Universidad Católica del Uruguay

Proyecto Final Programación 2

Grupo 7

Facultad de Ingeniería y Tecnologías

Integrantes: José Varela

Nicolás Lorenzo

Manuel Baldomir

Juan Campos

Docentes: Matias Olave

María Parapar

Felipe López

Universidad Católica del Uruguay

ÍNDICE

Funcionalidades añadidas...…… 3

Introduccion...…... ….5

Desarrollo……6

Funcionalidades añadidas

Se nos pide añadir nuevas funcionalidades al proyecto, por lo tanto se optó por agregar algunos modos de juegos diferentes al juego clásico y además agregarle algunas funcionalidades extras que se muestran a continuación.

- Bomb
- TimeTrial
- Challenge
- Quickchat
- Ranking

Bomb: El objetivo de esta clase es afectar directamente la forma básica de ataque que tienen los usuarios mientras juegan a la guerra naval. Al aplicar este modo, luego de que alguno de los jugadores erre 3 tiros seguidos en sus turnos correspondientes, su próximo disparo no será de solo 1 slot sino que será un cuadrado de dimensión 3x3.

TimeTrial: Esta clase afecta directamente en la modalidad de juego y en quien gana la partida. No solo gana quien hunde todos los barcos enemigos, sino también el que haya golpeado a más barcos enemigos en el tiempo límite que tendrá la partida contra reloj. Al finalizar el tiempo se verificará quien derribó y golpeó a más barcos enemigos y así se determinará quién ganó la partida si no fueron derribados todos los barcos. También afectará el tiempo que tiene cada jugador en realizar su ataque

.

Challenge: Tiene como objetivo realizar un reto con otro jugador y no solo jugar 1 partida entre sí, sino un mejor de 3.

QuickChat: Esta clase brinda la funcionalidad de poder intercambiar líneas rápidas de chat. Tendrá un pequeño repertorio de textos predefinidos que podrán ser utilizados por el jugador durante la partida para molestar al contrincante o simplemente ser amistoso.

Ranking: Con esta clase, siempre se va a mantener una única instancia de una lista de estadísticas de usuarios, ordenada según la cantidad de victorias de cada usuario y en caso de que coincidan, ordenadas también por su porcentaje de victorias. El administrador debe usar distintos métodos para mantener actualizada esta lista, mientras que el jugador puede ver el ranking global de todos los usuarios, el top 10 o su propia posición en el ranking global.

Introducción

Para abordar este proyecto en el cual se debería crear o implementar un juego clásico llamado "Batalla Naval" para la primera entrega se decidió realizar un diagrama UML para así tener un primer acercamiento a lo que sería la solución de este. Para esta segunda entrega nos vimos forzados a realizar cambios en dicho diagrama, ya que nuestra visión para el correcto funcionamiento del programa tomó una mejor y más específica forma al comenzar a escribir su código.

La solución tiene varias clases para facilitar su funcionamiento desde el momento en que ingresa un usuario hasta que finaliza el juego.

Debido a que en esta entrega lo solicitado fue que el juego funcionara por Telegram comenzamos con la implementación del bot, lo cual fue bastante compleja y complicada a la hora de llevarlo a código debido a que no teníamos conocimientos previos sobre el bot de telegram.

Comenzamos leyendo documentación sobre como implementar el bot y luego viendo ejemplos que nos dieron los docentes logramos poco a poco ir implementando el bot.

A la hora de decidir si llevar nuestro codigo solo a telegram o dejarlo implementado aun así en la consola, optamos por la opción de tener los dos modos, es decir, modificando algunas lineas básicas dentro del codigo podemos obtener la opción ejecutar el codigo por consola o directamente por telegram.

Nos pareció una muy buena opción que nuestro proyecto tuviese esta cualidad de ejecutarse por donde eligiéramos, pero nos trajo grandes consecuencias y problemas a la hora de ejecución lo cual nos atrasó bastante con la implementación, aún así, logramos dejar el código funcionando y jugando los diferentes modos de juego.

