UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Videojuego "Frozen Out". Programación, desarrollo e integración de mecánicas

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Tomás Ruiz Martín

Tutor: Adolfo Muñoz García

2019-2020

Resumen

Frozen Out es un proyecto de emprendimiento que nace de la unión de alumnos de la ETSINF y BBAA a través de las asignaturas de Introducción a la programación de videojuegos y de Desarrollo de videojuegos, respectivamente.

El juego se enmarca como una aventura gráfica 3D con algunos elementos de sigilo. Donde con diferentes herramientas y habilidades deberemos abrirnos paso por la ciudad hasta llegar al centro de la ciudad e intentar resolver la crisis de la falta de hielo en la ciudad.

El grupo de programación está compuesto por: Vicent Pla, encargado de la integración y extensión de herramientas externas al motor de videojuegos Unity y el diseño del código; Pablo López, responsable de la inteligencia artificial de los elementos del juego que lo requieran; Alejandro Gómez, quien ya no forma parte del equipo pero que, en su momento, se encargó de iniciar el diseño del código y la integración de herramientas externas; y Tomás Ruiz, como desarrollador de las mecánicas del juego y continuador el diseño del código. En cuanto a la parte artística, el encargado es Alejandro Jiménez, mientras que del diseño de interacción se encarga Adrián Reina.

El trabajo a realizar en este TFG es el de seguir y completar el desarrollo de las mecánicas del juego, así como la mejora y adaptación de las ya implementadas. Para ello, las diferentes acciones y/o mecánicas, se vincularán con las animaciones y efectos requeridas por cada una, además de con los elementos con los que se pueda interactuar, uniendo así la parte artística y la de programación. Todo esto siguiendo el diseño del código que nos planteamos, teniendo como objetivo el crear nuevos niveles de forma más sencilla y modular.

Palabras clave: videojuego, emprendimiento, Unity, mecánicas, “cambio climático”.

Abstract

Frozen Out is an entrepreneurial project born from the union of students from ETSINF and BBAA through the subjects of Introducción a la Programación de Videojuegos and Desarrollo de Videojuegos, respectively.

The game is framed as a 3D graphic adventure with some elements of stealth. Where with different tools and skills we will have to make our way through the city until we reach the city centre and try to solve the crisis of the lack of ice in the city.

The programming group consists of: Vicent Pla, in charge of the integration and extension of external tools to the Unity videogame engine and the design of the code; Pablo López, responsible for the artificial intelligence of the game elements that require it; Alejandro Gómez, who is no longer part of the team but who, at the time, was in charge of starting the design of the code and the integration of external tools; and Tomás Ruiz, as developer of the game mechanics and continuator of the design of the code. As for the artistic part, the person in charge is Alejandro Jiménez, while Adrián Reina is in charge of the interaction design.

The work to be done in this TFG is to follow and complete the development of the game mechanics, as well as the improvement and adaptation of those already implemented. To do so, the different actions and/or mechanics will be linked to the animations and effects required by each one, as well as to the elements with which it is possible to interact, thus joining the artistic and the programming part. All this following the design of the code that we set out, having as an objective to create new levels in a simpler and more modular way.

Keywords: video game, entrepeneurship, Unity, mechanics, “global warming”.

Tabla de contenidos

1. Introducción

La escena del videojuego indie en España ha estado cogiendo fuerza con juegos como *Blasphemous* [1] o *Temtem* [2]. Videojuegos españoles enfocados en experiencias para un solo jugador, los cuales tienen en común el haber sido posibles gracias a campañas de microfinanciación[3], en la plataforma Kickstarter, que basa su funcionamiento en dar visibilidad a proyectos de diferentes categorías, donde personas interesadas en que ese producto sea desarrollado aportan su dinero para que sus creadores puedan finalizarlo, ofreciendo a las personas que hayan aportado al proyecto diferentes recompensas según la cantidad aportada, normalmente de forma adicional al producto que ofrecen. Al ser los interesados los que aportan la financiación y con los juegos antes mencionados, es evidente que los productos enfocados en un solo jugador siguen siendo demandados hoy en día más allá de los juegos como servicio[4] o que ofrecen únicamente una experiencia online con otros usuarios.

Pese a que realmente es difícil que una campaña reúna la cantidad de dinero necesaria, debido principalmente al número de propuestas que se les presenta al consumidor, es posible para los desarrolladores poder vivir de ella mientras la desarrollan, siendo visible la buena acogida que pueden tener estos proyectos si la idea está bien ejecutada y cumple con lo que ofrecen. Ya que, desde el lanzamiento de la plataforma en 2008, se han conseguido con éxito financiar 186.888 proyectos con una suma total de 5.230.562.754 $, de los cuales 22.308 son sólo videojuegos y juegos de mesa, a fecha de 27 de agosto de 2020, según la propia página de Kickstarter (Kickstarter, 20). En uno de los ejemplos anteriores (*Temtem*), se llegaron a recaudar 573.939 $, alcanzando todas las metas propuestas para el desarrollo (Kickstarter, 2020).

