar una interfaz simplificada para un sistema más complejo. El patrón Facade se utiliza para ocultar la complejidad de un sistema detrás de una interfaz simple y unificada, lo que permite a los usuarios interactuar con el sistema sin tener que preocuparse por los detalles internos, sin embargo, no se pudo implementar.



Problemas encontrados con solución:

Antes de iniciar con el desarrollo del juego se investigaron librerías graficas en C que fueran convenientes para lo que se buscaba, se intentó con la librería Allegro, sin embargo la importación de esta fue un desastre, la documentación oficial es limitada y puede resultar difícil de seguir, por lo que si ocurría un problema era complicado encontrar una solución en corto tiempo, por otro lado se tuvieron problemas de rendimiento, pues habían lags en las pantallas, o los sprites agregados al moverse se distorsionaban, por ende se optó por utilizar Raylib y fue mucho más rápido avanzar con ella, pues al ser una biblioteca más actual cuenta con herramientas y características más avanzadas.

Hubo una transposición de sprites de enemigos, el movimiento de los cangrejos se colocaba encima de los del calamar, y estos con los del pulpo. Para solucionarlo se generó una variable booleana que es independiente para cada invasor, esta variable le indicará a cada figura la dirección hacia la derecha e izquierda que debe mantener durante todo el juego.

Para la ventana principal del servidor en Java, perteneciente al menú principal donde se insertan ovnis e invasores se inicio utilizando un form para realizar la interfaz, sin embargo, al correr el código colocaba los elementos de manera desordenada y distinto a como se esperaba, por ende se decide realizar de manera manual el código referente a esta ventana.

Conclusiones

Se comprueba la importancia de usar punteros en lenguajes de programación, a través de estos fue posible tener un acceso directo a la memoria y un mejor control sobre ella, reduciendo bastante tiempo al manipular los diferentes objetos. Hizo más eficiente el paso de estructuras de datos complejas de una función a otra, ayudando a no tener un overhead (exceso de código).

Fue posible aprender sobre los structs en C, una ventaja de las estructuras es que permiten definir variables de diferentes tipos de datos, contrario a los arreglos que trabajan únicamente elementos de un mismo tipo de dato. En el código de este videojuego todos los elementos como invasor, ovni, nave, etc fueron definidos bajo esta herramienta, esto ayudo mucho a la organización del código, pues lo hizo más legible y fácil de entender.

Para este proyecto fue sumamente eficiente trabajar el programa de manera modular, al escribir el código en módulos separados y autocontenidos que realizan tareas en específico, cada módulo podía ser probado de manera independiente y sin arriesgar a los demás, además se podía modificar facilitando la incorporación de nuevas funcionalidades.

Elegir una librería grafica correcta es un punto relevante, para esta tarea emplear Raylib fue una decisión muy acertada, al estar enfocada al desarrollo de videojuegos tiene una API sencilla y fácil de aprender, a través en su página oficial se encontraron muchos ejemplos similares y variedad de herramientas que fueron una guía para comenzar a trabajar. Se ahorraron "lags" y problemas de bajo rendimiento pues es una librería bastante ligera y rápida con tiempos mínimos de respuesta a una solicitud.

En definitiva, Java es un lenguaje muy amigable, pues su paradigma POO modela el mundo real en términos de objetos, esta forma de pensar y ver el mundo se asemeja a la manera en que los humanos entendemos el mundo que nos rodea. Por ende, como programadores el trabajar con atributos y comportamientos de objetos volvía más comprensible el manejo del código, por lo tanto, lo desarrollado en este lenguaje consumió menos tiempo y generó menos errores que en C.

Recomendaciones

Los punteros son una herramienta poderosa pero al trabajar con ellos se debe ser un poco precavido, pues al ser una variable que almacena la dirección de memoria de un objeto, un mal manejo puede provocar errores o fallos en el programa. Al inicializar un puntero se debe asegurar de hacerlo con valor valido, pues en caso contrario, sino se inicializa el puntero puede apuntar a una dirección de memoria no valida, además es importante liberar la memoria que fue asignada dinámicamente utilizando funciones como malloc.

Al programar en Java es necesario un repaso general de los conceptos básicos de POO y la comprensión de estos como la herencia de clases, polimorfismo, abstracción y encapsulamiento, los anteriores van a permitir al desarrollador crear código que sea modular, fácil de mantener, reutilizable y escalable.

