Facultad

de

Ciencias

Computer Science Wars: Obteniendo conocimientos mediante un juego de estrategia.

(Computer Science Wars:

Getting knowledge through a Strategy Game.)

Trabajo de Fin de Grado

para acceder al

GRADO EN INGENIERÍA INFORMÁTICA

Autor: Oscar López de la Jara

Director: Carlos Blanco Bueno

Noviembre – 2019

CONTENIDO

ILUSTRACIONES

TABLAS

ABSTRACT

A technique widely used in teaching to assess theoretical knowledge, is the use of test. For example: certification exams, driving licenses, etc. However, the study process to overcome such test is usually very tedious as it is based on the realization of test sets. On the other hand, lately techniques of the videogames world are being successfully implemented in other fields such as education (Serious Games). This way you try to get more entertaining and funnier this learning process.

This project's main objective is to create a turn-based strategy videogame as a funny and dynamic learning platform. To win, the player must demonstrate their knowledge in the field of computing. However, the project is easily expandable to other areas of knowledge and platforms. The game also includes leaderboards, test compatibility already defined in Moodle, etc.

Keywords: Serious Game, Videogame, Turn-Base Strategy, Unity 3D, Test, Multiplatform.

RESUMEN

Una técnica muy utilizada en docencia para evaluar los conocimientos teóricos, es el uso de test. Por ejemplo: exámenes de certificación, licencias de conducir, etc. Sin embargo, el proceso de estudio para superar pruebas de este tipo suele ser muy tedioso ya que se basa en la realización de conjuntos de test. Por otro lado, últimamente se están aplicando con éxito las técnicas del mundo de los videojuegos en otros campos como por ejemplo el de la educación (Serious Games). De esta forma se intenta conseguir hacer más ameno y divertido este proceso de aprendizaje.

Este proyecto tiene como objetivo principal crear un videojuego de estrategia por turnos que sirva como plataforma de aprendizaje divertida y dinámica. Para ganar, el jugador deberá demostrar sus conocimientos en el ámbito de la informática. No obstante, el proyecto es fácilmente ampliable a otros ámbitos de conocimiento. El juego además incluye tablas de clasificaciones, compatibilidad con test ya definidos en Moodle, versiones para escritorio y dispositivos móviles, etc.

Palabras Clave: Juego Serio, Videojuego, Estrategia por Turnos, Unity 3D, Test, Multiplataforma.

INTRODUCCIÓN

Pese a que existen una infinidad de definiciones de videojuego, lo que está claro es que en sus inicios fueron creados como método de entretenimiento. Esto ha continuado siendo así durante muchos años. No obstante, últimamente se han empezado a usar los videojuegos no solo como entretenimiento, sino como una herramienta educativa que es capaz de divertir, al mismo tiempo que enseña. Estos se conocen como Juegos Serios.

En palabras de Michael Schrage, investigador del MIT y autor del libro Juego Serio (Serious Play), Juego Serio es "cualquier herramienta, tecnología, técnica o juguete que permita a las personas mejorar la forma en que juegan en serio con la incertidumbre y que garantice el aumento de calidad de la innovación". [4]

Los Juegos Serios [5] cubren una amplia variedad de objetivos, por lo que se pueden clasificar según su finalidad:

- Advergame: Son utilizados por las empresas para dar a conocer su marca o sus productos.
- Edugame: Pueden darse dos casos, que el juego se centre en transmitir conocimientos a la persona que juega o que sirva para evaluar los conocimientos que el jugador ya tiene.
- Trainingame: Al contrario que el tipo anterior, este no transmite conocimientos, sino que se encarga de enfrentar al jugador a diferentes situaciones de forma repetida con la intención de que, en un futuro, este sea capaz de resolver dichas situaciones de una forma determinada.

Por otro lado, una forma típica de adquirir conocimientos en el ámbito educativo es mediante la realización de test. Moodle es una herramienta muy común a través de la cual estos se pueden crear ya sea para practicar o para poder evaluar el conocimiento en una materia.

Estos test dan como resultado una puntuación medible que ayuda a ser conscientes de los conocimientos que se poseen, permitiendo saber si se necesita seguir practicando o si los conocimientos son los esperados. No obstante, la realización de test suele ser ardua y aburrida.

Este proyecto se encuentra englobado en la categoría Edugame, con el objetivo de transmitir los conocimientos a través de test que son respondidos por el jugador. De esta forma, estos se pueden realizar de forma más amena. Además, la puntuación resultante se almacenará en un servidor externo, en donde se mantendrá una clasificación con los mejores resultados, de forma que sirva de motivación para los jugadores.

MOTIVACIÓN Y OBJETIVOS

Actualmente en el ámbito de la enseñanza, se utilizan a menudo los test como una forma de obtener conocimientos. Esto resulta aburrido y repetitivo, ya que para aprender mediante este método se requiere repetir las mismas preguntas múltiples veces.

Por lo tanto, el objetivo principal planteado en este proyecto es la creación de dos aplicaciones. La primera de las mismas proporcionará una plataforma en la que se puedan realizar dichos test de una forma más amena y divertida. Mientras que la segunda almacenará las puntuaciones obtenidas y creará una clasificación en un servidor externo que motive a los jugadores a intentar superarse.

Para lograrlo, también se han plantean los siguientes objetivos secundarios:

- Los test deben ser compatibles con los ya existentes en la plataforma Moodle.
- Debido a la proliferación de los teléfonos móviles y las tabletas, el videojuego debe ser compatible con Android.
- Además, como objetivo opcional se propone crear una tabla de puntuaciones.

PROCEDIMIENTO

Para la realización de este proyecto se ha elegido una metodología incremental. Esta metodología se basa en la iteración de todas las fases del proyecto, es decir, como se puede ver en la imagen siguiente [Ilustración 1], después de la fase de pruebas se vuelve a la de Análisis, volviendo a empezar un nuevo ciclo.

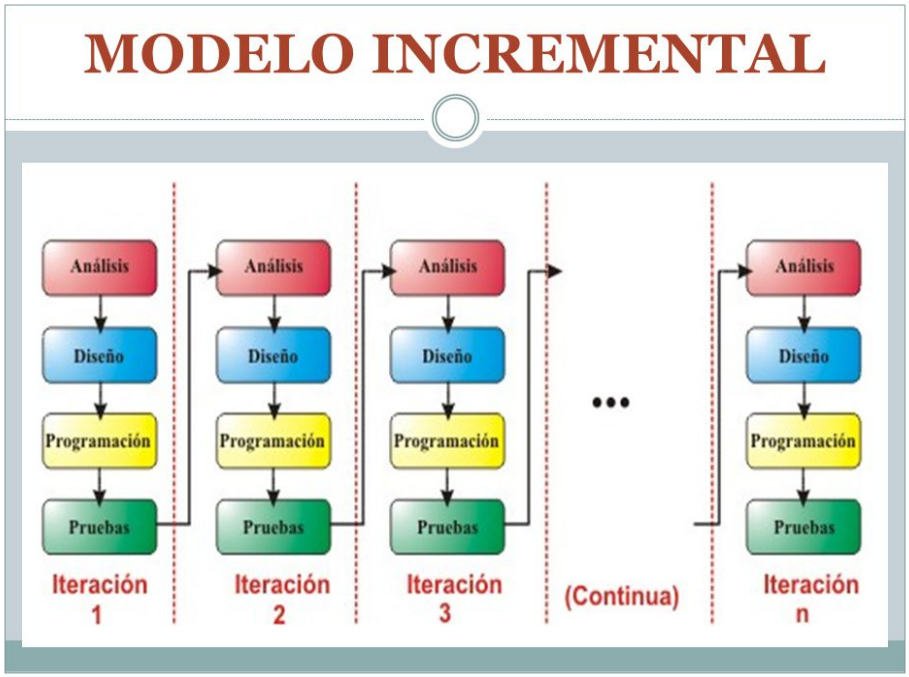


Ilustración 1 - Modelo Incremental [1]

Se eligió esta metodología ya que el desarrollo de un videojuego no tiene por qué tener un límite. O lo que es lo mismo, siempre se pueden añadir nuevas características al videojuego. Como consecuencia de esto, la aplicación que gestiona las puntuaciones también podría ser modificada indefinidamente. Esta metodología permite desarrollar todo lo que sea posible en los meses que dura el proyecto, e incluso, poder seguir desarrollando una vez que el proyecto termine sin establecer un límite.

TECNOLOGÍA

En este capítulo se muestran todas las herramientas que han sido utilizadas durante el desarrollo del videojuego y todos los lenguajes de programación usados. También se incluyen los diferentes frameworks utilizados.

Unity 3D

Para crear el juego se utilizó el motor gráfico Unity 3D. Este software dispone de una versión gratuita, que es la que se ha utilizado, y de diferentes versiones de pago. Además, las diferentes versiones dan acceso a herramientas adicionales. No obstante, hay un límite de ingresos que puedes percibir en función de la versión que elijas, obligándote a cambiarla si superas dicho límite. Este cambio suele implicar el pago de una cuota cada vez mayor.

Se aplica una capacidad de ingresos anuales o de fondos recaudados de $100 mil por ejercicio fiscal a los clientes de la categoría Personal. Se aplica una capacidad de ingresos anuales o de fondos recaudados de $200 mil por ejercicio fiscal para los clientes de la categoría Plus. Los clientes de las categorías Pro y Enterprise tienen capacidades de ingresos ilimitadas. [2]

Unity 3D soporta C# y javascript, pero existe mucha más información respecto a C# en Unity 3D por lo que fue el lenguaje elegido para llevar a cabo este proyecto.

Además, esta herramienta permite compilar la aplicación, realizando algunos ajustes, para una gran variedad de plataformas. Logra esto gracias a que en el caso de javascript, este puede ser ejecutado independientemente del Sistema Operativo, y en el caso de C#, puede funcionar usando el software Mono. Esto también implica que los editores que Unity 3D instala por defecto son MonoDeveloper o Visual Studio.

Visual Studio

Unity 3D es compatible con 2 IDEs los cuales incluye por defecto, Visual Studio y MonoDevelop. En el transcurso de este proyecto se ha utilizado exclusivamente Visual Studio. En concreto se utilizó Microsoft Visual Studio Community 2015.

Este es uno de los IDEs mas utilizados en el desarrollo de aplicaciones en C# ya que proporciona de forma gratuita un entorno con muchas herramientas y ayudas para el desarrollador. Como su propio nombre completo indica, esta desarrollado por Microsoft.

IntelliJ

Este es un IDE que permite entre otras cosas construir aplicaciones utilizando el lenguaje de programación Java. Se encuentran disponibles una versión gratuita llamada Community (la utilizada en este proyecto) y una versión de pago llamada Ultimate. En este caso, como veremos más adelante, se utilizo para construir un servicio con una Api REST con el objetivo de gestionar las puntuaciones obtenidas en el videojuego.

Gimp

Para la creación de algunos elementos de la interfaz de usuario fue necesario un programa que permita la edición de imágenes. Se opto por usar Gimp ya que es gratuito y cumple con los requisitos mínimos necesarios para este proyecto.

Inkscape

Inkscape es una herramienta de manejo sencillo, pero que a la vez proporciona potentes herramientas de dibujo vectorial. Esto permite que una persona con conocimientos sobre diseño gráfico pueda crear esplendidos diseños con ella, y a la vez, que una persona novata en estos temas pueda realizar pequeños diseños de forma sencilla. En este caso ha sido utilizada para crear imágenes sencillas de hexágonos que se utilizan durante el juego.

3D Max

3D Max es un software de renderización, animación y modelado 3D desarrollado por Autodesk. Utilizando este software se modelaron algunos assets con formas básicas. Y aunque es una aplicación compleja, fue necesaria su utilización para poder crear assets en 3D.

Git

Git es un software de control de versiones que permite almacenar el estado actual de los ficheros del proyecto y volver a ese estado más adelante si fuera necesario. Incluso existen páginas web donde se ofertan repositorios en línea, de forma que puedas enviar toda la información de tu repositorio local a uno en la nube.

Version control is a system that record changes to a file or set of files over time so that you can recall specific versions later. [9]

Durante el presente proyecto se ha utilizado este software, aunque no se han utilizado algunas funcionalidades que este ofrece, como utilización de ramas, etc.

Android SDK

Aunque Unity 3D es capaz de compilar los videojuegos para la plataforma Android, necesita tener disponible el SDK apropiado. Para esto se instaló el SDK Manager y todas las herramientas necesarias de desarrollo para compilar, depurar y ejecutar el proyecto para Android.

