Pachangas World Player

TFG - Videojuegos

Autor: Víctor Iglesias Posse

Tutor: Raúl Montoliu Colás

Profesor: Joan Arnedo Moreno

Grado de Ingeniería Informática

Computación

24/03/2024

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Pachangas World Player*
Nombre del autor:	*Víctor Iglesias Posse*
Nombre del colaborador/a docente:	*Raúl Montoliu Colás*
Nombre del PRA:	*Joan Arnedo Moreno*
Fecha de entrega (mm/aaaa):	*03/2024*
Titulación o programa:	*Grado de Ingeniería Informática*
Área del Trabajo Final:	*Videojuegos*
Idioma del trabajo:	*Español*
Palabras clave	*Arcade, multijugador, fútbol*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados y conclusiones del trabajo*

Pachangas World Player es el trabajo final de grado perteneciente al itinerario de Computación, que consiste en el desarrollo completo de un videojuego 2D, multijugador y arcade, de temática futbolística. El juego contará con una estética personalizada, junto con mecánicas y habilidades específicas, para las cuales se tendrán que cubrir distintas ramas técnicas: diseño gráfico, programación, sonido, animación, etc.

Debido a las características del proyecto, se pretende optar por una metodología en cascada con la planificación de tareas especificadas en el diagrama de Gantt correspondiente, pero también se pretende el uso de las herramientas que facilita GitHub (*Issues* y *Projects*) para llevar un control más detallado de los cambios en el código.

A pesar de la limitación temporal de un cuatrimestre para finalizar el proyecto, el producto final tratará de contar con todos los elementos necesarios para ser jugado en su totalidad, si bien en el futuro podría ampliarse el contenido con nuevos jugadores, escenarios, movimientos especiales, *power-ups*, etc.

Aun tratándose de un proyecto académico, este se abordará como otros desarrollos de software existentes en el mundo laboral: estableciendo una planificación detallada, generando la documentación necesaria y contando con una organización rigurosa para el éxito del producto.

Github: https://github.com/viglesiaspo/TFG
Video Youtube PEC 2: https://www.youtube.com/watch?v=-V84q93h8Gs
Video Youtube PEC 3: https://www.youtube.com/watch?v=K_zz8B1pLws

Versión Alpha (v1.0.0) PEC 3: https://github.com/viglesiaspo/TFG/releases/tag/Alpha
Versión Final (v2.0.0) PEC 4: https://github.com/viglesiaspo/TFG/releases/tag/Final

Abstract (in English, 250 words or less):

Pachangas World Player is the final degree project belonging to the Computer Science itinerary, which involves the full development of a 2D, multiplayer, arcade-style video game centred around the soccer theme. The game will have a personalized aesthetic, along with specific mechanics and skills, for which different technical branches will have to be covered: graphic design, programming, sound, animation, etc.

Due to the project characteristics, it is intended to choose a waterfall methodology along with the task planning specified in the corresponding Gantt diagram, but it is also intended to use the tools provided by GitHub (*Issues* and *Projects*) to keep more detailed control of code changes.

Despite the four months limitation period for development, the final product will try to have all the necessary elements to be fully played, although in the future the content could be expanded with new players, scenarios, special moves, power-ups, etc.

Even though it is an academic project, this will be approached like other software developments that exist in the world of work: establishing detailed planning, generating the necessary documentation and having a rigorous organization for the success of the product.

Github: https://github.com/viglesiaspo/TFG

Video Youtube PEC 2: https://www.youtube.com/watch?v=-V84q93h8Gs
Video Youtube PEC 3: https://www.youtube.com/watch?v=K_zz8B1pLws
Alpha Version (v1.0.0) PEC 3: https://github.com/viglesiaspo/TFG/releases/tag/Alpha
Final Version (v2.0.0) PEC 4: https://github.com/viglesiaspo/TFG/releases/tag/Final

A Manel, allí donde estés.

Índice

Figuras

Índice de figuras

1. Introducción

1.1. Prefacio

El uso de videojuegos sigue experimentado un crecimiento progresivo, no solo en términos de usuarios y ventas, sino también en su impacto social y cultural. Junto con el auge en el uso de *skins* (personajes del juego u opciones cosméticas para el personaje), son los fenómenos en los que se basa el proyecto seleccionado como trabajo final del grado de informática.

El proyecto cuenta con un componente diferencial que genera una motivación extra para el éxito del mismo: la personalización de los escenarios, música y jugadores, basados en la caracterización, movimientos, gustos, etc. de los amigos del autor. El producto final no se limitará solamente a un ámbito académico, sino que va a poder ser disfrutado como un regalo para los protagonistas del mismo.

Pachangas World Player representa la culminación de los estudios en Ingeniería Informática y se perfila como la oportunidad ideal para aplicar los conocimientos adquiridos a lo largo de la carrera. El desarrollo de un videojuego es un proceso integral que abarca todas las etapas del ciclo de vida del software y, no solo demanda una sólida base técnica, sino también una fuerte dosis de creatividad, dada la necesidad de incorporar distintos elementos audiovisuales y garantizar una experiencia de usuario óptima.

1.2. Descripción

El videojuego a desarrollar para el trabajo final del grado de informática es "*Pachangas World Player*" (en adelante PWP), un arcade multijugador de fútbol, que se presenta con unas *skins* personalizadas basadas en el público específico al que va dirigido.

Este proyecto surge de la observación del auge en el uso de *skins* en el mundo de los videojuegos y de cómo aplicar los conocimientos adquiridos a lo largo de la carrera de Ingeniería Informática en un proyecto integral que abarque todas las facetas de la creación de un videojuego.

El desarrollo de videojuegos es un campo en constante crecimiento y evolución, con un impacto social y cultural significativo. Este proyecto aborda la intersección entre la tecnología

y el impacto social, explorando cómo el videojuego personalizado aumenta la satisfacción del jugador al verse representado en el protagonista.

Actualmente, el mercado de videojuegos ofrece editores de creación de personajes con una basta personalización, pero la gran mayoría no son efectivas para representar fielmente al jugador, ya sea en sus rasgos faciales, movimientos, gustos, etc. Eventualmente, algunos videojuegos ponen a disposición del jugador réplicas de personajes famosos con un nivel elevado de similitud para poder ser utilizado como protagonista. La Figura 1 muestra un ejemplo de una *skin* en el popular videojuego Fornite:

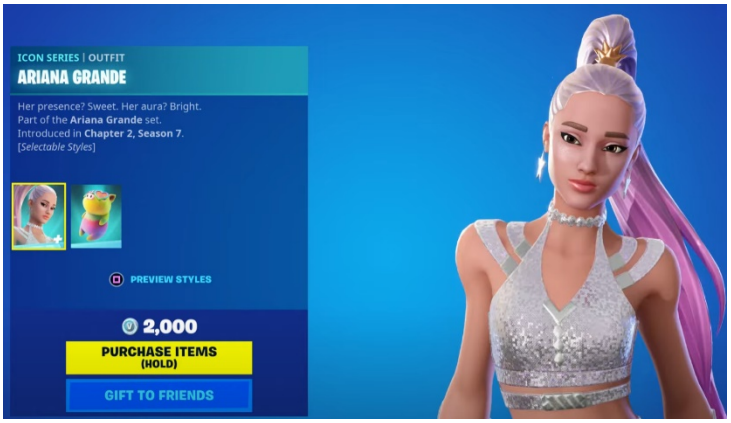


Figura 1: Ariana Grande en Fortnite. Fuente: KingAlexHD.

PWP trata de buscar esa réplica fiel de personajes con un grupo de personas no famosas, amigos del autor, que haga que su experiencia con el videojuego tenga un vínculo más profundo y personal que la que podrían ofrecer meros personajes genéricos o celebridades distantes.

El resultado deseado es tratar de crear un videojuego que no solo sirva como proyecto académico, sino que también ofrezca una experiencia personalizada y emocional para un público específico, pero igual de disfrutable para el público general.

En las subsecciones siguientes se describe la estructura general del videojuego para que el lector se haga una idea del funcionamiento del mismo.

1.2.1. Flujo de pantallas

El juego se dividirá en las siguientes 8 pantallas:

- Menú principal
- Selección de jugadores
- Cuadro del torneo
- Prepartido
- Pantalla de carga
- Partido
- Pospartido
- Entrega de premios

Como se puede observar, se ha prescindido de otras pantallas típicas como pueden ser la del menú de opciones, créditos, tutorial, controles, etc. Esto es debido a que se priorizan otros factores debido a la limitación de tiempo y se establece que la funcionalidad de dichas pantallas puede obviarse o reemplazarse de otras maneras.

Por ejemplo, ciertas opciones se pueden mostrar en el momento indicado como un mensaje modal o aprovechando otras pantallas: en la pantalla de carga mostramos una imagen con la mecánica del partido y cómo funcionan los controles. O en la pantalla de prepartido, los jugadores pueden seleccionar los controles que van a utilizar.

La Figura 2 muestra el flujo de pantallas del juego:

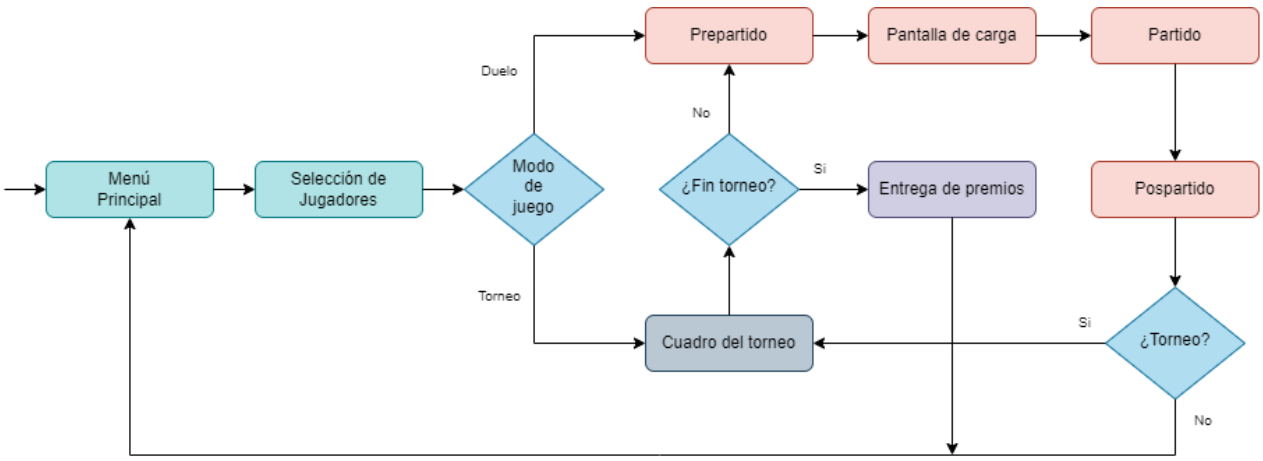


Figura 2: Flujo de pantallas de PWP. Fuente: Elaboración propia.

1.2.2. Pantallas

Describiremos a continuación cada una de las pantallas existentes en el videojuego. En cada una de las pantallas se ha incluido un borrador de cómo se vería y se ha establecido un color de fondo que coincida con el diagrama de flujo de pantallas para un mejor seguimiento.

Menú principal

Está será la primera pantalla del juego donde se verá una imagen animada de fondo y el logo del videojuego en el centro. Justo debajo, la cara del personaje que más torneos lleve ganados, junto con una copa y dicho número. Encima de la copa, la puntuación total de ese personaje. Finalmente, los iconos de unos botones o teclas con las opciones de “Jugar” y “Salir”. La Figura 3 muestra un boceto de la pantalla:

Figura 3: Menú principal. Fuente: Elaboración propia.

Al darle a “Jugar” se mostrará una ventana modal, como la indicada en la figura siguiente, que permita escoger entre las opciones “Torneo” y “Duelo” junto con los botones de “Confirmar” y “Volver”. La Figura 4 muestra un boceto de la pantalla:

Figura 4: Menú principal con opciones. Fuente: Elaboración propia.

Selección de Jugadores

En esta pantalla del juego es donde los usuarios seleccionaran a sus respectivos jugadores. Al iniciar la pantalla, lo primero que se muestra es una ventana modal donde se pregunta por el número de jugadores y la dificultad del juego. Si se ha escogido el modo duelo, el número de jugadores estará marcado a 2 sin opción de cambio. La opción de dificultad tiene 3 niveles (fácil, normal y difícil) donde varía la velocidad de la pelota y la dificultad propia de los rivales (excepto en el modo “Duelo”). La Figura 5 muestra un boceto de la pantalla:

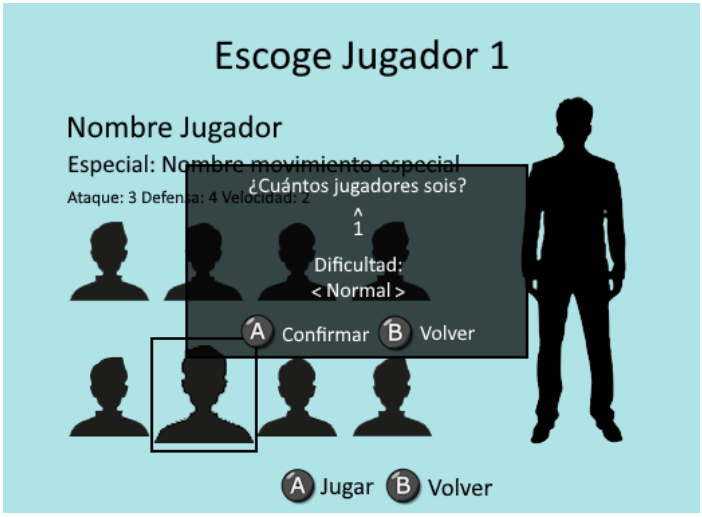


Figura 5: Opciones en selección de jugadores. Fuente: Elaboración propia.

Una vez seleccionadas las opciones, la ventana modal desaparece y nos mostrará las caras de los personajes a escoger junto con su nombre y sus respectivas habilidades. A la derecha

podremos ver el cuerpo completo del personaje que irá cambiando cada vez que cambiemos la selección del jugador. La Figura 6 muestra un boceto de la pantalla:

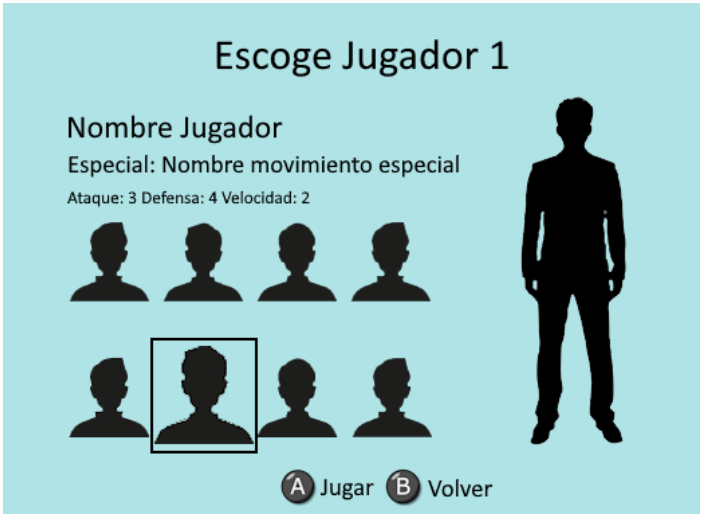


Figura 6: Selección de jugadores. Fuente: Elaboración propia.

Cuadro del torneo

Esta pantalla solamente se mostrará si hemos seleccionado el modo Torneo en el menú principal.

La idea es mostrar un cuadro de torneo de eliminación, en el cual se vaya mostrando la cara de los jugadores que se enfrentan y los resultados que han ido obteniendo durante los partidos. La Figura 7 muestra un boceto de la pantalla:

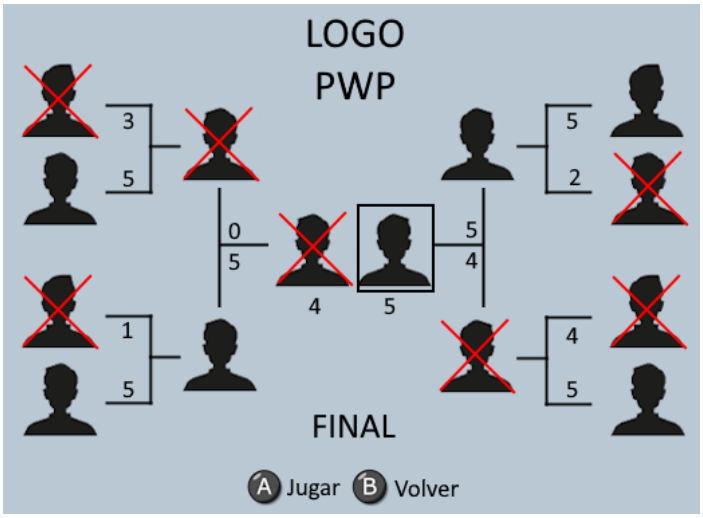


Figura 7: Cuadro de torneo. Fuente: Elaboración propia.

A esta pantalla se llegará cada vez que un partido finalice y estemos en el modo torneo. Cuando la final del torneo se ha jugado, al presionar el botón de “Jugar” nos llevará a la pantalla de “Entrega de premios”.

Prepartido

Tanto si se ha seleccionado el modo de juego duelo o torneo, cada vez que se vaya a iniciar un partido, se visualizará primero esta pantalla. La Figura 8 muestra un boceto de la pantalla:

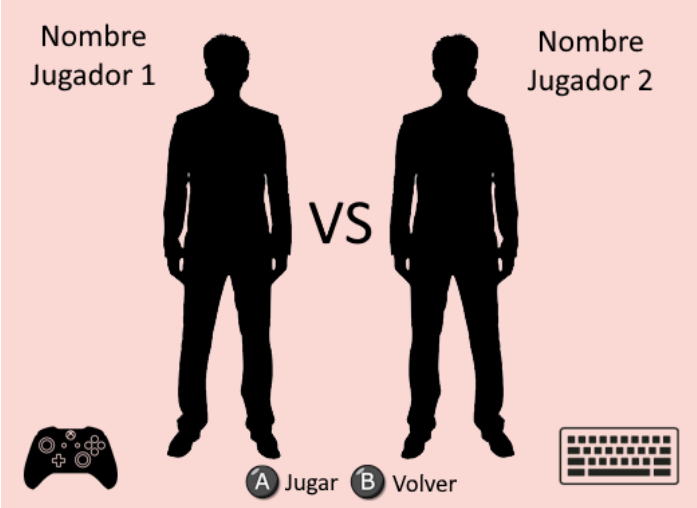


Figura 8: Pantalla de prepartido. Fuente: Elaboración propia.

Como podemos ver en la figura anterior, esta pantalla nos muestra los dos jugadores que se van a enfrentar en el partido junto con la selección del control que quieren manejar (mando y teclado).

Pantalla de carga

En esta pantalla se mostrará la información de los controles seleccionados por los jugadores mientras que se cargan todos los recursos del partido. En el punto 1.2.3 se explican estos controles.

Tal y cómo se ha explicado en el punto 1.2.1, se ha descartado la creación de una pantalla de menú con un tutorial de cómo es la mecánica del partido, así que en esta pantalla de carga

también se mostrarán una serie de consejos con información relativa al partido. La Figura 9 muestra un boceto de la pantalla:

Figura 9: Pantalla de carga. Fuente: Elaboración propia.

Partido

Esta pantalla es donde transcurre la mecánica principal del juego, en el que se enfrentan dos jugadores a ambos lados de un campo. Tal y como veremos en el punto 1.4, la disposición del terreno de juego está inspirada en los videojuegos *Pong* y *Windjammers*. La Figura 10 muestra un boceto de la pantalla:

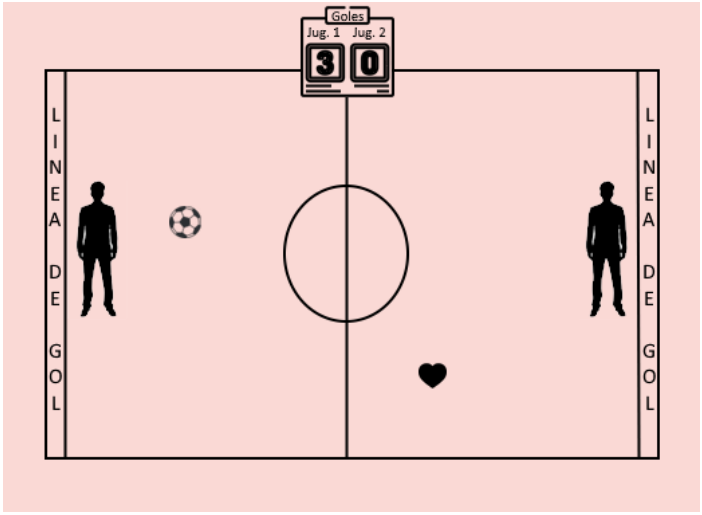


Figura 10: Pantalla de partido. Fuente: Elaboración propia.

En la figura anterior podemos observar ciertos elementos, como el marcador, la pelota, un corazón que representa los *power-ups*, la línea de gol y las líneas que delimitan el terreno de juego.

En el punto 1.2.3 se explica la mecánica propia del partido, ya que existen diferencias con los juegos mencionados, como el uso de movimientos especiales, disparo de objetos o recolección de *power-ups*.

Pospartido

Similar a la pantalla de prepartido, donde se nos muestra a los dos jugadores que se acaban de enfrentar, destacando al jugador vencedor junto con una frase de victoria. La Figura 11 muestra un boceto de la pantalla:

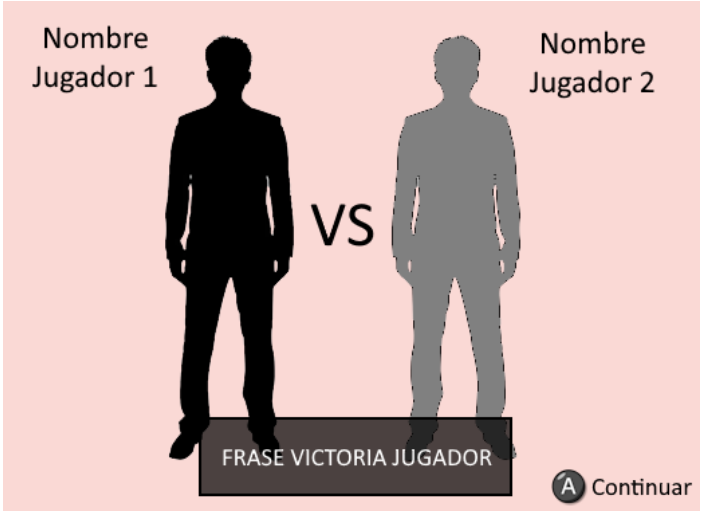


Figura 11: Pantalla de pospartido. Fuente: Elaboración propia.

Entrega de premios

Cuando finaliza un torneo, el jugador tiene la posibilidad de acceder a la pantalla de entrega de premios, donde se verá una escena con los jugadores que han quedado en las 3 primeras posiciones del torneo. La Figura 12 muestra un boceto de la pantalla:

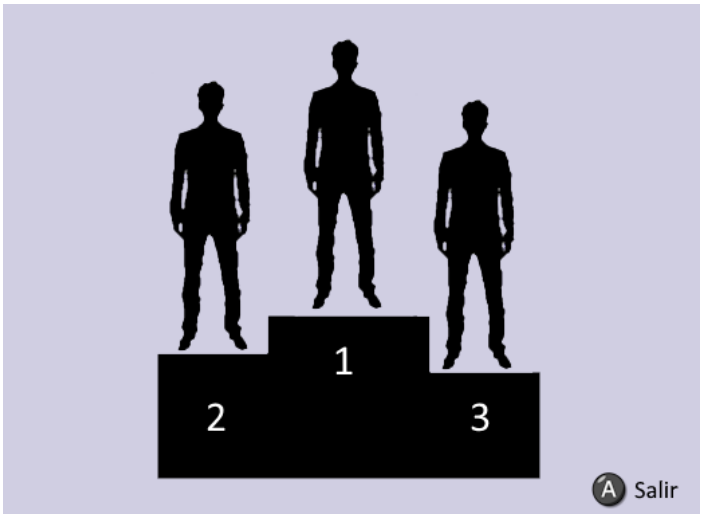


Figura 12: Entrega de premios. Fuente: Elaboración propia.

1.2.3. Mecánica del partido

Tal y como vimos en el apartado anterior, el partido comenzará con ambos jugadores a los lados opuestos del terreno de juego. Cada jugador cuenta con una vida y una energía especial representados en forma de barra que irá reduciéndose por el uso.

Como en el *Pong*, la pelota partirá aleatoriamente desde el centro del campo hacia uno de los lados para que el personaje de ese lado lo golpeé, simplemente chocando con su cuerpo con la pelota.

No hace falta que el jugador presione ningún botón para golpear la pelota: se ejecuta el movimiento de manera automática para que aquellas personas menos acostumbradas a los videojuegos puedan golpear la pelota de una manera más sencilla.

La velocidad de la pelota variará en función de la dificultad seleccionada en el menú principal, siendo la velocidad más lenta en el modo fácil y más rápida en el difícil. En todos los niveles, la velocidad de la pelota aumentará su velocidad gradualmente cada vez que un personaje la golpeé y se reiniciará cuando haya gol.

Antes de golpear la pelota, el jugador tiene la posibilidad de indicar la dirección de golpeo (apuntar) moviendo el *stick* analógico derecho del mando en la dirección deseada.