Así pues, desde el punto de vista de un proyecto de emprendimiento de carácter multidisciplinar, se mostrará la elaboración de la idea de negocio y en concreto el desarrollo de las mecánicas del juego y su integración con elementos tales como animaciones, efectos y partículas. El resto de los aspectos referentes al proyecto como el diseño, el arte, la integración de herramientas externas empleadas y la IA, están recogidos en los trabajos de mis compañeros de equipo:

- Alejandro Jiménez Carrasco - FROZEN OUT (I): Diseño y desarrollo artístico de un videojuego crítico 3D. (Carrasco, 2020)
- Adrián Reina Sáez - FROZEN OUT (II): Diseño de interacción y animación de caracteres de un prototipo de videojuego crítico 3D. (Sáez, 2020)

[1] Más información en <https://es.wikipedia.org/w/index.php?title=Blasphemous&oldid=128381228 >
[2] Más información en <https://en.wikipedia.org/w/index.php?title=Temtem&oldid=972816302>
[3] Financiación de proyectos por medio de pequeñas aportaciones económicas de un gran número de personas
[4] Juegos, normalmente gratuitos y en línea, cuya monetización se basa en la venta de paquetes mensuales o micro pagos para la obtención de mejoras visuales dentro del juego.

— Vicent Pla Madrid – "Frozen Out", videojuego de aventura gráfica 3D. Diseño y uso de los diálogos. (Madrid, 2020)
— Pablo Pérez Orrios - Desarrollo de un videojuego con Unity: implementación de la inteligencia artificial y funciones back-end. (Orrios, 2020)

Motivación

La motivación principal de este proyecto es la de desarrollar, en este caso, un videojuego con un equipo de personas cuyo punto de vista y manera de contar las cosas pueda darle un toque especial a la experiencia que nos proponemos desarrollar y mostrar. Ya que, habiendo sido consumidor de videojuegos desde una edad muy temprana y el interés acerca de cómo funciona la industria y las obras cada vez más sorprendentes que los desarrolladores realizaban, tenía claro que quería desarrollar uno, tanto si tenía éxito como si no una vez lanzado al mercado. Esto juntado a la inmensidad de información y ayudas que se pueden encontrar por internet para hacerlo posible y habiendo encontrado un buen equipo con el que trabajar y realizar, no sólo este proyecto si no muchos otros más en un futuro, era impensable no intentarlo.

En mi caso, siempre he disfrutado experiencias enfocadas en un solo jugador, donde la expresividad del entorno es un factor importante para entrar de lleno en el mundo. Es por eso por lo que, con Frozen Out hemos querido tomar ese rumbo durante la ejecución de la idea, donde el mapa y sus personajes te describan como está la situación en la que te encuentras sin necesidad de una extensa exposición en los diálogos. Debido a esto se ha escogido el género de aventura y plataformas, donde la interacción con el entorno y los personajes es una parte importante de la experiencia. Debiendo dejar claros los objetivos, pero permitiendo que el usuario pueda explorar y ver los detalles que dotan al producto de su encanto.

Objetivo

Al ser un proyecto un grupo, nuestros objetivos generales son los siguientes:

- Realizar un prototipo del primer nivel de un videojuego 3D dentro de la tipología puzle-exploración.
- Crear una experiencia cuya temática principal sea el cambio climático, siendo una experiencia basada en la jugabilidad y la narrativa.
- Experimentar de primera mano lo que es llevar a cabo un proyecto de emprendimiento de carácter multidisciplinar.

En cuanto a mis objetivos específicos, han sido planteados los siguientes:

- Crear las bases para que la elaboración de nuevos niveles o escenas jugables sea cómoda y eficiente.
- Aplicar los conocimientos adquiridos durante el grado, en particular aquellas relacionadas con el desarrollo de videojuegos y la gestión de proyectos con metodologías ágiles.
- Asentar mis conocimientos usando la herramienta Unity para la creación de mecánicas y sistemas de juego.

Estructura

En esta memoria se presentará el proceso de desarrollo de un videojuego dentro de un contexto de emprendimiento. Donde en primer lugar se expondrá la evaluación de la idea de negocio, la cual incluye un estudio de mercado; un análisis de las debilidades, amenazas, fortalezas y oportunidades (DAFO) de nuestro producto; una explicación del modelo de negocio a llevar por nuestra empresa a tres años vista, el Lean Canvas y las conclusiones a destacar tras el desarrollo.

En el apartado sobre el desarrollo de la idea de negocio, se indagará en las decisiones tomadas durante este. Dividiéndolo en dos partes y estando cada una centrada en uno de los dos MVP desarrollados, incluyendo los datos obtenidos tras realizar pruebas con usuarios y las reflexiones y decisiones tomadas en base a ello.

En la sección de aspectos técnicos, se enumerarán las herramientas principales que sirvieron para la organización y el desarrollo del juego; como las mecánicas fueron implementadas, abarcando las posibilidades presentadas en cuanto a su implementación se refiere, así como el funcionamiento de estas, su estructura e integración del código con herramientas del motor empleado y la utilización de los elementos visibles producidos.

Por último, se tratarán las conclusiones tras el desarrollo y el trabajo futuro que se nos presenta, a raíz de finalizar el desarrollo del nivel y haber realizado la pertinente prueba con usuarios. Así como la corrección de fallos y mejoras de ciertos aspectos en base a las diversas sugerencias y fallos localizados.

