DESARROLLO DEL PROYECTO BATTLESHIP

Juan Miguel López Piñero
Beatriz García Dueñas
Dustin White

Memoria descriptiva

1. Introducción

El proyecto Battleship consiste en reproducir el cásico juego de hundir la flota implementado en Python, trabajando de forma colaborativa y usando el sistema de control de versiones GitHub. Además, se ha empleado el entorno de desarollo Visual Studio Code y herramientas de gestión de proyectos como Miro y Trello.

La implementación del juego de Hundir la flota realizada en este proyecto sigue las siguientes reglas: se trata de un juego de dos jugadores: tú contra la máquina. Cada jugador, tiene un tablero, donde se posicionan sus barcos de manera aleatoria. Se juega con: 4 barcos de 1 posición de eslora, 3 barcos de 2 posiciones de eslora, 2 barcos de 3 posiciones de eslora,1 barco de 4 posiciones de eslora. Ambos jugadores, (el usuario y la máquina) tienen un tablero con sus barcos, oculto para el adversario, y se trata de ir "disparando" y hundiendo los del adversario hasta que un jugador se queda sin barcos, y por tanto, pierde. El juego funciona por turnos y empiezas tú. En cada turno, disparas a una coordenada del tablero adversario. Si aciertas, te vuelve a tocar. En caso contrario, le toca a la máquina. En los turnos de la máquina, las coordenadas del disparo son aleatorias, si acierta también le vuelve a tocar.

2. Estructura del proyecto

El proyecto se divide en diferentes archivos y carpetas:

- **README.md** - archivo de guía para el usuario, donde podrá encontrar la descripción y organización del proyecto.
- **Main.py -** archivo dónde se ejecuta el juego.
- **Requirements.txt** - archivo dónde se indican las librerías necesarias y su versión para el juego.

➢ **Src** - Carpeta que contiene los archivos del desarrollo del juego

- **Variables.py –** Constantes empleadas en el juego
- **Classes.py –** Clases (Board y Game) para instanciar los principales objetos empleados en el juego. Cada clase cuenta con diversos métodos.
- **Funciones.py –** Archivo que contiene aquellas funciones no incluidas en las clases como por ejemplo obtener las coordenadas del jugador y los disparos del juego.

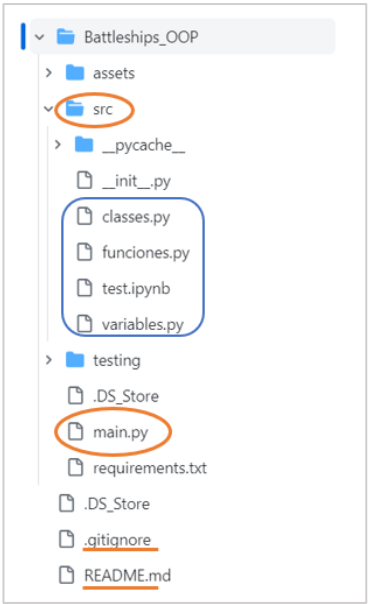
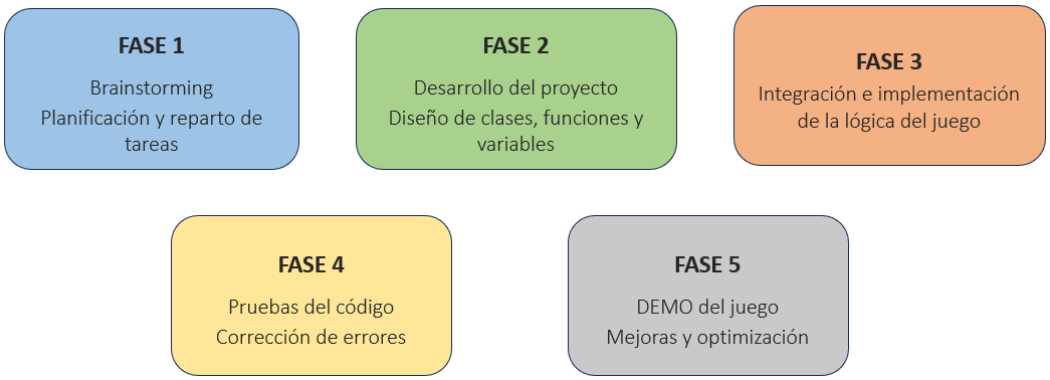


Figura 1. Estructura de carpetas y archivos del proyecto en git

Los archivos y carpetas no mencionadas, como 'test.ipynb' o 'testing' fueron creadas durante el desarrollo del juego, para comprobar el funcionamiento de las clases y funciones, pero no forman parte del código del juego.

3. Fases de desarrollo

Para el desarrollo del juego, se plantearon inicialmente 5 fases principales:



Fase 1 - Reunión inicial para poner en común ideas sobre la realización del juego. Planificación inicial en base a los objetivos y requisitos mínimos del juego y reparto de tareas.

Hitos: Documento Brainstorming;

Documento reparto de tareas (versión 1)

Reparto de tareas y subtareas (Documento final)

Finalmente, el reparto de tareas y subtareas del proyecto quedó de la siguiente manera:

- **Dustin**

Git : Crear el repositorio online y la estructura del mismo

Variables.py: Establecer las constantes del juego

Clase Board: Clase que inicializa un tablero y mediante métodos, coloca los barcos de manera aleatoria, en diferentes direcciones (N,S,E,O). Al colocar los barcos, se comprueba que no están superpuestos y que los barcos se pueden colocar sin salirse del tablero.

Main.py: Archivo desde donde se inicializa el juego.

Requierements.txt : Archivo con las librerías empleadas y sus versiones.