C#

C# es un lenguaje de programación que se ha diseñado para compilar diversas aplicaciones que se ejecutan en .NET Framework. C# es simple, eficaz, con seguridad de tipos y orientado a objetos. Las numerosas innovaciones de C# permiten desarrollar aplicaciones rápidamente y mantener la expresividad y elegancia de los lenguajes de estilo de C. [6]

C# es uno de los lenguajes más utilizados en Unity 3D y, por tanto, es del que más documentación se puede encontrar. Por este motivo se eligió este como lenguaje de programación para el videojuego.

Cabe remarcar que no todas las funcionalidades del lenguaje pueden ser utilizadas ya que para que funcione correctamente en algunos sistemas operativos, es necesario utilizar una versión antigua del lenguaje. Este es el caso por ejemplo de las funciones lambda.

JAVA

Este es un lenguaje de programación multiplataforma ampliamente extendido. En este proyecto, dicho lenguaje ha sido utilizado junto con algunos frameworks como son Spring Boot, Flyway o Swagger, para almacenar de forma remota las puntuaciones obtenidas por los distintos jugadores.

XML

Es un lenguaje de marcas bastante extendido, que permite crear tu propia estructura interna en los archivos. En este proyecto se ha utilizado para almacenar la información de las preguntas. No ha sido necesaria la realización de un archivo DTD o XSD, ya que el archivo XML utilizado ha sido creado por la plataforma Moodle. Por este motivo se da por hecho que la estructura interna del archivo es correcta.

CSV

Es un tipo de archivo que contiene datos separados por un carácter específico. En este caso se utilizó la "," (coma) para separar los diferentes datos. Este tipo de archivo se utilizó para definir el tablero de juego, tal y como veremos en el apartado Mapa.

Frameworks

Para la realización de este proyecto también se han utilizado varios frameworks:

- Spring Boot: Este es un framwork muy conocido y extendido para la construcción de aplicaciones de una forma rápida y sencilla.
- Flyway: Se encarga de gestionar el esquema de base de datos de forma que se pueda crear o modificar su estructura a través de parches. También permite inicializar la base de datos con determinada información, lo que es muy conveniente durante las pruebas.
- MyBatis: Permite ejecutar sentencias SQL en una base de datos de forma sencilla y segura.

- Swagger: Es un framwork que puede ser utilizado entre otras cosas para crear documentación interactiva de Apis REST de forma fácil y rápida.

ASSETS

Debido a la naturaleza del proyecto, la parte más artística del mismo no resulta tan relevante. Esto ha llevado a tomar decisiones como utilizar assets bajo licencia CC (Creative Commons) y crear assets sencillos utilizando programas gratuitos o con Licencias de Prueba.

Los assets gráficos creados son los siguientes:

- Hexágonos en 3 dimensiones: Creados con 3D Max.
- Imágenes de hexágonos: Todas las imágenes con formas hexagonales han sido creadas con Inkscape.
- Imágenes de UI: Han sido creadas con el programa Gimp.
- Vehículos y Bases: Todos estos assets se encuentran bajo la licencia EULA de Unity 3D. El creador de dichos assets es “VD GAMES” y a continuación se muestra su página web:

  https://vd_game_studio.artstation.com/

ANÁLISIS DE REQUISITOS

En este apartado se muestran de forma detallada todos los requisitos capturados durante las distintas fases de análisis de requisitos llevadas a cabo. Estos requisitos responden a diferentes necesidades del proyecto, como por ejemplo, que el juego sea capaz de ejecutarse en dispositivos móviles y tabletas. Estos requisitos se encuentran divididos entre funcionales y no funcionales.

Los requisitos funcionales son aquellos que definen las acciones que las aplicaciones deben hacer. Los requisitos funcionales recabados son los siguientes:

ID	Descripción
F001	Al principio de cada turno el jugador responderá a una pregunta. Si la respuesta es correcta, recibirá algún tipo de recurso. Además, esto será tenido en cuenta para calcular la puntuación obtenida por el jugador.
F002	El jugador podrá elegir cada turno si quiere que la pregunta sea de dificultad Difícil, Media o Fácil. Además, esto deberá verse reflejado en los recursos que reciba en caso de acertarla y en la puntuación obtenida.
F003	Ya que se desconoce cuántos turnos durará la partida, después de responder a la última pregunta del test, se deberá volver a empezar por la primera pregunta, de forma que el jugador pueda seguir jugando.
F004	El jugador podrá comprar unidades con los recursos ganados.
F005	Las unidades dispondrán de los siguientes atributos: Vida, Ataque y Movimiento.
F006	Existirá una Inteligencia Artificial la cual intentará vencer al jugador.
F007	El jugador gana la partida cuando conquista la base o edificio de la IA.
F008	El jugador pierde la partida cuando la IA conquista su base o edificio.
F009	El jugador podrá seleccionar su edificio y ejércitos para ver que unidades hay dentro.
F010	Cuando el jugador tenga seleccionado un ejército propio, se mostrarán las

	casillas a las que puede mover lo.
F011	Al final de la partida el jugador podrá visualizar la puntuación que ha obtenido.
F012	La aplicación de puntuaciones creará el esquema de la base de datos de forma automática durante el primer arranque de la aplicación.
F013	El acceso a los métodos de la aplicación de puntuaciones estará protegido por usuario y contraseña.

Tabla 1 - Requisitos Funcionales

Los requisitos no funcionales son aquellos que no indican las funcionalidades que debe tener la aplicación, sino como debe ser implementada.

ID	Descripción
NF001	El juego deberá ser capaz de ejecutarse al menos en Windows y Android.
NF002	Las preguntas deberán estar almacenadas en archivos xml, que sean compatibles con Moodle. Es decir, cualquier test exportado de Moodle deberá ser compatible con el juego.
NF003	El mapa deberá almacenarse en algún tipo de archivo de datos (xml, csv, …).
NF004	Las puntuaciones se almacenarán en un servicio externo al juego.
NF005	Para que pueda ser ejecutado correctamente en dispositivos Android, el consumo de memoria del videojuego debe ser menor de 2GB de memoria RAM.

Tabla 2 - Requisitos No Funcionales

ARQUITECTURA

Este proyecto está compuesto por dos aplicaciones cuya arquitectura es completamente diferente. Como veremos a continuación, mientras que el videojuego está dividido de forma funcional en componentes, la aplicación para gestionar las puntuaciones tiene un diseño en tres capas. A lo largo de este documento se encuentran varios diagramas UML que ilustran esto.

The Unified Modeling Language (UML) is a family of graphical notations, backed by single meta-model, that help in describing and designing software systems, particularly software systems built using the object oriented (OO) style. That's somewhat simplified definition. In fact, the UML is a few different things to different people. [13]

Arquitectura del Videojuego

Como ya hemos adelantado, el videojuego ha sido diseñado dividiéndolo de forma funcional en componentes tal y como se puede comprobar en el siguiente diagrama [Ilustración 2]. Dichos componentes tienen su propio diseño que veremos más adelante, compuesto por diferentes clases.

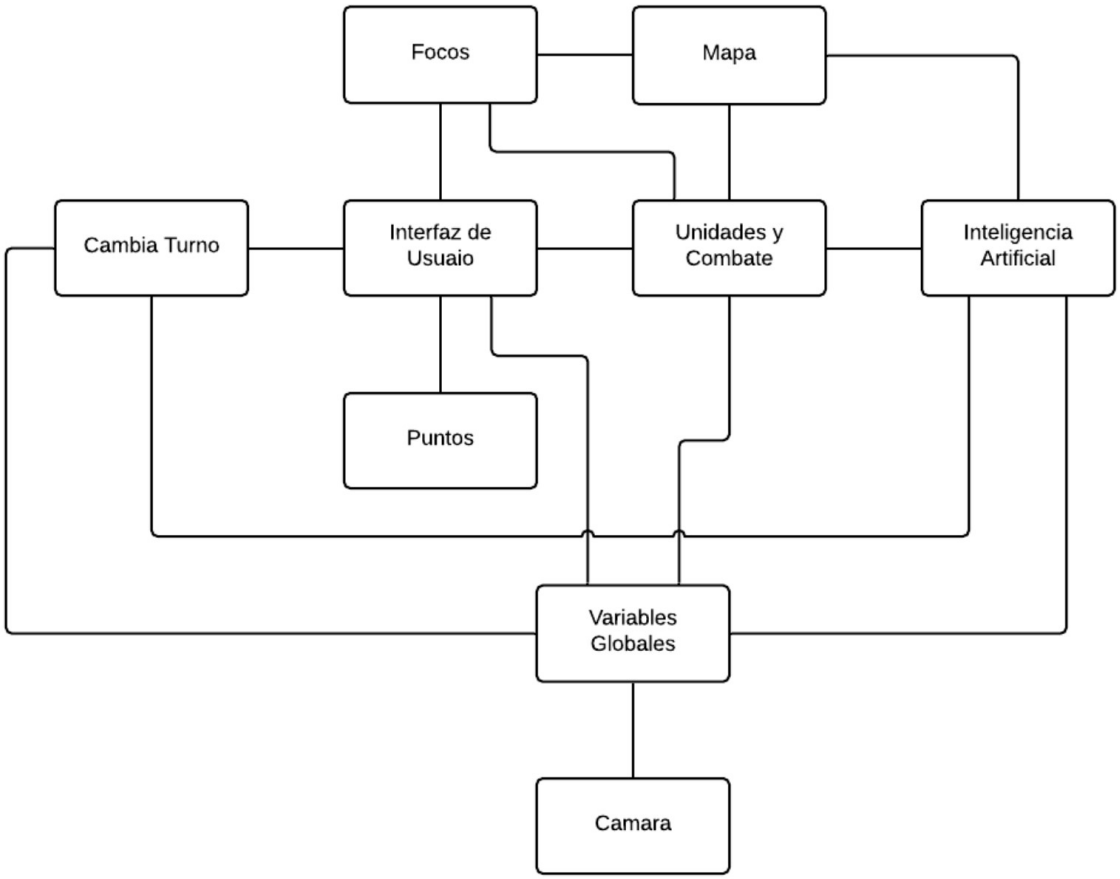


Ilustración 2 - Arquitectura del Videojuego

Estos componentes son:

- Mapa: Gestiona todo lo relacionado con el mapa de juego. Esto incluye el movimiento de las unidades del jugador. No obstante, las unidades de la IA son movidas desde el componente IA, aunque este hace uso en ciertas ocasiones de este componente. *
- Puntos: En este componente, como su nombre indica, se gestionan los puntos conseguidos por el jugador y su almacenamiento en la otra aplicación comprendida en este proyecto.
- Unidades y Combate: Una parte fundamental del videojuego son las unidades, las cuales poseen diferentes características. Estas características son utilizadas para realizar combates que determinan el vencedor.
- Focos: En este componente se gestionan las acciones que realiza el usuario. Desde detectar los clics del ratón hasta mostrarle la información pertinente. *

- Interfaz de Usuario: Comprende todos los elementos de la interfaz de usuario. Todas las llamadas realizadas desde otro componente a un elemento de la IU, se hacen a través de la clase CanvasController. Además, también se gestionan las preguntas y los recursos.
- Inteligencia Artificial: En este componente se encuentra todo lo relacionado con la IA del juego. Desde la toma de decisiones hasta el almacenamiento de la información.
- Cambia Turno: Gestiona los cambios de un turno a otro.
- Cámara: Se encarga de mover la cámara, ya sea con el teclado o a través de una pantalla táctil.
- Variables Globales: Componente donde se almacena información para que otros la puedan utilizar. También forma parte de este componente un objeto que persiste entre pantalla y pantalla del videojuego, el cual contiene dicha información.

*Con el fin de simplificar la explicación del diseño, la parte referente al movimiento de las unidades se explica en una sección llamada Movimiento, en vez de en la sección Mapa (en el caso de las unidades del jugador) y en la sección Inteligencia Artificial (en el caso de las unidades de la IA.)

