UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Paranoia: Desarrollo de un videojuego de rol mediante pixel art

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Cantó Moscardó, José

Tutor/a: Lluch Crespo, Javier

CURSO ACADÉMICO: 2021/2022

Resumen

En este trabajo se redacta el desarrollo de un videojuego 2D empleando Unity 5 2021.3.8f1 como motor gráfico y Jetbrains Rider como entorno de desarrollo. El objetivo del proyecto es documentar el proceso de creación de un juego tipo 'RPG' basado en rol de paranoia, estudiando casos de éxito del género y argumentando el porqué de las decisiones tomadas en el proceso.

Palabras clave: RPG; Paranoia; Unity.

Abstract

In this paper we'll see the development of a 2D video game using Unity 5 2021.3.8f1 as graphics engine and Jetbrains Ryder as development environment. The objective of the project is to document the process of creating an 'RPG' type game based on paranoid role-playing, studying success stories of the genre and arguing the reasons for the decisions made in the process.

***Keywords**: RPG; Paranoia; Unity.*

Índice general

Índice de figuras

Índice de tablas

1 Introducción

En este trabajo se va a redactar todos los pasos seguidos en el proceso de desarrollo de un videojuego ´RPG´, comenzando con una investigación previa de toda la evolución tecnológica de las tendencias de los videojuegos, pasando por la preproducción y finalizando con el proceso de desarrollo.

Empecé a jugar a videojuegos hace más de dos décadas, en todo este tiempo siempre había querido crear mis propios juegos. Desde edades tempranas y sobre todo con el auge de los juegos multijugador en línea (MMORPG) empecé a interesarme por el mundo del desarrollo y a modificar servidores para jugar con mis amigos y conocidos. Pero, desde siempre, había soñado con producir mis propios videojuegos. Sin embargo, no fue hasta que casi finalice mi grado en informática que tuve la confianza de intentar hacer realidad este pequeño sueño solicitando la oportunidad de desarrollar este trabajo de fin de grado.

Por suerte, no soy el único al que le entusiasman los videojuegos y en este mundo globalizado en el que vivimos la industria de los videojuegos ha crecido hasta ser la más importante en el ámbito de entretenimiento a nivel mundial.

1.1 Crecimiento del mercado

En las dos últimas décadas la industria del videojuego ha crecido a un nivel exponencial, llegando a mover casi 200 Billones de dólares en 2021 y con una previsión futura para este próximo año 2022 de más de 200 Billones de dólares (Wijman, 2022). El hecho de que se le dedique tanto dinero, ha provocado que se convierta en el mercado más grande dirigido al entretenimiento del mundo.

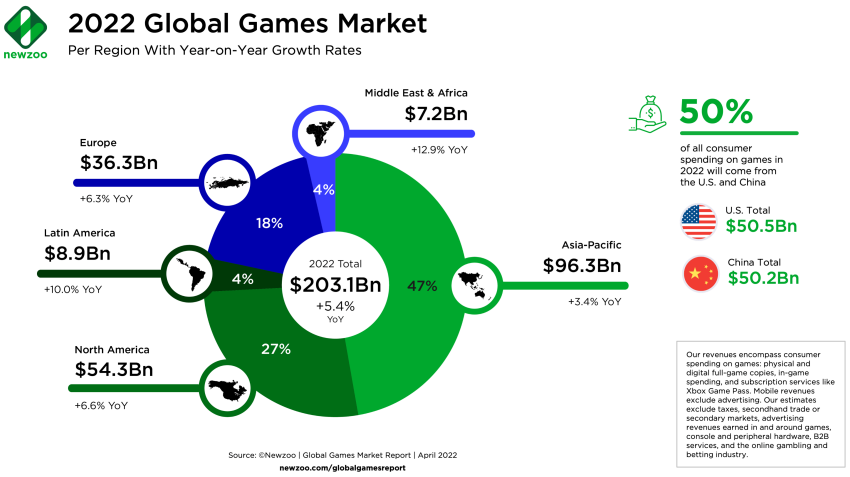

Figura 1.1: Previsión global del mercado de juegos para 2022

Manejando estos números, cada vez más empresas y particulares están dedicando sus esfuerzos a desarrollar juegos, emprendiendo en la creación de estudios de videojuegos a gran escala y juntando desarrolladores, artistas, compositores, directores, etc. Al mismo tiempo, más particulares están entrando en la industria a través del uso de motores gráficos cada vez más accesibles, fáciles de aprender y con modelos de negocio más flexibles económicamente. Esto ha hecho que el mismo software pueda ser utilizado no solamente por las grandes corporaciones (como había sido el caso hasta hace poco), sino también por estudios y particulares más pequeños, ya que la cantidad de dinero que cada usuario paga por el uso del software es proporcional a sus ganancias.

1.2 Interés y opinión pública

Debido a la desinformación de los medios de comunicación convencionales, ha surgido una visión e idea popular sobre los videojuegos bastante alejada de la realidad. Esta imagen los describe como pasatiempos basados únicamente en el interés lúdico, dirigidos a un jugador promedio de 13 años, y que al ser utilizados de forma continuada pueden dar pie a alteraciones del comportamiento. Se cree que esto puede fomentar la aparición de soledad, depresión y violencia, y en peor medida, se ha llegado a comparar este efecto al de drogas como la cocaína (Fox, 2018). De esta forma, se ha desarrollado la creencia de que al jugar se da una liberación de dopamina sin esfuerzo, y que esta es nociva para quien haga uso de los videojuegos.

Acciones como ver películas, leer libros o escuchar música también liberan dopamina en nuestro cuerpo, pero a diferencia de los videojuegos, estas acciones en muchos casos han sido consideradas cultura general. Jugar a un videojuego combina las tres acciones mencionadas anteriormente en una sola, y se ha demostrado que requiere un nivel de concentración mucho más alto para poder llevarse a cabo. Se han realizado estudios (barnardos.org.uk, 2021) que demuestran que los videojuegos estimulan funciones cere-

brales, resolución de problemas, memoria, tiempo de reacción, salud mental, habilidades cognitivas, etc.

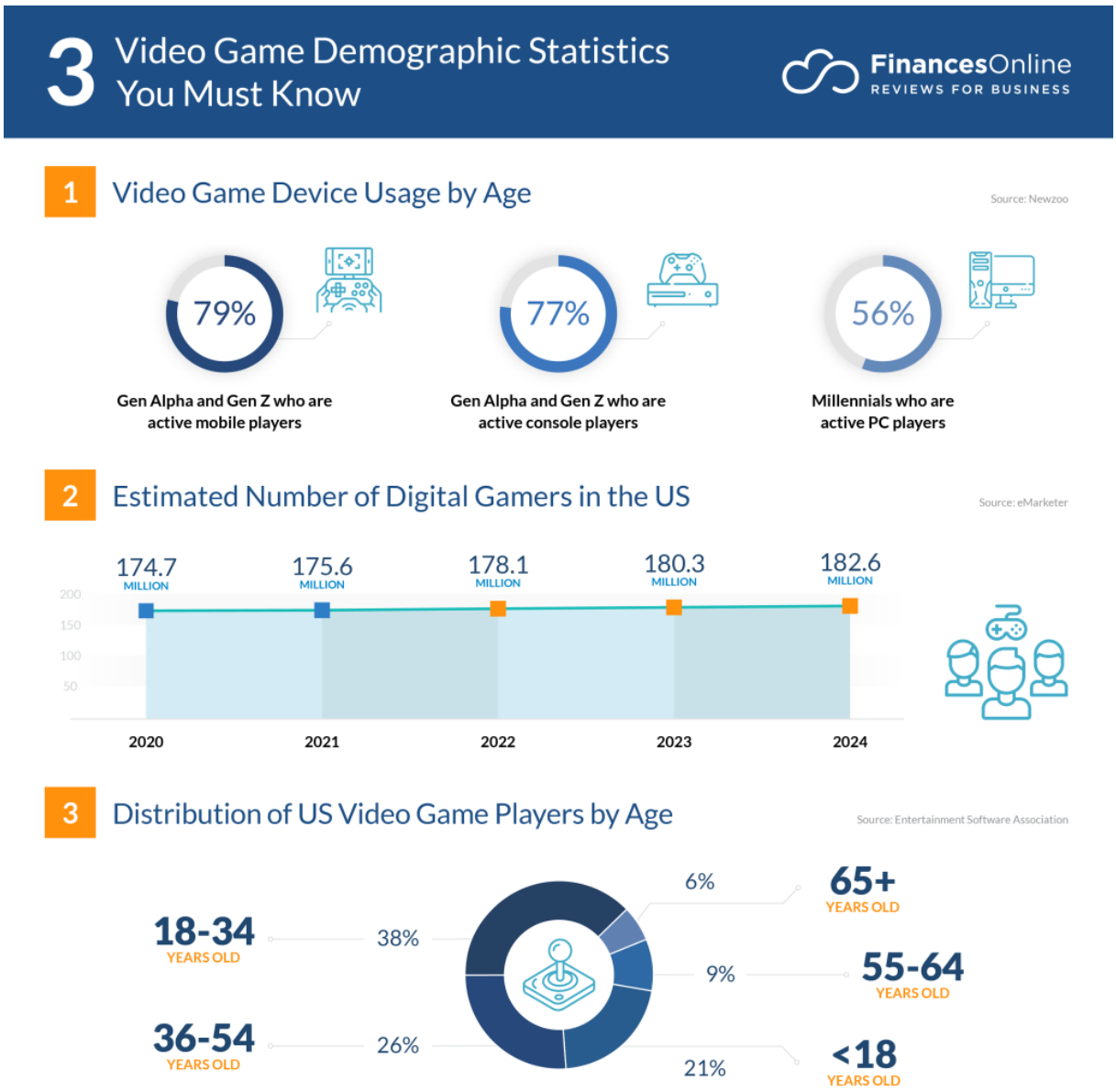

Figura 1.2: Demografía en videojuegos

A partir de los datos obtenidos en la imagen anterior (figura 1.2), si analizamos la edad de los jugadores, podemos ver que por lo general la demografía al jugar un videojuego es muy variada, siendo el rango de edad donde se concentra mayor porcentaje de jugadores el que va de los 18 a los 34 años (Chang, 2022).

Debido a diversos factores y como núcleo al relevo generacional, podemos decir que la idea preconcebida que hemos mencionado anteriormente de que los videojuegos son nocivos para la salud, está pasando a un segundo plano, y que cada vez más su uso está empezando a considerarse como cultura entre las personas.

1.3 Objetivos

Con la información expuesta en los puntos anteriores, este trabajo tiene diversos objetivos:

- Aprendizaje:
  - Aprender sobre la industria del videojuego y más enfocado a la producción de los mismos.
  - Aprender sobre las fases reales de desarrollo de un videojuego utilizadas por los estudios más exitosos en 2022.
  - Aprender y estimar costos monetarios y temporales realistas en el desarrollo de los videojuegos.
  - Aprender como analizar el mercado e intentar anticiparse a él.
  - Aprender a gestionar el tiempo en desarrollos de software largos.
  - Aprender a utilizar un motor gráfico ampliamente utilizado en la industria.
  - Aprender a gestionar los recursos disponibles para llevar un proyecto a su fin de forma exitosa.
  - Aprender a gestionar riesgos de forma controlada, siempre buscando la mayor rentabilidad posible.
- Desarrollo
  - Ser capaz de desarrollar un prototipo en el tiempo estimado.
  - Utilizar buenas prácticas a lo largo de todo el proyecto.
  - Emplear las herramientas más óptimas para realizar cada tarea.
  - Reutilizar código o assets que permitan mejorar la calidad del proyecto y acortar tiempos de producción.

1.4 Impacto Esperado

Una vez finalizado el desarrollo de todo el videojuego que comienza con el prototipo desarrollado en este trabajo de fin de grado, se espera haber conseguido un videojuego capaz de competir en el mercado, generar beneficios y conseguir buenas críticas por parte de los usuarios. Además, con la historia del videojuego me gustaría ayudar a que la gente se sienta más concienciada con diversos objetivos de desarrollo sostenible como la reducción de las desigualdades, las ciudades y comunidades sostenibles, la producción y consumo responsables, la acción por el clima, la paz y las alianzas.

1.5 Estructura

Este trabajo estará estructurado de la siguiente forma:

- 2. Contexto tecnológico:
  - Análisis de las modas referentes a los videojuegos desde la década de los 50 hasta la actualidad.
- 3. Análisis del problema:
  - Estudio del problema actual del mercado de los videojuegos y como solucionarlo con una buena preproducción.
- 4. Diseño de la solución:
  - Descripción sobre los puntos a desarrollar en el prototipo.
- 5. Desarrollo de la solución propuesta:
  - Desarrollo paso a paso de los objetivos del punto 4.
- 6. Pruebas:
  - Pruebas propias y de personas externas al trabajo con el fin de encontrar errores y estudiar la viabilidad del prototipo.
- 7. Conclusión:
  - Conclusiones y análisis del cumplimiento de los objetivos marcados en la introducción
- 8. Objetivos de desarrollo sostenible
  - Análisis sobre qué objetivos cumple este trabajo en relación con la agenda 2030 europea.

2 Contexto Tecnológico

En el pasado, crear un videojuego que fuese catalogado como 'bueno' era mucho más simple que hoy en día, y su vez, no se esperaba lo mismo al desarrollar y lanzar un videojuego en los años 80 que actualmente, por lo que podemos decir que las herramientas, la tecnología e incluso los propios humanos han evolucionado junto con los videojuegos. Existen determinadas mecánicas que en el momento de la implementación han marcado un antes y un después para determinados géneros, por ejemplo el primer juego de aventuras que incluyo minimapa hizo que todos los juegos posteriores plantearan el incorporarlo.

Figura 2.1: Pacman Original - 1980

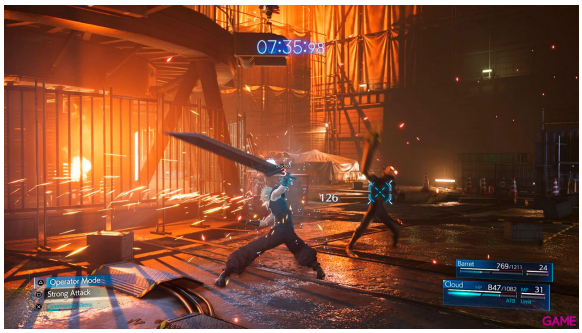

Figura 2.2: Final Fantasy VII Remake - 2020

Mientras que las diferencias más notorias a primera vista pueden parecer simplemente gráficas debido al aumento de potencia en este campo y en el computacional para permitir mover más polígonos al mismo tiempo. Si analizamos más profundamente encontramos evolución en todos los aspectos, música, historia, jugabilidad, inteligencia, iluminación, etc.

Sabiendo esto, analizar los juegos desde su origen en la década de los 50 puede ayudarnos a entender el camino que ha seguido la tecnología desde el comienzo de su historia. Esto puede darnos algún tipo de conclusión que ayude a entender como seguirá avanzando en el futuro.

2.1 Década de los 50

A inicios de esta década surgió el primer videojuego (UPC, 2019), era una versión interactiva del tres en raya llamada 'Nought and crosses' y fue desarrollada por Alexander S.Douglas en 1952. Esta versión del juego permitía que un humano se enfrentase a la máquina y se ejecutaba en la EDSAC, una de las primeras computadoras creadas.

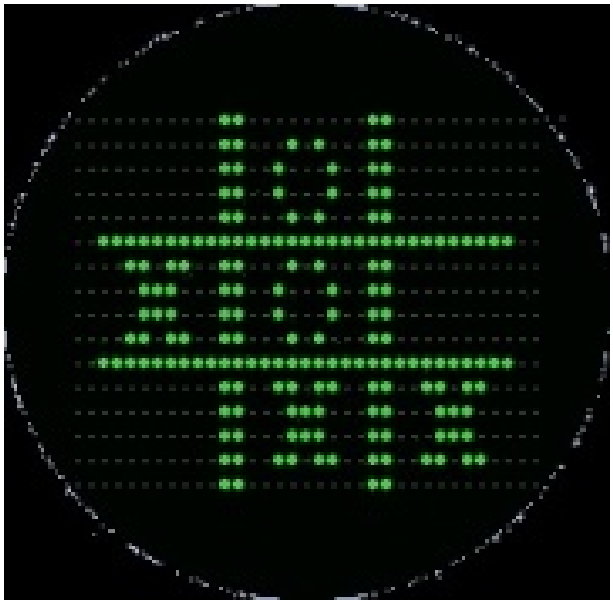

Figura 2.3: Nought and crosses - 1951

En 1958 William Higginbotham creo 'Tennis for Two', un videojuego que simulaba el tenis de mesa y que permitía a dos jugadores enfrentarse. Este fue registrado como el primer juego multijugador de la historia. Fue producido para asegurar el entretenimiento de los visitantes de la exposición Brookhaven National Laboratory.

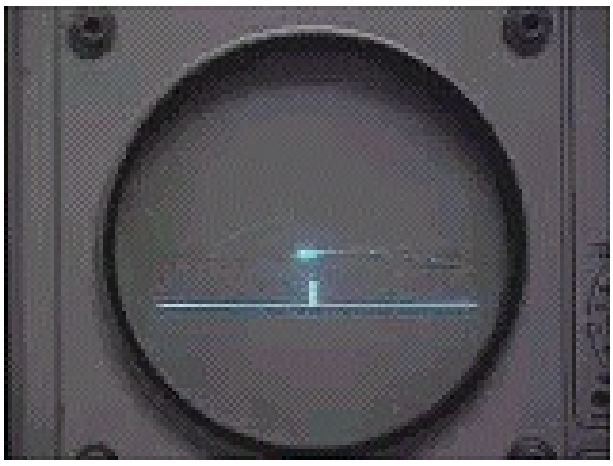

Figura 2.4: Tennis for Two - 1958

2.2 Década de los 60

Fue en esta década cuando se empezó a desarrollar el primer juego destinado al uso doméstico. En 1966 se empezó el desarrollo de ´Fox and Hounds´, el proyecto evoluciono hasta que en 1972 se lanzó con el nombre de ´Magnavox Odyssey´, el cual fue el primer sistema que podía conectarse a la televisión y ofrecía jugar a diversos juegos guardados en la memoria.

Figura 2.5: Magnavox Odyssey - 1968 hasta 1972

2.3 Década de los 70

En la década de los 70 fue cuando los videojuegos realmente comenzaron a funcionar como industria, ya que en 1972 se presentó Pong, la versión comercial de ´Tennis for Two´.

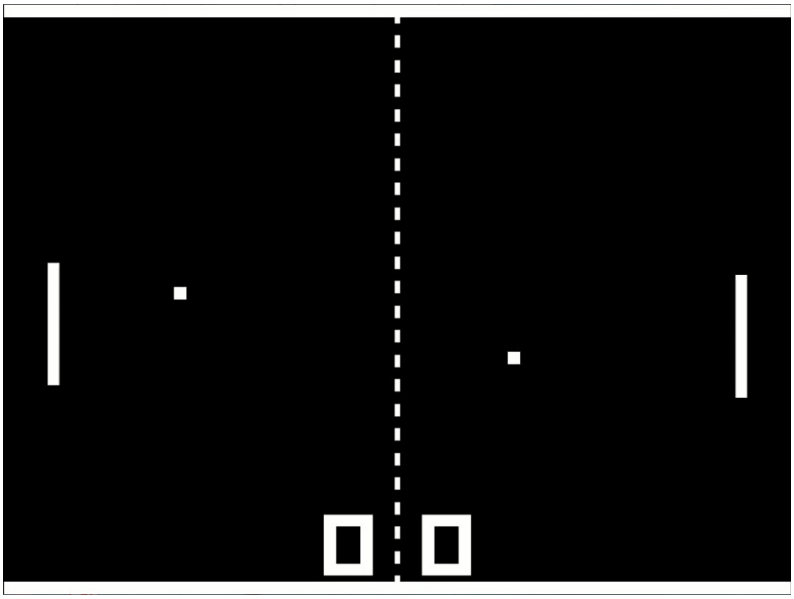

Figura 2.6: Pong

Además, en esta década se contó con numerosos avances en el campo de los ordenadores personales que también se aplicaban a los videojuegos (En especial el Microprocesador y los chips de memoria).

2.4 Década de los 80

En los años 80 se produjo un gran crecimiento de la industria del videojuego en todo el mundo, causado principalmente por el éxito de los salones recreativos y la aparición de las videoconsolas en la década de los 70. En esta década salieron juegos como Zelda, Tetris, Pacman, Donkey Kong, Mario Bros, etc.

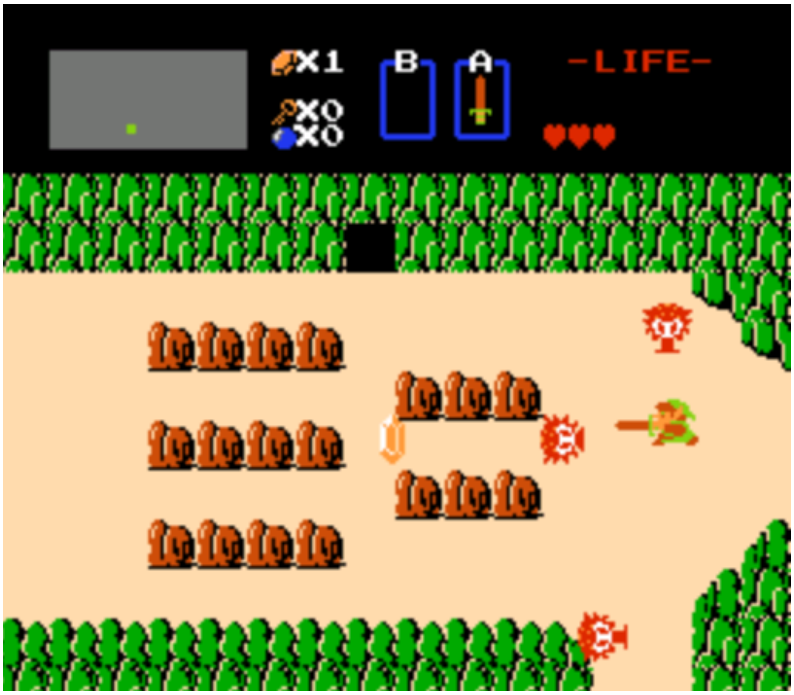

Figura 2.7: The Legend of Zelda - 1986

Hasta 1985 los juegos eran en su mayoría juegos arcade destinados a conseguir la mayor puntuación posible. Todo cambio con la salida de Mario Bros en 1985, en la que por primera vez un juego tenía un inicio y un final.

2.5 Década de los 90

A principios de esta década podemos destacar el comienzo del 3D propiciado gracias a las consolas de 32 y 64 bits, también esta fue la década en la cual surgió la tecnología del CD-ROM.

Además, en esta época ya se podía empezar a ver la diferencia de juegos entre plataformas y la popularidad de los mismos. Mientras en PC eran muy populares los FPS (First

Person Shooter) y RTS (Real Time Strategy) en consola encontrábamos juegos como Final Fantasy VII, Metal Gear y Gran Turismo.

Figura 2.8: Metal Gear Solid

Además, gracias a la conexión a internet en PC se facilitó mucho el juego multijugador, surgiendo así los primeros MMO como por ejemplo Ultima Online.

Figura 2.9: Ultime Online - MMORPG

2.6 Década de los 2000

En esta generación podemos encontrar consolas como la PlayStation 2, la primera Xbox o la Gamecube. El 3D ya estaba totalmente establecido como base para los juegos y cada año el número juegos y de jugadores crecía de forma exponencial.

Esta generación dejó juegos como GTA III, siendo este uno de los primeros mundos abiertos conocidos, además de ser uno de los primeros juegos sandbox. Está reconocido por desarrolladores de esa misma época que este juego volvió a cambiar las reglas. Desde 1985 con Mario Bros los juegos habían tomado la dirección de añadir un principio y un final como base, GTA III también disponía de una historia principal, pero te permitía hacer toda clase de cosas con las mecánicas del juego.

Figura 2.10: GTA III

Además, esta misma década surgió World of Warcraft, el MMORPG más jugado del mundo y que sin duda marco una época para millones de personas, siendo incluso 20 años después un juego en el que se apoyan muchos otros juegos actuales.

Figura 2.11: World of Warcraft

2.7 Década de 2010

En estos años los videojuegos continuaron creciendo a un ritmo exponencial, con un gran auge por los juegos multijugador en red. World of Warcraft continuaba existiendo, pero además podríamos ver juegos como League of Legends, Minecraft o en los últimos años de la década, Fortnite.

Figura 2.12: League of Legends

Todos ellos fueron juegos ampliamente jugados por millones de personas en todo el mundo, recaudando grandes sumas de dinero cada año.

En esta década también encontramos el comienzo de la tecnología VR utilizada en videojuegos, algunos fabricantes empezaban a sacar sus gafas, pero las más conocidas mundialmente fueron las Oculus.

Figura 2.13: Beat Saber - Oculus Quest

2.8 Actualidad

La actualidad y los últimos años de la década pasada se asemejan bastante, debido al Covid-19 la industria de los videojuegos a la vez que muchas otras sufrieron una ralentización y aunque a finales de la década pasada la nueva generación de consolas ya estaba en el mercado, la falta de unidades a la venta y los efectos del Covid-19 han hecho que aún no haya juegos que exploten su potencial. Aun así, disponemos de lanzamientos recientes muy realistas como Gran Turismo 7 u Horizon Zero Dawn 2.

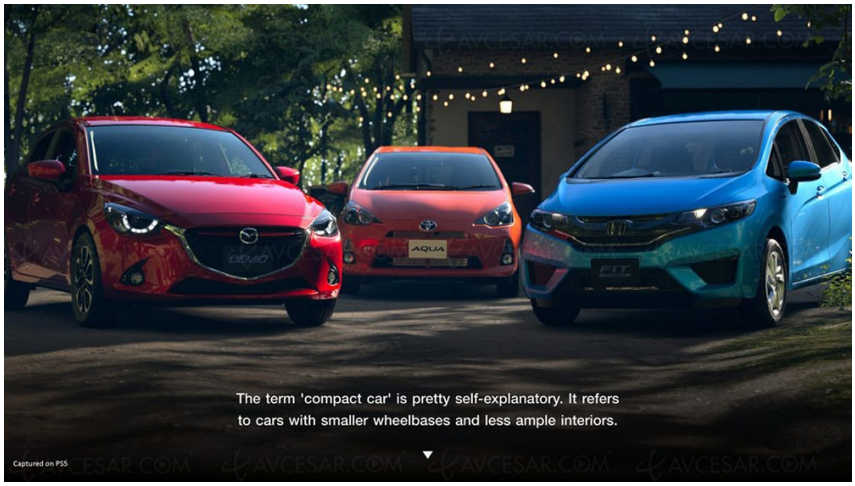

Figura 2.14: Gran Turismo 7

Además, desde hace unos pocos años, incluyendo también los últimos años de la década pasada. Podemos encontrar multitud de remakes de juegos de los 80 en adelante hechos con el objetivo de adaptarlos a los gráficos de las nuevas consolas.

2.9 Propuesta

Muchos trabajos de fin de grado se han hecho con la premisa de conseguir realizar un juego siguiendo ciertos tutoriales oficiales y han estado muy centrados en conseguir algo funcional, quedándose con eso como fin. Me gustaría enfocar mi TFG hacia la preproducción de los videojuegos, centrándome en la investigación previa para saber si un videojuego puede ser viable en el mercado hasta el punto de obtener rentabilidad por él. A su vez, me gustaría enfocar el diseño del videojuego desde una perspectiva algo más compleja, centrándome en puntos como patrones de diseño utilizados por desarrolladores con experiencia o en reutilizar assets de la store para acelerar el desarrollo.

De esta forma podré complementar el resto de trabajos efectuados que tienen que ver con la creación de un videojuego desde cero.

