Informe: Hundir la flota

Asignatura: Programación-I

Profesor: Juan Antonio Montes de Oca

Proyecto hecho por: José Miguel Perelló Ruiz y Miquel Perelló Pons

Objetivo

El objetivo de la práctica es recrear el juego “hundir la flota”, usando diferentes clases para facilitar su creación. El juego consiste en atacar el tablero del rival por turnos hasta que uno de los dos haya hundido todos los barcos del otro, cada jugador cuenta con sus propio tablero (10x10) y los barcos se distribuirán de forma aleatoria mediante los **ficheros configuración 10-10-5-4-3-3-2.** Además se tendrá un registro de todos los datos de las partidas jugadas donde se almacenará el número de disparos a barcos hundidos, número de disparos a barcos tocados pero no hundidos, etc

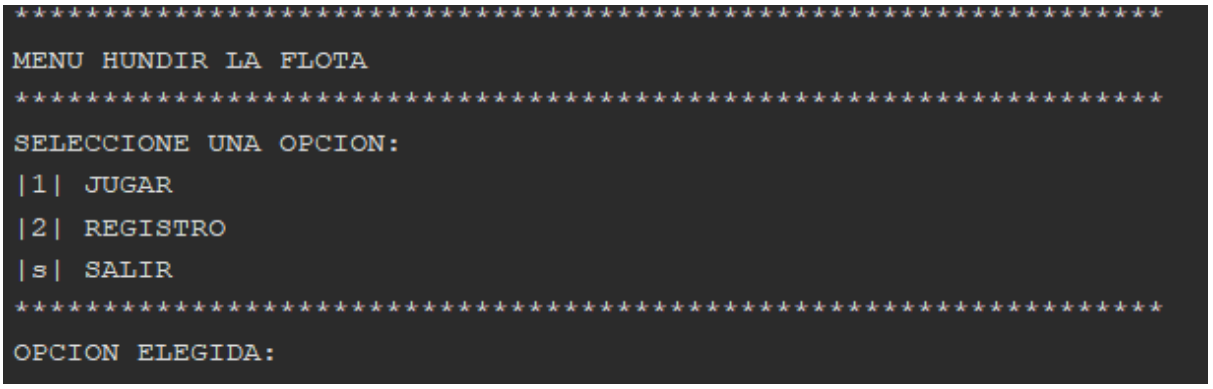
Hay 5 barcos por jugador (v), cada uno con diferentes medidas (5-4-3-3-2), en el caso de que solo se haya tocado una de las casillas del barco esta casilla se considerará tocada (t) y cuando todas las casillas de un barco hayan sido tocadas este se considerará hundido, al igual que todas sus casillas (x). En el caso de haber atacado una coordenada que no contenía ningún barco esta se considerará como agua (a).

Al principio de cada partida se pedirá el nombre del jugador para cuando haya finalizado la partida se guarden los datos de esta y el propio jugador pueda acceder a sus registros mediante el menú que se implementará para facilitar la navegación del usuario a través del programa.

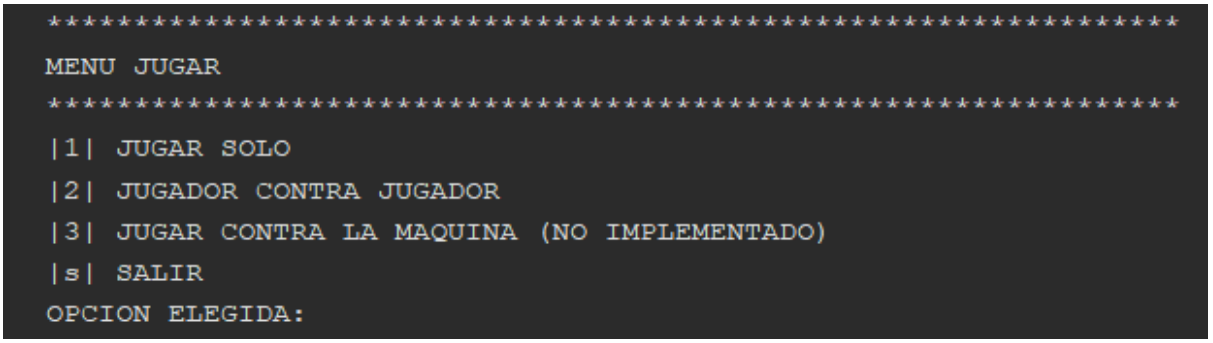
Para facilitar el traspaso del proyecto entre compañeros se ha usado un repositorio en la página Github para agilizar la programación del proyecto sin tener que estar constantemente enviando archivos de un compañero al otro (No se ha quitado el archivo de la carpeta porque no interfiere en la ejecución del proyecto).