PARTE I: Trabajo de Emprendimiento

2. Generación de la idea de negocio

La idea del proyecto se originó durante el curso de la asignatura Introducción a la Programación de Videojuegos (IPV), durante la cual se formaron grupos y se introdujeron estudiantes de la asignatura Desarrollo de Videojuegos (del grado de Diseño y Tecnologías Creativas en la facultad de Bellas Artes). Al principio de la asignatura se pidió a cada alumno, de ambas facultades, la propuesta de una idea para videojuego en formato *pitch doc*[5]. De todas las propuestas los profesores escogieron 7 y formaron grupos de 6 o 7 alumnos, en base a nuestras preferencias, con el objetivo de desarrollar la idea propuesta, liderada por el alumno que originó la idea.

En nuestro caso, la idea consistía en un juego de puzles en tercera persona con un parte de exploración, donde uno de los atractivos sería la historia: ambientada en una distopía[6] polar protagonizada por polos y helados con una clara temática medioambiental. La metáfora de juego deseada es la representación de los problemas de la crisis medioambiental y los vicios del modelo productivo que la han perpetrado. La idea estaba pensada para todos los públicos, pero tras una serie de decisiones de diseño, hubo que apostar por subir el PEGI[7] a 7. Los referentes a la hora de desarrollar la idea fueron *Zack and Wiki*[8], un videojuego 3D en tercera persona publicado en 2008 para la plataforma Nintendo Wii[9], donde se siguen las aventuras del personaje Zack en una isla perdida, la cual sirve como entorno de juego.

[5] Documento empleado para la presentación de una idea ante futuros inversores o interesados
[6] Representación ficticia de una sociedad futura de características negativas causantes de la alienación humana
[7] < https://pegi.info/es >
[8] Disponible en <https://en.wikipedia.org/w/index.php?title=Zack_%26_Wiki:_Quest_for_Barbaros%27_Treasure&oldid=937787682>
[9] < https://es.wikipedia.org/w/index.php?title=Wii&oldid=128666528>

3. Evaluación de la idea de negocio

En esta sección se analizará en profundidad la viabilidad de la idea de negocio mediante diferentes medios, como el análisis DAFO o el Lean Canvas. Mediante el resultado de estos, se ha elegido un modelo de negocio que se ajuste al marco en el que nos movemos debido a la naturaleza de nuestro producto, véase, un videojuego.

3.1 Estudio de mercado

Sobre el ejercicio del año 2019, la facturación digital de videojuegos en España ha crecido un 6,6 %, mientras que la facturación física y la total ha decaído un 11,2 % y un 3,3 %, respectivamente (AEVI, 2019). Estas cifras justifican la indiferencia en torno a la creación de una edición física, centrando los recursos y el esfuerzo en la edición digital para que esta tenga la mayor cantidad de contenido posible. Así pues, los estudios independientes de videojuegos suelen optar por esta opción debido al bajo coste de producción. En cuanto al público objetivo, atendiendo a cifras del mismo informe, el género aventura es el cuarto más popular, con 1.064.891 de unidades vendidas. El producto por desarrollar no pretende encontrar un vacío en el mercado, sino explotar un mercado existente y sobreponerse en cuestiones de historia y jugabilidad sobre sus competidores directos.

Atendiendo a datos del resto del mundo, España está situado en el décimo puesto dentro de los países en los que la industria del videojuego más factura, con unas cifras de 2.656 millones de dólares. Mientras que China se encuentra en el primer puesto con unas cifras de 40.854 millones de dólares (Newzoo, 2020), debido principalmente al auge de los juegos para dispositivos móviles, los cuales pese a ser gratuitos, presentan un modelo de negocio basados en microtransacciones[10] para desbloquear cosméticos o mejoras dentro del juego, por lo que no se podría confiar en este mercado para el lanzamiento de nuestro juego, ya que además debería de pasar por una serie de inspecciones para poder ser publicado en el país. Por otro lado, en E.E.U.U. se facturaron 36.921 millones de dólares, aunque pese a tener grandes beneficios en el sector móvil, los juegos en las demás plataformas siguen teniendo un papel principal en la industria. Por tanto, se diferencian por tener una mayor proporción de jugadores para consola y computadoras, y son más susceptibles a formar parte del público objetivo.

Visto el panorama del mercado móvil y las grandes campañas de *marketing* elaboradas por las editoras de videojuegos, la opción más viable para un estudio indie para crear un juego sin cualidades de servicio ni recurrencia es optar por un sistema de microfinanciación. Claros ejemplos de este modelo son *Moonlighter*[11] (2018) y *Summer in Mara*[12] (2020), los cuales

[10] Modelo de negocio donde los usuarios pueden comprar objetos virtuales mediante el uso de micropagos.
[11] Más información en <https://es.wikipedia.org/w/index.php?title=Moonlighter&oldid=122617059>
[12] Más información en <https://www.mobygames.com/game/summer-in-mara>

tuvieron gran éxito en su campaña de Kickstarter[13]. En concreto, *Moonlighter* obtuvo 134.276 $ de los 40.000 $ que pedían en la propuesta[14], y para el caso de *Summer in Mara* se superó en un 1000% el objetivo base de la propuesta[15], con 233.919 €.