3 Análisis del problema

En este punto, y ya habiendo realizado investigación previa sobre el contexto tecnológico, podemos observar que las variables para que un juego pueda ser rentable y así generar beneficios económicos y además hacer que reciba buenas críticas por parte de los usuarios han variado enormemente en el tiempo. Por ejemplo, hoy en día no se concibe un juego en el cual no guardes la partida de alguna forma, mostrando así una progresión y en las décadas de los 50 y los 60 esto ni siquiera se planteaba.

Además, el aumento desmesurado de la competencia en el mercado hace que el desarrollo sea una tarea cada vez más difícil, haciendo que el factor suerte realmente exista y tenga un peso relativamente alto. El número de juegos que salen a la venta crece exponencialmente cada año. Según SteamSpy, una plataforma desarrollada por Steam (la mayor plataforma de videojuegos del mundo). En 2021 salieron 10.429 juegos para esta plataforma, esto hace un total de 28,57 juegos diarios lanzados y, si contáramos los juegos lanzados en los mercados móviles, tendríamos que multiplicar esta cifra por 10. Por supuesto, la mayoría de estos juegos fracasan o no son lo suficientemente rentables para los creadores, pero, todos los desarrolladores luchan para sus juegos sean visibles, por lo que siguen siendo competencia directa en el mercado.

El factor suerte es reconocido por muchos desarrolladores famosos y tiene tanto peso que incluso hay casos en los que un juego que ha terminado siendo rentable para el estudio se ha descubierto años después de su lanzamiento. Por ejemplo, 'Among us', este fue un juego lanzado en 2018 que no vio el éxito hasta 9 meses después, cuando un streamer famoso lo retransmitió en Twitch haciendo que finalmente tuviese millones de euros en beneficios.

Figura 3.1: Among Us - 2018

Por lo tanto, encontramos diversos problemas listados a continuación:

- Resaltar el juego a desarrollar entre todos los publicados diariamente.
- Realizar un juego viable en el mercado actual.
- Planear un juego realizable de forma individual y en un marco de tiempo realista.
- Afrontar las posibilidades de que el juego desarrollado fracase y gestionar el riesgo que supone.
- Seleccionar la tecnología adecuada para acelerar el desarrollo.
- Desarrollar un calendario viable para las tareas a realizar.

Será necesario lidiar con todos los problemas encontrados y además con los que se encuentren en fases próximas de desarrollo con el fin de minimizar el factor suerte, podemos afirmar con certeza que el factor suerte se verá decrementado en función de la resolución de los problemas arriba expuestos. Para llevar a cabo las soluciones pertinentes será necesaria una correcta preproducción del juego, la cual se documentará en la sección correspondiente

3.1 Mentalidad

La mentalidad con la que se enfoca un videojuego es algo muy importante, dado que las estadísticas se encuentran en nuestra contra desde el principio. Hay que tener claro a lo que se puede optar desde el momento en el que se empieza a plantear el desarrollo de un videojuego, ya que los análisis dicen que lo más probable es un fracaso enorme en los primeros juegos que se preparen (Tyroller, 2021). Pero, por supuesto, que hay casos de éxito en primerizos en la industria, como es el caso de algunos de los autores estudiados para realizar la preproducción

Figura 3.2: Stardew Valley - Primer juego desarrollado por Eric Barone

Un videojuego es un desarrollo largo que involucra años de trabajo, por lo que debemos prepararnos mentalmente para las dificultades que puedan surgir y que surgirán en el proceso de desarrollo. Wlad Marhulets (Creador de Darq) o Jardar Solli (Creador de Seablip) relatan en libros como GAMEDEV (Marhulets, 2020) la importancia de realizar cambios de mentalidad cuando se desarrolla un videojuego y de como es una de las partes más relevantes para poder finalizar el desarrollo. Por mucho que en parte sea similar al campo de software, el cual es mayormente dominado por el grado de ingeniería informática, también se involucran muchos más campos como audiovisual o el artístico y además todo se debe plantear desde una perspectiva que plantee vender el resultado, lo cual incrementa mucho la necesidad de aprendizaje en campos diferentes al propio.

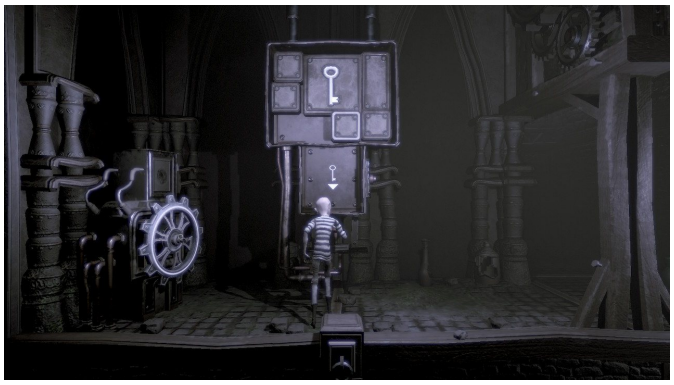

Figura 3.3: Darq - Primer juego desarrollado Wlad Marhulets

3.2 Gestión de riesgos

Cualquier proyecto involucra riesgos y por supuesto un videojuego también los tiene, se han enumerado los riesgos principales en la siguiente lista:

- Plantear un desarrollo demasiado largo
- Quedarse sin fondos en medio de un desarrollo
- No recuperar la inversión inicial al lanzar el juego

Como se puede observar, los diferentes riesgos planteados tienen que ver directamente con la gestión de recursos disponibles, que en este caso son tiempo y dinero.

Muchos desarrolladores veteranos coinciden en que es mejor no apuntar muy alto para no usar muchos de estos recursos en el desarrollo del primer juego, puesto que es probable que sea un fracaso. Este consejo se asegura de cuidar financiera, física y mentalmente al desarrollador. Parece objetivamente mejor hacer varios juegos de calidad que sean pequeños y estén pulidos, con los objetivos de aprender de los errores cometidos y familiarizarse con el proceso del desarrollo antes de realizar un juego grande.

Con este tipo de prácticas podemos asegurarnos de asumir un riesgo controlado, puesto que no es lo mismo perder 3 meses, aprender de los errores y tomar una nueva ruta haciendo un nuevo juego, que fallar después de un desarrollo de 3 años.

Aunque se quisiera no contratar a nadie y aprenderlo todo desde cero, el tiempo no es infinito y no se pueden aprender todas las habilidades del mundo, por lo que en el desarrollo del videojuego tendremos que asumir riesgos controlados con el objetivo de controlar los recursos y utilizarlos de la forma más óptima posible.

3.3 Gestión del tiempo

Muchos libros sobre desarrollo de videojuegos (Marhulets, 2020) (Schreier, 2017) o estudios sobre desarrollo de software (Levenson, 2021) hacen especial hincapié en la gestión del tiempo. En una tarea tan larga como el desarrollo de un videojuego, prima la necesidad por saber gestionar el tiempo y poder dar fechas de entrega (incluso a uno mismo si se trabajas solo).

Esto puede ayudar mucho a evitar el perfeccionismo con el que muchos desarrolladores contamos. Cuando estaba desarrollando el juego final en la asignatura optativa de cuarto sobre desarrollo de videojuegos, cometí el error de pasar más de cuarenta horas haciendo y moviendo assets en un mapa del prototipo. La realidad es que habían muchas otras cosas que hacer y la fecha de entrega se acercaba. Mis tareas no estaban acabadas y desde luego no estaba en una fase donde poderme permitir mejorar los acabados.

Gestionar el tiempo no quiere decir acabar con un juego mediocre porque no se le ha dedicado el tiempo suficiente, lo que quiere decir es que hay que saber qué priorizar y en

qué momento priorizar. Si se tuviese un año para hacer un juego rítmico, posiblemente mucho tiempo se usara en componer las canciones del juego o en mejorar el sistema de ritmos del mismo, en un juego de rol se invertirá mucho tiempo en la historia o en el sistema de batalla, en un juego de terror se invertirá mucho tiempo en los escenarios, Etc.

Hacer juegos también influye en establecer límites y poner fechas, puede parecer un trabajo muy artístico en el cual no hay que apresurarse. Pero la realidad es que es necesario maximizar el uso del tiempo e invertir el mismo en las cosas que de verdad importan. Establecer tareas y objetivos ayudará mucho a que el desarrollo de algo tan largo sea un éxito.

Algunos de los consejos proporcionados por autores que han triunfado son los siguientes:

- Utilizar un diario para monitorizar como se usa el tiempo y así saber cuanto tiempo se invierte en tareas triviales con el objetivo de ajustarlas al calendario.

- Antes de lanzarse con una idea dentro del juego y empezar a desarrollarla por completo o a buscar assets es mejor crear un prototipo rápido que no lleve más de un día. De esta forma se podrá observar y probar la idea en el juego antes de desarrollarla completamente. También es posible hacer tests con jugadores y ver que opinan de la idea.

- Establecer metas diarias y semanales puede ayudar a que el perfeccionismo no acabe con el tiempo total del juego. Los desarrolladores muchas veces tenemos cierta conexión emocional con lo que creamos y queremos que se vea de la mejor forma posible, esto resulta en que podemos invertir demasiado tiempo en fases tempranas del ciclo de desarrollo.

- Hacer sacrificios, muchas veces se quiere que el juego tenga de todo. Se planean sistemas imposibles o irrealizables por estudios pequeños. Cuando envíe a mi tutor la idea que tenía dije que quería hacer un sistema de clima, aventuras en función del mismo, sistemas de rol como los de mesa, Etc. La realidad es que en ese momento no había tomado en serio el desarrollo del juego, no había empezado a desarrollar del todo la idea y pensaba que tendría todo el tiempo del mundo para hacerlo. La realidad es que el juego que deseaba realizar podía simplificar muchos de estos sistemas y seguir siendo muy viable.

- Cuando exista un nivel jugable o un prototipo, seria óptimo realizar una prueba. Lo mejor es que lo prueben desconocidos. De esta forma se podrá obtener un feedback real del juego y se podrá obtener información sobre qué sistemas son divertidos, cuáles son demasiado complejos y detectar problemas en fases tempranas del desarrollo.

- Cuando se encuentre algo en lo que no somos buenos, es bueno considerar el delegar tareas , en mi caso soy programador, pero no se demasiado sobre arte. Por lo cual si comprara los assets de mi juego ahorraría mucho tiempo de desarrollo. Por supuesto hay que tener en cuenta los riesgos que esto puede traer. Gestionar el riesgo también influye en saber cuanto es viable gastar si tienes pruebas tempranas de que a la gente le gusta el juego y están dispuestos a pagar determinado dinero por su versión final.

3.4 Preproducción

Para poder realizar esta sección correctamente y de forma realista se han seguido numerosos casos de éxito en juegos desarrollados por estudios pequeños y documentados en libros como 'GAMEDEV' (Marhulets, 2020). Posiblemente, esta sea la parte más importante del desarrollo de un videojuego, la cual incluye la preparación, los estudios previos, el hecho de ver si el juego será viable en el futuro, Etc. En este momento es en el cual nos hacemos las preguntas básicas del juego: ¿Qué juego vamos a hacer?, ¿Qué género tendrá? ¿En qué plataforma lo lanzaremos?, ¿A quién va destinado?, ¿Cuál será la estrategia de monetización?.

Lo más probable si queremos hacer un juego es porque somos consumidores y nos dejemos llevar por una idea romántica sobre el desarrollo, pero hacer un juego viable no es nada fácil. La industria evoluciona muy rápido, las modas cambian cada año, por ejemplo, rondando los años 2016 y hasta más o menos 2019, podemos decir que la moda estuvo rondando los juegos con temática battle royale. En los últimos años, parece que los juegos de las empresas AAA tienden a desarrollar un mundo abierto muy grande aprovechando el potencial de las nuevas consolas y que ahora incorporan discos M.2 y memoria RAM DDR5 para hacer realidad cargas en tiempo real.

Figura 3.4: Fortnite - 2017

Figura 3.5: Stray - 2022

Planear todas estas cosas desde ahora va a hacer que nuestras posibilidades de éxito se incrementen y evitemos invertir nuestro tiempo en algún desastre irrecuperable.

3.4.1 El juego a desarrollar: ´Paranoia´

Este es el momento de definir el juego a desarrollar, puesto que posteriormente se realizaran los estudios de mercado para saber si puede llegar a ser viable. El juego estará basado en el rol de 'Paranoia', la idea de este tipo de rol surgió del mundo de ciencia ficción creado por Daniel Seth Gelber, Greg Costikyan, Eric Goldberg y Ken Rolston.

Figura 3.6: Juego de mesa de paranoia

Este mundo ya ha sido adaptado varias veces en diversos juegos de mesa, otros videojuegos y en historias aplicadas a libros o el cine, pero siempre manteniendo la misma idea principal. En este caso, la historia y el estilo gráfico se adaptarán para que puedan casar con el target escogido.

- El juego a realizar.

  - En un principio, el juego que me plantee hacer era un RPG lineal con múltiples finales en función de las decisiones tomadas durante la aventura, asemejando así una historia que podría suceder si se jugara a una partida de rol de mesa sobre paranoia. Pero, finalmente, me decanté por un juego estilo RPG con mecánicas sandbox, eventos aleatorios y en el que la muerte del personaje es el final de la historia. Si el personaje muere, podrás crear otro personaje para empezar de nuevo en el mismo mundo que dejo el personaje anterior. El juego aparentemente

no tiene fin o un objetivo final muy establecido, pero estará centrado en la supervivencia en un mundo en el cual todos quieren traicionarte por diversos motivos y en el que no puedes fiarte de nadie. El factor diferenciador será descubrir las historias del mundo e ir avanzando por él, tener la capacidad de convertirse en un aristócrata o en un renegado en función de las decisiones tomadas y poder desarrollarse en cualquiera de los dos bandos.

- Target del juego

  - Todo el juego será diseñado para personas de 18 a 34 años, el rango de edades con un mayor número de jugadores. De esta forma, el juego podrá ser atractivo para las personas que vivieron el nacimiento de paranoia en sus inicios. A su vez, con este rango de edad nos aseguraremos que una historia compleja como la que se plantea pueda ser un factor diferenciador que ayude al juego, siempre asumiendo que la edad de los jugadores será suficiente para poder entender la trama.

- Arte del juego

  - Se utilizará pixel art como estilo de dibujo para así rememorar los juegos que el rango de edad seleccionado jugo en su infancia.

- Historia principal del juego a realizar

  - En el año 2035, se produjo una gran hambruna mundial provocada por la falta de recursos, el cambio climático y la superpoblación. Esto provoco que las mayores potencias del mundo lucharan por las pocas zonas habitables, haciendo que comenzara la tercera guerra mundial. La cual en solo una semana produjo una extinción masiva a causa de las armas nucleares lanzadas entre los países y dejando cualquier zona del mundo completamente inhabitable. Esto obligo a la humanidad a refugiarse durante 13 años debido a la radiación.

    Finalmente, en el año 2048. Los niveles de radiación volvieron a ser aceptables en determinadas partes del mundo, pero en este punto solo quedaban cerca de 3.000 humanos restantes en la isla de Australia y las comunicaciones globales estaban totalmente destruidas. Por si todo esto no fuera suficiente, durante los primeros años de refugio la gran parte de la fauna y flora se extinguió. Pero parte de esta logro sobrevivir en conjunto con la radiación mutando en biomas y criaturas totalmente nuevas que suponían nuevos peligros para el resto de la civilización.

    Para que la guerra no se repitiese de nuevo y con temor a que un nuevo gobierno mundial repitiese los errores del pasado, se designó como nuevo líder supremo a un ordenador. El cual estaba programado con la más sofisticada inteligencia artificial posible. La cual tenía como objetivo garantizar la supervivencia de la especie e impedir que el ser humano volviera a entrar en guerra. Para ello, el

ordenador fundo ´La corporación´, una pequeña ciudad que agrupaba a todos los habitantes restantes.

Además, el ordenador promovía mandatos para todos los ciudadanos, si estos mandatos eran incumplidos los ciudadanos obtenían puntos de traición. En el caso de que un ciudadano superara cierto umbral de puntos hacía que fuera condenado a trabajos forzados o a la pena de muerte. El ordenador esperaba un comportamiento modélico por parte de todos los ciudadanos y no dudaba en acabar con quien fuera una amenaza para la preservación de la especie.

Con el tiempo, el ordenador creo departamentos para mantener y garantizar los recursos e hizo que toda la humanidad restante trabajara en puestos públicos. Cualquier anomalía fuera de este sistema sería descartada. Además, el ordenador designaba el departamento al que pertenecía cada ciudadano en función de sus capacidades y se aseguraba con ello de mantener los niveles de vida dentro de lo aceptable, garantizando alimento y seguridad dentro ´La corporación´.

En unos años la civilización volvió a ser estable, pero los continuos ataques de la nueva fauna y la falta de recursos hizo muy difícil la defensa y la expansión de la ciudad. En este punto, el ordenador instauro un mandato para que solo él pudiera generar nuevos habitantes de ´La corporación´, conservando y clonando lo que él consideraba los humanos con el mejor ADN posible. Con esto conseguía que la humanidad no se expandiera sin control y así garantizaba los recursos. El ordenador era capaz de realizar clones en edad de trabajar, con lo que pronto dejaron de haber niños. Este mandato hizo que un grupo de habitantes se manifestaran en contra del ordenador. Pronto, la mayoría fueron juzgados y condenados a morir. El resto escapo e intento vivir al margen, siendo perseguidos por el departamento de seguridad.

Aun así, el conflicto no abandono ´La corporación´. Los mandatos del ordenador eran cada vez más duros y siniestros, obligando a todo el mundo a ser feliz por obligación y condenando a los infelices a morir. Con esto surgieron más ciudadanos que querían cambiar el sistema de gobierno, pero en este punto también había fanáticos que adoraban el ordenador. A raíz de estos incidentes, el ordenador instauro las clases sociales que premiaban a los que veneraban y seguían las instrucciones del ordenador con puestos más altos en los departamentos y hacía que las clases sociales más bajas tuvieran que seguir las órdenes de las más altas. Para identificar a los ciudadanos, en el momento de su nacimiento se les daba una placa con su estado social y su nombre. Falsificar esta placa o no llevarla visible en todo momento era motivo de traición. Los conflictos hicieron que se fundarán pequeñas sociedades secretas dentro de ´la corporación´, cada una tenía diversos objetivos, pero todas en menor o mayor medida querían acabar con el ordenador.

Pasado un tiempo, el departamento de sanidad detecto que algunos ciudadanos mostraron capacidades sobrenaturales, esto era provocado por mutaciones debidas al exceso de radiación en el ambiente. Se detectaron mutaciones de todo tipo: superfuerza, regeneración, rayos-x, polimorfismo, piroquinesis, telepatía,

etc. Todas estas mutaciones fueron estudiadas en profundidad y en poco tiempo el ordenador dicto un nuevo mandato en el cual el hecho de ser mutante era considerado traición. Además, por mandato del ordenador, el departamento de inteligencia desarrollo joyas que otorgaban a su portador algunos de estos poderes. El plan era dárselas a los ciudadanos con mayor clase social y solucionar los problemas que estaban habiendo con la seguridad y la expansión.

- Ejemplo de partida

  - Se empieza una partida, avanzas por el entrenamiento base de la corporación y decides que quieres ser parte del departamento de seguridad. Una vez dentro, aceptas una misión en la que tienes que derrotar monstruos mutantes afectados por la radiación, pero trágicamente el equipo muere en esta misión. Los monstruos entonces arrasan un puesto de mercadería. Decides empezar de nuevo con otro personaje, en este caso el puesto de comida habrá sido destruido por unos monstruos mutantes y podrá ser restaurado por el departamento de obras y servicios.

3.4.2 Análisis de viabilidad

Lo primero que debemos hacer en el momento en el que sabemos que juego vamos a crear, va a ser preparar un estudio basado en diversas preguntas propuestas por el creador del juego DARQ y secundadas por muchos desarrolladores de juegos indie. Estas preguntas nos darán pistas sobre si el juego expuesto puede ser viable en el mercado y podrá generar beneficios que justifiquen los costes de desarrollo.

- ¿Cuáles son las modas del mercado actualmente?

  Analizar las modas es lo primero que debemos hacer si queremos hacer que la gente compre nuestro juego. Además, hay que ver que partes del mercado se analizan sabiendo qué tipo de juego queremos hacer. En mi caso, como se dijo al principio de la sección, quiero hacer un juego RPG que pueda competir con los juegos del mismo tipo.

  - Ventas por juegos desarrollados por estudios indie:

    Podemos afirmar según diversas tiendas online como PlayStation Store, Steam, Epic Games, Etc. Y según estudios demográficos que analizan las modas (Howarth, 2022), que los juegos indie son y seguirán siendo populares en el futuro. Estas tiendas muestran que los juegos desarrollados por estudios indie tienen mercado, y aunque muchos de ellos no puedan competir en ventas con juegos AAA los costes de desarrollo son mucho menores y los equipos mucho más pequeños.

    En conclusión, cubrir los costes del desarrollo y hacer que sean rentables para el estudio requiere de mucho menos dinero.

- Diversidad en juegos actuales:

  Podemos ver como la diversidad está cambiando. No es ningún secreto que la gran mayoría de jugadores hasta ahora han sido destinados a un público masculino, pero hay ciertos juegos, como por ejemplo 'Animal Crossing', que han atraído una gran parte del público femenino hacia ellos. De hecho, se estima que en 2025 casi la mitad de jugadores de videojuegos serán mujeres.

Figura 3.7: Animal Crossing

A la par que la sociedad, la industria también está cambiando, y cada vez ofrece más representación a las minorías en los mundos creados. Por ejemplo, en 2017 fue lanzado Kingdom Come: Deliverance. El juego obtuvo buenas reseñas, pero también fue criticado por la falta de representación en el juego. Esto nos dice que si el público objetivo del juego no es un sector determinado de la población y está enfocado a todos los públicos, deberemos hacer en la medida de lo posible una buena representación de las minorías en él.

Figura 3.8: Kingdom Come: Deliverance

- Aumento en el cloud gaming:

  También podemos ver la tendencia al aumento del cloud gaming:

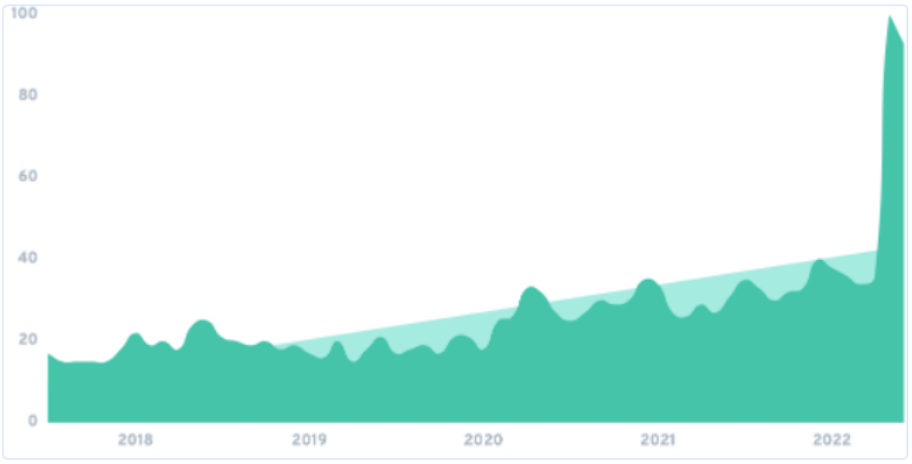

Figura 3.9: Expansión de servicios cloud

  Gracias a que las conexiones de internet han mejorado mucho, los servicios de cloud gaming permiten a los usuarios jugar por streaming pagando una suscripción mensual. Esto permite a los usuarios usar juegos muy exigentes en procesamiento sin ningún problema a pesar de tener una computadora desfasada.

  Incluso si en el juego deseado no se utiliza mucho procesamiento, con este dato sabemos que hay jugadores que no tienen una computadora especialmente potente en sus casas. Hacer un juego de bajos requisitos puede hacer ganar público entre estos sectores.

- Remakes, relanzamientos y posibilidad de abogar a la nostalgia:

  Lo siguiente a analizar es la subida repentina de remakes de juegos aclamados por la crítica lanzados en otras décadas. Motivos como el crecimiento de los jugadores de videojuegos, las llamadas a la nostalgia y la diferencia en el gameplay y el estilo de los juegos más antiguos hace que estos sean muy atractivos para tanto los jugadores jóvenes que no jugaron esos juegos en su lanzamiento como para los que ya los habían jugado anteriormente y quieren recordarlos. Juegos como el remake de Crash o de Final Fantasy VII han sido superventas y se prevé que esta tendencia siga aumentando.

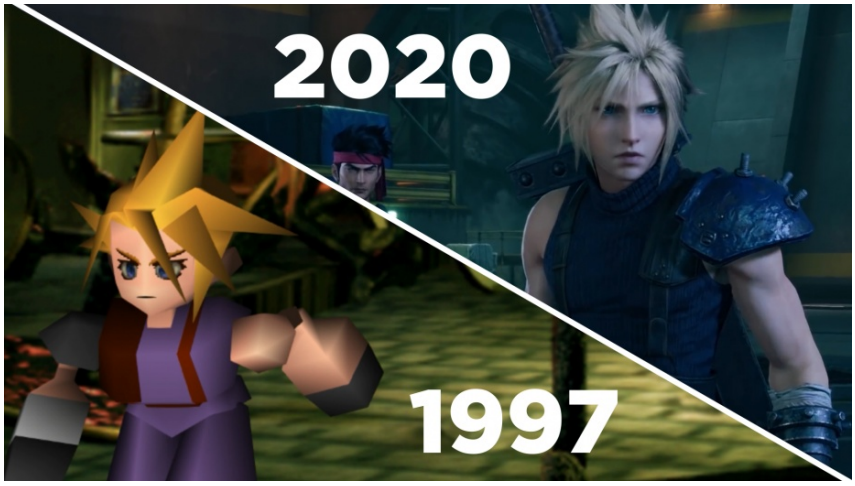

Figura 3.10: Final Fantasy VII Original vs Remake

Figura 3.11: Crash Original vs Remake

Figura 3.12: Live a Live Original vs Remake

Aparte de los remakes, también ha habido una serie de relanzamientos en los últimos años, los cuales no mejoran en gráficos o en gameplay, pero permiten jugar a los juegos antiguos en consolas modernas o en PC. Algunos de estos juegos también han sido un éxito rotundo, como por ejemplo Grandia o Chrono Cross, ambos lanzados en switch el último año.

Figura 3.13: Chrono Cross Switch

Figura 3.14: Grandia HD Switch

Por último, existen juegos como Triangle Strategy, Stardew Valley u Hollow Knight que han sido lanzados recientemente, pero que siguen la estética de juegos más antiguos, usando gráficos 2.5D o 2D. Estos juegos también han funcionado extraordinariamente bien.

Figura 3.15: Stardew Valley - 2D - 2017

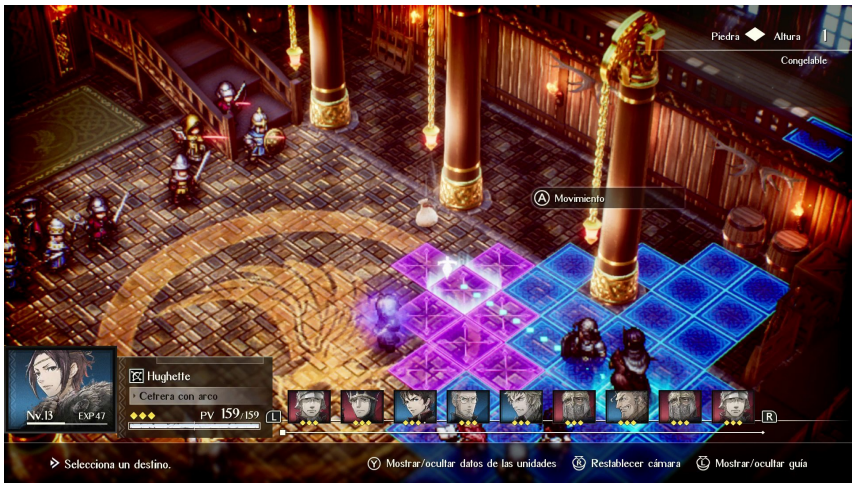

Figura 3.16: Triangle Strategy - 2.5D - 2022

En conclusión, podemos afirmar que lo que hizo que un juego vendiese en otra década puede volver a funcionar hoy en día. A pesar de que los gráficos no utilicen las últimas tecnologías, hay una gran cantidad de jugadores dispuestos a dar una oportunidad a juegos con gráficos menos exigentes.