C es un lenguaje de programación de medio nivel, un poco complicado de sintaxis por lo que no es un lenguaje sencillo de aprender, es un lenguaje estructurado, siendo esto una de las características más poderosas al tener capacidad de definir y utilizar estructuras, estos structs son muy similares a clases en Java, por lo tanto, si se te hace sencillo la definición de atributos y métodos será menos complejo entender. Se recomienda dividir en bloques de código mas pequeños y de manera incremental ir combinando para crear programas más complejos.

Si desea elaborar un videojuego multijugador es necesario el uso de sockets, se debe crear un servidor que sea capaz de escuchar las solicitudes de conexión que ingresan de los clientes, mediante el socket se crea la conexión, pudiendo a través de esto intercambiar datos bidireccionalmente. Se sugiere utilizar el protocolo TCP/IP pues es mas confiable y permite comprobar errores entre procesos, contrario de sockets datagramas. En el caso de C al utilizarse en Windows es mejor usar winsock2.

Los patrones de diseño son soluciones comunes y probadas para problemas de diseño de software recurrentes, para este juego en particular se solicita un cliente espectador el cual se puede unir a la partida pero solo observar el juego actual, para esto se aconseja utilizar el patrón observer , el funcionamiento es básicamente notificar y actualizar a un objeto dependiente de otro que este ultimo ha cambiado su estado. De esta manera podrá observar todos los cambios que han ocurrido pero no ser capaz de modificar ni participar en el juego.

Bibliografía

Buchanan, W. (1998). Java Socket Programming. En *Mastering Java* (pp. 188–198). Macmillan Education UK.

Descargar Winsock Interface Library C/C 3.1.1 Gratis para Windows. (s/f). Programas-gratis.net. Recuperado el 19 de abril de 2023, de https://winsock-interface-library-for-c-c-plus-plus.programas-gratis.net/

Faile, R. [@rudyfaile]. (2021, enero 26). *How to compile Raylib from scratch and set up your first projects (Windows / C) in 15 minutes*. Youtube. https://www.youtube.com/watch?v=HPDLTQ4J_zQ

Free Icons and Stickers - Millions of images to download. (s/f). Flaticon. Recuperado el 19 de abril de 2023, de https://www.flaticon.com/

Introducción a Winsock. (s/f). Microsoft.com. Recuperado el 19 de abril de 2023, de https://learn.microsoft.com/es-es/windows/win32/winsock/getting-started-with-winsock

Raylib – cheatsheet (s.f) Raylib. Recuperado de https://www.raylib.com/cheatsheet/cheatsheet.html

Plan de Actividades realizadas por estudiante:

Se realiza antes de comenzar con el proyecto, busca realizar un plan de acción, donde se describen las actividades, persona responsable de llevarlas a cabo y fecha en la cual debe ser entregada dicha actividad.

CRONOGRAMA			
ACTIVIDAD	**TIEMPO ESTIMADO DE COMPLETITUD**	**RESPONSABLE**	**FECHA DE ENTREGA**
Reunión inicial para definir roles y tareas. Se llega al acuerdo que se va a trabajar todo el código en Windows. Además, se	1 hora	Todos	3/4/2023

distribuyen las tareas iniciales. Nicol tendrá que elaborar el Servidor en Java, Meibel el Cliente en C, y Kevin irá avanzando la lógica del juego.			
Creación de repositorio	10 minutos	Meibel	4/4/2023
Se crean los documentos donde se realizará el manual de usuario y documentación técnica, ambos con su debido formato.	1 hora	Meibel	4/4/2023
Cada estudiante empieza a trabajar en la primera tarea que se le asignó	3 horas	Todos	6/4/2023
Reunión #2 para conectar el cliente y servidor, revisar una parte de la lógica elaborada y distribuir nuevas tareas.	1 hora	Todos	11/4/2023
Se acuerda que Kevin y Meibel tendrán			12/4/2023

como tarea realizar la interfaz gráfica en C y continuar con la lógica del juego. Nicol desarrollará la ventana de inicio del servidor con su debida interfaz gráfica.			
Reunión #3 para mostrar los avances, la ventana del servidor esta lista, y se ha finalizado la venta principal del cliente, además se tiene el movimiento base de los enemigos, el movimiento de la nave y sus disparos con las teclas. Se cuenta también con los bunkers.			16/4/2023
Se decide que los 3 estudiantes trabajaran simultáneamente con el código encargado de las colisiones, ovni, puntuación y vidas.			16/4/2023