Manual de usuario

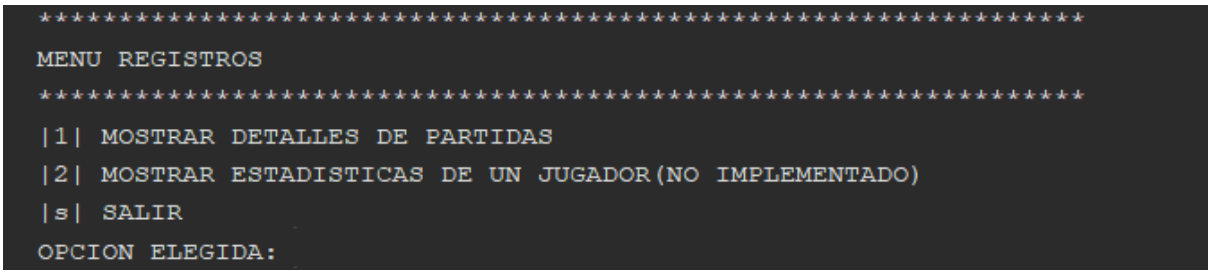
El programa, al contener menú con las diferentes opciones disponibles, facilita la estructuración de este. El menú permite dar a conocer al usuario las diferentes opciones que tiene: Jugar(1), Registros(2), Salir(s).



El jugador escribe una de las opciones en el programa y se muestra otro menú, en el caso de que haya elegido la opción Jugar(1) se muestran las opciones: JugarSolo(1), JugadorContraJugador(2) y JugadorContraLaMaquina(No implementado)(3),Salir(s).



En el caso de que haya elegido la opción Registros(2) se muestran las opciones: MostrarDetallesDePartidas(1),MostrarEstadístiscasDeUnJugador(2),Salir(s).



En el caso de que haya elegido la opción inicial Salir(s), el programa se cierra si está en el menú principal, pero si está en algún menú secundario volvería al principal.

Cuando el jugador elige una de las opciones del menú se inicia el método que lleva a cabo la opción seleccionada.

Los métodos menuJugar() y menuRegistros() visualiza un menú (mostrado en esta misma página) con diferentes opciones a elegir que el jugador podrá seleccionar.

Dentro del Menú de Registros no está implementada la función de Mostrar Estadisticas de un Jugador, cuando se seleccione está opción saldrá un mensaje que lo indicará.

Diseño descendente

Para poder realizar el proyecto primero, se definieron los aspectos clave, como la gestión de partidas, el almacenamiento de datos en archivos de texto y la estructura del tablero y los jugadores. A partir de ahí, el proyecto se dividió en varias partes, cada una con su función específica.

Para que todo estuviera bien organizado, se establecieron cinco puntos principales. La gestión de partidas y archivos se encargaría de guardar y recuperar información sobre las partidas jugadas, permitiendo consultarlas después. La parte de los jugadores definiría sus características esenciales, como el nombre, los barcos que tienen y los disparos que han hecho. El tablero y los elementos del juego es donde se desarrolla la partida, con sus casillas y barcos bien definidos. Además, se implementó un sistema de menú interactivo para que el jugador se pudiera mover fácilmente entre las opciones del juego, y finalmente, se programó el juego con los diferentes modos de partida.