A continuación, se destacan videojuegos considerados competencia del producto en desarrollo:

A Short Hike[16]

Videojuego 3D en tercera persona publicado en 2019 para Nintendo Switch, Windows, Linux y macOS, desarrollado por una sola persona. Se trata de un juego del género aventura y plataformas para un jugador donde controlas a un personaje pájaro a través de una isla. Este personaje pájaro empieza una aventura por la isla para llegar al punto más alto de una montaña con el objetivo de conseguir señal de telefonía, durante esta travesía conoce a otros personajes y entender la historia y naturaleza del entorno. Como se puede comprobar en la Figura 1, cuenta con una estética de baja resolución. El juego ha ganado premios en certámenes independientes, como el premio de la audiencia en el *Independent Games Festival*[17] de 2019, además de ser nominado en otros eventos como *D.I.C.E.*[18] o *Golden Joystick Awards*[19].

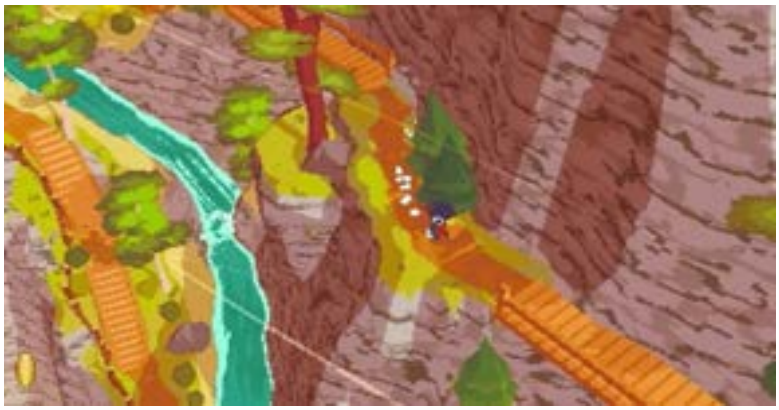

Figura 1. Imagen promocional de A Short Hike. A Short Hike (2019).

Además de mover al personaje en el entorno, el juego también ofrece otras mecánicas como saltar, escalar y planear que están limitadas por una barra dentro del juego, así como el uso de ítems esparcidos por el juego que permiten expandir sus acciones. En cuanto al diálogo, el juego también permite interactuar con otros personajes para entablar conversaciones con ellos, usualmente de carácter humorístico y alegre.

[13] Más información en <https://es.wikipedia.org/w/index.php?title=Kickstarter&oldid=127304922>
[14] Extraído de <https://www.kickstarter.com/projects/digitalsun/moonlighter>
[15] Extraído de <https://www.kickstarter.com/projects/chibig/summer-in-mara-an-adventure-set-in-a-tropical-ocea>
[16] Más información en <https://en.wikipedia.org/w/index.php?title=A_Short_Hike&oldid=975096031>
[17] Más información en <https://es.wikipedia.org/w/index.php?title=Independent_Games_Festival&oldid=119050744>
[18] Más información en <https://en.wikipedia.org/w/index.php?title=D.I.C.E._Awards&oldid=972060055>
[19] Más información en <https://es.wikipedia.org/w/index.php?title=Premios_Golden_Joystick&oldid=128294657>

Summer in Mara

Summer in Mara es un videojuego de aventura y gestión de recursos para un solo jugador, desarrollado por Chibig[20] y publicado el 16 de junio 2020 para las plataformas PS4, Xbox One, Nintendo Switch y Windows. Los creadores realizaron una campaña de Kickstarter con éxito, con un total de 9.523 patrocinadores y 233.919 euros recaudados, siendo la meta principal de 20 mil euros.

Figura 2. Imagen promocional de Summer in Mara. Summer in Mara (2020).

El videojuego trata de una niña que vive en una isla. La cual quiere explorar el océano y las islas que la rodea, mientras cuida de la isla en la que habita y hacer recados para los lugareños entregándoles lo que necesitan y así avanzar en la historia. Para ello, Koa, la protagonista, es capaz de moverse libremente por el entorno para recoger recursos y crear nuevos objetos útiles a partir de estos. Es un juego en 3D (Figura 2) en el que explorar el entorno es una parte vital para encontrar lo que necesitas.

[20] Más información en <https://chibig.com/>

A Hat in Time[21]

Videojuego de aventuras y plataformas desarrollado por Gears for Breackfast[22] y publicado en 2017 para las plataformas PS4, Xbox One, Nintendo Switch y Windows y macOS. Al igual que el anterior, también desarrolló gracias a una campaña de Kickstarter[23], con 9.169 patrocinadores y 296.360 $ recaudados. A destacar que el equipo de desarrollo consistía en un equipo fijo y en voluntarios que se iban sumando al proyecto para sacarlo adelante.

Figura 3. Imagen promocional de A Hat in Time. A Hat in Time (2017).

El videojuego, el cual se puede ver en la Figura 3, trata de una niña que viaja en una nave espacial a la que le falla la fuente de combustible, una especie de relojes de arena, que acaba desperdigada por los alrededores, Por lo que tendrá que volver a reunirlas, viajando a distintos mundos cercanos donde estos se encuentran, moviéndose por el mapa dando grandes saltos y cambiando de una plataforma a otra para alcanzar grandes alturas.

