Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Ajedrez táctico para móvil

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Aitor Bastida González

Tutor: Manuel Melchor Agustí

2019-2020

Resumen

El género de videojuegos *Autochess* es tremendamente novedoso, y una gran cantidad de prestigiosas compañías y pequeños estudios se han lanzado a su mercado de forma voraz. Sin embargo, todo lo que encontramos son clones de unas aplicaciones con otras con pequeñas modificaciones, pero sin ningún aspecto que las haga especiales. Nosotros proponemos uno: Este tipo de juegos son muy aptos para ordenador, pero en dispositivos móviles están pasando sin pena ni gloria ya que las partidas de 40 minutos en un teléfono móvil no le gustan a nadie. Frente a este problema, intentamos traer una solución, un juego *autochess* multijugador competitivo de 1 contra 1 en el que ideamos un modelo de juego para que las partidas duren entre 5 y 9 minutos, intentaremos hacer esta solución realidad a través del motor de videojuegos Unity.

Explicaremos de qué géneros de videojuegos hereda nuestra aplicación y qué elementos novedosos añadimos por nuestra parte para hacerla lo más especial posible e intentar hacernos un hueco en este nicho inexplorado de la industria, con lo que incluso, si la idea llegara a ejecutarse, podría definir o modificar el estado del arte actual en la industria de este tipo de videojuegos.

Resum

El gènere de videojocs *Autochess* és tremendament nou, i una gran quantitat de prestigioses companyies i xicotets estudis s'han llançat al seu mercat de manera voraç. No obstant això, tot el que trobem són clons d'unes aplicacions amb altres amb xicotetes modificacions, però sense cap aspecte que les faça úniques. Nosaltres proposem un: Aquest tipus de jocs són molt aptes per a ordinador, però en dispositius mòbils estan passant sense pena ni glòria ja que les partides de 40 minuts en un telèfon mòbil no li agraden a ningú. Enfront d'aquest problema, intentem portar una solució, un joc *autochess* multijugador competitiu d'1 contra 1 en el qual ideem un model de joc perquè les partides duren entre 5 i 9 minuts, intentarem fer aquesta solució realitat a través del motor de videojocs Unity.

Explicarem de quins gèneres de videojocs hereta la nostra aplicació i quins elements nous afegim per part nostra per a fer-la el més especial possible i intentar fer-nos un buit en aquest espai inexplorat de la indústria, amb el que fins i tot, si la idea arribara a executar-se, podria definir o modificar l'estat de l'art actual en la indústria d'aquest tipus de videojocs

Abstract

Autochess games are tremendously new, and a lot of high-qualified companies and indie studios want to enter its market in a strong way. Nonetheless, if we search for that content, we are only about to find the same game copied from another one with a small amount of modifications, with no aspect that makes them special or unique. We propose one: These games are perfect for PC, but in mobile devices, they are not hitting too much, and we think this is due to the game length, no one wants to spend 40 minutes in the same game on a mobile device. But we bring a solution, a multiplayer *autochess* game which has 1 versus 1 battle and we aim for games that last between 5 and 9 minutes. We will try to make this theory a real product with Unity Engine.

We will explain which videogame genres our application inherits, and which elements we add in order to make it more special and unique, to find this unexplored spot in the industry, and if the idea would make till the end, we could change or modify the current state of art in this genre of videogames industry.

Índice general

Índice de figuras

Índice de tablas

1. Introducción

El campo de los videojuegos es un sector que, a nivel empresarial, tecnológico y social, no ha parado de crecer de forma exponencial en los últimos años. Debido a esto, las grandes cantidades de propuestas que las empresas, pequeños estudios o incluso gente en solitario proponen, hacen del mercado de los videojuegos uno muy competitivo y voraz, por tanto, existe una gran disputa sobre cómo entrar en el mercado y crear un producto que destaque sobre los demás y capte la atención del público.

A pesar de las dificultades y de los pocos rincones inexplorados… ¡Hemos encontrado uno! El desarrollo de un videojuego con unas características que definirán un nuevo estado del arte, algo que nadie ha enseñado todavía en el mercado: Un *autochess* de partidas rápidas para dispositivos móviles. Puede que esto ahora mismo no tenga sentido, pero iremos desarrollando varios conceptos y características de la idea y el desarrollo a lo largo de la memoria.

¿Qué es un *autochess* y por qué el nuestro aporta algo nuevo si el género ya existe? ¿Por qué proclamamos un nuevo estado del arte? ¿Por qué nos sentimos tan seguros de atraer a todos los amantes del género en dispositivos móviles? Todas estas preguntas y más, irán desarrollándose y respondiéndose en los siguientes apartados.

1.1 Motivación

Este apartado de la memoria va a ser el único en el que se utilice la primera persona del singular, pero es necesario, ya que la motivación de este proyecto es muy importante:

Hay algo que los amantes del género *autochess* todavía no tienen, y es un juego de su género favorito para dispositivo móvil sin tener que estar treinta o cuarenta minutos pegado al mismo, porque seamos sinceros, todos a día de hoy somos usuarios de teléfono móvil, juguemos o no juguemos a videojuegos, y a nadie le gusta estar más de cinco minutos pegado a lo mismo. Cambiamos de una conversación a otra, de una red social a otra, de una aplicación a otra, **de una partida a otra**, todo en cuestión de segundos o minutos, y entre los usuarios de dispositivo móvil que también lo utilizan para jugar, no es distinto. Las partidas de este género de videojuegos (que luego explicaremos con detalle), son largas, perfectas para usuarios de ordenador o consola… Pero no para usuarios de móvil. Existen versiones de esos mismos juegos *autochess* de computadora para teléfono móvil, pero siguen siendo partidas demasiado densas y largas… ¿Por qué no simplificar un poco el juego y darle al usuario partidas de entre 5 y 9 minutos? Así es como funcionan la mayoría de juegos y aplicaciones móviles, para que lo que sea que hagas con tu móvil, lo puedas hacer de camino al trabajo, en el metro, mientras esperas la cena en un restaurante de comida rápida, para enseñárselo a tu amigo mientras esperáis al integrante del grupo que siempre llega tarde, e incluso, mientras vas al baño. Esta es nuestra primera motivación, moldear un género de videojuegos como es *autochess*, para crear algo completamente nuevo y destacar en el mercado, ser los primeros en esto.

La segunda motivación, pero no por ello menos importante, es mi amor incondicional por los videojuegos. Los llevo disfrutando desde pequeño y me han nutrido de valores y experiencias que faltaron en mi ámbito familiar e incluso social, personalmente, me han hecho crecer como persona y son parte de quién soy hoy, enseñándome a disfrutar y llorar de felicidad como

podrían hacerlo la música o la pareja sentimental más dedicadas y a su vez, mostrándome la dureza de la realidad como hace la vida misma. Siempre he querido estudiar videojuegos, pero debido a la situación económica de mi familia y la situación del sector en el ámbito académico en nuestro país, no he tenido siquiera la opción. Siempre he soñado con ser parte del desarrollo de un videojuego y escribir un argumento que mande un mensaje que, con que le llegara a una persona, ya me sentiría complacido. Este no va a ser el caso, no voy a volcar un mensaje porque en este tipo de juegos no tiene cabida, pero el desarrollo de este videojuego y esta memoria, son el primer paso para demostrarme a mí mismo que puedo agarrar parte de ese sueño y tirar de él para ver hasta dónde llego.

1.2 Objetivos

El objetivo principal es puramente el desarrollo de un videojuego para dispositivos móviles. Dicho objetivo es tan claro como amplio, por tanto, podemos descomponerlo en los siguientes subobjetivos, los cuales están estrechamente relacionados entre sí:

- Desarrollar exitosamente una aplicación interactiva y funcional, que pueda ser utilizada por el usuario con un fin lúdico.
    - Los objetivos específicos y técnicos sobre el desarrollo de la aplicación de detallan en el apartado 4.1.
- Definir un nuevo estado del arte utilizando aspectos de dos géneros ya existentes y añadiendo nuestro nuevo trabajo.
    - Cubrir las carencias que están teniendo este tipo de juegos en los dispositivos móviles a fin de volvernos punteros en el sector.
- Producir un videojuego que apuntara a ser competitivo en el mercado y hacer un estudio del mismo para prever el impacto que tendría y si fuera beneficioso invertir en él para obtener un beneficio monetario.

Cabe destacar que no sólo hay objetivos, existen limitaciones, y la más grande de ellas es el apartado gráfico. No tener diseñador gráfico nos obliga a utilizar modelos que no pueden ser publicados para utilizarlos de ejemplo en el desarrollo del juego, por tanto, esto entra en conflicto con el tercer subobjetivo. No obstante, consideramos que el estudio de mercado es necesario para considerar que impacto tendría con un equipo de desarrollo que pueda cubrir y solventar unas limitaciones medibles y determinadas.

1.3 Estructura de la memoria

Este trabajo está compuesto por 6 apartados. A continuación, se menciona la información que se encontrará en cada uno de ellos:

1. **Introducción.** En el primer apartado, se introduce al lector al tema para que entienda de qué va a tratar, y posteriormente, se plantea la motivación que ha dado paso al desarrollo y los objetivos a alcanzar con el mismo.

2. **Análisis del estado del arte.** Se estudia el estado del arte actual en el contexto del sector para el que se está desarrollando el proyecto, así como el impacto que este trabajo tendría en el estado del arte y en qué medida podría modificarlo.

3. **Géneros heredados.** Se exponen los géneros de videojuegos de los que principalmente heredamos características para nuestro producto. Posteriormente, se analizan en profundidad sus características y se comparan con las de nuestro producto, para que puedan entenderse las similitudes, diferencias y nuevos aspectos que conforman el desarrollo.

4. **Desarrollo del videojuego.** En este apartado, se encontrará todo lo referente al desarrollo técnico y producción del videojuego dentro del ámbito de programación y diseño, así como sus funcionalidades y aspectos propios.

5. **Estudio de mercado.** Aquí se mostrará el impacto que podría tener nuestro producto en el mercado de aplicaciones móviles, qué características avalan un prometedor resultado y cuáles serían necesarias para ser competitivos con el resto de productos.

6. **Conclusiones.** En el último apartado, se reflejan las conclusiones obtenidas durante el desarrollo del trabajo y al finalizar el mismo. Además, se plantean posibles mejoras que podrían desarrollarse para que el producto fuera más competitivo y atractivo para el usuario.

2. Análisis del estado del arte

¿Cuál es la situación del estado del arte actual en este campo de los videojuegos? ¿Por qué este trabajo es un posible nicho inexplorado en la industria? ¿Cómo es posible si existen tantos géneros de videojuegos? ¿Cuántos de estos hay y qué aspectos definen que un videojuego pertenezca a un género u otro?

Todas estas preguntan serán respondidas en este apartado con detalle. Además, analizaremos (sin entrar todavía en las especificaciones técnicas, para ello, véase el apartado 3) la situación actual en este lado de la industria del videojuego y cómo está estrechamente relacionada con nuestra propuesta y, obviamente, por qué nuestro trabajo podría definir un nuevo estado del arte, detallando las diferencias que lo hacen único.

2.1 Géneros de videojuegos

En primer lugar, para poder entender hacia dónde vamos, necesitamos saber dónde nos encontramos y qué hay a nuestro alrededor, para ello, es necesario hablar del concepto: **Géneros de videojuegos**. Nuestra propuesta podría definir un nuevo estado del arte gracias a extraer características de dos géneros distintos combinados con la adición de características que lo hacen más único si cabe.

Existen muchas formas de clasificar los videojuegos en base a sus similitudes para poder ordenarlos, pero lo más típico es utilizando su género. Lo que define que un videojuego pertenezca a un género u otro es **la manera en la que el jugador interactúa con el videojuego para cumplir los objetivos del mismo.** Por ejemplo, existen muchos videojuegos con ambientación ciberpunk[1], sin embargo, algunos de ellos son juegos de apuntar y disparar, otros son novelas visuales en las que simplemente seleccionamos diálogos y acciones en unas circunstancias dadas… Y no sólo eso, no solo hay videojuegos ciberpunks, es una temática utilizada en otras ramas artísticas como el cine o la literatura.

¿Qué sería el género del videojuego entonces? Precisamente lo hemos nombrado en el ejemplo del párrafo anterior. No es la temática ciberpunk, es **cómo interactuamos con ese contexto**, el juego es de acción, o de estrategia, o de deportes… Etc. Por tanto, lo que define el género de un videojuego, no es el contexto en el que se encuentra, ni la historia que relata, ni la ambientación sobre la que se construye. Son las herramientas que tenemos para interactuar con todos y cada uno de esos aspectos del videojuego.