Una vez que la pelota haya sido golpeada se dirigirá al campo contrario, rebotando en los límites de pista superiores e inferiores. Si la pelota traspasa la línea de gol del jugador, el contrario habrá conseguido un gol.

El jugador puede presionar el botón de disparar, que ejecutará la acción de lanzar un objeto en línea recta al jugador contrario. Esto puede realizarse en cualquier momento, excepto en ocasiones especiales: cuando el partido está parado, cuando se ejecuta un movimiento especial específico, cuando el jugador ya ha lanzado un objeto y aún no ha llegado a su destino, etc.

Si el objeto impactado colisiona contra el jugador contrario, este perderá un porcentaje de vida. Si el jugador pierde toda su vida, el personaje del jugador no se podrá mover hasta que ocurra un gol o recupere vida (*power-ups*). Por otro lado, el jugador que haya conseguido golpear con el objeto al contrario, recibirá un porcentaje de energía especial.

Cada jugador podrá hacer uso del movimiento especial cuando su barra de energía especial esté completa. Los movimientos especial ejecutarán una acción para el personaje que lo desencadene proporcionando una ventaja temporal. Ejemplos de movimientos especiales personalizados serían:

- “Bruno y Martín”: aparecen dos niños gemelos que se colocan en las esquinas de la línea de gol del jugador invocador, evitando que la pelota la traspase en esa zona.
- “Lanzamiento de barril”: el jugador que lo invoque lanza un barril al personaje contrario y, si se produce el impacto, este realizará sus movimientos con los controles invertidos (presionando el botón de arriba moverá a su personaje hacia abajo y viceversa).

Durante el partido irán apareciendo una serie de *power-ups* que, al ser golpeados por la pelota, desencadenarán un efecto en el partido o en el personaje del jugador que lo haya golpeado. Ejemplos de *power-ups* que se podrían implantar son: ganar vida, ganar energía especial, escudo protector, etc.

Cuando uno de los jugadores llegue a anotar 5 goles, el partido finaliza.

1.2.4. Productos finales

El producto final de este trabajo será tanto una copia digital para PC del videojuego, como la copia física, única y limitada (DVD y USB) para cada uno de los protagonistas del videojuego con su personaje en la portada.

Las copias físicas consistirán en la impresión del logo de PWP en el USB y de un diseño con motivos del videojuego para la portada y contraportada de una caja genérica de DVD, así como el propio soporte de DVD. La Figura 13 muestra un boceto del DVD y USB:

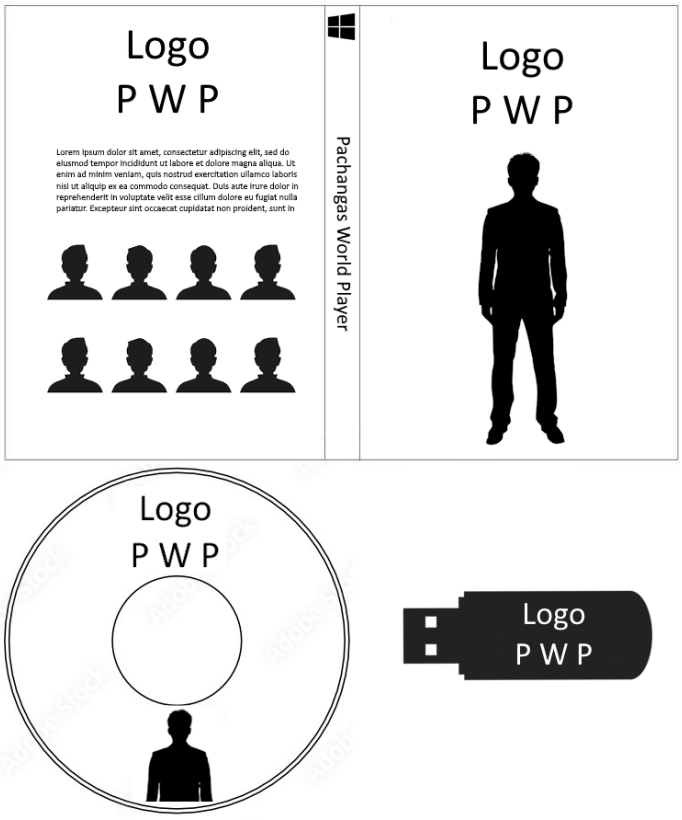


Figura 13: Boceto de diseño del DVD y USB. Fuente: Elaboración propia.

1.3. Objetivos generales

El objetivo general es la producción completa de un videojuego, a través de las distintas fases del ciclo de vida del desarrollo de software, pero adaptado a las distintas técnicas y herramientas específicas del desarrollo de videojuegos.

Como objetivo principal se destaca el desarrollo de la funcionalidad completa del videojuego, ya que debido a su estilo arcade de género deportivo, el ciclo de juego está basado en la repetición de partidos hasta alcanzar el final del mismo.

La rejugabilidad es otro de los objetivos del juego, porque la mecánica principal depende de la habilidad del jugador, pero influenciada por otros elementos de estrategia y de aleatoriedad para generar distintas situaciones a las que el jugador se debe adaptar.

El videojuego tratará de conectar emocionalmente con un público específico, así que otro de los objetivos es la consecución de un alto grado de similitud entre las características de dicho público y los personajes, movimientos, gustos, etc. representados por los componentes audiovisuales del videojuego.

Como el autor es conocedor de la relación entre los integrantes del público anterior y su dedicación a los videojuegos, que abarca desde la nula práctica hasta un nivel de práctica avanzado, otro de los objetivos tiene como fin lograr que cualquier persona disfrute del producto sin importar sus habilidades para con los videojuegos.

Para la consecución de estos objetivos principales se necesitan alcanzar otros objetivos secundarios que profundizan en los conocimientos del autor en los siguientes aspectos:

- Motor de videojuegos Unity.
- Programación en lenguaje C#.
- Edición de imágenes.
- Modelado de personajes.
- Iluminación y VFX.
- Generación de animaciones.

- Control de versiones y herramientas de GitHub.

1.4. Metodología y proceso de trabajo

El proyecto se realizará desde cero, pero partiendo de la inspiración de otros videojuegos: el clásico *Pong* y el *Windjammers*. La mecánica sencilla del *Pong*, en el que dos palas golpean una pelota que rebota en los límites del escenario, es la mecánica más simple de la acción del videojuego, donde únicamente moviendo al jugador sobre su vertical, golpeará la pelota automáticamente al chocar contra ella. La Figura 14 muestra una captura de pantalla del *Pong*:

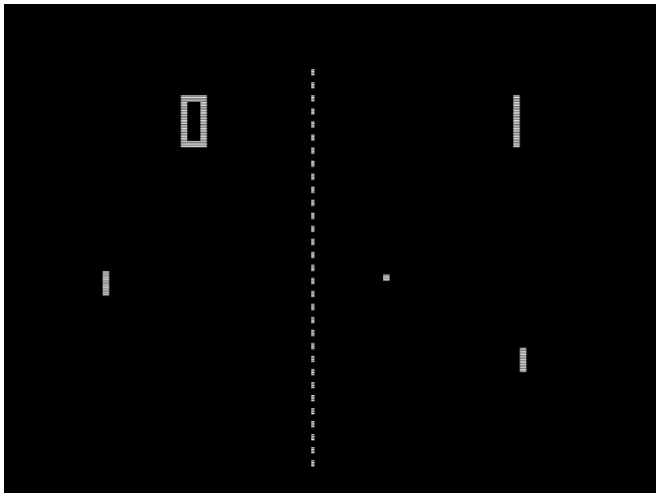


Figura 14: Pantalla del *Pong*. Fuente: Wikipedia.

Por otro lado, la inspiración en *Windjammers* se verá reflejado en el diseño general del videojuego y, especialmente, en la pantalla de enfrentamiento entre jugadores. Cada uno de los personajes contarán con unos movimientos especiales específicos, unas características propias (velocidad, ataque y defensa), etc. logrando que la mecánica de acción pueda ser más compleja a la del *Pong* si el jugador así lo desea. La Figura 15 muestra una captura de pantalla del *Windjammers*:

Figura 15: Pantalla del *Windjammers*. Fuente: Wikipedia.

Partiendo de la inspiración de estos dos juegos, podremos definir la estructura general del videojuego, así como la rejugabilidad y el grado de complejidad que el jugador quiere lograr en cada una de las partidas.

1.4.1. Metodología de desarrollo

Se aplicará una metodología de desarrollo en cascada porque el proyecto se ajusta mejor a ella al contar con las siguientes características:

- Al tener un alcance muy definido, determinado por las PECs, se tiene clara la línea temporal del proyecto antes de iniciarlo.
- Las propias PECs indican los plazos y los tiempos de cada hito a conseguir.
- Se disponen de unos requisitos fijos y una visión clara del producto que se desea crear.
- No se esperan cambios por parte del cliente: el producto está totalmente basado en la visión del autor, por lo que otras metodologías con un enfoque más ágil no aportan un valor significativo.

1.4.2. Proceso de trabajo

El proceso de trabajo viene definido por un fuerte componente creativo a nivel gráfico: mientras que otros proyectos pueden realizarse completamente con *assets* (activos) gratuitos y de libre disposición para el desarrollador, en este proyecto, y debido a la personalización específica de ciertos componentes gráficos, es necesario la elaboración propia de una gran parte de los mismos.

De todas formas, se intentará utilizar *assets* creados por terceros en gran medida debido a las limitaciones temporales del proyecto. Se ha valorado como uno de los puntos fuertes dicha personalización por encima de otros componentes gráficos que no aportan tanto valor al producto final, por lo tanto, el uso de *assets* de terceros en determinados apartados permite un desarrollo más dinámico.

En las siguientes secciones se explicará cómo será el proceso de trabajo de los dos bloques principales de desarrollo: la creación de los personajes y el uso de Unity. La idea general es crear rápidamente modelos 3D de cada jugador con la herramienta Daz Studio de la empresa Daz 3D y hacer *renders* (representaciones gráficas) en 2D para utilizarlas en Unity.

La ventaja de utilizar *renders* en 2D en este videojuego específico en lugar de utilizar directamente los modelos 3D son múltiples:

- Como solamente se tiene una cámara fija en el videojuego (ver Figura 15) no son necesarias las 3D para cubrir distintos ángulos o movimientos de la cámara, lo que simplifica los procesos de desarrollo al trabajar con una dimensión menos.
- El uso de recursos es menor al no tener que hacer uso de las 3D, ya que todas las operaciones matemáticas asociadas son prescindibles (físicas de colisiones, iluminación, sombras, etc.).
- Las pequeñas modificaciones correctivas que haya que realizar a los gráficos, son más rápidas con editores 2D que realizar nuevamente el proceso con el modelo 3D.
- Como se verá más adelante en el detalle del presupuesto, para el uso de los *assets* de Daz3D no es necesaria una licencia interactiva, lo que implica una reducción de costes.

Creación de personajes

PWP tiene unos personajes que serán la réplica digital en el videojuego de los amigos del autor. Para lograrlo, se utilizará la aplicación Daz Studio junto con la herramienta Face Transfer Unlimited y distintos *assets* para la ropa y complementos.

Daz Studio permite generar rápidamente humanoides en 3D, así como crear poses y animaciones. Y con Face Transfer Unlimited, podemos utilizar una fotografía de la cara de

una persona para transferirla al modelo de Daz Studio. La Figura 16 muestra una captura de pantalla del Daz Studio:

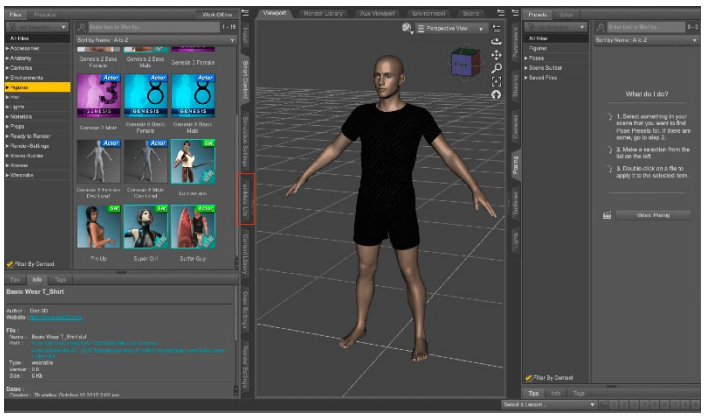


Figura 16: Modelo 3D en Daz Studio. Fuente: Styly.cc

Aunque Daz Studio tiene herramientas para generar el *rigging* del modelo 3D (controles que actúan como su esqueleto), se utilizará la web de Adobe Mixamo que nos genera este *rigging* de manera automática y tiene una biblioteca de animaciones para incorporar directamente a los modelos. La Figura 17 muestra una captura de pantalla de Adobe Mixamo:

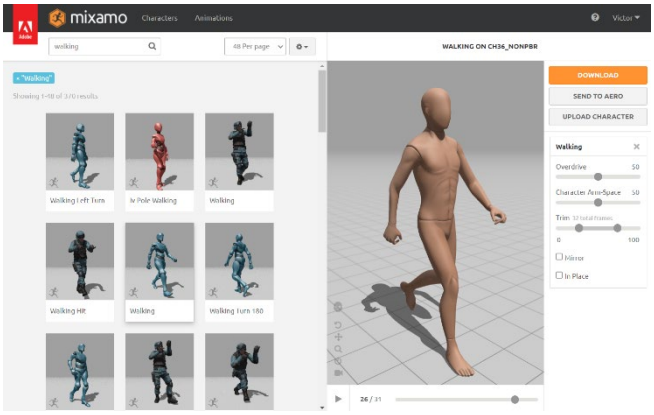


Figura 17: Modelo 3D caminando en Mixamo. Fuente: Mixamo.

El siguiente paso es el uso del software Blender para realizar las modificaciones necesarias en las animaciones generadas y se realizarán los *renders* en este mismo programa. La Figura 18 muestra una captura de pantalla del Blender:

Figura 18: Dina (The last of us) en Blender. Fuente: Devianart.

Finalmente, las imágenes generadas se importarán a Unity, que cuenta con las herramientas necesarias para tratarlas y manejarlas.

Unity

La idea general es tener una escena de Unity por cada pantalla del videojuego y unos scripts asociados a las escenas para controlar su funcionamiento. Unity permite distintos lenguajes para realizar la parte de programación asociada, así que se hará uso del lenguaje C# con el entorno de desarrollo de Visual Studio. Dicho entorno tendrá asociado el proyecto alojado en Github para poder realizar la subida de los cambios a la plataforma.

Por otro lado, los gráficos generados en el apartado anterior serán manejados a través de *Sprites* (mapas de bits) y *Animator Controllers* que nos permitirán manejar las distintas animaciones. La Figura 19 muestra una captura de pantalla del *Animator Controller* en Unity:

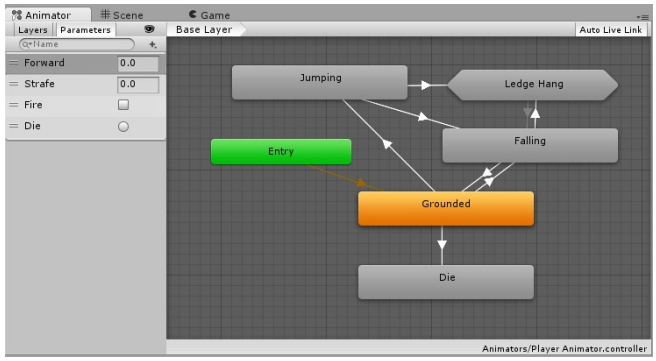


Figura 19: *Animator Controller* en Unity. Fuente: Unity.

Los cambios que se hagan en Unity estarán asociados a un proyecto alojado en GitHub utilizando el sistema de control de versiones Git, permitiendo tener un histórico de cambios del proyecto.

También se hará uso de otras herramientas propias de la plataforma, como *Issues* y *Projects*, que permiten llevar un control de las tareas e incidencias y ver los cambios realizados en el proyecto asociadas a ellas. La Figura 20 muestra una captura de los *Projects* en Github:

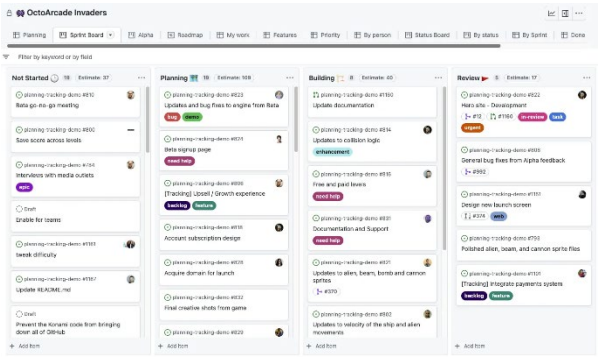
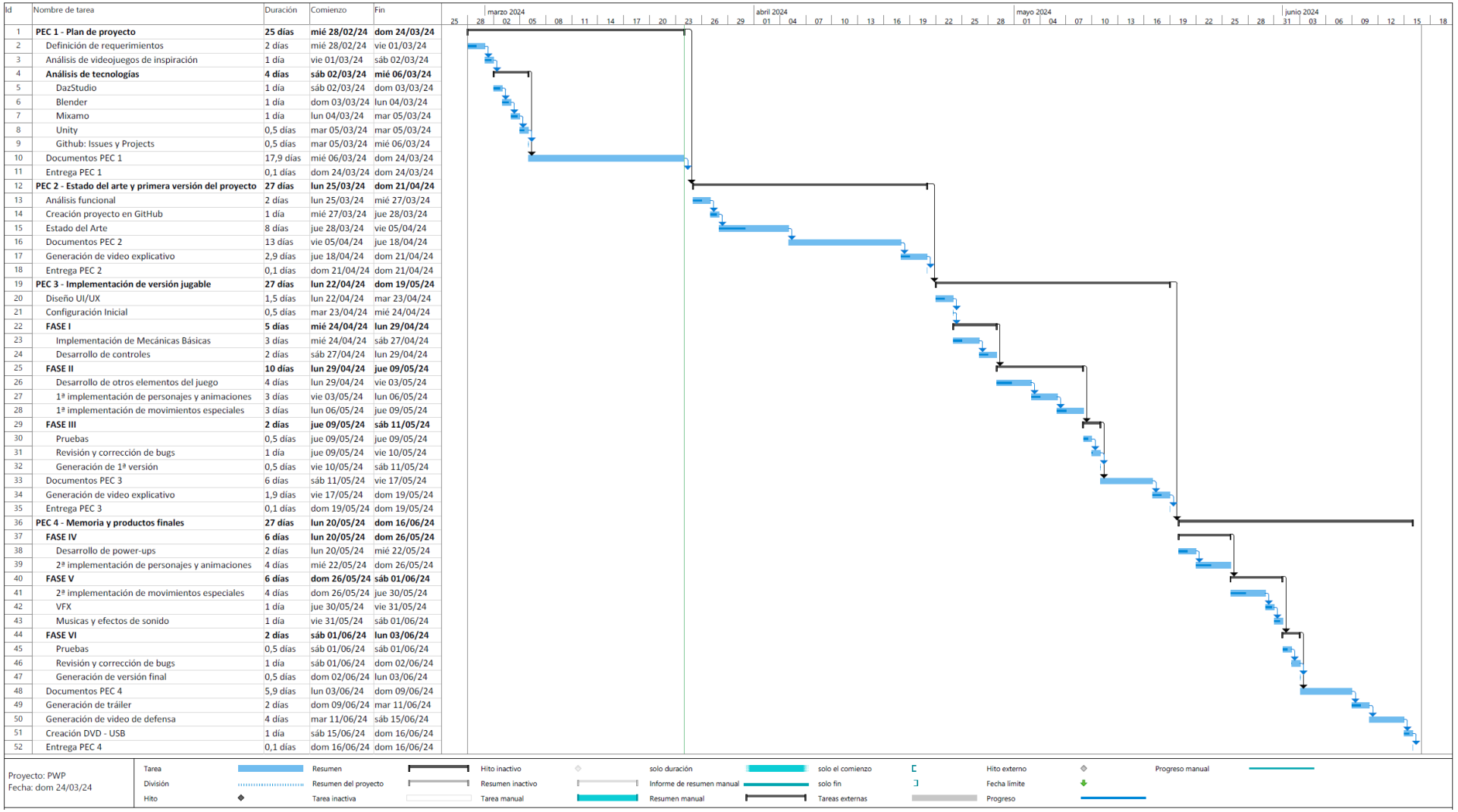


Figura 20: *Projects* en Github. Fuente: Github.

1.5. Planificación

En el siguiente diagrama de Gantt puede verse la planificación de tareas, segmentadas en las distintas PECs, para lograr la consecución del proyecto. Se ha establecido un calendario no laboral, con disposición para trabajar en el proyecto durante todos los días hasta su finalización.

Como puede apreciarse, cada bloque de PECs contiene distintas secciones y subsecciones que se explicarán a continuación. Destacar que las tareas específicas del desarrollo del producto se han agrupado por fases que nos darán una idea del estado del videojuego dentro de cada PEC. La Figura 21 muestra el diagrama de Gantt del proyecto:



Id	Nombre de tarea	Duración	Comienzo	Fin
1	**PEC 1 - Plan de proyecto**	**25 días**	**mié 28/02/24**	**dom 24/03/24**
2	Definición de requerimientos	2 días	mié 28/02/24	vie 01/03/24
3	Análisis de videojuegos de inspiración	1 día	vie 01/03/24	sáb 02/03/24
4	**Análisis de tecnologías**	**4 días**	**sáb 02/03/24**	**mié 06/03/24**
5	DazStudio	1 día	sáb 02/03/24	dom 03/03/24
6	Blender	1 día	dom 03/03/24	lun 04/03/24
7	Mixamo	1 día	lun 04/03/24	mar 05/03/24
8	Unity	0,5 días	mar 05/03/24	mar 05/03/24
9	Github: Issues y Projects	0,5 días	mar 05/03/24	mié 06/03/24
10	Documentos PEC 1	17,9 días	mié 06/03/24	dom 24/03/24
11	Entrega PEC 1	0,1 días	dom 24/03/24	dom 24/03/24
12	**PEC 2 - Estado del arte y primera versión del proyecto**	**27 días**	**lun 25/03/24**	**dom 21/04/24**
13	Análisis funcional	2 días	lun 25/03/24	mié 27/03/24
14	Creación proyecto en GitHub	1 día	mié 27/03/24	jue 28/03/24
15	Estado del Arte	8 días	jue 28/03/24	vie 05/04/24
16	Documentos PEC 2	13 días	vie 05/04/24	jue 18/04/24
17	Generación de video explicativo	2,9 días	jue 18/04/24	dom 21/04/24
18	Entrega PEC 2	0,1 días	dom 21/04/24	dom 21/04/24
19	**PEC 3 - Implementación de versión jugable**	**27 días**	**lun 22/04/24**	**dom 19/05/24**
20	Diseño UI/UX	1,5 días	lun 22/04/24	mar 23/04/24
21	Configuración Inicial	0,5 días	mar 23/04/24	mié 24/04/24
22	**FASE I**	**5 días**	**mié 24/04/24**	**lun 29/04/24**
23	Implementación de Mecánicas Básicas	3 días	mié 24/04/24	sáb 27/04/24
24	Desarrollo de controles	2 días	sáb 27/04/24	lun 29/04/24
25	**FASE II**	**10 días**	**lun 29/04/24**	**jue 09/05/24**
26	Desarrollo de otros elementos del juego	4 días	lun 29/04/24	vie 03/05/24
27	1ª implementación de personajes y animaciones	3 días	vie 03/05/24	lun 06/05/24
28	1ª implementación de movimientos especiales	3 días	lun 06/05/24	jue 09/05/24
29	**FASE III**	**2 días**	**jue 09/05/24**	**sáb 11/05/24**
30	Pruebas	0,5 días	jue 09/05/24	jue 09/05/24
31	Revisión y corrección de bugs	1 día	jue 09/05/24	vie 10/05/24
32	Generación de 1ª versión	0,5 días	vie 10/05/24	sáb 11/05/24
33	Documentos PEC 3	6 días	sáb 11/05/24	vie 17/05/24
34	Generación de video explicativo	1,9 días	vie 17/05/24	dom 19/05/24
35	Entrega PEC 3	0,1 días	dom 19/05/24	dom 19/05/24
36	**PEC 4 - Memoria y productos finales**	**27 días**	**lun 20/05/24**	**dom 16/06/24**
37	**FASE IV**	**6 días**	**lun 20/05/24**	**dom 26/05/24**
38	Desarrollo de power-ups	2 días	lun 20/05/24	mié 22/05/24
39	2ª implementación de personajes y animaciones	4 días	mié 22/05/24	dom 26/05/24
40	**FASE V**	**6 días**	**dom 26/05/24**	**sáb 01/06/24**
41	2ª implementación de movimientos especiales	4 días	dom 26/05/24	jue 30/05/24
42	VFX	1 día	jue 30/05/24	vie 31/05/24
43	Musicas y efectos de sonido	1 día	vie 31/05/24	sáb 01/06/24
44	**FASE VI**	**2 días**	**sáb 01/06/24**	**lun 03/06/24**
45	Pruebas	0,5 días	sáb 01/06/24	sáb 01/06/24
46	Revisión y corrección de bugs	1 día	sáb 01/06/24	dom 02/06/24
47	Generación de versión final	0,5 días	dom 02/06/24	lun 03/06/24
48	Documentos PEC 4	5,9 días	lun 03/06/24	dom 09/06/24
49	Generación de tráiler	2 días	dom 09/06/24	mar 11/06/24
50	Generación de video de defensa	4 días	mar 11/06/24	sáb 15/06/24
51	Creación DVD - USB	1 día	sáb 15/06/24	dom 16/06/24
52	Entrega PEC 4	0,1 días	dom 16/06/24	dom 16/06/24

Figura 21: Diagrama de Gantt. Fuente: Elaboración propia.