Arquitectura de la Aplicación de Puntuaciones

Esta aplicación ha sido diseñada siguiendo una arquitectura en tres capas las cuales se definen según su función. No obstante, como se puede ver en el siguiente diagrama [Ilustración 3], la aplicación contiene además otras clases utilizadas para configurar la aplicación. Estas tres capas son las siguientes:

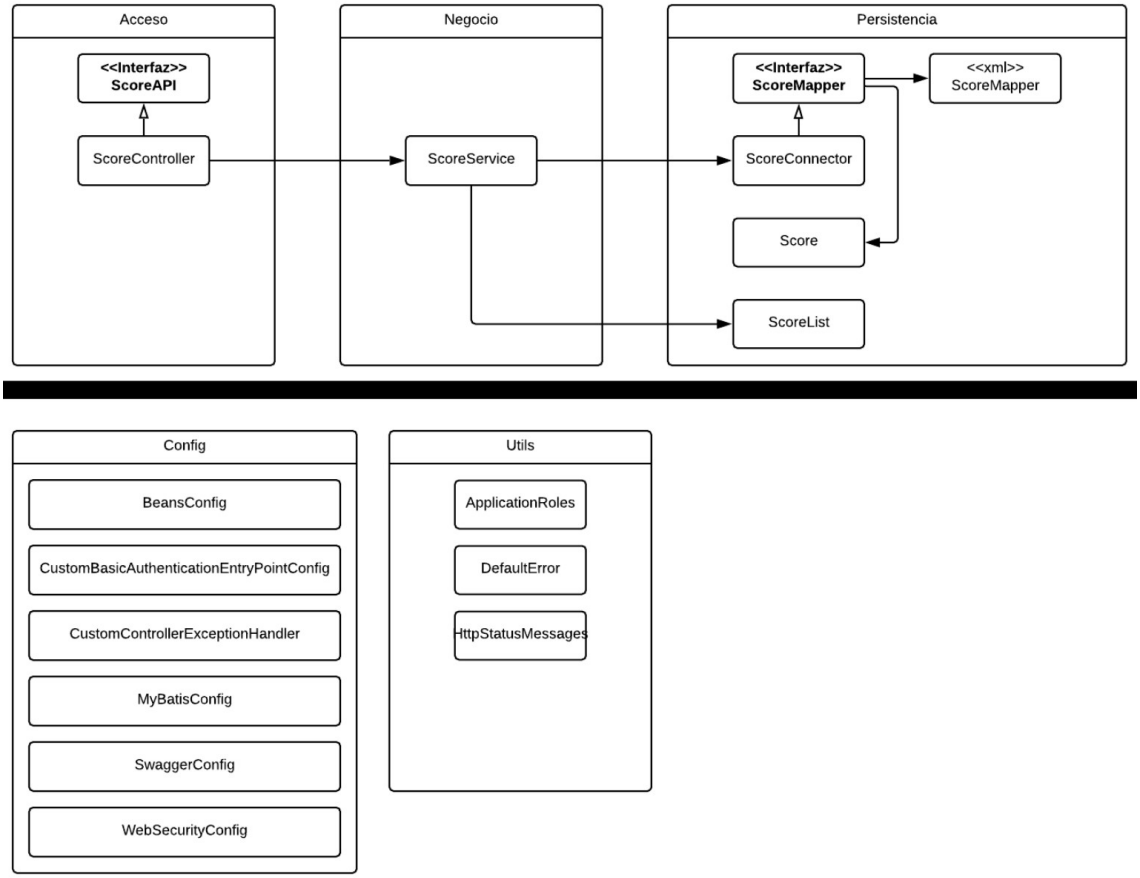


Ilustración 3 - Arquitectura de la Aplicación de Puntuaciones

- Acceso: La capa de acceso contiene todos los recursos de la Api REST de la aplicación, así como los modelos que estos retornan. Además, se han añadido interfaces para poder separar las notaciones de Swagger de los recursos de la Api.
- Negocio: En esta capa se encuentran las clases que contienen la lógica de la aplicación.
- Persistencia: Comprende todas las clases y ficheros que se utilizan desde la creación del esquema de la base de datos durante el primer arranque hasta las consultas SQL para almacenar o recuperar datos.

Cabe destacar que existen muchos otros archivos que no se encuentran dentro de ninguna de estas capas. Algunos de estos ficheros los veremos en detalle más adelante en la sección “Otros”.

DISEÑO E IMPLEMENTACIÓN

El Diseño de Software es la actividad del ciclo de vida del software en la cual se analizan los requisitos para producir una descripción de la estructura interna del software que sirva de base para su construcción. [7]

En esta sección vamos a ver más en detalle cómo están diseñadas e implementadas ambas aplicaciones.

Diseño e Implementación del Videojuego

El videojuego dispone de 9 componentes diferentes diferenciados de forma funcional. A continuación vamos a verlos en detalle de uno en uno. Por último, en este capítulo se muestra una sección que explica las modificaciones necesarias para lograr que este videojuego pueda ejecutarse bajo el sistema operativo Android.

Mapa

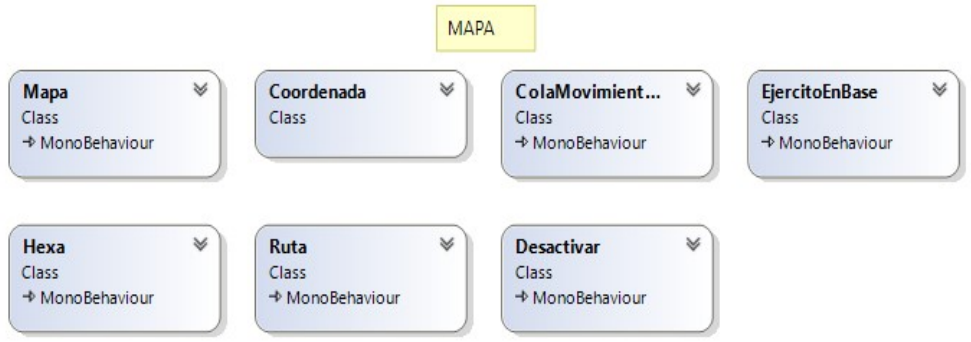


Ilustración 4 - Componente Mapa

Toda la gestión del mapa es llevada a cabo por un objeto llamado Mapa. Para realizar estas funciones dispone de un componente de tipo Script llamado también Mapa. Este componente sobrescribe el método Awake de la clase MonoBehavior para que el mapa este creado antes de que otros objetos puedan necesitarlo.

Con la intención de que el mapa sea fácilmente modificable, este se genera a partir de un archivo CSV. Tal y como se puede ver en el ejemplo que se encuentra a continuación, cada tipo de casilla tiene un número determinado.

```
18, 9
20,20,20,20,20,20,20,20,20
20,10,10,10,10,10,10,10,20
20,10,10,10,1,10,10,10,20
20,10,10,10,10,10,10,10,20
20,10,20,20,20,20,20,10,20
20,10,10,10,10,10,10,10,20
20,10,10,10,10,10,10,10,20
20,10,10,10,10,10,10,10,20
20,20,20,20,10,20,20,20,20
20,10,10,10,10,10,10,10,20
20,10,10,10,10,10,10,10,20
20,10,10,10,10,10,10,10,20
20,10,10,10,10,10,10,10,20
20,10,20,20,20,20,20,10,20
20,10,10,10,10,10,10,10,20
20,10,10,10,7,10,10,10,20
20,10,10,10,10,10,10,10,20
20,20,20,20,20,20,20,20,20
```

Ilustración 5 - Mapa CSV

Como se puede ver en el ejemplo adyacente [Ilustración 5], los 2 primeros números indican el largo y el ancho del mapa respectivamente. A continuación, el archivo contiene un número de líneas igual al largo del mapa, y en cada una de estas líneas tantos números como casillas de ancho sea el mapa.

Con la intención de facilitar la introducción de nuevos tipos de casillas, los números de estas no son continuos de forma que se puedan introducir en números intermedios para su mejor organización.

Como es lógico, cada número corresponde a un único objeto predefinido. Estos objetos predefinidos tienen los componentes necesarios para renderizar los modelos necesarios, más una serie de componentes extra de tipo Script para ejecutar la lógica del juego.

Los objetos predefinidos disponibles para componer el mapa son los siguientes:

- Hexágono Azul: Representa una zona de agua en la que las unidades no pueden estar ni cruzar, por lo que no tiene ningún componente añadido,
- Hexágono Verde: Representa una casilla en la que las unidades pueden estar y pueden pasar libremente. Tiene los componentes Ruta y Hexa.
- Hexágono Amarillo: Tiene el componente Ruta. No se utiliza de forma individual, sino que se utiliza para componer los siguientes objetos predefinidos:
    - HexagonoBaseX*: Representa una de las bases del jugador. En ella se gastan los recursos para producir unidades. Tiene los componentes EjercitoEnBase, ColaProduccion y Hexa.

- HexagonoEnemigaBaseX*: Representa una de las bases de la Inteligencia Artificial. En ellas la IA produce sus unidades. Tiene los componentes EjercitoEnBaseIA y Hexa.

* La X representa un número del 1 al 3, ya que por cada uno de estos objetos predefinidos hay 3 modelos diferentes con distintos colores.

Para poder crear una instancia de un nuevo objeto, es necesario entre otras cosas, especificar la ubicación en la que va a aparecer dicho objeto. Al ser hexágonos, es necesario calcular su "radio corto" [Ilustración 6] para conocer la distancia a la que se deben colocar unos de otros.

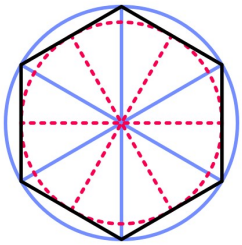


Ilustración 6 - Diagrama de Hexágono [3]

Esta distancia es importante ya que el centro de los hexágonos vecinos debe encontrarse al doble de esta distancia. Este radio se calcula multiplicando la longitud del "radio largo" por (√3) /2. [3]

Por último, después de crear sus instancias, se les da un nombre con el que se puedan identificar fácilmente tanto ellos, como su posición en el mapa. El formato de dicho nombre es "I_X_J_Y_NombreDelObjetoPredefinido" siendo X e Y las coordenadas dentro del mapa. Además, se les incluye como hijos del objeto Mapa.

Adicionalmente a lo explicado anteriormente, también se encarga de crear un minimapa que se encuentra en la parte inferior izquierda. Se genera de forma similar al mapa normal, pero con imágenes de hexágonos en vez de usar los modelos en 3 dimensiones, y adaptando su tamaño al tamaño del panel que los va a contener, tal y como se ve en la imagen siguiente [Ilustración 7]. Por último, se añaden como hijos del objeto minimapa.

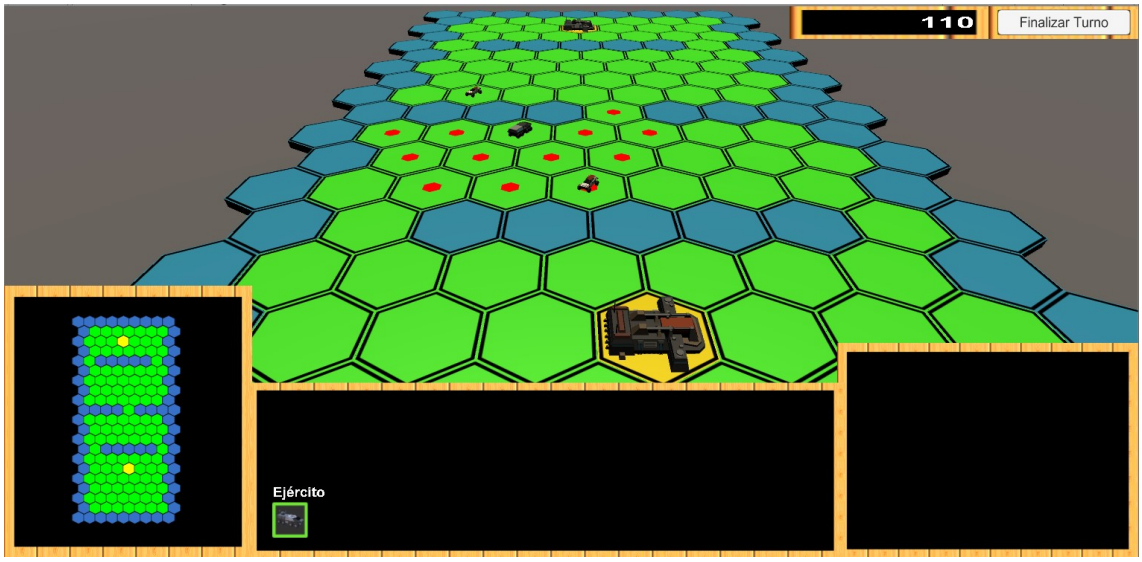


Ilustración 7 - Minimapa

PUNTOS

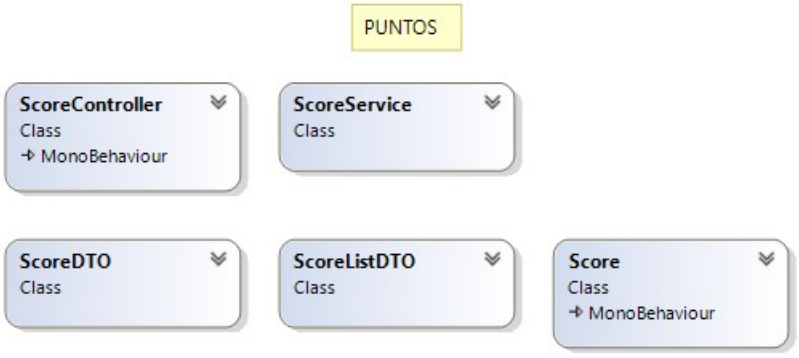


Ilustración 8 - Componente Puntos

Cada vez que el jugador responde a una pregunta de forma correcta, como se verá en el apartado de Interfaz de Usuario, este recibe una serie de Recursos y de Puntos.