Para clarificarlo podemos usar otro ejemplo: Un género de videojuegos conocido son los videojuegos de estrategia. En los que dirigimos a nuestras unidades para derrotar al ejército enemigo, eliminar a su comandante o cumplir cualquier otro tipo de misión militar. Sin embargo, existen videojuegos de estrategia medievales (p. ej. Age of Empires), otros ambientados en el futuro con invasiones alienígenas (p. ej. XCOM) u otro que nombraremos mucho a lo largo de esta memoria, una saga de videojuegos estratégicos con elementos fantásticos como magias, pegasos y dragones: Fire Emblem. Como podemos observar, estos 3 ejemplos tienen contextos históricos, circunstancias, historias, personajes, armamento, fechas de desarrollo y publicación… Completamente distinguidos entre ellos, y, sin embargo, la forma en

[1] Es un subgénero de la ciencia ficción que suele mostrar distopías en el futuro con dos características marcadas: bajos niveles de vida y tecnología muy desarrollada.

la que interactuamos con el videojuego y cumplimos los objetivos de este son muy similares, por tanto, podemos que todos ellos pertenecen a un mismo género: El estratégico.

No sólo eso, dentro de cada género de videojuegos, existen subgéneros (Vince, 2018). Por ejemplo, dentro del género de videojuegos de deportes podemos encontrar videojuegos de conducción, deportes de equipo, de lucha… Etc. También, existen videojuegos que pertenecen a más de un género, de hecho, esto es lo más normativo, es muy extraño encontrar un videojuego que pertenezca puramente a un solo género.

2.2 Género estratégico

Una vez definidos los géneros de videojuegos y en qué consisten, vamos a centrarnos en el que nos concierne: El género de estrategia. Durante toda la memoria hemos hablado y hablaremos, de dos géneros concretos de los cuáles hereda nuestro trabajo principalmente, pero técnicamente son **subgéneros** del género estratégico. Aun así, por comodidad para el lector, una vez hecha la aclaración y la explicación, seguiremos utilizando siempre el término género.

Estos dos subgéneros han sido mencionados anteriormente en la introducción y serán desarrollados detalladamente en comparación con nuestro trabajo en el apartado 3, sin embargo, hablaremos de ellos ahora para introducirlos. Ambos subgéneros pertenecen al género estratégico, así que tienen múltiples similitudes, pero a su vez, unas diferencias que otorgan una experiencia bien distinguida al jugador. Ellos son los géneros conocidos como **táctico y ajedrez automático.**

2.3 Estado del arte en el género estratégico

Existen muchos videojuegos estratégicos hoy en día con un impacto contundente en la industria, tanto a nivel tecnológico (la evolución de los videojuegos y de las técnicas empleadas para el desarrollo de los mismos), como económico. Por ejemplo, el último título de la saga Fire Emblem para Nintendo Switch[2], estuvo liderando las listas de ventas en Japón y Reino Unido con 2,29 millones de ventas en menos de 3 meses desde su salida (Aido, 2019). Este es un ejemplo de que los juegos de estrategia pueden tener un gran éxito en una videoconsola de carácter familiar como es la Nintendo Switch (o cualquier otro dispositivo de la compañía), pero ¿Qué hay del resto de plataformas entonces? Es bien sabido que los usuarios de ordenador son el público más numeroso y a la vez el más exigente en cuánto a videojuegos se refiere, pues bien, XCOM 2, un videojuego triple A[3], sobrepasó su millón de ventas en sus 6 primeros meses de vida (Sánchez, 2016). ¿Pero entonces solo funcionan los videojuegos de grandes compañías? No, Wargroove, otro juego muy similar a Fire Emblem, recuperó su coste de desarrollo en sólo 3 días gracias a sus ventas, a pesar de ser un indie[4] (Guadalupe, 2019).

---

2 Última consola de la compañía Nintendo.

3 Se denominan AAA (o triple A) a los videojuegos desarrollados por estudios prestigiosos y con gran alcance económico.

4 Se conocen como *indies* a los videojuegos desarrollados por pequeños estudios con un alcance económico muy reducido.

Estos videojuegos que hemos utilizado como ejemplo pertenecen al género **táctico,** y es innegable que por los números que están generando en distintas plataformas y teniendo en cuenta que pueden funcionar vengan de estudios prestigiosos o pequeños, es un terreno que vale la pena explorar, entonces ¿Qué hay del género *autochess*? ¿Promete tanto como el género táctico? ¿Qué números tenemos sobre él? Os adelantamos que, es cuánto menos, sorprendente.

Hablaremos detalladamente de él en el apartado 3.1 pero el juego insignia del género *autochess*, Teamfight Tactics, registró en marzo de 2020 más de 33 millones de jugadores (González, 2019), así que, podemos afirmar que sí, es otro género con un terrero bastante prometedor. El género está teniendo un gran alcance en el sector de los videojuegos y está siendo recibido con los brazos abiertos por los usuarios, eso sí, los detalles mecánicos y técnicos del mismo serán explicados en el apartado 3.1 y 3.2. Existen un montón de aplicaciones móviles intentando emular, copiar y reinventar este género, pero no están terminando de cuajar y de conseguir buenos números… ¿A qué puede deberse esto?

Los juegos *autochess* se caracterizan por restringirnos a la gestión de un ejército que pelea automáticamente contra el de los rivales a lo largo de las rondas en partidas de 8 jugadores, partidas que duran, alrededor de treinta o cuarenta minutos. El propio Teamfight Tactics, tiene su versión en dispositivo móvil, pero… ¿Alguien que no sea un amante ya del género va a estar cuarenta minutos jugando la misma partida sin cambiar de aplicación? No es así, su público en ordenador es muchísimo más grande que su público en móvil: De las 33 millones de descargas datadas en marzo de 2020, que ahora habrán crecido (la empresa no ha declarado datos oficiales más actualizados), sólo 5 millones (como podemos observar en la figura 2.1), en julio de 2020, son en teléfono móvil: **Sólo un 15% de sus descargas son en teléfonos móviles** en el mejor de los casos, teniendo en cuenta su crecimiento más acelerado en ordenador, la diferencia real debería ser mayor y por tanto, el porcentaje de descargas móviles, más pequeño. Podemos deducir que es una fórmula y un género que está funcionando muchísimo mejor en ordenador, y que, en telefonía móvil, está pasando sin pena ni gloria.

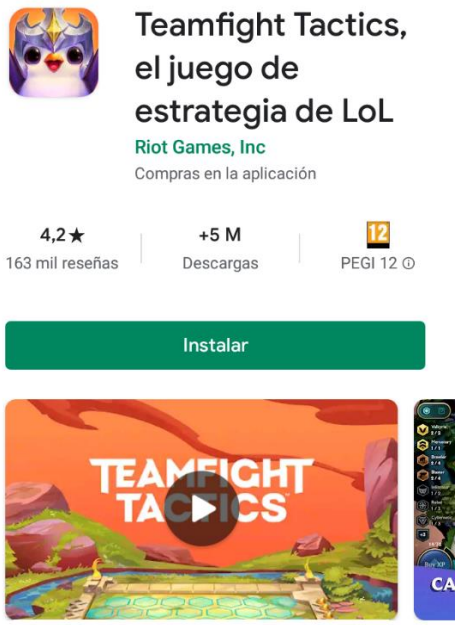

Figura 2.1: Página de Teamfight Tactics en la Play Store de Android

Tanto la propia versión de Teamfight Tactics como los intentos de otros estudios de emulación del mismo tienen estas mismas características, y todavía, no ha habido nadie que plantee una misma mecánica de juego, pero en un escenario de 1 contra 1 con el fin de reducir el tiempo de partida a 5 o 10 minutos. Y ese creemos que es el principal problema: El tiempo de partida. Pasarse cuarenta minutos en la misma partida en un teléfono móvil… No le gusta a casi nadie. No nos guste que nos gaste tanta batería, ni que gaste demasiados datos de forma tan continuada, ni que no nos permita jugar una partida en nuestro trayecto en metro o autobús… Etc.

2.4 El nuevo posible estado del arte

Pues eso es lo que vamos a cambiar nosotros, utilizar alguna característica (se detallará en el apartado 3.2) de los videojuegos de género táctico, para crear un juego estilo *autochess*, con algunas modificaciones que lo harán ideal para un público sediento de un videojuego de este estilo que, a la vez, cumpla con las exigencias de los jugadores de teléfono móvil, y, la primera de esas exigencias es el tiempo de juego, que las partidas duren entre 5 y 10 minutos máximo, es uno de nuestros principales objetivos.

Reducir el tiempo de partida es algo que tiene diversas implicaciones que serán descritas en el apartado 3.2, pero a groso modo, nuestro juego dejará de ser un *autochess*, ya que las peleas no serán automáticas, sino manuales, además de como se ha mencionado antes, plantear el videojuego como un 1 contra 1. Otra parte importante para poder pulir el producto es simplificar lo máximo posible las mecánicas y la densidad del juego, con el fin de reducir el tiempo de partida, pero en su justa medida (esto va a ser la parte más complicada del desarrollo), porque si simplificáramos demasiado, se volvería trivial, aburrido y poco desafiante.

Esta filosofía está muy sustentada en que los videojuegos de jugador contra jugador en dispositivos móviles con una duración corta de partidas, están teniendo un éxito rotundo entre sus usuarios, sólo que hay echar un ojo a los números de Brawl Stars, por ejemplo, un juego de la compañía Supercell que alcanzó 75 millones de descargas en sus 3 primeros meses de vida (Guillem, 2019).

Esto no es casualidad, en la sociedad actual prácticamente todos somos usuarios de móvil, y lo que queremos es fugacidad, cambiar de una aplicación a otra de forma rápida, en cualquier lugar en el que nos encontremos. El hecho de plantearle a un jugador que pueda jugar un par de partidas de nuestro juego en su trayecto de metro diario o en su descanso del trabajo, es una idea que se vende ella sola.

Ese es nuestro objetivo, nuestra meta. Utilizar como arma el hambre de una comunidad que todavía no ha probado un plato que combina dos de sus sabores favoritos: Que puedan jugar su *autochess* estratégico contra rivales mediante conexión en línea y que puedan hacerlo en cualquier momento y lugar.

3. Géneros heredados

El videojuego hereda dos géneros principalmente, que, sumado con otros aspectos añadidos y nunca vistos con este género, definimos un posible nuevo estado del arte. Los géneros son: el *Tactics* (o Táctico) y el *Autochess* (o Ajedrez Automático).

El género **táctico**[5] nace de la unión de otros dos muy conocidos: Los juegos clásicos de rol y los juegos de estrategia. Su nacimiento podría datarse en el año 1990 dónde el primer Fire Emblem aparece, aunque tan sólo en Japón para NES[6], pero podríamos decir que esta saga ha sido la precursora de este género concreto de videojuegos del que han bebido centenas de títulos hasta el día de hoy. La mecánica de juego es muy sencilla: Posees unidades con nombre e historia formando ejércitos para pelear en batallas por turnos contra una IA en una cuadrícula, como se puede observar en la Figura 3.1. En tu turno, puedes realizar acciones con todas tus unidades y luego, la IA hace lo propio.

Figura 3.1: Fire Emblem Awakening para 3DS

El género **ajedrez automático** (conocido como *autochess*), es extremadamente nuevo, ya que nació en enero de 2019 como un *mod*[7] de DotA 2, llamado Dota Auto Chess, un videojuego de la conocida empresa Valve. Este juego también hereda aspectos del género táctico ya que peleamos contra otros ejércitos en una cuadrícula, sin embargo, en el género *Autochess*, nos centramos estrictamente en la gestión y mejora de nuestro ejército, ya que nuestras unidades pelean automáticamente a lo largo de varias rondas de forma recurrente contra los ejércitos de otros jugadores.

---

5 El género Táctico es conocido generalmente como *Tactics*, aunque esto solo es una abreviación del término real: *Tactical role-playing game*, o videojuego de rol táctico, abreviado como TRPG.

6 *Nintendo Entretainment System.* La primera consola de Nintendo.

7 Se conoce así a una modificación desarrollada en un videojuego de forma que afecta a su jugabilidad, añade mecánicas extra, un nuevo modo de juego etc. Utilizando el propio juego (o su motor) de manera que conserve la esencia del mismo. Normalmente, estos *mods* son creados por usuarios de la comunidad del juego y no por los desarrolladores originales.

Figura 3.2: Dota Auto Chess.