[21] Más información en <https://es.wikipedia.org/w/index.php?title=A_Hat_in_Time&oldid=126612097>
[22] Más información en < https://gearsforbreakfast.com/>
[23] Disponible en <https://www.kickstarter.com/projects/jonaskaerlev/a-hat-in-time-3d-collect-a-thon-platformer>

Spyro Reignited Trilogy[24]

Videojuego 3D en tercera persona para un jugador publicado en 2019 para PS4, Xbox One, Nintendo Switch y Windows, desarrollado por Toys for Bob[25]. Se trata además de una remasterización[26] de una serie de juegos publicados entre 1998 y 2000. Se encuentra dentro del género de aventuras, puzles y plataformas y sigue las andanzas de un dragón para restablecer la paz en una gran variedad de entornos y mundos. El juego permite controlar libremente al personaje principal, así como usar su habilidad especial para lanzar fuego como ataque para enemigos, tal como se ve en la Figura 4. El objetivo del juego es el de liberar otros dragones, pero también ofrece la posibilidad de recoger coleccionables esparcidos por los diferentes mundos que están conectados entre ellos por un mapa abierto.

Figura 4. Imagen promocional de Spyro Reignited Trilogy. Spyro Reignited Trilogy (2019).

La serie de juegos originales gozó de popularidad en su momento, en gran parte al carisma de sus personajes y la estética de dibujo animado, además no estaban disponibles en hardware actual, por lo que era una revisión muy demandada dentro de la comunidad.

[24] Más información en <https://es.wikipedia.org/w/index.php?title=Spyro_Reignited_Trilogy&oldid=119196281>
[25] Más información en <https://www.toysforbob.com/>
[26] Recreación de un videojuego publicado con anterioridad para actualizarlo a nuevo hardware y gráficos modernos con el fin de hacerlo accesible a un público actual

Crash Bandicoot 4: It's About Time[27]

Crash Bandicoot 4 es un videojuego 3D en tercera persona pendiente de publicarse en 2020 para PS4 y Xbox One, y desarrollado por Toys for Bob. Se trata de un juego de aventuras y plataformas, donde se controla a un *bandicut*[28] a través de una multitud de entornos. El objetivo es derrotar a un malvado científico, superando y desbloqueando niveles que se seleccionan en un mapa. El juego permite controlar el personaje en diferentes estilos, intercalando secciones en 3D donde el movimiento es hacia dentro de la pantalla o hacia fuera (Figura 5) y secciones en 2D como un juego de plataformas clásico.

Figura 5. Imagen promocional de Crash Bandicoot 4. Crash Bandicoot 4: It's About Time (2020).

Las tres primeras entregas, publicadas originalmente entre 1996 y 1998, recibieron una remasterización recientemente en 2017, por lo que se recuperó la base de usuarios y se captaron a nuevos interesados actuales.

[27] Más información en <https://en.wikipedia.org/w/index.php?title=Crash_Bandicoot_4:_It%27s_About_Time&oldid=974546032>
[28] Más información en <https://es.wikipedia.org/wiki/Peramelidae>

Yooka-Laylee[29]

Videojuego de plataformas de mundo abierto desarrollado por Playtonic Games[30] y publicado el 11 de abril de 2017 por Team17 en las plataformas PS4, Xbox One, Nintendo Switch, Windows, Linux y macOS, aunque más adelante fue publicado también para Switch. Los creadores realizaron una campaña de Kickstarter para financiar el juego[31], en esta participaron 73.206 personas las cuales contribuyeron 2.090.104 £.

Figura 6. Imagen promocional de Yooka-Laylee. Yooka-Laylee (2017).

El jugador controla a Yooka en un mundo 3D para explorar una variedad de niveles (Figura 6) y resolver puzles. Además, se hace énfasis en los coleccionables. Esta exploración se realiza a través de las habilidades especiales que se desbloquean durante el progreso de la partida, los cuales permitirán avanzar y terminar la historia.

[29] Más información en <https://es.wikipedia.org/w/index.php?title=Yooka-Laylee&oldid=127889060>
[30] Más información en <https://www.playtonicgames.com/>
[31] Disponible en <https://www.kickstarter.com/projects/playtonic/yooka-laylee-a-3d-platformer-rare-vival>

Night in the Woods[32]

Videojuego de aventuras desarrollado por Infinite Fall, un estudio formado por tres personas y publicado en 2017 para las plataformas PS4, Windows, Linux y macOS, aunque más adelante también fue publicado para las plataformas Xbox One y Nintendo Switch. Los creadores realizaron una campaña de Kickstarter para financiar el juego[33], en esta participaron 7.372 personas las cuales contribuyeron 209.375 $.

Figura 7. Imagen promocional de Night in the Woods. Night in the Woods (2017).

El juego trata de una joven que vuelve a su pueblo tras irse a la universidad. Esta intenta reunirse con sus amigos y recuperar su antigua vida, pero todo es distinto debido al paso del tiempo. Centrado en la exploración, historia y personajes, se hace énfasis en los diálogos y las interacciones con los personajes, visible en la Figura 7. Los desarrolladores de este juego además desarrollaron Yarn Spinner, el sistema de diálogos que utilizamos en este proyecto.

[32] Más información en <https://es.wikipedia.org/w/index.php?title=Night_in_the_Woods&oldid=128563295>
[33] Disponible en <https://www.kickstarter.com/projects/1307515311/night-in-the-woods>

Resumen

Los videojuegos descritos sirven como análisis de mercado para comprobar con qué tipo de juegos puede llegar a competir nuestra propuesta. Por otro lado, también es útil para comprobar qué características son populares y/o esperadas por los usuarios de este género de videojuegos. A continuación, se muestran las funcionalidades más relevantes de estos juegos en una tabla para una comprensión más fácil y rápida. Además, adjuntamos nuestra propuesta en la Tabla 1 a modo de comparativa crítica de las funcionalidades deseables.