Los puntos que el jugador consigue al responder una pregunta pueden ser tanto positivos como negativos, por lo que este podría llegar a terminar la partida con una puntuación negativa.

Una vez termine la partida, se cambiará de pantalla. En esta, la puntuación del jugador se envía a un servidor externo que la almacena en una base de datos. Esto se lleva a cabo mediante una aplicación que se describe más adelante.

Después de esto, se recuperan de dicho servidor la posición en el ranking de puntuaciones del resultado obtenido por el jugador, así como las mejores puntuaciones registradas.

UNIDADES Y COMBATE

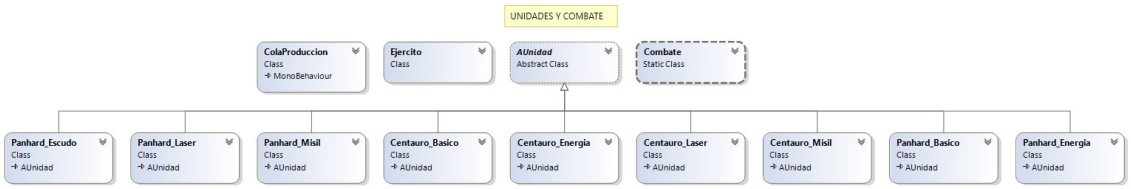


Ilustración 9 - Componente Unidades y Combate

Hay distintos tipos de unidades con diferentes valores en sus atributos. Para representar esto se ha utilizado un patrón Fabrica Abstracta, es decir, todos ellos implementan la clase abstracta AUnidad para asegurarse de que implementan todos los métodos necesarios y poder usar un mismo tipo de variables para todos. Los atributos según el tipo de unidad se almacenan como constantes en la clase y son los siguientes:

	Vida	Ataque	Movimiento	Coste
Panhard Básico	1	0	8	50
Panhard Escudo	200	0	7	100
Panhard Laser	50	20	7	60
Panhard Misil	50	30	7	70
Panhard Energía	50	40	7	80
Centauro Básico	1	0	5	50
Centauro Laser	75	30	5	110
Centauro Misil	75	40	5	120
Centauro Energía	75	50	5	130

Tabla 3 - Unidades

Estas unidades se agrupan para formar ejércitos, es decir los ejércitos están formados de unidades. Una sola unidad puede formar un ejército.

Existen 2 tipos de unidades que apenas tienen vida y que no tienen ataque. Estas unidades, las cuales conoceremos más adelante como “Exploradores”, tienen además un precio en recursos más barato que las demás. Esto se debe a que se pretende implementar en un futuro un sistema de “Niebla de Guerra” (Solo puedes ver lo que está dentro de la distancia de visión de tus unidades o bases), por lo que tener unidades repartidas por el mapa puede suponer una ventaja táctica.

Además, existe otra unidad llamada Panhard Escudo que tiene ataque 0. Sin embargo, tiene una gran cantidad de vida. Esto es porque, como veremos más adelante, en un combate, el objetivo de los ataques se selecciona de forma aleatoria entre las unidades que queden con vida dentro del ejército enemigo, por lo que puede ayudar a absorber daño de ataques enemigos.

El combate se realiza mediante la clase estática Combate, la cual retorna el ejército vencedor.

En un combate empieza atacando el defensor, es decir, el que ha sido atacado. Esto otorga ventaja al defensor. Además, el combate no termina hasta que uno de los 2 ejércitos es destruido por completo.

Los ataques producen un daño fijo según el tipo de unidad, pero la unidad enemiga que recibe el golpe se selecciona de forma aleatoria. Esto hace que al final del combate las unidades restantes del vencedor estén, en la mayoría de los casos, con menos vida de la que empezaron. Por este motivo después del combate se restaura la vida de todas las unidades supervivientes.

Lo que se pretende conseguir con estas reglas es beneficiar a los que reciben el ataque, de forma que el jugador busque obligar a la maquina a tacar y viceversa. Y beneficiar también a los grandes ejércitos frente a los pequeños, ya que cada unidad realiza su ataque, y por tanto, si hay una gran diferencia en torno al número de unidades el ejército grande casi no recibirá daño.

Por último, en esta categoría también se gestionan los ejércitos y su producción. Cuando desde un Foco, los cuales veremos más adelante, se selecciona un botón de creación de unidad, esta llama a CanvasController, y este tras comprobar que el jugador dispone de suficientes recursos, llama a este componente. Entonces se añade la unidad (de momento es solo una imagen) a la cola de producción. Cuando se produzca un cambio de turno, se pasará la primera imagen de la cola al ejército

que se encuentra en la base, creando uno si no lo hubiera. También se puede anular la creación de una unidad pulsando sobre su imagen.

FOCOS

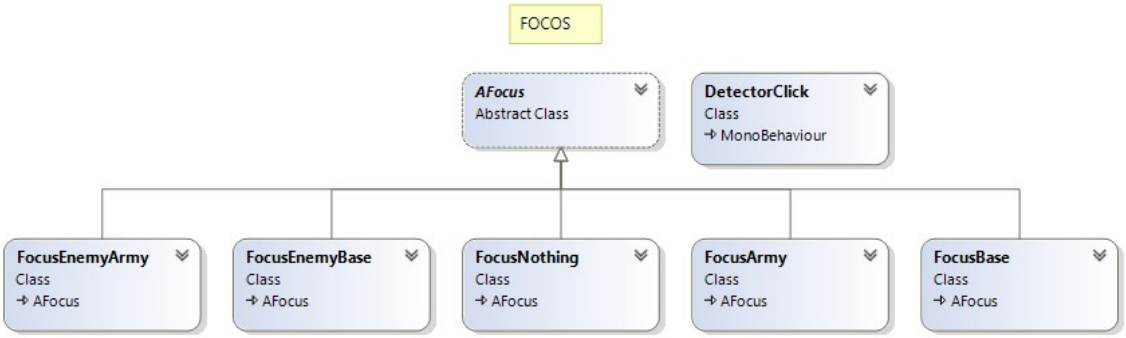


Ilustración 10 - Componente Focos

Durante el juego el jugador puede hacer clic sobre diferentes objetos (Bases, Ejércitos, etc.), a lo cual el juego debe reaccionar mostrando diferentes informaciones o incluso marcando algunas casillas del mapa.

Para lograr esto, se ha implementado una clase llamada DetectorClick la cual verifica si el jugador ha hecho clic sobre algo. En caso afirmativo lanza un Raycaster para averiguar sobre que objeto se hizo. Esto devuelve un objeto RaycastHit a través del cual podremos acceder al objeto clicado. Raycaster y RaycastHit son dos objetos disponibles en el motor de Unity 3D, que explicados de forma sencilla, se utilizan para calcular con que objetos chocaría un haz láser lanzado desde un punto en una dirección concreta.

Una vez hecho esto, se le pasa esta información a una máquina de estados. Esta máquina de estados está compuesta por DetectorClick, una clase abstracta AFocus y 5 clases más que implementan AFocus:

- DetectorClick: Además de todo lo explicado con anterioridad, esta clase también se encarga de alojar el estado actual de la máquina de estados.
- AFocus: Es una clase abstracta que sirve para verificar que todas las clases que heredan de ella implementan los métodos necesarios. Además, también se utiliza como un tipo genérico para poder almacenar las otras clases en una misma variable.

- FocusArmy, FocusBase, FocusEnemyArmy, FocusEnemyBase, FocusNothing: Como se ha explicado antes estas 5 clases implementan a AFocus y representan los estados de la máquina. Como el cambio de un estado a otro se hace con bastante frecuencia, se ha implementado un patrón Singleton [12] en cada una de ellas evitando así el andar creando y destruyendo estos objetos. A continuación, se muestra un resumen de la función de cada una de estas clases:

    - FocusArmy: Se activa cuando el jugador selecciona uno de sus ejércitos. Al activarse marca tantas casillas alrededor del ejército seleccionado como casillas de movimiento le queden a dicho ejército, y muestra en el panel del medio-inferior de la pantalla las unidades que lo componen. Posteriormente si se presiona el botón derecho sobre una de las casillas marcadas el ejército se moverá hasta ella.

    - FocusBase: Se activa cuando el jugador selecciona una de sus bases. Al activarse marca tantas casillas alrededor de la base seleccionada como casillas de movimiento le queden al ejército que se encuentre en su interior (en caso de haberlo), y muestra en el panel del medio-inferior de la pantalla tanto las unidades que se encuentran en la base como las que se encuentran en la cola de producción. Posteriormente si se hace clic con el botón derecho sobre una de las casillas marcadas, las unidades que se encuentren dentro de la base formarán un nuevo ejército y se moverán hasta dicha casilla.

    - FocusEnemyArmy: Se activa cuando el jugador selecciona uno de los ejércitos de la IA. Al activarse muestra en el panel del medio-inferior de la pantalla la cantidad unidades que componen el ejército, pero no el tipo de cada una de ellas.

    - FocusEnemyBase: Se activa cuando el jugador selecciona una de las bases de la IA. Al activarse muestra en el panel del medio-inferior de la pantalla el número de unidades que se encuentran en la base, pero no el tipo de cada una de ellas.

- FocusNothing: Se activa cuando se selecciona cualquier otra parte que no active uno de los estados anteriores y cuando se termina el turno. Este estado no muestra ningún tipo de información y es el estado por defecto.

INTERFAZ DE USUARIO

Ilustración 11 - Componente Interfaz de Usuario

La interfaz de usuario es un aspecto muy importante en el mundo de los videojuegos ya que es la principal fuente de información del jugador. Esta debe ser intuitiva y sencilla, mientras que proporciona todos aquellos datos necesarios para que el jugador sea consciente de lo que está ocurriendo en todo momento.

En este proyecto podemos encontrar 5 paneles:

- El Panel de Recursos: Se encuentra en la parte superior derecha de la pantalla. Muestra la cantidad de recursos disponibles que tiene el jugador para gastar en cada momento.
- El Panel de Fin del Turno: Se encuentra en la parte superior derecha de la pantalla, junto al Panel de Recursos. Muestra un botón que al pulsarlo finaliza el turno del jugador y comienza el de la IA.
- Panel del Minimapa: Se encuentra en la parte inferior izquierda de la pantalla. Muestra una recreación del mapa actual a escala.
- Panel de Estado: Se encuentra en la parte inferior central de la pantalla. Cada vez que el Foco se encuentra en un ejército o base, este panel muestra información al respecto.
    - En caso de ser un ejército del jugador, se muestran las unidades que lo componen.

- En caso de ser una base del jugador, se muestran las unidades como si se tratase de un ejército, pero además se muestran las unidades que están en la cola de producción. En este último caso, las imágenes que se muestran son una representación de las unidades las cuales tienen añadido un evento de tipo onClick. De forma que cuando se hace clic en una de ellas, esta es eliminada de la cola de producción.

- En caso de ser un ejército de la IA o una base de la IA, se muestra la cantidad de unidades que hay en ese ejército o base. Esto no se hace mostrando un número, sino que al igual que si fuese del jugador se muestra una imagen por unidad, pero en este caso, todas las imágenes son símbolos de interrogación, tal y como se ve en la siguiente imagen [Ilustración 12]. Esto permite al jugador conocer el número de unidades, pero no el tipo de las mismas.

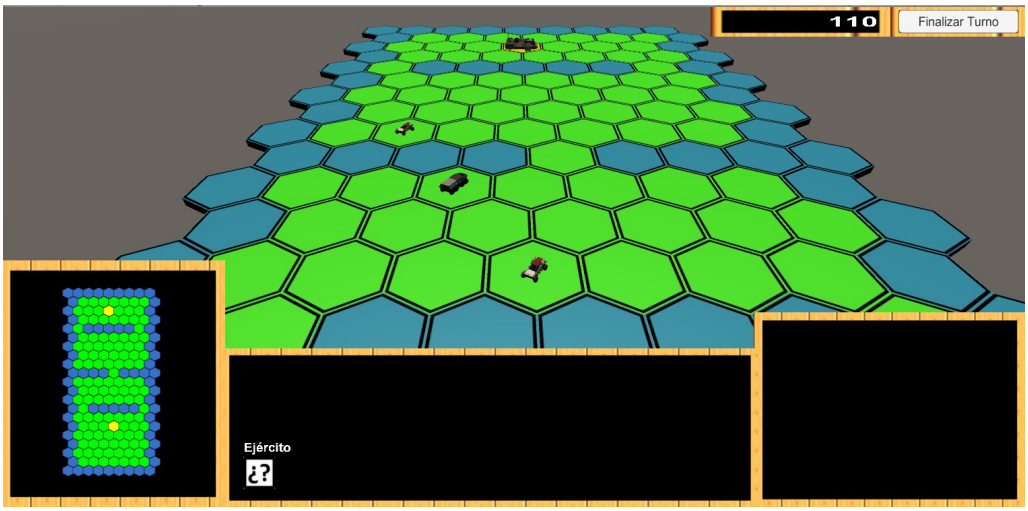


Ilustración 12 - Vista de Paneles

- Panel de Botones: Se muestra en la parte inferior derecha de la pantalla. Cada vez que el Foco se encuentra en una base del jugador, este panel muestra un botón con cada una de las unidades que el jugador puede crear.

