Programación II

Proyecto final

Battleship

Sección: 3.02

Integrantes:

Código	Apellidos y Nombres	% Participación
	Reyes Caceres, Angel Aaron	100
	Espinoza Hernandez, Gabriel Enrique	100
	Miranda Jaime, Jorge Enrique	100

Profesor: Estanislao Contreras Chavez

15/12/2021

LIMA - PERÚ

Índice de contenidos

Antecedentes

El proyecto Battleship tiene como finalidad evaluar los temas aprendidos en el curso Programación II ofertado por la escuela de ciencias de la Universidad de Ingeniería y Tecnología, UTEC. En esta, se replica el juego de mesa del mismo nombre con la diferencia que se pasa a un entorno virtual, con la capacidad de juego 1vs1 online, máquina vs jugador, máquina vs máquina. En la metodología y desarrollo del mismo, se aplicará un gráfico UML donde figuran las clases usadas y las partes más importantes del código. Para finalizar, se brindarán conclusiones al terminar el informe, junto con la bibliografía usada para explicar conceptos teóricos.

1. Fundamento teórico

1.1. POO (explicar cada una de estas y usar fuentes para anexarlas al final, en la bibliografía)

Programación orientada a objetos es un paradigma de programación, un modelo o estilo de programación en el que se organiza el diseño del software en objetos en lugar de funciones. Esto es eficiente cuando los programas que se desarrollan son grandes, complejos y se mantendrán en actualización activamente. (IBM. s/f.)

1.2. Librerías

Una librería es una serie de funcionalidades desarrolladas y compiladas en un determinado lenguaje de programación, las cuales sirven como métodos o interfaces para realizar la funcionalidad que se requiere. (Zorita, A. 2016)

1.3. Funciones

En programación, una función es una sección de un programa que calcula un valor de manera independiente al resto del programa. (USM. s/f.)

1.4. Clases

Son la representación de un tipo particular de objeto. Cada clase posee atributos y métodos que definen el comportamiento de la misma. (Pavón, J. 2007)

1.5. Punteros

Un puntero es una variable que almacena la dirección de memoria de una variable u objeto. Se usan para asignar la variable u objeto al heap, para pasar funciones a otras funciones, etc. (Microsoft. 2021)

1.6. Vectores

Los vectores son estructuras de datos similares a los arreglos, pero más desarrollados, ya que entre otras cosas, crecen y decrecen dinámicamente, según se necesite. (UNR. 2006)

1.7. Herencia

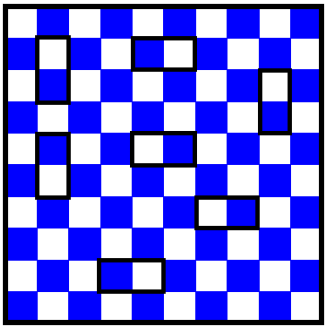
La herencia es el mecanismo de implementación mediante el cual elementos más específicos incorporan la estructura y comportamiento de elementos más generales. (Cachero, C., Ponce de León, P. s/f.)

1.8. Funciones Amigas

Una función friend de una clase es una función que no pertenece a la clase, pero que tiene permiso para acceder a sus variables y funciones miembro privadas por medio de los operadores punto (.) y flecha (->), sin tener que recurrir a las funciones miembro públicas de la clase. Si una clase se declara friend de otra, todas sus funciones miembro son friend de esta segunda clase. El carácter de friend puede restringirse a funciones concretas, que pueden ser miembro de alguna clase o pueden ser funciones generales que no pertenecen a ninguna clase. (CCIA UGR . s/f.)

2. Metodología y Desarrollo

Definiciones:
Sean A y B casillas del tablero, A y B son **vecinas** si es que ambas casillas tienen un lado en común.
Sean A, B y C casillas del tablero, A y C son **opuestos** respecto a B**,** si A y C son vecinos de B, y además los lados en común entre A con B y C con B son paralelos.