3.1 Instancias de los géneros

Es importante aclarar que, para explicar de dónde venimos y a dónde vamos, necesitamos videojuegos que valgan como ejemplos o representantes de cada uno de los géneros anteriormente nombrados. Cuando hablemos de *Tactics* de forma concreta, estaremos hablando del anteriormente mencionado Fire Emblem, y cuando hablemos de a*utochess*, estaremos hablando del juego de Riot Games[8]: Teamfight Tactics. A pesar de que el género *autochess* nació con DotA 2 como señalamos antes, Riot Games consiguió perfeccionar el género y atraer una enorme cantidad de jugadores, para ser más concretos: "Más de 33 millones de jugadores en marzo de 2020" (González, 2019), así que, a día de hoy, nos sirve mucho mejor como referente para el género.

3.1.1. Fire Emblem

Una saga de videojuegos de un jugador que plantea conflictos bélicos en contextos de medievo y magia con los que contar una historia con planteamiento, nudo y desenlace. Lo que diferencia a Fire Emblem del resto de videojuegos del género táctico, no es sólo que este sea su precursor, fue innovador desde el principio planteando un sistema de juego en el que, si uno de los personajes es derrotado en combate y muere, no puede ser recuperado y no se dispondrá de él en el resto del juego. Teniendo en cuenta que los juegos suelen durar alrededor de cuarenta horas, es bastante determinante. Nuestros personajes poseen nombre e historia, son los protagonistas de un conflicto único, con los que iremos avanzando a través de varios niveles en los que nos enfrentaremos a ejércitos enemigos que intentarán frenarnos o tomaremos desvíos para salvar a un soldado acorralado que se unirá a nuestras filas.

La mecánica principal es, como se ha mencionado de forma sencilla anteriormente, utilizar a nuestro ejército de la mejor forma posible para derrotar al ejército enemigo o su comandante impidiendo que ninguna de nuestras unidades caiga en combate. Durante nuestro

8 Empresa insignia en el sector del videojuego. Es la creadora del famoso League of Legends, siendo Teamfight Tactics un juego de estilo *autochess*, que utiliza a los icónicos personajes de League of Legends como unidades del juego.

turno podremos realizar acciones con todas nuestras unidades en el campo de batalla como se muestra en la Figura 3.1 y cuando acabe, la IA hará lo mismo con las suyas. Además, cada vez que atacamos o somos atacados, el plano del juego pasa de ser el de la Figura 3.1, que nos muestra el campo de batalla completo en vista isométrica, a un duelo entre los dos personajes enfrentados como se muestra en la Figura 3.3 o 3.6.

Figura 3.3: Combate en Fire Emblem: Blazing Sword

Fire Emblem incluye desde el principio apartados secundarios de jugabilidad que influyen a lo largo de todos los niveles (con el tiempo cada vez más evolucionados y sofisticados), como por ejemplo que nuestros personajes puedan subir su nivel de relación al pelear juntos, y que la siguiente vez que combatan contra los enemigos si se encuentran en casillas adyacentes, aumenten sus estadísticas. O que se enamoren y tengan un hijo que herede las habilidades de sus progenitores y podamos usarlo para combatir en el futuro.

3.1.2. Teamfight Tactics

Videojuego de Riot Games lanzado oficialmente el 26 de julio de 2019, el cual ofrece una experiencia multijugador a sus usuarios. Curiosamente, este juego también hereda aspectos del propio género táctico, ya que las unidades pelean en una cuadrícula (aunque en el caso de Teamfight Tactics son casillas hexagonales), y la parte más importante es la estrategia que utilicemos para salir victoriosos.

Avanzamos en la partida a través de un sistema de rondas. En cada una de ellas tenemos una tienda en la que aparecen 5 unidades aleatorias y podemos decidir comprar (o no) cualquiera de ellas si tenemos el oro suficiente. Los 8 jugadores empiezan la partida con cien puntos de vida, cada vez que perdamos una ronda nos restarán unos pocos. Si nos quedamos a 0, seremos eliminados de la partida. Cabe destacar que al final de cada ronda nuestras unidades recuperarán todos sus puntos de vida.

Su mecánica principal es conseguir campeones con sinergia con los que formar nuestro ejército. Cada campeón tiene un origen y una clase (Ninja-Asesino, por ejemplo), si juntamos varias unidades con el mismo origen y/o clase, obtendremos bonificaciones para nuestro ejército

(cuando juntamos 3 asesinos tienen índice extra de golpe crítico, por ejemplo). Además, si juntamos varias unidades iguales, se juntarán haciendo una unidad más fuerte.

Figura 3.4: Combate de Teamfight Tactics.

Todos estos aspectos pueden apreciarse en la Figura 3.4. En la parte inferior tenemos la tienda de campeones de la ronda actual, además de nuestro nivel, ya que, vamos ganando experiencia a lo largo de las rondas o invertimos dinero para subir más rápido: Nuestro nivel determina el número de unidades que podemos tener en el campo de batalla (si somos nivel 8, podremos tener 8 unidades). En la parte izquierda se muestran las sinergias que tenemos activas en nuestro ejército gracias a juntar varias unidades con la misma clase u origen, y a la derecha los jugadores y su nivel de vida. En el centro podemos observar como las unidades están peleando automáticamente, las que tienen el marco plateado en su barra de vida son de nivel 2, por haber juntado 3 unidades iguales. Y las que tienen el marco dorado son unidades de nivel 3, por haber juntado 3 unidades de nivel 2. Estás unidades mejoradas tienen características mucho mejores y, por tanto, son más fuertes. Además, a lo largo de las rondas iremos obteniendo objetos que podemos juntar y otorgar a nuestras unidades para otorgarles bonificaciones especiales.

Existen un gran número de campeones, sinergias y clases que hacen que cada partida sea completamente distinta debido al gran número de combinaciones, que unidades vayamos comprando y que objetos vayamos formando, además de tener que utilizar la estrategia para contrarrestar la estrategia de nuestros rivales.

Es complicado resumir por escrito la compleja jugabilidad y procesos de una partida de Teamfight Tactics, pero el anuncio oficial del juego lo consigue en tan sólo dos minutos: https://www.youtube.com/watch?v=vupLCjDwCKk.

Estos son los aspectos principales de los juegos insignia de ambos géneros para que pueda entenderse qué vamos a construir nosotros, qué vamos a coger de cada uno de ellos y cómo vamos a moldear cada una de las piezas que extraigamos. Para ello, entramos más en detalle en el siguiente apartado, en el que se contrastan los aspectos de cada juego respecto a

nuestra propuesta, entrando más en detalle sobre alguna de las pinceladas ya dadas en los apartados anteriores.

3.2 Diferencias y nuevos aspectos. Análisis comparativo.

De forma directa, podría decirse que queremos extraer los siguientes aspectos de cada uno de los géneros para adaptarlos a nuestras necesidades:

- Del género **táctico** utilizaremos la cuadrícula como campo de batalla y el hecho de que controlemos el turno de cada una de las unidades de forma manual (por tanto, no será automático como los *autochess* originales).

- Del género ***autochess*** aprovecharemos más apartados, ya que nos aproximan más para un producto de jugador contra jugador, así como: Comprar unidades entre rondas, que nuestro ejército pueda contar con un mayor número de unidades con el paso de las rondas, que nos enfrentemos a otro jugador y que nuestras unidades tengan sinergia entre ellas si juntamos varias del mismo tipo.

Para que pueda definirse mejor el proyecto, se muestran las tablas 3.1 y 3.2, que respectivamente, sintetizan y contrastan características de jugabilidad y técnicas de las instancias de los géneros anteriormente mencionadas con nuestro juego.

Tabla 3.1

Los aspectos principales de jugabilidad de los géneros y de nuestra propuesta.

	Contrincante	**Duración (min)**	**Combate**	**Gestión de ejército**	**Continuidad entre partidas**
Táctico	IA	30 – 60	Manual	No	Sí
Autochess	7 jugadores	25 – 40	Automático	Sí	No
Nuestro juego	1 jugador	5 – 10	Manual	Sí	No

1. **Contrincante.** Referente a quién nos enfrentamos y al número de rivales.
   - **1.1. Táctico.** Cuando el género nació, no existían ni mucho menos los videojuegos con modo multijugador en línea, así que, el rival siempre era la IA controlando a todo su ejército en su turno.
   - **1.2. *Autochess.*** Las partidas son arenas de 8 jugadores, en las que, en cada una de las rondas, nos enfrentamos a uno de ellos aleatoriamente. Cada vez que la vida de un jugador llega a 0 es eliminado de la sala, por tanto, también es un juego *Battle Royale*[9] dónde el último en quedar con vida es el vencedor.

[9] Un género de videojuegos que se ha hecho muy popular en los últimos tres años que destaca por hacer que el vencedor de la partida sea el último que quede. Cuando alguien es eliminado es libre de abandonar el encuentro y buscar otro, lo que lo hace muy dinámico. El más conocido y que más impacto ha tenido es Fortnite.

1.3. **Nuestro juego.** Nos enfrentaremos a rivales de carne y hueso en lugar de IAs, pero serán partidas de 1 contra 1. El objetivo principal de este cambio es la aceleración del tiempo de partida.

2. **Duración.** La duración aproximada en minutos de cada partida.
   2.1. **Táctico.** Las partidas son misiones que vamos completando, y en cada uno de nuestros turnos, tenemos que hacer acciones con todas y cada una de nuestras unidades (pueden llegar a haber más de quince), por tanto, el jugador necesita realizar bastantes decisiones e invertir bastante tiempo en pensar cuál es la jugada óptima en cada turno con un número alto de unidades aliadas y enemigas que hacen más compleja la jugada.
   2.2. ***Autochess.*** Las rondas tienen un tiempo determinado de treinta segundos de gestión (como podemos observar en la Figura 3.5) más el tiempo en el que pelean las unidades automáticamente, que suelen ser aproximadamente otros treinta segundos, así pues, podemos asumir que **una ronda completa dura 1 minuto.** Depende del nivel de los jugadores y las decisiones que tomen con sus composiciones, las partidas pueden tener más o menos rondas, pero normalmente suelen ser aproximadamente unas treinta. Estás treinta rondas (algunas arriba, algunas abajo) de aproximadamente 1 minuto, sumado a las rondas extra de carrusel, nos da duración de partida estipulada en la tabla. Sin embargo, cabe recordar que al ser un estilo de juego *Battle Royale*, si somos eliminados de los primeros, habremos jugado unos quince minutos aproximadamente y podremos abandonar la partida para entrar a una nueva directamente, sin necesidad de esperar a que nosotros veamos al vencedor.

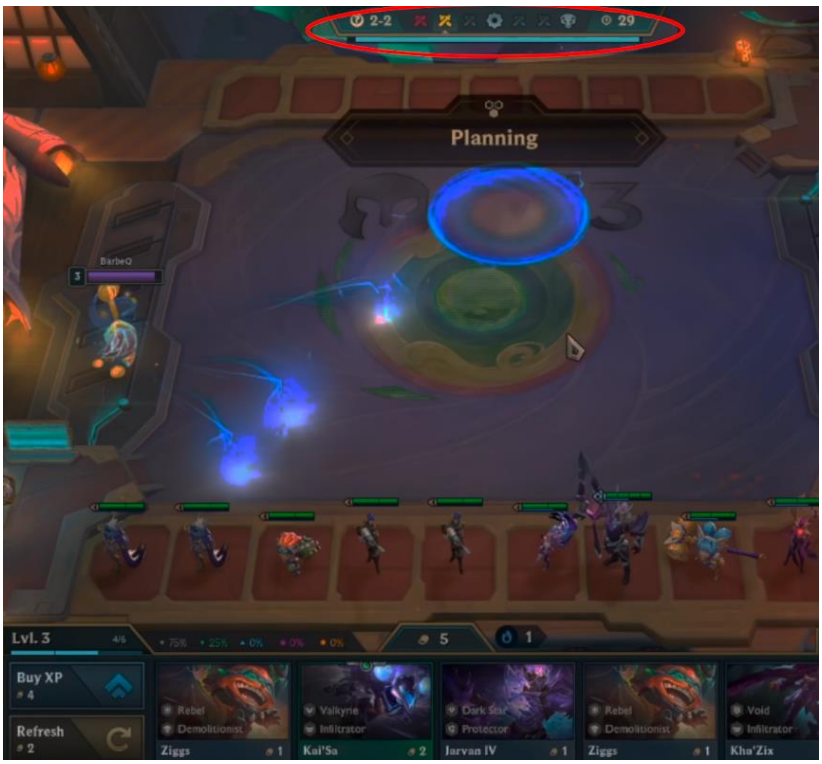

Figura 3.5: Muestra de tiempo de ronda en Teamfight Tactics.