1.5.1. PEC 1 - Plan de proyecto

Definición de requerimientos: Se establecen las características y requisitos necesarios que van a formar parte del producto a desarrollar teniendo en mente el alcance del proyecto.

Análisis de videojuegos de inspiración: Como ya se ha comentado anteriormente, el juego está inspirado en el *Pong* y el *Windjammers*, por lo tanto, se debe de profundizar en dichos juegos y extraer la información necesaria para aplicarla al proyecto.

Análisis de tecnologías: Comprobar que las distintas tecnologías que se vayan a utilizar sean las correctas para el desarrollo del proyecto.

Documentos PEC 1: Redacción de los documentos obligatorios para la entrega de la PEC.

Entrega PEC 1.

1.5.2. PEC 2 - Estado del arte y primera versión del proyecto

Análisis funcional: Una vez se ha definido el plan de proyecto y se tiene una visión concreta del producto, es necesario detallar lo máximo posible cómo tiene que funcionar el producto y cuáles son los riesgos asociados.

Creación proyecto en GitHub: Una de las tecnologías necesarias de esta PEC es el uso de Github, por lo tanto, en este apartado realizaremos la creación del proyecto en dicha plataforma y la configuración de Git en Visual Studio enlazada a Github para poder subir los cambios del proyecto.

Estado del Arte: Se debe de realizar un profundo análisis del Estado del Arte del proyecto, recabando toda la información necesaria a través de Internet u otras fuentes.

Documentos PEC 2: Redacción de los documentos obligatorios para la entrega de la PEC.

Generación de video explicativo: Grabación de un vídeo corto con la explicación del funcionamiento del producto y los aspectos más destacados.

Entrega PEC 2.

1.5.3. PEC 3 - Implementación de versión jugable

Diseño UI/UX: En esta PEC se detallarán las características de diseño, por lo tanto, aquí podemos englobar las tareas de analizar y documentarse con respecto a la experiencia y la interfaz de usuario.

Configuración Inicial: Antes de comenzar el desarrollo del producto, se debe configurar aquellos aspectos del entorno de trabajo (p. ej. .gitignore, readme, etc.) que establecerán las bases iniciales para el desarrollo.

Fase I: En esta primera fase se comienza con la generación del código y los elementos básicos para crear la estructura del videojuego, como la implementación de mecánicas básicas y el desarrollo de los controles.

Fase II: Se continúa avanzando en el producto, incorporando otros elementos propios del videojuego, así como la primera implementación de los personajes, con sus animaciones y movimientos especiales correspondientes.

Fase III: En esta fase se realizan las pruebas pertinentes, se corrigen errores y se genera una primera versión del videojuego.

Documentos PEC 3: Redacción de los documentos obligatorios para la entrega de la PEC.

Generación de video explicativo: Grabación de un vídeo corto detallando el funcionamiento del juego.

Entrega PEC 3.

1.5.4. PEC 4 - Memoria y productos finales

Fase IV: Esta fase se centra en la implementación de los *power-ups* y en la segunda iteración del resto de personajes y sus animaciones correspondientes.

Fase V: Se realiza la segunda implementación de los movimientos especiales correspondientes a los personajes de la fase anterior. También se incorporan los efectos visuales y de sonido en las distintas escenas del videojuego.

Fase VI: En esta fase se realizan las pruebas pertinentes, se corrigen errores y se genera la versión del videojuego.

Documentos PEC 4: Redacción de los documentos obligatorios para la entrega de la PEC.

Generación de tráiler: Grabación de un vídeo corto a modo de anuncio publicitario o pitch para un posible *publisher*.

Generación de vídeo de defensa: Grabación de un vídeo a través de una presentación oral guiada con transparencias y *gameplay* del videojuego.

Entrega PEC 4.

1.6. Presupuesto

A continuación, se muestra un posible presupuesto con los costos de desarrollo del videojuego, obviando los costos de operación, *Marketing* y de Distribución y Comercialización ya que el producto final es un regalo a un público específico y no ha lugar los costos indicados.

Debido a la particularidad del producto, que es un desarrollo personal del autor con conocimientos en distintas áreas, en lugar de mostrar el presupuesto de trabajo de una única persona, se considera interesante mostrarlo cómo un trabajo de 3 personas (equipo de proyecto) con conocimientos específicos.

Vamos a suponer que el autor es la persona que quiere realizar el regalo a sus amistades, pero no quiere desarrollarlo él mismo. Por lo tanto, será la persona que encargue el trabajo, se conocerá en adelante como el cliente o patrocinador del proyecto, y va a ser incluido en las distintas fases del proyecto, pero cuyas horas de trabajo no se estiman en el presupuesto debido a que el cliente no es un recurso pagado del equipo de proyecto.

Por lo tanto, podemos dividir el equipo humano en:

- Cliente
- Diseñador de videojuegos

- Programador
- Artista

Por supuesto, el proyecto podría contar con muchas más personas que se encargasen de roles específicos del desarrollo de un videojuego, como podría ser un artista de UI, artista de VFX, un animador 3D, diseñadores de sonido, etc.

Pero como ocurre en muchos juegos *indies* o de bajo presupuesto, cada uno de estos componentes del equipo tendrá conocimientos en otras ramas técnicas de su área y colaborando entre ellos podrán cubrir los roles específicos.

1.6.1. Costos por miembros del equipo

Como la carga de trabajo puede variar para los distintos componentes del equipo de proyecto, se considera interesante plantear el presupuesto con un modelo de bolsa de horas, estableciendo una tarifa de horas para cada rol y estimando una jornada laboral de 8 horas.

Podemos utilizar el diagrama de Gantt visto en el anterior punto como la planificación del proyecto real, simplemente obviando las tareas de “Documentos PEC” y “Entrega PEC”. En otros proyectos del desarrollo del software podrían sustituirse por otras tareas de documentación y análisis, pero en el caso del producto actual no son necesarias.

Por lo tanto, a partir de dicho diagrama y con los miembros definidos del equipo humano, podemos concretar en qué tareas participaran:

- **Cliente**: El cliente participará en las reuniones de definición de requerimientos, dará su opinión en análisis funcionales, diseños de UI/UX y será una de las personas que haga pruebas de las versiones del producto.
- **Diseñador de videojuegos**: El diseñador de videojuegos participa en todos los bloques de la planificación, en algunos momentos no tan activamente como otros miembros, pero por su visión general del videojuego es necesaria su alta participación.
- **Programador**: Su trabajo se centra en las fases en las que se desarrolla directamente el producto, siendo su participación máxima en dicho momento.
- **Artista**: Como en el caso del programador, desarrollará sus principales tareas en las fases de desarrollo del producto.

La Figura 22 muestra el desglose de las tareas por cada miembro del equipo:

Id	Nombre del recurso	Trabajo real	Detalles	tri 2, 2024 feb	mar	abr	may	jun	tri 3, 2024 jul	ago
	Sin asignar	**0 horas**	Trab. real							
	Documentos PEC 1	*0 horas*	Trab. real							
	Entrega PEC 1	*0 horas*	Trab. real							
	Documentos PEC 2	*0 horas*	Trab. real							
	Entrega PEC 2	*0 horas*	Trab. real							
	Documentos PEC 3	*0 horas*	Trab. real							
	Entrega PEC 3	*0 horas*	Trab. real							
	Documentos PEC 4	*0 horas*	Trab. real							
	Generación de video de defens	*0 horas*	Trab. real							
	Entrega PEC 4	*0 horas*	Trab. real							
1	**Cliente**	**52 horas**	Trab. real	16h	16h	12h	4h	4h		
	Definición de requerimientos	*16 horas*	Trab. real	16h						
	Análisis funcional	*16 horas*	Trab. real		16h					
	Diseño UI/UX	*12 horas*	Trab. real			12h				
	Pruebas	*4 horas*	Trab. real				4h			
	Pruebas	*4 horas*	Trab. real					4h		
2	**Diseñador de videojuegos**	**226,4 horas**	Trab. real	16h	128h	35,2h	27,2h	20h		
	Definición de requerimientos	*16 horas*	Trab. real	16h						
	Análisis de videojuegos de insp	*8 horas*	Trab. real		8h					
	DazStudio	*8 horas*	Trab. real		8h					
	Blender	*8 horas*	Trab. real		8h					
	Mixamo	*8 horas*	Trab. real		8h					
	Unity	*4 horas*	Trab. real		4h					
	Github: Issues y Projects	*4 horas*	Trab. real		4h					
	Análisis funcional	*16 horas*	Trab. real		16h					
	Creación proyecto en GitHub	*8 horas*	Trab. real		8h					
	Estado del Arte	*64 horas*	Trab. real		64h					
	Generación de video explicativ	*23,2 horas*	Trab. real			23,2h				
	Diseño UI/UX	*12 horas*	Trab. real			12h				
	Pruebas	*4 horas*	Trab. real				4h			
	Generación de video explicativ	*15,2 horas*	Trab. real				15,2h			
	Musicas y efectos de sonido	*8 horas*	Trab. real				8h			
	Pruebas	*4 horas*	Trab. real					4h		
	Generación de tráiler	*16 horas*	Trab. real					16h		
3	**Programador**	**244 horas**	Trab. real			76h	156h	12h		
	Configuración Inicial	*4 horas*	Trab. real			4h				
	Implementación de Mecánicas	*24 horas*	Trab. real			24h				
	Desarrollo de controles	*16 horas*	Trab. real			16h				
	Desarrollo de otros elementos	*32 horas*	Trab. real			32h				
	1ª implementación de persona	*24 horas*	Trab. real				24h			
	1ª implementación de movimie	*24 horas*	Trab. real				24h			
	Revisión y corrección de bugs	*8 horas*	Trab. real				8h			
	Generación de 1ª versión	*4 horas*	Trab. real				4h			
	Desarrollo de power-ups	*16 horas*	Trab. real				16h			
	2ª implementación de persona	*32 horas*	Trab. real				32h			
	2ª implementación de movimie	*32 horas*	Trab. real				32h			
	VFX	*8 horas*	Trab. real				8h			
	Musicas y efectos de sonido	*8 horas*	Trab. real				8h			
	Revisión y corrección de bugs	*8 horas*	Trab. real					8h		
	Generación de versión final	*4 horas*	Trab. real					4h		
4	**Artista**	**235,2 horas**	Trab. real			44h	159,2h	32h		
	Diseño UI/UX	*12 horas*	Trab. real			12h				
	Desarrollo de otros elementos	*32 horas*	Trab. real			32h				
	1ª implementación de persona	*24 horas*	Trab. real				24h			
	1ª implementación de movimie	*24 horas*	Trab. real				24h			
	Revisión y corrección de bugs	*8 horas*	Trab. real				8h			
	Generación de video explicativ	*15,2 horas*	Trab. real				15,2h			
	Desarrollo de power-ups	*16 horas*	Trab. real				16h			
	2ª implementación de persona	*32 horas*	Trab. real				32h			
	2ª implementación de movimie	*32 horas*	Trab. real				32h			
	VFX	*8 horas*	Trab. real				8h			
	Revisión y corrección de bugs	*8 horas*	Trab. real					8h		
	Generación de tráiler	*16 horas*	Trab. real					16h		
	Creación DVD - USB	*8 horas*	Trab. real					8h		

Figura 22: Trabajos por horas por los miembros del equipo. Fuente: Elaboración propia.

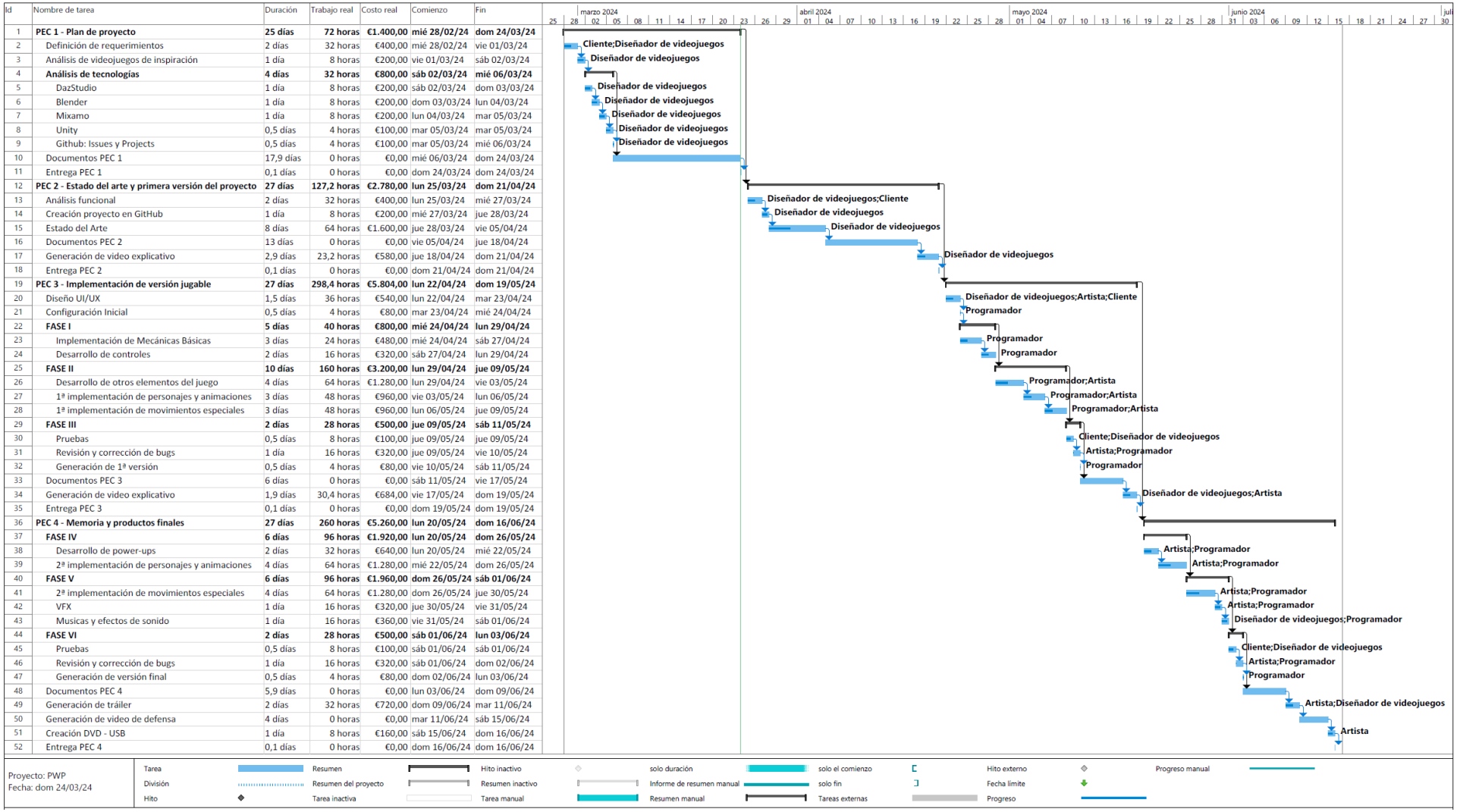
Para calcular el coste por hora de cada componente del equipo, se van a tomar los valores medios para un diseñador videojuegos senior [1], programador junior [2] y artista junior [3].

Por lo tanto, tenemos la siguiente tabla de coste por hora por cada miembro del equipo, tal y como muestra la Figura 23:

	Nombre del recurso	Tipo	Etiqueta de	Iniciales	Grupo	Capacidad	Tasa
1	Cliente	Trabajo		C		100%	0,00 €/hora
2	Diseñador de videoj	Trabajo		D		100%	25,00 €/hora
3	Programador	Trabajo		P		100%	20,00 €/hora
4	Artista	Trabajo		A		100%	20,00 €/hora

Figura 23: Coste por hora de los miembros del equipo. Fuente: Elaboración propia.

Finalmente, podemos ver el diagrama de Gantt con las horas realizadas por cada miembro del proyecto y el coste real de las tareas en la Figura 24:



Id	Nombre de tarea	Duración	Trabajo real	Costo real	Comienzo	Fin
1	**PEC 1 - Plan de proyecto**	**25 días**	**72 horas**	**€1.400,00**	**mié 28/02/24**	**dom 24/03/24**
2	Definición de requerimientos	2 días	32 horas	€400,00	mié 28/02/24	vie 01/03/24
3	Análisis de videojuegos de inspiración	1 día	8 horas	€200,00	vie 01/03/24	sáb 02/03/24
4	**Análisis de tecnologías**	**4 días**	**32 horas**	**€800,00**	**sáb 02/03/24**	**mié 06/03/24**
5	DazStudio	1 día	8 horas	€200,00	sáb 02/03/24	dom 03/03/24
6	Blender	1 día	8 horas	€200,00	dom 03/03/24	lun 04/03/24
7	Mixamo	1 día	8 horas	€200,00	lun 04/03/24	mar 05/03/24
8	Unity	0,5 días	4 horas	€100,00	mar 05/03/24	mar 05/03/24
9	Github: Issues y Projects	0,5 días	4 horas	€100,00	mar 05/03/24	mié 06/03/24
10	Documentos PEC 1	17,9 días	0 horas	€0,00	mié 06/03/24	dom 24/03/24
11	Entrega PEC 1	0,1 días	0 horas	€0,00	dom 24/03/24	dom 24/03/24
12	**PEC 2 - Estado del arte y primera versión del proyecto**	**27 días**	**127,2 horas**	**€2.780,00**	**lun 25/03/24**	**dom 21/04/24**
13	Análisis funcional	2 días	32 horas	€400,00	lun 25/03/24	mié 27/03/24
14	Creación proyecto en GitHub	1 día	8 horas	€200,00	mié 27/03/24	jue 28/03/24
15	Estado del Arte	8 días	64 horas	€1.600,00	jue 28/03/24	vie 05/04/24
16	Documentos PEC 2	13 días	0 horas	€0,00	vie 05/04/24	jue 18/04/24
17	Generación de video explicativo	2,9 días	23,2 horas	€580,00	jue 18/04/24	dom 21/04/24
18	Entrega PEC 2	0,1 días	0 horas	€0,00	dom 21/04/24	dom 21/04/24
19	**PEC 3 - Implementación de versión jugable**	**27 días**	**298,4 horas**	**€5.804,00**	**lun 22/04/24**	**dom 19/05/24**
20	Diseño UI/UX	1,5 días	36 horas	€540,00	lun 22/04/24	mar 23/04/24
21	Configuración Inicial	0,5 días	4 horas	€80,00	mar 23/04/24	mié 24/04/24
22	**FASE I**	**5 días**	**40 horas**	**€800,00**	**mié 24/04/24**	**lun 29/04/24**
23	Implementación de Mecánicas Básicas	3 días	24 horas	€480,00	mié 24/04/24	sáb 27/04/24
24	Desarrollo de controles	2 días	16 horas	€320,00	sáb 27/04/24	lun 29/04/24
25	**FASE II**	**10 días**	**160 horas**	**€3.200,00**	**lun 29/04/24**	**jue 09/05/24**
26	Desarrollo de otros elementos del juego	4 días	64 horas	€1.280,00	lun 29/04/24	vie 03/05/24
27	1ª implementación de personajes y animaciones	3 días	48 horas	€960,00	vie 03/05/24	lun 06/05/24
28	1ª implementación de movimientos especiales	3 días	48 horas	€960,00	lun 06/05/24	jue 09/05/24
29	**FASE III**	**2 días**	**28 horas**	**€500,00**	**jue 09/05/24**	**sáb 11/05/24**
30	Pruebas	0,5 días	8 horas	€100,00	jue 09/05/24	jue 09/05/24
31	Revisión y corrección de bugs	1 día	16 horas	€320,00	jue 09/05/24	vie 10/05/24
32	Generación de 1ª versión	0,5 días	4 horas	€80,00	vie 10/05/24	sáb 11/05/24
33	Documentos PEC 3	6 días	0 horas	€0,00	sáb 11/05/24	vie 17/05/24
34	Generación de video explicativo	1,9 días	30,4 horas	€684,00	vie 17/05/24	dom 19/05/24
35	Entrega PEC 3	0,1 días	0 horas	€0,00	dom 19/05/24	dom 19/05/24
36	**PEC 4 - Memoria y productos finales**	**27 días**	**260 horas**	**€5.260,00**	**lun 20/05/24**	**dom 16/06/24**
37	**FASE IV**	**6 días**	**96 horas**	**€1.920,00**	**lun 20/05/24**	**dom 26/05/24**
38	Desarrollo de power-ups	2 días	32 horas	€640,00	lun 20/05/24	mié 22/05/24
39	2ª implementación de personajes y animaciones	4 días	64 horas	€1.280,00	mié 22/05/24	dom 26/05/24
40	**FASE V**	**6 días**	**96 horas**	**€1.960,00**	**dom 26/05/24**	**sáb 01/06/24**
41	2ª implementación de movimientos especiales	4 días	64 horas	€1.280,00	dom 26/05/24	jue 30/05/24
42	VFX	1 día	16 horas	€320,00	jue 30/05/24	vie 31/05/24
43	Musicas y efectos de sonido	1 día	16 horas	€360,00	vie 31/05/24	sáb 01/06/24
44	**FASE VI**	**2 días**	**28 horas**	**€500,00**	**sáb 01/06/24**	**lun 03/06/24**
45	Pruebas	0,5 días	8 horas	€100,00	sáb 01/06/24	sáb 01/06/24
46	Revisión y corrección de bugs	1 día	16 horas	€320,00	sáb 01/06/24	dom 02/06/24
47	Generación de versión final	0,5 días	4 horas	€80,00	dom 02/06/24	lun 03/06/24
48	Documentos PEC 4	5,9 días	0 horas	€0,00	lun 03/06/24	dom 09/06/24
49	Generación de tráiler	2 días	32 horas	€720,00	dom 09/06/24	mar 11/06/24
50	Generación de video de defensa	4 días	0 horas	€0,00	mar 11/06/24	sáb 15/06/24
51	Creación DVD - USB	1 día	8 horas	€160,00	sáb 15/06/24	dom 16/06/24
52	Entrega PEC 4	0,1 días	0 horas	€0,00	dom 16/06/24	dom 16/06/24

Figura 24: Diagrama de Gantt por costes. Fuente: Elaboración propia.

1.6.2. Costos por hardware y software

El equipamiento técnico consistirá en equipos informáticos para cada uno de los componentes del equipo que, debido a la duración tan corta del proyecto, es interesante realizar un alquiler de los equipos informáticos [4] en lugar de la compra de los mismos.

También se incluyen en el presupuesto los gastos de las licencias del software, algunas de las cuales tendrán un coste gratuito ya que el producto no va a tener comercialización ni unas ganancias derivadas.

El software gratuito de Daz Studio [5] de la empresa Daz 3D es la herramienta que se utilizará para generar los modelos 3D de los personajes, así como otros *props* (atrezo) presentes en el videojuego.

Como los gráficos del proyecto son renderizaciones en 2D, no es necesaria una licencia interactiva a mayores de la licencia estándar para el uso de los *assets* de Daz 3D [6].

En el presupuesto está indicada una estimación de los *assets* para la ropa, complementos y distintas *props* del videojuego, teniendo una media de 3 *assets* por personaje (algunos *assets* pueden reutilizarse) y 10 *props* en total para las distintas escenas, movimientos especiales, etc.

Se estima el coste de media por *asset* en 10€, ya que se va a suponer que el equipo de trabajo es nuevo y no tienen en posesión ninguna licencia de los *asset*s que vayan a utilizar ni se van a adquirir mediante *packs* o promociones específicas.

Por otro lado, es necesaria una licencia de uso del plugin Face Transfer Unlimited [7] en Daz Studio, que logra transferir la cara de una fotografía al modelo 3D y cuyo coste en la actualidad es de 49,95€.

En el caso de Unity, como el producto es para un uso no comercial, puede utilizarse el plan Personal gratuito [8].

Blender [9] está publicado bajo la GNU General Public License, por lo tanto, su uso y distribución es gratuita.

El uso de las tecnologías de la plataforma de Adobe Mixamo [10] están disponibles de forma gratuita, sin licencias ni derechos de autor, para uso comercial o no comercial ilimitado.

La plataforma de desarrollo de Github [11] tiene disponibles distintos planes, pero el plan gratuito cubre todas las necesidades del proyecto.

1.6.3. Resumen costos

Finalmente, podemos ver los costos resumidos del proyecto en el gráfico de la Figura 25:

		2024					
Roles	Costo por hora	Feb	Mar	Abr	May	Jun	Totales
Cliente	- €	16	16	12	4	4	- €
Diseñador de videojuegos	25,00 €	16	128	35,2	27,2	20	5.660,00 €
Programador	20,00 €			76	156	12	4.880,00 €
Artista	20,00 €			44	159,2	32	4.704,00 €
	Totales	32	144	167,2	346,4	68	**15.244,00 €**
Hardware	Costo por unidad	Feb	Mar	Abr	May	Jun	Totales
Alquiler equipo informático	50,00 €	1	1	3	3	3	550,00 €
Creacion DVD - USB	15,00 €					8	120,00 €
	Totales	50,00 €	50,00 €	150,00 €	150,00 €	270,00 €	**670,00 €**
Software	Costo por unidad	Feb	Mar	Abr	May	Jun	Totales
Licenses Unity, Blender, Mixamo, Github	- €						- €
Face Transfer Unlimited Daz3D	49,95 €				1		49,95 €
Assets Daz3D	10,00 €				17	17	340,00 €
	Totales	- €	- €	- €	219,95 €	170,00 €	**389,95 €**
					TOTAL		**16.303,95 €**

Figura 25: Resumen costos. Fuente: Elaboración propia.

