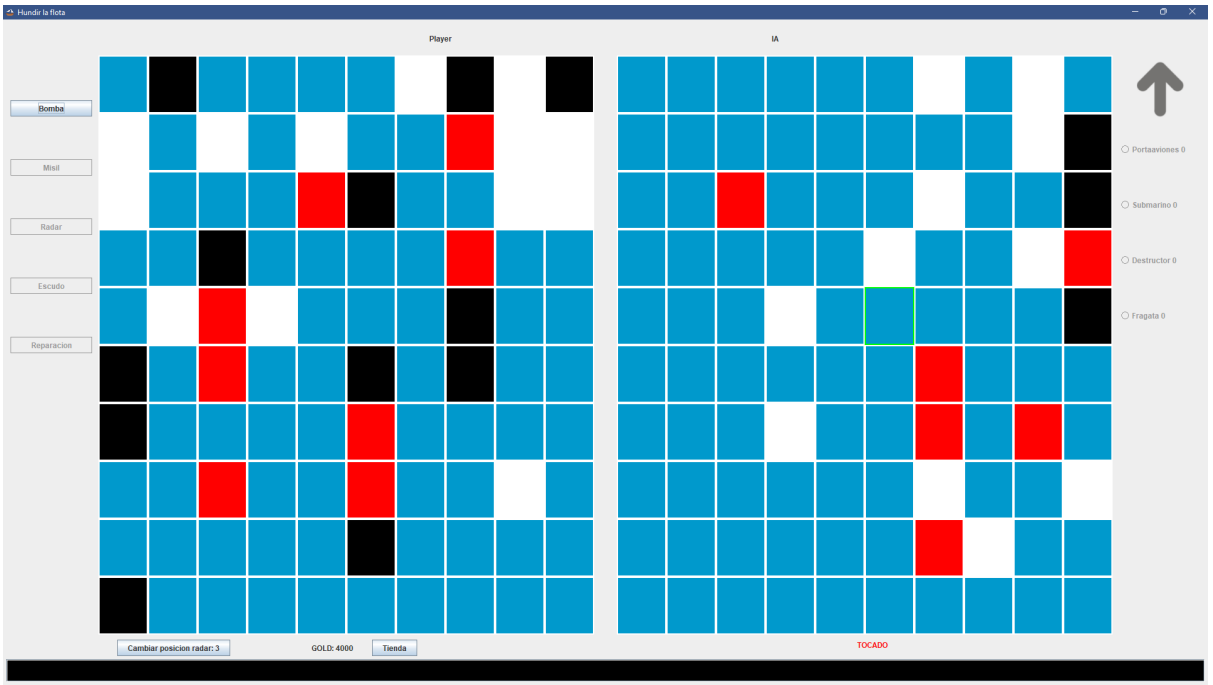
eman ta zabal zazu
UPV EHU

Ingeniería Informática de Gestión y Sistemas de Información
Ingeniería del Software

INGENIERÍA DEL SOFTWARE

Hundir la flota



Francisco González
Estefania Oñate
Diego García
Xabier Gabiña
Asier Santesteban
Javier Criado

ÍNDICE

INTRODUCCIÓN

Presentación del proyecto:

Cómo proyecto para la asignatura de Ingeniería del Software, se nos ha propuesto hacer un videojuego en Java. Dicho juego es el tradicional “Hundir la Flota” y su dinámica principal de juego es la siguiente:

- Tanto el jugador como el ordenador disponen de una flota formada por 10 barcos de distintos tipos: 1 portaaviones (cuya longitud es 4), 2 submarinos (de longitud 3), 3 destructores (de longitud 2) y 4 fragatas (de longitud 1).
- El jugador dispone de dos tableros de tamaño 10x10 (Uno para él y otro para la IA).
- Los barcos pueden colocarse tanto en posición horizontal como vertical, pero nunca pueden estar juntos, es decir, cada barco debe estar totalmente rodeado de agua.
- Las casillas vacías son agua
- La IA colocará sus barcos de forma aleatoria
- Existe una tienda donde se podrán adquirir objetos utilizables a cambio de dinero.
- Existen 4 tipos de objetos utilizables:
    - Radar: Visualiza una zona aleatoria 3x3 del mapa enemigo
    - Bomba: Si impacta en un barco daña la casilla impactada, y si impacta en el agua esa casilla queda marcada.
    - Misil: Funcionamiento similar a la bomba, sin embargo, si uno de estos impacta en el barco, éste queda directamente destruido.
    - Reparación: Arregla un barco tocado o hundido.
    - Escudo: Refuerza el barco elegido con un escudo que aguante un ataque.