Característica	*A Short Hike*	*Summer in Mara*	*A Hat in Time*	*Spyro Reignited Trilogy*	*Crash Bandicoot 4: It's about time*	*Yooka-Laylee*	*Night in the Woods*	*Frozen Out*
Plataformas	Windows macOS Linux Switch	Windows PS4 XBO Switch	Windows macOS PS4 XBO Switch	Windows PS4 XBO Switch	PS4 XBO	Windows macOS Linux PS4 XBO Switch	Windows macOS Linux PS4 XBO Switch	Windows macOS Linux
N.º de jugadores	1	1	1-2	1	1	1-2	1	1
Precio de salida	6,59 €	21,99 €	27,99 €	39,99 €	-	39,99 €	19,99 €	19,99 €
Pocas mecánicas	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí
Duración aprox.	3 h	25 h	10 h	15 h	-	15 h	8 h	8 h
Rejugabilidad por argumento	No	No	No	No	No	No	Sí	No
Ambientación original	Sí	No	Sí	Sí	No	No	Sí	Sí
Diálogo entretenido	Sí	No	Sí	No	-	Sí	Sí	Sí
Estilo artístico simple	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí
Crítica medioambiental	No	No	No	No	-	No	No	Sí
Simplicidad de niveles	No	Sí	No	Sí	-	No	Sí	Sí
Centrado en la exploración	Sí	Sí	Sí	Sí	No	Sí	No	Sí
Uso de ítems	Sí	Sí	No	No	No	No	No	Sí
Coleccionables	Sí	No	Sí	Sí	Sí	Sí	No	No

Tabla 1. Tabla comparativa de juegos competidores con la propuesta a desarrollar. Elaboración propia.

Examinando la tabla se puede extraer que un juego de plataformas y aventuras debe tener una ambientación original, usualmente dentro de un mundo fantástico, con personajes carismáticos y con un estilo de dibujo animado o con diseños simples, donde gran parte del encanto viene de la interacción con ellos a través de los diálogos, dejando también sitio para la crítica en estos. En cuestión de mecánicas, aun siendo simples deben cumplir con su función y permitir al jugador poder explorar el entorno. Al igual que en nuestro proyecto, los esfuerzos en la mayoría de los casos se centran en crear una aventura para un jugador. En cuanto a las plataformas donde se encuentran disponibles, visto la presencia de consolas es una buena idea considerar en un futuro estas plataformas, ya que usualmente estos juegos consideraron las consolas una vez tuvieron una base de jugadores en computadoras. La rejugabilidad en este tipo de juegos no suele ser importante, ya que el jugador sigue las historia y rara vez tiene alternativas para completar sus misiones, tampoco se ofrecen incentivos para volver a jugar exceptuando casos en los que existen coleccionables o personajes que solo aparecen si se cumplen ciertos requisitos.

3.2 Análisis DAFO

El análisis DAFO permite a cualquier proyecto conocer sus puntos fuertes y débiles, de modo que los desarrolladores pueden centrar sus esfuerzos en los aspectos que harán su producto destacar, además de conocer de antemano en qué secciones se va a ver perjudicado por su competencia y se deberían reforzar o minimizar su impacto. A continuación, se muestra en la Tabla 2 la matriz realizada para este proyecto usando esta técnica:

Debilidades	**Amenazas**
• Escaso contenido al principio • Falta de financiación • Equipo inexperto • Poca presencia online • Primer producto en el mercado • No presente para plataformas móviles ni consolas	• Exceso de lanzamientos de juegos independientes en plataformas digitales • Competencia asentada • Mayor duración de los juegos • Auge de la popularidad de juegos como servicio
Fortalezas	**Oportunidades**
• Equipo multidisciplinar • Entorno estudiantil • Soporte de profesorado experto • Participación en ferias universitarias • Idea innovadora • Personajes y entorno del juego originales • Disponibilidad de espacio de trabajo	• Jugadores en busca de contenido diferente • Jugadores con poco tiempo en busca de experiencias cortas • Comunidad centrada en juegos independientes

Tabla 2. Matriz DAFO para el proyecto Frozen Out y sus componentes. Elaboración propia.

El proyecto se va a realizar bajo el marco de la universidad, por lo que va a existir una amplia cantidad de recursos disponibles, además de la ayuda directa o indirecta de profesorado experto a través de seminarios, así como la asistencia a ferias creadas por la comunidad universitaria. Cabe destacar también la disponibilidad de espacios de trabajo por parte de la universidad para casos como el nuestro, en el que un grupo de estudiantes decide crear un proyecto de emprendimiento. La variedad de personalidades y especialidades en los componentes del grupo de trabajo permite crear un entorno donde contrastar diferentes ideas y diferentes puntos de vista. Creemos que se trata también de una idea innovadora, dado que no hemos encontrado ejemplos de juegos protagonizados por helados y polos, y con una crítica medioambiental, por lo que será una de las características que reforzaremos en la campaña de mercadotecnia para centrar la atención del tipo de consumidor que busca juegos de corta duración y que ofrezcan experiencias distintas a la que encuentran en los juegos y géneros más populares.