- Gran aumento de juegos lanzados en Early Access:

  Con el aumento del mercado indie existe una necesidad de financiación por vías alternativas, por lo que también aumenta los juegos que deciden emplear este tipo de recurso. El early access permite que los desarrolladores obtengan fondos y críticas previas a versiones finales del juego. Los jugadores pueden obtener el juego a un precio más bajo y además jugarlo previamente cuando aún no está acabado y aún se está desarrollando. Además, los desarrolladores pueden usar las críticas para cambiar dirección del juego para mejorarlo acorde a su público actual.

- ¿Qué juegos se están vendiendo en las plataformas que quieres abordar?

  Como la tendencia nos dice que la gran mayoría de juegos indie se lanzan primero en PC debido a que el coste de licencia es nulo, debemos ver qué juegos del mismo tipo se están vendiendo actualmente.

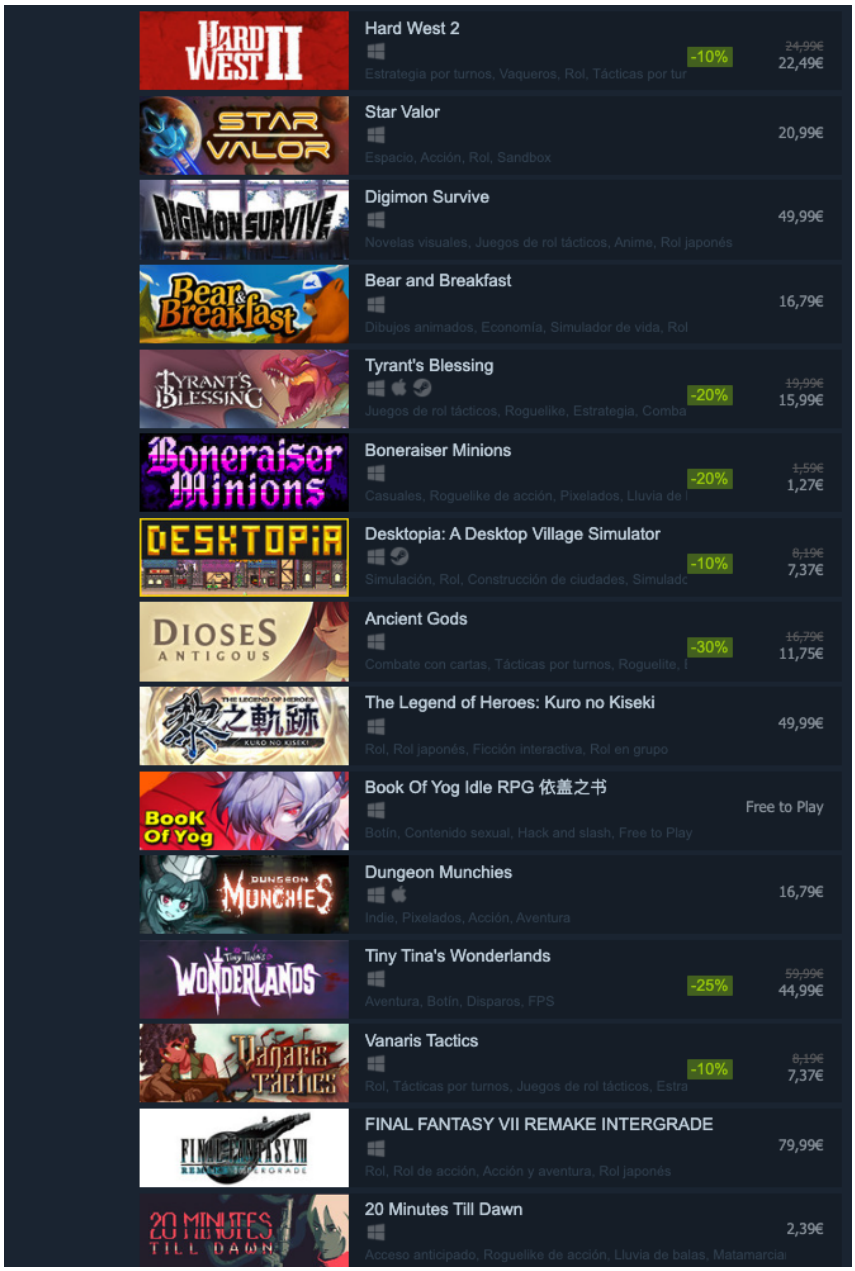

Figura 3.17: Juegos más vendidos en Steam - Agosto de 2022

En la categoría de juegos de ROL podemos encontrar varios juegos AAA como Digimon Survive o Final Fantasy VII Remake Intergrade. Pero podemos ver en su mayoría de juegos en tendencia una gran cantidad de juegos indie como Vanaris Tactics, Bear and Breakfast, Desktopia o 20 Minutes Till Dawn, incluyendo este último early access. Los juegos AAA siempre suelen estar en tendencias como mínimo el

mes en el que salen, pero hay muchos menos lanzamientos AAA que de juegos indie. Por lo que podemos concluir que habría que evitar sacar nuestro juego un mes que haya muchos lanzamientos AAA del mismo género si queremos tener una posibilidad para aparecer en la primera página de tendencias en estas plataformas. Incluso si el público objetivo no es el mismo, aparecer en la primera página de las búsquedas puede hacer que nuestro juego obtenga críticas tempranas, se siga recomendando en las plataformas de venta y que llame la atención de la prensa.

- ¿Qué géneros de juegos tienen un incremento de popularidad y cuáles se han mantenido en el tiempo?

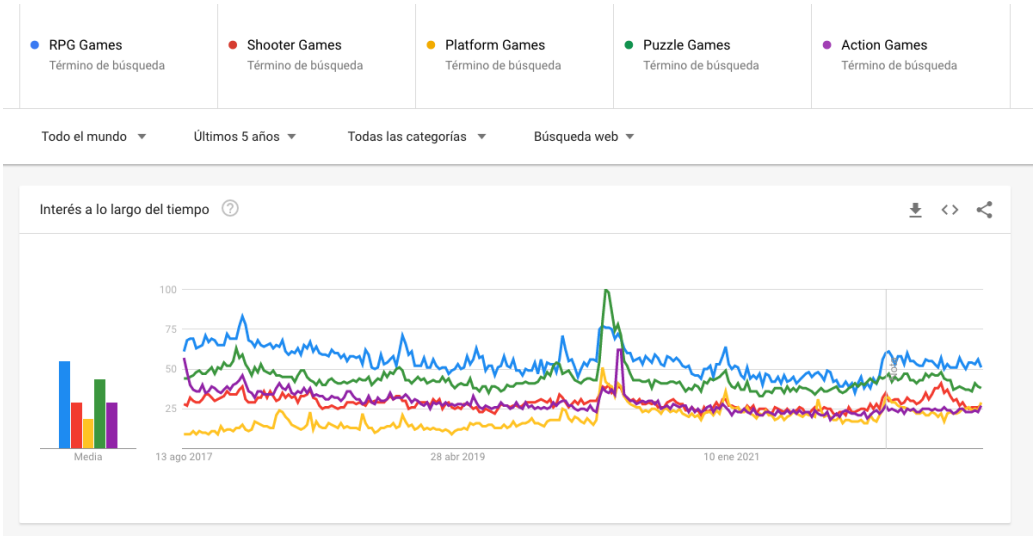

Figura 3.18: Géneros de videojuegos principales - Google Trends

Si comparamos los principales géneros usando la herramienta google trends podemos ver que todos se han mantenido en el tiempo con picos momentáneos que duran unos días en función de lanzamientos populares en esos intervalos. Con los datos obtenidos podemos concluir que estos géneros se mantendrán en el tiempo, siendo el género RPG uno de los más buscados.

- ¿Cuáles son los estudios indie más exitosos? ¿Cómo les está yendo?

  El estudio indie más exitoso es sin duda Mojang que creo Minecraft en el año 2009, siendo este el juego más vendido del mundo. Mojang empezo siendo un estudio unipersonal y acabo siendo adquirido por Microsoft por 2.5 billones unos años más tarde. Por supuesto, no deberíamos tomar Mojang como ejemplo de lo habitual, pero si de lo que ha podido llegar a ser un estudio que empezó como indie.

Figura 3.19: Mojang - Estudio indie

Otro estudio muy famoso es Supergiant Games creadores de juegos como Bastion, Transistor o Hades, todos ellos muy aclamados por la crítica.

Figura 3.20: Supergiant Games - Estudio indie

Otro caso de éxito fue Team Cherry, creadores de Hollow Knight, además, este es otro caso en el cual el primer juego hizo triunfar al estudio.

Figura 3.21: Team Cherry - Estudio indie

Hay muchos casos de éxito entre estudios indie, otro éxito rotundo fue Concerned Ape con Stardew Valley o PlayDeath con Limbo. Todos ellos han ofrecido juegos muy diferentes a lo que suelen ofrecer los estudios AAA, pero los juegos que han acabado teniendo éxito tenían factores que los hacían diferenciales y se sentían nuevos.

- ¿Qué hace que estos estudios sean exitosos?

  Analizando estudios como SuperGiant Games, podemos decir que en cada juego intentaron mejorar lo anterior, todos los juegos que han sacado tienen temáticas diferentes, pero ofrecen un sistema de batalla muy similar. De hecho, este estudio podría ser utilizado como caso de estudio debido a que han ido en una dirección totalmente ascendente desde su creación, utilizando la misma fórmula y mejorándola en cada juego.

  Concerned Ape o Team Cherry intentaron hacer que su juego estuviera lo más pulido posible, explotando además ciertos sentimientos con los mapeados y bandas sonoras. En ningún juego se puede sentir falta de contenido, incluso si han sido desarrollados por pocas personas. El bucle de juego está muy conseguido en ambos casos, haciendo que no te aburras al jugar y que ambos tengan fácilmente más de 30 horas de juego.

- Tomando en cuenta un juego similar al que quieres hacer ¿Cuánta gente hizo falta para hacerlo?

  En mi caso, quiero hacer un juego RPG basado en rol de paranoia que incorpore mecánicas de aventuras, supervivencia, peso en las decisiones y además mecánicas sandbox. Los juegos AAA que me han inspirado serian juegos como la saga Persona, la saga Xenoblade, la saga Elders Scrolls, los Goldensun, los primeros Suikoden y juegos como Fantasy Life. También algunos juegos indie como Eastward, Starbound, Kenshi y Stardew Valley.

  En el caso de los AAA, todos contaron con estudios grandes y muchas personas destinadas a su desarrollo, los estudios indie, por otro lado, contaron con pequeños equipos o en el caso de Stardew Valley a solo una persona destinada a su desarrollo.

  En cualquier caso, todos ellos tardaron varios años en desarrollar los juegos, pero los presupuestos para ello fueron extremadamente diferentes. Para los juegos indie en muchos casos las personas encargadas tenían segundos trabajos y dedicaban su tiempo libre a realizar los juegos, en el caso de los AAA los desarrolladores fueron pagados por los estudios con presupuestos de millones de euros.

- ¿Cuáles son las plataformas más populares para tu tipo de juego?

  Por el tipo de juego ya hemos visto en el gráfico de google trends que los juegos RPG son muy populares, incluso entre los géneros más populares, pero podemos analizar las plataformas más en detalle con google trends:

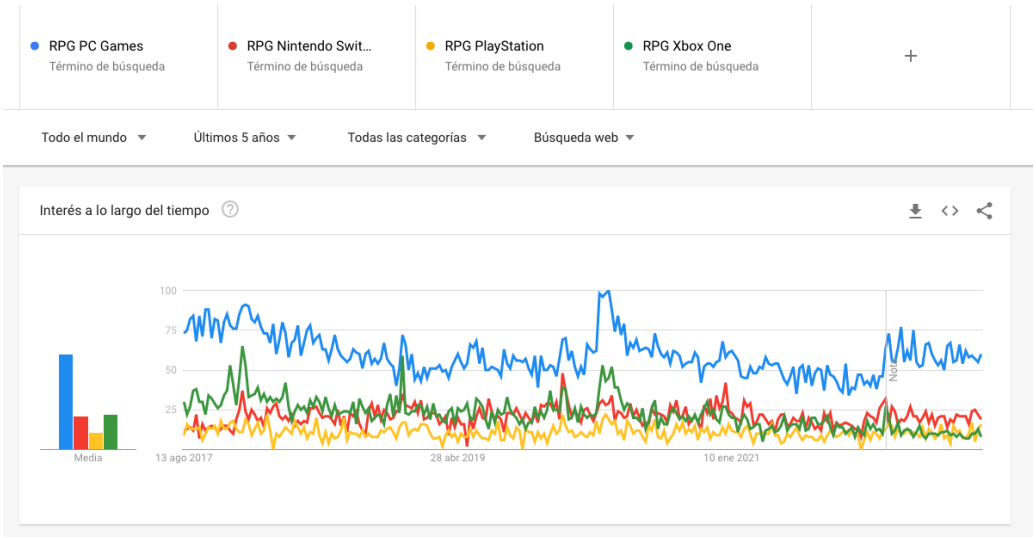

Figura 3.22: Búsqueda de RPG por consolas - Google Trends

La plataforma más popular es la de PC, seguida por las consolas. Con lo cual concluimos que tiene sentido lanzar primero en PC por la sencillez de la plataforma, las posibilidades de utilizar herramientas como el early access y además que el coste de licencia es nulo, a diferencia del resto de plataformas.

- Basándose en el estado actual del mercado, ¿Se pueden anticipar modas futuras?

  En el caso de los géneros principales como pueden ser RPG, shooter o acción, la tendencia indica que seguirán siendo populares con picos esporádicos. En el caso de subgéneros es muy difícil prever como va a comportarse el mercado. El género de battle royale se volvió popular de la nada gracias a algunos juegos, pero no hubo un precedente que dijera que esto sucedería, lo más cercado podría ser el remake de la película que se estrenó en 2012. Pero no fue hasta 2017 que se volvieron populares. En conclusión, se puede decir que apostar por un género principal como RPG puede ser una apuesta relativamente segura en cuanto a lo que al mercado se refiere. Pero el juego puede seguir fracasando si no se le da visibilidad, es muy parecido a otros, no llama la atención o no es divertido, aunque podemos asegurar que el factor género no tendrá nada que ver.

Habiendo hecho estas preguntas, podemos concluir diversos aspectos:

- Mercado:

  - Si hacemos un juego con bajo potencial de mercado, estamos fallando, por lo menos en un sentido empresarial. Además, si intentamos replicar exactamente alguno de los juegos en los cuales nos inspiramos, es muy posible que acabemos con un juego sin sentido, peor que el original y que no llame nada la atención de los usuarios.

- Amplitud del proyecto:

  - Los análisis previos nos ayudan a saber aproximadamente lo que podemos abarcar, por ejemplo hay pocos juegos RPG con historias densas dentro del mundo indie debido a que su tiempo de desarrollo es muy alto. Pero sí que podemos encontrar juegos RPG que incorporan mecánicas sandbox o roguelike y dejan la historia un poco en el aire, o incluso, tienen historias complejas, pero esta es relatada de forma sencilla o a través del gameplay. Un factor diferencial de nuestro juego puede ser por ejemplo una historia completa y que cree un mundo diferencial, pero para hacer esto, es posible que debamos conformarnos acortando otras mecánicas o incluso la duración del juego. De otra forma, podríamos intentar abarcar más de lo posible y hacer que abandonemos el proyecto a mitad del desarrollo.

- Viabilidad a corto plazo:

  - Podemos asegurar que intentar sacar al juego al mercado demasiado tarde o cuando ya esté en fases finales puede hacer que el desarrollo sea un fracaso o no llegue a su fin. Mirando como ejemplo el resto de juegos indie que han tenido éxito, estos consideraban la viabilidad del juego en fases tempranas del desarrollo. No todos estaban disponibles en early access, pero algunos de ellos buscaban financiación en páginas de crowdfunding o utilizaban las redes sociales y aceptaban compras anticipadas justo después de tener los primeros prototipos

del juego. Esto ayudaba a que en proyectos de bajo presupuesto se pudiera dedicar más tiempo al desarrollo y que el juego se fuera moldeando empleando las opiniones de los futuros jugadores.

- El gancho:

  - Podemos referirnos al gancho como a una mecánica que hace que el juego difiera del resto, frecuentemente podemos ver que intentan añadir algún giro a un género determinado. Un buen gancho debe llamar la atención de los jugadores en el momento en el que lo vean, idealmente debe ser novedoso, visual y fácil de entender. Como desarrollador indie esto es algo primordial, debido a que en muchos otros aspectos no se puede competir con estudios más grandes o juegos AAA. Hay que tener en cuenta que no solo competimos con los juegos nuevos, si no, también con los juegos antiguos que ya tienen una base de jugadores y están en la lista de deseados de muchos otros. Hay que darles un motivo para jugar a nuestro juego y para ello hay que ofrecer algo nuevo y que el resto no posean. Por ejemplo, el gancho de Skyrim es que puedes explorar todo en el juego, cualquier armario, libro o incluso los bolsillos de un NPC sin importancia, el gancho de Super Hot es que el tiempo solo se mueve cuando tú te mueves, Etc.

- Duración:

  - Lo mejor que podemos hacer es un juego largo que de a los usuarios muchas horas de juego, rejugabilidad o incluso un juego infinito con programación procedural. Pero si por ejemplo queremos hacer un juego lineal, podemos bajar la calidad de los assets para que sea más rápido producirlos y así poder extender más la duración del juego, también puede ser una buena idea hacer aleatorias ciertas secciones para que cada nuevo juego sea único.

- Plataformas de venta:

  - Si queremos vender más, es mejor que nuestro juego esté en todas las plataformas en las que sabemos que puede ser popular. Pero esto no es tan simple como parece, por ejemplo: los juegos para teléfonos se monetizan de forma muy distinta a los de PC o consola y tienden a estar hechos para proporcionar sesiones de juego cortas, la base de jugadores de Steam son hombres, la Nintendo Switch tiene una base de jugadores de ambos géneros y prefiere juegos para toda la familia. Cada plataforma tiene su propia demografía, género, monetización y barreras para entrar.

Una vez realizados los estudios correspondientes y sabiendo que el videojuego planteado puede ser viable, quedará por analizar el modelo de negocio, la tecnología que se empleara para realizar el proyecto, el presupuesto del mismo y el calendario.

3.4.3 Precio y monetización

Una vez hemos llegado a este punto, necesitamos idear un modelo de negocio que haga que el juego sea rentable y que justifique el tiempo y dinero que se invertirá en el proceso de desarrollo. En este punto ya conocemos el tipo de juego queremos hacer y además por el estudio de mercado sabemos qué juegos similares son capaces de vender, por lo menos, en primera instancia. Así que, se puede investigar por género y tipo, cuanto dinero cuesta este tipo de juegos de media, teniendo en cuenta cosas como horas de gameplay, rejugabilidad, características, etc. Si el juego que queremos hacer es similar, debemos poner un precio que iguale a la competencia. Si el juego no puede competir en contenido, debemos reflejarlo en el precio o plantear añadir más contenido, como se realiza este análisis en fase de preproducción, se plantea simplemente como un primer análisis para ayudar a contrastar el contenido del juego con otros juegos similares y empezar a generar una estrategia de ventas.

Según cuenta el desarrollador de Darq en su libro (Marhulets, 2020), poner un precio más bajo no va a hacer necesariamente que el juego genere más ventas, ya que esto puede hacer que sea percibido como un juego de baja calidad. Dicho esto, los jugadores quieren ser tratados con honestidad. Podemos encontrar muchas reviews de juegos en las cuales los jugadores recomiendan comprar el juego solo si está rebajado, pues no les parece una oferta justa al precio original. Hoy en día los jugadores pueden devolver el juego si no les han gustado las primeras horas de gameplay o el juego no les parece merecedor de lo que han pagado por él, así que críticas negativas y un número consistente de devoluciones pueden hacer que el juego fracase desde el momento en el cual se le pone un precio.

Hay que tener en cuenta, según recomendaciones de otros desarrolladores y también aplicando el sentido común, que un precio más alto va a suponer en que el juego pueda tener un tiempo de vida más grande. Si el precio base comienza siendo muy bajo, el juego no va a poder hacer rebajas, puesto que teniendo en cuenta las comisiones de las plataformas, el beneficio generado por venta podría llegar a ser nulo. Hacer que el juego pueda ser rebajado al 50 % o al 75 % y aun así generar beneficios hará que el juego pueda estar a la venta varios años. Comúnmente los desarrolladores ofrecen descuentos que van incrementándose año a año tras su lanzamiento. Además, podemos subir o bajar el precio después de que el juego haya sido lanzado, pero una subida de precio es muy rara a no ser que el juego pase de acceso anticipado a una versión completa, además esto debe ser anunciado con mucha anterioridad. Un decremento del precio original probablemente pase desapercibido a no ser que el juego sea rebajado por tiempo limitado.

El objetivo de pensar el precio del juego antes de empezar a crear un prototipo es adaptar las características que se tendrán que implementar para que pueda entrar en la categoría del precio elegido, pero en este punto el precio será solo orientativo.

Tras una pequeña investigación a la plataforma de Steam, podemos ver que los mejores juegos indie con estética pixel se venden entre 7 euros y 25 euros, Stardew Valley cuesta 14 euros, Terraria cuesta 10 euros, Dead Cells cuesta 25 euros, Hollow Knight cuesta 15 euros, etc. Con lo cual, calculo que el juego deberá costar entre 7 euros y 15 euros en función del contenido final que se pretende añadir. Se podría lanzar en early access por unos 9 euros y acabar costando 14 euros en su versión final.

3.4.4 Tecnología a utilizar

Un videojuego utiliza muchas herramientas para su desarrollo, usualmente lo mínimo para este tipo de juegos sería un motor gráfico apropiado, un entorno de programación y una herramienta que nos ayude a crear o modificar los sprites, además, se podría considerar también el uso de herramientas que nos faciliten el flujo de trabajo o nos ayuden a documentar el proceso de desarrollo.

- Selección de motor gráfico:

  Sabemos que los desarrolladores suelen elegir Unity o Unreal Engine para sus proyectos, los estudios AAA también suelen utilizar estos dos motores por encima del resto disponible. Además, en el caso de aprender a usar cualquiera de estos dos motores gráficos, también sería posible optar a un trabajo como desarrollador de videojuegos en un estudio que ya esté creado.

  Para realizar el juego deseado, sin embargo, se ha tenido en cuenta además de estos motores ya mencionados, el motor RPG Maker MV, este, a pesar de no ser tan conocido, merece la pena mencionarlo, ya que está especialmente diseñado para RPG 2D hechos con pixel art y además es conocido por la velocidad con la que permite desarrollar prototipos.

  - RPG Maker MV

    - RPG Maker se define a si mismo como un motor gráfico potente y capaz de servir a desarrolladores novatos o experimentados por igual (Games, 2015), por poco menos de 30 euros podemos conseguir una clave permanente de este motor que nos permitiría utilizarlo y vender juegos producidos con él sin ningún coste adicional.

Figura 3.23: RPG Maker MV

RPG Maker MV es la octava versión de RPG Maker la cual debuto el 23 de octubre de 2015, pero sigue siendo la más popular hasta la fecha, la siguiente versión RPG Maker MZ lanzada en 2020 tuvo críticas bastante grandes destacando su similitud con la versión de 2015, empeorando sistemas que funcionaban bien en 2015 y añadiendo muy pocas mejorar sobre la antigua. Esto hace que los desarrolladores que han elegido este motor gráfico aún se decanten por la versión de 2015.

Figura 3.24: Learn Japanese Katakana War - RPG Maker MV

Figura 3.25: Omori - RPG Maker MV

Habiendo usado este motor por un par de semanas puedo destacar lo siguiente:

Tabla 3.1: Ventajas y desventajas - RPG Maker MV

Ventajas y desventajas de RPG Maker MX	
Ventajas	Desventajas
Diseñado para 2D.	Difícilmente escalable.
Diseñado para RPG.	Muchos cálculos innecesarios en segundo plano.
Muchísima funcionalidad ya implementada.	Penaliza mucho los errores y cualquier cosa puede hacer que el juego empiece a ir lento.
Muchas capas por encima de la programación que aceleran mucho el desarrollo.	Comúnmente asociado con malos juegos.
Facilidad para mapear debido a que implementa las capas automáticamente.	No hay forma de proteger tus archivos.
Comunidad amigable y facilidad de aprendizaje.	Complica mucho el usarlo para algo para lo que no ha sido pensado.
Exportación multiplataforma	Parece más un motor desarrollado para realizar prototipos o pruebas.

Se puede decir que este motor es muy útil si quieres fabricar un prototipo muy rápidamente o en su lugar un juego que siga estrictamente los cánones

de un RPG clásico. No utilizaría este motor si debo modificar cosas para lo que no ha estado pensado, como por ejemplo: la cámara o el sistema de batalla por turnos. A cambio, debo destacar que solo en un par de horas puedes realizar un prototipo, cosa que en el resto de motores nombrados en este documento sería totalmente imposible.
Las modificaciones al código base se pueden efectuar empleando javascript y extendiendo las clases que ya existen. Pero, finalmente, he elegido no usar este motor para este proyecto debido a que muchas de las cosas que planeo hacer requieren escalar mucho la funcionalidad básica del mismo. El motor sería muy utilizable para llevar a cabo un prototipo del videojuego, pero no para la versión final.
Puedo concluir que lo bueno que tiene RPG Maker es que puede usarse sin tener nociones avanzadas sobre programación o arte, siempre que te adaptes a lo que te ofrece el motor base. Todo está hecho para que puedas tomar decisiones sobre el diseño de niveles, enemigos y balance sin necesidad de programar y ofreciendo scripts de calidad muy depurados y sin errores. Como contra-parte, adaptarte a todo lo que ofrecen llevaría a que el juego no fuera nada innovador y no tuviese un factor diferenciador atractivo. Además, para poder efectuar cambios y hacer de nuestro juego algo viable, necesitaríamos modificar muchas de las herramientas que ofrece, quitándonos su mayor ventaja de poder hacer un prototipo rápido. Dicho esto, Unity o Unreal Engine nos brindan muchas más posibilidades a medio y largo plazo. A su vez, conocer en profundidad estos motores puede brindar muchas más oportunidades. Finalmente, puedo decir que RPG Maker es digno de mención y merece ser considerado, puesto que los RPG tienden a tener un tiempo de desarrollo más largo que la mayoría de juegos y este nos permitiría llevar a cabo un prototipo que integre muchos de los sistemas que podrían acabar en una versión final en solo un par de semanas.
Está previsto que para finales de 2022 RPG Maker va a lanzar RPG Maker Unite.