Cada parte se descompuso en clases específicas para gestionar distintos aspectos del juego. Para manejar la información de las partidas, se creó la clase Partida(), que almacena los datos esenciales, como la fecha, la disposición de los barcos y el resultado de la partida. También se añadieron PartidaFicherosLectura() y PartidaFicherosEscritura(), que permiten leer y escribir estos detalles en un archivo de texto.

Por otro lado, el jugador está representado por la clase Jugador(), que almacena su nombre, su flota y los disparos realizados. Sus métodos permiten acceder a esta información y comprobar si todos sus barcos han sido hundidos.

El tablero del juego se gestiona con Tablero(), que representa la cuadrícula de 10x10 casillas donde se desarrolla la partida. Cada celda es un objeto de la clase Casilla(), que almacena su estado (si ha sido atacada o no) y su contenido (agua, barco intacto, barco tocado o barco hundido). Para los barcos, se creó la clase Barco(), que registra el número de casillas ocupadas, cuántas han sido alcanzadas y a qué tipo de barco pertenece.

Para que el usuario pueda navegar por las opciones del juego, se implementó la clase Menu(), que muestra las opciones principales: jugar, consultar registros de partidas anteriores o salir del juego. Dentro hay submenús jugar y registros que permiten jugar a los diferentes modos de juego o visualizar los detalles de las partidas.

Finalmente, la lógica del juego se encuentra en la clase Jugar(), donde están los dos modos principales: jugarSolo(), para jugar solo intentando adivinar donde están los barcos, y jugadorContraJugador(), para jugar contra otro jugador. En estos modos se controlan las reglas del juego, la actualización del tablero y la interacción entre los jugadores.

El juego es bastante claro: el jugador entra al menú, elige si quiere jugar o ver partidas anteriores. Si decide jugar, se configura el tablero y se colocan los barcos. Durante la partida, los jugadores atacan, el tablero se actualiza y se comprueba si alguien ha ganado. Cuando la partida termina, se guarda el resultado en un archivo para futuras consultas.

Algoritmos importantes

De la clase Jugar():

jugarSolo(): Primero inicializa los objetos partida y tablero, luego visualiza un mensaje al jugador para que introduzca su nombre y se visualiza el tablero para que empiece a jugar. A partir de este momento se realiza un bucle hasta que se cumpla el método todosHundidos() (es TRUE cuando todas las casillas de los barcos del jugador están hundidas). En este bucle se ejecuta el método Atacar() que permite al usuario introducir la coordenada del tablero enemigo que quiere atacar, luego se accede a la casilla enemiga, se usa el método registrarDisparos() para modificar los disparos realizados por el jugador y, mediante el método clasificarContenidoTableroRival() se ejecutan los métodos necesarios según el contenido de la casilla accedida. En el caso de que la casilla sea un barco se aumentará el número de casillas tocadas de este y, si todo el barco ha sido tocado, se convierten todas las casillas del mismo barco en casillas hundidas. En el caso de que haya una casilla sin contenido esta se convierte en una casilla de agua.

Luego de haber realizado los cambios en la casilla accedida se vuelve a mostrar el tablero del rival y se reinicia el bucle hasta que acabe la partida. Cuando la partida acaba se visualiza un mensaje para felicitar al jugador por su victoria (como juega solo siempre ganará) y se guardan los registros de la partida.