Además de estos 5 paneles que se encuentran siempre visibles existen otros 2 que aparecen y desaparecen según el momento del turno en el que se encuentre el jugador:

- Panel de Dificultad: Se muestra en el centro de la pantalla al inicio del turno del jugador. Contiene 3 botones que indican la dificultad de la pregunta que

se lanzará a continuación. Desaparece al pulsar cualquiera de los 3 botones y provoca la aparición del panel que se explica a continuación.

- Panel de Preguntas: Se muestra en el centro de la pantalla. En él se pueden ver una pregunta y 4 botones con las posibles respuestas. Desaparece al pulsar cualquiera de las cuatro respuestas y activan uno de los dos mensajes que se explican a continuación, dependiendo de si era la respuesta correcta o no.

Aparte de los paneles existen 2 mensajes que también se muestran en pantalla en unos momentos determinados. Estos mensajes desaparecen pasado un segundo:

- Correcto!!!: Se muestra cuando el jugador ha pulsado la respuesta correcta en el Panel de Preguntas.
- Incorrecto!!!: Se muestra cuando el jugador ha pulsado una de las respuestas incorrectas en el Panel de Preguntas.

Por defecto Unity 3D crea un objeto llamado Canvas, en el que agrupa todos los componentes de IU [11]. Aunque este componente puede ser borrado, es recomendable no hacerlo ya que ayuda a organizar los demás objetos de tipo IU dentro de la pantalla.

Inicialmente, cada vez que se necesitaba realizar un cambio en un componente de la Interfaz de Usuario desde una parte del código, esta accedía directamente al objeto que necesitaba modificar. Esto origino una red de llamadas que hacía casi imposible conocer el estado de la IU en cada momento. Por este motivo se implementó el componente CanvasControler.

Este componente se añadió al objeto Canvas con la intención de ordenar las llamadas a la IU. De esta forma, este componente actúa como un proxy que permite la interacción entre los elementos de la IU y el resto de elementos. No obstante, algunos elementos de la IU se comunican entre ellos sin utilizar el componente CanvasControler.

Este componente también gestiona las preguntas. Las preguntas, los recursos y la puntuación son tres de las cosas más importantes de este proyecto, y además

están fuertemente ligadas, ya que como veremos a continuación dos de ellas dependen de la tercera.

Cada turno se le mostrará al jugador un panel con 3 dificultades. Una vez seleccione una, aparecerá una pregunta de la dificultad seleccionada y cuatro posibles respuestas que se repartirán de forma aleatoria entre cuatro botones. Solo una de las respuestas es correcta. Entonces el jugador deberá elegir una de estas respuestas. Si selecciona la verdadera, se le darán una cantidad de recursos y puntos que variará en función de la dificultad. En caso contrario, se le darán puntos negativos al jugador, el cual comenzará su turno con normalidad.

- Las preguntas Fáciles darán un total de 50 recursos y puntos.
- Las preguntas Medias darán un total de 100 recursos y puntos.
- Las preguntas Difíciles darán un total de 150 recursos y puntos.

Dichos recursos pueden gastarse para generar unidades mientras que la suma de todos los puntos se utilizará para generar una clasificación. Esto se detalla más profundamente en la sección “Unidades y Combate” y en “Puntos” respectivamente.

Las preguntas son leídas de tres archivos diferentes que tienen el mismo formato que los test creados a través de la plataforma Moodle. De esta forma cualquiera puede exportar los test de dicha plataforma e introducirlos en el juego, convirtiéndolos en las preguntas que los jugadores deberán responder para ganar.

Si se llega al final de los archivos de preguntas, el lector del archivo se reiniciará y volverá a empezar desde el principio.

INTELIGENCIA ARTIFICIAL

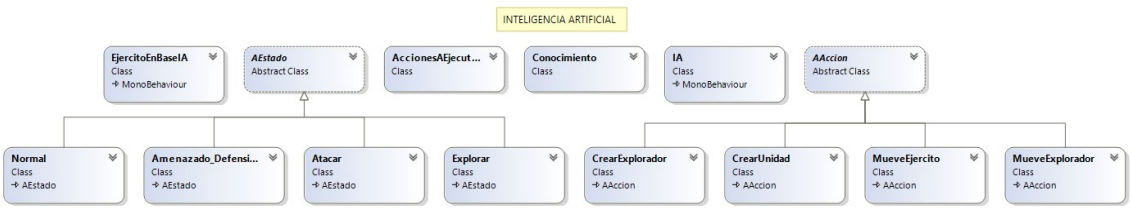


Ilustración 13 - Componente Inteligencia Artificial

Aunque la IA es una parte fundamental del juego, ya que sin ella no existiría un rival a vencer, no es la parte central del proyecto. Por este motivo se ha optado por un

diseño simple pero que permita la escalabilidad. De esta forma, el desarrollo de la IA podrá continuarse en un futuro sin tener que cambiar dicho diseño.

Antes de entrar en materia, es necesario ser consciente de que se han tenido en cuenta las siguientes premisas:

- La IA no puede tener conocimientos que no debería tener. Es decir, ya que el jugador no puede saber que unidades integran un ejército de la IA, esta no puede conocer las unidades que integran un ejército del jugador.
- Al igual que el jugador, la IA solo puede crear una unidad por turno por cada base que posea.
- Ya que en un futuro se pretende implementar un sistema por el cual el jugador solo pueda ver el contenido de las casillas que se encuentren a una distancia determinada de uno de sus ejércitos o bases (conocido en el mundo de los videojuegos como Niebla de Guerra), y no conozca el mapa hasta que lo explore, la IA se ha diseñado para implementar este comportamiento.

Para que la IA sea fácilmente escalable se han tomado 2 medidas. La primera de ellas es dividir el turno de la IA en las siguientes fases que se ejecutan por orden:

1. Fase de Exploración: En esta fase se mueven todas las unidades de tipo explorador.
2. Fase de Movimientos Agresivos: En esta fase se realizan movimientos de ejércitos de carácter ofensivo, es decir, se busca destruir ejércitos o conquistar la base del jugador.
3. Fase de Movimientos Defensivos: En esta fase se realizan movimientos destinados a eliminar ejércitos enemigos que puedan suponer una amenaza o a mover unidades a la base con la intención de que sirvan de defensa.
4. Fase de Resto de Movimientos: En esta fase se mueven todos aquellos ejércitos que sigan teniendo puntos de movimiento. En otras palabras, aquellos ejércitos que no se hayan movido, o no lo hayan hecho hasta agotar todo su movimiento.

5. Fase de Producción: En esta fase se producen las unidades pertinentes.

La segunda medida que se ha tomado para hacer que la IA sea fácilmente escalable está relacionada en cierta media con la anterior. Los estados de la IA solo toman las decisiones, pero no las llevan a cabo. Esto se realiza mediante Acciones. Estas Acciones implementan el patrón Fabrica Abstracta. La IA crea estas acciones y las incluye en una cola. De esta forma, se consigue separar la toma de decisiones y la realización de las mismas. Además, permite la reutilización de código, ya que se evita que esto tenga que ser definido en cada uno de los estados. También es posible que una acción cree otras acciones durante su ejecución. Las acciones implementan un atributo llamado prioridad y la cola de acciones se encuentra dentro de la clase AccionesAEjecutar. De esta forma para conseguir transformar la cola en una cola ordenada con prioridad bastaría con cambiar la clase Acciones AEjecutar. Esto puede resultar interesante si se utilizan acciones que creen otras acciones, ya que así se podría controlar el orden en el que se ejecutan. No se ha realizado este cambio ya que por el momento no se le va a dar uso y podría tener repercusión en el rendimiento.

Por otra parte, la IA necesita tener ciertos conocimientos para poder tomar las decisiones, esto se consigue gracias a una clase llamada Conocimiento. Como su propio nombre indica, almacena todo el conocimiento que la IA tiene sobre el contexto de la partida. Más concretamente:

- Ejércitos de la IA: En este caso, se hace distinción entre ejércitos con unidades de combate y con solo unidades exploradoras para evitar tener que comprobar el tipo de las unidades antes de utilizarlas. Ya que, si no comprobásemos el tipo de unidades y no hiciéramos esta distinción, la IA podría enviar por error a unidades de exploración a atacar y unidades de combate a explorar.

- Bases del jugador: Almacena donde se encuentran todas las bases del jugador encontradas hasta el momento.

- Bases de la IA: Almacena las bases de la IA para poder acceder a ellas cuando sea necesario.

- Mapa: Esta es la parte más compleja ya que almacena una gran cantidad de información. Almacena el conocimiento del mapa, no el de las unidades que

se encuentran en ella. Es decir, la forma que tiene el mapa y el tipo de cada casilla.

También almacena la información del estado del mapa en el turno actual. Esto significa que, en cada turno, se realiza una copia de la información del mapa conocido y se añaden las posiciones de todos los ejércitos y bases amigas y enemigas. Durante el turno y a medida que se toman las decisiones, se incluyen en este mapa las casillas que se espera que sean exploradas a lo largo del mismo. Es decir, si se toma la decisión de mover un ejército, aparte de crear la acción que lo llevará a cabo e introducirla en la lista de acciones, se almacenan las casillas que se espera que sean exploradas debido a dicho movimiento. De esta forma se consigue que la IA explore el mapa de una forma más eficiente.

También se encuentran los métodos de exploración usados por la IA. Por ejemplo, el método que determina las amenazas o ejércitos enemigos que se encuentran cerca de la base de la IA. No obstante, algunos de estos métodos se apoyan en la clase Mapa.

Por último, desde esta parte se realizan los movimientos de los ejércitos de la IA. Esto se debe a que aquí es donde reside toda la información necesaria para hacerlo. Esto se verá más en detalle en la sección Movimiento.

La IA consiste en una máquina de estado que actualmente posee 3 estados. Estos estados cambian conforme la partida va avanzando y el jugador genera y mueve sus ejércitos. Incluso pueden cambiar entre cada una de las fases del turno de la IA. Estos estados son:

- Explorar: Este es el estado por defecto, es decir, el estado inicial de la máquina de estados. Este estado se encuentra activo hasta que la IA encuentre la base del jugador.

  En este estado la IA creará unidades de exploración y unidades para defenderse.

  En caso de que alcance el número máximo de unidades en su base, moverá dichas unidades dependiendo de si hay algún ejército enemigo cerca. Si lo hay, lo atacara. Si no lo hay moverá el ejército a una casilla contigua a su

base para poder defenderse en caso de que fuera necesario. Las unidades de exploración serán movidas para explorar el mapa.

- Atacar: Este estado es activado cuando se conoce la ubicación de la base del jugador y no existe ninguna amenaza inmediata para la base de la IA. Los exploradores serán enviados a explorar las cercanías de la base del jugador. Más concretamente se situarán a una distancia igual a la distancia de visión de las unidades. De esta forma se puede saber aproximadamente cuantas unidades se necesitarían para conquistar dicha base.

  En este momento todos los ejércitos de la IA serán enviados a las cercanías de la base del jugador. En el caso de los ejércitos que tengan al menos un número de unidades igual a la mitad de unidades permitidas por ejército, se enviaran a atacar la base. El resto de ejércitos se quedarán cerca, juntándose entre ellos para formar un número suficiente de unidades para atacar la base en próximos turnos.

  En cuanto a la producción de unidades se centrará en producir unidades de tipo Panhard, ya que poseen un mayor movimiento. Además, se asegurará de que, si en la base no se encuentra ningún Panhard Escudo, este sea el tipo de unidad en concreto que se producirá.

- Amenazado_Defensivo: La máquina de estados pasará a encontrarse en este estado cuando existan un número de unidades (independientemente de cuantos ejércitos allá) del jugador igual a las que se encuentren en la base menos 2. En este momento se mandarán todos los exploradores a las cercanías de la base de la IA con la intención de aumentar el rango de visión que esta posee de esa parte del mapa.

  Los ejércitos atacaran a aquellas amenazas a las que superen numéricamente en cuanto a unidades se refiere y estén dentro de su alcance independientemente de donde se encuentren. A continuación, los ejércitos atacarán o se dirigirán hacia las amenazas cercanas a la base de la IA. Y, por último, las unidades restantes se moverán hacia la base de la IA para protegerla de cualquier posible ataque.