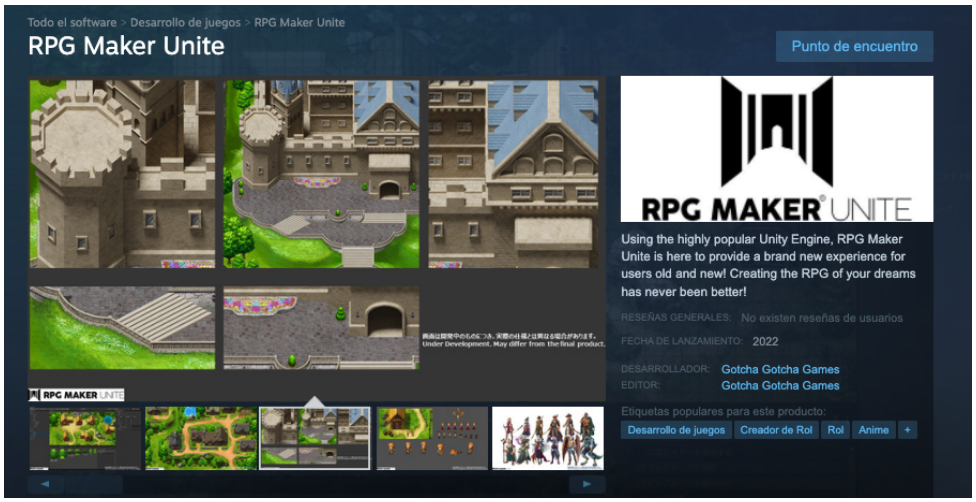

Figura 3.26: RPG Maker Unite

Esta será la próxima versión de RPG Maker, la cual se integrará con Unity como un paquete de terceros, permitiendo así integrar todas las herramientas de RPG Maker en Unity. En el caso de haber sido lanzado ya, es muy probable que se reconsiderará mi decisión sobre no utilizarlo.

- Unreal Engine
  - Unreal Engine ha sido sin duda la elección de la mayoría de juegos AAA en los últimos años. Posee muchos assets gratuitos de calidad descargables desde un mercado oficial que incorpora muchísimas herramientas de terceros. Además, puedes descargar el motor de forma gratuita, emplearlo indefinidamente y solo pagar un 5 % de lo recaudado si tu juego alcanza un millón de dólares en ganancias, con lo que permite mantener unos costes de desarrollo bajos.

Figura 3.27: Unreal Engine

Este motor gráfico está altamente orientado a la creación de juegos 3D, ya que los creadores del motor no están priorizando demasiado las herramientas enfocadas al 2D. Aun así, en los últimos años han salido algunos juegos que utilizan 2D y 2.5D utilizando Unreal Engine como Octopath Traveler y Triangle Strategy.

Figura 3.28: Octopath Traveler - Unreal Engine

Unreal Engine permite el uso de C++ y de una herramienta propia llamada BluePrints. Esta herramienta permite al usuario programar visualmente por medio de grafos y variables.

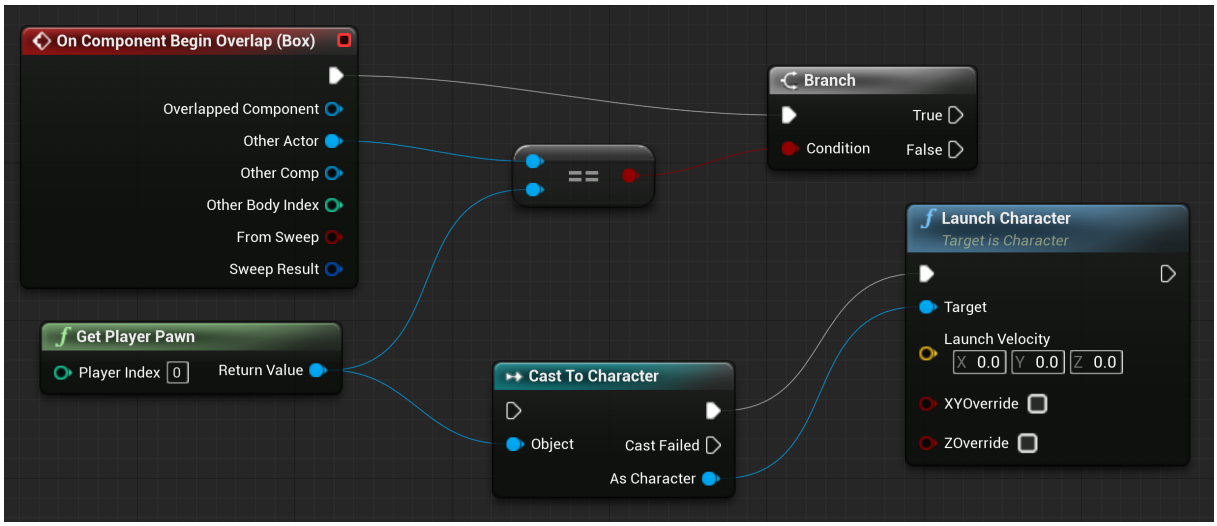

Figura 3.29: Blueprints - Unreal Engine

Los blueprints están enfocados sobre todo para dar vida a objetos y se pueden combinar con C++ para realizar acciones más complejas, además, en los últimos años han salido muchos juegos al mercado que únicamente utilizan blueprints.
Nunca había usado Unreal Engine y aunque este momento sería una buena oportunidad para aprender más a fondo. He preferido no utilizar este motor gráfico para mi proyecto después de haberlo probado solo una semana, los motivos son los siguientes:

- Muy poca documentación enfocada a juegos en 2D.
- Enfocado a proyectos muy grandes y a trabajar en equipo.
- Empezar en Unreal Engine es un proceso muy laborioso, por lo menos mucho más que en Unity o RPG Maker.

- ◇ Dificultad para optimizar el renderizado en 2D debido a que está enfocado para juegos en 3D.
- ◇ Al utilizarlo te das cuenta de que todas las clases están enfocadas al 3D y tienes que trabajar mucho más para que algo en 2D funcione, a pesar de ser objetivamente más simple.

- Unity

  - Unity es un motor gráfico ampliamente usado por los desarrolladores indie en la actualidad. Está enfocado a 3D, pero no descuida las herramientas 2D y recientemente le está dando más prioridad a este apartado. Se puede descargar Unity de forma gratuita y ofrece diversos planes en función de lo que hayas recaudado el último año, ya bien sea en financiación o ventas. Unity es gratuito hasta que tu juego ha generado 100.000 dólares en un año, en este momento tendrías que abonar 399 dólares por cada por persona que lo use al año. Si tu juego genera más de 200.000 dólares al año, tendrías que abonar 1800 dólares por cada persona.

Figura 3.30: Unity

Unity permite utilizar C# y programación visual con una herramienta a la cual llaman bolt, pero la cual es infinitamente menos usada que los blueprints de Unreal Engine. También dispone de un mercado oficial que permite añadir assets gratuitos o de pago al proyecto y entre otras ventajas, ofrece múltiples tutoriales gratuitos sobre como hacer juegos 2D con más de 750 horas de vídeo oficial enfocado al aprendizaje.
Estas ventajas son las que me han hecho elegir Unity para hacer mi proyecto, además, ya había usado Unity en algunas asignaturas optativas del grado y me siento más cómodo utilizándolo a medio y largo plazo.

Todos los motores gráficos aquí expuestos podrían ser una opción viable para realizar un juego de estas características, pero la decisión tomada deberá tomar en cuenta ciertas variables como la experiencia actual del desarrollador, la facilidad de aprendizaje, la velocidad de desarrollo, Etc. Elegir la correcta herramienta de desarrollo no es trivial, ya que para cada proyecto la herramienta óptima puede varias. Además, una buena elección pue-

de ahorrar horas de desarrollo. Si hubiese elegido otra opción con la que estuviera menos cómodo, tendría que justificar mi decisión en función de las variables arriba expuestas.

Estar cómodo con Unity habiéndolo usado anteriormente hace que me sienta seguro sobre el uso de buenas prácticas y velocidad en el desarrollo. Pero, ciertamente, se ha dado casos de que equipos de desarrollo prefieran desarrollar un addon para un motor en vez de utilizar otro que ya implemente la característica que buscan, esto es debido a que ser experto con un motor en concreto es una tarea que requiere varios años de experiencia.

- Entornos de desarrollo

  El entorno de desarrollo es la herramienta la cual se utiliza para escribir el código que después pasara por el motor gráfico, estos usualmente incorporan funciones que ayudan al auto-completado, encontrar y eliminar errores, mantener la persistencia en repositorios, ejecutar casos de uso, Etc. En el caso de Unity este nos permite instalar por defecto Visual Studio, pero Unity también permite el uso de otros entornos de desarrollo que también son soportados y que pueden ser cambiados en función de las preferencias del desarrollador.

  En este caso se han considerado dos opciones, Visual Studio y Jetbrains Rider.

  - Visual Studio

    - Es posible instalar este editor en el mismo momento que instalamos cualquier versión de Unity, se integra automáticamente de forma instantánea y ofrece muchas herramientas que controlan el flujo de Unity. Además, puede ser usado con la misma licencia de Unity por lo que no incrementaría el costo de desarrollo total.

Figura 3.31: Visual Studio

  - Jetbrains Rider

    - Este editor está desarrollado por Jetbrains, originalmente fue orientado para la programación en .NET, pero tiene un plugin oficial que lo integra con Unity. Jetbrains es la misma empresa que ha desarrollado Android Studio y otros entornos de desarrollo muy famosos y que soportan multitud de lenguajes. El coste de utilizarlo son 140 euros el primer año, 120 euros el segundo

y 90 euros desde el tercero en adelante, pero ofrece planes que permiten utilizarlo de forma indefinida mientras sigas siendo un estudiante.

Figura 3.32: Jetbrains Ryder

- Comparativa
  - A grandes rasgos Rider ofrece todas las características que ofrece Visual Studio y además trae mejoras en la vista de ficheros, la integración de la consola, el trabajo con shaders, búsqueda de funciones, asistencia y análisis de código, eliminación de errores y además incluye la posibilidad de realizar casos de uso. Hay ciertos plugins en Visual Studio que intentan implementar muchas de las funciones que incorpora Rider, pero en la práctica Rider gana por mucho a Visual Studio.

En este caso se va a utilizar la versión de estudiante de Jetbrains Rider porque este trabajo va a cubrir hasta la creación del primer prototipo, que tendrá por supuesto fines educativos, pero en el presupuesto se supondrá que se va a utilizar la versión de pago. Me siento mucho más cómodo usando Jetbrains Rider por su parecido con Jetbrains PhpStorm, una herramienta la cual he utilizado ampliamente por varios años en mi trabajo como desarrollador web. El coste de la herramienta puede ser fácilmente justificado por la velocidad en el desarrollo causada por la experiencia con una herramienta que incorpora los mismos atajos de teclado y funcionalidad.

- Editor de sprites

  Para cumplir con nuestras necesidades en algún punto vamos a necesitar un editor de sprites. Este software está dedicado únicamente a trabajar con imágenes de este tipo y las herramientas que ofrecen están totalmente enfocadas en ello. Técnicamente, cualquier imagen de tipo pixel es simplemente una imagen muy pequeña, con lo que se podría utilizar Photoshop (El software más usado dedicado a la edición) como herramienta, pero el coste de las herramientas de edición como esta es muy alto y no ofrecen ventajas sobre las aplicaciones especializadas con un coste más bajo. Las dos aplicaciones más empleadas son Pyxel Edit y Aseprite.

En este caso no será necesario realizar una comparativa, pues la comunidad recomienda usar ambas herramientas y además su bajo coste lo permite. Pyxel Edit es mucho mejor para el trabajo con fondos y Aseprite excede en el dibujo de personajes.

El coste si utilizamos las dos herramientas no supera los 24 Euros por licencias permanentes. Pyxel Edit cuesta 9 Euros y Aseprite 15 Euros, ambas aplicaciones se verán reflejados en el presupuesto.

3.4.5 Calendario

Una vez hemos llegado a este punto, es prioritario afrontar la tarea de poner tiempo a las fases de desarrollo para así posteriormente poder realizar un presupuesto estimado en función de las horas empleadas. Cabe decir que el tiempo en este punto puede variar en función de las decisiones posteriores tomadas en fases de desarrollo y que este calendario puede necesitar modificaciones en el futuro, pero sigue siendo prioritario hacerlo para poder realizar un presupuesto estimado, además, como se ha mencionado en apartados previos, colocar fechas puede incrementar la productividad a la par que evitar el perfeccionismo en fases tempranas.

Para este proyecto, se estimará la fecha del primer prototipo, que ni de lejos estará cerca de ser una versión final, pero se podrán ver varias de las mecánicas que se utilizarán en la versión de producción. También cabe decir que en las horas marcadas está reflejado un tiempo dedicado al aprendizaje de la misma si fuese necesario y además a contar con assets de refuerzo tanto en el apartado artístico como el de software.

Tabla 3.2: Contador de horas por tarea

Contador de horas por tarea	
Tarea	Número de horas
Investigación previa al desarrollo (Patrones de diseño, Assets a emplear, Etc.)	20
Adaptación de sprites del personaje, movimiento y cámara	10
NPC	20
Gestión de personajes y equipos	10
Gestión de Quests	10
Batalla	25
Mapeado zonas prototipo	15
Iluminación	10
Pulimento final del prototipo	10

Incluso aunque el calendario sea aproximado, existen muchos beneficios en realizarlo antes de empezar el desarrollo. Si sentimos que el calendario es excesivo, estaremos en un punto en el cual poder retomar la toma de decisiones sobre el alcance del juego y reconsiderar si es viable hacerlo de forma diferente omitiendo o simplificando mecánicas.

En este caso se han contabilizado 130 horas, las cuales se repartirán en 3 semanas, haciendo 43 horas aproximadamente cada semana, además, se contará con una semana extra para acabar de pulir el prototipo, esto es debido a que muchas de las tareas posiblemente se queden cortas por el tiempo de aprendizaje de las mismas.

3.4.6 Presupuesto

Por último, en esta sección se contabilizará de forma aproximada el presupuesto necesario para realizar el prototipo planteado.

Tabla 3.3: Presupuesto

Presupuesto estimado	
Operación	Dinero destinado
Tareas	1950 Euros
Assets	100 Euros
Software	24 Euros
Hardware	1500 Euros
TOTAL	3574 Euros

Para las tareas se ha estudiado el salario medio para un desarrollador de videojuegos en España y se ha concluido un sueldo de 15 Euros / Hora, lo cual hace un total de 1950 euros. En el caso de los assets, se aporta un presupuesto de 100 euros que se utilizarán para comprar los gráficos y scripts necesarios. El software aborda Pixel Edit y Aseprite, puesto que Unity es gratuito mientras no recaudemos 100.000 euros con el proyecto y Rider ofrece versiones para estudiantes. El Hardware hace referencia al equipo empleado para desarrollar el proyecto, el cual es un Macbook Pro de 2020 valorado en esa cantidad.

4 Diseño de la solución

Este punto cubrirá el diseño detallado del videojuego, explicando más a fondo las mecánicas que incorporara el prototipo a desarrollar. Además, se hará especial hincapié en patrones de diseño, puesto que a pesar de que el prototipo a desarrollar no sea especialmente grande, se pretende escalarlo en el futuro. Errores en el diseño pueden hacer perder mucho tiempo en fases intermedias del desarrollo, obligando estos a rehacer sistemas del juego por completo.

En el punto destinado a preproducción se ha dado una idea vaga de lo que será el videojuego en su versión final, describiendo su historia y un ejemplo del bucle de juego. Esto era necesario para poder realizar los análisis necesarios que aseguraban que el desarrollo del juego sería viable, pero en los siguientes puntos se detallara más a fondo el contenido del prototipo.

4.1 Apartado gráfico

Se pretende desarrollar un juego 2D con gráficos píxel, estos acostumbran a ir en escalas de 16x16 bits, 32x32 bits o 64x64 bits en función de la calidad deseada. Estos bits hacen referencia a la cantidad de píxeles que existen en un cuadrado, puesto que el mapeado se desarrolla íntegramente en cuadrículas compuestas de estos bits. Usualmente, se acostumbra a que el jugador y el mapeado tengan el mismo número de bits para que el nivel de detalle sea similar, pero se da muchos casos en el personaje tiene un número de bits superior al mapeado para así resaltarlo.

Los gráficos utilizados han sido creados por ´finalbossblues´ un artista dedicado al píxel art y comprados desde itch.io utilizando parte del presupuesto destinado para assets. Estos incluyen mapeado, personajes para usar en batallas laterales, algunas animaciones, Etc.

Figura 4.1: Time Fantasy - finalbossblues

Los gráficos tienen un acabado profesional y están listos para ser empleados, esto ahorra una gran parte de tiempo y hace que sean un gran punto de partida, además, se adecuan a la temática de forma apropiada.

Existe la posibilidad de utilizar estos gráficos en formato 32x32 o 64x64, siendo este último el que se usará. Aunque será posible el re-escalado desde el propio motor gráfico siempre que fuera necesario para adaptar los píxeles a determinadas resoluciones. El autor permite además la modificación de los propios sprites aportados, haciendo que integrar nuevos sprites en el futuro utilizando el software expuesto sea mucho más viable.

4.2 Scripts de partida

Debido a la amplitud del proyecto y la cantidad de sistemas a personalizar en el futuro, se partirá de la base del '2D RPG KIT' de la tienda oficial de Unity, comprado con el resto del dinero asociado a assets del presupuesto.

Figura 4.2: 2D RPG Kit - Arktentrion

Este incluye una gran variedad de herramientas que podremos utilizar como base en el desarrollo del juego. Una de las cosas aprendidas en el grado es que no tiene sentido crear algo que ya está creado, es infinitamente mejor empezar con una base y mejorarla para hacer que se adapte a nuestras necesidades.

Esto permitirá que el desarrollo del prototipo se adecue a las fechas marcadas (Las cuales ya tenían en cuenta este tipo de base) y además garantizará las buenas prácticas en el proyecto mediante el patrón de diseño ´Mánager´.

4.2.1 Patrón de diseño ´Mánager´

Este Kit utiliza un conocido patron de diseño llamado ´Mánager´ el cual está estructurado de la siguiente manera:

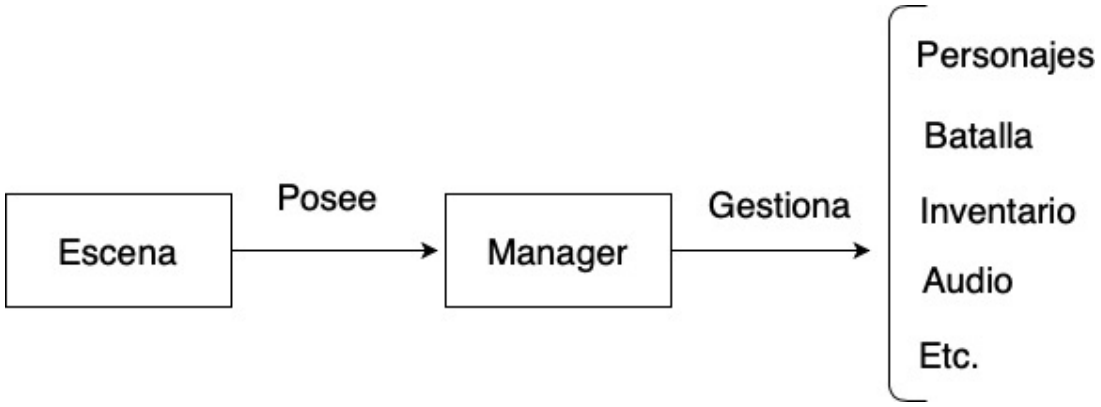

Figura 4.3: Patron de diseño - Mánager

Dentro de la escena de Unity tenemos un gameobject mánager destinado a controlar el flujo del juego, dentro de este mánager padre tenemos diversos mánager hijos los cuales en este caso podrían ser el mánager del equipo, batalla, inventario actual, Etc. Usualmente en Unity cada escena tiene sus propios gameobjects los cuales no se mantienen al cambiar a otra escena, pero existe la posibilidad de indicar a un gameobject que no se destruya entre escenas. Con esto conseguimos controlar todo el flujo de la aplicación.

Este patrón es ampliamente utilizado en la industria actual, con lo que se adecua a lo aceptado por Unity, esto hace que sea compatible con herramientas como la persistencia de datos, es decir, hacer que el usuario sea capaz de guardar partida. Esto Unity lo realiza permitiendo la persistencia de clases individuales, con lo que solo tendríamos que guardar el mánager padre para tener todos los datos necesarios en el mismo lugar.

4.3 Modo de juego

El objetivo del prototipo será que el jugador complete el entrenamiento destinado a los nuevos clones de la corporación, el cual será el inicio de la historia que situara al jugador en el mundo. En este modo de juego el jugador será capaz de experimentar distintas etapas que incluirán todas las mecánicas disponibles y le situaran en un departamento u otro al final del entrenamiento.

Además, se incorporará un selector de dificultad que hará que los enemigos varíen su fuerza en función de lo escogido, esto hace que el juego pueda ser rejugado en otra dificultad y además hace que los jugadores más exigentes no se aburran del contenido.

4.4 Cámara

La cámara del juego se situará de forma típica en los juegos 2D, estando en la cabeza del jugador y siguiéndolo permitiéndole ver el mapa que lo rodea.

Figura 4.4: Pokemon esmeralda - Ejemplo de cámara

4.5 Pantalla principal

El juego dispondrá de un menú principal en el cual poder iniciar el juego eligiendo la dificultad, cargar partida y salir al menú principal.

4.6 Pantalla de fin del juego

El juego dispondrá de una pantalla que se mostrará en el caso de ser eliminados en una batalla, esta llevara de vuelta al menú principal.

4.7 Personaje y grupo

El juego dispondrá de un personaje principal y además podrá incorporar personajes a su grupo. Los personajes podrán dispones de armas y armaduras y poseerán estadísticas de vitalidad, fuerza, velocidad, habilidad, Etc. Además, estos aprenderán habilidades nuevas cuando suban de nivel.

Todo esto podrá ser seguido desde el menú de personaje, el cual mostrara información actual de los personajes en el grupo, estadísticas, equipo, Etc.

4.8 Sistema de batalla

El juego dispondrá de un sistema de batalla que será establecido por turnos, haciendo que jugadores y enemigos dispongan de sus turnos en función del parámetro de velocidad. En cada turno podrán elegir si atacar, utilizar una habilidad, un objeto o bien escapar de la batalla. Este sistema estará manejado por las teclas o el ratón y finalizará cuando uno de los equipos este muerto o se escape de la batalla. En el caso de que el equipo del jugador muera, se acabará el juego y el jugador tendrá que cargar partida o empezar de nuevo, si los enemigos mueren el jugador recibirá experiencia y objetos.

Este sistema ha probado ser muy válido en juegos de todas las épocas y además hará que las batallas tengan parte de estrategia.

4.9 Sistema de encuentros con enemigos

El juego dispondrá de zonas en el mapa para que los grupos de enemigos aparezcan de forma aleatoria si el personaje camina por ellas.

4.10 NPC

Los NPC son una parte muy importante de los juegos RPG, ya que usualmente estos son los que cuentan la historia por medio de diálogo. Usualmente, pueden hablar, darnos y completar tareas, vender o comprar objetos y dejarnos descansar para recuperarnos. En este juego se hará lo mismo con ellos, extendiendo la funcionalidad en el caso de ser necesario en el futuro.

4.11 Sistema de tareas

Los NPC del juego podrán darnos diversas tareas que se anunciaran en la parte superior de la pantalla cuando se reciban y a la vez cuando se completen. Este tipo de sistemas ayudan a que el usuario no se pierda y tenga siempre una tarea pendiente.

Se podrá completar tareas al hablar con otro NPC o al completar una batalla específica.

5 Desarrollo de la solución propuesta

En este punto se explicarán todos los pasos a seguir para disponer de todo lo diseñado en el punto anterior, empezando por la descarga del motor y acabando en la persistencia de partida.

5.1 Descarga y configuración de Unity y Rider

5.1.1 Descarga de Unity

El primer paso será ir a la web oficial de Unity y descargar la versión más reciente de Unity Hub e instalarlo. Este es un software creado por Unity para gestionar las versiones de Unity instaladas y facilitar la creación de proyectos. Una vez descargado, podemos ver una interfaz como esta:

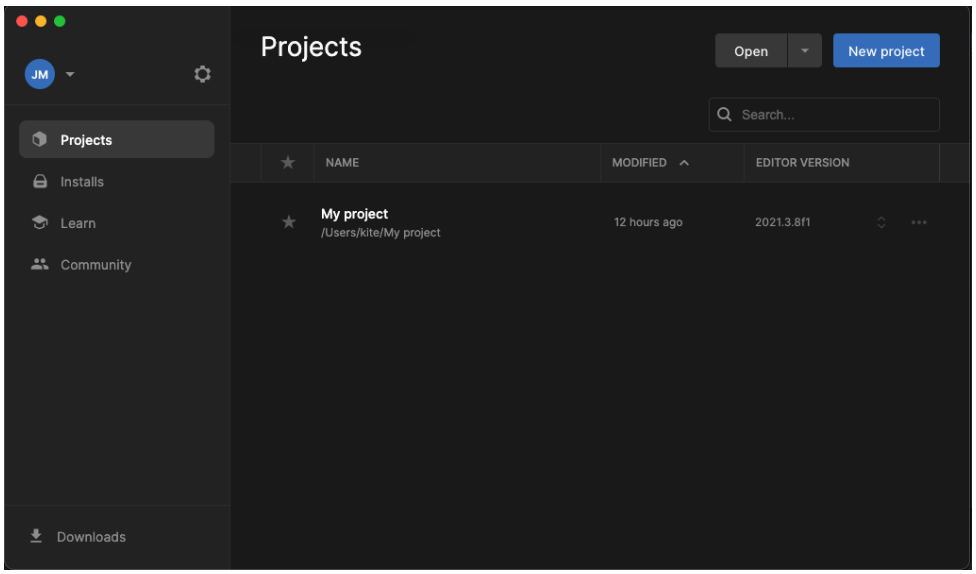

Figura 5.1: Unity Hub - Interfaz

Una vez aquí podemos encontrar cuatro pestañas a mano izquierda, las cuales sirven para controlar el flujo de la aplicación. Primero debemos ir a la pestaña de instalaciones y seleccionar el botón en la parte superior que nos permite instalar nuevos editores. Una vez aquí se nos preguntará si queremos instalar una versión experimental o una versión LTS de larga duración.

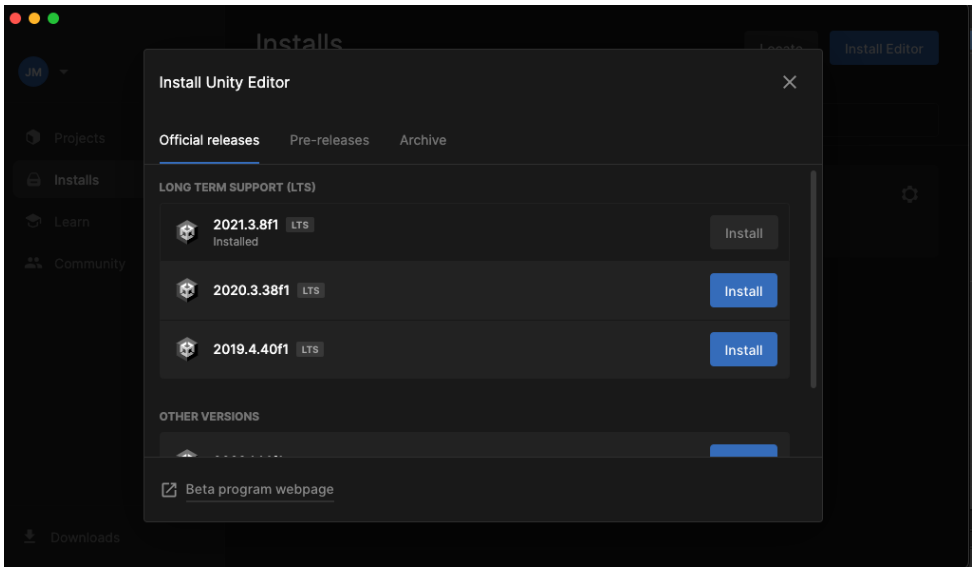

Figura 5.2: Unity Hub - Instalación del editor

Las versiones LTS son mantenidas y actualizadas por Unity de forma segura con paquetes ya probados y aseguran una mayor estabilidad que las versiones experimentales, aunque estas tienen más funcionalidad que las versiones LTS. Unity nos asegura soporte en las versiones LTS por varios años desde que son creadas, por lo que la mejor decisión es descargar la última versión LTS disponible en este momento, la 2021.3.8f1. Las versiones experimentales están hechas para que los desarrolladores prueben los últimos paquetes actualizados, pero no se recomienda utilizarlas a largo plazo, además, es posible descargar un paquete experimental en una versión estable de Unity si esto fuera necesario.