2.3. **Nuestro juego.** Con el fin de que el juego sea atractivo e innovador para el público de dispositivos móviles, uno de los aspectos más importantes es reducir el tiempo de partida. Los ejércitos van a tener menos unidades que en un *Autochess*, y, por ende,

muchas menos que en un videojuego táctico convencional. Se dispondrán de veinte segundos de gestión en cada ronda, pero la fase de combate será más larga que en Teamfight Tactics, ya que el combate es manual y no automático. Por tanto, para poder reducir mucho el tiempo de partida las decisiones tomadas son: Reducir el número de contrincantes a 1, lo cual ya reduce inherentemente el número de rondas, y además construiremos el juego de tal manera a nivel mecánico y jugable (se describirá más adelante) que nos permitirá reducir aún más el número de rondas necesarias para completar una partida.

3. **Combate.** El estilo de juego de cada uno de los géneros. Qué podemos y decidimos hacer con nuestras unidades para enfrentar al rival.
   - **3.1. Táctico.** El campo de batalla es una cuadrícula enorme con casillas (véase la Figura 3.1) que añaden modificadores de terreno (en los árboles tenemos más evasión, los caballos no pueden moverse por montaña… Etc.) En nuestro turno podemos hacer una acción con cada una de las unidades de nuestro ejército de forma manual. Por ejemplo, podemos mover una unidad y que ataque a otra adyacente enemiga, que otra unidad se ponga al lado de otra aliada y le intercambie un objeto, que nuestra unidad sanadora cure con magia a distancia a otra de nuestras unidades… Etc. Una vez hemos hecho una acción con cada unidad, nuestro turno termina y la IA enemiga hace lo mismo con todas sus unidades. También, podemos terminar nuestro turno a placer sin realizar acciones con todas y cada una de nuestras unidades.
   - **3.2. *Autochess.*** Como el género indica, el combate es automático. Nosotros solo controlamos el posicionamiento de nuestras unidades antes de que comience una ronda de batalla con otro jugador, una vez la ronda empiece, las unidades atacan por sí mismas al enemigo más cercano, habitualmente. Existen excepciones como por ejemplo: Los asesinos saltan a la retaguardia enemiga, los francotiradores atacan a la unidad más lejana… Etc.
   - **3.3. Nuestro juego.** Mucho más parecido al género táctico, nosotros haremos acciones manuales con nuestras unidades, pero intercalando turnos con el enemigo, es decir, primero el jugador 1 hará una acción con una de sus unidades y después el jugador 2 podrá responder con una de las suyas, y así sucesivamente. En cada ronda, uno de los jugadores será el que tenga el primer turno. Este es un apartado muy importante, ya que estamos eliminando el combate automático insignia de los juegos *autochess*, pero si queremos hacer partidas tan cortas es muy importante eliminar todo el RNG[10] posible. En partidas largas como el Teamfight Tactics, el RNG siempre tiende a estabilizar las partidas (tengamos peor o mejor suerte), sin embargo, en partidas tan cortas como las que planteamos nosotros, la aparición de un RNG muy determinante puede ser frustrante para el jugador, ya que puede generar partidas con escenarios muy extremos e irreales. Si el jugador controla el combate de forma manual, nunca existirá la excusa de que una unidad (IA) ha atacado de una forma cuestionable o ha seleccionado a una unidad claramente errónea para atacar.

---

10 Hace referencia a *Random Number Generation*, y cubre los apartados de aleatoriedad de los videojuegos para dotarles de diferentes salidas ante entradas similares, con lo cuál se extiende el tiempo de jugabilidad y añade un factor que, al no poder ser totalmente controlado por el jugador, le mantiene en intentos y repetición constante para seguir considerando y disfrutando de distintos escenarios y posibilidades.

4. **Gestión de ejército.** Lo que podemos hacer con nuestras unidades fuera del campo de batalla: Cómo podríamos mejorarlas, si podemos otorgarles equipamiento, cuánto se centra el juego en cuestión en este apartado… Etc.
   - **4.1. Táctico.** La saga Fire Emblem posee una gran cantidad de aspectos de jugabilidad fuera del campo de batalla pero que no nos sirven en nuestra propuesta, ya que, son características inherentes a que el juego tenga una historia y argumentos contextualizados que le dan continuidad entre misiones (esto será descrito más detalladamente en el apartado 5.1 de esta lista).
   - **4.2.** ***Autochess.*** En este género la gestión lo es todo, como las batallas son automáticas, toda nuestra participación ocurre antes de las mismas. Tenemos que decidir en cada ronda: Qué campeones comprar en la tienda, cuánto dinero ahorrar, qué unidades sacar al campo de batalla, cuáles guardar en el banquillo, cómo juntar nuestros objetos, a quién se los ponemos y cómo posicionamos nuestras unidades. Estas acciones concretas han sido explicadas más detalladamente en el apartado de Teamfight Tactics.
   - **4.3. Nuestro juego.** Aquí también es dónde marcamos la diferencia, ya que, a pesar de tener batallas manuales, también tenemos gestión de ejército entre rondas. La gestión es mucho más reducida de lo que es en Teamfight Tactics. La tienda es de 3 campeones por ronda y no podemos gastar dinero para obtener otra, no existen objetos que podamos juntar o utilizar en nuestros personajes, así que, nos quedamos con la parte de formar unidades de nivel 2 (no existirán las de nivel 3), y gestionar nuestro campo de batalla y banquillo.

5. **Continuidad entre partidas.** Hace referencia a qué impacto tiene una partida sobre la siguiente a nivel mecánico y jugable (se obvia que el jugador obtiene experiencia de juego y tiene más conocimiento para la siguiente).
   - **5.1. Táctico.** De sus apartados clave, debido a que, la saga Fire Emblem fuera del campo de batalla, ofrece un montón de opciones para gestionar la relación entre nuestras unidades, las cuáles pueden ganar amistad y pelear mejor juntas, o enamorarse, tener hijos y en el futuro usar a dichos hijos como nuevas unidades con habilidades heredadas de sus padres en el campo de batalla. Tenemos otros apartados menores como: Alimentar a nuestras unidades para mejorar características para la siguiente batalla, reforzar armas en el herrero para mejorarlas… Etc.
   - **5.2.** ***Autochess.*** No existe continuidad entre partidas, todo lo que conseguimos en una partida se pierde en la siguiente. Volvemos a empezar siendo nivel 1, sin unidades, sin objetos… Y tampoco existe una línea argumental en el juego.
   - **5.3. Nuestro juego.** Al igual que el género *Autochess*, al tener como objetivo partidas rápidas y buscar el enfrentamiento con otros jugadores, en cada partida empezamos desde cero en igualdad de condiciones con nuestro rival.

Una vez descritas las similitudes y diferencias a nivel de jugabilidad y cómo creamos e introducimos nuevos aspectos para mejorar la experiencia del jugador al que nos queremos acercar, tenemos que contrastar los aspectos técnicos de nuestra propuesta:

Tabla 3.2

Los aspectos técnicos de los géneros y de nuestra propuesta.

	Juego en red	**2D/3D**	**Plataforma destino**	**Control**	**Optimización**
Táctico	No	2D	Consola	Botonera	Alta
Autochess	Sí	3D	PC	Teclado y ratón	Media
Nuestro juego	Sí	2D	Móvil	Táctil	Alta

1. **Juego en red.** Si el género o videojuego posee modo multijugador en línea.
   - **1.1. Táctico.** Los últimos Fire Emblem y juegos tácticos, poseen funcionalidades en línea como eventos en fechas especiales, compra y descarga de *DLCs*[11], e incluso combates en línea… Pero no a tiempo real. Nos enfrentamos al ejército de otro jugador controlado por la IA. De todos modos, esto es el resultado del género táctico adaptándose a los nuevos tiempos, siguen siendo videojuegos cuya esencia es puramente de un jugador.
   - **1.2. *Autochess.*** No tendría sentido sin juego en red, ya que, está diseñado para enfrentar a otros jugadores desde su esencia.
   - **1.3. Nuestro juego.** Al igual que los juegos *autochess,* el objetivo es que puedan enfrentarse dos jugadores en línea.

2. **2D/3D.** Las dimensiones que posee el juego a todos los efectos, tanto artísticamente como mecánicamente.
   - **2.1. Táctico.** Los hay de todo tipo, pero en referencia a Fire Emblem, siempre tenemos una vista isométrica del campo de batalla con modelos en 2D. En los títulos de la saga más modernos, ha habido introducción al 3D, por ejemplo, cuando dos unidades se enfrentan (véase la Figura 3.6). Podría decirse que, en este apartado, todos empezaron siendo 2D y han ido evolucionando en sintonía con el avance tecnológico en el sector del videojuego.

[11] Hace referencia a *Downloadable Content.* Contenido descargable en español. Es un término muy utilizado en videojuegos y una estrategia cada vez más popular y explotada por las empresas. Dónde después de vender su juego base, diseñan nuevo contenido y lo venden digitalmente como una ampliación del mismo.

Figura 3.6: Escena de combate 3D en Fire Emblem Awakening.

2.2. ***Autochess.*** Puramente 3D. La empresa Riot Games utiliza sus modelos 3D ya modelados en League of Legends, como ya se especificó anteriormente, los personajes de Teamfight Tactics y League of Legends son los mismos, aunque las jugabilidades sean completamente diferentes. En el resto de juegos estilo *autochess*, también encontramos modelos 3D, ya que como comentamos anteriormente, es un género que nació en 2019 y como *mod* de un juego ya creado, por tanto, se reutilizan los modelos 3D. El resto de juegos diseñados desde cero, han mantenido este aspecto.

2.3. Nuestro juego. 2D. Como el objetivo de nuestro proyecto es la funcionalidad y el análisis del estado del arte con una nueva declaración del mismo, no podemos invertir recursos y tiempo en hacer nuestro producto puntero en el apartado gráfico. Tampoco se poseen los conocimientos de diseño y arte necesarios, este va a ser uno de los aspectos más restrictivos de nuestra propuesta.

3. Plataforma destino. A que dispositivo está dirigido el producto esencialmente.

3.1. Táctico. Originalmente siempre ha estado destinado a consola, ya que ha sido un género que nació en el 1990, era la única plataforma viable. Hoy en día podemos encontrar juegos tácticos en cualquier plataforma.

3.2. ***Autochess.*** Nacido en ordenador y como principal fuente de jugadores en el mismo, el propio Teamfight Tactics apareció en plataformas móviles el 19 de marzo de 2020 (Mogroviejo, 2020, 23 de marzo) después de su gran triunfo en PC.

3.3. Nuestro juego. A pesar de que existen juegos tácticos y *autochess* para móvil, los primeros están orientados a la experiencia de un solo jugador y los segundos tienen el enfrentamiento en línea, pero manteniendo sus partidas de treinta minutos o más, algo muy lejano a lo que el público de dispositivos móviles está acostumbrado. Por tanto, ya que nuestro juego tiene como destino los teléfonos móviles, priorizamos reducir al máximo el tiempo de partida. Coloquialmente, podría decirse que buscamos que cualquiera pueda *echarse una partida en el metro* o *en el descanso del trabajo*. Esta es una fórmula utilizada por otros juegos de móvil (con otro género), como Clash Royale y Brawl Stars de la empresa Supercell.

4. Control. La forma con la que interactuamos con el dispositivo en cuestión para jugar.

4.1. Táctico. Al ser de consola siempre se ha usado el mando de las consolas de Nintendo o la botonera de los dispositivos portátiles.

4.2. ***Autochess.*** Utilizamos ratón esencialmente, clicando en las unidades que queremos comprar, arrastrándolas para dejarlas en la posición que queremos… Etc. Con el teclado podemos utilizar un reducido número de teclas que nos sirven de atajo para que nuestras acciones sean más rápidas.

4.3. **Nuestro juego.** En dispositivo móvil podemos simular de manera casi exacta el control por ratón por pantalla táctil, además de ser el control más intuitivo de todos.

5. **Optimización.** La**s** especificaciones técnicas de desarrollo y qué impacto tienen en base a que plataforma está destinado el producto.

5.1. **Táctico.** Teniendo como referencia Fire Emblem, marcamos un nivel de optimización alto ya que, todo lo que no esté destinado a computador, tiene un *hardware* más concreto y restrictivo al que adaptarse para que el producto funcione de forma correcta.

5.2. ***Autochess.*** Al estar destinado para computador, hay mucho más margen para no darle excesiva prioridad en este aspecto, ya que como sabemos los ordenadores son los dispositivos más potentes por norma general.

5.3. **Nuestro juego.** Uno de los objetivos es poder llegar al mayor número de jugadores posible, por tanto, debemos desarrollar el proyecto de manera que pueda funcionar en el mayor número de dispositivos móviles posible.