  La producción de unidades se centra en unidades de combate y de defensa. En concreto primero crea Centauros Energía, excepto cuando posee más de

3 unidades y menos de 2 Panhard Escudo. En este caso produce un Panhard Escudo.

CAMBIA TURNO

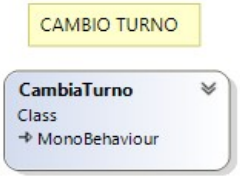


Ilustración 14 - Componente Cambio Turno

Esta sección es muy sencilla. Se encarga de almacenar el número del turno actual. De esta forma los ejércitos saben cuándo renovar sus puntos de movimiento. Además, se encarga de:

- El foco debe ser FocusNothing al principio de cada turno, así que le pide a DetectorClick que pase a ese estado.
- Le indica a la IA que ejecute su turno.
- Como ya se ha dicho cambia el número del turno. Almacena dicho número en el componente Variables Globales para que este accesible por todos los Scripts que lo necesiten.
- Se notifica a todas las bases del jugador que se ha cambiado de turno para que estas produzcan una unidad en caso de tener alguna en la cola de producción.
- Le indica a la clase Preguntas que lance una pregunta.

CÁMARA

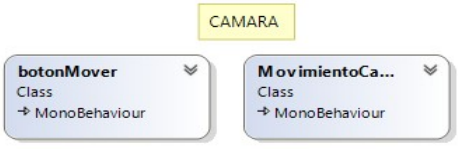


Ilustración 15 - Componente Cámara

Se encarga de mover la cámara ya sea con los botones del teclado o a través de una pantalla táctil.

VARIABLES GLOBALES

Ilustración 16 - Componente Variables Globales

Este componente contiene una clase con varias propiedades estáticas. Estas se encuentran accesibles desde cualquier otro Script. Estas propiedades pueden ser de configuración, como por ejemplo los recursos que da una pregunta acertada, o datos que necesitan ser accedidos por una gran variedad de objetos, como el número del turno. Además, contiene el objeto GameManager, el cual implementa un patrón Singleton [12] y se asegura de que siempre exista una instancia de la clase VariablesGlobales.

MOVIMIENTO

Esta sección no comprende un componente, sino que pretende explicar de una forma más clara el movimiento de las unidades del jugador, el cual está implementado en el componente Mapa, y de las unidades de la IA, implementado en el componente Inteligencia Artificial.

El movimiento de los ejércitos es llevado a cabo por la clase Mapa en el caso del jugador, y por la clase Conocimiento en el caso de la Inteligencia Artificial. Pero independientemente de esto, todos cumplen con lo siguiente:

Primero, el componente Ruta de la casilla desde la que va a partir el movimiento almacena un valor nulo. Más adelante se explicará por qué.

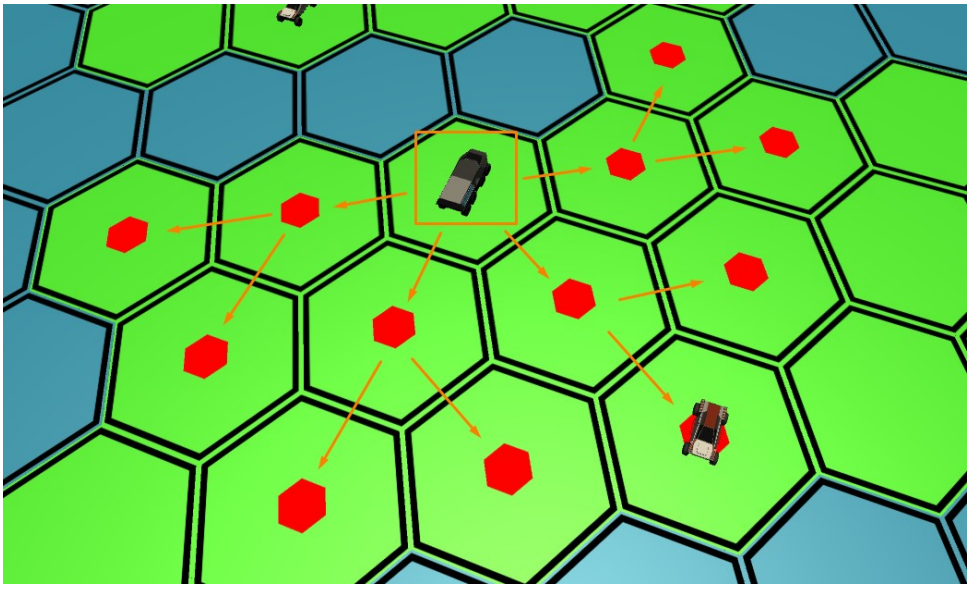


Ilustración 17 - Movimientos Posibles

A continuación, se realiza una exploración en anchura tomando como nodo inicial la casilla desde donde va a partir el movimiento, tal y como se ve en la imagen anterior [Ilustración 17].

Para controlar la distancia se hacen dos colas. Cada vez que se encola una nueva casilla se hace en la segunda de las mismas, y cuando la primera de las colas está vacía, ambas colas se intercambian. Al realizar esto sabemos que hemos bajado un nivel en la búsqueda y podemos controlar que solo exploramos aquellas casillas a las que el ejército puede llegar con los puntos de movimiento que tenga disponibles. También al explorar cada casilla se almacena en su componente Ruta un enlace hacia la casilla desde la que se exploró, ya veremos más adelante para qué. En caso de encontrar un ejército o base enemiga, esta se almacena como casilla a la que se puede acceder, pero no se exploran nodos desde ella. Es decir, no se introduce a la segunda cola como si ocurre con el resto de casillas. No obstante, si después de mover a dicha casilla al ejército le quedan puntos de movimiento, este podrá volver a moverse tantas casillas como puntos de movimiento le queden.

En la siguiente imagen se muestra como se almacenan en las casillas, desde que otra casilla fueron exploradas por el algoritmo. Por ejemplo, como se ve en la imagen siguiente [Ilustración 18], la casilla C1 fue explorada desde la casilla B1, y por lo tanto, esto se almacena en el componente Ruta de la casilla C1.

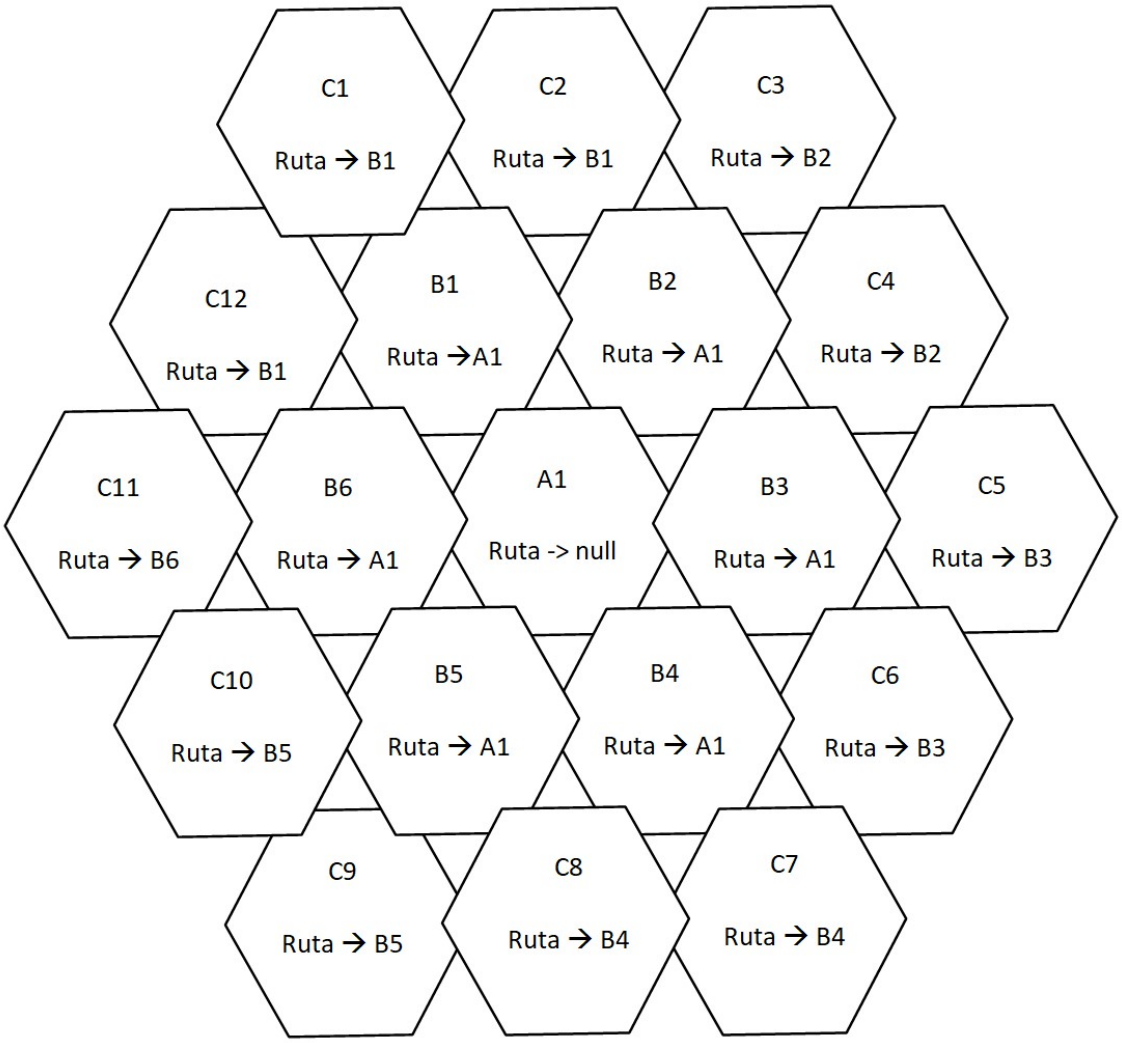


Ilustración 18 - Malla Hexagonal

Si fue el jugador el que selecciono uno de sus ejércitos o una de sus bases con un ejército dentro, además se van marcando visualmente las casillas exploradas de esta forma para que el jugador sepa hasta donde puede mover el ejército.

Por último, hay 2 posibilidades, que la IA determine hacia donde quiere mover el ejército, o que el jugador haga clic en una de las casillas marcadas. Sea como fuere es el momento de mover el ejército, pero primero debemos determinar su ruta.

Determinar la ruta es la parte más sencilla, puesto que sabemos donde tiene que acabar el ejército. Basta con almacenar la casilla de destino y extraer el enlace que hemos almacenado previamente en el componente Ruta. Repetiremos este proceso hasta que encontremos un componente Ruta con un enlace nulo. Eso significa que hemos llegado al punto de partida y debemos mover lo que haya en esa casilla por los nodos que acabamos de almacenar de forma inversa. Es decir, el

ultimo nodo que almacenamos será la primera casilla a la que movamos el ejército y así sucesivamente.

Estos son todos los componentes que forman parte del videojuego. A continuación, se encuentra otra sección que explica las modificaciones realizadas para lograr que el juego pueda ser ejecutado bajo el sistema operativo Android.

Versión Android

Aunque Unity 3D es capaz de compilar los proyectos para multitud de plataformas, se requiere una personalización dependiendo de las características de la plataforma.

…, los videojuegos se pueden encontrar en ordenadores personales, consolas de juego de sobremesa, consolas portátiles, dispositivos móviles como por ejemplo los smartphones, o incluso en las redes sociales como medio de soporte para el entretenimiento de cualquier tipo de usuario. [10]

En este caso, el uso del videojuego usando ratón y teclado o pantalla táctil. Para ello, Unity 3D proporciona una herramienta que permite que se realicen acciones antes de que el proyecto sea compilado, dependiendo de la plataforma, tal y como se muestra la siguiente imagen [Ilustración 19].

```
#if UNITY_ANDROID
       if (canvControler.BotonMoverPulsado)
       {
              return rightClick(hit);
       }
#endif
```

Ilustración 19 - Código Android

De esta forma, el código anterior solo se incluirá en el código final cuando estemos compilando el proyecto para una plataforma Android.

Para adaptar el proyecto a las plataformas con pantallas táctiles se han realizado las siguientes modificaciones:

- Inclusión de botón MOVER: En la versión de computadora, el jugador movía sus ejércitos haciendo clic con el botón derecho del ratón. Para adaptar esto a las pantallas táctiles se ha incluido un botón llamado MOVER. Este botón es mostrado por los estados FocusArmy y FocusBase cuando pasan a ser el

estado activo, y lo ocultan cuando dejan de serlo. Esto solo ocurre en la versión Android gracias al método mostrado anteriormente.

Este botón es semitransparente cuando se encuentra desactivado y totalmente visible en caso contrario. Este estado cambia cuando se toca con el dedo. Cuando esta desactivado todo funciona de forma normal, es decir, al seleccionar una casilla marcada como movimiento posible se cambia el foco en vez de realizar el movimiento (clic izquierdo del ratón). Sin embargo, cuando está activado y se toca una casilla marcada, el ejército seleccionado es movido a dicha casilla (clic derecho del ratón).