Cuando hayamos seccionado la versión estable, nos preguntara para qué plataformas, vamos a querer compilar el juego una vez este desarrollado y además si queremos descargar la versión de Visual Studio preparada para integrarse en Unity.

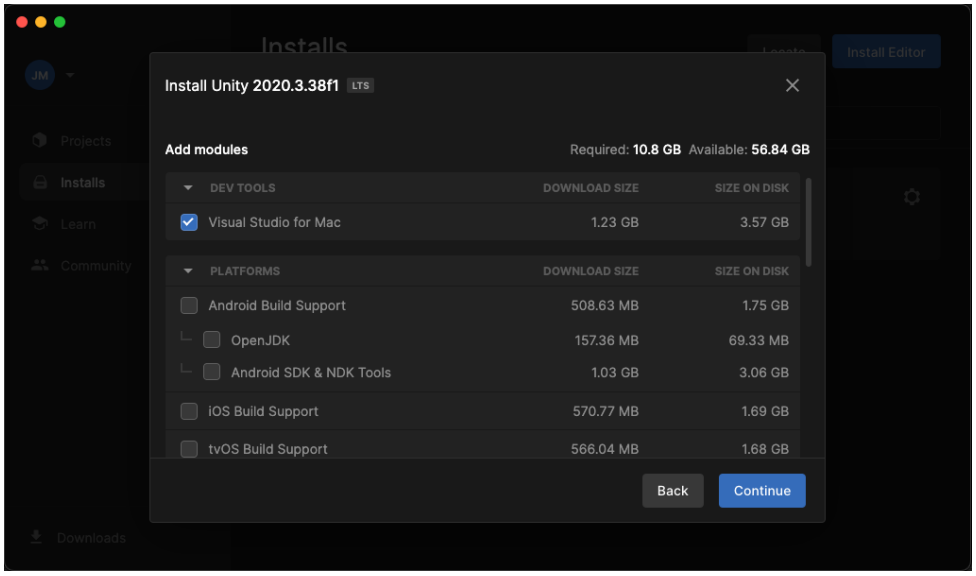

Figura 5.3: Unity Hub - Descarga del editor

Yo solo seleccionaré la compilación para mi plataforma nativa que en este caso es Mac OS, pero no seleccionaré el Visual Studio, puesto que voy a utilizar Rider como editor.

Una vez descargado, ya podemos ir a la pestaña de proyectos y crear un nuevo proyecto en el botón superior, aquí Unity nos preguntará que tipo de proyecto vamos a crear. Incluso si el editor usado es el mismo para 2D que para 3D, los paquetes instalados con el editor nos serán los mismos, si seleccionamos 3D Unity nos descargará un montón de paquetes para animaciones, terrenos, texturas, Etc. Y, si, por otra parte, seleccionamos 2D, Unity nos descargará muchas herramientas para trabajar con Sprites, Tilesets, Etc. Además, podemos seleccionar la opción de 2D (URP), esto descargaría todos los paquetes 2D y además muchos paquetes para trabajar con la Universal Render Pipeline, un método de renderizado diferente al habitual instalado en proyectos 2D que permite emplear shaders, iluminación 2D, Etc. En nuestro caso vamos a instalar Unity 2D sin URP, puesto que prefiero agregar URP a través de los paquetes de Unity cuando sea necesario. Cuando hayamos hecho esto, el proyecto empezara a abrirse y ya quedara Unity preparado para trabajar.

5.1.2 Instalación de Rider

Para instalar Rider deberemos ir a la página oficial de Jetbrains y en la parte superior buscar Jetbrains Rider, esto abrirá una página como esta:

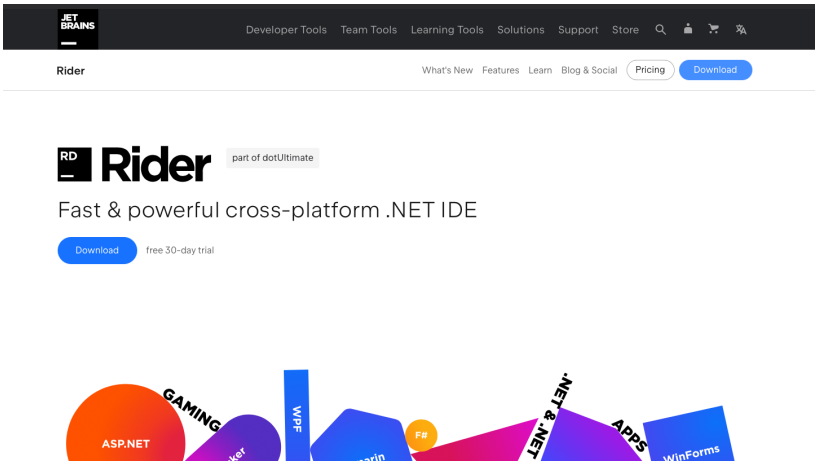

Figura 5.4: Jetbrains Rider - Web

Una vez descargado e instalado nos pedirá una cuenta de activación, en este caso podemos darle a probar 30 días o como en mi caso si nos hemos registrado para la versión de estudiantes tendremos un año de forma gratuita para proyectos destinados a la educación.

5.1.3 Integración de Rider con Unity

Una vez dentro del editor de Unity, podemos ir a Unity ->Preferencias en las pestañas superiores y una vez aquí a la pestaña de herramientas externas. Aquí podemos seleccionar que Unity se integre con Rider.

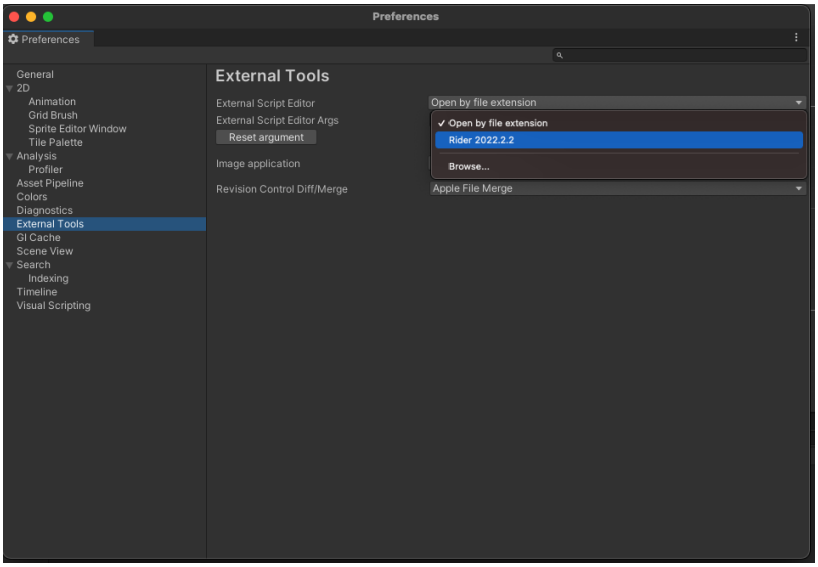

Figura 5.5: Unity - Integración con Rider

5.2 Descarga e importación de paquetes

Lo siguiente a realizar es la descarga e importación de los assets que se van a utilizar en el proyecto.

5.2.1 2D RPG Kit

Como este asset ha sido adquirido desde la tienda oficial de Unity, se debe utilizar el paquete de descarga de la misma. Desde el editor vamos a Window ->Package Manager y una vez dentro de esta ventana a la parte de nuestros assets. Aquí aparecerá todo lo que ha adquirido en la tienda oficial de Unity con la cuenta asociada al proyecto.

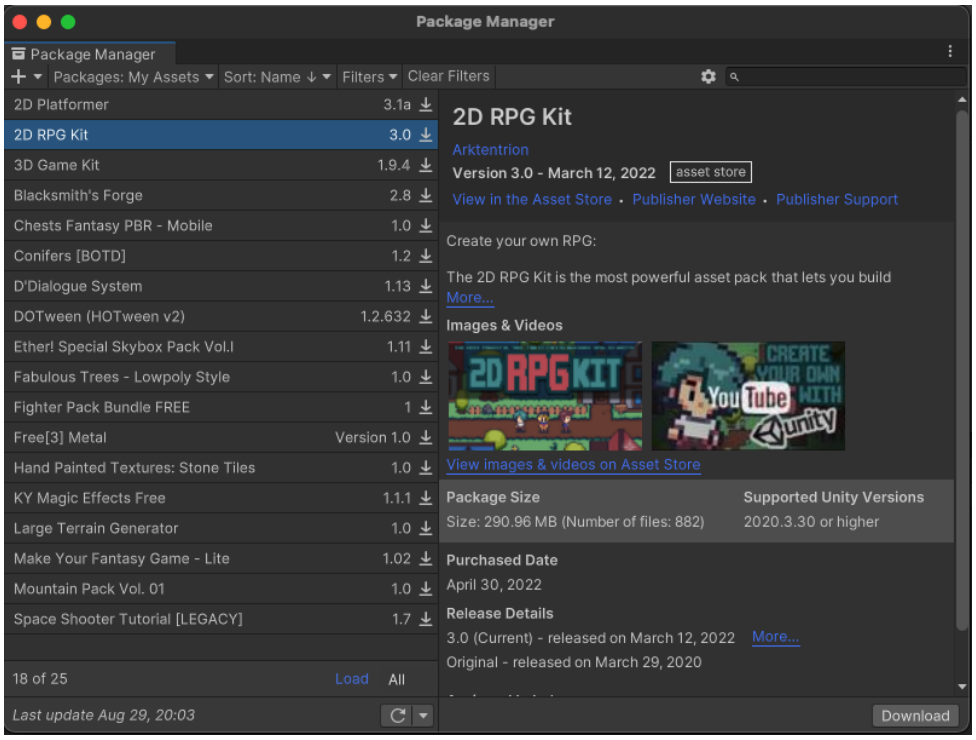

Figura 5.6: Unity - Descarga de paquetes de la tienda

Una vez seleccionado el paquete pertinente, en este caso el 2D RPG Kit, descarga desde el botón de la derecha. Cuando la descarga haya finalizado, se podrá importar al proyecto.

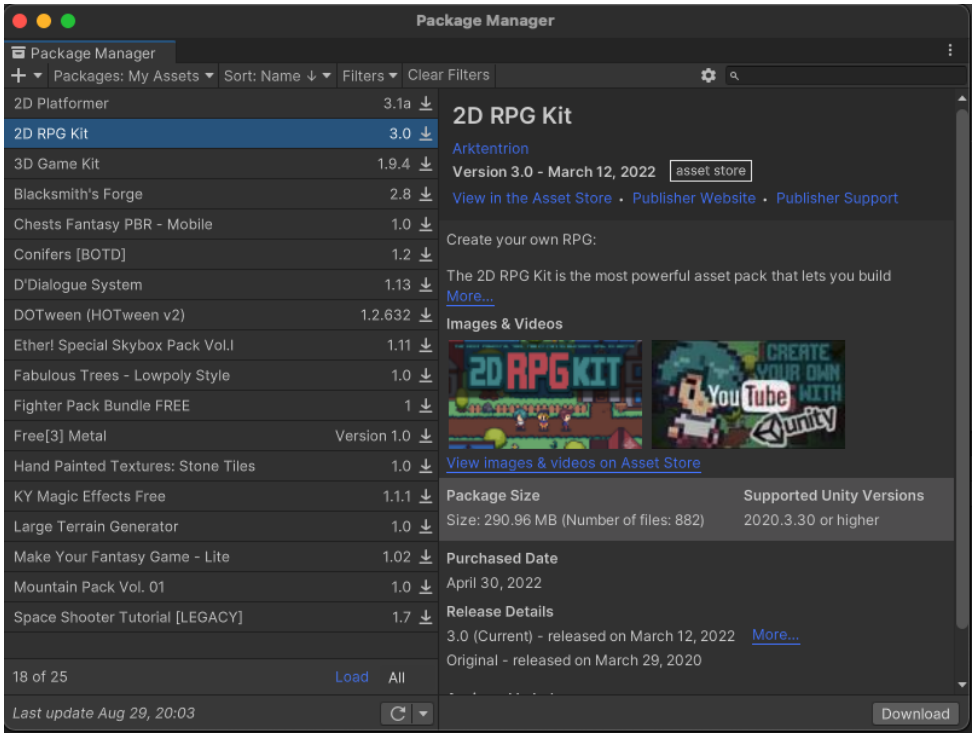

Figura 5.7: Unity - Importación

Una vez importado aparecerán todos los archivos en la parte inferior destinada a ellos.

Figura 5.8: Unity - Archivos

Podemos ver que los archivos siguen una estructura óptima para mantener el desarrollo de forma ordenada y poder extenderlo en función de nuestras necesidades.

5.2.2 Assets gráficos

Los assets gráficos no han sido adquiridos desde la tienda de Unity por lo que su importación es diferente. En este caso debemos descargar los assets desde la página correspondiente, en este caso itch.io. Y, una vez descargados y con los archivos en nuestro ordenador, tendremos que arrastrar los archivos que queramos importar a la parte de archivos de Unity como si de otra carpeta de nuestro ordenador se tratase. De esta forma Unity importará los archivos y ya podrán usarse dentro del editor.

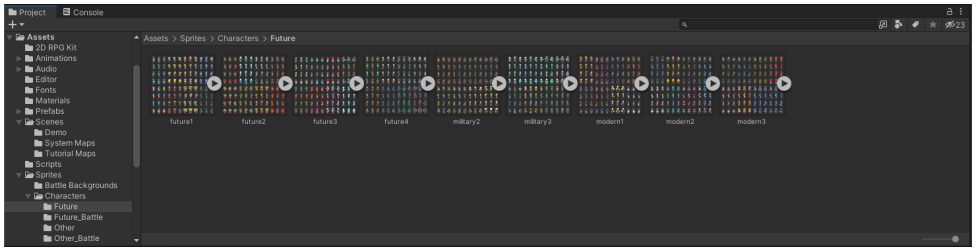

Figura 5.9: Unity - Archivos importados

Una vez importados, es recomendable moverlos para seguir la estructura ya marcada por el Kit descargado anteriormente, de esta forma aseguraremos las buenas prácticas y no se perderán archivos en el futuro.

5.3 Modificar el flujo de trabajo

Cuando Unity se abrió por primera vez, trajo este flujo de trabajo por defecto:

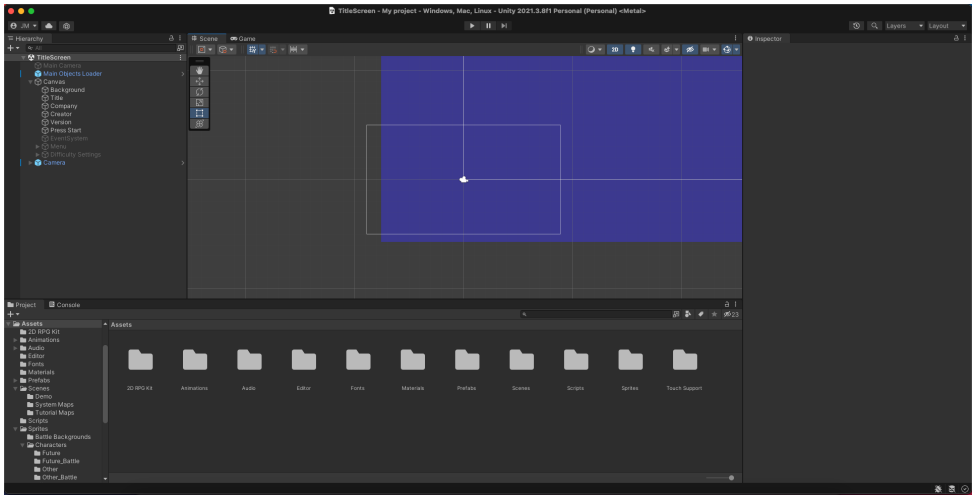

Figura 5.10: Unity - Flujo por defecto

En el cual podemos ver la jerarquía de objetos en la escena a la izquierda, la escena en medio junto con una pestaña para alternar con la pantalla de juego que mostrara lo que verá el jugador en cada momento y es mostrado por la cámara configurada, el inspector de Unity a la derecha y los archivos del proyecto junto con la consola en la parte inferior. Este flujo es totalmente ineficiente, pues no nos permite poder ver la escena y la cámara de juego al mismo tiempo, a su vez, cuando trabajemos con animaciones necesitaremos tener los archivos del proyecto en varias partes al mismo tiempo.

Para poder cambiar el flujo de trabajo podemos ir a la pestaña Window e ir sacando las ventanas que nos interesen para acomodarlas al flujo, en mi caso lo he dejado todo de la siguiente manera:

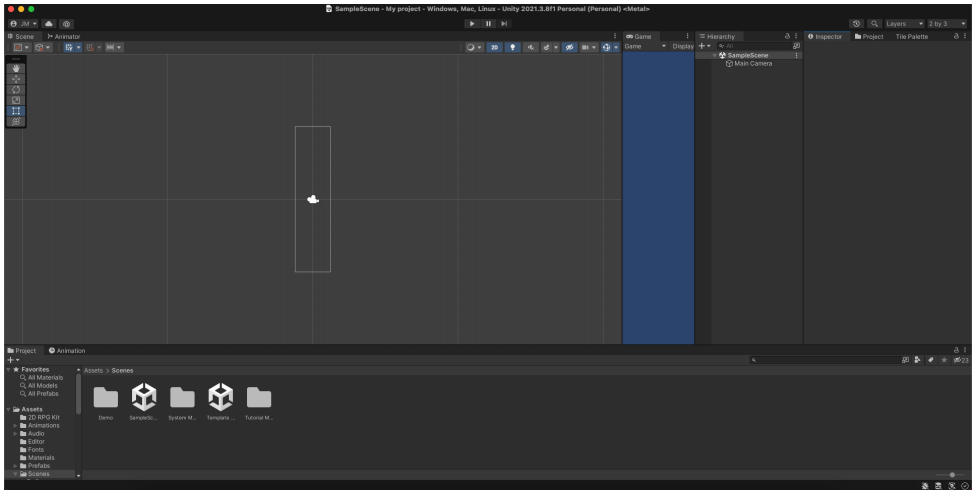

Figura 5.11: Unity - Flujo modificado

A la izquierda cambiaré entre la escena y el grafo de animaciones, la pantalla del juego estará un poco más a la derecha y en el caso de ser necesaria se hará más grande, acto seguido podemos ver la jerarquía de la escena, el inspector, los archivos del proyecto y la paleta de tiles, en la parte inferior disponemos de nuevo los archivos y las barra de animaciones por tiempo.

Según las pruebas realizadas, esta es una de las interfaces más eficientes para el trabajo con juegos en 2D.

5.4 Gameobjects, prefabs e inspector

A partir de este punto se utilizarán conceptos como gameobjects, prefabs y el inspector de forma bastante avanzada, así que en este punto voy a explicarlos brevemente, pues su uso se complicara en los puntos posteriores.

5.4.1 Gameobjects

Los gameobjects son la forma que tiene Unity de trabajar con sus componentes (Herramientas creadas por unity para facilitar el trabajo). Las escenas están compuestas por gameobjects los cuales poseen otros componentes dentro, como por ejemplo: sprites, scripts, mánager de audio, cámaras, Etc.

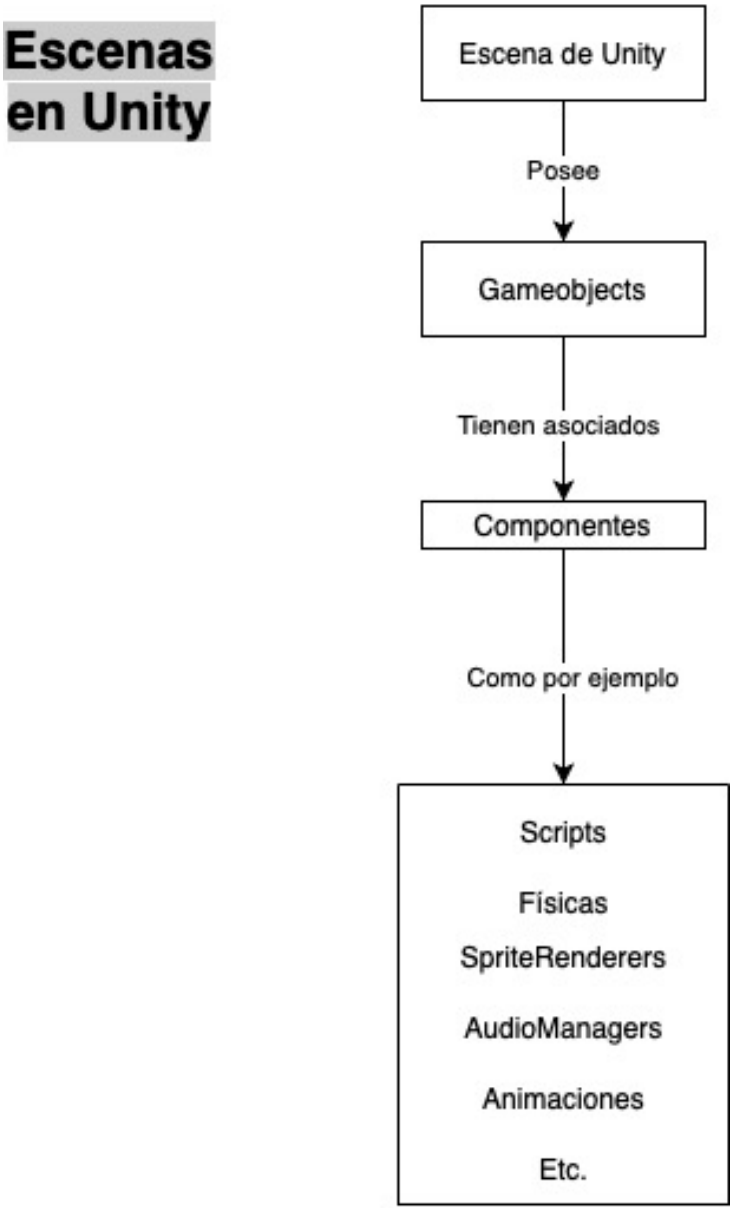

Figura 5.12: Unity - Gameobjects

Los componentes mínimos que necesita un gameobject para funcionar son su nombre, capa, etiquetas y su posición en el espacio de tridimensional de la escena, pero pueden tener asociados la cantidad de elementos que sea necesario. Con esto lo normal es tener diversos gameobjects que controlen distintas partes de la aplicación para así crear módulos reutilizables en otras partes del juego.

Por ejemplo, podríamos crear un gameobject que incorporara un componente sprite renderer con la forma de un cofre cerrado, incorporar físicas con un rigidbody2D para que el jugador pueda detectar colisiones cuando se choque, animarlo con un animator y asociar un script al mismo que hará que cuando un jugador se acerque y pulse una tecla este reproduzca la animación de abrirse y añada un objeto al inventario del usuario.

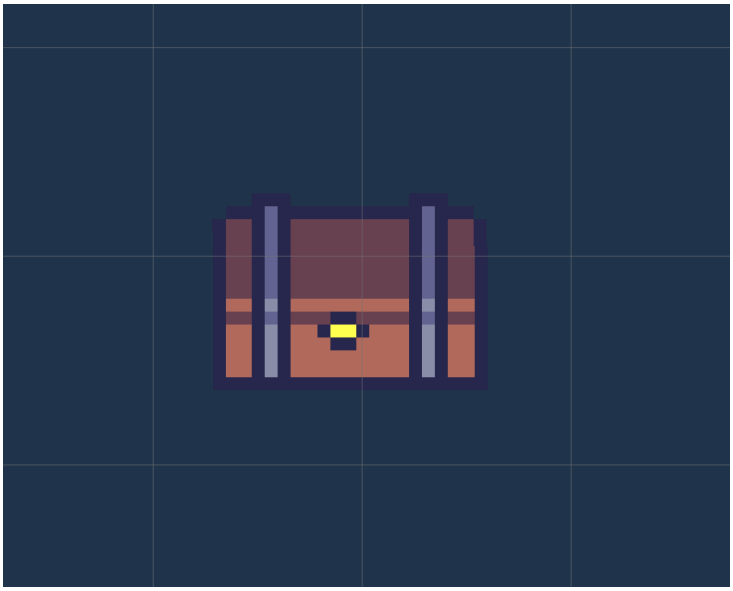

Figura 5.13: Unity - Cofre situado en una escena

Este tipo de gameobject es reutilizable en más de una escena y aquí es donde entran en juego los prefabs.

5.4.2 Prefabs

Los prefabs son gameobjects que han sido creados en una escena, pero que se plantea querer utilizarlos en otra escena, en el ejemplo situado al final del punto anterior se utilizaba un cofre para explicar de forma sencilla su uso, pero esto puede extenderse a sistemas enteros. Para crear un prefab a través de un gameobject solo hay que arrastrarlo fuera de la escena hacia los archivos, este se guardara como un archivo .prefab y podrá ser arrastrado a otras escenas.

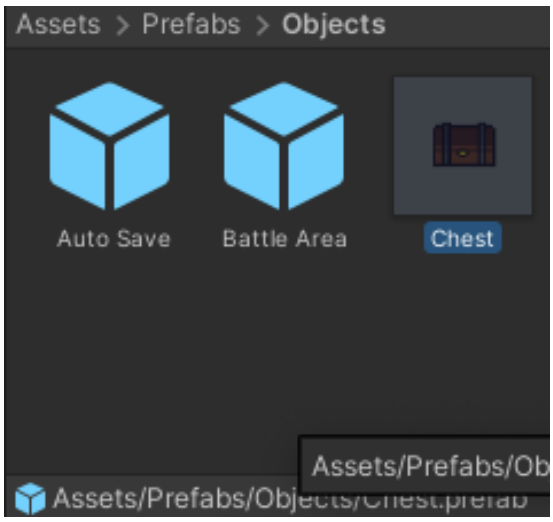

Figura 5.14: Unity - Prefab de un cofre

Una de las mejores partes que tiene el sistema de prefabs de Unity es que si editas el prefab principal se editarán todas las escenas que lo utilicen, a su vez Unity es capaz de crear prefabs padres e hijos. Los prefabs padres actualizarían los componentes de los hijos, pero los de los hijos no los de los padres, exactamente igual que funciona la herencia de clases en la programación orientada a objetos.