3.3 Conclusión sobre el análisis comparativo

Resumiendo, podemos decir que nuestra propuesta será un juego con la esencia del Teamfight Tactics, en el cuál avanzaremos a través de las rondas comprando unidades, subiendo de nivel para poder tener cada vez más de ellas y juntando las que sean de la misma clase para obtener bonificaciones adicionales, con la gran diferencia de que nos enfrentaremos a un solo jugador, en un número de rondas mucho más reducido con el fin de reducir el tiempo de partida y hacerlo más frenético, dinámico y adecuado para el público de dispositivos móviles, con la clave de los juegos tácticos tradicionales: Nosotros controlaremos unidad por unidad su acción de forma manual.

4. Desarrollo del videojuego

En este apartado encontraremos toda la información detallada sobre todos los aspectos específicos del proyecto, a nivel técnico y de desarrollo. Se expondrán los objetivos de diseño a alcanzar, así como los límites del desarrollo, y por supuesto, la explicación del funcionamiento de la aplicación y cómo está construida a nivel de programación y diseño dentro del motor Unity.

4.1 Alcance del proyecto

Para hablar del alcance del proyecto, necesitamos exponer y detallar dos características clave del desarrollo: **Los objetivos y límites.** A pesar de que vayamos a listar y detallar las características de cada uno de estos dos aspectos por separado, es importante mencionar que están estrechamente relacionados entre sí: Un objetivo puede poner límites, así como un límite puede condicionar o delimitar objetivos.

4.1.1 Objetivos

- Diseñar un control del juego íntegramente con pantalla táctil.
- Poder interactuar con la aplicación mostrando las principales mecánicas de este género de videojuegos.
- Completar una primera versión de prueba en la que con un par de objetos (personajes dentro del juego) podamos observar e interactuar con el mayor número de funcionalidades posibles.

4.1.2 Límites

- El diseño gráfico. Estamos obligados a utilizar modelos 2D que pertenecen a otros videojuegos o marcas para representar a los personajes ya que no disponemos de un diseñador gráfico. Este es el principal aspecto que impide su publicación en la Play Store de Android.
- Las funcionalidades de red. A pesar de que el juego está destinado para ser multijugador en línea en dispositivos móviles, es una tarea difícil de experimentar en el proyecto, pues idealmente, el anfitrión de las partidas debería ser un servidor (lo cual tiene un coste mensual) y no los usuarios.
- Escalabilidad. Consideramos más oportuno diseñar más mecánicas y resolver más problemas que añadir más y más personajes y/o bonificaciones de grupo, lo cual podría añadirse gradualmente una vez las mecánicas base estén bien construidas.

4.2 Funcionamiento ideal de la aplicación

Aquí intentaremos explicar y desarrollar el funcionamiento ideal de la aplicación de la manera más concisa y directa posible, antes de entrar en nuestro diseño técnico para intentar acercarnos lo máximo posible a este funcionamiento.

La partida empezaría con 3 rondas de rápidas en las que solo se puede comprar y no hay fase de enfrentamiento. Se generarían 3 tiendas y el usuario compraría lo que considere oportuno con una pequeña cantidad inicial de oro, así tenemos unas pocas unidades con las que empezar la partida, ya que si no podría ser un comienzo muy lento y aburrido. Empezaríamos siendo nivel 3 (el nivel define cuántas unidades puede haber dentro de la zona de juego a la vez), y este aumentaría con el paso de cada ronda, siendo 8 el máximo. Empezaríamos la primera fase de gestión dónde movemos nuestras unidades y las colocamos dónde creamos oportunas para preparar la fase de enfrentamiento. Cuando esta comienza, cada vez, uno de los dos usuarios tendrá el turno inicial. Empezará un jugador moviendo y atacando con una de sus unidades, y luego su rival hará lo mismo, así sucesivamente hasta que: O todas las unidades de un bando mueran, o todas las unidades de ambos bandos hayan realizado una acción cada una. Esta decisión de finalizar la ronda aún no ha sido estipulada ya que idealmente escogeríamos la primera por el bien del juego, pero si las partidas más largas de lo que esperamos, optaríamos por la segunda opción para acelerar el juego. Una vez acabara esta fase, volveríamos de nuevo a la siguiente fase de gestión, con un nivel más, una nueva tienda generada aleatoriamente y después, otra fase de enfrentamiento. El jugador que más rondas gane después de disputar la ronda en la que los jugadores son nivel 8, sería el vencedor.

4.3 Especificaciones técnicas de diseño y programación

En este apartado abordaremos todas las cuestiones, problemas y su correspondiente resolución a nivel de diseño y programación con el software Unity. Este motor de videojuegos nos permite escribir nuestros scripts en dos lenguajes de programación: JavaScript y C#, eligiendo el segundo para el desarrollo de nuestro proyecto. Cabe destacar que la aplicación está específicamente diseñada para ser controlada mediante entada táctil en nuestro teléfono, por ende, el diseño ha debido de ser focalizado alrededor de esta característica, no sólo para el control, si no para manejar los eventos y consecuencias una vez ocurran dichas entradas por parte del usuario.

Antes de comenzar con los detalles, hay que aclarar que en Unity los scripts que diseñemos tienen dos funciones clave: *start* y *update*. La primera ejecuta su código cuando el objeto en el que se encuentra el *script* se inicializa, y la segunda, se activa con cada marco (*frame*). Especialmente, y como suele ser común, haremos mucho uso de la segunda (gran parte de nuestro código siempre se encontrará dentro de esta función), ya que para manejar los eventos y actualizar el resultado de los mismos en tiempo real, es ideal.

4.3.1. Control táctil

Para recoger la entrada táctil en nuestro teléfono necesitamos dos cosas: El toque que queremos recoger, y dónde se encuentra el mismo. Para ello hacemos uso del paquete *Touch* de Unity. Podemos registrar qué número de toque queremos recoger (podríamos utilizar mecánicas en las que se usen varios dedos), ya que cada vez que un dedo toca la pantalla, se le asigna un ID (0, 1, 2, 3…), normalmente hasta 5, aunque depende del dispositivo. Como a nosotros nos es irrelevante el número de dedos que use el usuario, simplemente nos interesa que exista algún toque para recogerlo, así que usamos el atributo *count* para comprobar que es mayor que 0 y entonces, al saber que existe algún toque, realizar las acciones pertinentes. Lo segundo, es saber dónde se encuentra, pero necesitamos convertir el lugar del toque en la pantalla del dispositivo, en una posición manejable en nuestro escenario, ya que ni todas las pantallas ni todos los dedos son iguales. Para ello utilizamos el método *ScreenToWorldPoint*, y le damos como argumento *touch.position*, el cual nos permite realizar esta traducción.

Hay algo más que nos interesa de los toques, y es el estado de los mismos. Cada toque tiene diversos estados, pero a nosotros nos interesan 3 de ellos:

- Al comenzar el toque
- Durante el toque
- Al acabar el toque

Esta característica es muy importante ya que siempre vamos a estar arrastrando unidades de un sitio a otro, así que necesitamos los tres. El primero de ellos para recoger el toque inicial, el segundo para poder llevarnos la unidad con nuestro dedo y el tercero para actualizar el estado de la partida, ya que, dependiendo de en qué lugar del tablero soltemos la unidad que hemos recogido, ocurrirán unas cosas u otras o interaccionarán de una forma u otra con el resto de unidades aliadas o enemigas, así que, este tercer estado va a recoger gran parte de los eventos de la aplicación que tengan que ver con nuestras unidades.

Tenemos otras opciones para registrar toques de pantalla en Unity, la usada comúnmente en entornos 3D sobre todo son los rayos. Creamos un *ray*, desde el lugar en el que el usuario toca la pantalla y desde ahí podemos hacer diversas comprobaciones, como, por ejemplo, que el rayo colisione con algo.

La última opción es la más simple y eficiente, sobre todo si estamos desarrollando una aplicación que no requiere registrar más de un toque en el mismo momento (como podría ser el caso de una botonera táctil, que si requiere de toques simultáneos), y es utilizar simplemente los métodos de clic del ratón. El clic izquierdo del ratón es una traducción directa de un toque en la pantalla una vez construimos el proyecto para Android y lo transferimos a nuestro dispositivo móvil. Es la forma de registrar toques que usaremos de ahora en adelante, ya que simplifica y hace más eficiente el diseño de la aplicación puesto que no necesitamos en ningún momento de más de un toque simultáneo, y además, nos permite hacer comprobaciones de nuestros últimos cambios dentro del propio editor de Unity, es decir, si usamos la clase *Touch*, tenemos que exportar el proyecto a nuestro teléfono para hacer comprobaciones y depurar nuestro código, algo, que ralentiza más el desarrollo del proyecto.

No obstante, se ha dejado almacenado dentro del proyecto, el *script LucinaTouch.cs*, que es la construcción del movimiento de objetos mediante la clase *Touch*, completamente

funcional, para demostrar que se han explorado y trabajado dichas opciones y características.

4.3.2. Colisiones

Las cajas de colisión son clave en cualquier videojuego, y aunque este juego sea de gestión y no tengamos que saltar por plataformas o intercambiar golpes como en un juego de lucha, son importantes por la existencia del control táctil. Si la entrada que el usuario utilizara para interactuar con la aplicación fuera por mando de juegos, teclado o ratón, no necesitaríamos abordar este apartado, ya que diseñaríamos un control por entrada de botonera u otro dispositivo. Sin embargo, para poder tocar el objeto en cuestión con nuestro dedo y poder arrastrarlo, tiene que tener una caja de colisión, así pues, si nuestro dedo entra en contacto con ella, podremos interactuar de la manera que deseemos.

Aun así, esto trajo consigo un problema en nuestra aplicación, ya que al ser un videojuego centrado en la gestión no queremos que las unidades colisionen entre ellas, pero al tener caja de colisión las unidades chocaban entre ellas al moverlas. Para resolver este problema, hemos creado una nueva capa en Unity llamada *Units*, donde colocamos todos los *sprites*[12], y debemos especificar en la matriz de colisiones que la colisión entre un objeto perteneciente a *Units* y otro también perteneciente a *Units*, debe ignorarse. No podemos eliminar toda colisión del objeto ya que entonces no podríamos recogerlos con nuestro dedo.

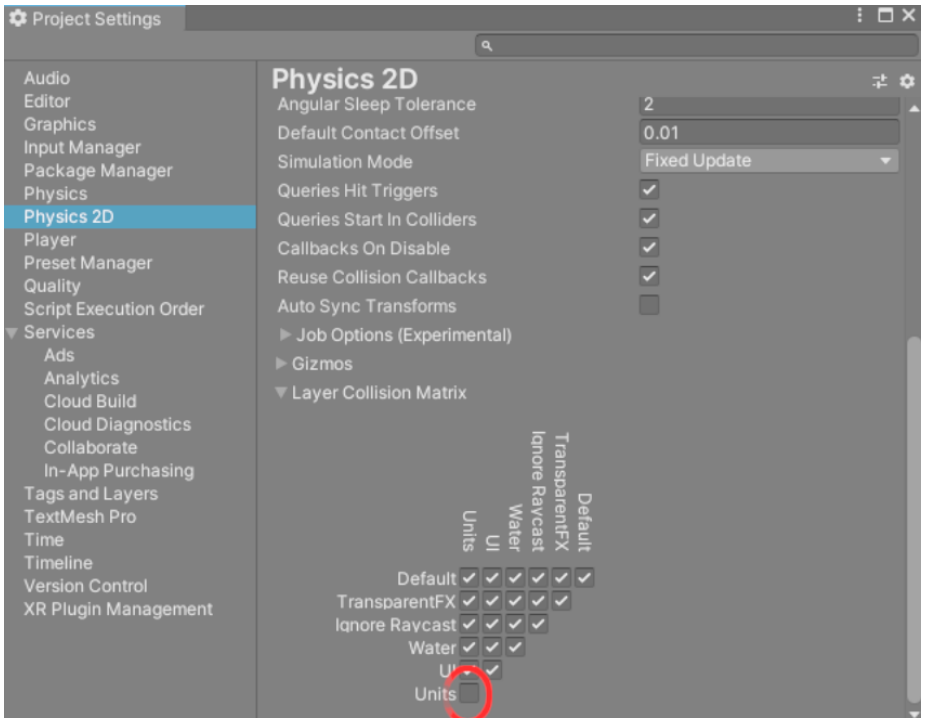

Figura 4.1: Matriz de colisión de Unity.

[12] Un sprite es un objeto gráfico diseñado para ser incluido en un videojuego. Su característica especial es que todos ellos son de dos dimensiones (normalmente tienen apariencia pixelada). Puede ser un personaje, un árbol, una piedra, un animal… Etc. Podemos definirlo también como todo elemento gráfico no poligonal de un videojuego.