Por último, se ha modificado el código de DetectorClick para que en caso de arrastrar el dedo, como se ve más adelante, para mover la cámara, no se reconozca como un clic con el que seleccionar o mover ejércitos. Si simplemente se toca la pantalla, se generan dos eventos en el punto tocado, uno al presionar y otro al dejar de hacerlo. Estos eventos no solo contienen coordenadas, sino también un identificador común a ambos eventos. De esta forma, y utilizando dicho ID, se puede determinar cuándo se toca la pantalla y cuando se arrastra un dedo por ella comparando las coordenadas.

- Movimiento de la cámara: En un teléfono móvil o una tableta por lo general no se dispone de un teclado, por lo tanto, los controles diseñados para computadora (las teclas w, a, s y d) no pueden ser utilizados. Para solucionar esto se ha incluido en el script MovimientoCamara un código que hace lo siguiente:

    1. Comprueba si hay exactamente una pulsación en la pantalla. En caso afirmativo continúa, en caso contrario termina.

    2. Comprueba si tiene alguna pulsación almacenada, si esta corresponde con la recibida en este ciclo de ejecución del videojuego, y si la pulsación es de tipo movimiento. En caso afirmativo, mueve la cámara proporcionalmente a la distancia que hay entre la pulsación almacenada y la recibida en este ciclo.

    3. Almacena la pulsación recibida en este ciclo de ejecución del videojuego y termina. Al almacenar la pulsación, sobrescribe la almacenada anteriormente en caso de haberla.

Diseño e Implementación de la Aplicación de Puntuaciones

Esta aplicación ha sido diseñada usando una arquitectura en tres capas, las cuales veremos a continuación. Para implementar esta aplicación, se ha utilizado el lenguaje de programación Java, junto con el framework Spring Boot y Maven.

Spring Boot makes it easy to create stand-alone, product-grade Spring based Applications that you can "just run". [8]

CAPA ACCESO

En esta capa se encuentran definidas dos tipos de clases. El primero de ellos es lo que se ha denominado como controladores. Estas clases definen los recursos que la API expone para ser usados desde otras aplicaciones tal y como se muestra en la siguiente imagen [Ilustración 20].

```
public ScoreList getScoreList(@ApiParam(value = "Skip an amount of results.") @RequestParam(value = "skip", defaultValue = "0") int skip,
                              @ApiParam(value = "Maximum amount of results.") @RequestParam(value = "maxItems", defaultValue = "20") int maxItems){
    LOGGER.debug("Getting softwareWars list. GET REST API called.");
    return scoreService.getScoreList(skip, maxItems);
}
```

Ilustración 20 - Código Controller

Por último pero no por ello menos importante, se encuentran una serie de interfaces que también definen los recursos de la API. De forma adicional, se han utilizado dichas interfaces para separar las notaciones del framework Swagger de forma que las clases que contienen los recursos queden lo más limpias posibles. Por este motivo podemos ver en la imagen siguiente [Ilustración 21] más notaciones que en la clase mostrada en la imagen anterior [Ilustración 20].

```
@ApiOperation(value = "Get softwareWars list", nickname = "getScoreList", response = ScoreList.class, tags="scores")
@GetMapping(value = "",
        produces = "application/json")
@ApiResponses({
        @ApiResponse(code = 200, message = HttpStatusMessages.SUCCESSFUL_OPERATION, response = ScoreList.class),
        @ApiResponse(code = 500, message = HttpStatusMessages.UNEXPECTED_INTERNAL_ERROR, response = DefaultError.class)
})
ScoreList getScoreList(@ApiParam(value = "Skip an amount of results.", required = true ) @RequestParam("skip") int skip,
                       @ApiParam(value = "Maximum amount of results.", required = true ) @RequestParam("maxItems") int maxItems);
```

Ilustración 21 - Notación Swagger

Estas notaciones se utilizan para generar una interfaz que sirve como documentación de la API, pero que además es interactiva tal y como se ve en la imagen siguiente [Ilustración 22]. De esta forma no solo se genera de forma

programática documentación sobre los recursos expuestos por la Api, sino que además es posible probar dichos recursos fácilmente. Dicha interfaz está disponible en el mismo servidor en el que se encuentra la aplicación.

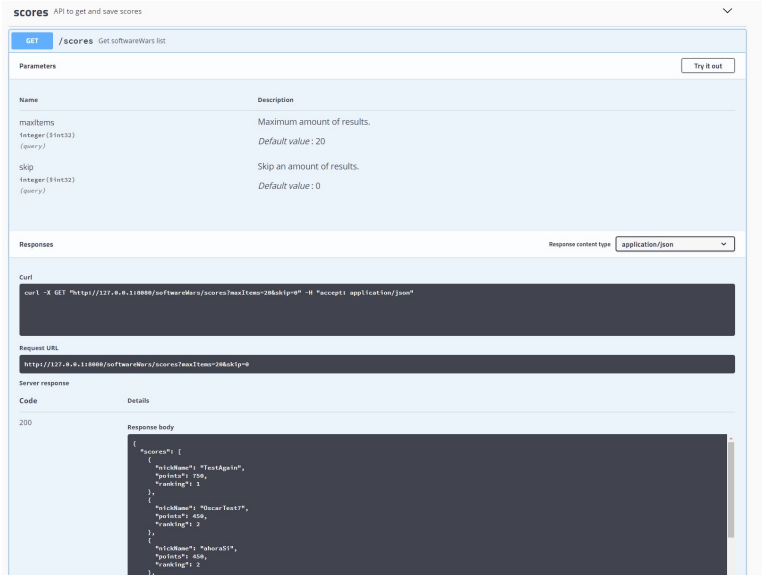
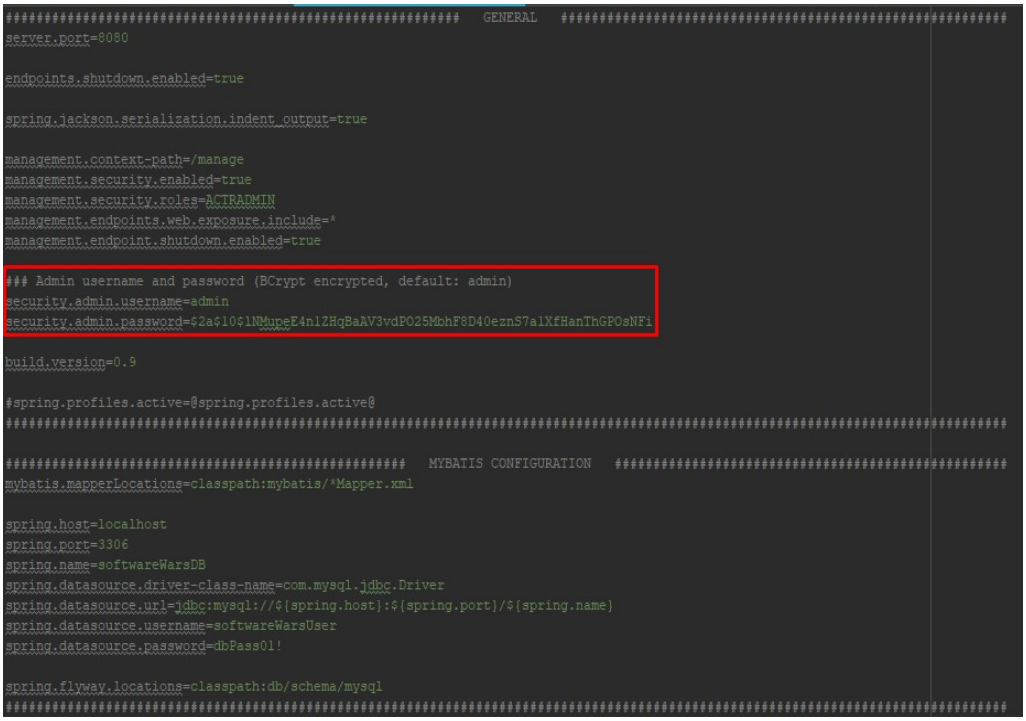


Ilustración 22 - Interfaz Swagger

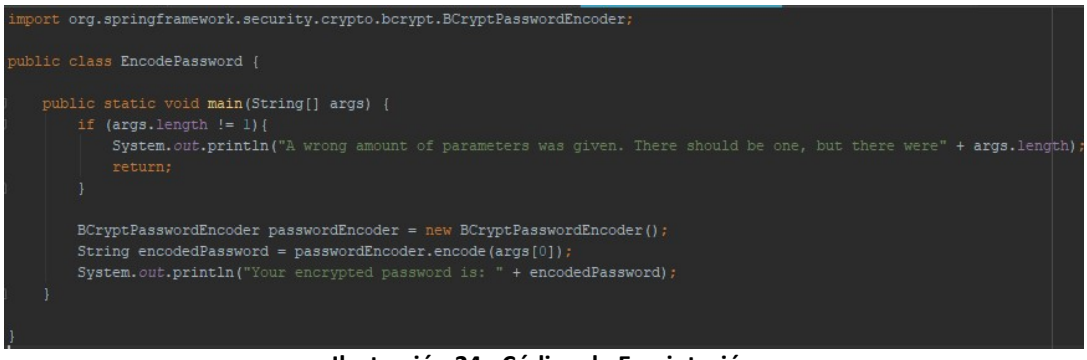
Por supuesto, esta interfaz y los recursos de la API no están disponibles para cualquiera que conozca la URL. Todo ello está protegido mediante usuario y contraseña. Dicha contraseña se encuentra en el archivo de configuración por defecto de todas las aplicaciones Spring Boot “application.properties”, del cual se muestra un fragmento a continuación [Ilustración 23].



```
#############################################################   GENERAL   ##########################################################
server.port=8080

endpoints.shutdown.enabled=true

spring.jackson.serialization.indent_output=true

management.context-path=/manage
management.security.enabled=true
management.security.roles=ACTRADMIN
management.endpoints.web.exposure.include=*
management.endpoint.shutdown.enabled=true

## Admin username and password (BCrypt encrypted, default: admin)
security.admin.username=admin
security.admin.password=$2a$10$lNMupeE4n1ZHqBaAV3vdPO25MbhF8D40eznS7alXfHanThGPOsNFi

build.version=0.9

#spring.profiles.active=@spring.profiles.active@
#################################################################################################################################

################################################   MYBATIS CONFIGURATION   ######################################################
mybatis.mapperLocations=classpath:mybatis/*Mapper.xml

spring.host=localhost
spring.port=3306
spring.name=softwareWarsDB
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.datasource.url=jdbc:mysql://${spring.host}:${spring.port}/${spring.name}
spring.datasource.username=softwareWarsUser
spring.datasource.password=dbPass01!

spring.flyway.locations=classpath:db/schema/mysql
#################################################################################################################################
```

Ilustración 23 - Archivo de Propiedades

La contraseña se encuentra cifrada usando Bcrypt. Para facilitar esto en el código se encuentra una clase llamada EncodePassword, que se puede compilar como una aplicación de consola con la cual se puede encriptar cualquier texto que se le dé como parámetro. El código de dicha aplicación puede encontrarse en la imagen siguiente [Ilustración 24].



```
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

public class EncodePassword {

    public static void main(String[] args) {
        if (args.length != 1){
            System.out.println("A wrong amount of parameters was given. There should be one, but there were" + args.length);
            return;
        }

        BCryptPasswordEncoder passwordEncoder = new BCryptPasswordEncoder();
        String encodedPassword = passwordEncoder.encode(args[0]);
        System.out.println("Your encrypted password is: " + encodedPassword);
    }

}
```

Ilustración 24 - Código de Encriptación

CAPA NEGOCIO

Esta capa es la más uniforme, ya que no está compuesta por diferentes tipos de clases. Las clases contenidas en esta capa se denominan como servicios, y contienen toda la lógica de la aplicación.

CAPA PERSISTENCIA

Las funcionalidades de esta capa se centran en el almacenaje de la información, pudiéndose distinguir múltiples partes. La primera de estas es la creación y mantenimiento del esquema de la base de datos. Para este propósito se hace uso del framework Flyway. Este permite realizar un versionado del esquema, generando y manteniendo automáticamente una tabla dentro de la base de datos en la que se registra la versión actual del esquema. Esto permite la creación de parches, los cuales solo se aplican si no han sido aplicados ya.

La segunda y última funcionalidad de esta capa es el almacenaje y recuperación de información de la base de datos. Para realizar esto, se ha utilizado MyBatis. Este framework permite ejecutar sentencias SQL de forma sencilla. Para esto, dichas sentencias se definen en un archivo xml, y automáticamente se enlazan con métodos definidos en una interfaz a través del nombre del método y un atributo en la etiqueta xml que contiene la sentencia.