Como vemos en la Figura 26, los costos estimados estarían en la media de una demo jugable o quizás un juego *Hyper-Casual*:

Game Types	Popular Examples	Cost Estimation
Test Prototype / Playable Demo	Demo Games	$8K to $15K
Hyper-Casual Game	Rise Up	$25K to $100K
Casual Game	Subway Surf	$45K to $100K
Midcore Game	Clash of Clans	$70K to $500K
Hardcore Game	Pubg, Free Fire	$100K to 3 Million
AAA Game	God Of War	1 Million to 25 Million

Figura 26: Tabla de costos del desarrollo de un videojuego. Fuente: Dotcominfoway.

1.7. Estructura del resto del documento

El resto de los apartados del documento podrían ser los siguientes:

2. Estado del Arte

Este apartado se propone como un análisis detallado de los elementos que van a jugar un papel crucial en la concepción y desarrollo de PWP. Este análisis abarca una serie de apartados centrados en las características específicas del videojuego, contribuyendo a una mejor comprensión de las decisiones de diseño y desarrollo tomadas en este proyecto.

2.1. Historia de los arcades

Los videojuegos arcade [12] originalmente fueron diseñados para atender a las máquinas recreativas (entonces popularmente llamadas "arcade"), y fueron introduciéndose y usurpando, en gran medida, la posición que ocupaban los juegos electromecánicos tradicionales en sitios como centros comerciales, restaurantes, bares o salas de ocio especializadas.

Hoy en día se considera un término genérico utilizado para nombrar el estilo o género de los videojuegos, a pesar de que estos ya no se utilizan en máquinas recreativas, sino que se han trasladado al nivel doméstico a través de las consolas y PCs.

El término proviene del francés arcade, que se traduce como "galerías" o "plataformas" sobre las que se construyeron las primeras máquinas de juego. Posteriormente, el término pasó a utilizarse para designar las salas dedicadas a estas máquinas y, con el tiempo, a los aparatos y juegos propiamente dichos. En la Figura 27 se puede ver una máquina arcade del *Pong*:

Figura 27: Máquina arcade de *Pong*. Fuente: Wikipedia.

El primer juego arcade, Galaxy Game, se lanzó por primera vez en septiembre del 71, replicando el juego Spacewar de 1962. Nutting Associates publicó Computer Space en noviembre de 1971: era una versión hecha por Ted Dabney y Nolan Bushnell, pero no había sido tan compleja, sino más económica. Más tarde, Bushnell y Dabney fundaron Atari y contrataron a Allan Alcorn, quien creó Pong en 1972, el primer gran éxito en los videojuegos. En la Figura 28 se puede ver una máquina arcade del Pong:

Hasta 1975, los videojuegos utilizaban una lógica hecha de transistor a transistor (TTL), pero cuando en 1975 comenzaron a utilizar microprocesadores, la lógica del juego pasó a ser manejada por software. La era dorada, desde los últimos años de la década de 1970 hasta los primeros años de la década de 1980, ha visto un aumento notable tanto en la popularidad como en las ventas de estas máquinas. En la Figura 28 se puede una línea temporal de los juegos de arcade:

Figura 28: Línea temporal de los arcades. Fuente: Bing.

Aunque existe un debate sobre la duración exacta de esta era, se reconoce la importancia de estos videojuegos en el crecimiento de la industria hacia las consolas domésticas y los ordenadores personales. La era dorada comprendió las consolas de videojuegos de primera y segunda generación, y acabó con la crisis del videojuego de 1983, dando paso al resurgimiento de los videojuegos con las consolas de tercera y cuarta generación.

2.2. Auge de las *skins*

Como ya se vio en el apartado 1.1 y 1.2 de este documento, las *skins*, también conocidas como pieles en castellano, son modificaciones gráficas que cambian la apariencia de un personaje u objeto.

Estas modificaciones gráficas pueden lograr un vínculo más profundo y personal entre el jugador y el videojuego, así como una distinción con el resto de jugadores. Las desarrolladoras de videojuegos se dieron cuenta de que podían renovar el interés de los

jugadores simplemente modificando las *skins* de objetos y personajes, por ejemplo, en las campañas de Navidad, Halloween, etc.

Generalmente, el desarrollo de *skins* está controlado por las propias desarrolladoras de videojuegos, permitiendo en ocasiones la creación y modificación de las mismas a través de editores integrados en el propio juego o a través de *mods* (software específico para realizar modificaciones en el juego original).

Con la popularización de internet y de los videojuegos online, el uso de *skins* se ha disparado, generando uno de los mercados más lucrativos de la industria de los videojuegos [13]. En la Figura 29 se puede ver una *skin* de un arma para el videojuego *Counter-Strike: Global Offensive*:

Figura 29: *Skin* para el AK-47 en *CS:GO*. Fuente: Dmarket.

Las grandes desarrolladoras han encontrado nuevas vías para facilitar a los usuarios la creación y venta de sus propias *skins*, obteniendo así un porcentaje de cada transacción y asegurándose unos beneficios extra sin tener que depender de los ingresos directos por la venta del videojuego.

Este mercado cerrado controlado por las desarrolladoras está haciendo que grupos de usuarios presionen para que las *skins* actúen como tokens no fungibles (NFT, por sus siglas en inglés), permitiendo así a los creadores de *skins* tener el control de sus compras digitales y que el mercado de los mismos sea más justo, tal y como muestra un análisis de CoinMarketCap [14].

2.3. Videojuegos similares

Como se ha comentado en otros apartados de este documento, PWP básicamente es un videojuego multijugador y arcade, de temática futbolística cuya mecánica de juego reside en el enfrentamiento 1 contra 1 entre dos jugadores.

Al ser de temática futbolística, el espacio de juego está limitado a unas dimensiones fijas en la que cada jugador deberá proteger su portería y a la vez tratar de meterle gol al jugador contrario.

Esta idea básica recuerda a uno de los primeros videojuegos de la historia, *Pong*, cuya mecánica maneja lo mismos conceptos solo que representando unas palas en lugar de futbolistas, como se mostró en la Figura 14.

Esta sencilla mecánica ha funcionado a lo largo de la historia de los videojuegos, ya sea a través de las distintas interpretaciones del *Pong*, como de distintas versiones del *Breakout*, juego creado por Steve Wozniak e inspirado en el *Pong* de Nolan Bushnell.

Breakout es un videojuego en el que el jugador usa nuevamente una pala para golpear una pelota contra la parte superior de la pantalla, que rompe unos ladrillos de colores al ser golpeado, tratando de eliminarlos todos para superar el nivel, como se puede observar en la Figura 30:

Figura 30: Versión arcade del *Breakout*. Fuente: Wikipedia.

Breakout ha servido de inspiración para unos de los movimientos especiales del PWP, donde el jugador “invoca” a un objeto en pantalla que le hace de muro protector durante un tiempo determinado.

Existen múltiples videojuegos basadas en la mecánica del *Pong*, muchos de ellos proporcionando más complejidad, velocidad, estrategia, etc. para darle una mayor profundidad y disfrute al videojuego.

Un ejemplo es el ya comentado *Windjammers*, distribuido por Data East a través de la Neo Geo en 1994, donde 2 jugadores se enfrentan cara a cara lanzándose un disco volador y tratando de que este sobrepase al rival para anotar distintos puntos, como vimos en la Figura 15.

A pesar de ser un videojuego publicado hace 30 años, en 2022 se lanzó una segunda parte del videojuego para PC y las principales consolas del mercado, contando además con un modo online para enfrentamientos a través de internet. Una captura de pantalla del *Windjammers 2* podemos verla en la Figura 31:

Figura 31: *Windjammers 2*. Fuente: Steam.

PWP se inspira en los videojuegos mencionados, tratando de diferenciarse con distintas modificaciones en la mecánica principal y añadiendo un componente de representación de los personajes muy específico y dedicado al público al que va dirigido.

Esta personalización gráfica consiste en la inclusión de distintos detalles en los escenarios, música y personajes del juego, basados en la caracterización, movimientos, gustos, etc. de los amigos del autor. Y tratando de conseguir un resultado similar a otros videojuegos con

fuerte similitud artística con personales reales, como vimos en la Figura 1 que representa a un personaje famoso dentro del videojuego Fortnite.

2.4. Motores de videojuegos

Los motores de videojuegos [15] facilitan una serie de tareas comunes en el proceso de su creación, como suelen ser el renderizado de gráficos, las animaciones, los sonidos, simulación de leyes de la física, scripting, etc.

Esto es una ayuda importante para los desarrolladores, ya que les evita programar desde cero todas esas funcionalidades ya conocidas y pueden entonces centrarse en las tareas específicas de su videojuego.

Históricamente, las empresas creaban sus propios motores para desarrollar los videojuegos que luego iban a comercializar. De esta manera obtenían un motor específico para un determinado tipo de juego que luego podrían reutilizar para sacar segundas partes del videojuego o nuevos títulos de características similares. En la Figura 32 se muestra una captura del *Maniac Mansion*:

Figura 32: *Maniac Mansion* desarrollado con el motor SCUMM. Fuente: Transmediaparade.

A medida que los videojuegos fueron teniendo más relevancia, hubo empresas que se dedicaron a crear motores para vendérselos a las desarrolladoras, ahorrándoles mucho trabajo a un precio asequible.

Como ocurre con otro tipo de software, nos encontramos con motores de licencia gratuita, otros de pago y otros mixtos. Si bien los primeros suelen ser suficientes para proyectos amateur o juegos de corte indie, los segundos suelen ser los más usados en la industria.

A continuación se describen unos ejemplos de motores de videojuegos:

2.4.1. Unreal Engine

Programado en C++ y C# y lanzado en 1998 por la desarrolladora de videojuegos Epic Games, este motor se creó con la idea de facilitar el desarrollo de videojuegos del género *Shooter* en primera persona (*First Person Shooter,* FPS en inglés).

Con las posteriores versiones del motor, los géneros de videojuegos a los que iba encaminado se ampliaron así como las plataformas a las que iba dirigida: desde los distintos tipos de PCs a las consolas de última generación, pasando por dispositivos móviles y equipos de realidad virtual.

Una de las características más destacadas de este motor es el renderizado fotorrealista (generar una imagen a partir de un modelo 3D) que se puede alcanzar gracias a las tecnologías implementadas de luz, cinemática y geometría virtualizada. En la Figura 33 se muestra una captura del motor Unreal:

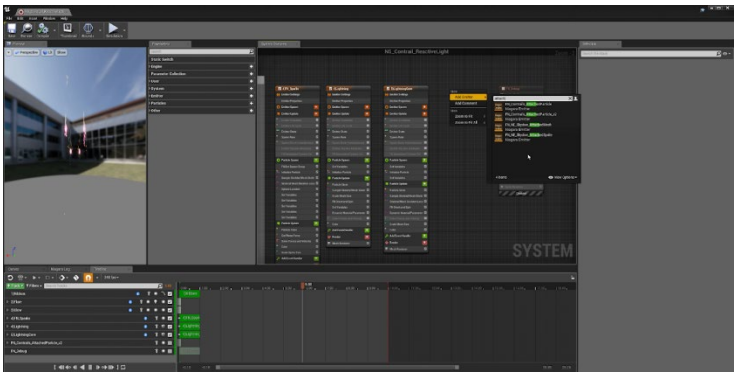


Figura 33: Sistema Niagara VFX en *Fortnite* con Unreal Engine. Fuente: Unrealengine.

Es un motor de uso gratuito, cobrando un 5% de las ganancias cuando estas son superiores el millón de dólares. Factores como el anterior y otros ya vistos como sus espectaculares resultados en el apartado gráfico, han hecho que otros sectores se hayan visto interesados en su uso: cine, medicina, automoción, construcción, etc.

Algunos ejemplos de videojuegos creados con este motor son: *Fortnite, PlayerUnknow's Battleground (PUBG), Gears of War, Bioshock* (I y II), la serie de *Batman Arkham, Street*

Fighter V, Star Wars Jedi: Fallen Order, Resident Evil (2 y 4 VR), Ark (I y II), Rocket League, etc.

2.4.2. Unity

Igual que ocurría con el Unreal Engine, Unity está programado en C++ y C#. Fue lanzado en 2005 por la empresa *Unity Technologies*, pero a diferencia de la anterior, el videojuego para el cual desarrollaron el motor no tuvo éxito.

Sin embargo, vieron potencial en el motor que habían creado, se centraron en mejorarlo y distribuirlo a precios populares, ya que su idea era que cualquier programador pudiera hacer uso de la herramienta.

Esta es una de las características más importantes de este motor, puesto que desde sus inicios ha permitido a muchas desarrolladoras indies generar sus juegos a coste cero o muy bajos gracias a su licencia de tipo personal (gratuita).

Como consecuencia, Unity posee una gran comunidad de usuarios y destaca también en la facilidad de uso del motor, la enorme cantidad de recursos en su tienda (gratuitos y de pago) e, igual que ocurre con Unreal Engine, también cubre el espectro de plataformas existentes.

Algunos ejemplos de videojuegos creados con este motor son: *Cuphead, Fallguys, Among Us, Hearthstone, Call of Duty: Mobile, Rust, Cities: Skylines, Subnautica, Disco Elysium, Genshin Impact, Kerbal Space Program, Hollow Knight, Monument Valley* (1 y 2), *Ori and the Blind Forest, Ori and the Will of the Wisps*, etc. En la Figura 34 se muestra una captura del *Monument Valley 2* en Unity:

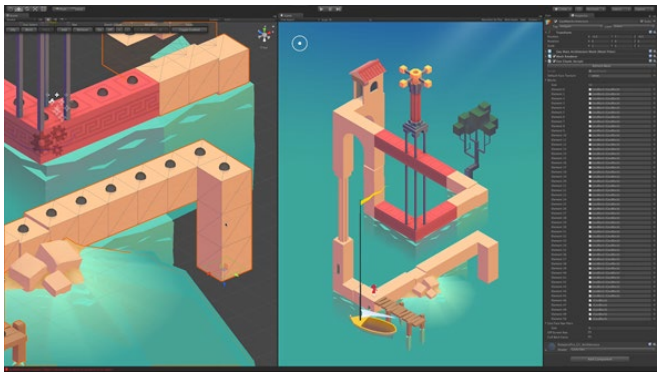


Figura 34: Desarrollo de *Monument Valley 2* en Unity. Fuente: Unity.

2.4.3. GameMaker

Originalmente conocido como Animo y programado inicialmente en Delphi, C++ y C#, fue lanzado en 1999 por Mark Overmars y continuado, a partir del 2007, por la empresa YoYo Games. Diseñado para que cualquier persona sin conocimientos de programación pudiera crear videojuegos en 2D.

Esta manera de originar juegos de manera visual, arrastrando y soltando componentes (*drag and drop* en inglés) es su principal baza, ya que permite producir juegos de manera sencilla. Por otro lado, también ve mermado su potencial, por el hecho de que el número de acciones es limitado.

Pero posee un lenguaje de scripts propio (GML, *Game Maker Language*) para que aquellos programadores más avanzados puedan expandir las posibilidades del motor. Incluso aunque esté diseñado para generar juegos en 2D, gracias al GML, pueden llegar a producirse juegos en 3D.

Igual que ocurría con los motores de Unreal Engine o Unity, sus juegos pueden ejecutarse en la mayoría de plataformas, dispone de licencias tanto gratuitas como de pago y pueden desarrollarse juegos de cualquier género.

Algunos ejemplos de videojuegos creados con este motor son: *Spelunky, Hotline Miami* (1 y 2), *Undertale, Hyper Light Drifter, Cook, Serve, Delicious!* (1, 2, y 3*), Risk of Rain, Heartbound, Gunpoint, Nuclear Throne Deltarune, Gods Will Be Watching, Nidhogg* (1 y 2), *The Red Strings Club*, etc. En la Figura 35 se muestra una captura del *Spelunky* en GameMaker:

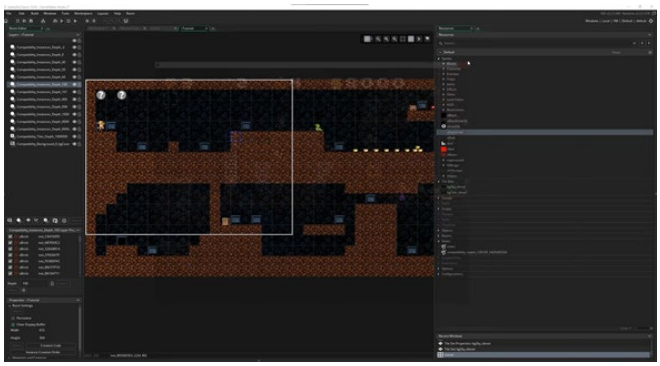


Figura 35: Desarrollo de *Spelunky* en GameMaker Studio 2. Fuente: Youtube.

2.4.4. Godot

Programado en C y C++, fue desarrollado en el año 2001 por OKAM Studios y en 2014 su código fuente fue liberado como código abierto y se pasó a una licencia MIT (licencia de software libre).

A diferencia del resto de motores vistos, Godot es completamente gratuito. Y al ser de código abierto, la comunidad de usuarios colabora activamente en la mejora del motor, que permite el desarrollo de videojuegos en 2D, en 3D y, en la última versión, en 2.5D.

Los desarrolladores pueden utilizar múltiples lenguajes de programación en Godot: GDScript, C++, C# y VisualScript. Pero la comunidad ha ampliado el número de lenguajes y permite el uso de Rust, Lua, Javascript, Nim, D, Haskell y Clojure.

Igual que ocurre con otros motores, *Godot* dispone de desarrollo de realidad virtual y sus juegos pueden ejecutarse en la mayoría de plataformas: dispositivos móviles, web PC y consolas. Aunque el soporte de estas últimas no ocurre de manera nativa, sino a través del empleo de editores externos.

Algunos ejemplos de videojuegos creados con este motor son: *Sonic Colors: Ultimate, The Interactive Adventures of Dog Mendonça & Pizzaboy, Deponia* (*ports* de iOS y PlayStation 4), *Commander Keen in Keen Dreams* (*port* de Nintendo Switch), *Carol Reed Mysteries, Cruelty Squad, Hardcoded*, etc. En la Figura 36 se muestra una captura del *The Interactive Adventures of Dog Mendonça & Pizzaboy* en Godot:

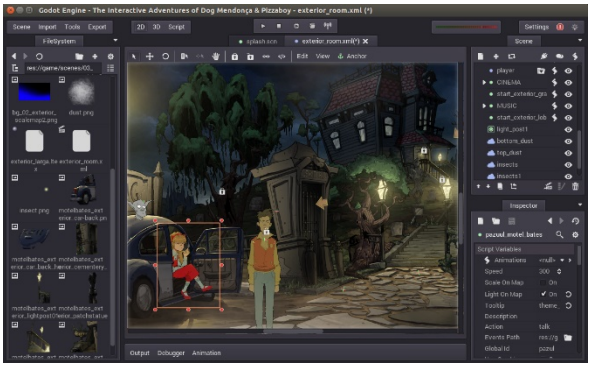


Figura 36: *The Interactive Adventures of Dog Mendonça & Pizzaboy* en Godot. Fuente: 80LV.

Para el desarrollo de PWP se ha escogido Unity debido a los conocimientos del autor con esta herramienta y porque cubre todas las necesidades que el proyecto necesita.

2.5. Modelado y animación 3D

El modelado 3D [16] es un proceso que representa objetos gráficos en tres dimensiones a través de un software especializado siendo, estos modelos representados, desde simples polígonos hasta objetos tridimensionales realmente complejos.

Se utilizan en una amplia variedad de aplicaciones, siendo una técnica esencial que se utiliza para crear diversas representaciones en el mundo digital. El modelado 3D ha consolidado su posición en muchas industrias creativas, por ejemplo, el cine, televisión, videojuegos, etc.

Las técnicas básicas del modelado 3D incluyen: modelado poligonal, modelado de curvas y escultura digital. Una de las técnicas más populares utilizadas en la realización de videojuegos y animaciones es el modelado poligonal, gracias a su sencillez y a la rapidez de renderizado.

En el modelado poligonal, se realiza la conexión de puntos en un espacio tridimensional (vértices) formando un polígono maleable (malla) que define el aspecto de la superficie del modelo, y sobre ella se pueden colocar texturas y efectos visuales, como vemos en la Figura 37:

Figura 37: Modelado poligonal en *Daz Studio*. Fuente: Artstation.

Por otro lado, la animación 3D [17] es la que da vida a estos modelos, creando una ilusión a través de una serie de imágenes en rápido movimiento (fotogramas) que representan una etapa del mismo y, cuando se muestran a altas velocidades, produciendo un resultado realista.

En los inicios de los videojuegos, la animación se limitaba a *sprites* bastante simplistas y animaciones básicas. Fue con la llegada de la animación 3D donde los creadores pudieron diseñar personajes y entornos más reales e inmersivos, desarrollando así narrativas y mundos mucho más complejos En la Figura 38 se muestra una animación en Blender:

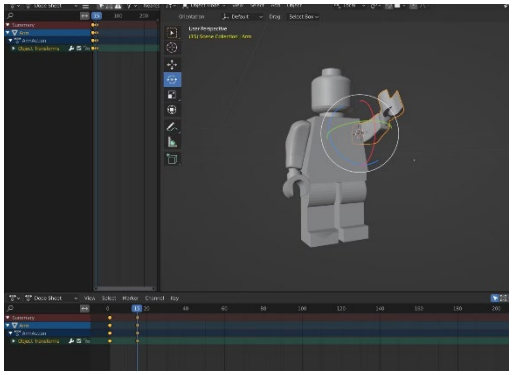


Figura 38: Animación en *Blender*. Fuente: Artstation.

En el desarrollo de PWP, el modelado y la animación 3D juegan un papel fundamental a la hora de representar a los jugadores del público objetivo con su carácter, movimientos, gustos, etc.

2.5.1. Herramientas 3D

A continuación se listan una serie de herramientas que permiten tanto el modelado como la animación 3D:

- **Autodesk Maya**: Propiedad de Autodesk, se destaca por su conjunto de herramientas especializadas para el desarrollo de personajes y escenarios.

- **3DS Max**: También propiedad de Autodesk, conocido por su versatilidad en la creación de gráficos en 3D para arquitectura, videojuegos, animación, efectos especiales, etc.

- **Cinema 4D**: Desarrollado por Maxon, Cinema 4D es conocido por su interfaz intuitiva y amigable para principiantes.

- **Blender**: Programa de código abierto y gratuito que permite desarrollar todo lo relacionado con el 3D: modelado, rigging, renderizado, iluminación, animación, etc.

- **Zbrush**: Propiedad de Pixologic, Zbrush es conocido por su capacidad para crear modelados 3D muy realistas y detallados.

- **Daz3D**: Desarrollado por Daz Productions, se distingue por ser una plataforma de figuras 3D altamente personalizables y por su gran librería de recursos.

Cada una de estas herramientas de la lista anterior permiten a los artistas crear y animar personajes, objetos y entornos en 3D. Aunque cada una de ellas tiene sus propias características y ventajas, todas facilitan la creación de contenido 3D de alta calidad.

Para PWP se ha escogido Daz3D y Blender debido a los conocimientos del autor con estas herramientas y por el uso de un *plugin* específico (Face Transfer Unlimited) que ya se ha mencionado en el apartado 1.4.2.

2.6. Referencias en juego al público objetivo

Como ya se ha comentado a lo largo del documento, PWP ofrece una experiencia personalizada y emocional para un público específico (los amigos del autor) a través de distintos elementos que forman parte del videojuego.

Estos elementos pueden ser referencias directas, como los personajes del videojuego, pero también hay otras referencias indirectas, como músicas u objetos que pueden ser más sutiles y pasar desapercibidas en un primer visionado del juego.

2.6.1. Personajes

Es el elemento más evidente en el que el público específico se ve reflejado en el videojuego: los personajes protagonistas son la réplica digital de ellos mismos.

Para lograrlo se han creado modelos 3D de cada uno de ellos, basados en sus proporciones físicas y el modelado de sus rasgos faciales en consonancia con fotos y los conocimientos del autor acerca de sus personas.

La ropa y complementos también han sido seleccionados en función de las características de los protagonistas: por ejemplo, uno de ellos es fan de la banda estadounidense Metallica y lleva una camiseta con el logo del grupo, que puede verse en la Figura 39:

Figura 39: Logo de Metallica. Fuente: Wikipedia.

2.6.2. Animaciones

Para conseguir un resultado creíble es necesario que los personajes representados se animen de manera personalizada, para que el efecto logre un mayor impacto y cada uno de ellos se identifique de manera más satisfactoria.

Realizar todas las animaciones personalizadas para cada uno de los personajes es un proceso largo y que quizás sea muy ambicioso para el límite temporal establecido para la realización del proyecto.

Por lo tanto, muchas animaciones comunes, como pueden ser las de moverse en el terreno de juego, golpear la pelota, lanzar un objeto, etc. serán obtenidas mediante la web de Adobe Mixamo (ver Figura 17).

Sin embargo, las animaciones faciales de los protagonistas en la pantalla de selección de jugadores serán realizadas con la herramienta Daz Studio de Daz3D para replicar expresiones y gestos típicos de cada uno de ellos.