jugadorContraJugador(): Sigue la misma estructura que jugarSolo() pero con la diferencia de que hay 2 jugadores, en este caso se inician 2 objetos partida y tablero. Luego se insertan los nombres de los 2 jugadores y empieza un bucle que acaba cuando uno de los tableros haya sido hundido. El bucle está separado en dos

turnos, en los que se realizan las mismas acciones. Primero se visualizan los tableros, (el de la izquierda es el del jugador que ataca y el de la derecha el enemigo) luego se ataca el tablero rival (usando el método Atacar() mencionado en jugarSolo()), se accede a la casilla atacada y se registran los disparos realizados y recibidos por los jugadores (registrarDisparos() y registrarDisparosRecibidos). Luego se ejecuta el método clasificarContenidoTableroRivaJugadorContraJugadorl() para ejecutar los métodos necesarios y se registran los disparos del jugador (es igual al método del modo Solitario con unas pequeñas diferencias debido a que ahora son 2 jugadores). Luego pasa a ser el turno del otro jugador, donde se hace exactamente lo mismo. Cuando el turno del jugador 2 ha pasado se reinicia el bucle hasta que uno de los dos hunda todos los barcos del otro.

Cuando uno de los dos ha hundido todos los barcos del rival se visualiza el tablero del ganador y un mensaje para felicitarlo y se guardan los registros de la partida.

De las diferentes clases Partida:

De PartidaFicherosEscritura(), el más importante sería el método **escritura()**, ya que permite la escritura de los diferentes datos de la partida en el fichero de texto. También sería muy importante el método de **lectura()** de la clase PartidaFicherosLectura(), debido a que nos garantiza la lectura de los diferentes datos separados y crea un objeto partida con esos datos. Finalmente el **toString()** de la clase Partida(), y a su vez de la clase Jugador(), también es muy necesario ya que nos permite la realizar la visualización por pantalla de los datos de las Partidas.

Estructura de datos

En la carpeta del proyecto se encuentran 2 tipos de ficheros: ficheros de distribuciones 10-10-5-4-3-3-2, y el fichero que contiene los registros de cada partida (*DetallesPartidas.txt*).

Los ficheros de distribuciones 10-10-5-4-3-3-2 se usan al iniciar un tablero de forma aleatoria con el método distribucionRandom(), al escoger un número aleatorio se lee el fichero de distribución con el mismo número y se lee, caracter a caracter, el contenido del fichero mientras se escribe, al mismo tiempo, en la matriz que contiene las casillas del tablero. Al final del método se cierra el fichero y no se hace nada más con él.

El fichero de *“DetallesPartidas.txt”* almacena la información registrada de cada partida jugada. Cada vez que un jugador inicia una partida, se crean y utilizan diferentes instancias de la clase **Partida** para gestionar los datos relevantes del juego. Durante el transcurso de la partida, se registran los siguientes detalles: la **fecha y hora**, el **nombre del jugador**, la **modalidad de juego**, el **tamaño del tablero** y la **distribución de los barcos**. También se contabilizan los **disparos a barcos hundidos**, **barcos tocados** y **disparos recibidos** . Finalmente, se

almacena el **resultado de la partida** indicando si el jugador ha ganado o perdido. Al finalizar la partida, los datos se guardan en *“DetallesPartidas.txt”* con la clase PartidaFicherosEscritura(), manteniendo el orden de los parámetros y separándolos con *‘#’*, evitando sobrescribir partidas anteriores. Desde el **menú de Registros**, el usuario puede ver todas las partidas jugadas. La lectura se gestiona con PartidaFicherosLectura(), que reconstruye cada partida y permite visualizar su información de forma clara.

Conclusión

Después de definir el objetivo del juego, explicar las clases y detallar los métodos principales, el resultado ha sido un proyecto funcional. Desde el menú, los jugadores pueden acceder a los modos jugarSolo() y jugadorContraJugador(), además de que el sistema registra automáticamente los datos de cada partida para que puedan ser visualizados después en el menú de registros.

Este proyecto ha sido una buena oportunidad para aplicar conocimientos de *Programación - I* en un proyecto más grande. A lo largo del proceso han surgido varios errores que han necesitado horas de pruebas pero hemos llegado a las soluciones poco a poco, lo que nos ha ayudado a mejorar nuestra capacidad de resolución de problemas. Gracias a esto, en futuros proyectos será más fácil encontrar fallos y corregirlos de manera más rápida.