5.4.3 Inspector

El inspector de Unity nos permite ver los componentes que posee un gameobject y modificar los parámetros de los mismos para adecuarlos a lo que necesitemos, por ejemplo el cofre anterior se vería de la siguiente manera:

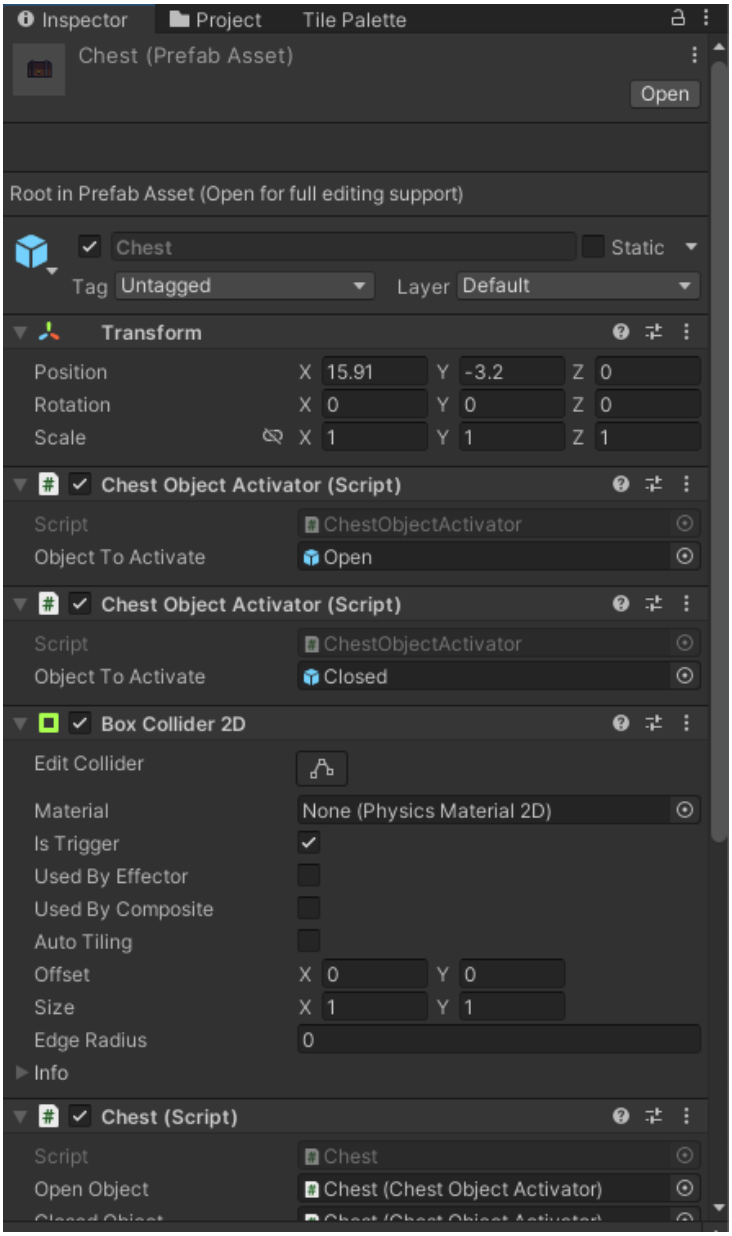

Figura 5.15: Unity - Editor que muestra el prefab de un cofre

Como se puede ver podemos utilizar el inspector para arrastrar elementos y utilizarlos en los componentes asociados, consiguiendo así una programación muy visual y haciendo que reutilizar scripts o prefabs sea bastante sencillo. Además, una vez hacemos cambios en un gameobject y guardamos la escena Unity guarda estos cambios internamente, con lo cual las modificaciones pueden ser permanentes si así se desea, esto consigue que podamos programar scripts empleando las funciones del editor para avanzar más rápido en el desarrollo.

Una de las mejores ventajas es que además Unity permite la modificación en tiempo real de parámetros desde el editor con el juego en marcha para propósitos de testeo, incrementando también notablemente la velocidad de ajuste de parámetros.

5.5 Pantalla principal

Una vez hemos llegado a este punto y hemos acabado la preparación pertinente, empezamos con el desarrollo del prototipo.

Disponemos de una pantalla principal en el Kit 2D, la cual ya incorpora muchas de las mecánicas que necesitamos. Está hecha con la herramienta canvas de Unity la cual es utilizado para situar todos los elementos de la interfaz de usuario dentro. El mismo canvas incorpora un gestor de eventos para detectar cada vez que se cumplen ciertas condiciones y así lanzar el evento correspondiente en consecuencia.

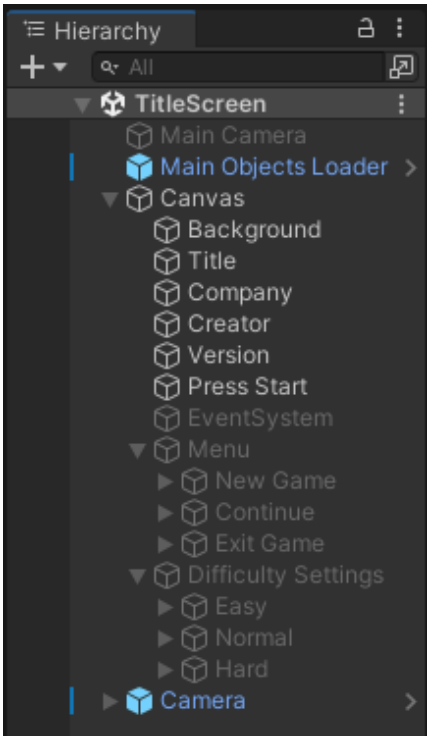

Figura 5.16: Jerarquía de Unity - Escena de título

En este caso, la escena por defecto da una base que incorpora una imagen de fondo con un título, diversos botones para iniciar o continuar la partida y un selector de dificultad al seleccionar nuevo juego. Todo está gestionado con el sistema de eventos del canvas y dentro del mismo sitúan componentes como botones, imágenes y textos, todos ellos siendo herramientas por defecto de Unity.

Los eventos son gestionados por un script situado en el gameobject del canvas principal, el script se encarga de hacer aparecer y desaparecer elementos en función de los clicks, de cambiar de escena si se carga partida o si se comienza una nueva, de establecer el nivel de dificultad en una nueva partida y de reproducir música. Todo ello siguiendo las recomendaciones de Unity de utilizar el editor para no tener que modificar el script cada vez que se necesite realizar algún cambio.

Esta pantalla incorpora las funciones que necesitamos por ahora, con lo cual simplemente se van a dar un toque estético a la misma y a modificar la posición de los elementos respetando el reescalado por resolución del canvas.

Para modificar la posición de los elementos en el canvas debemos emplear el gestor del mismo, el cual nos permite indicar una posición de entre nueve para el elemento y acto seguido corregir la misma añadiendo píxeles en los ejes X o Y.

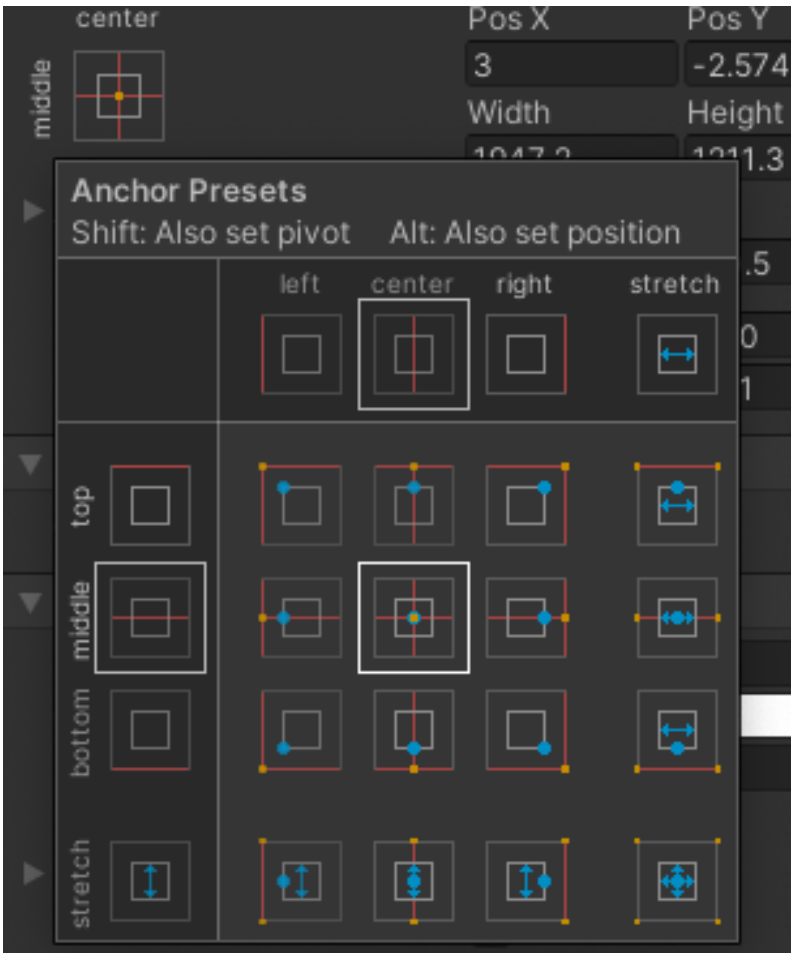

Figura 5.17: Editor de Unity - Modificar posición de un elemento de canvas

Haciéndolo todo de esta manera y no modificando la posición añadiendo píxeles directamente, conseguimos mucha consistencia entre resoluciones y hacemos que el reescalado mantenga la proporción de la interfaz de usuario correctamente.

La pantalla una vez finalizada queda dispuesta de la siguiente manera:

Figura 5.18: Paranoia - Pantalla principal

Como se ha utilizado la herramienta mencionada previamente para situar los elementos, la resolución funciona de forma consistente en resoluciones de 16:9, las cuales son las más utilizadas por los jugadores de PC y consolas.

5.6 Preparación de los assets gráficos para el mapeado

Una vez hemos realizado la pantalla de inicio, el siguiente paso que quiero realizar es la preparación de los archivos del mapeado de la zona de entrenamiento de la corporación.

Se importaron los assets necesarios en un punto anterior, pero estos no están preparados para utilizarse todavía. Usualmente, los sprites vienen en tilesets, los cuales son un conjunto de sprites en la misma imagen.

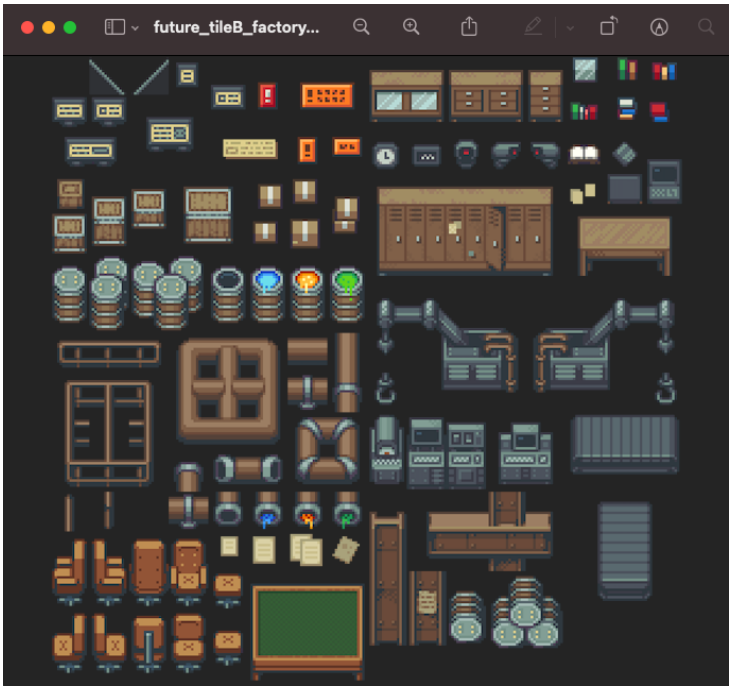

Figura 5.19: Paranoia - Future Tileset

Antes de poder utilizarlos necesitamos decirle a Unity que en la misma imagen existen diferentes tiles, lo primero que debemos hacer es seleccionar la foto e ir al inspector para modificar la forma en la cual Unity tratara al archivo:

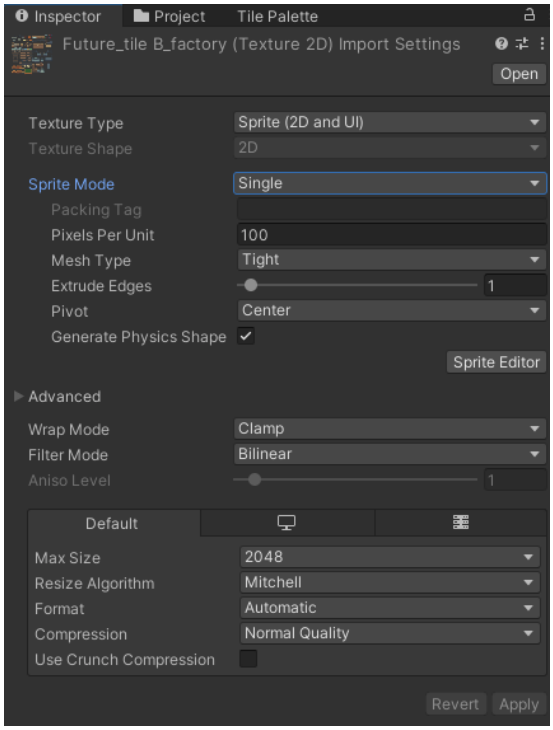

Figura 5.20: Unity - Inspector al seleccionar una imagen

Las opciones que nos aparecen nos permiten indicar si dentro de la imagen existen varias imágenes o solamente una, los píxeles por elemento, la posición del pivote y además nos da muchas opciones de compresión con el objetivo de aligerar la carga en el juego. Este tipo de imágenes deben de tratarse sin ningún tipo de compresión, en cualquier otro caso los píxeles se ven algo borrosos y afecta mucho a la calidad visual.

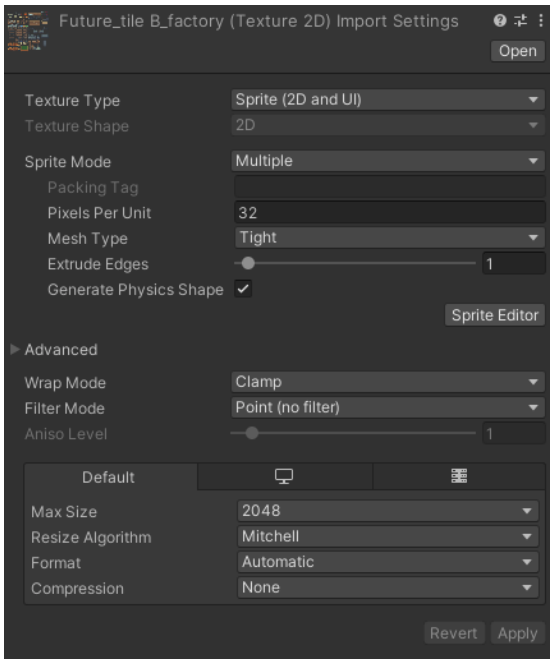

Figura 5.21: Unity - Inspector al seleccionar una imagen (2)

De esta forma podemos indicar a Unity que existen varias imágenes dentro de la misma, que tienen 32 píxeles por unidad y que no utilice filtros ni compresión.

El siguiente paso será dividir la imagen utilizando el editor de sprites que nos brinda Unity, para ello solo debemos hacer click en el botón correspondiente en el editor y se nos abrirá una ventana para poder realizar los recortes:

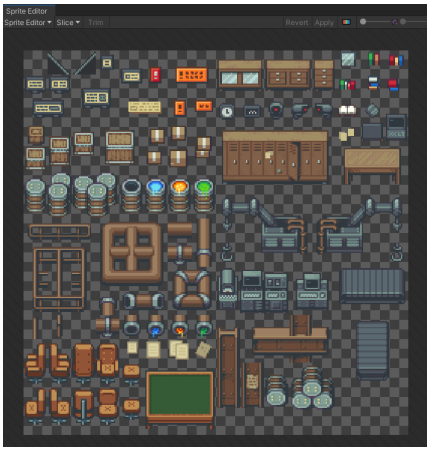

Figura 5.22: Unity - Sprite Editor

Aquí debemos seleccionar la opción de recortes e indicar que haga recortes de 32 x 32 píxeles y acto seguido pulsar el botón de aplicar. En el momento esto quede hecho, al seleccionar la imagen en el inspector de archivos podremos ver que esta ha sido recortada correctamente:

Figura 5.23: Unity - Imagen después del recortado

Se realizarán estos pasos cada vez que se necesite recortar una imagen.

5.7 Main Objects Loader

Este gameobject se asocia a la jerarquía de todas las escenas y lleva asociado un script que usa el patrón ´Mánager´ y que porta diversos gameobjects encargados de gestionar todos los sistemas del juego.

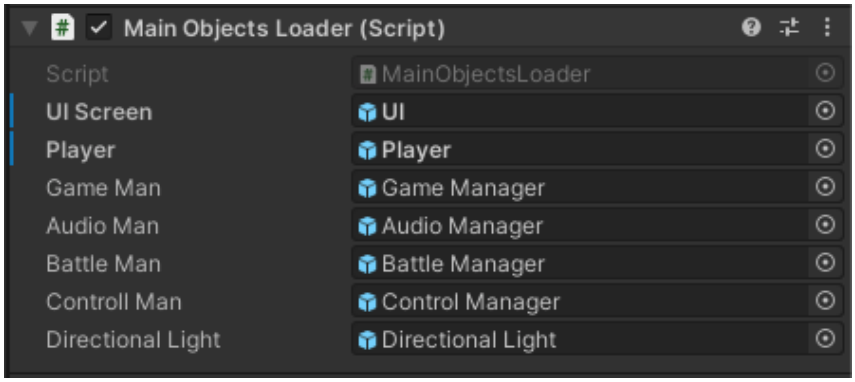

Figura 5.24: 2D RPG Kit - Main Object Loader

Todos serán utilizados salvo los dos últimos, ya que por ahora no serán necesarios.

5.7.1 Main Objects Loader - UI

Este gameobject estará encargado de mostrar toda la interfaz de usuario del juego, la cual está desarrollada con la herramienta canvas explicada previamente.

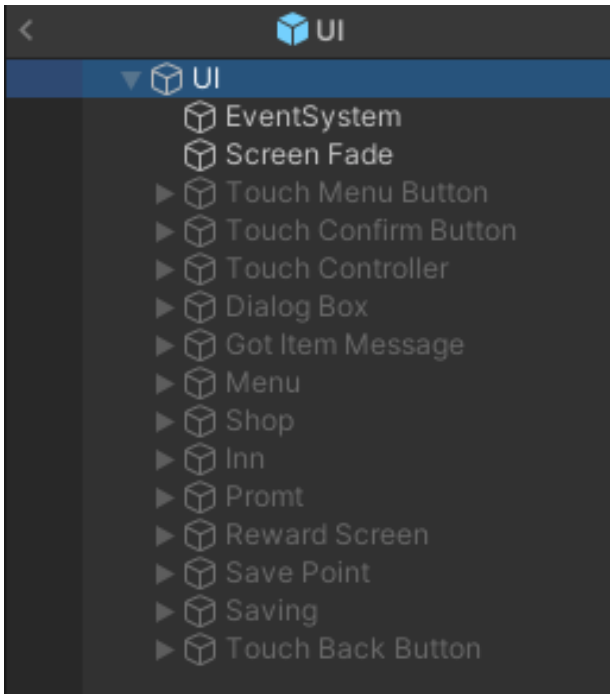

Figura 5.25: Main Objects Loader - UI

Como se puede ver en su estado original, dentro del gameobject padre existe un hijo para cada menú del juego, los cuales originalmente se encuentran desactivados y se activan en los momentos en los que el juego los requiera. Esto permite que para implementaciones futuras solo sea necesario seguir el mismo esquema.

5.7.2 Main Objects Loader - Player

Este gameobject trae un sprite renderer que llevara el sprite del jugador, diversos colliders encargados de gestionar las colisiones del mismo con el entorno, un script que gestiona el movimiento, un rigidbody para que pueda ser afectado por las físicas y por último un gameobject hijo con un box collider 2D encargado de gestionar los encuentros con otros NPC.

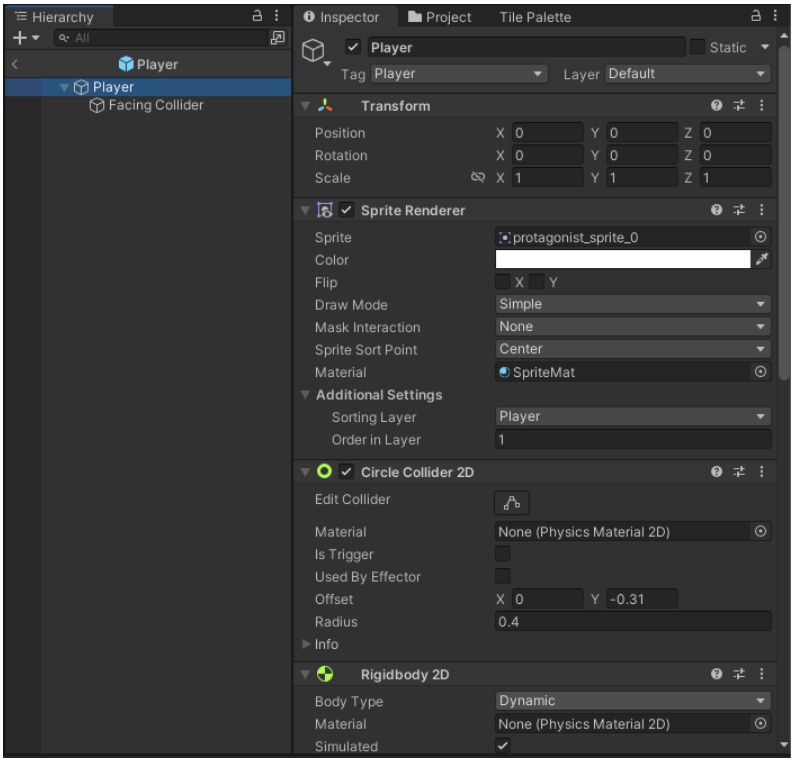

Figura 5.26: Main Objects Loader - Player

Una vez modificado con el sprite renderer asignado a nuestro jugador y con las medidas apropiadas para el mismo, todo queda de esta forma:

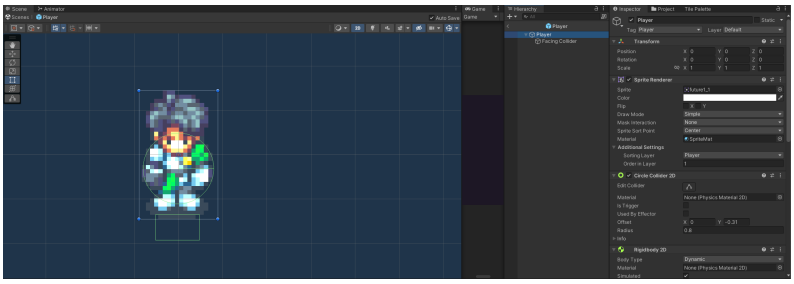

Figura 5.27: Main Objects Loader - Player (2)

5.7.3 Main Objects Loader - Game Manager

Este gameobject es encargado de gran parte de la lógica del juego e incorpora lo siguiente:

- Lógica de los personajes disponibles junto con sus estadísticas y habilidades.
- Lógica de los cofres existentes y un registro de los que han sido abiertos hasta el momento en la partida.

- Lógica de las tareas existentes y un registro de las que han sido realizadas hasta el momento en la partida.
- Lógica de los eventos existentes y un registro de los que han sido realizados hasta el momento en la partida.
- Registro de todos los items disponibles en el juego.
- Tamaño del inventario y objetos actuales en él.

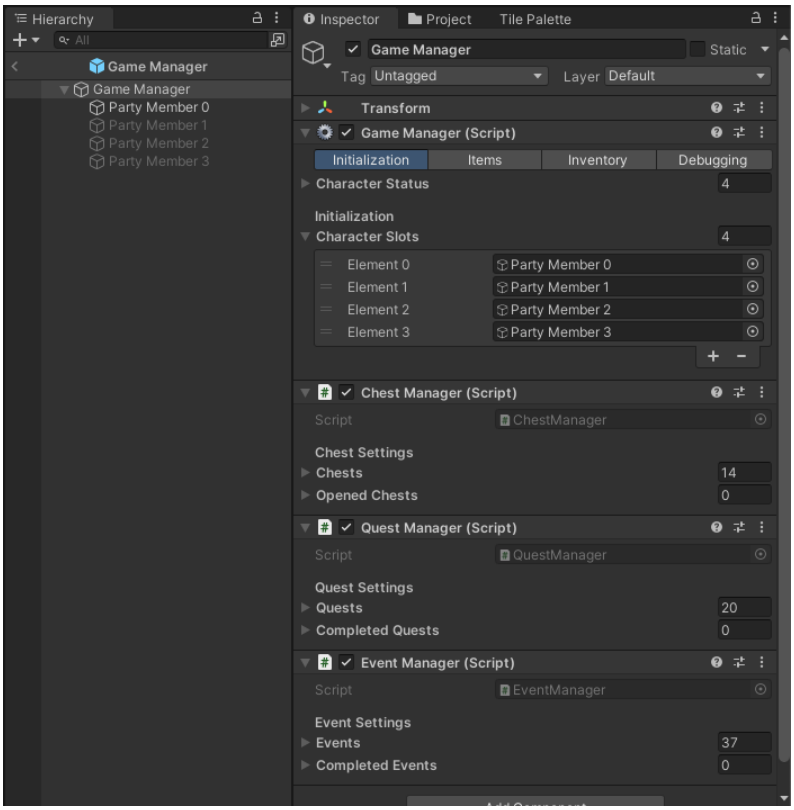

Figura 5.28: Main Objects Loader - Game Manager

Todo es editable desde el inspector y la forma en la que trabaja permite la extensión de forma muy óptima, haciendo que simplemente tengas que clonar prefabs, editarlos y añadirlos.

5.7.4 Main Objects Loader - Audio Manager

Este gameobject es el encargado de llevar un registro de toda la música existente en el juego:

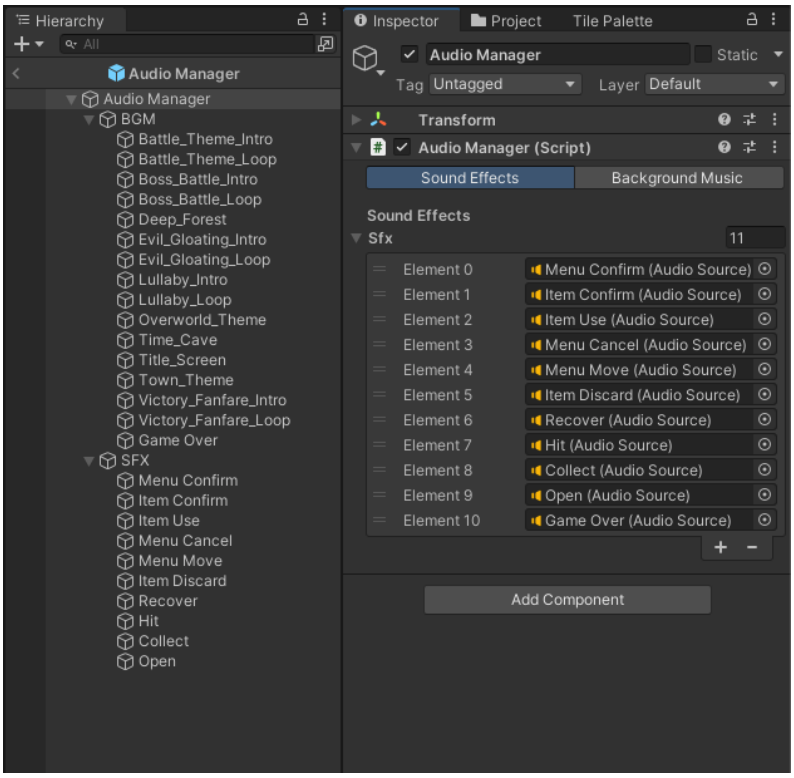

Figura 5.29: Main Objects Loader - Audio Manager

Cada gameobject posee un componente Audio Source de Unity el cual nos permite controlar totalmente cuando se reproducen los sonidos y como. El Audio Manager actúa más bien como una base de datos de sonidos, los cuales son llamados desde otros scripts, por ejemplo, el script de cámara hace referencia al ID asociado al Audio Manager para encontrar el sonido a reproducir.