Por ejemplo, uno de los protagonistas saca la lengua de una manera determinada que puede ser replicada eficazmente con dicho programa. En la Figura 40, vemos un ejemplo de cómo pueden modificarse expresiones faciales con Daz Studio:

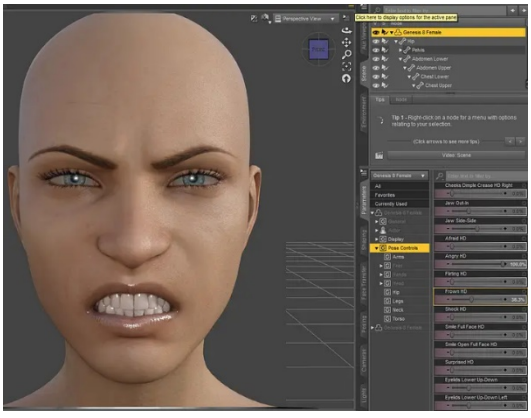


Figura 40: Expresiones faciales en Daz Studio. Fuente: Renderguide

2.6.3. Props

El juego contendrá múltiples *props* que estén relacionados con los protagonistas para, superada la sorpresa inicial de ver una copia digital suya en el juego, seguir fortaleciendo el vínculo entre la persona y el personaje.

Por ejemplo, uno de los protagonistas tiene como afición la pesca, así que en su movimiento especial tendrá una caña de pescar en las manos al ejecutar dicho movimiento.

En general, la mayoría de los *props* serán genéricos, como la caña de pescar, si bien pueden existir otros personalizados, como por ejemplo una bandera de decoración en un escenario con la imagen del escudo de Santa Cruz (lugar dónde han crecido los protagonistas), como en la Figura 41:

Figura 41: Escudo del equipo de Santa Cruz C.F. Fuente: SantaCruzCF

2.6.4. Escenarios

Los escenarios donde transcurran las principales interacciones entre los protagonistas, como son las pantallas de prepartido, partido, pospartido y entrega de premios, tendrán escenarios personalizados.

El campo de juego del partido tiene que tener unos elementos específicos como vimos en el apartado 1.2.2, por ejemplo, un marcador con información de la vida y energía que, evidentemente, no existe en la vida real. Pero sí que dispondrá de otros elementos que recuerdan a dónde los protagonistas disputaban esos encuentros: la verja protectora, los árboles y la hierba del entorno, etc.

2.6.5. Textos

Otra manera de reforzar el vínculo entre los personajes del juego y las personas que lo están jugando es a través de los textos que aparecen en las pantallas de selección de jugadores y pospartido.

En la pantalla de selección, se mostrará el nombre del movimiento especial del personaje seleccionado, que hace referencia directa al jugador ya sea a través de una broma o característica del mismo.

En la pantalla de pospartido, se mostrará una frase típica del jugador en la zona inferior de la pantalla, como ocurre en otros videojuegos (típicamente de lucha) tras haber vencido al contrario después de un combate, como el ejemplo de *Street Fighter 2* en la Figura 42:

Figura 42: Frase de victoria de Chun Li en SF2. Fuente: Meristation

2.6.6. Músicas

Las músicas y efectos de sonido juegan un papel muy importante en la mayoría de videojuegos para ambientar y sumergir al jugador en el mismo. En algunos de ellos son el elemento principal de la mecánica de juego, como ocurre en *Guitar Hero, Dance Dance Revolution*, etc.

Sin embargo, para el proyecto actual, disponemos de un tiempo muy limitado para poder desarrollar nuestras propias músicas y efectos de sonido, así que se utilizaran las ya existentes en internet pero que a la vez tengan algún tipo de vínculo con algunos de los protagonistas del juego.

Por ejemplo, varias de las personas incluidas en el juego han compartido parte de su infancia en un salón recreativo, disfrutando del *Outrun* y de la canción *Passing Breeze*, que se muestra en la Figura 43 y que se incluirá en PWP:

Figura 43: Pantalla de selección de música en *Outrun*. Fuente: Recalbox

3. Definición de los productos

Como ya se ha comentado en el apartado 1.2.4, PWP se distribuirá en un formato digital para PC pero también en soporte físico a través de una copia única y limitada (DVD y USB) para cada uno de los protagonistas del videojuego con su personaje en la portada.

A continuación se explicará el contenido del videojuego en la copia digital, que a su vez irá incluida en la copia física, apartado dónde se verá un boceto del diseño para el DVD y USB y se detallará el proceso de realización.

3.1. Copia digital

3.1.1. Modos de juego

PWP contendrá 2 modos de juego: torneo y duelo. En el modo torneo los jugadores seleccionados se irán enfrentando entre ellos a través de un sistema de eliminación por cruces, tal y como vimos en la Figura 7. Mientras que en el modo duelo, será un enfrentamiento directo entre dos jugadores y, al finalizarse, el juego volverá a la pantalla de menú principal.

Como la mecánica de los partidos en el juego está pensada para el enfrentamiento entre dos jugadores simultáneamente, a nivel de desarrollo no supone un esfuerzo añadido permitir la selección múltiple de jugadores controlados por humanos. Es decir, aunque haya 8 personas dispuestas a jugar, en los enfrentamientos de partidos solamente se van a enfrentar uno contra uno.

Como el número de jugadores controlados por humanos puede variar, hay rondas del torneo en que un jugador se enfrente a un personaje controlado por la computadora. En esos casos, la dificultad del enfrentamiento se verá incrementada en función de la ronda del torneo en qué se encuentre el jugador.

En el siguiente apartado explicaremos los distintos niveles de dificultad y cómo son dichos incrementos.

3.1.2. Niveles de dificultad

En cualquiera de los modos de juegos existirán 3 niveles de dificultad: fácil, normal y difícil. La especificación de los niveles tienen dos funciones principales: ajustar el juego a la habilidad del jugador y aumentar la rejugabilidad.

Los parámetros que se van a modificar en función del nivel de dificultad seleccionado son:

- **Velocidad inicial de la pelota**:
  Cada nivel de dificultad irá incrementando la velocidad inicial de la pelota, para que el jugador se acostumbre al ritmo del partido y se ajuste a su nivel.

- **Velocidad del personaje controlado por la computadora**:
  El movimiento de dicho personaje se irá posicionando en la pantalla en función de la posición de la pelota en el terreno de juego. Su velocidad estará determinada por el atributo de “velocidad” del jugador que esté manejando, pero también por un factor determinado por el nivel de dificultad seleccionado, para así asegurar que cada vez que se supere una ronda del torneo sea más complicado vencer a la computadora.

- **Cadencia de lanzamiento de objetos del personaje controlado por la computadora**:
  Como en el caso anterior, a medida que se superen rondas, el personaje manejado por la computadora incrementará la cadencia de lanzamiento de los objetos en función de la dificultad seleccionada.

Con estas 3 variaciones de parámetros, junto con los condicionantes de si el jugador está enfrentándose a la computadora o en qué ronda del torneo se encuentra, se conseguirá que el jugador tenga opciones de superar el partido y que a la vez le suponga un reto mayor el siguiente enfrentamiento.

3.1.3. Sistema de puntuación

Otra de las maneras para potenciar la competitividad entre jugadores y la rejugabilidad es disponer de un sistema de puntuación en el juego. PWP tendrá dos sistemas de puntos: número de torneos ganados y número de puntos totales.

El número de torneos totales es un contador incremental por personaje del juego. Si hay un personaje dentro del juego llamado Javi y gana un torneo, aumentará en 1 el número de torneos ganados que tiene ese personaje.

Por otro lado, también se llevará un contador de puntos totales para cada personaje, que se verá incrementado en función de los siguientes parámetros:

- Cada vez que golpea al contrario con un objeto: 3 puntos.
- Cada vez que mete un gol: 15 puntos.
- Cada vez que gana un torneo: 75 puntos.

En la pantalla del menú principal se mostrará aquel jugador que tiene más número de torneos ganados y su puntuación total, como vimos en la Figura 3. En caso de que haya un empate en el número de torneos ganados entre varios jugadores, está puntuación total es la que servirá de desempate. En la Figura 44 vemos la tabla de puntuaciones en el juego de *The Simpsons*, pero mostrando los primeros 5 clasificados en lugar de 1 como en PWP:

Figura 44: Tabla de puntuaciones en *The Simpsons*. Fuente: Gamesdatabase

3.1.4. Controles de juego

El juego puede utilizarse con dos controles: teclado y mando. El uso del teclado está habilitado porque no todos los jugadores disponen de un mando, pero si de un teclado de ordenador.

Tal y como vimos en la Figura 9, los controles a utilizar serán:

- Moverse: Tanto en los menús como en el partido.
    - Teclado: Jugador izquierda: W y S, jugador derecha: Flechas
    - Mando: Botones de dirección y Joystick izquierdo

- Disparar: Lanza objetos al contrario.
    - Teclado: Jugador izquierda: G, jugador derecha: L
    - Mando: Botón A

- Especial: Utiliza el movimiento especial del jugador.
    - Teclado: Jugador izquierda: H, jugador derecha: K
    - Mando: Botón B

- Apuntar: Establece la dirección de la pelota al golpearla.
    - Mando: Joystick derecho

Como puede verse en las indicaciones anteriores, en el caso de que haya 2 jugadores utilizando un mismo teclado, se especifica que teclas utilizará el personaje situado a un lado y al otro del campo.

También se observa que la opción de apuntar no está disponible cuando se juega con teclado: esto es debido a que hay personas al que va dirigido el juego que no tienen mando y no están acostumbradas a jugar a videojuegos. Y se ha optado por simplificar los controles del teclado para que no se sientan abrumados con toda la información para jugar.

Como se ha visto en la pantalla de carga del punto 1.2.2, se mostrará la información de los controles antes de cada partido, tanto para teclado como para mando, así los jugadores lo tienen presente justo antes del encuentro.

Se ha decidido de no disponer de un menú para redefinir teclas o botones del mando, dónde los jugadores podrían asignar los controles a su gusto, porque se dará prioridad a otras partes del desarrollo dadas las limitaciones de tiempo del proyecto.

3.1.5. Personajes

Cada jugador controlará a un personaje del juego a lo largo de cada partida. Como ya se ha explicado, los personajes intentarán ser una réplica digital de los amigos del autor, siendo este apartado una de las principales atracciones para el público al que va dirigido. En la Figura 45 puede verse los personajes de *Mortal Kombat 1* que son la representación digital de los actores:

Figura 45: Pantalla de personajes en *MK1*. Fuente: Fightersgeneration

En el modo torneo los jugadores se enfrentarán en un cuadro de eliminación directa de 8 jugadores: 4 jugadores por la parte izquierda del cuadro y otros 4 jugadores por la parte derecha. Se ha establecido ese número de personajes porque se considera que 3 partidos como máximo (si un jugador llega a la final) para un torneo es un número aceptable de partidos.

Como puede verse en la Figura 6, cada uno de los personajes tiene tres atributos:

- **Ataque**: Reduce la vida del contrario que es impactado con un objeto lanzado. Cuanto mayor ataque, más daño genera.

- **Defensa**: Habilidad para reducir el daño generado por los impactos de los objetos. Cuanta mayor defensa, menor daño generado por el objeto.

- **Velocidad**: Es la habilidad para mover al jugador más rápido por el terreno de juego. Cuanta mayor velocidad, más rápido se mueve el jugador.

Además cada uno de los personajes tiene un movimiento especial que le aportará una ventaja temporal y que veremos en detalle en el apartado siguiente.

3.1.6. Movimientos especiales

Cada jugador podrá hacer uso del movimiento especial cuando su barra de energía especial esté completa y su personaje ejecutará una acción determinada que proporcionará una ventaja temporal al jugador.

Cada personaje del juego tiene un movimiento especial personalizado, tal y como vimos con los ejemplos del apartado 1.2.3, que está relacionado estrechamente con el propio personaje y que sirve para reforzar el vínculo entre ellos. Como ejemplo, uno de los personajes del juego, Javi, tiene en la vida real gemelos y que entrarán en acción para cubrirle dos zonas del campo para evitar el gol. Un ejemplo de un movimiento especial es el *Tiro del águila* en *Captain Tsubasa: Rise of New Champions* de la Figura 46:

Figura 46: *Tiro del águila* en *Captain Tsubasa*. Fuente: Fandom

La duración del movimiento especial no es estrictamente la misma para cada jugador, depende del tipo de movimiento: por ejemplo, puede haber un movimiento especial que golpee la pelota de una determinada manera y sólo durará ese golpeo de balón, mientras que otros movimientos, como el de los gemelos de Javi, durará hasta que ocurra un gol en el partido.

Los movimientos especiales estarán balanceados para que no aporten una ventaja excesiva al jugador que lo invoque. Algunos movimientos especiales podrán contrarrestar o mermar el

efecto de otro movimiento especial, dotando así de un componente estratégico al partido del cual el jugador puede sacar provecho.

3.1.7. Vidas y energía especial

Cada jugador contará con 2 atributos, vida y energía especial, que se consumirán y recargarán en determinadas circunstancias. Se dispondrá en el marcador del partido dos barras para cada jugador que representan la cantidad de vida y energía especial que tienen en cada momento. En la Figura 47 se puede ver un ejemplo de barra de vida (roja) y de energía (verde) en *Dark Souls*:

Figura 47: Barras de vida y energía en *Dark Souls*. Fuente: Reddit

En PWP, la barra de vida se representará de color rojo y, cuando el partido se inicia o ocurre un gol, se recargará por completo automáticamente. Cada vez que un jugador recibe el impacto de un objeto lanzado por el contrario, su barra de vida disminuirá acorde al atributo de defensa del jugador y al atributo de ataque del jugador contrario.

Cuando un jugador vacía su barra de vida, no podrá moverse, lanzar objetos, ejecutar movimientos especiales y tampoco golpear la pelota. Se mostrará una animación para que el jugador sepa que su personaje ha agotado su vida.

La barra de energía especial se representa de color azul y se recarga automáticamente en los siguientes escenarios:

- Cuando se ha seleccionado el nivel de dificultad “Fácil” y el partido se inicia.
- Cuando se ha seleccionado el nivel de dificultad “Normal” y el partido se inicia en un duelo o la primera ronda del torneo.

Cada vez que un jugador impacta en el contrario con el lanzamiento de un objeto, se recargará una parte de su barra de energía especial. Cuando dicha barra esté completa, el

jugador podrá ejecutar el movimiento especial de su personaje presionando el botón adecuado del controlador y vaciando por completo la barra de energía especial.

Cuando la barra de vida esté a punto de agotarse o la barra de energía especial esté completa, la barra correspondiente parpadeará para que el jugador pueda darse cuenta de esa situación.

En el juego existirán una serie de *power-ups* que permiten recargar las barras de vida y energía especial, como veremos a continuación en el siguiente apartado.

3.1.8. Power-ups

Los *power-ups* son una serie de objetos que aparecerán aleatoriamente en el terreno de juego y que, cuando la pelota pase por encima de ellos, se activarán desencadenando alguna acción en el partido.

Ejemplos de *power-ups* en PWP:

- **Vida**: Recargará parcialmente la barra de vida del jugador que lo haya activado con la pelota.

- **Energía especial**: Recargará parcialmente la barra de energía especial del jugador que lo haya activado con la pelota.

- **Escudo protector**: Proporcionará al jugador que lo haya activado con la pelota, un escudo que lo hará inmune temporalmente a los objetos lanzados por el contrario.

En la Figura 48 podemos ver un ejemplo de *power-up,* la seta en el *Super Mario Bros.:*

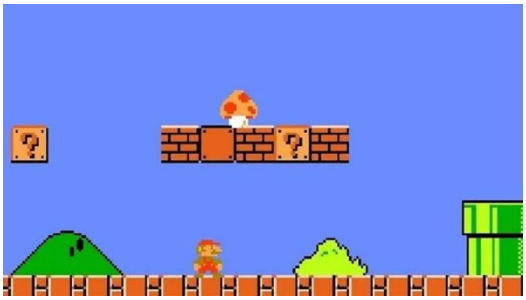


Figura 48: *Power-up* de la seta en *Super Mario Bros.* Fuente: Denofgeek

3.2. Copia física

Tal y como vimos en el apartado 1.2.4, la copia física del juego constará de:

- Caja genérica de DVD con carátulas
- DVD
- USB

3.2.1. Diseño

Todos los elementos anteriores estarán personalizados de la siguiente manera (consultar Figura 13):

- La carátula frontal mostrará al personaje del videojuego que representa la persona a la que se le entregue la copia física junto con el logo del videojuego.

- La carátula trasera mostrará la información básica del videojuego, la cara de cada uno de los personajes que salen en el mismo, el logo de PWP y distintas capturas de pantalla del juego.

- La carátula lateral mostrará el nombre del videojuego.

- El DVD llevará pegada una etiqueta en la que irá impreso el busto del personaje de la carátula frontal en la parte inferior y el logo del videojuego en la parte superior.

- El USB llevará impreso el logo del videojuego.

3.2.2. Realización

Para la realización de la copia física se adquirirán tantas cajas genéricas, DVDs y USBs como personajes haya en el juego y se utilizará un software de edición gráfica como Photoshop o Paint.net para los diseños.

Las etiquetas del DVD se pegarán al mismo con un *kit* de etiquetado, como puede ser APLI 10959-Kit centrador de etiquetas para CD/DVD, tal y como se muestra en la Figura 49:

Figura 49: *Kit* centrador de etiquetas de Apli. Fuente: Amazon

Para la impresión de los USBs, hay tiendas especializadas a las cuales se les remite un diseño y ellos lo imprimen en el propio USB, como por ejemplo la tienda CeaMere Elec Store en Aliexpress.

Un juego de imanes con película adhesiva, como los de la tienda de Yizhet en Amazon, para sujetar el usb en el interior de la caja del DVD. En la Figura 50 se muestra una representación del interior de la caja del DVD, junto con el imán y el USB personalizado:

Figura 50: Caja del DVD junto con el imán y el USB. Fuente: Elaboración propia

4. Diseño

En este apartado se identifican los equipos, herramientas, *assets*, *props* y recursos utilizados para el desarrollo del videojuego. También se muestran las principales escenas, personajes, objetos y *power-ups* y finalmente se muestra la arquitectura del juego.

4.1. Hardware

El proyecto ha sido desarrollado con distintas estaciones de trabajo debido a que algunas de las tareas pueden realizarse en paralelo. Un ejemplo es la generación de los *renders* de los personajes: utilizando distintos equipos a la vez se consigue reducir los tiempos de generación de manera considerable.

En la Figura 51 se indican los componentes utilizados por una de las estaciones de trabajo, con las especificaciones de hardware más bajas de todos los equipos utilizados en el desarrollo del proyecto:

Resumen

Computadora:	
Tipo de computadora	Equipo basado en x64 ACPI
Sistema operativo	Windows 10 Enterprise Professional
Service Pack del sistema operativo	-
Internet Explorer	9.11.19041.0
DirectX	DirectX 12.0
Fecha / Hora	2024-05-19 / 00:08
Motherboard:	
Tipo de CPU	6x , 3600 MHz
Nombre del motherboard	Gigabyte B550M DS3H
Memoria del sistema	32636 MB
Tipo de BIOS	AMI (12/24/18)
Puerto de comunicación	Puerto de comunicaciones (COM1)
Monitor:	
Placa de vídeo	NVIDIA GeForce GTX 950 (2048 MB)
Monitor	Monitor PnP genérico [NoDB] (U5V0V55V)
Multimedia:	
Placa de sonido	Controladora de High Definition Audio [1002-1637] [NoDB]
Placa de sonido	Controladora de High Definition Audio [1022-15E3] [NoDB]
Placa de sonido	Controladora de High Definition Audio [10DE-0FBA] [NoDB]
Almacenamiento:	
Controlador IDE	Controladora SATA AHCI estándar
Controlador de almacenamiento	Controladora de espacios de almacenamiento de Microsoft
Controlador de almacenamiento	OSFDisk Virtual Adapter
Controlador de almacenamiento	Xvdd SCSI Miniport
Disco rígido	CT240BX300SSD1 (223 GB)
Disco rígido	Hitachi HTS541612J9SA00 (120 GB, 5400 RPM, SATA)
Disco rígido	Netac SSD 720GB (670 GB)
Disco rígido	ST3500418AS (500 GB, 7200 RPM, SATA-II)
Estado SMART de los discos rígidos	OK
Particiones:	
C: (NTFS)	99455 MB (1432 MB libre)
D: (NTFS)	72785 MB (2757 MB libre)
E: (NTFS)	140.3 GB (1.0 GB libre)
F: (NTFS)	144.5 GB (14.0 GB libre)
G: (NTFS)	80994 MB (5376 MB libre)
H: (NTFS)	101.9 GB (35.8 GB libre)
I: (NTFS)	111.8 GB (24.4 GB libre)
J: (NTFS)	125.8 GB (34.0 GB libre)
K: (NTFS)	599.6 GB (58.5 GB libre)
Tamaño total	1471.2 GB (177.1 GB libre)
Dispositivos de entrada:	
Teclado	Dispositivo de teclado HID
Mouse	Mouse compatible con HID
Red:	
Dirección IP primaria	192.168.1.10
Placa de red	Realtek PCIe GBE Family Controller (192.168.1.10)
DMI:	
DMI Fabricante del BIOS	American Megatrends International, LLC.
DMI Versión del BIOS	FA
DMI Fabricante del sistema	Gigabyte Technology Co., Ltd.
DMI Nombre del sistema	B550M DS3H
DMI Versión del sistema	-CF
DMI Número de serie del sistema	Default string
DMI UUID del sistema	74025603-3C044E05-7106CA07-00080009
DMI Fabricante del motherboard	Gigabyte Technology Co., Ltd.
DMI Nombre del motherboard	B550M DS3H

Figura 51: Especificaciones de hardware de uno de los equipos de desarrollo. Fuente: Elaboración propia

4.2. Software

Es importante destacar que el software utilizado en el proyecto se ejecuta bajo el sistema operativo de Windows 10, aunque podría ejecutarse bajo otros sistemas operativos porque existen versiones disponibles para ello o herramientas alternativas para conseguir el mismo resultado.

4.2.1. Unity

Como vimos anteriormente, Unity es un motor de videojuegos que facilita una serie de tareas comunes en el proceso de desarrollo, como el renderizado de gráficos, animaciones, sonidos, simulación de físicas, scripting, etc.

En PWP, se tiene una escena de Unity por cada pantalla del videojuego y unos scripts asociados a las escenas para controlar su funcionamiento, haciendo uso del lenguaje C# con el entorno de desarrollo de Visual Studio. En la Figura 52 vemos la escenas del proyecto:

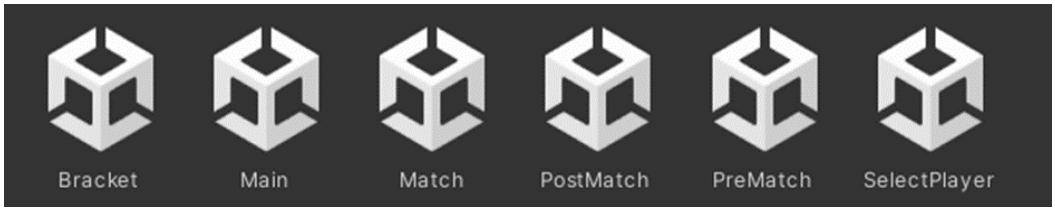


Figura 52: Escenas de PWP en la actualidad. Fuente: Elaboración propia

Por otro lado, los gráficos de los modelos son manejados a través de *sprites* y *Animator Controllers* que nos permitirán manejar las distintas animaciones. La Figura 53 muestra una captura de pantalla del *Animator Controller* para el personaje Javi y uno de los *sprite*s que se utiliza en la animación cuando celebra la victoria:

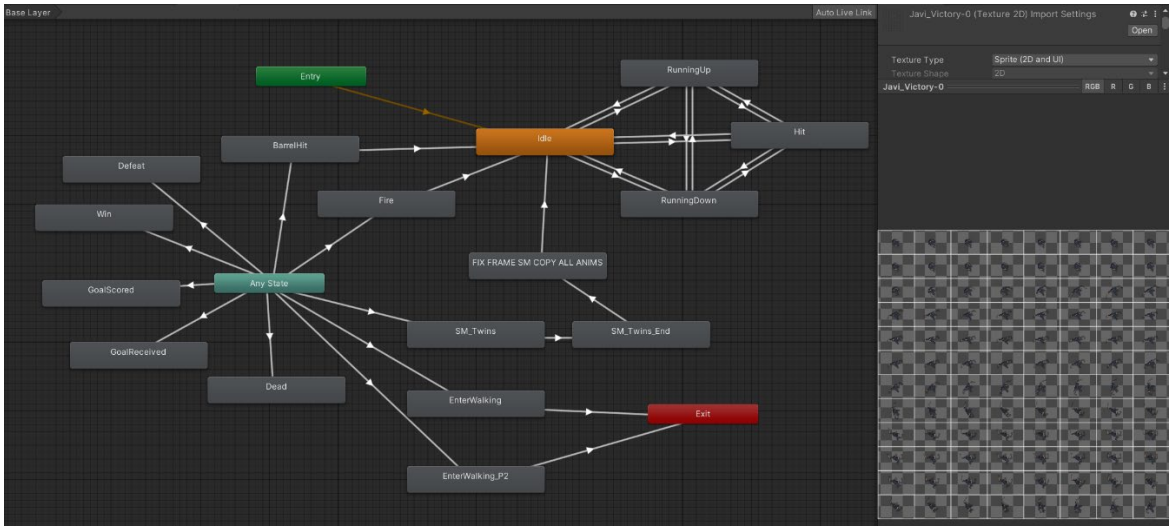


Figura 53*: Animator controller* y primer *Victory sprite* de Javi. Fuente: Elaboración propia

Los cambios que se hacen en Unity están asociados a un proyecto alojado en GitHub utilizando el sistema de control de versiones Git, permitiendo tener un histórico de cambios del proyecto.