- Al comienzo del combate, las flotas disponen de una cantidad predeterminada de armamento y de dinero (ambas flotas poseen lo mismo).
- El juego se desarrolla por turnos, comenzando siempre el jugador.
- Tras un disparo, la flota adversaria (ordenador) decide si desea gastar dinero en comprar armamento
- A continuación, el turno pasa al ordenador, repitiéndose el proceso: el ordenador podrá consultar su radar y disparar con su armamento sobre alguna posición de la flota del jugador.

Además de todo esto, se han añadido adicionalmente las siguientes características:

- Barra de comandos: Agiliza el proceso de realización de pruebas. Se han creado 4 comandos diferentes, el primero de ellos es “see_all”, una vez introducido este código lo que hará será mostrar el tablero enemigo. El segundo comando es “random” que coloca los barcos aleatoriamente, el tercero sería “idfka” que nos daría armas infinitas y para terminar “motherlode” que nos daría 50.000 monedas.
- Diferentes escalas de juego: Originalmente el juego dispone de tableros 10x10. En nuestro caso, el jugador puede elegir si jugar tableros 10x10, 20x20 o 30x30.
- Sonidos: Se da la posibilidad de usar sonidos en el juego (No recomendable).

Aspectos importantes:

Este proyecto sigue un proceso inspirado en la metodología ágil Scrum, por lo cual se han realizado 3 sprints en la que se nos pedían diferentes requisitos:

- Sprint 1:
  - ➔ Inicializar el juego
  - ➔ Disparar el jugador
  - ➔ Disparar el ordenador

- Sprint 2:
  - ➔ Activar escudo jugador
  - ➔ Activar escudo ordenador
  - ➔ Consultar radar jugador
  - ➔ Consultar radar ordenador

- Sprint 3:
  - ➔ Reparar barco jugador
  - ➔ Reparar barco ordenado
  - ➔ Comprar armamento jugador
  - ➔ Comprar armamento ordenador

Para la realización del proyecto se han tenido que utilizar diferentes conocimientos de los adquiridos en la asignatura. Entre esos conocimientos se encuentran el patrón arquitectónico Modelo-Vista-Controlador, patrones de diseño y el uso de java 8 junto con expresiones lambda.

GESTIÓN

Hemos llevado la cabo a gestión del proyecto documentando reuniones, tanto entre los integrantes como con el cliente, a través de las plantillas y ejemplos a seguir que nos han sido otorgados, subdivididos en las siguientes categorías:

Reparto de tareas:

En la primera reunión hecha para cada Sprint a entregar se han establecido las tareas a completar y quiénes llevarían a cabo las mismas hasta llegar la hora de entregar. En esta documentación, cuyo contenido se puede observar con más detalle el apartado 4 “Desarrollo”, hemos ido añadiendo la cantidad de horas estimadas para completar cada tarea a continuación de las finalmente requeridas, lo cual nos ha servido de guía para mejorar la estimación de horas y observar de forma general qué errores han sido más problemáticos a cada entrega realizada. Durante el desarrollo ha habido algún cambio repentino de tareas,en tanto a ofrecer ayuda a otro miembro del proyecto, pero por lo general cada miembro se ha centrado en completar su parte.

Reuniones:

Las reuniones se han dividido en dos apartados para potenciar la efectividad en las tareas a desarrollar y llevar un registro de los problemas que han ido surgiendo, siendo estos dos apartados por un lado las “actas de reunión” enumeradas de menor a mayor, y por otro lado las de “Reunión con el cliente”, que han sido completadas tras entregar y explicar al cliente cada Sprint.

Los documentos “acta de reunión” son una versión simplificada de los documentos entregados en cada Sprint, donde se ha ido recogiendo periódicamente los miembros que han asistido a las reuniones, que tareas eran necesarias ir completando dependiendo de su urgencia y un recordatorio de a que miembros les correspondía hacer dichas tareas. A

diferencia de en la documentación entregada para los Sprints, en estos documentos no se han recogido las estimaciones de horas para evitar la monotoneidad de reiterar la misma información en demasía.

El los documentos “Reunión con el cliente” se han ido recogiendo los comentarios del cliente respecto a los diversos apartados que se iban exponiendo y analizando durante las reuniones tras entregar su respectivo Sprint, con la intención de tener una lista de tareas con la que enfocar nuestro diseño e implementación de subapartados y un espacio reservado para las reflexiones y conclusiones de por qué nos han o no surgido los variopintos problemas que nos ha ido indicando el cliente.