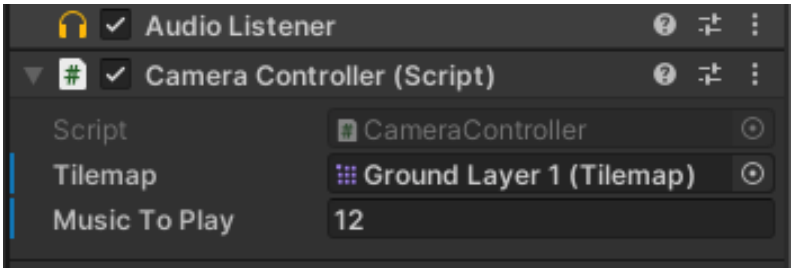

Figura 5.30: Main Objects Loader - Audio Manager - Cámara

5.7.5 Main Objects Loader - Battle Manager

El battle mánager gestiona toda la interfaz de las batallas y su flujo, contiene lo siguiente:

- Gestión de toda la UI de batalla

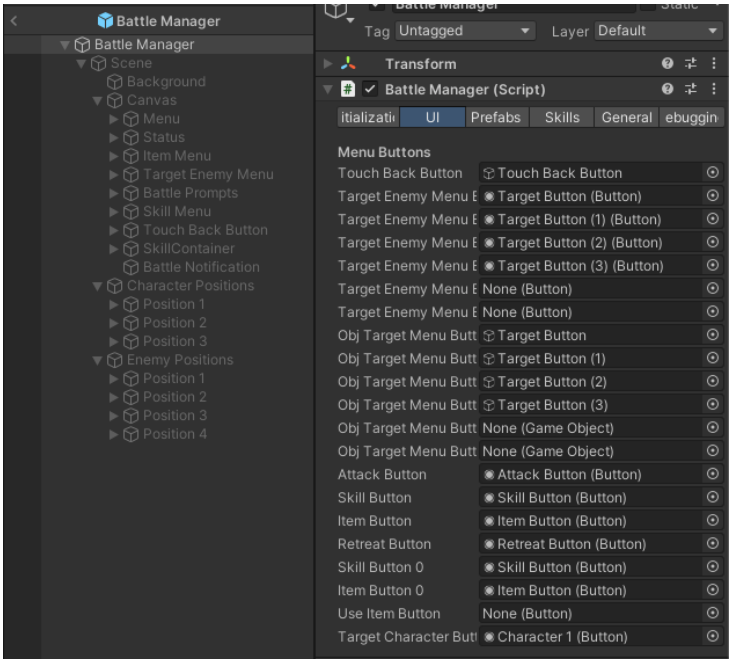

Figura 5.31: Main Objects Loader - Battle Manager UI

- Todos los personajes disponibles con su posición de batalla almacenados en un prefab que contiene las animaciones de la batalla utilizadas por el script que controla el flujo.

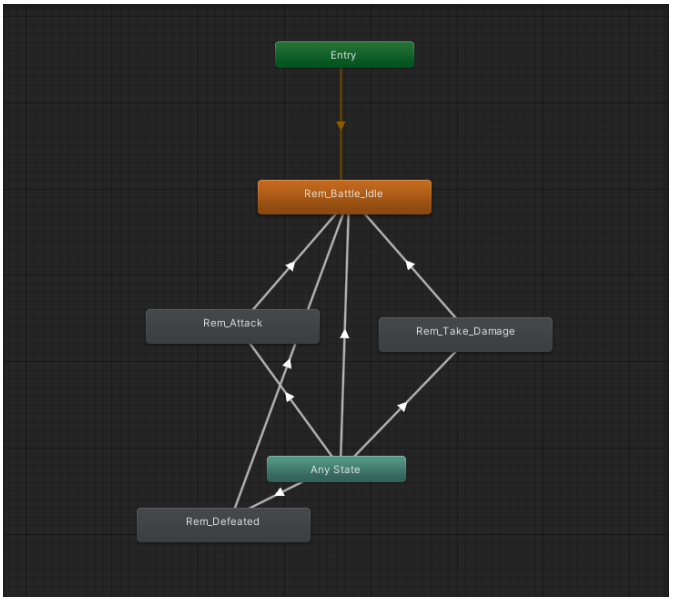

Figura 5.32: Main Objects Loader - Battle Manager - Animaciones de un personaje

- Todas las habilidades existentes junto a sus efectos

Figura 5.33: Main Objects Loader - Battle Manager - habilidades

- La tasa de retirada y la escena de fin del juego que enlaza con el menú principal

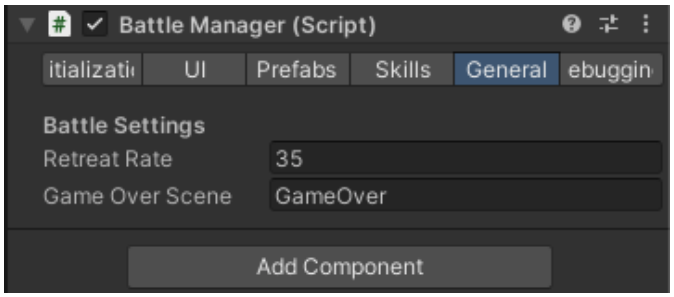

Figura 5.34: Main Objects Loader - Battle Manager - General

Todo se controla desde el inspector, sigue las buenas prácticas recomendadas por Unity y hace que su modificación sea muy simple.

5.8 Mapeado

Con los assets ya preparados, ya podemos empezar a dibujar el mapeado del juego. Para ello, lo primero será ir a la escena base del Kit 2D que hemos descargado previamente. Una vez dentro de la escena podemos ver que incorpora lo siguiente:

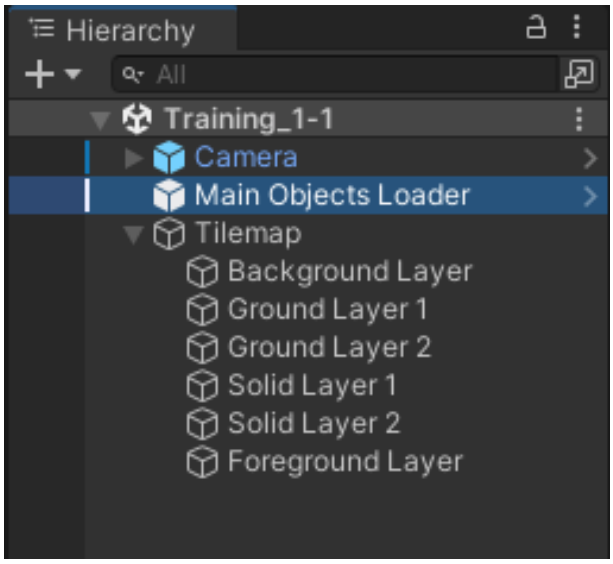

Figura 5.35: Unity - Escena por defecto (2D RPG Kit)

La cámara dispone de un script asociado el cual seguirá al jugador siempre que se encuentre en el tileset marcado y además será la encargada de reproducir música de ambiente, el ´Main Objects Loader´ explicado previamente y un tilemap con las capas que se ven en la figura 5.35 y las cuales están ordenadas de la siguiente manera:

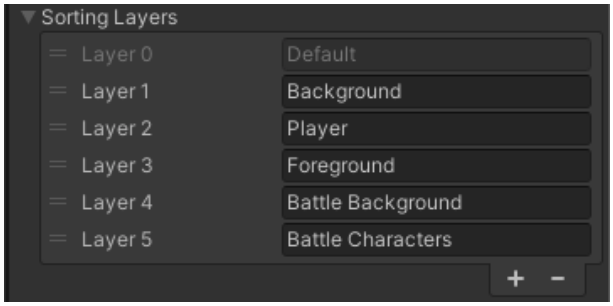

Figura 5.36: Unity - Capas

- Background:
  - Capa utilizada en todas las capas menos en la ´Foreground Layer´. La ´Background Layer´, ´Ground Layer 1´ y ´Ground Layer 2´ no contienen colisiones, mientras que la ´Solid Layer 1‘ y la ´Solid Layer 2´ si que contienen colisiones aplicando un tilemap collider y composite collider a las mismas.

- Player:
  - Hace referencia solo al jugador que siempre se encuentra en esta capa.
- Foreground:
  - Capa auxiliar empleada en ‘Foreground Layer‘ por si necesitamos mostrar algo por encima del jugador, no contiene colisiones, por lo que es utilizada sobretodo para cuando queremos que el jugador pueda rodear algo por detrás.
- Battle Background:
  - Esta capa es usada para el fondo de las batallas, debe ir por encima del resto, ya que cuando una batalla comienza el juego se pausa y el fondo tapa todo lo demás.
- Battle Characters:
  - Es usada por los jugadores y enemigos en posición de batalla.

Una vez sabemos como funcionan las capas podemos crear una ´Tile Palette´ y empezar a pintar el mapa. Para crear una nueva ´Tile Palette´ y aplicarle los sprites cortados previamente, debemos ir a Window->2D->Tile Palette y acomodar la ventana que aparece en función de nuestros gustos. En esta ventana podremos darle a Create a new Palette y guardar una nueva paleta en los archivos del proyecto. Cuando hayamos hecho esto podremos arrastrar los sprites cortados a la paleta para poder empezar a organizarlos.

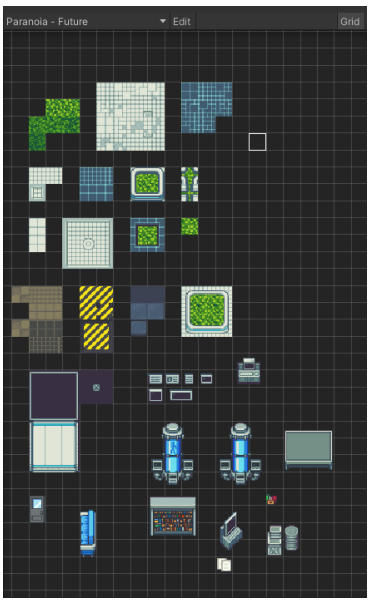

Figura 5.37: Unity - Tile Palette

Por mi experiencia es mejor ir poco a poco pasando sprites a la paleta para ir dejándolos ordenados, si se pasan muchos sprites de golpe se desordenan y luego es imposible dejarlos bien. Una vez tenemos la paleta lista podemos utilizar las herramientas de dibujo para dibujar en las capas mencionadas anteriormente. De esta forma, el resultado final del mapeado para el prototipo fue el siguiente:

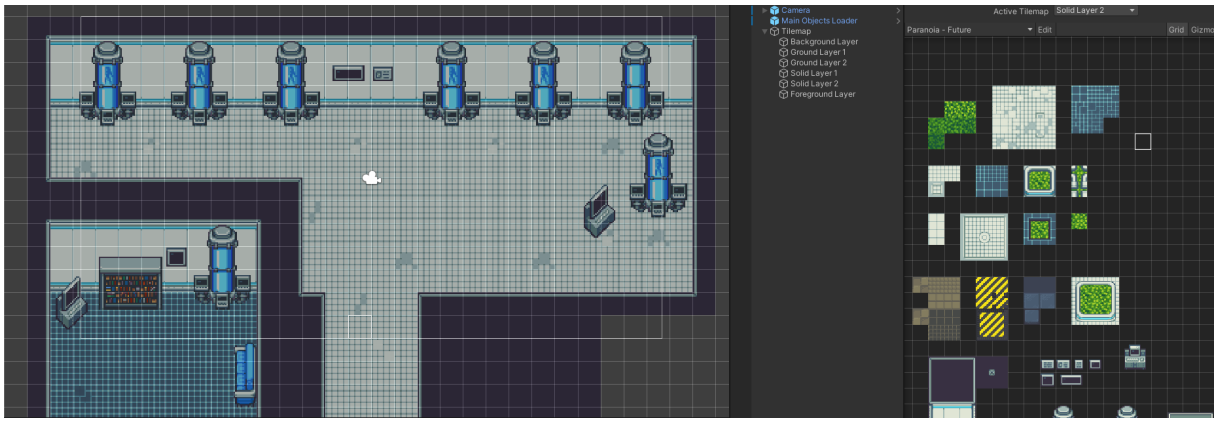

Figura 5.38: Unity - Mapeado de la primera zona

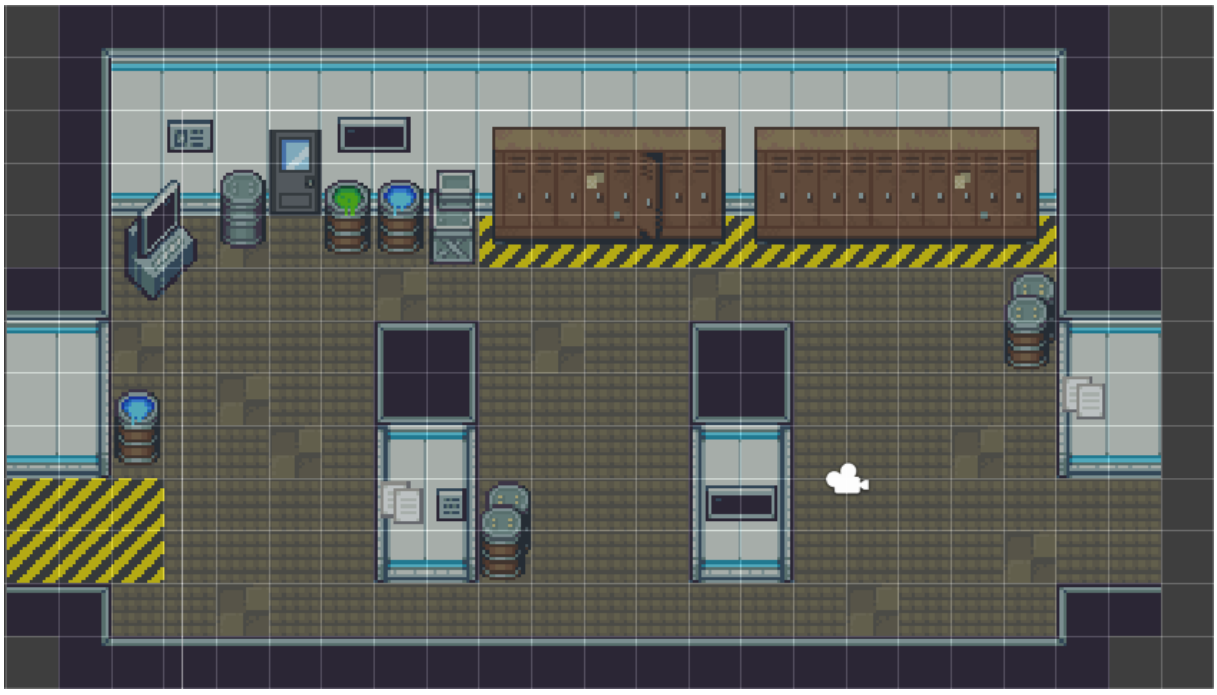

Figura 5.39: Unity - Mapeado de la segunda zona

Figura 5.40: Unity - Mapeado de la tercera zona

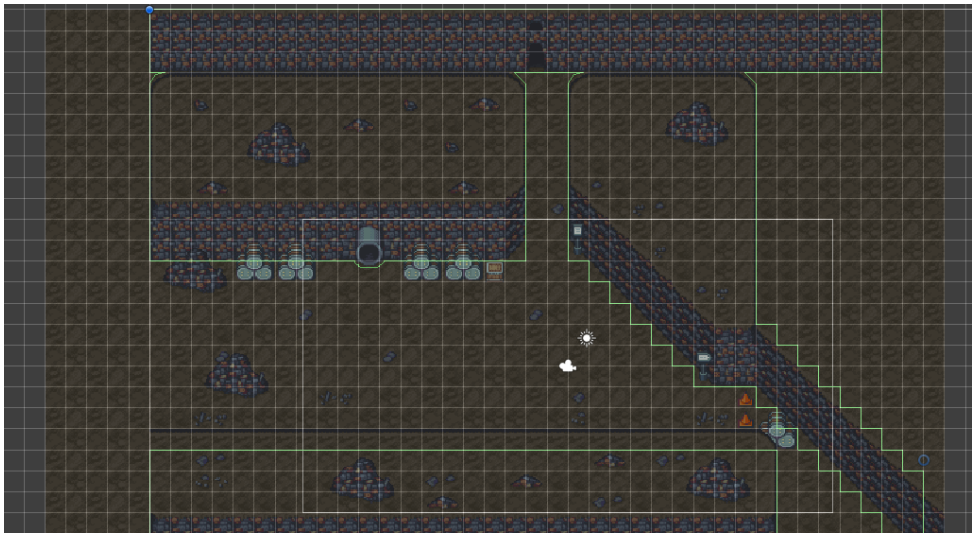

Figura 5.41: Unity - Mapeado de la cuarta zona

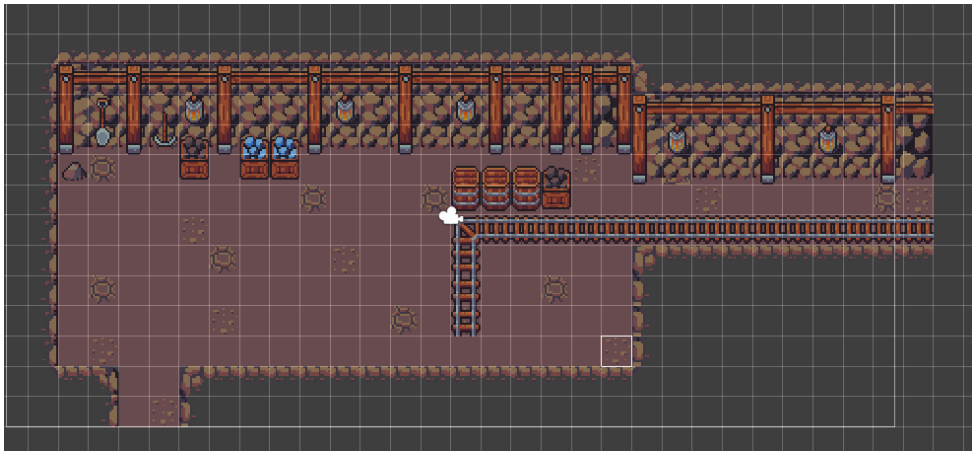

Figura 5.42: Unity - Mapeado de la quinta zona

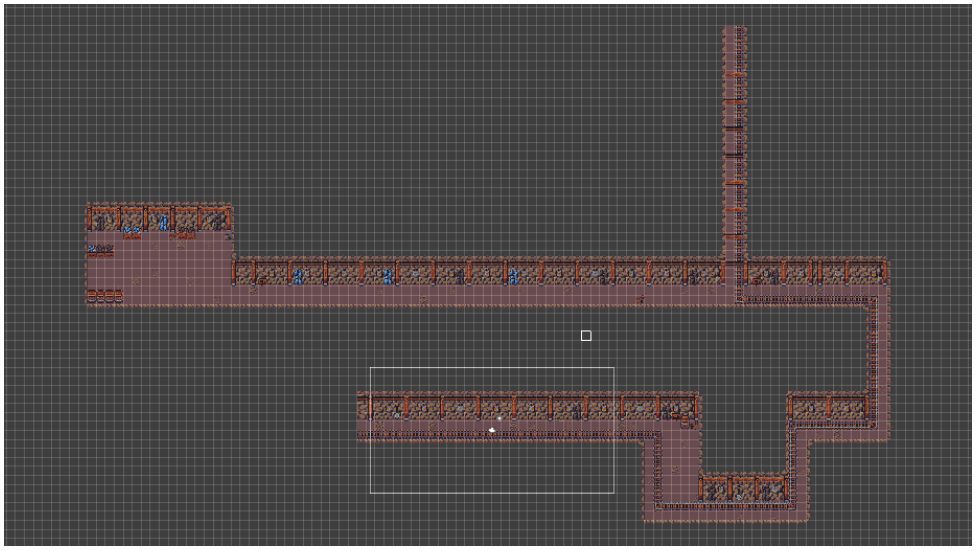

Figura 5.43: Unity - Mapeado de la sexta zona

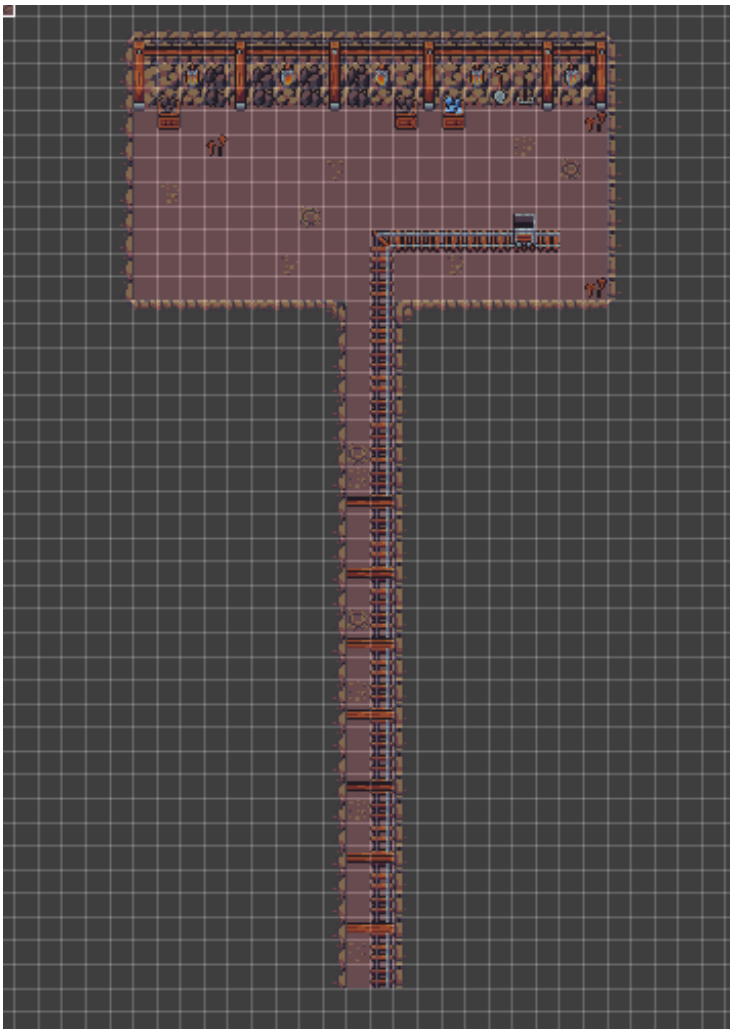

Figura 5.44: Unity - Mapeado de la séptima zona

5.9 Animaciones

Una vez tenemos hechos los mapas, el siguiente paso será modificar las animaciones existentes desde el Kit 2D y crear nuevas para adecuarlas a nuestros sprites. Para esto se utilizan las herramientas ´Animation´ y ´Animator´ de Unity.

5.9.1 Animation

La herramienta Animation es usada para indicar lo que hace una animación, esto implica en este caso los sprites por los que pasa, cada cuanto tiempo cambia de sprite y cuantos frames dura en total.

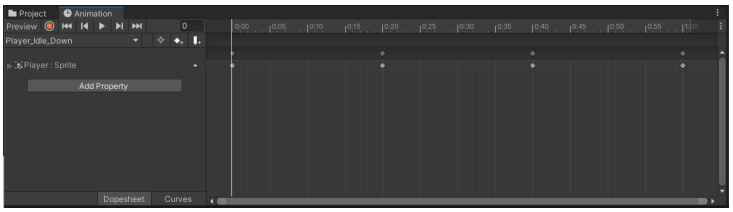

Figura 5.45: Unity - Herramienta ´Animation´

Para utilizar esta herramienta hay que crear una animación e ir arrastrando los sprites por los que queremos que pase a la barra de tiempo situada en la derecha. Una vez aquí hay que ajustar los tiempos de las animaciones hasta que vayan a la velocidad deseada. En este caso se modificarán las animaciones de caminar, idle y batalla del jugador principal siguiendo estos pasos.

Para poder probar las animaciones lo mejor es moverlas a una escena y utilizar el botón play de la herramienta, de esta forma la animación se moverá en la escena y podremos ver lo rápido que se mueve.

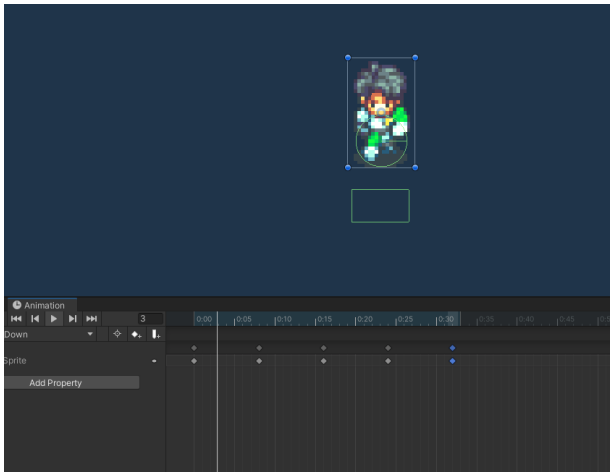

Figura 5.46: Unity - Herramienta ´Animation´ - En uso

5.9.2 Animator

La herramienta Animator sirve para añadir variables que se utilizaran para realizar transiciones entre animaciones. Por ejemplo, cuando el parámetro velocidad del personaje sea nulo, debemos hacer que deje de reproducir la animación que camina y pase a la de idle. Para facilitar esta tarea, Unity utiliza programación visual por medio de grafos y variables asociadas al componente, de forma que pueden ser llamadas desde los scripts utilizando el mismo componente ´Animator´.

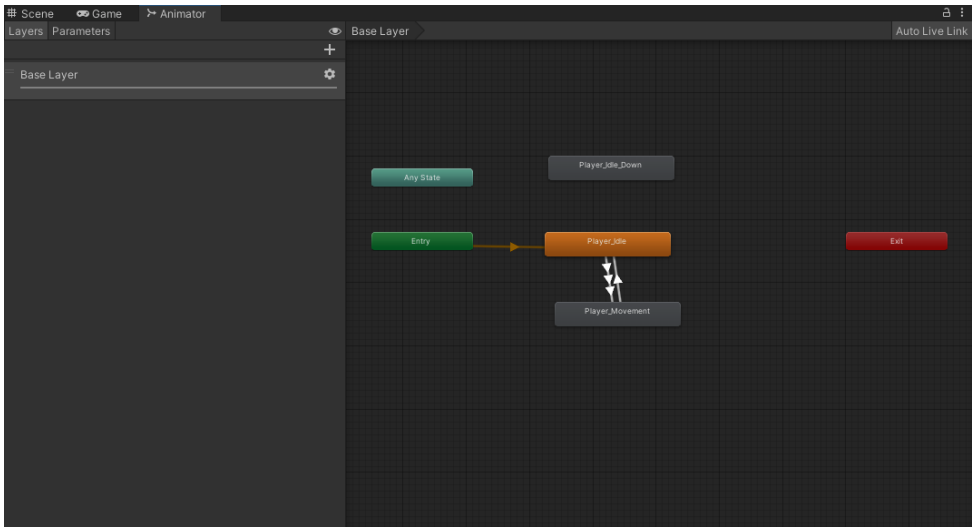

Figura 5.47: Unity - Herramienta ´Animator´

En este punto, ya podemos ver a nuestro jugador moverse en el mapeado creado previamente:

Figura 5.48: Unity - Jugador en una escena

5.10 NPC

En este punto vamos a incorporar los NPC al juego, disponemos de tres tipos de NPC creados en el ´2D RPG Kit´, los cuales se dividen en tiendas que permiten comprar y vender objetos del inventario, NPC que recuperan vida al usuario a cambio de la moneda del juego y NPC que entablan conversación y pueden otorgar tareas o completarlas. Por ahora no serán necesarios más NPC, pero el script donde están creados los NPC existentes hace que sea muy simple crear más.

5.10.1 NPC - Tienda

Para configurar una tienda primero debemos crear objetos, para hacer esto debemos irnos a los objetos predefinidos por el ´2D RPG Kit´ y duplicar cualquiera de ellos. Cuando hayamos hecho esto podremos cambiar en el inspector los parámetros de objeto e indicar si es un objeto de equipo, si recupera vida, si revive al usuario, Etc. También debemos indicar un valor de ataque o de defensa en el caso de ser un equipo o un valor de recuperación si es un objeto consumible.