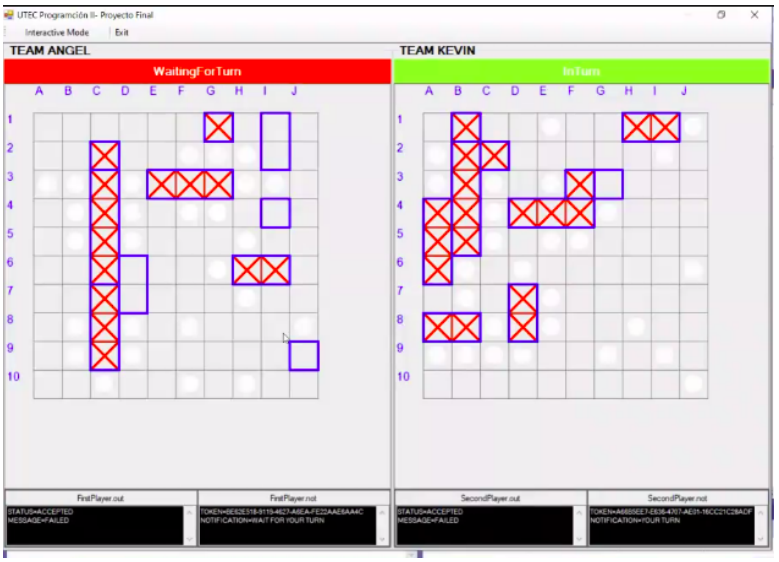


La estrategia principal del juego era encontrar, en el mapa enemigo, respuestas del tipo 'Y'. Para hacer esto se buscaba en coordenadas aleatorias del mapa hasta encontrarla. La estrategia que usamos fue dividir el mapa en dos colores al estilo de un tablero de ajedrez. Ya que todas las naves compuestas por más de una casilla se encuentran posicionadas en estos dos colores a la vez. Lo que nos aseguraría que encontraríamos las 6 naves de estas longitudes en menos de 50 movimientos. En general, las 'Y' que buscábamos son sólo aquellas que tengan al menos un vecino no explorado. Una vez encontrado un 'Y' con está condición, seleccionamos aleatoriamente uno de sus vecinos no explorados y realizamos el siguiente disparo en la coordenada seleccionada. Sin embargo, la selección anterior toma como prioridad aquel vecino que tiene como opuesto a una 'Y' respecto a la casilla seleccionada. Si no se encuentra ninguna 'Y' con estas condiciones, entonces se sigue buscando de forma aleatoria.

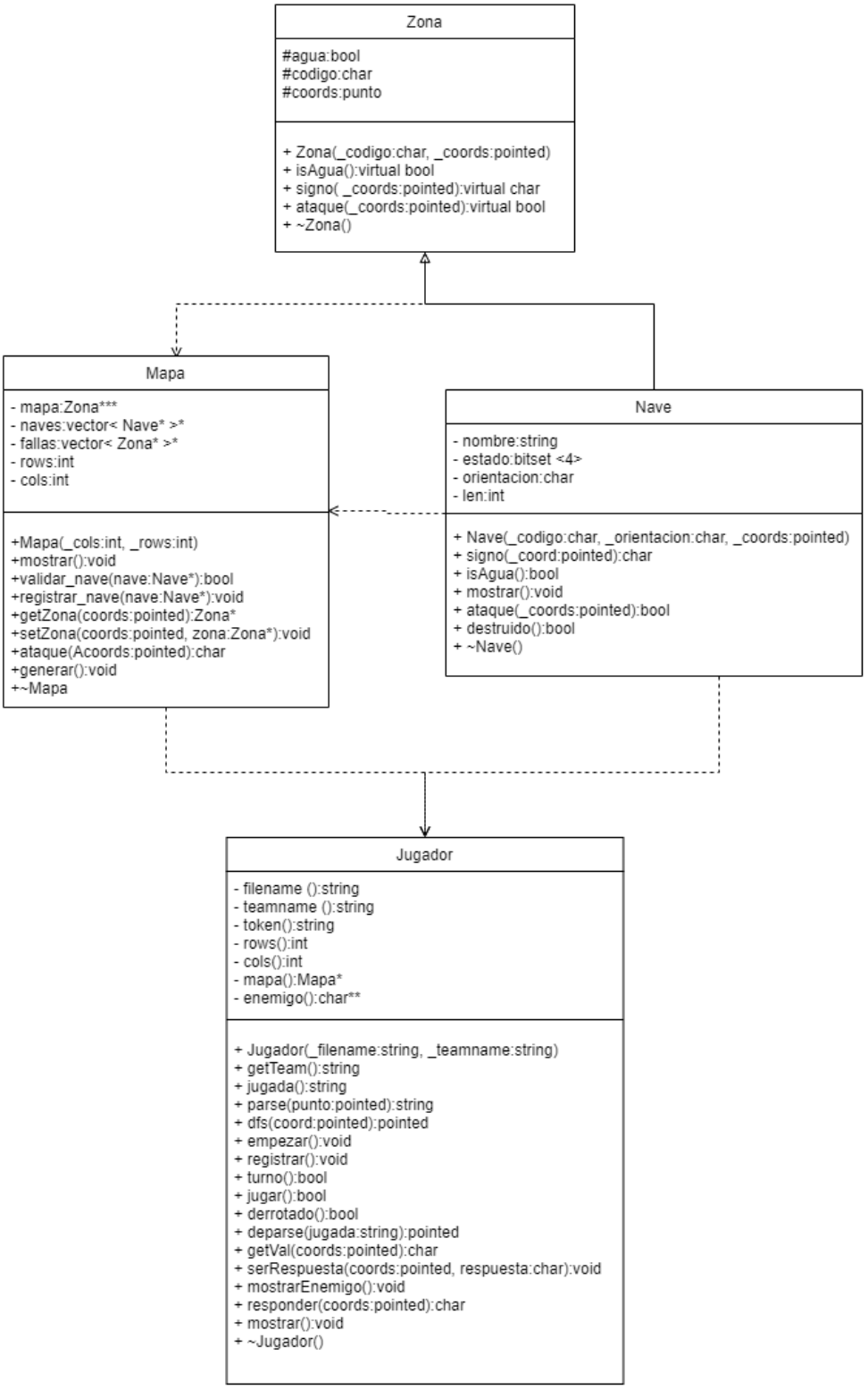
Notar que como existen casillas compuestas una sola casilla, el proceso podría no encontrarlas hasta al final del juego, es decir, al alcanzar las 100 jugadas. Ya que encontrar estas naves de una sola casilla es un proceso completamente aleatorio.

```
Turno: 68 de Team Kevin
Team Kevin ataca a Team Angel con B5
Respuesta: .
    A  B  C  D  E  F  G  H  I  J
1   .  Y  Y  Y  X     .     .
2      .  .  .     .     .     .
3   .     .  .  .     .     .
4        .  Y  Y  X  .     .
5   .  .  .  .  .     .     .
6        Y     .     .     .  X  .
7   .  X  .     .     .  .  Y  X
8   .  Y  .  .     .     .  .  .
9   .  .  .     .  .  X     .
10       .     X  Y  Y  .  .     .

Turno: 69 de Team Angel
Team Angel ataca a Team Kevin con I3
Respuesta: X
    A  B  C  D  E  F  G  H  I  J
1   X     .     .     X     .
2      .     .     .  Y  .     .
3   .     .     .  .  Y  X  X
4        .  .  .     .  .  .     .
5   .  Y  Y  X  .  .  Y  X  .
6        .  .  .     X  Y  X     .
7   .     .     .  .  Y  .  .
8        .     .     .  X  .     .
9   .     .     .  .  .  .  .
10       .     .  .  Y  Y  Y  X  .

Angel derrotó a Kevin
```