Recursos:

Los recursos utilizados para la creación del proyecto en cuanto a implementación ha sido para la parte del diseño “**Eclipse java**” y aunque para el resto de la implementación también se ha usado, es cierto que para ejecutar el código más rápido, hemos usado “**Visual studio code**”. Para la creación de los diagramas tanto de secuencia como los de clases se ha utilizado la herramienta “**Visual Paradigm**”. Para poder tener todos el código actualizado y poder ejecutar y programar hemos usado “**Github**” , esta herramienta ha sido de gran uso ya que nos ha permitido tener todos los integrantes del equipo no solo el código actualizado sino que también los diagramas de clases y secuencia y la documentación.

Además de estas herramientas también hemos necesitado información para poder implementar correctamente el patrón MVC, los patrones de diseño, java 8… y la información la hemos sacado de los pdfs facilitados por el profesor en la plataforma “**eGela**”.

DISEÑO

Diagrama de clases final:

https://drive.google.com/file/d/1ldSL1_jMOU66xzFNyymHSzP6MT6wMV9H/view?usp=sharing

Diagrama de secuencia (disparar jugador):

https://drive.google.com/file/d/1xGN0JtOoO3iz-2RqqneZmOWAH4_nBpAn/view?usp=sharing

Diagrama de secuencia (consultar radar ordenador):

https://drive.google.com/file/d/1TtrNCH8c89LdTszTQBRWLLEMhuEHfNsZ/view?usp=sharing

DESARROLLO

Como hemos mencionado anteriormente, al haber utilizado la metodología SCRUM hemos trabajado en una secuencia de 3 sprints. En cada uno, hemos tenido que diseñar, implementar y probar nuestro proyecto, además de eso también hemos tenido reuniones con el cliente en los que nos ha pedido cambios en el trabajo hecho o nos ha informado de disconformidades tanto con el diseño como con la implementación que explicaremos a continuación.

En la reunión del primer sprint teníamos como requisitos a mostrar el diagrama de clases principal, el diagrama de secuencia (disparar jugador), un informe sobre el uso del patrón MVC, la planificación junto con el reparto de tareas y una demo funcional en la que se pueda jugar de manera simple, es decir solo lanzando bombas. Una de las observaciones del cliente fue que nuestro diagrama de clases era confuso ya que no distinguiamos una clase para el usuario y otro para la IA[1] y juntabamos todas las funciones en una sola clase llamada “Tablero”, eso hacía que esa clase tuviera más funciones de las que debía. Al finalizar este primer sprint concluimos con que a pesar de que todo funcionaba correctamente, de cara a posibles nuevos usuarios y funcionalidades, debíamos separar las clases de usuario e IA.

En la reunión del segundo sprint teníamos que mostrar más o menos lo mismo que en el anterior sprint pero con los nuevos requisitos que se nos pedían para esta etapa y con las observaciones y cambios que el cliente solicitó. En esta reunión con el cliente tuvimos muchas observaciones por su parte. Para empezar, al mostrar la demo el cliente nos dejó claro que el radar no estaba bien implementado ya que no habíamos leído bien el funcionamiento de esa herramienta. Por otro lado, también nos comentó que la relación del jugador y la IA seguía sin estar del todo clara por lo que teníamos que corregirlo y en el diagrama de clases nos recomendó que la clase radar no saliese de nuestra “fábrica de armas” ya que no es un arma y así habría menos confusión, para acabar con el diagrama de clases nos comentó que la dependencia entre el dinero que tiene el jugador y la tienda debía ser más clara. Respecto al diagrama de secuencia, en este caso el de consulta radar de la IA, nos explicó que en vez de for each era conveniente usar iteradores.

Finalmente en el tercer y último sprint al haber ido haciendo día a día el proyecto, la carga de trabajo ha sido algo menor. Se han hecho las correcciones que se nos pidieron en el anterior sprint pero la implementación del proyecto estaba completa.

[1] IA: Inteligencia Artificial, en este caso el rol de ordenador (enemigo)

Documentación de tareas

Sprint	Tarea	Responsables	Tiempo planificado	Tiempo real	Comentarios
1	Creación de código	Xabier y Javier	+8h	12h	Horas extra por avanzar a los siguientes sprints y añadir extras
1	Diseño de diagrama de clases	Diego y Estefanía	+3h	4h	Actualizaciones requeridas en el diseño
1	Diseño de diagrama de secuencia	Francisco y Asier	+2h	3h	Confusión y dudas a la hora de diseñar el diagrama
1	Documento informe MVC	Xabier	30’	20’	Explicación del modelo
1	Documento reparto de tareas	Asier	30’	40’	Falta de comprensión de los documentos a entregar
2	Creación de código	Xabier y Javier	+5h	7h	Errores y dificultades surgidas de nuevas implementaciones que pedía el cliente
2	Diseño de diagrama de clases	Diego	2h	1h	Ligeros cambios en las clases por petición del cliente y mejora de aspecto.
2	Diseño de diagrama de secuencia	Francisco y Estefania	+1h	1h 30’	Se tuvo que rehacer parte del diagrama debido a un fallo de github
2	Documento reparto de tareas	Asier	30’	25’	Mayor claridad sobre cómo organizar la documentación