Por otra parte, el hecho de que esta sea la primera toma de contacto del equipo con la industria conlleva una serie de problemas: falta de presupuesto, poca presencia destacable en redes y baja experiencia en las mismas, lo que nos pone en gran desventaja frente a nuestros competidores. Además, entre tanta cantidad de lanzamientos de juegos, sobre todo videojuegos independientes, es bastante común que productos prometedores se vean enterrados entre tantos productos disponibles. En caso de no ser así, también habría que tener en cuenta la cantidad de tiempo empleado por los jugadores en juegos como servicio, lo que hace que no tengan disponibilidad para probar otros juegos.

3.3 Modelo de negocio y proyección económica a 3 años

En cuanto al modelo de negocio propuesto para nuestro videojuego, se va a optar por un precio de licencia único de 19,99 €, ya que en los juegos con una historia cerrada los consumidores prefieren un precio final para despreocuparse de ítems y contenido adicional que pueda crear experiencias distintas y separar la comunidad. La razón de este precio es por la relación de contenido y duración con los ejemplos mostrados durante el estudio de mercado, concretamente con *A Hat in Time* y *Yooka-Laylee*. Aun así, no se descarta la creación de expansiones que exploren la historia de otros personajes o del personaje principal, de modo que se extienda la historia existente del juego original. A la hora de crear una nueva empresa, al ser el objetivo de un proyecto de emprendimiento, es de alta relevancia analizar la proyección económica con tal de revisar la viabilidad del proyecto durante su desarrollo y comprobar de manera aproximada los gastos que puedan generar y los ingresos del proyecto.

Las plataformas de venta elegidas para su distribución van a ser primariamente plataformas para computadoras, ya que la publicación de videojuegos para estos sistemas requiere de una inversión nula o más reducida en comparación con las plataformas de consola, donde existe una

única plataforma digital de venta y es controlada por las compañías que desarrollan las consolas. Esto también se debe a la libertad de desarrollo para sistemas de computadoras, ya que la creación del ejecutable para consolas requiere de un permiso de desarrollador de las compañías propietarias de dichas consolas. Además, se descarta la publicación física debido, por una parte, a la decaída de las ventas físicas durante los últimos años según (AEVI, 2019), y, por otra parte, a la necesidad de contratar una editora de videojuegos, la cual debería encargarse de las distribución y control de la venta de las versiones físicas, accesible con más facilidad en títulos y estudios con gran estimación de ventas o ya establecidos en el mercado. De este modo, las plataformas consideradas son: Steam[34], la cual cuenta con un alto porcentaje de adopción en los consumidores de computadora atendiendo a estadísticas de (Pearson, 2011); y itch.io[35], una plataforma dirigida a pequeños estudios y desarrolladores independientes.

A continuación, se muestra la Tabla 3 con la proyección económica a tres años vista dividida en trimestres, así como su representación gráfica. Donde se muestra el balance a lo largo de este periodo y como se encontrarían las cuentas del equipo a raíz del modelo de negocio elegido, las decisiones tomadas, las plataformas en las que se comercializará y el número de gente que se encargará de terminar con su desarrollo tras el segundo MVP.

Es importante remarcar la fórmula utilizada para calcular los ingresos por licencia: por una parte, están divididos en las principales plataformas de venta consideradas, valorando que en Steam se realizarán el 85 % de las ventas totales, y, el resto, por consiguiente, en itch.io; por otra parte, se debe tener en cuenta las comisiones de venta que se lleva cada plataforma, siendo de un 30 % por cada venta en Steam según (Valentine, 2018) y del 10 % en itch.io, aunque esta plataforma permite la asignación libre de esta comisión (itch.io, 2020), pudiendo incluso ser nula. Por tanto, la fórmula para cada plataforma tiene en cuenta las licencias estimadas a vender en ese trimestre multiplicado por el precio del producto (19,99 €), la comisión de venta y el porcentaje de ventas dirigido a esa plataforma.

[34] Más información en <https://es.wikipedia.org/w/index.php?title=Steam&oldid=126892293>
[35] Más información en <https://es.wikipedia.org/w/index.php?title=Itch.io&oldid=119199832>