El promedio de turnos es de 60, cuando enfrentamos de forma automática dos instancias de jugador.



2.1. Diagrama de Clases



2.1.1. Descripción de las Clases

Clase Zona:
Define el terreno del mapa. Generaliza cualquier objeto para que pueda ser ubicado en el tablero. Tiene la responsabilidad de pasarle algunos datos a su clase hija, nave.

Clase Nave:
Clase hija de zona. Define las piezas del juego (naves), su longitud, estado, orientación, etc.

Clase Mapa:
Clase que controla la matriz usada para representar el tablero del juego.

Clase Jugador:
Esta clase define al jugador. Se encarga de que el jugador posea un tablero propio y tenga la capacidad de jugar de forma autónoma.

Librerías usadas:
cstdlib: Usada en el proyecto para obtener las función random().
ctime: Usada para proveer una semilla a la función random().
tuple: Usada para hacer operaciones de pares ordenados.
windows.h: Usada para obtener la función Sleep() para crear tiempos de espera en algunas funcionalidades.
fstream: Lectura y escritura de archivos.
string: Manipulación de objetos string.
bitset: Manipulación de bits.
vector: Librería de arreglos dinámicos.

3. Conclusiones

- La programación orientada a objetos nos da herramientas claves para desarrollar procesos complejos con códigos simples y ordenados.
- Desarrollar archivos ".h" nos ayudó a repartirnos los deberes y crear códigos ordenados y entendibles para todos.
- El proyecto representaba un reto que nos enseñó mucho sobre el diseño de clases y funciones de forma adecuada para evitar la aparición de bugs u otros inconvenientes.

Bibliografía

Cachero, C., Ponce de León, P. (s/f.). *Herencia.* Universidad de Alicante.
Extraído de: https://rua.ua.es/dspace/bitstream/10045/15995/1/POO-3-Herencia-10-11.pdf (consultado el 15/12/2021)

CCIA UGR. (s/f.). *Clases y funciones friend*. Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada.
Extraído de :https://ccia.ugr.es/~jfv/ed1/c++/cdrom3/TIC-CD/web/tema18/teoria_7.htm (consultado el 15/12/2021)

Cortez, C. (2021, 1 diciembre). *Punteros (C++)*. Microsoft Docs.
Extraído de: https://docs.microsoft.com/es-es/cpp/cpp/pointers-cpp?view=msvc-170 (consultado el 15/12/2021)

Pavón, C (2007-2008). *Fundamentos de la Programación Orientada a Objetos. Objetos y Clases.* Dep. Ingeniería del Software e Inteligencia Artificial de la Universidad Complutense de Madrid.
Extraído de: https://www.fdi.ucm.es/profesor/jpavon/poo/1.1.objetos%20y%20clases.pdf (consultado el 15/12/2021)

IBM (s/f.). *Programación orientada a objetos*.
Extraído de: https://www.ibm.com/docs/es/spss-modeler/SaaS?topic=language-object-oriented-programming (consultado el 15/12/2021)

UNR. (2006). *Estructura de datos: Vectores*. Facultad de Ciencias Exactas, Ingeniería y Agrimensura de la Universidad Nacional del Rosario.
Extraído de :https://www.fceia.unr.edu.ar/estruc/2006/vector.htm (consultado el 15/12/2021)

USM. (s/f.). *Funciones.* Programación USM.
Extraído de: http://progra.usm.cl/apunte/materia/funciones.html (consultado el 15/12/2021)

Zorita, A. (14 de octubre, 2016). *Librería informática ¿De qué se trata?*. Bytecode.
Extraído de : https://bytecode.es/que-es-una-libreria-informatica/ (consultado el 15/12/2021)