3	Solución de código	Xabier y Francisco	4h	4h 30’	No demasiadas complicaciones para solucionar los cambios sugeridos por el cliente
3	Rediseño del diagrama de clases	Xabier	1h	1h	Ligeros cambios
3	Diagramas de secuencia solucionados	Diego y Javier	2h	4h	Tuvimos que rehacer ambos diagramas ya que los cambios del cliente lo requerían
3	Documentación	Asier y Estefania	2h	5h	Pocos problemas ya que estaba todo bien documentado

Documentación de pruebas

Prueba id	**Objetivo**	**Input**	**Condiciones de ejecución**	**Resultado esperado y obtenido**	**Comentarios**
0	Iniciar la pantalla principal	Nada	Tablero iniciado	Pantalla con el tablero	Sin problemas
1	Colocar barcos en posición	Tipo de barco, posición y dirección	Tablero iniciado	Colocación correcta/ colocación correcta	Sin problemas
2	Prueba de comando “random”	“random”	Tablero iniciado	Todos los barcos colocados aleatoriamente	Sin problemas
3	Prueba de comando “see_all”	“see_all”	Tablero con los barcos	Muestra todos los barcos del tablero enemigo	Sin problemas
4	Comprar Armas	Arma a elegir	Dinero suficiente y Arma	Compra completada	Sin problemas

			disponible		
5	Prueba de comando “idkfa”	“idkfa”	Tablero iniciado	Armas infinitas	Al principio no conseguiamos que nos diese solo armas infinitas
6	Prueba de comando “motherlode”	“motherlode”	Tablero iniciado	50.000 monedas	Sin problemas
7	Dañar Barcos no tocados	Posición en tablero rival	Tablero iniciado	Si existe barco en posición tocado o hundido/ Si no existe agua.	Sin problemas
8	Dañar Barcos tocados	Posición en tablero rival	Tablero iniciado y barco tocado o hundido	Sin resultado.	Sin problemas
9	Dañar barcos con escudo	Posición en tablero rival	Tablero iniciado y barco con escudo	Se elimina el escudo	Sin problemas
10	Poner escudo a un barco	Posición con barco en tablero jugador	Tablero iniciado y Escudo disponible	Se activa un escudo en el barco elegido	Sin problemas
11	Reparar un barco	Posición con barco en tablero jugador que esté tocado o hundido	Tablero iniciado, barco tocado o hundido y Reparación disponible	Se repara el barco seleccionado completamente	Sin problemas
12	Terminar una partida ganando el jugador	Posición con barco en tablero IA	Tablero iniciado y sólo una casilla de barco sin tocar en el tablero de la IA	Tocado y hundido. Jugador gana la partida	Sin problemas
13	Terminar una	Nada	Tablero	Tocado y	Sin problemas

	partida ganando la IA		iniciado y sólo una casilla de barco sin tocar en el tablero del jugador	hundido. IA gana la partida	

CONCLUSIONES

En primer lugar, tras la realización del proyecto hemos aprendido cómo implementar el patrón MVC a los proyectos, para así tener un mejor diseño de ellos mismos y ayuda a tener una separación entre la lógica del juego y la visualización.

Por otro lado, hemos visto un caso práctico para los otros tipos de patrones de diseño que al igual que el anterior, nos permiten crear diseños más eficientes para la creación de diferentes clases (Factory, Singleton…).

Gracias al uso de la metodología Scrum hemos aprendido a cómo organizar de una manera diferente los trabajos en equipo para así completar cada sprint con el cliente y no retrasar ni dejar atrás ninguna de las tareas.

También nos hemos dado cuenta de que cada comentario del cliente nos facilitaba después el entendimiento e implementación del proyecto ya que aunque nos provocaba realizar cambios, eran cambios necesarios. Por ejemplo el tema de la separación de clases de la IA y el usuario, nos dimos cuenta de que finalmente era necesario ya que por ejemplo si en cambios futuros se quiere añadir algún jugador o implementar una función multijugador es la única manera de hacerlo.

Finalmente, con todo esto hemos reflexionado acerca de cómo debemos plantear un proyecto ya que inicialmente solo teníamos una visión concreta sin ver a largo plazo los cambios que se podrían hacer y sin tener en cuenta si con nuestra implementación sería más o menos complejo de añadir. Hemos aprendido a tener una visión más general y nos ha sido de gran ayuda el entendimiento de patrones porque son sobre todo los que nos han permitido tener esta visión.