4.3.3. Personajes/unidades

Este será probablemente el subapartado técnico más denso de todos, ya que el *script* que modela el comportamiento de los personajes es el que más funcionalidades cubre, como es lógico, pues estamos gestionando estas unidades siempre de una manera u otra.

Tomaremos como ejemplo el *script Lucina.cs*, que modela nuestro personaje de pruebas. Aunque existe otro personaje *Kagerou.cs*, ambos son muy muy similares, difieren en la clase y la facción a la que pertenecen, sus parámetros de ataque y defensa… Y poco más. Los métodos y funcionalidades son muy similares.

En primer lugar, tenemos varias variables que nombrar:

- *isPlaying*: Nos dirá en que estado de juego se encuentra la unidad que modela este *script*, 0 si está en la *pool* esperando a ser comprada (hablaremos de la tienda más adelante), 1 si está en el banquillo (las casillas azules de la cuadrícula) y 2 si está en juego. Esto es importante para algunas acciones que iremos mencionando a lo largo del apartado, pero, por ejemplo, si una unidad está en el banquillo no debemos contar con su clase y facción para las bonificaciones de grupo.

- *gridpos*: La utilizamos para manejar en qué lugar de la cuadrícula se encuentra esta unidad, además, le enviamos esta información a *GridManagement.cs*, que se encarga de gestionar toda la cuadrícula y llevar un control de que casillas están ocupadas y cuáles no.

- *role* y *faction*: Guardamos qué clase y qué facción tiene cada unidad para las bonificaciones de grupo.

- *registered*: Utilizamos esta variable para saber si una unidad en juego ya ha sumado su presencia para las bonificaciones de grupo. Lo explicaremos con un ejemplo: Cada vez que movemos una unidad al espacio de juego, *isPlaying* tiene un valor de 2, por tanto, como sabemos que está en juego, debemos sumar 1 a la clase y facción a la que pertenezca la unidad para las bonificaciones de grupo. El problema es que, si moviéramos una unidad de una casilla del espacio de juego, a otra casilla del espacio de juego, volvería a sumarse. Por tanto, hacemos que la unidad registre sus parámetros al entrar en el espacio de juego y quite el registro de los mismos si vuelve al banquillo.

Ya hemos hablado de las variables importantes y qué gestionan, pero tenemos diversos métodos. Dentro del método *update*, encontramos a su vez *OnMouseDown()* y *OnMouseUp()*, los cuáles gestionan el movimiento de la manera que explicamos en el apartado 4.3.1.

Por otro lado, tenemos el método *Bought()*, del cuál hablaremos más detalladamente en el apartado de la tienda, pero básicamente añade una instancia de la unidad que maneja el *script* a la primera posición del banquillo tras comprarla en la tienda.

Por último, tenemos un método llamado *LvlUp()* que ejecutaremos cuando la orden de hacerlo venga desde la tienda. Cuando tengamos dos unidades iguales y se compre otra, devolveremos estas dos a la *pool* de la tienda y reiniciaremos sus atributos a los valores por

defecto. La unidad de nivel 2 aparecerá habitualmente en nuestro banquillo con el método *Bought()*.

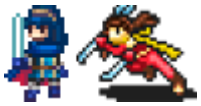

Figura 4.2: *Sprites* de Lucina y Kagerou.

Figura 4.3: Lucina de nivel 2 y Kagerou de nivel 1 en juego.

Hay que apuntar que todas las características que hemos diseñado para el *script* *Lucina.cs* son fácilmente escalables a otros personajes. *Kagerou.cs* es otra unidad completamente funcional y utiliza el mismo *script* con pequeñas variaciones, obviamente, como el nombre, rol y facción.

4.3.4. Cuadrícula de juego

En esta parte hablaremos del escenario y del *script* que contiene: *GridManagement.cs*, que gestiona la posición y el número de unidades que este contiene con diversos propósitos.

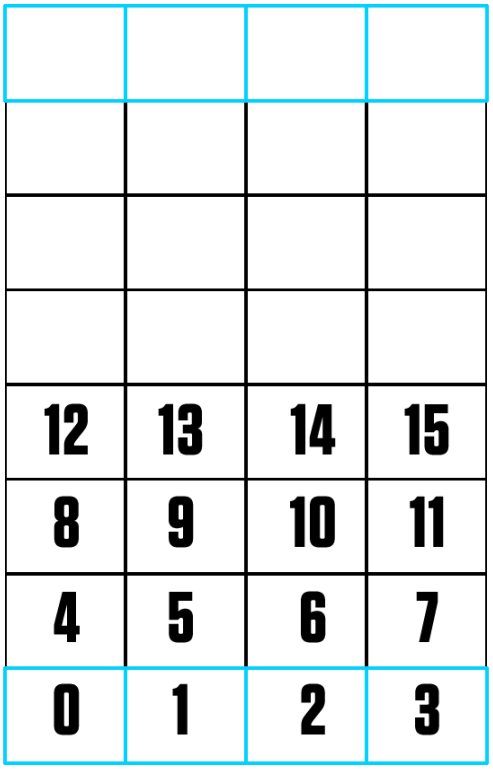

Figura 4.4: Tablero de juego y posiciones.

En este script definimos un array de tipo booleano llamado *grid*, el cual tiene 16 posiciones [0, 15], representando con valor falso si una posición está libre, y con verdadero si contiene una unidad en esa casilla. Es importante decir que no numeramos ni utilizamos toda la cuadrícula, si no la mitad de la misma, y esto se debe a que en la fase de gestión en la cual movemos unidades, las compramos, preparamos las que van a pelear y las que se van a quedar en el banquillo (las casillas azules) … Etc. No podemos invadir nunca el campo del rival, así que solo disponemos de nuestra mitad de la cuadrícula, la otra mitad, perteneciente a nuestro rival, solo podrá invadirse cuando termine la fase de gestión y comience la fase de enfrentamiento.

No solo manejamos la posición de las unidades si no cuántas hay de cada una, tanto en el banquillo como en juego, ya que, si llegáramos a obtener 3 iguales, deberíamos cambiar esas 3 unidades por una igual, pero de nivel 2. Para ello tenemos una variable para cada personaje que gestiona el número de unidades que hay en la cuadrícula, *numLucinas* y *numKagerous* son las existentes, referentes a nuestras dos unidades de prueba. Están en este momento inicializadas a 1, puesto que, para hacer pruebas, las tenemos desde el principio para poder interactuar con ellas. Cada vez que compramos una unidad, incrementamos este contador.

4.3.5. Bonificaciones de grupo

Aquí trataremos la funcionalidad de las bonificaciones de grupo. Cuando una unidad está dentro de la arena, debemos sumar su clase y facción a la lista de bonificaciones. Por ejemplo, lucina es de clase espadachina y de facción gran lord. Los espadachines aumentan su ataque ligeramente si hay dos en juego, y aumenta dramáticamente si hay cuatro en juego. Gran lord, por otro lado, otorga una ligera bonificación de ataque y defensa si es el único gran lord en juego, si colocáramos dos de ellos, perderíamos esta bonificación.

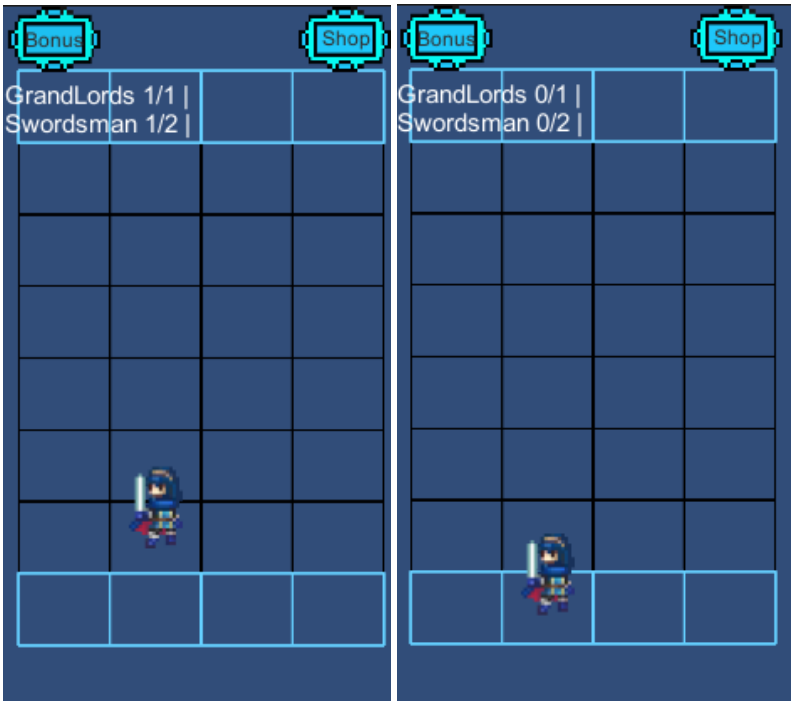

Figura 4.5: Bonificaciones de grupo.

Como podemos observar en la Figura 4.4, si Lucina está en el campo de batalla contamos con sus bonificaciones en la lista, pero si la colocamos en el banquillo, dejan de estar activas para ella y el grupo.

Hay que destacar dos aspectos importantes: El primero es que, si no hay unidades en juego, es decir, están todas en el banquillo, la lista no debería existir. Nunca debería aparecer 0/X en una bonificación, ya que, si hiciéramos esto con todas, la pantalla estaría siempre llena de bonificaciones y mancharía la zona de juego, además, teniendo en cuenta que estamos desarrollando para dispositivo móvil, cuánto más espacio podamos ganar, mejor, ya que este es bastante limitado. Se ha tomado la decisión de permitir el 0/X para poder comprobar que funciona correctamente, pero es algo que jamás llevaríamos hasta el desarrollo final. El segundo y último aspecto tiene mucha relación con lo que acabamos de explicar, el espacio en pantalla. Como podemos observar en la Figura 5.5, si pulsamos sobre el botón *Bonus* de arriba a la izquierda, ocultaremos la lista de bonificaciones, con el fin, otra vez, de permitir al usuario tener el máximo espacio en pantalla posible.

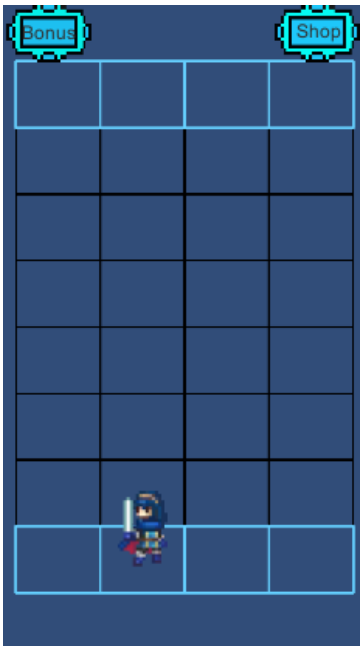

Figura 4.6: Ocultación de bonificaciones.

¿Y cómo realizamos estas acciones? Bien, para empezar, tenemos el script *BonusText.cs*, el cual actualiza su atributo *text* una vez por marco, ya que se encuentra dentro de la función *update*. El atributo *text* del objeto *bonustext*, guarda el *string* que muestra el objeto en la interfaz de usuario. Tenemos dos variables *numGrandLords* y *numSwordsman* (lo mismo con Kagerou y *numHoshidians*, ella es espadachina y Hoshidiana) que aumentamos en una unidad cada una cada vez que Lucina entra en el campo de juego, esto lo hacemos desde *Lucina.cs* y para ello necesitábamos la variable *registered* descrita en el apartado 4.3.3. Convertimos esas variables a tipo *string*, en la línea de código que actualiza el texto, así pues, cada cambio en las unidades se actualiza instantáneamente por pantalla.

Por otro lado, nos encontramos la funcionalidad del botón, el cual gestionamos en *BonusOpener.cs* y, dentro de él, definimos una variable de tipo booleana que almacena el estado del objeto de texto (si está activo en la escena o no) y ejecutamos una orden que cambie el estado del objeto de texto al contrario que hemos almacenado. Para poder activar esta función presionando el botón, le hemos otorgado al Sprite que vemos en las esquinas superiores en las figuras 4.4, y 4.5, una componente en el editor de Unity (*Button*) para convertirlo en botón y le agregamos en su evento de pulsado, que ejecute la función *OpenBonus()* que acabamos de describir dentro del *script BonusOpener.cs*, podemos observarlo en la figura 4.6.

Cabe destacar que para que el botón sepa que objeto *Text* debe tener en cuenta para modificar su estado, definimos un objeto *Text* dentro de *BonusOpener.cs* y en el inspector de Unity, arrastramos el objeto de texto para referenciarlo como podemos ver en la figura 4.7.