Durante el transcurso del desarrollo del código se nos presentaron varios problemas los cuales supimos superar y continuar.

Desarrollo

Durante el avance del proyecto, nos fuimos encontrando con distintas dificultades para llevar a cabo el objetivo de crear un juego de guerra naval capaz de ser manipulado a través de telegram y tener distintos modos de juego.
Uno de los primeros fue el de evitar la rigidez de código, el cual fue detallado por los profesores y no fuimos capaces de resolverlos. Esta rigidez iba enfocada a que si en un futuro deseábamos agregar un nuevo modo de juego iba a ser necesario modificar la clase Menu y también Administrator lo cual es lo ideal. Tal y como mencionamos en la segunda entrega, intentamos resolverlo con un Singleton de Administrator y de allí adquirir los distintos modos de juego pero no fue posible. Tal fue así que volvimos a la solución anterior y dejamos un diccionario en Administrador que guarda la información del usuario y el modo elegido y en función de ese valor ir emparejando jugadores.

Otra de las problemáticas grandes que tuvimos fue que la gran mayoría de nuestros métodos o funciones eran *void* y eso nos generó muchos problemas a la hora de querer realizar tests que demostraran la correcta operación del juego. Para la tercera entrega mejoramos en este aspecto, ya que para el correcto funcionamiento del bot y los handlers fue necesario modificar algunos de estas funciones y que retornen elementos que serían utilizados por la lógica del juego y fomentar la reutilización de código. Ésto último nos permitió hacer una mayor cantidad de tests que en la entrega anterior.

Al realizar los distintos modos de juego, una de las problemáticas encontradas fue la manipulación de los atributos de cada clase debido a la herencia configurada. Esto nos llevó a veces a reutilizar código para que funcionaran los distintos modos debido a la urgencia de lograr que funcionaran correctamente.

En cuanto a los Handlers su funcionamiento es bastante básico, se basa en una cadena la cual una vez es invocada con una palabra clave, la va recorriendo hasta encontrar un handler que pueda manipular esa condición o esa palabra clave, al encontrar un handler el cual pueda manipular esa palabra clave, recurre a entrar en él, luego este ya teniendo un comportamiento definido realiza diferentes operaciones, la gran mayoría de estas operaciones son interacciones con el usuario por medio de mensajes.

Nos chocamos con diferentes problemas al momento de interactuar con los usuarios debido a que podían ser más de un usuario los que estaban escribiendo o realizando alguna acción, lo cual llevaba a confusiones a nivel del código, es decir, si ambos intentan realizar una acción a la vez, o envían la misma palabra clave, el Handler no es capaz de diferenciar quien lo está enviando y puede almacenar cambiar los datos de los usuarios con los que interactúa y demás. Esto era bastante importante por lo tanto recurrimos a solucionarlo asignando estados al usuario, de esta tarea se encarga el administrador ya que es quien tiene conocimiento general sobre los usuarios.

Cuando le asignamos estados a los usuarios no estamos refiriendo a que en cada parte o sección del juego que el usuario va ejecutando, se le va a ir asignando diferentes tipos de estados, esto nos fue bastante útil para saber en qué parte del juego se encontraba el usuario, para que al interactuar con varios usuarios no se entreveran los mensajes y de lugar a errores.

Un claro ejemplo que nos pasó durante el desarrollo del proyecto sobre lo comentado anteriormente fué al momento de posicionar los barcos, ya que se pedían varias coordenadas

y direcciónes antes de realizar varias validaciones, por lo tanto al estar más de un usuario a la vez ingresando constantemente diferentes mensajes al bot lo que producía era una confusión entre los mensajes y asignaba los barcos en el tablero del enemigo y demás.

Luego de haber adquirido esta solución de los estados logramos que el juego se comporte de la forma que nosotros necesitábamos y todos los estados fueran individuales por usuario.