Una vez se tenga terminado el juego, se busca enlazar ambos archivos .java y .c mediante los sockets.			20/4/2023
Reunión #4 donde se verifica que todos los archivos funcionen correctamente.	1 hora	Todos	20/3/2023
Detalles finales de SpaceInvaders	1 hora	Todos	21/4/2023
Creación de manual de usuario	3 horas	Meibel	20/4/2023
Reunión #5 para distribuir las secciones de la documentación	1 hora	Todos	18/4/2023
Comenzar con la redacción de la documentación técnica	2 horas	Todos	19/4/2023
Finalizar las secciones del trabajo escrito	3 horas	Todos	20/4/2023
Subir la carpeta comprimida que contiene todos los	10 minutos	Todos	20/4/2023

entregables de la tarea 3			
Reunión final donde se discute y se prepara la presentación para la defensa.	1 hora	Todos	21/4/2023

Bitácoras

Se describen las actividades realizadas, reuniones entre compañeros de trabajo, investigaciones, consultas al profesor, entre otras. Se debe describir cada detalle, es un diario de trabajo, donde se llevará un seguimiento del tiempo empleado en el proyecto. En caso de que se requiera un relevo u ocurra algún problema, algún miembro del equipo debe ser capaz de continuar su trabajo con solo leer sus bitácoras.

Bitácora Meibel Ceciliano

Diario de Trabajo			
Fecha	**Actividades realizadas**	**Duración**	**Estado**
4/4/2023	Se diseña el formato que se va a usar para la documentación técnica, y el manual de usuario	1 hora	Completado
4/4/2023	Creación de repositorio	10 minutos	Completado
6/4/2023	Comienzo de investigación sobre el funcionamiento de sockets en lenguaje C y utilizando sistema operativo Windows.	2 horas	Completada

7/4/2023	Se empieza a trabajar en el cliente, programando el socket. Y empleando la biblioteca winsock2.	5 horas	Completada
12/3/2023	Como nueva tarea se tendrá que trabajar en la interfaz gráfica del cliente en C, por lo que se inicia investigando librerías graficas de C	2 horas	Completada
12/3/2023	Se encuentra que una buena opción es trabajar con la librería Raylib por lo que se comienza el proceso de instalación e importación de esta en el juego.	3 horas	Completada
14/3/2023	Se hace la ventana principal del cliente, con sus respectivos botones “mirar” y jugar” y el enlace de estos a otras ventanas secundarias al presionarlos	3 horas	Completada
15/3/2023	Se coloca la nave, su movimiento de izquierda y derecha con el evento de las teclas, seguidamente el disparo	4 horas	Completada

	de proyectiles al presionar la tecla espacio. Se coloca los bunkers en la ventana.		
18/3/2023	Trabajar en el ovni, su movimiento de izquierda a derecha y que aparezca en pantalla por el momento solo si se presiona la tecla “O”	4 horas	Completada
18/3/2023	Diseño del manual de usuario	2 horas	Completada
18/3/2023	Se realiza en conjunto la documentación técnica	4 horas	Completada
21/3/2023	Revisión de detalles finales para entregar la Tarea 3	2 horas	Completada
21/3/2023	Ultima reunión para preparar la defensa.	1 hora	Completada

Bitácora Kevin Lobo

Diario de Trabajo			
Fecha	**Actividades realizadas**	**Duración**	**Estado**
05/04/2023	Asignación de la tarea de lógica del juego, planeamiento general del proyecto	3 horas	Completado

12/4/2023	Creación de la ventana de juego, además de investigación de la instalación de raylib	6 horas	Completado
12/4/2023	Creación de la nave	30 minutos	Completado
13/4/2023	Creación de los invasores, individuales.	3 horas	Completado
14/4/2023	Creación de los movimientos y los límites de la pantalla donde en la pantalla se van a dibujar los invasores.	8 horas	Completado
15/4/2023	Investigación de código para trabajar los invasores por fila y columnas.	3 horas	Completado
15/4/2023	Investigación de los structs para formar una lista de invasores	4 horas	Completado
18/4/2023	Creación de los movimientos de los demás invasores, además de incluirlos a una matriz	4 horas	Completado
19/4/2023	Detección de colisiones de todos los invasores con los disparos de la nave.	6 horas	Completado