Presentación PowerPoint del proyecto

- **Juanmi**

Función get_guess: recibir coordenadas del jugador.

Función disparo del jugador: Creación de una función que recibe un input del jugador y comprueba si ha acertado en el disparo de la máquina.

README.md: archivo de guía para el usuario, donde podrá encontrar la descripción y organización del proyecto.

- **Beatriz**

Función display_boards: Imprime por pantalla el tablero

Función disparo máquina: A partir de unas coordenadas aleatorias, la máquina comprueba en el tablero del jugador si ha acertado en un barco o no. No se repiten coordenadas de disparo.

Función end_game: Función que comprueba si ha ganado alguno de los jugadores y en ese caso, se termina el juego.

Clase Game: Implementa la lógica del juego y usa las diferentes clases, funciones y constantes.

DEMO: Crear la demo del juego que se mostrará en la presentación.

Añadir funcionalidad salir del juego: Comando para salir del juego en cualquier momento.

Añadir funcionalidad barcos: Poner un símbolo diferente para barcos de diferentes tamaños.

Presentación PowerPoint del proyecto

Memoria del proyecto

Fase 2 – Desarrollo del código del juego. Diseño de clases , funciones y variables. Durante esta fase, los miembros del equipo trabajaron paralelamente en sus tareas gracias.

Hitos: Archivos variables.py, funciones.py y classes.py con el código desarrollado

Fase 3 – Implementación de la lógica y flujo que sigue el juego. Integración de todas las clases, funciones y variables en una clase 'Game'.

Hitos: Creación de la clase 'Game'

Fase 4 – Pruebas del código desarrollado y corrección de los errores encontrados.

Hitos: Creación de main.py. Actualización de las funciones, clases y variables

Fase 5 – Crear una DEMO del juego e implementar mejoras o funcionalidades extra del juego.

Hitos: DEMO funcional del juego. Juego final implementado en con los extra

Todas las fases y aportaciones en el desarrollo del código se fueron integrando gracias al sistema de control de versiones GitHub, en el cual se siguió el siguiente flujo de trabajo colaborativo:

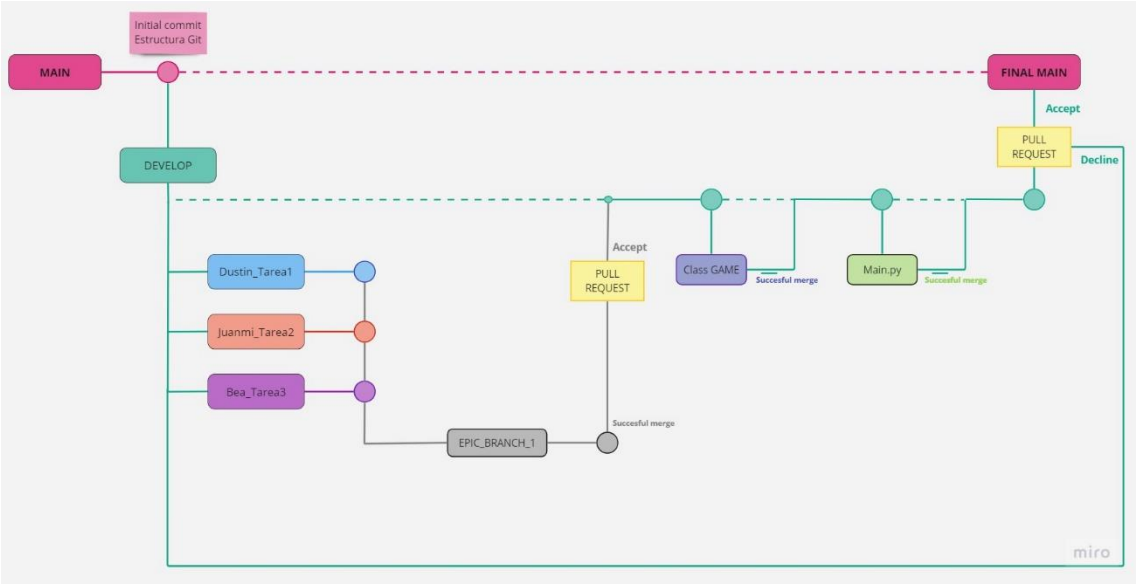


Figura 2. Workflow en GitHub

4. Resultados y conclusiones. Mejoras identificadas.

Tras la realización del proyecto, se consiguió crear un juego funcional y una DEMO interactiva para el usuario. Además, se pudieron implementar alguna funcionalidad extra para mejorar la experiencia del usuario como, por ejemplo, que el usuario pudiera abandonar el juego en cualquier momento y mostrar diferentes emoticonos de barcos para cada tamaño.

Sin embargo, aunque el resultado ha sido satisfactorio, se identificaron posibles mejoras del juego para el futuro:

- Mejorar la interfaz del juego, por ejemplo, añadiendo sonidos, una leyenda sobre los símbolos que aparecen durante el juego o imprimir los títulos de los tablero de una forma más visual.
- Añadir una funcionalidad al juego donde el usuario pueda escoger la duración de la partida (corta, media, larga) e iniciar un juego con tableros más grandes para partidas más largas y tableros pequeños (5x5) para partidas rápidas. Es decir, que la constante tamaño del tablero pudiera ser escogida por el jugador.
- Crear un algoritmo para que cuando la máquina acierte en su disparo, pruebe las coordenadas que tiene alrededor, tal y como lo haría el usuario en su turno.
- Añadir una funcionalidad al juego donde el usuario pueda escoger la dificultad del juego, en caso de escoger ‘Difícil’, se implementaría el algoritmo anteriormente mencionado.
- Límite de tiempo del juego, es decir, que el jugador tenga un tiempo para realizar su disparo, si no, pierde el turno.