Hay que tener en cuenta que en la carpeta test se encuentran algunas clases de esta capa, las cuales son utilizadas para realizar los test. Cuando se ejecuta la aplicación, esta es capaz de establecer una conexión con una base de datos MySQL. Sin embargo, cuando se ejecutan los test, la aplicación crea una base de datos H2 dentro del propio IDE IntelliJ.

OTROS

Existen múltiples archivos de configuración para configurar todos los frameworks utilizados. Además, existen archivos de propiedades que se pueden utilizar para modificar algunos parámetros. Cabe destacar que estos archivos pueden ser sobrescritos fuera del archivo war de la forma habitual cuando se utiliza una aplicación Spring Boot.

PRUEBAS

Últimamente se está poniendo de moda en el mundo de los videojuegos, los juegos con acceso anticipado y las betas abiertas. Esto es utilizado por multitud de compañías para ahorrarse parte de las pruebas necesarias para asegurarse de que sus juegos funcionan correctamente. Y es que esta es una fase muy importante, ya que un solo error puede arruinar la experiencia del jugador. Durante el transcurso de este proyecto se han llevado a cabo 4 tipos de pruebas diferentes: pruebas unitarias, pruebas de aceptación, pruebas de integración y pruebas de rendimiento.

Pruebas Unitarias

Debido a la imposibilidad de realizar pruebas unitarias de forma normal en el videojuego, ya que la mayoría de las clases hace uso del motor gráfico, se han realizado estas pruebas de 3 formas diferentes para comprobar el correcto funcionamiento del mismo.

Primero se han realizado pruebas unitarias en un proyecto en Visual Studio de las clases en las que fue posible. Esas clases son AccionesAEjecutar, Coordenada y todas las clases que representan las unidades.

En segundo lugar, se han realizado pruebas con la herramienta Test Tool para Unity 3D, la cual se puede encontrar de forma gratuita en la asset store. Dicha herramienta se utiliza añadiendo el componente Assertion Component a aquellos objetos que queramos testear. Esto solo nos permite testear los valores de sus atributos, por lo que es una herramienta limitada. Después de realizar los test, se ha removido el componente de los objetos del juego para evitar que suponga una carga a la hora de ejecutar el juego. No obstante, el componente sigue incluido entre los assets, ya que se utilizará en un futuro cuando se continúe el desarrollo del videojuego.

Por último, se ha utilizado la clase Debug proporcionada por Unity 3D, la cual permite definir mensajes que aparecen en la consola cuando se ejecuta el juego desde el motor. Estos mensajes pueden ser de tipo Log, LogWarning y LogError.

Después de hacer las pruebas. Dichas instrucciones se han borrado o comentado, ya que dificultaban la realización de pruebas de otras clases debido al gran número de mensajes en la consola. Los únicos que no se han eliminado han sido los de la IA, ya que se prevé que se siga desarrollando en un futuro próximo.

Respecto a la aplicación Java, existe dentro de la propia aplicación test unitarios desarrollados con Junit. Para esto, primero se encuentran los test de la capa de persistencia. Una vez que se ha comprobado que esta funciona como se esperaba, se realizan los de la capa de Negocio. De esta forma, se hacen pruebas unitarias y de integración de forma simultánea. Por último, no se han diseñado test para la capa de acceso, ya que esta no contiene ningún tipo de lógica o complejidad. En el caso de que esto cambie en un futuro, será necesario crearlas. Como ya hemos explicado, las pruebas se realizan utilizando una base de datos H2 creada localmente bajo demanda en el IDE IntelliJ durante la ejecución de los test.

Pruebas de Integración

Al igual que en el caso anterior, las posibilidades de hacer este tipo de pruebas en la aplicación del videojuego son muy limitadas incluso con la utilización de software de terceros como es la herramienta Test Tool para Unity 3D. No obstante, se puede considerar que cada vez que se ejecuta el juego se está realizando pruebas de integración, ya que durante el transcurso, las diferentes pantallas, los objetos del juego, etc, interaccionan entre ellos.

Respecto a la aplicación que gestiona las puntuaciones, como ya hemos explicado, las pruebas de unitarias se realizan de forma incremental, lo que sirve además de pruebas de integración.

Pruebas de Sistema

Debido a que el videojuego debe poder ejecutarse en dispositivos Android, es muy importante que utilice la menor cantidad posible de recursos. Para poder medir el rendimiento del videojuego e intentar optimizarlo se ha utilizado la herramienta Profiler. Esta herramienta calcula en tiempo real los recursos utilizados por el videojuego.

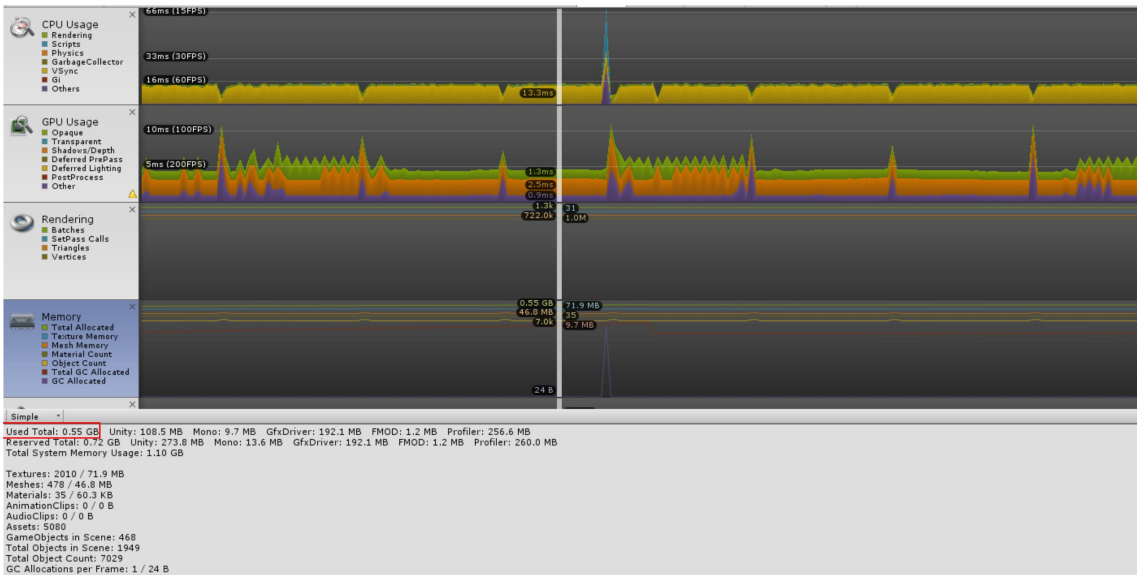


Ilustración 25 - Prueba de Rendimiento

Como se puede comprobar en la imagen anterior [Ilustración 25], el uso de memoria del videojuego es de 0,55GB. Además, se han reservado otros 0,72GB de RAM. Esto cumple con el requisito no funcional establecido, el cual es que el uso de memoria máximo debe ser menor de 2GB.

Pruebas de Aceptación

Las pruebas de aceptación están orientadas a comprobar que el juego desarrollado cumple con los requisitos establecidos por el cliente. En este caso, para la aplicación Spring Boot esto es suficiente, sin embargo, para el videojuego, no solo basta con cumplir con los requisitos, sino que tiene que debe resultar una experiencia entretenida.

Las pruebas de aceptación fueron llevadas a cabo por un grupo de egresados, de estudiantes y personas de rango de edad similar al público objetivo. Estas pruebas constaron de múltiples ejecuciones del juego creando la consiguiente clasificación, la cual fue común a todas las pruebas de aceptación realizadas. Después de dichas pruebas se comprobó que ambas aplicaciones cumplen con los objetivos establecidos. Además, se propusieron algunas mejoras o nuevas funcionalidades como la mejora de la IA o la creación de un nuevo panel en el que en el que se puedan visualizar de una forma sencilla los combates en vez de simplemente ver las unidades supervivientes. Estas mejoras y añadidos se abordarán en trabajos futuros.

CONCLUSIÓN

Una vez finalizado el proyecto, es el momento de echar la vista atrás para recapacitar sobre el trabajo realizado, las dificultades encontradas y como se han superado.

Este proyecto en realidad está compuesto por dos aplicaciones diferentes. La primera de estas es un videojuego, el cual permite entre otras cosas utilizar test creados en la plataforma Moodle como preguntas que el jugador debe responder para obtener recursos y puntos. De esta forma, si el jugador quiere ganar y lograr un buen puesto dentro de la clasificación deberá responder correctamente a las mismas.

Y es que ganar a la IA no es la única motivación que tiene el jugador para intentar responder correctamente a las preguntas. La segunda aplicación esta creada usando el framework Spring Boot entre otros, y permite almacenar de forma externa las puntuaciones de todos los jugadores. Esto permite mostrar una clasificación según los puntos obtenidos.

Esto junto al hecho de que la aplicación, como se demostró durante las pruebas de aceptación, se puede ejecutar en dispositivos Android, hace evidente que el proyecto ha cumplido con todos los objetivos primarios y secundarios establecidos.

Conclusiones personales

En un principio se eligió este proyecto por iniciativa del propio alumno, ya que le pareció interesante adentrarse en un mundo en el que normalmente hace de consumidor y no de desarrollador. No obstante, la realización de este proyecto le ha ayudado a entender por qué algunas cosas en los videojuegos son como son y a valorar mucho más a los profesionales que se dedican a esto. Además, se ha conseguido cubrir los motivos para desarrollar este proyecto, ya que el juego puede ser utilizado para realizar test con los que obtener conocimientos de una forma dinámica y entretenida.

Este proyecto se ha realizado tal y como lo haría un ingeniero. Se ha realizado una recopilación de requisitos, se ha diseñado e implementado de forma que el

software cumpla con dichos requisitos y sea fácilmente escalable. También se han realizado diferentes tipos de pruebas que verifiquen que todo funciona correctamente.

A lo largo del Grado, el alumno tuvo que superar con éxito las diferentes asignaturas, adquiriendo los conocimientos necesarios para realizar actividades como la recolección de requisitos, diseño software, etc. Pero no es hasta el momento en el que alguien se sienta solo frente a un sistema que desconoce, que empieza a desarrollar la que en mi opinión es una de las habilidades más importantes en el mundo de los videojuegos. Ser capaz de aprender de forma autónoma, es decir, ser capaz basándose en la documentación de aprender a solucionar todos los problemas que se le planteen. Lo más difícil de este proyecto no ha sido la programación en sí misma, sino ser capaz de forma autónoma, de aprender el funcionamiento de una gran herramienta como es Unity 3D y de la amplia variedad de tecnologías utilizadas.

Con todo esto, el alumno considera que el proyecto ha sido un éxito tanto por haber alcanzado los objetivos esperados, como por lo aprendido en el transcurso del mismo.

FUTUROS TRABAJOS

Aunque el proyecto está terminado, el alumno pretende seguir ampliando el juego, con la intención de que algún día sea publicado. Y es que aún existen un montón de funcionalidades que se pueden implantar, como por ejemplo, sistemas multijugador. Además, durante las pruebas de aceptación algunas de las personas que realizaron las pruebas expresaron que algunos de los trabajos futuros podrían ser la mejora de la IA o la creación de un nuevo panel en el que en el que se puedan visualizar de una forma sencilla los combates en vez de simplemente ver las unidades supervivientes.

Además, se pretende realizar un tutorial, una versión para IOS, y algunas mejoras gráficas.

BIBLIOGRAFÍA

[1] Modelo Incremental - http://goo.gl/9dJAdA - Visto a 01/09/2016

[2] Versiones de Unity 3D - http://goo.gl/E47BSf - Visto a 23/08/2016

[3] Mapas Hexagonales - http://goo.gl/Na2nGh - Visto a 23/08/2016

[4] JuegoSerio - http://goo.gl/KGGiIR - Visto a 01/09/2016

[5] Tipos de Serious Games - http://goo.gl/zfQkj5 - Visto a 01/09/2016

[6] Microsoft C# - http://goo.gl/ul6zH8 - Visto a 01/09/2016

[7] Diseño Software - http://goo.gl/p4CqpZ - Visto a 02/09/2016

[8] Spring - https://bit.ly/2X0IZc5 - Visto a 10/11/2019

[9] Pro Git – ISBN: 978-1-4842-0077-3 - Visto a 12/11/2019

[10] Desarrollo de Videojuegos: Un enfoque práctico – ISBN: 978-1517309558

[11] Unity 5.X Cookbook – ISBN: 978-1-78439-136-2

[12] Programación concurrente en Java: principios y patrones de diseño – ISBN: 84-7829-038-9

[13] UML distilled: a brief guide to the standard object modeling language – ISBN: 0-321-19368-7