También se hace uso de los *Issues* y *Projects* de GitHub, que permiten llevar un control de las tareas e incidencias y ver los cambios realizados en el proyecto asociadas a ellas. La Figura 54 muestra una captura de pantalla donde se ve, en la parte izquierda, las *Issues* abiertas actualmente en el proyecto y, en la parte derecha, el *backlog* del *Project* de PWP en GitHub:

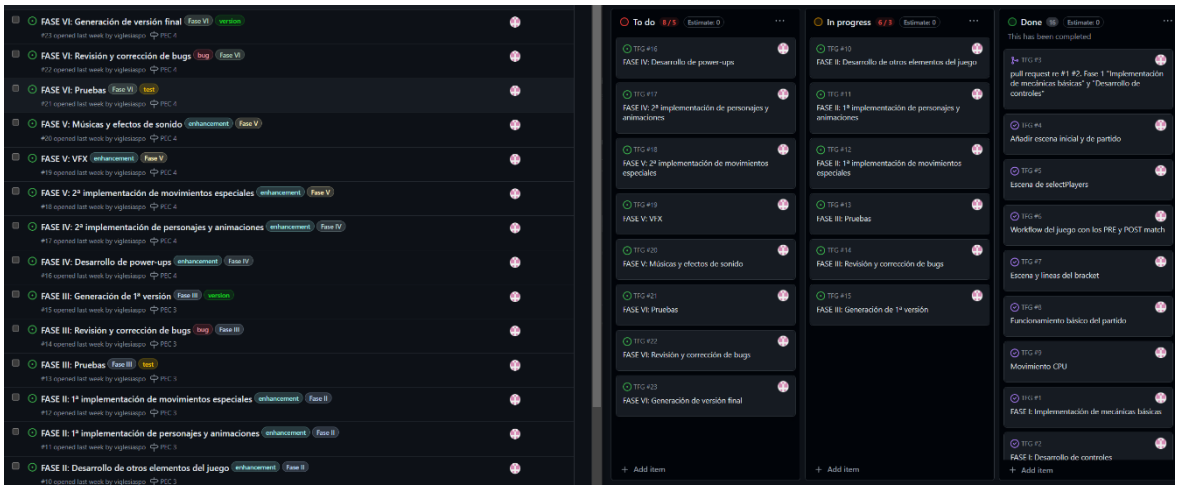


Figura 54*: Issues* y *Project* de PWP en GitHub. Fuente: Elaboración propia

4.2.2. Daz 3D y Face Transfer Ultimate

Daz Studio es el entorno de trabajo de Daz 3D que permite generar rápidamente humanoides en 3D, así como crear poses y animaciones. Junto con el plugin de Face Transfer Unlimited, podemos utilizar una fotografía de la cara de una persona para transferirla al modelo de Daz Studio. La Figura 55 muestra el modelo 3D de Yago en Daz Studio:

Figura 55*:* Modelo 3D del personaje Yago en Daz Studio*.* Fuente: Elaboración propia

Las animaciones faciales de los jugadores en la pantalla de selección de personajes están creadas también con el Daz Studio, ya que su editor permite modificar de manera sencilla los distintos partes del modelo 3D. En la Figura 56 se muestra un *frame* de una de las animaciones faciales de Yago en Daz Studio:

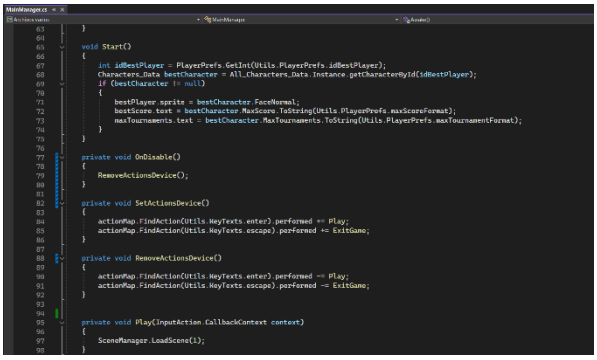
Figura 56*: Frame* de una animación de Yago en Daz Studio. Fuente: Elaboración propia

4.2.3. Visual Studio 2022

Visual Studio [18] es un entorno de desarrollo integrado compatible con múltiples lenguajes de programación que permite la creación aplicaciones para distintas plataformas. No es obligatorio el uso de Visual Studio: podría utilizarse cualquier otro entorno de desarrollo o editor de C#.

El uso del lenguaje de programación de C# en el proyecto es debido a la compatibilidad con Unity, que permite el control de los distintos aspectos relacionados con la programación del juego.

En la Figura 57 se muestra un ejemplo de un fichero en C# perteneciente a PWP en Visual Studio 2022:



```
    }

    void Start()
    {
        int idBestPlayer = PlayerPrefs.GetInt(Utils.PlayerPrefs.idBestPlayer);
        Characters_Data bestCharacter = All_Characters_Data.Instance.getCharacterById(idBestPlayer);
        if (bestCharacter != null)
        {
            bestPlayer.sprite = bestCharacter.FaceNormal;
            bestScore.text = bestCharacter.MaxScore.ToString(Utils.PlayerPrefs.maxScoreFormat);
            maxTournaments.text = bestCharacter.MaxTournaments.ToString(Utils.PlayerPrefs.maxTournamentFormat);
        }
    }

    private void OnDisable()
    {
        RemoveActionsDevice();
    }

    private void SetActionsDevice()
    {
        actionMap.FindAction(Utils.KeyTexts.enter).performed += Play;
        actionMap.FindAction(Utils.KeyTexts.escape).performed += ExitGame;
    }

    private void RemoveActionsDevice()
    {
        actionMap.FindAction(Utils.KeyTexts.enter).performed -= Play;
        actionMap.FindAction(Utils.KeyTexts.escape).performed -= ExitGame;
    }


    private void Play(InputAction.CallbackContext context)
    {
        SceneManager.LoadScene(1);
    }
```

Figura 57*:* C# en Visual Studio 2022. Fuente: Elaboración propia

4.2.4. Adobe Mixamo

La web de Adobe Mixamo nos permite generar el *rigging* de manera automática para el modelo 3D y cuenta con una biblioteca de animaciones para incorporar directamente a los modelos.

Esto permite crear animaciones de una manera rápida y sencilla para cada uno de los modelos de los jugadores y ser exportadas a un formato que es compatible con Unity o con Blender.

La Figura 58 muestra uno de los personajes del videojuego en la plataforma Mixamo para crear las distintas animaciones:

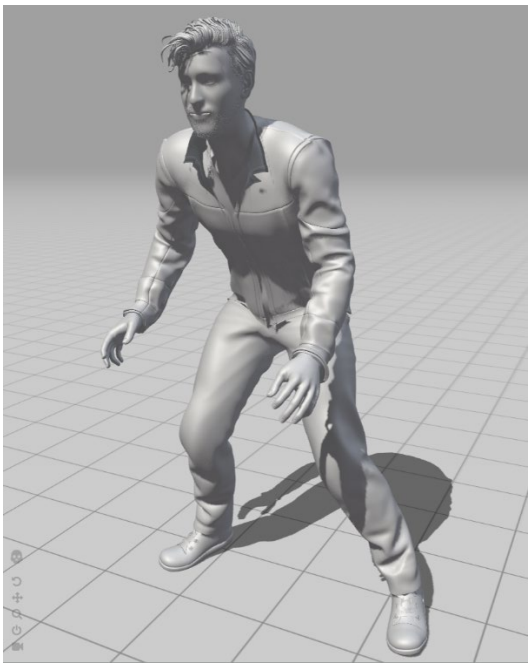


Figura 58*:* Modelo 3D de Yago en Mixamo. Fuente: Elaboración propia

4.2.5. Blender

Es un programa de código abierto y gratuito que permite el desarrollo de todo lo relacionado con el 3D: modelado, *rigging*, renderizado, iluminación, animación, etc.

En PWP, utilizamos Blender para hacer modificaciones en las animaciones descargadas de Mixamo y para las renderizaciones de los gráficos de los jugadores en el partido.

Hay ciertos movimientos especiales que requieren ser modificados, como por ejemplo la animación del movimiento especial de Miguel, que tiene una serie de *props* en las manos y enfrente de él, como muestra la Figura 59:

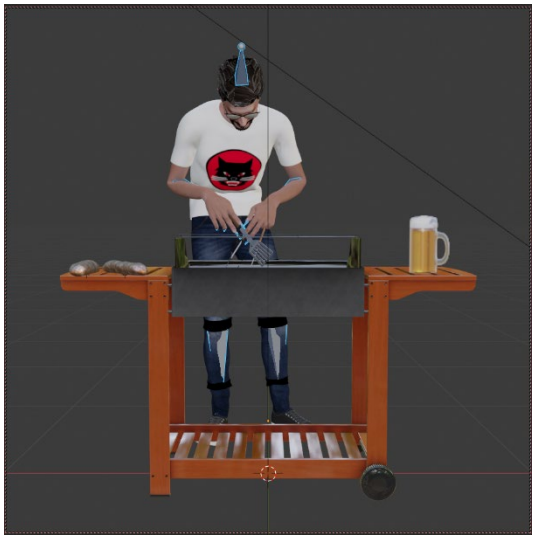


Figura 59*:* Animación de Miguel en Blender. Fuente: Elaboración propia

4.2.6. Texture Packer

Esta herramienta [19] permite la creación de para crear plantillas de *sprites* a partir de las imágenes que hemos generado con Daz Studio o Blender.

Puede usarse cualquier otra herramienta que cumpla el mismo objetivo: se le pasan las imágenes individuales y el programa se encarga de crear una imagen nueva que las une para ser utilizadas en Unity.

En la Figura 60 podemos ver un ejemplo del uso de Texture Packer con el movimiento del lanzamiento de objetos para uno de los jugadores:

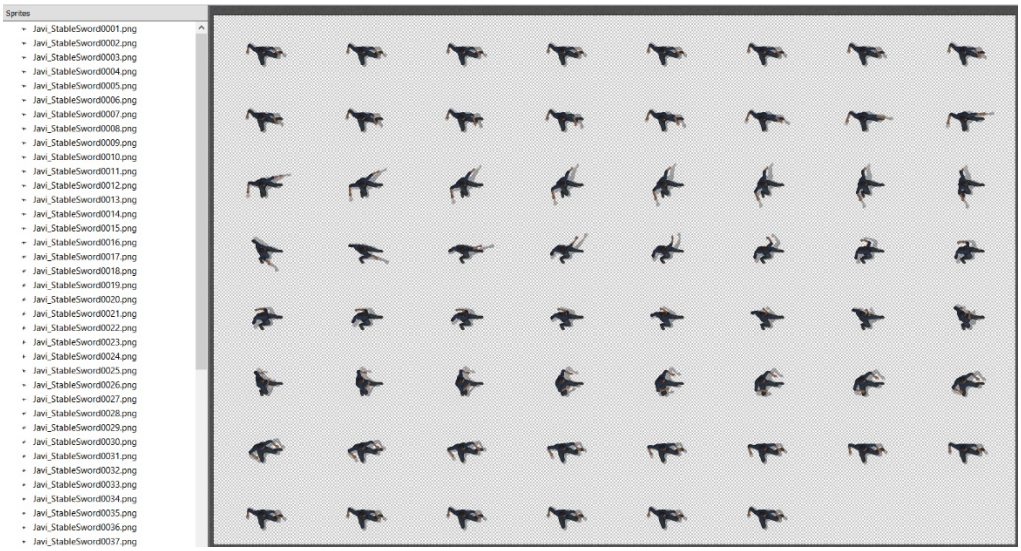


Figura 60*:* Creación de un *sprite* en Texture Packer. Fuente: Elaboración propia

4.2.7. Pngquant

Es una utilidad [20] de línea de comandos y una biblioteca para la compresión con pérdida de imágenes PNG, permitiendo reducir el tamaño de los archivos y mantener la transparencia alfa total.

Con esta utilidad, conseguimos reducir ampliamente el tamaño de las múltiples imágenes usadas en el juego, a cambio de perder un poco de calidad gráfica de las mismas.

En la Figura 61 vemos una comparación de un gráfico original y otro modificado con Pngquant. El gráfico de la izquierda ocupa 2,04 MB mientras que el de la derecha ocupa 148 kb:

Figura 61*:* Comparación de un png original con uno comprimido con Pngquant. Fuente: Elaboración propia

4.2.8. Advanced installer

Advanced Installer [21] es una herramienta para crear paquetes de instalación de software. Entre sus características destaca el soporte para distintas plataformas y formatos, integración con entornos y herramientas de desarrollo, así como la creación de instaladores personalizables.

Esta herramienta no es estrictamente necesaria en el proyecto, pero se hará uso de la misma en la versión final del juego, para facilitar la instalación de la aplicación. Como otras herramientas del proyecto, existen otras aplicaciones que cumplen con el mismo objetivo.

En la Figura 62 vemos un ejemplo de las plantillas disponibles en Advanced installer para la creación de distintos instaladores:

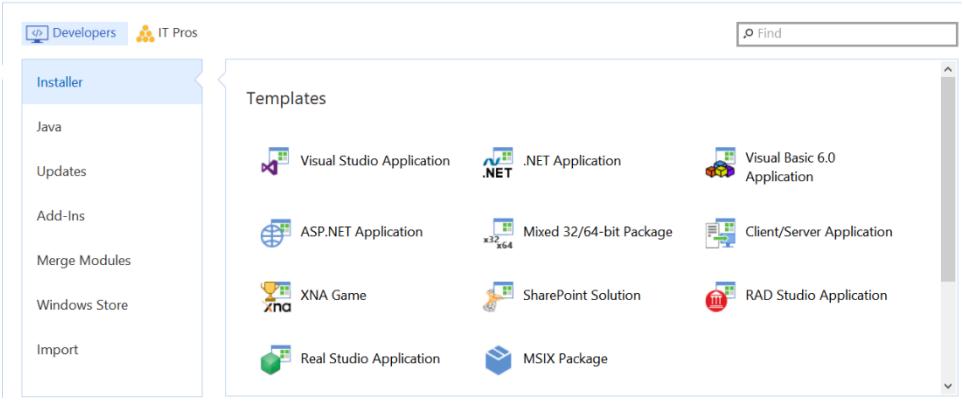


Figura 62: Ejemplo de plantillas en Advanced Installer. Fuente: Elaboración propia

4.3. Recursos

En este apartado se listan los principales medios utilizados en el desarrollo del proyecto.

4.3.1. Unity

- Materiales

  Este tipo de recurso es utilizado por el sistema de renderizado de Unity para determinar cómo se ven las superficies de los objetos en el juego.

  Como vemos en la Figura 63, los materiales siguientes son utilizados en el aura de los jugadores cuando ejecutan un movimiento especial, en el logo de la cerveza y el

barril, en el movimiento especial de la bomba de humo, para cambiar a escalas de grises ciertas imágenes y en el uso de los sistemas de partículas de las estrellas:

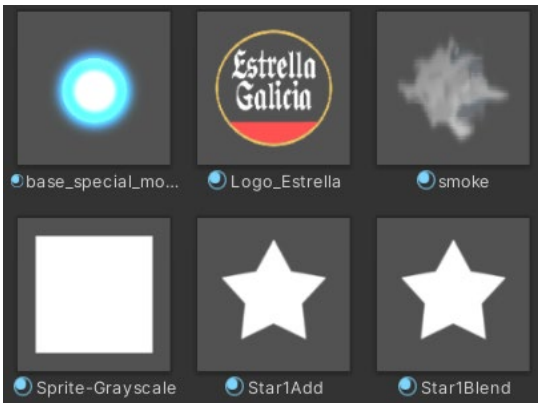


Figura 63*:* Materiales utilizados en PWP. Fuente: Elaboración propia

Existen otros materiales en el juego, pero están integrados en otros recursos, como por ejemplo los sistemas de partículas específicos que veremos más adelante.

- *Shaders*

  Estos scripts contienen los algoritmos necesarios para calcular el color de cada píxel renderizado, basándose en la iluminación y la configuración del material.

  En este caso, se hace uso del *shader* Sprite-Grayscale que a su vez hace uso del material Sprite-Grayscale, utilizado en el juego para pasar a escala de grises ciertas imágenes, como las caras de los jugadores que no están desarrollados o cuando se pierde un partido.

Figura 64*: Shader* utilizados en PWP. Fuente: Elaboración propia

Igual que ocurría con los materiales, los *shaders* también se encuentran integrados en otros recursos.

- Sistemas de partículas

Los sistemas de partículas son una herramienta que permite a los desarrolladores simular y representar diversos efectos visuales, como líquidos en movimiento, humo, nubes, llamas, etc.

Clasificaremos los sistemas de partículas utilizados en PWP por escenas:

1. **Menú principal**:

   Sistema de partículas con estrellas sobre la cara del mejor jugador, como muestra la Figura 65:

Figura 65*:* Sistema de partículas de estrellas en PWP. Fuente: Elaboración propia

   Los siguientes sistemas de partículas (excepto los indicados) que vamos a ver en el resto de escenas pertenecen al *asset* gratuito de Jean Moreno “Cartoon FX Remaster” disponible en la AssetStore:

   https://assetstore.unity.com/packages/vfx/particles/cartoon-fx-remaster-free-109565

2. **Selección de jugadores**:

   Sistema de partículas de explosión de humo (CFX3_Hit_SmokePuff) al cambiar la selección de jugador, como muestra la Figura 66:

Figura 66*:* Sistema de partículas de explosión de humo en PWP. Fuente: Elaboración propia

   Sistema de partículas de brasas (CFX3_Flying_Ember_Upward) que están en el fondo de la escena, como muestra la Figura 67:

Figura 67*:* Sistema de partículas de brasas en PWP. Fuente: Elaboración propia

3. **Partido**:

Sistema de partículas de mini explosión (CFX4 Sparks Explosion B) cuando un jugador recibe el impacto de un objeto o cuando chocan dos objetos, como muestra la Figura 68:

Figura 68*:* Sistema de partículas de mini explosión en PWP. Fuente: Elaboración propia

Sistema de partículas de golpeo en el aire (CFX4 Drill Air Hit (NO COLLISION)) cuando un jugador recibe el impacto del barril de cerveza, como muestra la Figura 69:

Figura 69*:* Sistema de partículas de golpeo en el aire en PWP. Fuente: Elaboración propia

Sistema de partículas de aturdimiento (CFX3_Skull_Explosion) cuando un jugador está aturdido por el impacto del barril de cerveza, como muestra la Figura 70:

Figura 70*:* Sistema de partículas de aturdimiento en PWP. Fuente: Elaboración propia

Sistema de partículas de aura (CFX3_MagicAura_D_Runic) cuando un jugador ha ejecutado su movimiento especial, como muestra la Figura 71:

Figura 71*:* Sistema de partículas de aura en PWP. Fuente: Elaboración propia

Sistema de partículas de aura de *power-ups* (CFX4-Aura-Bubble-C) cuando un power-up aparece en escena, como muestra la Figura 72:

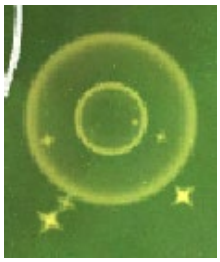


Figura 72*:* Sistema de partículas de aura de *power-ups* en PWP. Fuente: Elaboración propia

Sistema de partículas de diamante (CFX2_PickupDiamond2) cuando aparece el *power-up* de energía especial, como muestra la Figura 73:

Figura 73*:* Sistema de partículas de diamante en PWP. Fuente: Elaboración propia

Sistema de partículas de corazón (CFX2_PickupHeart) cuando aparece el power-up de vida, como muestra la Figura 74:

Figura 74*:* Sistema de partículas de salud en PWP. Fuente: Elaboración propia

Sistema de partículas de bomba de humo (CFX_Explosion_B_Smoke+Text) cuando se ejecuta el movimiento especial de la bomba de humo, como muestra la Figura 75:

Figura 75*:* Sistema de partículas de bomba de humo en PWP. Fuente: Elaboración propia

Sistema de partículas de humo (Smoke_Thick_Ground) cuando está activo el movimiento especial de la bomba de humo, como muestra la Figura 76:

Figura 76*:* Sistema de partículas de humo en PWP. Fuente: Elaboración propia

Este sistema de partículas pertenece al *asset* gratuito de Onpolyx “Flames of the Phoenix”:

https://www.gameassetdeals.com/asset/46176/flames-of-the-phoenix

Sistema de partículas de portales (CFX4 Fairy Dust) cuando está activo el movimiento especial de la manada de jabalíes, como muestra la Figura 77:

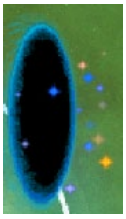


Figura 77*:* Sistema de partículas de portales en PWP. Fuente: Elaboración propia

Sistema de partículas de humareda (CFX4 Smoke Trail A (Curved, Black)) cuando está dañado el coche de policía, como muestra la Figura 78:

Figura 78*:* Sistema de partículas de humareda en PWP. Fuente: Elaboración propia

Sistema de partículas de explosión (CFX4 Explosion SoftEdge Air) cuando explota el coche de policía, como muestra la Figura 79:

Figura 79*:* Sistema de partículas de explosión en PWP. Fuente: Elaboración propia

Sistema de partículas de humo de pira (Smoke) y de chispas (Flare) cuando está activo el movimiento especial de la barbacoa, como muestra la Figura 80:

Figura 80*:* Sistema de partículas de barbacoa en PWP. Fuente: Elaboración propia

Este sistema de partículas pertenece al *asset* gratuito de Unity “Standard Assets”:

https://assetstore.unity.com/packages/essentials/asset-packs/standard-assets-2018-4-check-out-starter-assets-first-person-thi-32351

4.3.2. Daz Studio

- *Assets y Props*

  Los distintos modelos de jugadores y escenarios hacen uso de *assets* y *props* específicos de Daz Studio. La Figura 81 muestra un listado con aquellos utilizados actualmente:

Figura 81: *Assets* y *props* de Daz Studio utilizados en PWP. Fuente: Elaboración propia

4.3.3. Gráficos 2D

- Propios

Los gráficos realizados por el autor del proyecto son aquellos relativos a las caras de los personajes, ya que han sido modelados con la aplicación de Daz Studio y el plugin de Face Transfer Ultimate, tal y como muestra la figura 82:

Figura 82: Caras de los protagonistas de PWP . Fuente: Elaboración propia.

También otros gráficos como los marcos de la selección de jugadores y cuadro del torneo, como muestra la Figura 83:

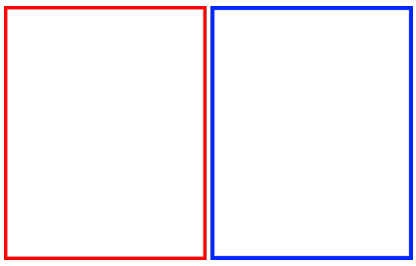


Figura 83: Marcos utilizados en PWP . Fuente: Elaboración propia.

O las flechas de selección que muestra la Figura 84:

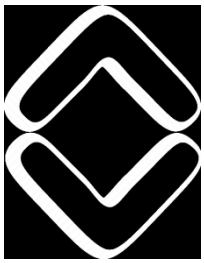


Figura 84: Flechas de selección utilizados en PWP . Fuente: Elaboración propia.

- Externos

  Si bien hay gráficos de los *assets* y *props* de Daz Studio que se han utilizado de manera independiente en algún momento del juego, a continuación se muestra un listado con los distintos gráficos en 2D que han sido recopilados de distintas fuentes de Internet:

La Figura 85 muestra la imagen de Goblet.png:

Figura 85*:* Imagen de Goblet.png. Fuente: opengameart

La Figura 86 muestra la imagen de la animación del barril:

Figura 86*:* Imagen de Goblet.png. Fuente: Pixabay

La Figura 87 muestra la imagen del logo de la Estrella Galicia:

Figura 87*:* Imagen de Goblet.png. Fuente: Seeklogo

La Figura 88 muestra la imagen de la copa de la escena del cuadro del torneo:

Figura 88*:* Imagen de la copa. Fuente: Supercoloring

La Figura 89 muestra la imagen del mando de XBox:

Figura 89*:* Imagen del mando. Fuente: Hiclipart

La Figura 90 muestra la imagen del mando de XBox:

Figura 90: Imagen del mando. Fuente: Hiclipart

La Figura 91 muestra la imagen del botón A del mando:

Figura 91: Imagen del botón A del mando. Fuente: Wikipedia

La Figura 92 muestra la imagen del botón B del mando:

Figura 92: Imagen del botón B del mando. Fuente: Wikipedia

La Figura 93 muestra la imagen del logo del Santa Cruz C.F.:

Figura 93: Escudo del equipo de Santa Cruz C.F. Fuente: SantaCruzCF

La Figura 94 muestra la imagen de las estrellas:

Figura 94: Imagen de estrellas. Fuente: Adobe

La Figura 95 muestra la imagen de la pelota de tenis temporal:

Figura 95*:* Imagen de la pelota temporal. Fuente: Spriters-resource

La Figura 96 muestra el fondo de los movimientos especiales:

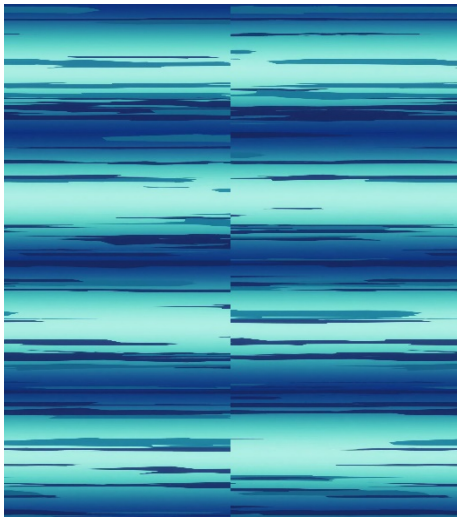


Figura 96*:* Fondo de los movimientos especiales. Fuente: Youtube

La Figura 97 muestra las imágenes GIF usados en las pantallas del partido:

Figura 97*:* Gifs utilizados en las pantallas del partido. Fuente: Giphy

4.3.4. Fuentes

- TextMesh Pro

  Es una herramienta de Unity para la creación de texto en 2D y 3D, que utiliza técnicas de renderizado de texto junto con un conjunto de *shaders* personalizados.