20/4/2023	Arreglo de bugs con el movimiento de los invasores, se designa el cambio de movimiento para cada uno los tipos de invasores.	2 horas	Completado
21/4/2023	Se crea la función que lleva puntaje.	10 minutos	Completado
21/4/2023			
21/4/2023			

Bitácora Nicol Otárola

Diario de Trabajo			
Fecha	**Actividades realizadas**	**Duración**	**Estado**
06/04/2023	Investigación de como implementar sockets en Windows y que librerías sirven para ello.	2 horas	Completado
07/04/2023	Se crea el servidor en java, además, creé un cliente en java para hacer pruebas de conexión y envió de datos sencillos.	5 horas	Completado
12/04/2023	Se investiga sobre lo que es hacer interfaz gráfica en java.	1 hora	Completado

13/04/2023	Se comienza utilizando un Form para realizar la interfaz, pero debido a errores en la compilación, se realiza de manera manual el código referente a la ventana.	6 horas	Completado
14/04/2023	Instalación de la librería Raylib para posteriormente trabajar en C.	2 horas	Completado
18/04/2023	Trabajar en las colisiones de las balas con los invasores.	6 horas	Completo
19/04/2023	Terminar de verificar las colisiones.	3 horas	Completo
21/04/2023	Función para verificar puntos	30 minutos	Completo
21/04/2023	Trabajar en el documento escrito, en el desarrollo y explicación de diagramas, y lo referente a los problemas.	5 horas	Completo
21/04/2023	Revisión del proyecto para la entrega.	1 hora	Completo

Evidencias de reuniones

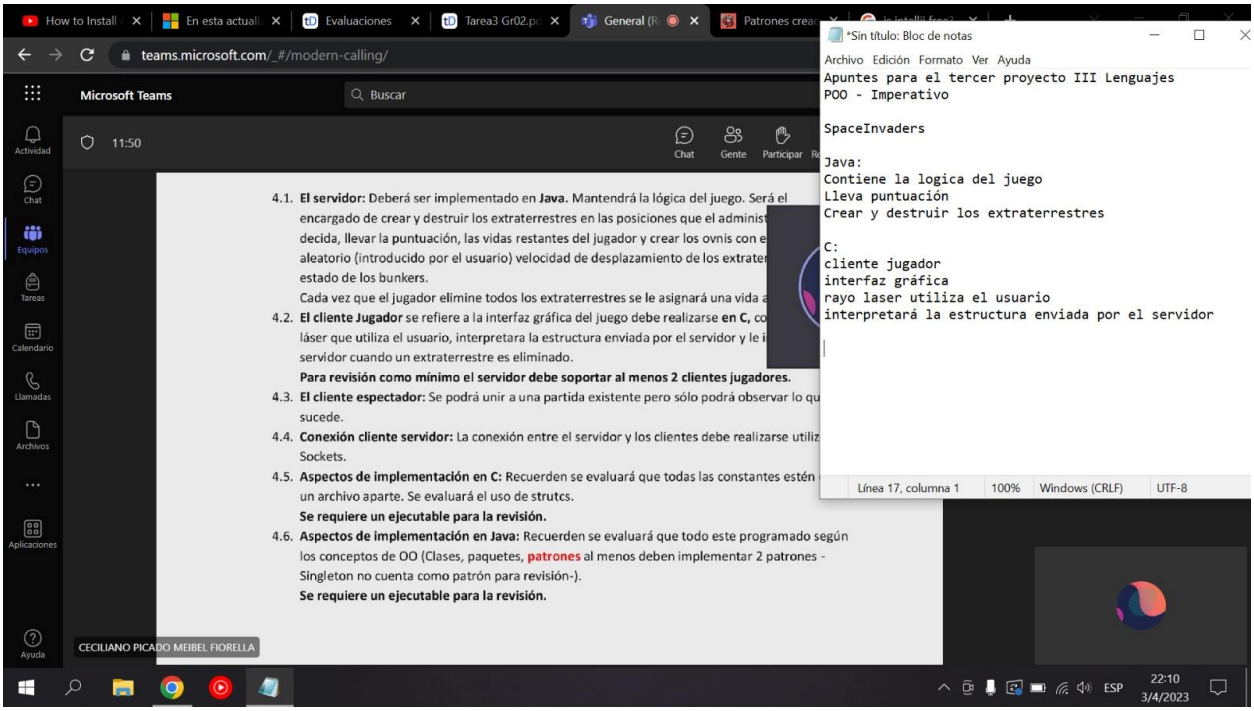


Figura #1: Reunión inicial para definir roles y tareas.