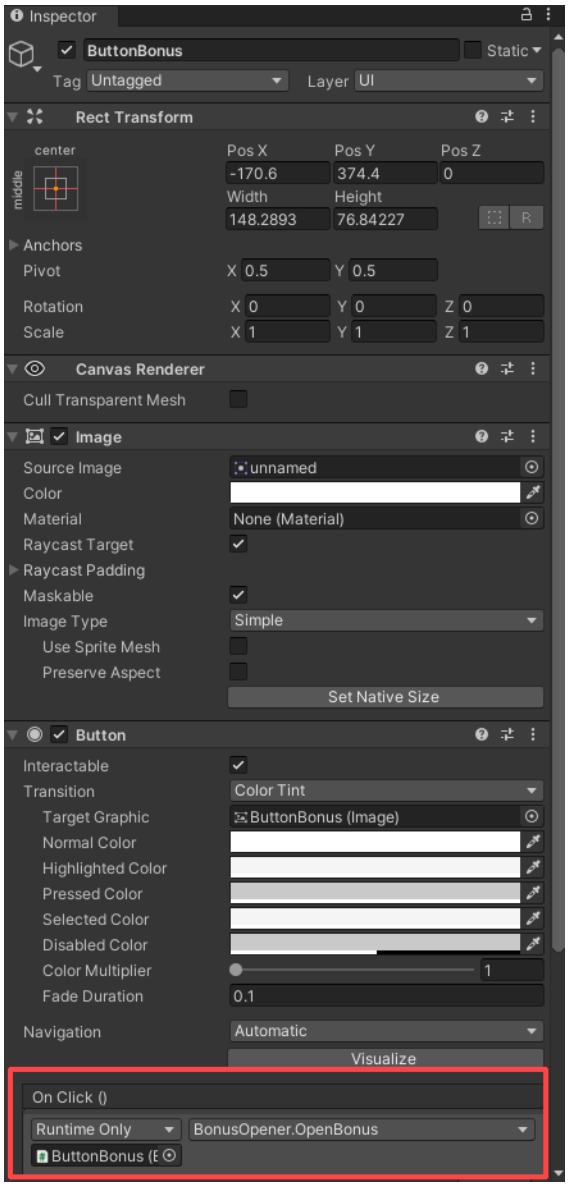

Figura 4.7: Botón de activación/desactivación de bonificaciones.

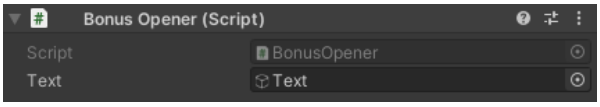

Figura 4.8: El objeto texto que modificamos con el botón *Bonus*.

4.3.6. Tienda

En este último subapartado, explicaremos como funciona la tienda de nuestra aplicación y qué efectos tiene.

En primer lugar, tenemos un *script* llamado *ShopOpener.cs* que tiene exactamente la misma funcionalidad que el anterior para las bonificaciones de grupo, con la pequeña diferencia que esta vez en lugar de ocultar un solo objeto (como el caso anterior con el texto), deberíamos ocultar varios *sprites* que representan a las unidades que podemos comprar, para ello, lo que hacemos es añadir todos los *sprites* a la escena y hacer que

hereden todos de un mismo objeto vacío que hemos denominado *Shop* como podemos observar en la figura 4.8.

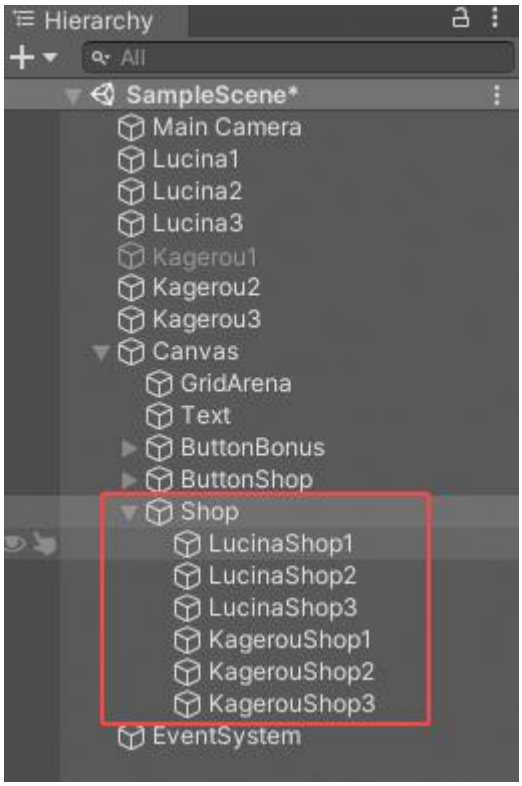

Figura 4.9: Unidades en la tienda heredando de un objeto vacío *Shop.*

La tienda debe ser de generación aleatoria cada ronda, pero como aún no tenemos diseñadas las rondas, hemos realizado una generación aleatoria en la inicialización de la tienda para poder mostrar esta característica, de manera que cada vez que pulsamos el Play en el editor de Unity o cerramos y abrimos la aplicación en nuestro teléfono tendremos una tienda distinta. Por ahora solo estamos manejando dos unidades de prueba, así que para cada hueco de la tienda (esta tiene 3), generamos un número aleatorio entre 1 y 2, si sale 1, cargaremos en el hueco el *sprite* de compra de Lucina, y si sale 2, el de Kagerou, así para cada uno de los 3 huecos.

Si pulsamos con el dedo cualquiera de estos *sprites* de compra, haremos uso de la función *buyexample()* situada en *BuyExample.cs*, en la que obtenemos el nombre del *sprite* que se ha pulsado y buscamos el objeto de juego correspondiente a ese nombre (si pulsamos sobre la imagen de compra de Lucina, buscamos una Lucina para añadirla como objeto de juego), y una vez encontramos el objeto con la función *Find()* de Unity, ejecutamos el método *Bought* que hemos definido dentro del *script* de cada personaje, lo que coloca esa pieza, por defecto, en la primera posición del banquillo y la inicializa lista para su uso.

Por último, hay que anotar que, si tenemos dos unidades iguales y decidimos comprar una tercera, las dos presentes desaparecerán y en su lugar obtendremos una única unidad, pero de nivel 2. Para esto, utilizamos el atributo numLucinas de *GridManagement.cs*, así pues, si el contador es 2 y compramos otra, ejecutamos el método *LvlUp* de los personajes y el método estándar *Bought* de la unidad de nivel 2.

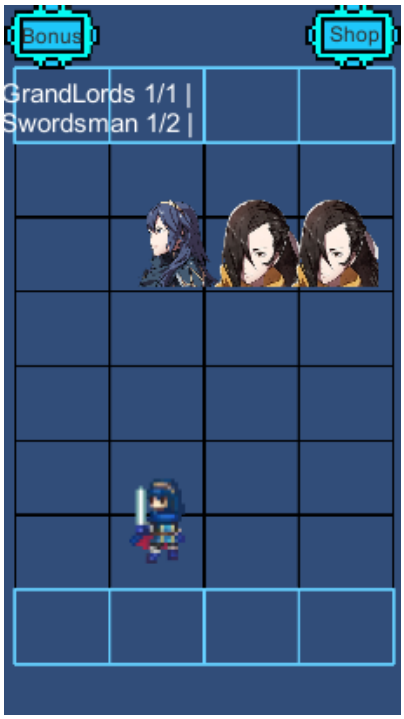

Figura 4.10: Pantalla de juego en la que podemos observar la tienda abierta.

4.4 Impresiones y análisis del desarrollo técnico

En esta parte, trataremos de abordar las sensaciones que nos deja el desarrollo, en qué punto de la aplicación estamos y cuánto nos hemos acercado a los objetivos propuestos en el apartado 4.1.1.

En cuanto a los dos primeros objetivos, estamos bastante contentos, toda la aplicación funciona correctamente con control táctil en un dispositivo Android de manera fluida, y apunta a que podría seguir escalando sin problemas en este sentido, e incluso proponer mayores niveles de complejidad, ya que se ha estudiado y abordado de forma densa las diferentes formas de manejar estos controles dentro de Unity. Cabe destacar, que el desarrollo en Unity para Android es más restrictivo de lo que lo es en iOS, donde este motor ofrece mejores funcionalidades y más óptimas para la segunda opción. Al ser Android más restrictivo, también nos hemos permitido el lujo de construir y exportar el proyecto en un dispositivo móvil de Apple y hemos comprobado que funciona, lo cual, abre más opciones de mercado.

En segundo lugar, creemos que se han representado correctamente en la memoria las características del juego, de dónde viene, a dónde va y cómo estamos reflejando estas características en el desarrollo.

En último lugar, y aquí si vamos a ser más duros, esperábamos que para el día de la entrega la aplicación tuviera más funcionalidades que enseñar. Somos conscientes de los límites expuestos en el apartado 4.1.2, pero a nivel de programación y diseño de mecánicas es cierto que la aplicación podría estar un par de pasos más por delante de lo que está ahora, y tener alguna funcionalidad más añadida, como el cambio de fase entre gestión y enfrentamiento, una IA con la que hacer pruebas… Etc. Sin embargo, el contexto personal y laboral ha sido inesperado en los últimos meses y nos ha dejado mucho menos tiempo del esperado para el desarrollo de las funcionalidades.

5. Estudio de mercado y costes

En este apartado analizaremos el coste del proyecto real, así como el margen de beneficio esperado publicando la aplicación en una plataforma digital móvil.

5.1 Presupuesto

Para obtener un presupuesto debemos tener en cuenta varios factores principales y otros más secundarios. Nos centraremos primero en los principales los cuáles son:

5.1.1 Coste de desarrollo.

Este coste lo obtenemos prácticamente de la mano de obra para realizar el proyecto. La duración estimada del desarrollo son 480 horas (2 meses) por persona, y el sueldo de desarrollador junior oscila entre 1000€ y 1200€, aunque el rol concreto de programador suele estar mejor cotizado en España ya que hay más demanda (Fernández, 2020). Necesitamos tres personas como mínimo para llevar adelante el desarrollo, dos artistas y un programador, esto se debe a que, una de las características clave para llegarle al público es un apartado gráfico llamativo y embaucador, por tanto, queremos invertir seguridad en conseguir un producto muy cómodo a la vista del usuario. Adhiriéndonos a la referencia de sueldos medios, asumimos un coste de 1100€ por cada artista y 1300€ por programador. Eso nos deja el coste del proyecto en: (1100*2 + 1300*1)*2 = **7000€**

5.1.2 Coste de publicación.

Una vez tenemos la aplicación desarrollada, el siguiente paso es la publicación en una plataforma digital móvil. Afortunadamente, publicar en dispositivos móviles es un proceso tan sencillo como barato, 25 dólares (21,13 euros) para ser desarrolladores en Google Play y poder publicar en su tienda. No obstante, Google se queda con el 30% de todos los beneficios generados por la aplicación, ya sea por precio de compra, publicidad, micropagos o suscripciones (soporte de Google Play Console). Para publicar nuestra aplicación tenemos varias opciones:

- Publicarla como aplicación gratuita con publicidad dentro de la aplicación. Colocaríamos un anuncio a cada usuario al iniciar la aplicación y tras cada partida (entre 5 y 10 minutos aproximadamente)[13].

- Publicar la aplicación con un coste dado y eliminar la publicidad dentro del juego. El coste debería ser bajo, 1,99€ por ejemplo.

[13] En las aplicaciones más famosas como YouTube o Instagram, los anuncios aparecen cada 5 minutos como es el caso de YouTube o menos incluso en Instagram. Siendo dos de las aplicaciones más usadas del mundo, utilizamos sus tiempos de anuncio como referencia siendo nosotros más permisivos, dado que nuestra aplicación no tiene su alcance e impacto.

Sea cual sea la decisión, mantenemos el objetivo de incluir micropagos dentro del juego que permitan al usuario cambiar la apariencia de los personajes o de la arena de combate, entre otras cosas. Elegiremos **publicarla gratuitamente,** prácticamente todas las aplicaciones de juegos en dispositivo móvil son gratuitas, así que debemos ser competitivos y adaptarnos a estas exigencias para llamar la atención del usuario. Además, tan sólo un 35% de usuarios de móviles ha pagado alguna por una aplicación (El Publicista, 2019).
Por tanto, el coste de publicación será:
Coste de publicación y mantenimiento = 21,13 + beneficios*0,3
(Aclararemos la variable beneficios cuando hablemos de los mismos y hagamos el cálculo de ingresos estimados).