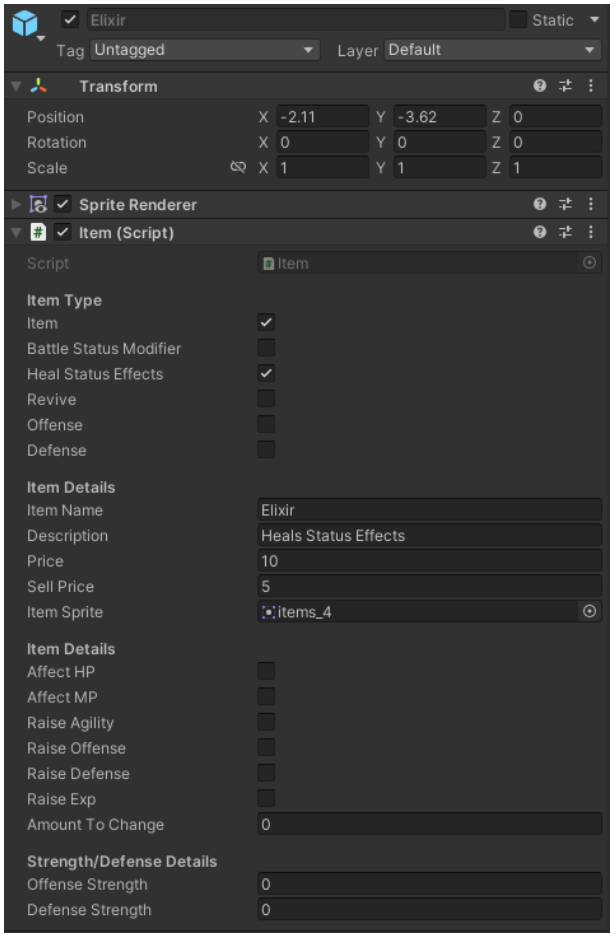

Figura 5.49: 2D RPG Kit - Objeto de ejemplo

Siguiendo el esquema propuesto se han creado los siguientes objetos:

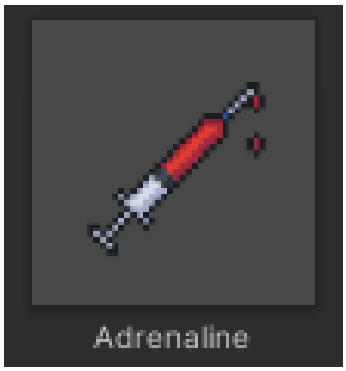

Figura 5.50: Paranoia - Objeto - Adrenalina

Figura 5.51: Paranoia - Objeto - Sudadera de la corporación

Figura 5.52: Paranoia - Objeto - Bebida de Electrolitos

Figura 5.53: Paranoia - Objeto - Kit de primeros auxilios

Figura 5.54: Paranoia - Objeto - Espada oxidada

Figura 5.55: Paranoia - Objeto - Medicina de nivel 1

Una vez con los objetos ya creados solo debemos ir al prefab NPC, duplicarlo, indicar que es una tienda y los objetos que vende, cambiar los sprites para adecuarlos a nuestro juego y situarlo en la escena correspondiente. Si a lo largo del juego ofrecemos más tiendas, solo deberemos duplicar el prefab y repetir el proceso. Una vez realizado este proceso y adaptada la interfaz de usuario, la tienda queda de la siguiente manera:

Figura 5.56: Paranoia - NPC - Tienda

5.10.2 NPC - Recuperar vida

Todos los RPG tienen un sistema de descanso para que el usuario recupere vida y en este caso el kit también trae uno. Para utilizarlo solo debemos adaptar el prefab correspondiente, cambiar los sprites, editar el texto y el precio y situarlo en la escena correspondiente, una vez finalizado queda de la siguiente manera:

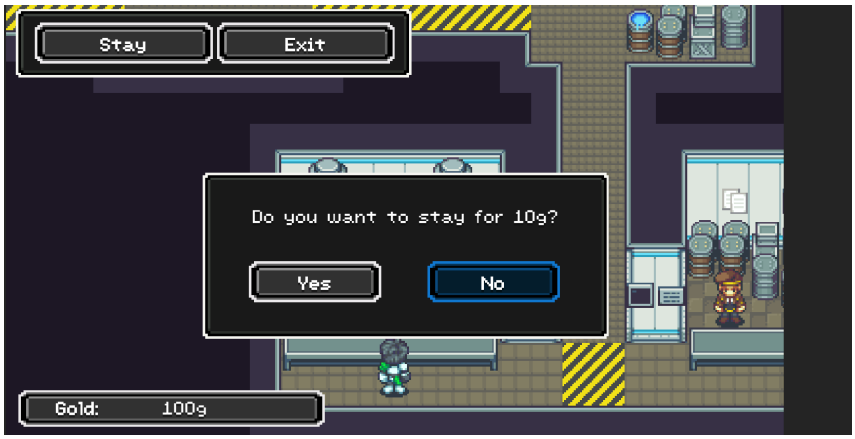

Figura 5.57: Paranoia - NPC - Descanso

5.10.3 NPC - Diálogo

Para crear NPC que simplemente nos digan cosas, solo debemos arrastrar a la escena correspondiente el prefab de NPC simple, cambiar los sprites y los diálogos. Una vez situados algunos en las escenas correspondientes podemos ver esto:

Figura 5.58: Paranoia - NPC - Diálogo

5.10.4 NPC - Tareas

Para que un NPC te otorgue una tarea, primero se deben crear las tareas en el Game Manager. Simplemente, debemos ir y añadirlas al array de tareas, una vez hecho esto se han creado 8 tareas para el prototipo.

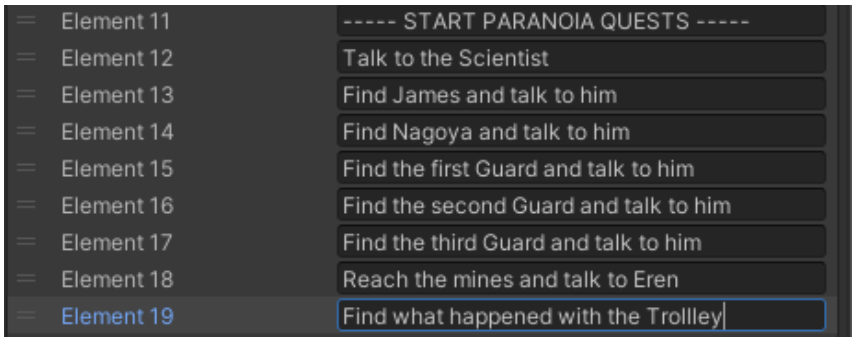

Figura 5.59: Paranoia - Tareas

Para hacer que un NPC nos dé una tarea o la complete solo debemos indicarlo en el inspector, además, el kit trae un script para permitir la comprobación de tareas en orden de activar y desactivar NPC, con esta herramienta podemos crear el flujo de la historia y forzar al jugador a realizar diversas tareas poniendo bloqueos en las zonas a las que no queremos que acceda hasta que no complete las tareas correspondientes. Una vez realizado esto tenemos el flujo de la historia del prototipo:

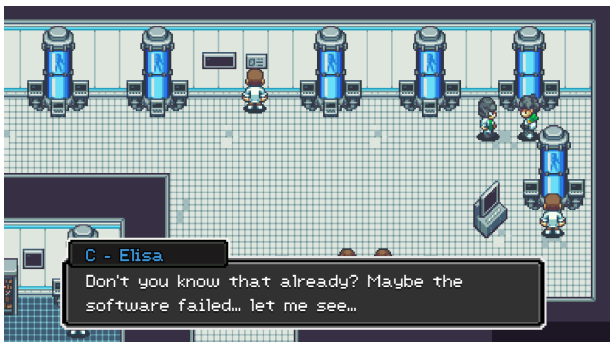

Figura 5.60: Paranoia - Historia - 1

Cuando el jugador aparece en la primera escena hacemos que un NPC le hable instantáneamente utilizando una opción del prefab NPC donde podemos hacer que el NPC empiece la conversación, en el momento el collider del jugador y el del NPC se choquen, el primer NPC manda una tarea al jugador y lo manda a buscar a otro NPC que está situado en otro punto de la escena, acto seguido, el primer NPC desaparece.

Figura 5.61: Paranoia - Historia - Guardia

El guardia de abajo del todo no nos permite pasar hasta que completemos la tarea hablando con el NPC correspondiente, en este momento desactivaremos el gameobject del guardia con el script de comprobar si una tarea se ha realizado o no.

Figura 5.62: Paranoia - Historia - 2

Seguimos el flujo y hacemos que este NPC nos dé una misión para ayudar a limpiar de animales las minas de carbón cercanas.

Figura 5.63: Paranoia - Tarea completada

Cuando acabamos de hablar con el NPC marcamos la tarea como completada e informamos al jugador. Para continuar la historia se utilizará este método.

5.11 Batallas

Una vez el usuario salga de la zona de investigación tendrá que afrontar batallas, para hacer que el jugador pueda pelear vamos a usar de nuevo el kit mencionado a lo largo del desarrollo.

5.11.1 Creación de enemigos

Lo primero que hay que hacer es crear enemigos, estos se crean de la misma que se han creado NPC's y objetos. Hay que duplicar el prefab de un enemigo existente, editarlo y registrarlo en el mánager correspondiente, en este caso el ´Battle Manager´, se utilizaran los siguientes enemigos:

Figura 5.64: Paranoia - Enemigo - Murciélago mutante

Figura 5.65: Paranoia - Enemigo - Abeja mutante

Figura 5.66: Paranoia - Objeto - Serpiente mutante

5.11.2 Fondo de la batalla

Para el fondo de la batalla se ha creado una escena como las usadas en la mina, pues es donde se situaran los enemigos, la escena ha quedado de la siguiente manera:

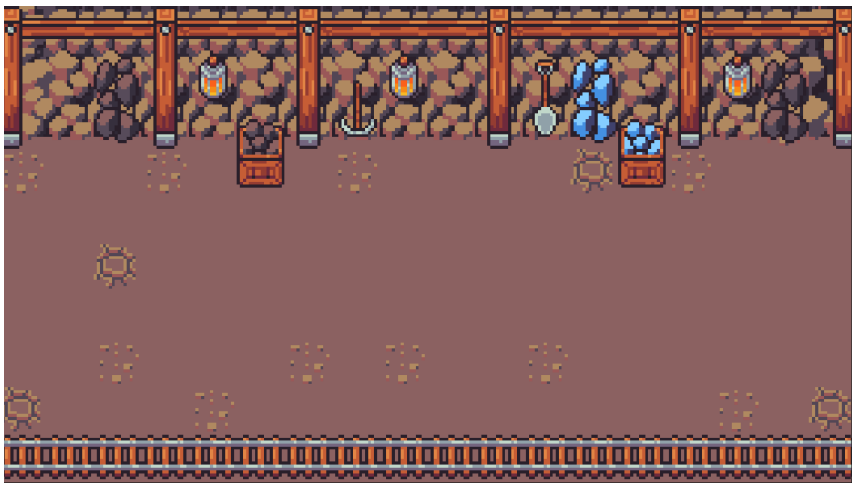

Figura 5.67: Paranoia - Batalla - Fondo

5.11.3 Empezar una batalla aleatoria

Para esto utilizaremos un prefab llamado ´Battle Zone´, este permite designar un ´box collider´ alrededor de una zona y propiciar encuentros aleatorios con los grupos de enemigos que configuremos desde el inspector al caminar por esta zona.

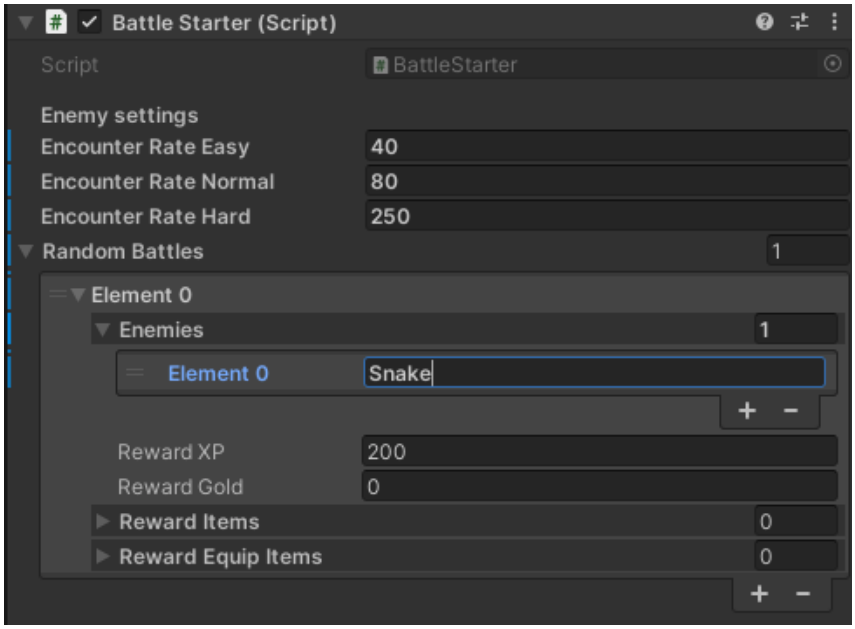

Figura 5.68: Paranoia - Batalla - Zona de batalla

Una vez configurada una batalla, cuando caminemos por la zona hay una posibilidad aleatoria programada desde el inspector de entrar en combate, una vez en combate podemos ver la escena completa:

Figura 5.69: Paranoia - Batalla en curso

Una vez finaliza la batalla con una victoria se nos dan puntos de experiencia:

Figura 5.70: Paranoia - UI - Puntos de experiencia

Y si el enemigo nos causa algún tipo de estado alterado que permanezca durante la batalla, lo podemos ver desde el menú:

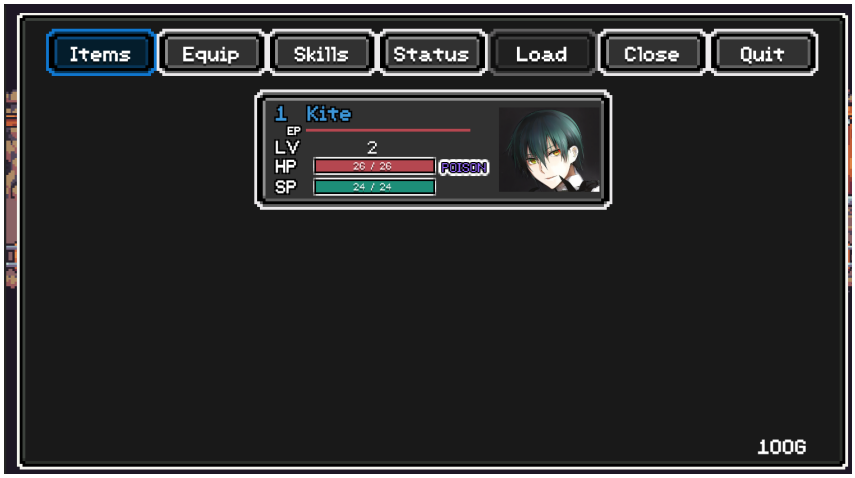

Figura 5.71: Paranoia - UI - Menú

6 Pruebas

Para todos los sistemas diseñados se ha realizado una extensa fase de pruebas, primero probándolo personalmente encontré fallos en los siguientes puntos:

- Sistema de tareas:
  - El sistema de tareas mostraba un error al hablar con cualquier NPC, estos al final de cada ejecución lanzaban un evento con la intención de completar una quest incluso si no había ninguna quest marcada para completar. Esto se arregló aplicando un checkbox al inspector del script y añadiendo un if en consecuencia al trigger del evento.
- Cámara:
  - La cámara no seguía correctamente al jugador si el mapa era más pequeño que la amplitud de la cámara, esto sucedía debido a que la cámara nunca contemplaba esta posibilidad. Se solucionó haciendo que la cámara mantuviese un punto fijo siempre y cuando el mapa fuera más pequeño que la amplitud total que la cámara podía cubrir.

Una vez arreglados estos problemas, decidí enviar mi juego a algunos amigos y les pedí que fueran los más objetivos que fuera posible e intentaran romper el juego, estos encontraron los siguientes errores:

- Tienda:
  - El menú del juego podía ser abierto con el menú de la tienda abierto, esto hacía que las interfaces se solaparan rompiendo el juego y necesitando reiniciarlo, se arregló aplicando un condicional que revisaba si había algún menú abierto antes de abrir el menú principal.

- Posada:
  - El menú del juego podía ser abierto con el menú de la posada abierto, esto hacía que las interfaces se solaparan rompiendo el juego y necesitando reiniciarlo, se arregló aplicando un condicional que revisaba si había algún menú abierto antes de abrir el menú principal.
- Diálogos:
  - El menú del juego podía ser abierto con un diálogo abierto, esto hacía que las interfaces se solaparan rompiendo el juego y necesitando reiniciarlo, se arregló aplicando un condicional que revisaba si había algún diálogo abierto antes de abrir el menú principal.

Y, también encontraron diversos problemas con el balance del juego que impedían a los usuarios avanzar en determinados puntos:

- Sistema de batalla
  - En cualquier modo de dificultad, algunos de los primeros enemigos eran demasiado fuertes, esto hacía que el usuario no tuviera la oportunidad de entrenar en una zona anterior para prepararse para la siguiente. Se arregló haciendo la primera zona mucho más asequible, reduciendo los parámetros de todos los enemigos en cualquier modo de dificultad.
- Tienda
  - Los objetos de la tienda eran muy caros para la cantidad de oro que daban los enemigos, esto se solucionó haciendo que NPC que te otorgaba la misión de ir a la mina te diera algo de dinero para prepararte en tu aventura. Además, se ha reducido el precio del objeto de curación menos potente. Los objetos no son imprescindibles para avanzar en la aventura, pero sí que hace que no tengas que ir a la posada tan asiduamente y sea menos tedioso y más divertido jugar.

7 Conclusión

Teniendo en cuenta objetivos planteados al inicio del trabajo, podemos comentar lo siguiente:

- Aprendizaje:
  - Aprender sobre la industria del videojuego y más enfocado a la producción de los mismos.
    - Se ha logrado aprender mucho sobre la industria del videojuego, llevando a cabo el análisis tecnológico y estudiando las modas pasadas y actuales.
  - Aprender sobre las fases reales de desarrollo de un videojuego utilizadas por los estudios más exitosos en 2022.
    - Se ha logrado aprender lo máximo posible sobre las fases que se usan en entornos reales, estudiando casos de éxito de diferentes estudios que han compartido sus métodos de trabajo a través de los libros citados a lo largo de todo el trabajo.
  - Aprender y estimar costos monetarios y temporales realistas en el desarrollo de los videojuegos.
    - Se ha logrado realizar estudios de tiempos y de costes teniendo en cuenta los gastos de producción de otros estudios.
  - Aprender como analizar el mercado e intentar anticiparse a él.
    - Se ha utilizado herramientas estadísticas como ´Google Trends´ para estudiar el mercado pasado e intentar anticipar el futuro.
  - Aprender a gestionar el tiempo en desarrollos de software largos.

    - Se realizó un calendario para el prototipo en el cual los tiempos estimados han sido bastante cercanos a los tiempos reales.

  - Aprender a utilizar un motor gráfico ampliamente utilizado en la industria.

    - Desde el inicio del proyecto, indudablemente se ha mejorado mucho con el uso de Unity. Pero, aun así, siento que mis conocimientos sobre el motor siguen siendo algo bajos como para considerarme un usuario de un nivel aceptable.

  - Aprender a gestionar los recursos disponibles para llevar un proyecto a su fin de forma exitosa.

    - Se han usado los tiempos y recursos monetarios de manera eficiente, logrando que el desarrollo finalice en el tiempo estimado y creando un prototipo atractivo para los jugadores.

  - Aprender a gestionar riesgos de forma controlada, siempre buscando la mayor rentabilidad posible.

    - Considero que se ha invertido bien el tiempo y el dinero empleados en el proyecto y que se ha gestionado el riesgo de manera correcta.

- Desarrollo

  - Ser capaz de desarrollar un prototipo en el tiempo estimado.

    - El prototipo ha sido completado eliminando algunas mecánicas innecesarias que se plantearon el día que se escribió la descripción del trabajo, además se pudo detectar antes de la planificación que estas mecánicas no iban a ser necesarias, con esto, considero que este objetivo también ha sido un éxito.

  - Utilizar buenas prácticas a lo largo de todo el proyecto.

    - Gracias a la reutilización de assets de terceros y a intentar ser consciente de que la organización es clave, se han extendido sistemas manteniendo las buenas prácticas y haciendo más código y prefabs reutilizables en el futuro.

  - Emplear las herramientas más óptimas para realizar cada tarea.

    - Considero que el estudio sobre software a utilizar fue acertado y que se emplearon los mejores estándares de la industria para cada tarea.

  - Reutilizar código o assets que permitan mejorar la calidad del proyecto y acortar tiempos de producción.

    - Gracias al buen análisis efectuado en fases tempranas, se logró encontrar muy buenos assets que permitieron la finalización de un prototipo con muchos sistemas en solo unos meses.

Como conclusión final, puedo decir que sin duda este trabajo me ha servido para poner al límite muchos aspectos aprendidos durante mis años en el grado y en mi vida profesional. Estoy muy contento con el resultado final, pues he podido realizar un prototipo muy completo en un tiempo récord gracias a haber utilizado las herramientas adecuadas para ello. Este proyecto me ha hecho aprender sobre gestión de riesgos, gestión de tiempos, optimización de tareas, marketing de producto, depuración de errores y sobretodo sobre fases de desarrollo de un proyecto. Siento que este proyecto me ha hecho crecer como profesional y me hace dar valor a todos los años invertidos en mi formación hasta ahora.

7.1 Relación del trabajo desarrollado con los estudios cursados

Una vez acabado el desarrollo del prototipo, puedo relacionar muchas de las competencias adquiridas a través de las asignaturas cursadas en el grado con los resultados del trabajo desarrollado:

- Programación, estructuras de datos y algoritmos, ingeniería del software, técnicas de optimización y Algorítmica.

  - Las competencias adquiridas en estas asignaturas han sido claves para el correcto desarrollo del videojuego. Gracias a ellas, puedo decir que se han utilizado las mejores prácticas posibles de legibilidad, reusabilidad y optimización del código. Además, se ha conseguido un código eficiente en el ámbito de rendimiento en diferentes equipos.

- Análisis matemático, estadística, fundamentos físicos de la informática y álgebra.

  - Gracias a las competencias adquiridas en estas asignaturas se ha podido valorar de forma correcta los gráficos y las estadísticas analizadas a lo largo del proyecto. A su vez, estas asignaturas han reforzado mi base matemática, permitiendo esto entender los sistemas tridimensionales de Unity y las fórmulas vectoriales aplicadas sin demasiado esfuerzo.

- Desarrollo de videojuegos, introducción a los sistemas gráficos interactivos.

  - Estas asignaturas me han permitido conocer más a fondo muchos de los conocimientos sobre motores gráficos y resoluciones aplicados en este trabajo, además, se realizaron diversos proyectos a pequeña escala que ayudaron a que un juego más grande como este fuera mucho más sencillo de abordar.

- Fundamentos de organización de empresas, deontología y profesionalismo.

  - Estas asignaturas me permitieron adquirir competencias con las que entender como funcionan empresarialmente los estudios de videojuegos, ayudando así a entender su toma de decisiones en función del riesgo a afrontar.

8 Objetivos de desarrollo sostenible

Objetivos de Desarrollo Sostenible	**Alto**	**Medio**	**Bajo**	**No procede**
ODS 1. **Fin de la pobreza.**			X	
ODS 2. **Hambre cero.**			X	
ODS 3. **Salud y bienestar.**		X		
ODS 4. **Educación de calidad.**		X		
ODS 5. **Igualdad de género.**				X
ODS 6. **Agua limpia y saneamiento.**		X		
ODS 7. **Energía asequible y no contaminante.**		X		
ODS 8. **Trabajo decente y crecimiento económico.**		X		
ODS 9. **Industria, innovación e infraestructuras.**				X
ODS 10. **Reducción de las desigualdades.**	X			
ODS 11. **Ciudades y comunidades sostenibles.**	X			
ODS 12. **Producción y consumo responsables.**	X			
ODS 13. **Acción por el clima.**	X			
ODS 14. **Vida submarina.**		X		
ODS 15. **Vida de ecosistemas terrestres.**		X		
ODS 16. **Paz, justicia e instituciones sólidas.**	X			
ODS 17. **Alianzas para lograr objetivos.**	X			

8.1 Reflexión sobre la relación del TFG con los ODS y con el/los ODS más relacionados.

En este caso, creo que mi TFG esta directamente relacionado con muchos de los objetivos mencionados en la tabla superior, el juego desarrollado trata sobre un mundo bizarro post-apocalíptico el cual se sitúa en un futuro en el cual muchos de esos objetivos obviamente no se han cumplido, es más diría que esos objetivos han ido a peor en los años siguientes a la trama. Se podría decir que con los objetivos que más se identifica son los siguientes: Paz, justicia e instituciones sólidas (El juego empieza después de la tercera guerra mundial provocada enteramente por que la falta de recursos en las ciudades es totalmente insostenible), Ciudades y comunidades sostenibles (De haberse cumplido este objetivo de forma real es muy probable que nunca se hubiese llegado al punto de no retorno que provoco la guerra de la historia), Producción y consumo responsables (En el caso de cuidar los recursos del medio ambiente y tratarlos de forma responsable, la historia del videojuego tendría muchas menos posibilidades de acabar pasando) y Acción por el clima (La cual va relacionada con todos los puntos anteriores). Todos estos objetivos tienen mucho que ver con la historia del juego, la cual a pesar de estar basada en una trama de ciencia ficción de hace aproximadamente tres décadas y que en su día no fuese pensada para ello, hoy en día es perfecta para concienciar a la población de un posible escenario en el cual una guerra por los últimos recursos del planeta puede provocar la infelicidad de muchas personas y la extinción del ser humano. Me gustaría mucho que si este juego acaba saliendo a la venta en el futuro, este pueda ser usado como una referencia más de lo importante que es cuidar el medio ambiente y respetar los ecosistemas en los que vivimos. A su vez, pienso desarrollar la trama del juego con muchos de estos puntos en consideración, intentando que así quien lo juegue a parte de pasar un buen rato vea las atrocidades cometidas por muchos seres humanos a los cuales no les importa acabar con los ecosistemas del mundo por su mero beneficio. De esta forma, me gustaría que los jugadores del juego puedan relacionar que todos estos puntos son necesarios para la simple supervivencia y que no son algo opcional que puede hacerse o no hacerse, por que, de hecho, de no hacerse. Un mundo bizarro y absurdo como el del juego podría llegar a ser posible.

Bibliografía

barnardos.org.uk. (2021). *Why gaming is better for your mental health than you might think*. https://www.barnardos.org.uk/blog/why-gaming-better-your-mental-health-you-might-think. (Vid. pág. 2).

Chang, J. (2022). *51 Significant Video Game Demographic Statistics: 2022 Data on Age and Gender*. https://financesonline.com/video-game-demographic-statistics. (Vid. pág. 3).

Fox, J. (2018). *Video Games Are Bad For You*. https://georgiastatesignal.com/video-games-are-bad-for-you. (Vid. pág. 2).

Games, G. G. (2015). *RPG Maker MV Steam*. https://store.steampowered.com. (Vid. pág. 37).

Howarth, J. (2022). *7 Huge Gaming Industry Trends 2022-2025*. explodingtopics.com. (Vid. pág. 23).

Levenson, D. (2021). *The Importance of Time Management for Software Development*. insureyourcompany.com. (Vid. pág. 17).

Marhulets, W. (2020). *GAMEDEV, 10 Steps to making your first game successful Editorial: Unfold Games ISBN-10: 1735232505*. Unfold Games, LLC. (Vid. págs. 16, 17, 19, 36).

Schreier, J. (2017). *Blood, Sweat, and Pixels*. Harper Paperbacks. (Vid. pág. 17).

Tyroller, J. (2021). *10 Steps To Making a Successful game*. https://www.youtube.com/watch?v=GoQ2D (Vid. pág. 16).

UPC. (2019). *Historia de los videojuegos*. https://www.fib.upc.edu/retro-informatica/historia/videojocs (Vid. pág. 7).

Wijman, T. (2022). *Games Market Revenues Will Pass 200 Billion Dollars for the First Time in 2022*. https://newzoo.com/. (Vid. pág. 1).