Por otro lado otro de los grandes desafíos que nos enfrentamos fue el de hacer que funcione el handler de atacar, que sin dudas, fue uno de los más extensos.

Este tiene un comportamiento muy similar al handler de posicionar los barcos, debido a que interactúa de la misma forma con el usuario, pero tiene que hacer otra serie de validaciones las cuales eran bastante diferentes a las validaciones que realizamos en el handler de posicionar barcos.

Luego de realizar las validaciones correspondientes cuando el usuario quiere atacar, se llama al método atacar que se le pasan por parámetro las coordenadas y el jugador que realiza el ataque. A continuación de enviar los parámetros correspondientes, el método de atacar nos devuelve cuál fue el resultado del ataque, es decir, si había algún barco en esa ubicación nos devuelve “tocado”, si hundió algún barco nos devuelve “hundido” y si no logró alcanzar ningún barcos nos devuelve “agua”. Transcurrido esto , se procede a modificar los tableros y se le muestran a los usuarios para que sepan que fue lo ocurrido en el disparo además de enviarles el mensaje con lo sucedido.

El juego continúa su ejecución hasta que alguno de los jugadores llegue a 15 tiros acertados, ya que cuando un jugador logre alcanzar los 15 disparos acertados será porque hundió todos los barcos de su enemigo.

En cuanto a las diferentes modalidades de juego, nos enfocamos en el modo de juego base el cual denominamos “Game”, pero se les agrega algunas nuevas funcionalidades, para que estas funcionen también por telegram recurrimos a los handlers, estos se basan en la misma

lógica ya que cuando un usuario ingresa el modo que quiere jugar y se los lleva a jugar una partida, el proceso de posicionar barcos y atacar es en todos exactamente igual.

Contratiempos

Debido a que tuvimos varias limitaciones nos vimos obligados a ponerle especial énfasis en poder jugar el modo clásico y que el juego se comporte como corresponde, también nos pareció de gran importancia que se puedan registrar varios mensajes de usuarios a la vez y el bot no tenga lugar a confusión por lo tanto no logramos llegar con todas los modos de juegos funcionando correctamente, pero si con la mayoría de ellos.

Los modos de juego “Classic”, “Time Trial” y “Challenge” funcionan como corresponde y cumplen nuestras expectativas, pero en cuanto a nuestro modo de juego “Bomb” no logramos hacerlo funcionar por telegram como corresponde por falta de tiempo.

Ejecución

Para lograr ejecutar nuestro programa es necesario correr el proyecto y en telegram enviando el comando “Iniciar” nos responderá un mensaje el cual procedemos a registrarnos, luego nos lleva a otro menu por donde seleccionamos que opción deseamos realizar, en caso de seleccionar “jugar”, nos llevará a seleccionar el modo de juego.

Una vez seleccionado el modo de juego nos dejara esperando hasta que otro jugador quiera jugar a ese mismo modo de juego, en el momento que otro jugador se una a nuestra partida, en ese instante comenzará el juego.

Lo siguiente es recurrir a posicionar los barcos y finalmente jugar hasta derribar todos los barcos del enemigo!.

A tener en cuenta

-Si se quiere ejecutar el bot hay que setearle “Administrador.Instance.BotEnabled = true;” en program y dejar comentad la parte de consola.

-Si se quiere dejar por consola hay que setearle “Administrador.Instance.BotEnabled = false;” y dejar descomentado lo de consola, tambien tener en cuenta que hay que descomentar el método “SnowBoard” de la clase “ Game”.

-Se debe tener el token del bot especificado en la carpeta que nos propusieron los docentes.

-En caso de que ocurra algún problema relacionado a la carpeta “appsettings” esta debe estar contenida en la ruta que se pide por consola al momento de lanzar el error y debe contener la información que aparece a continuación:

```
{ "BotSecret": { "Token": "Set in Azure. For development, set in User Secrets" } }
```