  Actualmente y mientras no se finalice la versión final del juego, se utiliza en la mayoría de los textos del juego, como por ejemplo en el texto de “Jugar” del menú principal, tal y como se indica en la figura 98:

Figura 98*:* Texto de TextMesh Pro en PWP. Fuente: Elaboración propia

  Las fuentes utilizadas por TextMesh Pro en el proyecto son Akzidenz-grotesk-light (Fuente: Dafont) y Typodermic - OctinSportsRg-Regular SDF (Fuente: Dafont), pero a mayores utilizamos la fuente Score Board para el marcador del partido, tal y como muestra la Figura 99:

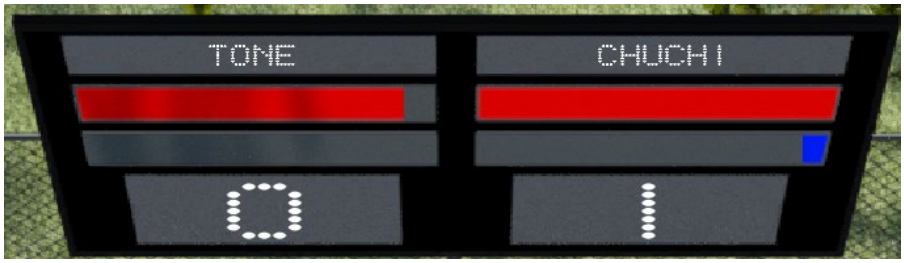


Figura 99*:* Fuente Score Board con TextMesh Pro. Fuente: Dafont

4.3.5. Música y efectos de sonido

En la versión actual del juego (PEC 3) no están incluidas ni músicas ni efectos de sonido. Tal y como se detalló en el diagrama de Gantt visto anteriormente, la planificación de esta tarea se realizaría en la PEC 4, como muestra la Figura 100:

PEC 4 - Memoria y productos finales
FASE IV
Desarrollo de power-ups
2ª implementación de personajes y animaciones
FASE V
2ª implementación de movimientos especiales
VFX
Musicas y efectos de sonido

Figura 100*:* Detalle del diagrama de Gantt con la planificación de músicas y sonidos. Fuente: Elaboración propia

4.4. Arte

A continuación, se muestra el proceso creativo y técnico que ha dado lugar a la apariencia visual del juego, incluyendo las influencias y las decisiones de diseño que llevaron a este estilo.

4.4.1. Escenas

- Menú principal

  Esta escena no está finalizada todavía, la intención es que sea una pantalla sencilla en el que se muestre el logo del juego, el mejor jugador hasta el momento y los botones para jugar y salir.

  La Figura 101 muestra una comparativa del boceto inicial (izquierda) y la pantalla en la actualidad (derecha):

Figura 101*:* Comparativa entre boceto y menú principal actual. Fuente: Elaboración propia

- Selección de jugadores

  Escena prácticamente finalizada, como se puede ver en la comparativa de la Figura 102. Se modificará la fuente escogida, se activarán los jugadores deshabilitados y quizás se realicen cambios menores.

Figura 102*:* Comparativa entre boceto y selección de jugadores actual. Fuente: Elaboración propia

- Cuadro del torneo

  Otra escena por finalizar con cambios menores, tal y como vemos en la Figura 103, la idea es mostrar los resultados de los encuentros, indicar la fase del torneo en la que se encuentra, mostrar el logo del juego, cambiar la fuente y la imagen del fondo.

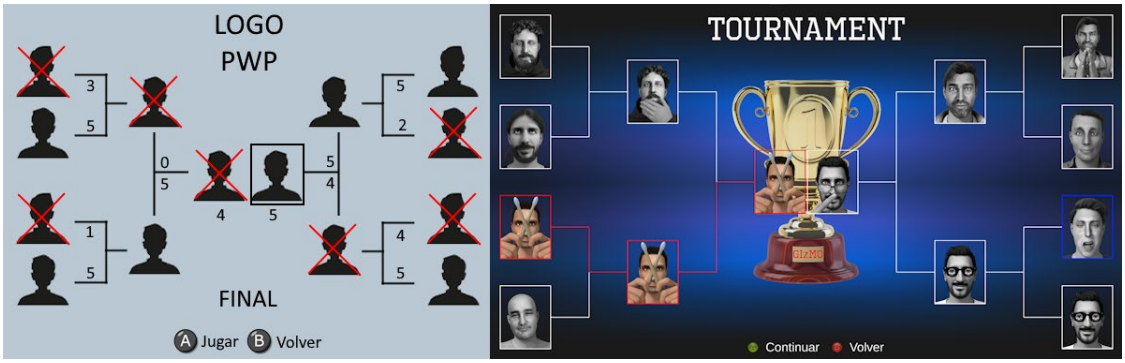


Figura 103*:* Comparativa entre boceto y cuadro del torneo actual. Fuente: Elaboración propia

- Prepartido

  En esta escena faltan cambios estéticos, tal y como vemos en la Figura 104, como los *renders* de los jugadores de cuerpo completo, los iconos del teclado y cambiar la fuente.

Figura 104*:* Comparativa entre boceto y prepartido actual. Fuente: Elaboración propia

- Pantalla de carga

  Esta escena directamente no está implementada. En la Figura 105 se muestra como debería de verse, pero es uno de los cambios a implantar para la versión final de la aplicación.

Figura 105: Boceto de pantalla de carga. Fuente: Elaboración propia

- Partido

  Esta escena está bastante completa a nivel estético, tal y como vemos en la Figura 106, faltan cambios funcionales, implementación varios jugadores, cambio de fuentes, etc. Pero la lógica general del partido está casi completamente implementada.

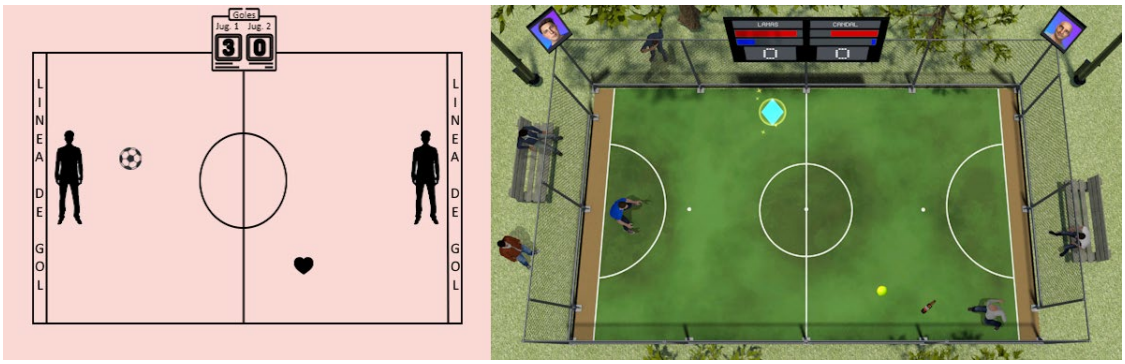


Figura 106: Comparativa entre boceto y partido actual. Fuente: Elaboración propia

- Pospartido

  Escena con una situación parecida a la de prepartido, tal y como vemos en la Figura 107, y a la que le faltan cambios similares: los *renders* de los jugadores de cuerpo completo, cambiar la fuente y modificaciones estéticas en la frase de la victoria.

Figura 107*:* Comparativa entre boceto y pospartido actual. Fuente: Elaboración propia

- Entrega de premios

  Igual que ocurría con la escena de la pantalla de carga, esta escena todavía no está implementada. En la Figura 108 se muestra como debería de verse, pero es uno de los cambios a implantar para la versión final de la aplicación.

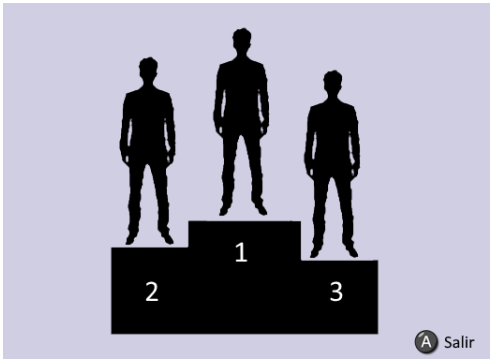


Figura 108*:* Boceto de entrega de premios. Fuente: Elaboración propia

4.4.2. Personajes

Uno de los objetivos principales de este videojuego era tratar de replicar fielmente a los amigos del autor y lograr así una conexión emocional con dicho público.

Gracias al proceso explicado anteriormente de generación de modelos con Daz Studio + Face Transfer Ultimate se consigue de un alto grado de similitud entre los personajes del juego con el público objetivo. Y con el apoyo de Mixamo y Blender, se logran también unas animaciones fluidas para aportar naturalidad a los movimientos de los personajes.

En la Figura 109 se muestran los personajes protagonistas del juego:

Figura 109: Personajes protagonistas de PWP. Fuente: Elaboración propia

Y en la Figura 110, se muestra el primer plano de cada una de las caras sobre las que el jugador interactúa para seleccionar a su personaje:

Figura 110: Primer plano de las caras de los personajes de PWP. Fuente: Elaboración propia

4.4.3. Objetos y *power-ups*

Otros elementos destacables en el desarrollo del partido son los objetos con los que los personajes interactúan durante el partido.

En la Figura 111 se muestra el objeto que lanzan los jugadores, un botellín, con la intención de impactar al contrario y así conseguir reducirle la salud a la vez que recarga su barra de energía especial.

Figura 111*:* Animación del botellín al no alcanzar al contrario y romperse. Fuente: Elaboración propia

Los *power-ups* son un tipo especial de objeto que aparecerán aleatoriamente en el terreno de juego y que, cuando la pelota pase por encima de ellos, se activarán desencadenando alguna acción en el partido.

En la actual versión de la aplicación se han implementado los dos 2 *power-ups* siguientes:

- **Vida**: Recarga parcialmente la barra de vida del jugador que lo haya activado con la pelota.

- **Energía especial**: Recarga parcialmente la barra de energía especial del jugador que lo haya activado con la pelota.

En la Figura 112 vemos el power-up de Vida y en la Figura 113 el power-up de energía especial:

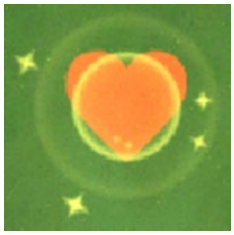


Figura 112*: Power-up* de vida. Fuente: Elaboración propia

Figura 113*: Power-up* de energía especial. Fuente: Elaboración propia

En la versión final del juego se implementarán los siguientes *power-ups*:

- **Escudo**: Protegerá durante 5 impactos de objetos al jugador que lo haya activado con la pelota.

- **Nieve**: Desencadenará un efecto de nieve sobre el terreno de juego, haciendo que los jugadores resbalen un poco al moverse.

- **Luna**: Reducirá la visión de los jugadores al oscurecer la luz del partido.

- **Sol**: Solo aparecerá cuando se haya activado el *power-up* de la luna y, al activarse, devolverá al estado de luz anterior a la activación del *power-up* de la luna.

4.5. Arquitectura de software

La arquitectura de software define de manera abstracta los componentes, interfaces y la comunicación entre ellos, proporcionando solución a problemas de manera eficiente manteniendo determinados atributos de calidad.

Para poder mostrar un esquema de la arquitectura, es necesario comprender la idea principal del diseño del juego: tener separados la parte lógica de los datos mediante una serie de clases ScriptableObject y Singleton que nos permita acceder a sus métodos desde cualquier parte mediante <NombreDeClase>.Instance.<metodo>.

En ella, sólo tendremos la lógica de aquellas funciones que no puedan ser implementadas en los managers clásicos Monobehaviour (porque no son accesibles desde cualquier parte de la aplicación).

El uso del patrón de diseño Singleton en Unity nos ofrece una serie de ventajas, como asegurarnos de tener solamente una instancia de una clase en la ejecución del juego, el acceso a esa instancia desde cualquier otra clase, persistir la clase entre escenas junto con el uso de DontDestroyOnLoad, etc.

Por otro lado, el uso de los ScriptableObjects también nos aporta otras interesantes ventajas, como almacenar grandes cantidades de datos independientes, persistir la clase a través de las escenas, modularidad separando los datos de la lógica y con acceso directo a través de los menús de Unity, sin tener que entrar en el código directamente.

En la Figura 114 podemos ver el uso de ScriptableObject para el jugador Yago:

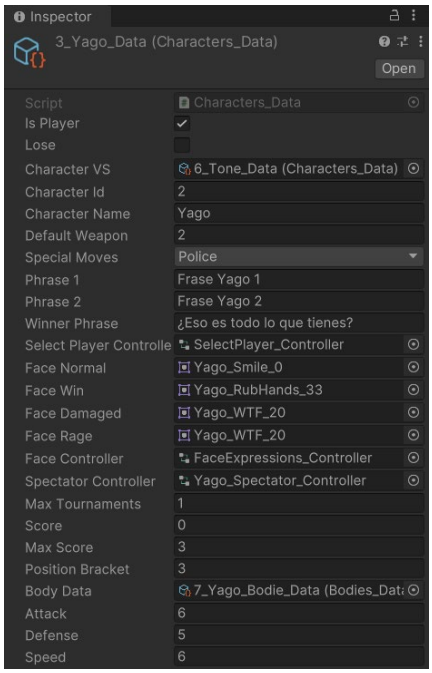


Figura 114: Datos de Yago en Unity. Fuente: Elaboración propia

Esta lógica de uso de ScriptableObject y Singleton en una misma clase está basado en este código de dvddarias en GitHub:
https://gist.github.com/dvddarias/7af26f6588bc61dfb43682cbc7ed889e
Que permite principalmente el uso de una clase Singleton que mantenga los datos más allá de la fase del tiempo de ejecución.

Finalmente se tienen los distintos managers clásicos Monobehaviour (como el MainManager.cs por ejemplo) que su ciclo de vida sólo existe en la escena a la que esté asociada y con las funcionalidades propias de la escena.

Con este diseño, además de las ventajas indicadas, proporcionan otras a la hora de ejecutar y probar el juego:

- Al depurar, se pueden ver los valores de las variables de los ScriptableObjects.
- Al tener archivos *asset*, mantiene los datos modificados a pesar de que pare la ejecución en Unity y se pueden modificar los valores de velocidad, ataque, defensa, frase de victoria, etc. directamente desde el editor.
- Se pueden tener varios archivos de configuración y cambiarlos rápidamente sin tocar código.
- Se pueden ejecutar directamente la escena del partido sin pasar por todo el flujo de pantallas.

En la Figura 115 se muestra una captura de la estructura actual de los archivos de scripts, donde se puede ver como principalmente están ordenados por escenas y subcarpetas pertenecientes al mismo ámbito:

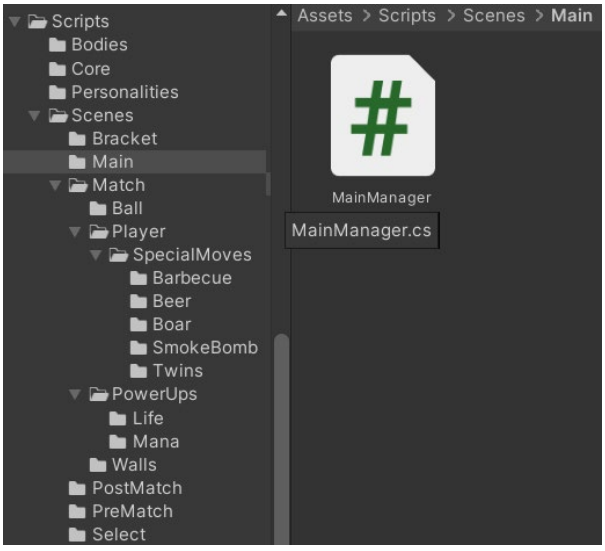


Figura 115*:* Estructura de carpetas de scripts en PWP. Fuente: Elaboración propia

4.5.1. Diagrama

Ahora que se tiene una visión global del diseño del juego, se puede representar mediante el diagrama que se muestra en la Figura 116:

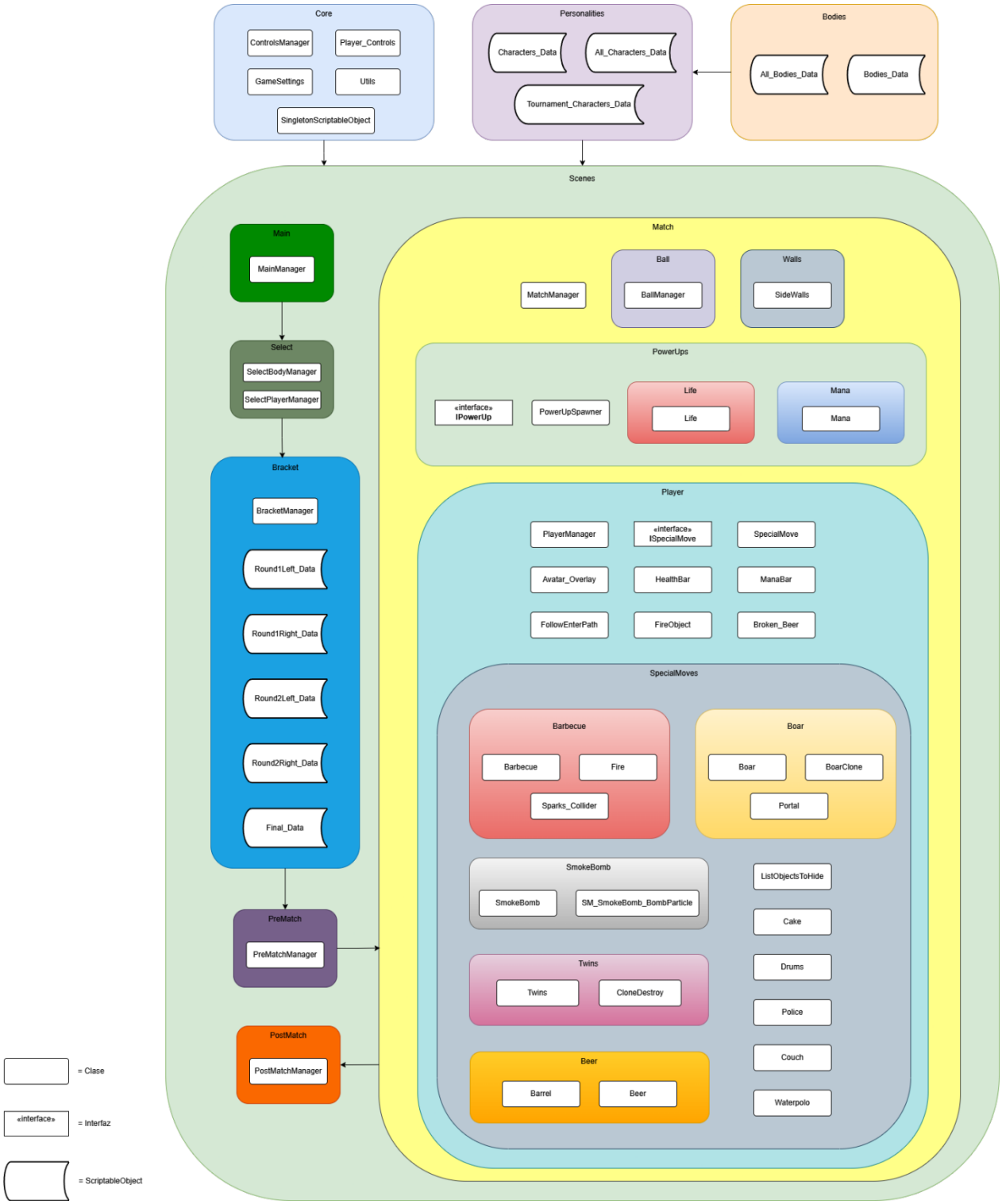


Figura 116: Diagrama actual de la arquitectura de software de PWP. Fuente: Elaboración propia

5. Implementación

Unity ofrece un entorno de desarrollo que permite el diseño y la programación del videojuego de manera integrada haciendo uso de distintos scripts. Dichos scripts permiten manipular los elementos del juego con facilidad, utilizando distintos lenguajes de programación para escribir comportamientos personalizados.

Ofrecen una gran flexibilidad y control sobre el juego, y gracias a la integración con el editor de Unity, los cambios pueden verse en tiempo real. Además, el acceso a las APIs de Unity y la compatibilidad con *assets* de terceros amplían enormemente las posibilidades de desarrollo.

5.1. Corrutinas

Una corrutina es una función especial que permite pausar su ejecución y reanudarla en el siguiente frame o después de un tiempo determinado, sin bloquear el resto del código.

Esto es útil para crear secuencias de eventos, animaciones, o comportamientos que se extienden a lo largo del tiempo, como esperas o transiciones, sin interrumpir el flujo principal del juego.

Las corrutinas son una parte esencial de la programación en Unity para gestionar tareas asíncronas y temporizadas de manera eficiente.

5.2. Bucle del partido

Uno de las principales características de un videojuego arcade es poder controlar de una manera sencilla el ciclo de vida de una partida: tiene que ejecutarse continuamente hasta llegar a un estado en el que se altere el propio bucle.

En PWP se ha optado por trabajar con corrutinas dividiendo el bucle del partido en 3 fases: Inicio, juego y final. En la clase gestora del partido (MatchManager) se tiene la corrutina GameLoop que es la encargada de realizar dicho proceso.

La corrutina GameLoop es un ciclo de juego que se repite y consta de las fases indicadas, donde cada fase se ejecuta secuencialmente, esperando a que la anterior finalice antes de comenzar la siguiente.

La fase de inicio es la correspondiente a la entrada de los jugadores en el terreno de juego, el inicio de las pantallas de video de los laterales de la pista, etc. tal y como se muestra en la Figura 117.

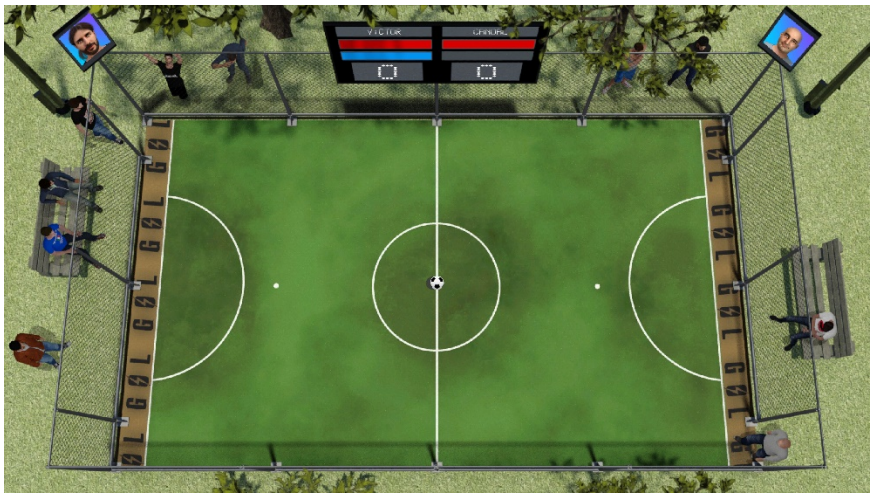


Figura 117*:* Entrada de los jugadores al inicio del partido. Fuente: Elaboración propia

Una vez que se han hecho los preparativos para el inicio, da comienzo la fase principal en la que discurre el partido: los jugadores golpean la pelota, se lanzan objetos, ejecutan movimientos especiales, etc.

Cuando se produce un gol, se llama a la fase final, donde se desarrollan las animaciones de los personajes celebrando el gol, el cambio en el marcador, etc. y donde se evalúa si alguno de los jugadores ha alcanzado los goles necesarios para ganar el partido.

En ese caso, se ejecutan acciones adicionales como reproducir un sonido de victoria, guardar las puntuaciones, etc. y se efectúa el cambio de pantalla correspondiente.

Si no es así, la corrutina GameLoop se llama a sí misma para comenzar una nueva ronda (esta vez evitando la fase de inicio porque sólo sucede una vez durante el partido), manteniendo el ciclo de juego activo hasta que se determine un ganador.

Este tipo de implementación, con una estructura modular de las fases, permite una fácil adaptación y mantenimiento del código, así como la posibilidad de expandir o modificar el comportamiento del juego en el futuro.

5.3. Esquivar a los jabalíes

El movimiento especial de Lamas es la “Manada de jabalíes”, donde el jugador en cuestión invoca progresivamente a una serie de jabalíes que tratan de chocar sobre el jugador contrario para restarle vida, tal y como muestra la Figura 118:

Figura 118: Lamas invocando a su manada de jabalíes. Fuente: Elaboración propia

En el caso de que el contrario sea un jugador controlado por un humano, depende de su habilidad para poder esquivarlos y evitar que su barra de salud se vea reducida.