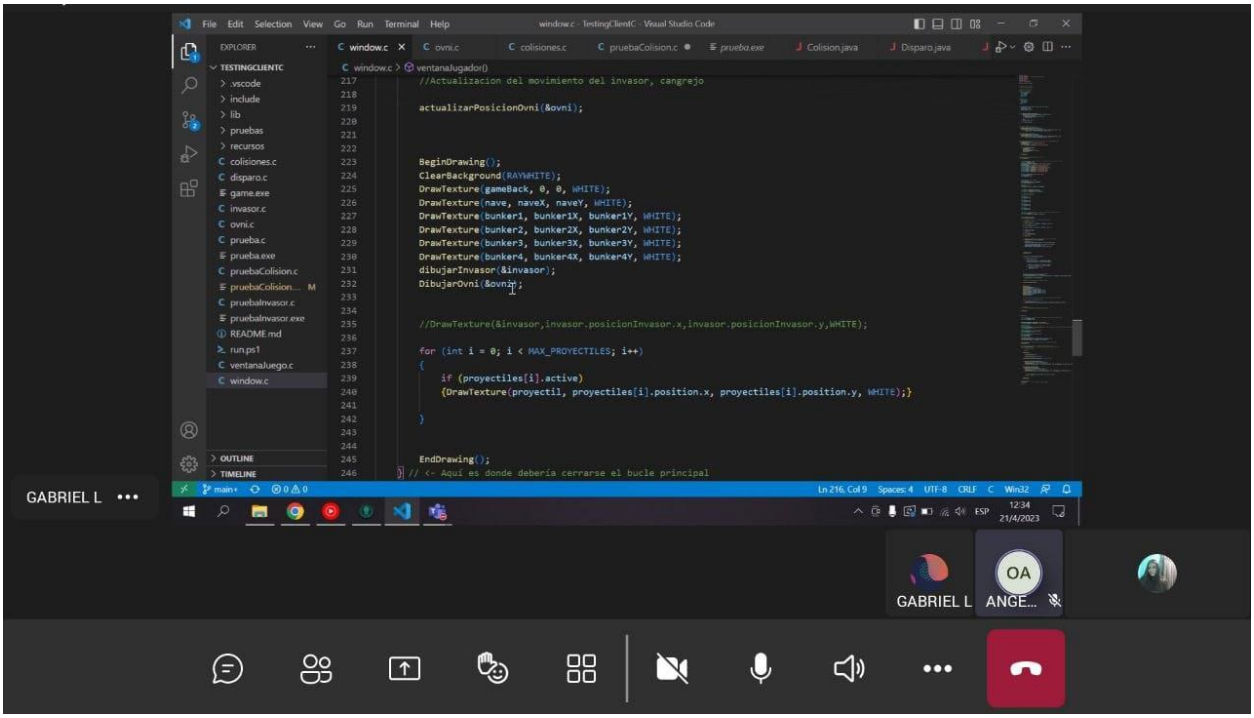


Figura #2: Conectar cliente-servidor , revisión de una parte de la lógica

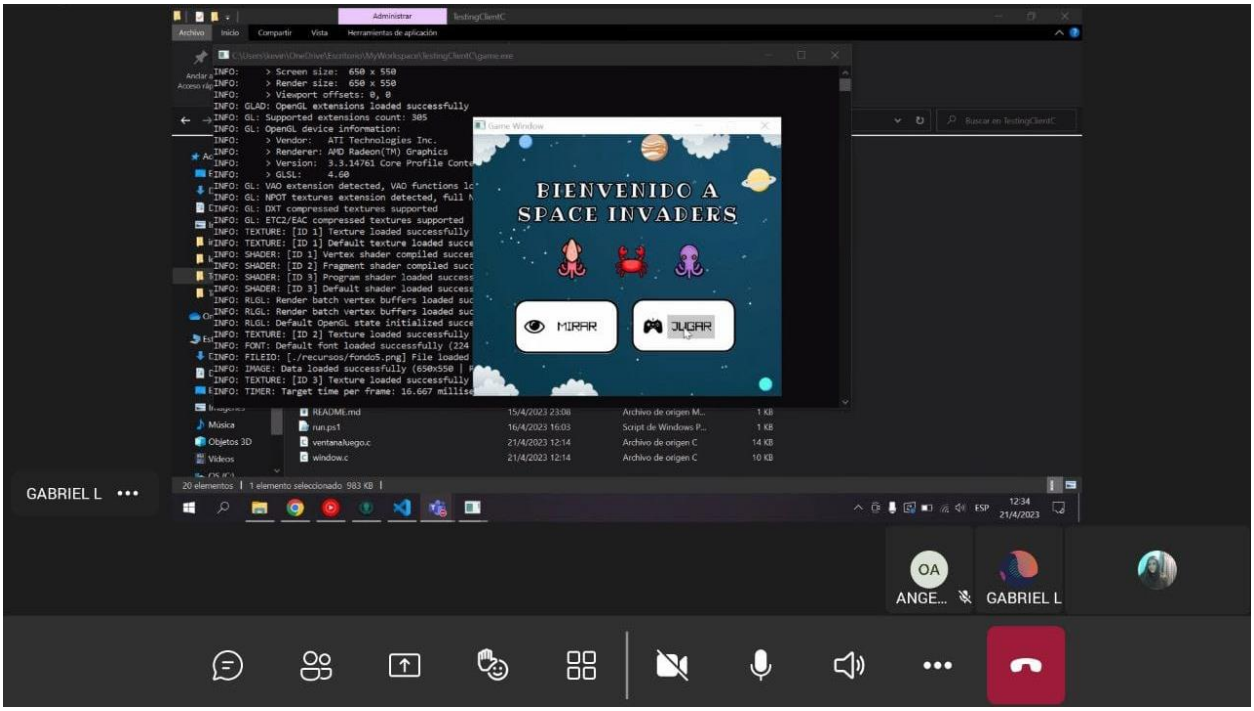


Figura #3: Venta principal del cliente y servidor listas.