5.2 Ingresos estimados

Ya sabemos cuánto nos cuesta llevarlo adelante y cómo vamos a hacerlo dentro del mercado, ahora toca estimar cuándo beneficio podríamos obtener en un año llevando el proyecto adelante y, por tanto, si sale rentable ejecutarlo. Cabe destacar que las estimaciones las vamos a encarar de una forma segura y conservadora, es decir, a la hora de tomar números y referencias para hacer los cálculos, vamos a estar siempre en el escenario más neutro para poder hacer unos cálculos lo más normativos posibles, pero en el caso de tener que "suponer" qué van a hacer los usuarios, nos acercaremos más a un escenario peor que a uno positivo.

Los anuncios en móvil generan entre 0,01 y 0,03 euros para el desarrollador (El Androide Libre, 2015), así que usaremos la mediana para hacer un cálculo estimado: 0,02.

Ahora tenemos que tenemos el beneficio de cada anuncio, tenemos que estimar cuántos anuncios van a ver los usuarios. Los juegos móviles ocupan un 10% del tiempo de los consumidores de smartphone (Que Nube, 2019). Las sesiones diarias medias de dispositivos son de dos horas y 53 minutos (El Publicista, 2019). Por tanto, de media, 17,3 minutos al día van a ser dedicados a jugar a videojuegos en dispositivos portátiles.

Empezamos con el cálculo, si nuestras partidas duran entre 5 y 10 minutos de media, vamos a asumir que un usuario juega dos partidas al día. En Google Play podemos ver que otras aplicaciones del género de nuestro juego que son mediocres y poco innovadoras, tienen más de 200.000 descargas en un año, así que asumámoslas también.

Si cada usuario juega dos partidas al día, verá tres anuncios, al iniciar el juego y al final de cada partida. Supongamos también que tenemos en nuestra tienda aspectos alternativos para nuestras arenas o personajes con un valor de 1,99€ (muy barato comparado al resto de micropagos en cualquier otro juego), 7 de cada 10 jugadores tiende a pagar por contenido adicional en juegos gratuitos (González, 2018), pero asumamos que nuestro contenido le gusta un poco menos a la gente por lo que pueda salir mal, así que asumimos 4 de cada 10 jugadores para nuestro producto. Hagamos las matemáticas:

100.000*(0,02*3) + (100.000*0,4)*1,99 = 6000 + 79.600 = 85.600€

Ahora debemos considerar la parte que se queda Google Play: 85.600 * 0,3 = 25.680€

Por tanto, el beneficio estimado el primer año es: 85.600 – 25.680 – 7000 – 21,13 = **52.898,87**

5.3 Análisis del resultado

A primera vista podemos observar que desde luego el producto es rentable, y, sobre todo, teniendo en cuenta de que hemos sido pesimistas en algunos apartados y no hemos tenido en cuenta el impacto que puede tener un producto que encuentra un nicho inexplorado en el sector de los videojuegos y lo explota, lo cual podría inflar en gran escala los beneficios obtenidos. Otro punto importante es que como podemos observar que la gran mayoría de beneficios los obtenemos de los micropagos, ese es uno de los motivos por los cuáles en el presupuesto incluimos dos artistas en lugar de uno, para poder seguir trabajando y retroalimentar la generación de contenido adicional para el usuario constantemente.

Es necesario apuntar que, todo este estudio se ha realizado para su publicación en la tienda de dispositivos Android, pero como mencionamos en el apartado 4.4, la aplicación sería fácilmente exportable a iOS y podríamos tener nuevas opciones de mercado que abordar.

En conclusión, podemos decir que el proyecto es cuánto menos prometedor y que en muy poco tiempo cubriría los costes de desarrollo de forma casi segura.

6. Conclusiones

Las aplicaciones móviles son muy parecidas la gran mayoría entre sí, casi todo lo que encontramos en las tiendas digitales es una copia de otra aplicación o modelo ya existente con una pequeña modificación para intentar hacerlo único. Este modelo de negocio funciona muy bien y tiene pocos riesgos, por eso vemos tantas aplicaciones nuevas cada día, así que encontrar un terreno inexplorado con una aplicación móvil es algo impactante a día de hoy, incluso para mí que desarrollé la idea, me sorprendió pensar que nadie hubiera hecho esto antes cuando en nuestra sociedad "ya está todo inventado". Ese impacto de encontrar un nicho inexplorado, puede tener mucho valor si la aplicación en cuestión se presenta bien desde un primer momento, y ese impacto es tanto monetario como de prestigio.

Otro punto positivo de la conclusión es que creemos que hemos aprovechado bien una de las competencias más importantes aprendidas en el grado, y es la capacidad de ser ingenioso y autodidacta. A pesar de que se nos ha enseñado a programar con Java y otros lenguajes, en entornos como BlueJ, Eclipse… Etc. Hemos hecho uso de una herramienta que siempre nos recordaban que puliéramos, y es el extrapolar dichas enseñanzas a otro lenguajes y entornos, pues las posibilidades son casi infinitas. Así que, nos hemos lanzado a un mar que no tiene costa en la península de la carrera, ya que el desarrollo en Unity es algo que ni en las asignaturas optativas de videojuegos se trabaja, y estamos orgullosos de haber podido utilizar los conocimientos tradicionales, fuertes como la base de una pirámide que se nos otorgan en el grado, para ser capaces de aprender a manejar una de las herramientas más punteras y a la orden del día en el apartado de la programación de videojuegos.

El punto negativo de la conclusión ha sido mencionado en el apartado 4.4, nos habría gustado que la aplicación estuviera en una fase más avanzada de desarrollo pero por motivos personales y de tiempo, nos ha sido imposible ir más lejos, es una pena porque queríamos mostrar todavía más potencial e impacto, pero hemos tenido que parar aquí para la entrega, aun así, no mentimos si hablamos de que el esfuerzo en intentar llegar muy lejos en un lapso muy corto de tiempo ha sido titánico.

6.1 Mejoras a largo plazo

En el apartado 4.4 hemos hablado de mejoras a corto plazo en cuánto a programación y diseño de funcionalidades se refiere, pero en esta parte hablaremos de qué mejoras podría tener la aplicación a largo plazo para ser más competitiva y funcional:

- Tener un flujo de contenido nuevo constante. Una vez la aplicación se asentara, habría que seguir sacando contenido por el que la gente pueda usar estos micropagos para cambiar su apariencia, arena de combate, la apariencia de sus unidades… Etc.
- En el futuro probablemente, habría que innovar con nuevos modos de juego o nuevas clases. Ese tipo de juegos competitivos tienen lo que se conoce como *metagame*[14], y

[14] El metagame hace referencia en los videojuegos a cuál es la forma óptima de jugar en ese momento de vida del juego. Por ejemplo, si en nuestro juego con el paso de tiempo los usuarios descubren que una composición que incluya a los magos es demasiado potente, la moda va a ser jugar esos magos por parte de todo el mundo.

nuestro juego debería tener un equipo de personas que analicen este apartado y escuchen a los usuarios para retroalimentarse mutuamente y hacer que el juego evolucione y no se estanque.

- Portabilidad a otras plataformas. Si el juego triunfara, habría que considerar rápidamente la posibilidad de lanzarlo en la tienda digital de Nintendo Switch por ejemplo, una consola que ahora mismo está muy familiarizada con los juegos destinados a dispositivos portátiles.

Entonces el juego sufriría una reducción de la potencia de los mismos o un aumento de la potencia de otra clase, y con ello, la forma óptima de jugar evolucionaría a partir de ese momento.

Agradecimientos

Me gustaría agradecer a mi tutor Manuel Agustí, el cariño e interés que ha mostrado siempre por mis ideas y proyectos. En la asignatura que compartimos juntos me dejó explorar nuevas formas de desarrollo y acercarme más a las aplicaciones contemporáneas que utilizan estudios y empresas a día de hoy. Es algo que en algunos momentos eché de menos en el grado, ya que se estudian unas bases muy tradicionales y genéricas (lo cual es completamente lógico), y siempre eché en falta desarrollos contemporáneos y palpables. Además, siempre me he sentido un poco distante con la mayoría de profesores en mi vida, imagino que por mi forma de ser, educación y cómo he madurado a lo largo de los años, ya que por motivos personales, mi vida académica fue muy difícil y, sin embargo, con Manuel me he atrevido a ser sincero y hablarle de cómo veía las cosas, me sentía o qué problemas estaba teniendo, sin miedo a recibir desprecios, rechazo o negativas directas y la respuesta siempre fue positiva y paciente. El tenerle a él como tutor ha hecho que este desarrollo sea un poco más fácil.

En segundo lugar, me gustaría agradecerle el cariño y la paciencia a mi pareja María, sabe que las últimas fases del TFG han sido muy duras a nivel personal por la gran carga laboral y personal que he estado llevando en los últimos meses, y ha visto como mis largas sesiones de trabajo me dejaban exhausto y desesperado, ha estado siempre aguantando por mí, no dejándome que me rinda y que siga apretando, que solo quedaba la recta final y había que jugar las cartas.

Y, por último, a mis amigos, en primer lugar, a Yeray, estudiante de videojuegos que me ha animado mucho a llevar adelante un TFG de esta temática, motivándome de forma positiva y queriendo demostrarle de que a pesar de no haber podido estudiar videojuegos, puedo ser capaz de ser programador de ellos, una competición sana. Y, en segundo lugar, al resto, Sergio, Ernesto, Ángela, Miguel y Daniel me han visto pasarlo mal con tanta carga de trabajo y la ansiedad y locura que eso me generó en los últimos meses, y siempre han tenido la mano en mi espalda por si caía.

Bibliografía

- Alberto González (2019, 30 de septiembre). Artículo para la revista digital Vandal. https://vandal.elespanol.com/noticia/1350727602/teamfight-tactics-tiene-33-millones-de-jugadores-mensuales/

- Xavi Mogroviejo (2020, 23 de marzo). Artículo para la revista digital IGN España. https://es.ign.com/teamfight-tactics-pc/161882/feature/por-que-deberias-estar-jugando-a-tft-teamfight-tactics-en-tu-movil

- Vince (2018, 12 de abril). Artículo para el blog iD Tech. https://www.idtech.com/blog/different-types-of-video-game-genres

- Rubén Aido (2019, 31 de octubre). Números de ventas de Fire Emblem. Arítculo para la revista digital Areajugones. https://areajugones.sport.es/videojuegos/fire-emblem-three-houses-revela-sus-numeros-totales-de-ventas/

- Carlos Sánchez (2016, 25 de octubre). Tomb Rider y XCOM 2 se convierten en million-sellers. Artículo para la revista digital Juegosadn. https://juegosadn.eleconomista.es/rise-of-the-tomb-raider-y-xcom-2-ya-son-million-seller-en-steam-no-100341/

- Sebastián Guadalupe (2019, 5 de febrero). Wargroove inicia su lanzamiento con excelentes ventas. Artículo para la revista digital MasGamers. http://www.masgamers.com/wargroove-lanzamiento-ventas

- Gonzalo Guillem (2019, 13 de marzo) Brawl Stars alcanza los 75 millones de descargas. Artículo para el apartado de videojuegos de la revista digital eldesmarque. https://esports.eldesmarque.com/brawl-stars/brawl-stars-descargas-63973

- Yeray Fernández (2020, 19 de agosto). Estudiante de cuarto año de desarrollo de videojuegos en la Escuela Universitaria de Diseño Innovación y Tecnología (ESNE) de Madrid.

- Soporte de Google Play Console. Información sobre los costes de publicación y los porcentajes de beneficios para la empresa en cada aplicación. https://support.google.com/googleplay/android-developer/#topic=3450769

- Artículo para la web El Androide Libre (2015, 5 de abril) https://elandroidelibre.elespanol.com/2015/05/la-publicidad-en-las-apps-android-desde-un-punto-de-vista-tecnico.html

- Artículo para El Publicista. "¿Cuánto tiempo le dedicamos al móvil?" https://www.elpublicista.es/mundo-online/cuanto-tiempo-dedicamos-mobile

- Los juegos móviles ahora representan el 33% de las instalaciones, el 10% del tiempo y el 74% de gasto de los consumidores. Artículo en la revista digital Que Nube (2019, 11 de junio). https://quenube.com/los-juegos-moviles-ahora-representan-el-33-de-las-instalaciones-el-10-del-tiempo-y-el-74-del-gasto-de-los-consumidores/

- Sergio C. González (2018, 27 de junio). Fortnite: ¿Cuánto dinero gastan de media sus jugadores? Artículo para la revista digital de videojuegos MeriStation. https://as.com/meristation/2018/06/27/noticias/1530074640_176956.html

- Manual de uso de Unity. Documentación. https://docs.unity3d.com/Manual/index.html