Pero en el caso de que el contrario sea manejado por el ordenador, se tuvo que implementar un algoritmo para que el adversario tratara de esquivar a los jabalíes automáticamente. El problema era que el algoritmo era muy eficiente y el jugador controlado por el ordenador siempre esquivaba a los jabalíes, lo que restaba diversión cada vez que se utilizaba dicho movimiento especial.

Así que se tuvo que desarrollar un algoritmo que encontrara un equilibrio entre que los jabalíes golpearan al jugador y que los pudiera esquivar. Esto es lo que hace la función AvoidBoars de la clase PlayerManager.

La lógica del algoritmo se divide en los pasos siguientes:

- Obtiene una referencia a la lista de jabalíes existentes en el terreno de juego.
- Recorre la lista y calcular la posición de cada jabalí.
- Calcula la posición del jugador controlado por el ordenador.
- Si el jabalí ha superado el centro del campo y está en el mismo eje horizontal con el contrario, entonces hay una colisión potencial con el jugador:
    - Si el jugador no está chocando contra la pared superior y está configurado para moverse hacia arriba para evitar jabalíes, el jugador se desplaza hacia arriba.
    - Si no es así, se comprueba si el jugador está chocando contra la pared inferior y si está configurado para moverse hacia abajo, entonces el jugador se desplaza hacia abajo.
    - Si ninguna de las condiciones anteriores se cumple, se cambia la dirección de evasión en la próxima comprobación.

Con este algoritmo el jugador controlado por el ordenador esquivará algunas veces a los jabalíes pero otras veces no, haciendo que el uso de este movimiento especial sea más satisfactorio.

6. Manual de usuario

En este apartado se detallaran los pasos para realizar la instalación del videojuego, que controles pueden usarse y cómo es la jugabilidad de PWP.

6.1. Instalación

Para la instalación del videojuego, es recomendable seguir los siguientes pasos:

- Ejecutar el archivo PWP_Instalador.exe
- Hacer clic en Siguiente en la pantalla de la Figura 119:

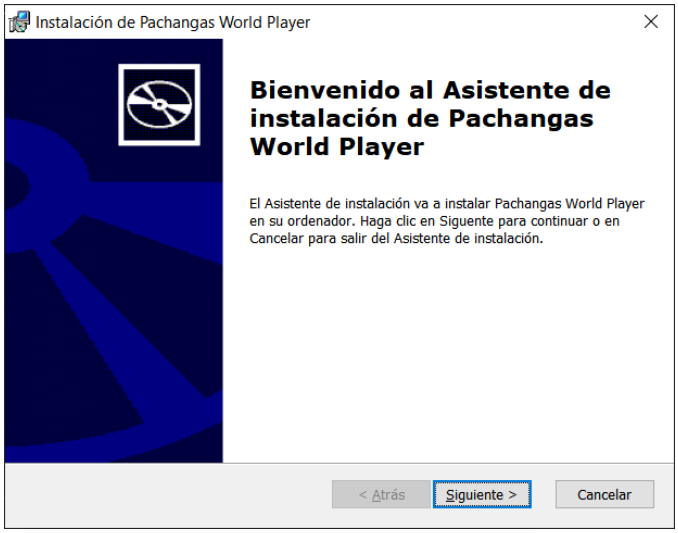


Figura 119: Pantalla inicial de instalación de PWP. Fuente: Elaboración propia

- Seleccionar la ruta de instalación y hacer clic en Siguiente como se muestra en la Figura 120:

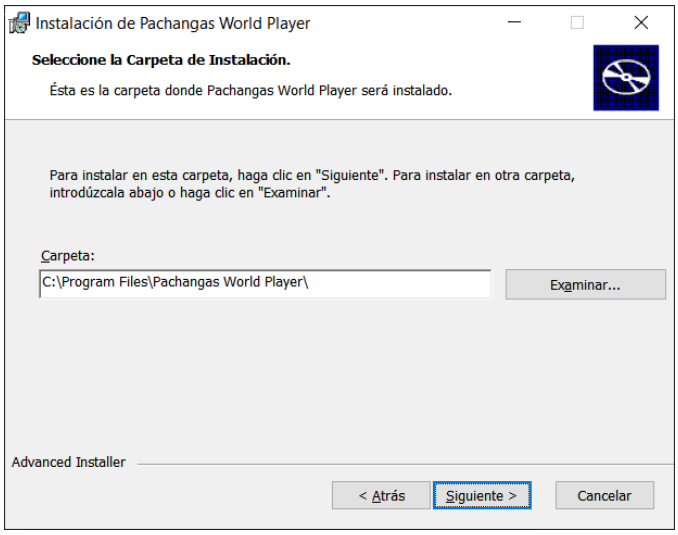


Figura 120: Pantalla de selección de directorio de instalación. Fuente: Elaboración propia

- Hacer clic en Instalar tal y como se indica en la Figura 121:

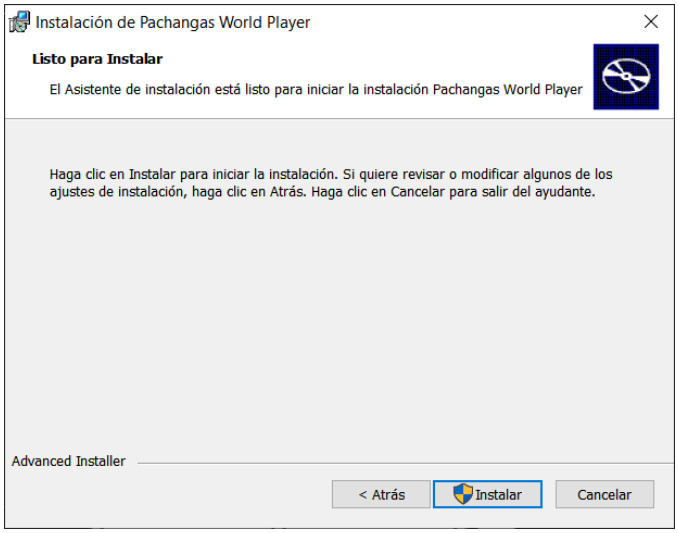


Figura 121*:* Pantalla previa a la instalación de PWP. Fuente: Elaboración propia

- Una vez finalice la instalación, hacemos clic en el botón de Finalizar que muestra la Figura 122:

Figura 122*:* Pantalla previa a la instalación de PWP. Fuente: Elaboración propia

- El instalador habrá creado un acceso directo en el escritorio (PWP_Launcher), hacemos doble clic sobre él y se nos mostrará una pantalla de carga inicial como se muestra en la Figura 123 y, a continuación, se lanzará el juego:

Figura 123*:* Pantalla de carga inicial de PWP. Fuente: Elaboración propia

6.2. Controles

En PWP se puede utilizar tanto teclado como mando donde las teclas/botones a utilizar están definidos de la siguiente manera:

6.2.1. Teclado

En las pantallas del juego distintas a la del partido, se navegará por los menús con las teclas:

- W o Flecha arriba: Arriba
- A o Flecha izquierda: Izquierda
- S o Flecha abajo: Abajo
- D o Flecha derecha: Derecha
- ENTER: Confirmar
- ESC: Volver

Mientras que en la pantalla del partido se utilizaran:

- W: Arriba
- S: Abajo
- G: Lanzar objeto (disparar)
- H: Activar movimiento especial
- ENTER: Confirmar

- ESC: Volver

En caso de que se estén enfrentando 2 jugadores controlados por humanos, el segundo jugador utilizará las teclas:

- Flecha arriba: Arriba
- Flecha abajo: Abajo
- L: Lanzar objeto (disparar)
- K: Activar movimiento especial
- ENTER: Confirmar
- ESC: Volver

En la Figura 124 se muestra la pantalla de carga previa al partido, donde se indica el recordatorio de las teclas a utilizar cuando se enfrentan 2 jugadores con teclado:

Figura 124*:* Pantalla de carga con controles para 2 jugadores. Fuente: Elaboración propia

6.2.2. Mando

En el caso anterior se observaba que con un mismo teclado podían jugar 2 jugadores simultáneamente. En el caso de disponer de un mando, cuando se enfrentan 2 jugadores, se utilizará o bien un mando y un teclado o bien 2 mandos. Por lo tanto la configuración de botones será la siguiente para cualquier mando:

- Botón de dirección Arriba o Joystick izquierdo Arriba: Arriba
- Botón de dirección Izquierda o Joystick izquierdo Izquierda: Izquierda
- Botón de dirección Abajo o Joystick izquierdo Abajo: Abajo
- Botón de dirección Derecha o Joystick izquierdo Derecha: Derecha
- A: Lanzar objeto (disparar) y Confirmar en menú
- B: Activar movimiento especial y Volver en menú
- Y: Ver otro controlador en enfrentamiento de 2 jugadores
- Joystick derecho: Ángulo de golpeo de pelota (apuntar)

En la Figura 125 se muestra la pantalla de carga previa al partido, donde se indica el recordatorio de los botones del mando a utilizar:

Figura 125: Pantalla de carga con controles para mando. Fuente: Elaboración propia

6.3. Jugabilidad

El partido se inicia con los jugadores posicionados en lados opuestos del campo y cada uno dispone de una barra de vida y energía especial que se reduce al recibir daño o usar el movimiento especial.

Los jugadores golpean automáticamente la pelota, sin necesidad de pulsar ningún botón, facilitando la participación a quienes no están familiarizados con los videojuegos, ya que uno de los objetivos de PWP tiene como fin lograr que cualquier persona disfrute del producto sin importar sus habilidades para con los videojuegos.

La velocidad de la pelota varía según la dificultad elegida en el menú principal, incrementándose progresivamente con cada golpe y reiniciándose tras un gol. Los jugadores

pueden apuntar la dirección del golpeo antes de impactar la pelota, usando el joystick derecho.

Cuando la pelota es golpeada, se dirige al campo contrario, rebotando en los límites de la pista. Si atraviesa la línea de gol del oponente, el jugador se apunta un gol. Además, los jugadores pueden disparar objetos al rival y, si el objeto impacta, el rival pierde vida. Si su vida se agota, queda inmóvil hasta que se anota un gol o recupera vida mediante *power-ups*.

Los movimientos especiales se pueden activar cuando la barra de energía especial está llena, otorgando distintas ventajas temporales al jugador que lo ejecute. Durante el partido, aparecen *power-ups* que al ser golpeados por la pelota, otorgan beneficios como vida extra, energía especial, escudos protectores, etc. añadiendo una capa estratégica al juego.

6.3.1. Experiencia de juego

El videojuego ofrece una experiencia de juego intuitiva y accesible, permitiendo a los jugadores de todos los niveles disfrutar del juego sin necesidad de tener habilidades previas en videojuegos.

La inclusión de power-ups y movimientos especiales añade una capa estratégica que puede aumentar la satisfacción del jugador al permitirle cambiar el curso del juego. Además, la posibilidad de ajustar la dificultad del juego permite a los jugadores adaptar su experiencia de juego, lo que puede aumentar su satisfacción. En la Figura 126 se puede ver en acción el *power-up* del hielo:

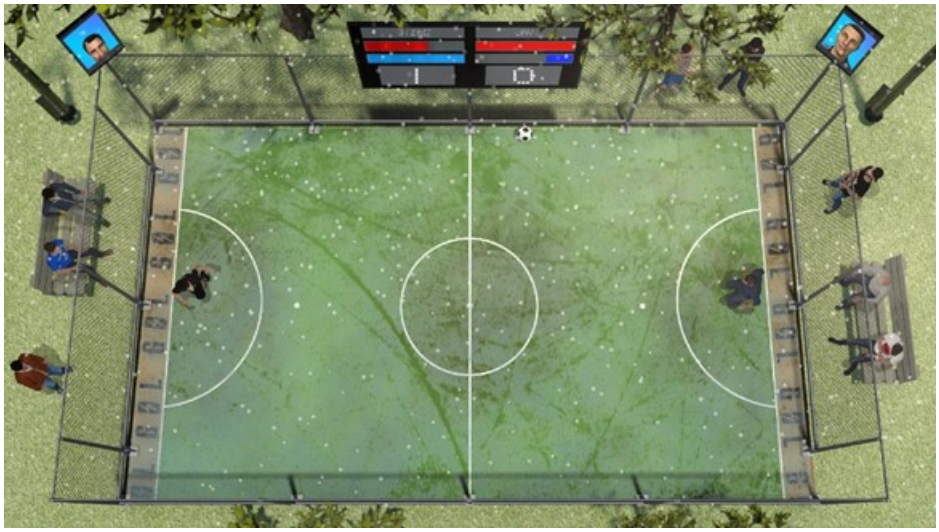


Figura 126*:* Acción del *power-up* de hielo. Fuente: Elaboración propia

6.3.2. Aprendizaje

El videojuego facilita el aprendizaje de los jugadores simplificando la acción más básica de PWP como es el golpeo de la pelota, para que cualquier jugador que juegue por primera vez no se sienta abrumada por el uso de distintos controles.

Esto permite a los jugadores disfrutar del juego directamente, sin tener que aprenderse distintas normas, estrategias, etc. Y cuando el jugador decida ahondar en las mecánicas del juego, puede dominar otros aspectos, como el uso de movimientos especiales, power-ups y la posibilidad de apuntar la dirección del golpeo antes de impactar la pelota.

En la Figura 127 se puede ver en acción el *power-up* del noche mientras se ejecuta el movimiento especial de Gizmo:

Figura 127*: : Power-up* de noche y movimiento especial de Gizmo. Fuente: Elaboración propia

6.3.3. Inmersión

Uno de los aspectos más destacables de PWP es la personalización de cada uno de los personajes que en él aparecen, logrando que el jugador se vea reflejado y, por lo tanto, inmerso totalmente en el videojuego.

El videojuego mantiene la atención del jugador al ofrecer una experiencia de juego dinámica y emocionante. La velocidad de la pelota, que varía según la dificultad elegida y se incrementa progresivamente con cada golpe, mantiene a los jugadores en constante alerta.

Además, la posibilidad de disparar objetos al rival y la aparición de power-ups durante el partido añaden elementos de sorpresa que pueden aumentar la inmersión del jugador. En la Figura 128 se puede ver la animación de Gizmo en el menú de selección de personajes:

Figura 128*:* Animación de Gizmo en la selección de personajes. Fuente: Elaboración propia

7. Conclusiones y líneas de futuro

Este Trabajo Final de Grado es el reflejo de algunas de las habilidades adquiridas por el autor durante su aprendizaje en el Grado de Informática y, a la vez, se ha convertido en un regalo a sus amigos en forma de videojuego.

El objetivo principal de este proyecto era lograr el desarrollo completo del videojuego, dado que el mismo se iba a transformar en un regalo, y la finalización de un proyecto de estas características es una tarea bastante compleja debido a los distintos aspectos de la informática que se tienen que manejar.

El resultado final del proyecto cumple con las expectativas depositadas por el autor, siendo consciente de la limitación de tiempo del proyecto y tratando de que el mismo no fuera demasiado ambicioso.

Esta limitación temporal hizo que, ya en la primera idea del juego, se descartaran determinadas funcionalidades (menú de opciones, créditos, tutorial, etc.) para priorizar otros factores que tienen más peso en el juego.

7.1. Claves

Las claves para la consecución del proyecto han sido las siguientes:

Ideas claras: El autor conocía perfectamente el tipo de videojuego que quería lograr, partía de una visión clara del resultado que deseaba obtener y de sus habilidades en las distintas tecnologías para poder abordarlo.

Análisis: Como en la mayoría de desarrollos de software, un análisis exhaustivo del proyecto a desarrollar siempre es una inversión de tiempo necesaria y valiosa para poder cumplir con los objetivos**.**

Planificación: Determinar qué tareas hay que desarrollar y estimarlas en días de trabajo ayudan a no perder la perspectiva y poder seguir una rutina diaria.

Herramientas de gestión: Si bien no forman parte del producto final, estas herramientas facilitan en gran medida el desarrollo del mismo. Las *Issues* y el *Project* de Github han permitido llevar un control de las tareas a realizar, asociarlas a las fechas claves del proyecto y no perder en ningún momento el control del mismo. En la Figura 129 se puede ver un ejemplo de tareas de proyecto en Github *Issues*:

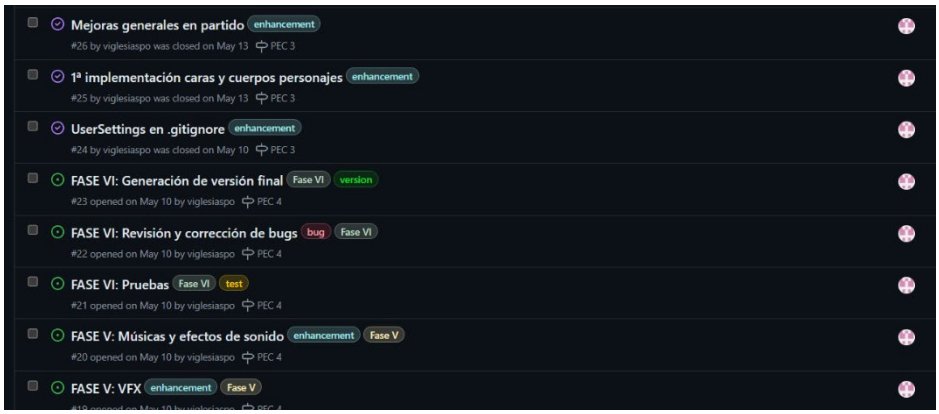


Figura 129*:* Ejemplo de tareas de PWP en GitHub *Issues.* Fuente: Elaboración propia

7.2. Contratiempos

Por otro lado, en el desarrollo de un videojuego (o software en general) siempre surgen imprevistos o hay necesidad de hacer alguna modificación, tomando decisiones que alteran algún aspecto del proyecto en la fase del desarrollo.

Como ejemplo, en este proyecto existía la duda inicial del número de jugadores a representar en el mismo: finalmente se estableció en 8 jugadores, dejando sin implementar a un amigo del autor para que el número de jugadores fuera par y el cuadro del torneo pudiera comenzar desde los cuartos de final.

Como el desarrollo del proyecto transcurría según lo planeado y el autor disponía de varios ordenadores para la renderización de los gráficos, se tomo la decisión de incluir al jugador descartado inicialmente.

Pero esta decisión repercutía en que los jugadores representados fueran impares (9) y el cuadro del torneo tuviera que ser modificado. Así que se decidió incorporar también como jugador representado al propio autor, dejando un número par de jugadores (10) y el cuadro del torneo sin cambios estéticos (si se seleccionan los 10 jugadores controlados por humanos, se juega una ronda previa de clasificación). En la Figura 130 se pueden ver 10 personajes que se pueden seleccionar en el juego:

Figura 130: Selección de personajes en PWP. Fuente: Elaboración propia

Si bien esta decisión conllevó a un esfuerzo adicional en el proyecto, al tratarse de un regalo para los amigos, el poder incluir a una persona más siempre merece la pena.

7.3. Líneas de futuro

El producto final de este trabajo es una copia digital para PC del videojuego, pero también una copia física, única y limitada en DVD y USB. Si bien las actualizaciones del producto son posibles, para la copia física quizás no tendría tanto sentido al tratarse de un regalo que no se puede modificar (se puede grabar otro DVD, otras carátulas, etc. pero perdería la razón de ser del regalo).

En la Figura 131 se puede ver la carátula del DVD de la edición limitada de Yago:

Figura 131*:* Caratula de la copia física de PWP, edición limitada de Yago. Fuente: Elaboración propia

De todas maneras, a continuación se detallarán las posibles mejoras que podrían incluirse a futuro en el videojuego:

- **Funcionalidades descartadas**: anteriormente se comentó que ciertas funcionalidades (menú de opciones, créditos, tutorial, etc.) se había descartado inicialmente para priorizar otros factores que tienen más peso en el juego. Se podrían incluir en el futuro como una actualización del producto.

- **Nuevos personajes**: Como ya se comentó en el apartado 6.2, en el desarrollo del juego se decidió incorporar dos personajes más. Una posible actualización podría incorporar nuevos personajes y sus movimientos especiales para generar nuevas estrategias y ampliar el catálogo de jugadores.

- **Nuevos *power-ups***: Como en el caso de los personajes, incluir nuevos *power-ups* podría ser una futura actualización del videojuego que lo dotara con nuevas estrategias para desnivelar la balanza en un partido.

- **Nuevos escenarios**: Actualmente, la pantalla de partido dispone de un único escenario que sirve para establecer las dimensiones del terreno de juego pero que no influye directamente en el desarrollo del partido, más allá del rebote de la pelota. Incluir nuevos escenarios con obstáculos, porterías de distintos tamaños u otras ideas que hagan que el escenario influya en la jugabilidad del partido podría ser una futura actualización.

- **Nuevos diseños de pantallas**: Hay ciertas pantallas (menú principal, menús de selección de modo de juego, de dificultad, etc.) que se han diseñado de una manera sencilla para que el proyecto no se tornara demasiado ambicioso y corriera el riesgo de no concluirse. Una futura actualización podría mejorar dichas pantallas.

- **Nuevas animaciones**: En determinadas pantallas, existen gráficos estáticos que podrían cambiarse por animaciones para darles un aspecto más dinámico. Por ejemplo en el pantalla de Prepartido o de Postpartido, se podrían sustituir las imágenes de los jugadores por unas animaciones en bucle de los mismos.

- **Nueva mecánicas**: Dotar de nuevas mecánicas de juego en los partidos podría ser una buena manera de revitalizar el videojuego, ya que es el apartado principal en el que los jugadores se enfrentan y en el que radica la principal diversión del producto.

Bibliografía

[1] Tokyo school, «Sueldo de un diseñador de videojuegos: ¿cuánto cobra?,» [En línea]. Available: https://www.tokioschool.com/formaciones/cursos-videojuegos/diseno/sueldo/. [Último acceso: 18 03 2024].

[2] Tokyo School, «¿Cuál es el sueldo de un programador con Unity?,» [En línea]. Available: https://www.tokioschool.com/formaciones/cursos-videojuegos/programacion-unity/sueldo/. [Último acceso: 18 03 2024].

[3] Tokyo School, «Modelador 3D para videojuegos: su sueldo al detalle,» [En línea]. Available: https://www.tokioschool.com/formaciones/creacion-modelado-personajes-3d-videojuegos/sueldo/. [Último acceso: 18 03 2024].

[4] Grover, «Alquiler de portátiles,» [En línea]. Available: https://www.grover.com/es-es/computers/laptops?filter=rentalPlan%3D3&sort=PRICE_ASC. [Último acceso: 18 03 2024].

[5] Daz 3D, «Download Daz Studio,» [En línea]. Available: https://www.daz3d.com/technology/. [Último acceso: 20 03 2024].

[6] Daz 3D, «Interactive Licenses,» [En línea]. Available: https://www.daz3d.com/interactive-license-info. [Último acceso: 20 03 2024].

[7] Daz 3D, «Face Transfer Unlimited,» [En línea]. Available: https://www.daz3d.com/face-transfer-unlimited. [Último acceso: 20 03 2024].

[8] Unity, «Planes y precios: Comienza a crear con Unity,» [En línea]. Available: https://unity.com/es/pricing#plans-student-and-hobbyist. [Último acceso: 20 03 2024].

[9] Blender, «Blender is Free Software,» [En línea]. Available: https://www.blender.org/about/license/. [Último acceso: 20 03 2024].

[10] Adobe, «Mixamo FAQ - Licensing, Royalties, Ownership, EULA and TOS,» [En línea]. Available: https://www.blender.org/about/license/. [Último acceso: 20 03 2024].

[11] Github, «Get the complete developer platform,» [En línea]. Available: https://github.com/pricing. [Último acceso: 20 03 2024].

[12] Wikipedia, «Arcade,» [En línea]. Available: https://es.wikipedia.org/wiki/Arcade. [Último acceso: 16 04 2024].

[13] Yahoo! Finances, «Gaming Skins Just Became A $50 Billion Industry,» [En línea]. Available: https://finance.yahoo.com/news/gaming-skins-just-became-50-143352555.html. [Último acceso: 16 04 2024].

[14] Criptonoticias, «Cada vez más gamers quieren que sus ‘skins’ sean NFT, muestra un estudio,» [En línea]. Available: https://www.criptonoticias.com/mercados/gamers-quieren-skins-nft-muestra-estudio/. [Último acceso: 16 04 2024].

[15] Wikipedia, «Motor de videojuego,» [En línea]. Available: https://es.wikipedia.org/wiki/Motor_de_videojuego. [Último acceso: 17 04 2024].

[16] Wikipedia, «Modelado 3D,» [En línea]. Available: https://es.wikipedia.org/wiki/Modelado_3D. [Último acceso: 17 04 2024].

[17] Wikipedia, «Animación por computadora,» [En línea]. Available: https://es.wikipedia.org/wiki/Animaci%C3%B3n_por_computadora. [Último acceso: 17 04 2024].

[18] Wikipedia, «Microsoft Visual Studio,» [En línea]. Available: https://es.wikipedia.org/wiki/Microsoft_Visual_Studio. [Último acceso: 19 05 2024].

[19] Codeandweb, «TexturePacker,» [En línea]. Available: https://www.codeandweb.com/texturepacker. [Último acceso: 19 05 2024].

[20] Pngquant, «Pngquant,» [En línea]. Available: https://pngquant.org/. [Último acceso: 19 05 2024].

[21] Advancedinstaller, «Advancedinstaller,» [En línea]. Available: https://www.advancedinstaller.com/. [Último acceso: 19 05 2024].