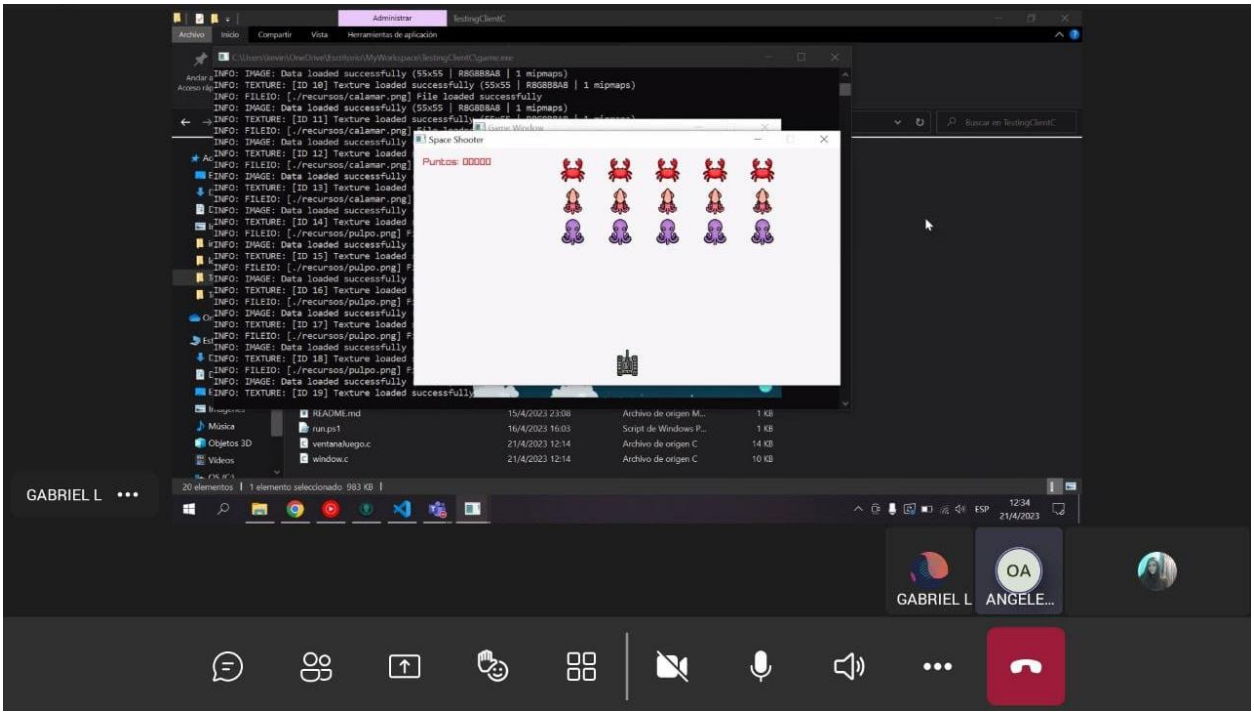


Figura #4: Verificación de que todos los archivos funcionan correctamente.

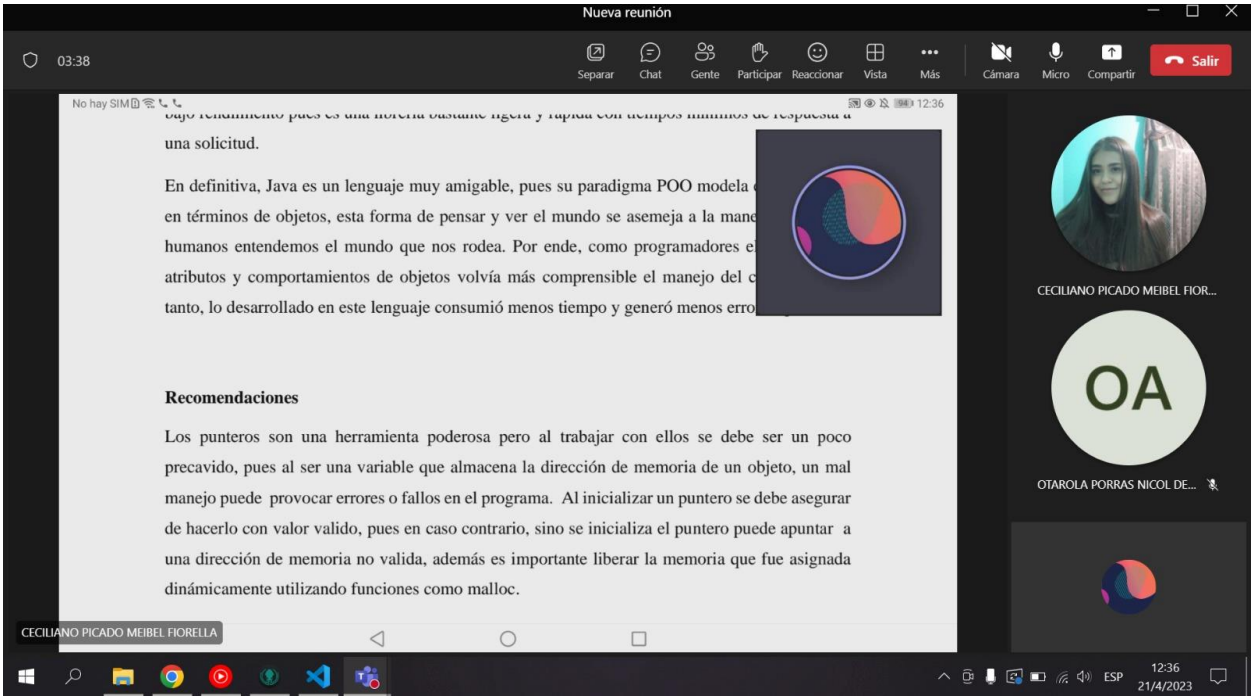


Figura #5: Revisión de secciones de la documentación

Enlace repositorio Github:

Para llevar un trabajo en línea y simultaneo se decide crear un repositorio en Github donde cada integrante subiera sus avances, aquí también se puede visualizar los commits frecuentes demostrando un trabajo constante.

https://github.com/gabrielwolfw/TestingClientC.git