Tipos de licencias vendidas	**Trim. 1 Año 1**	**Trim. 2 Año 1**	**Trim. 3 Año 1**	**Trim. 4 Año 1**	**Trim. 1 Año 2**	**Trim. 2 Año 2**	**Trim. 3 Año 2**	**Trim. 4 Año 2**	**Trim. 1 Año 3**	**Trim. 2 Año 3**	**Trim. 3 Año 3**	**Trim. 4 Año 3**
Licencias	0	0	0	0	2500	500	10000	3000	200	200	100	50
Ingresos Trimestrales												
Licencias Steam	0,00 €	0,00 €	0,00 €	0,00 €	29.735,13 €	5.947,03 €	118.940,50 €	35.682,15 €	2.378,81 €	2.378,81 €	1.189,41 €	594,70 €
Licencias Itch.io	0,00 €	0,00 €	0,00 €	0,00 €	6.746,63 €	1.349,33 €	26.986,50 €	8.095,95 €	539,73 €	539,73 €	269,87 €	134,93 €
Total Ingresos	0,00 €	0,00 €	0,00 €	0,00 €	36.481,75 €	7.296,35 €	145.927,00 €	43.778,10 €	2.918,54 €	2.918,54 €	1.459,27 €	729,64 €
Gastos Trimestrales												
Gastos GitHub	13,23 €	13,23 €	13,23 €	13,23 €	13,23 €	13,23 €	0,00 €	0,00 €	0,00 €	0,00 €	0,00 €	0,00 €
Licencia Steam Direct	0,00 €	0,00 €	0,00 €	0,00 €	88,20 €	0,00 €	0,00 €	0,00 €	0,00 €	0,00 €	0,00 €	0,00 €
Mercadotecnia	0,00 €	0,00 €	0,00 €	0,00 €	1.000,00 €	500,00 €	100,00 €	0,00 €	0,00 €	0,00 €	0,00 €	0,00 €
Empleado	18.000,00 €	18.000,00 €	18.000,00 €	18.000,00 €	18.000,00 €	18.000,00 €	18.000,00 €	18.000,00 €	0,00 €	0,00 €	0,00 €	0,00 €
Total Gastos	18.013,23 €	18.013,23 €	18.013,23 €	18.013,23 €	19.101,43 €	18.513,23 €	18.100,00 €	18.000,00 €	0,00 €	0,00 €	0,00 €	0,00 €
Personal Contratado	3	3	3	3	3	3	3	3	0	0	0	0
Resultado Trimestral	-18.013,23 €	-18.013,23 €	-18.013,23 €	-18.013,23 €	17.380,32 €	-11.216,88 €	127.827,00 €	25.778,10 €	2.918,54 €	2.918,54 €	1.459,27 €	729,64 €
Resultado Trimestral Acumulado	-18.013,23 €	-36.026,46 €	-54.039,69 €	-72.052,92 €	-54.672,60 €	-65.889,48 €	61.937,52 €	87.715,62 €	90.634,16 €	93.552,70 €	95.011,97 €	95.741,60 €

Tabla 3. Proyección económica a tres años. Elaboración propia.

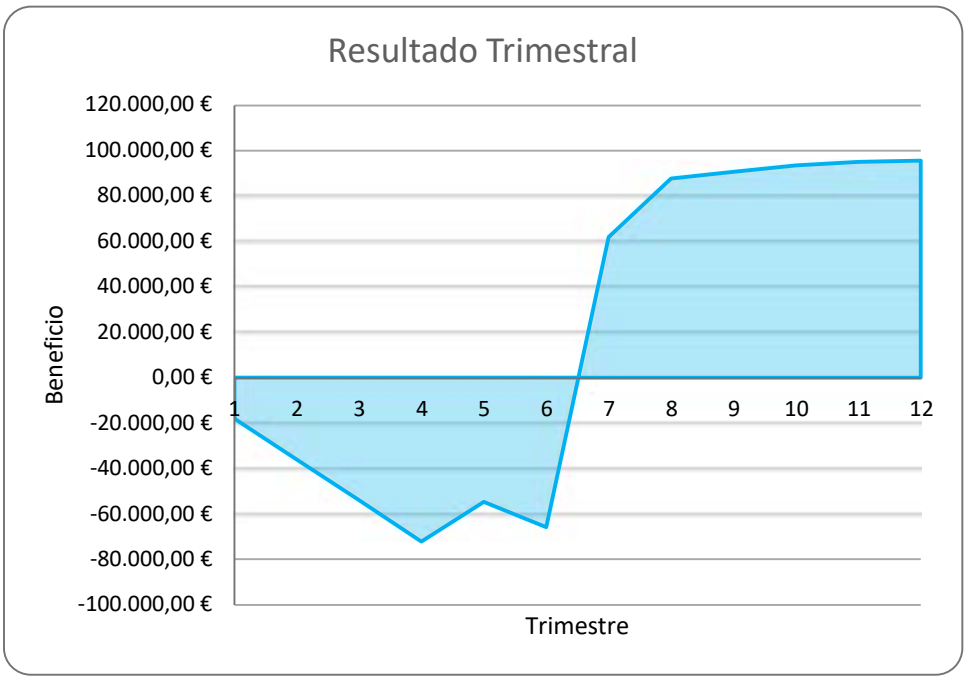

Figura 8. Gráfica sobre el resultado trimestral dividido en trimestres. Elaboración propia.

Para financiar el proyecto, se plantea el uso de la plataforma en línea Kickstarter, cuyo funcionamiento se basa en la realización de campañas para apoyar a un producto en desarrollo. Las campañas de entre 1 a 60 días, basan su funcionamiento en el cumplimiento de una meta impuesta por el equipo, la cual ha de alcanzar para poder obtener la cuantía conseguida. En caso de no llegar a esa cantidad en el periodo seleccionado, no se obtendría ningún ingreso. En nuestro caso, y como se puede comprobar en la Figura 8, nos planteamos una campaña de 30 días con una meta de 75.000 €. Esta elección se debe a que, con un tiempo de desarrollo estimado de un año y 9 meses, el total acumulado tendría un máximo de 72.052,92 €. Por lo que la cifra impuesta se justifica con el pago de los sueldos de los trabajadores, la necesidad de pagar la licencia necesaria para publicar un juego en la plataforma digital Steam, las tasas a pagar a Kickstarter por haber percibido ingresos a través de su plataforma y tener así un margen para posibles imprevistos, como la posible necesidad de un implicado más en la finalización del juego.

El desarrollo de los primeros MVP supondrá la inclusión de los aspectos clave del proyecto, conocido comúnmente como *vertical slice*[36], en el cual se desarrollan por completo las funcionalidades necesarias para un nivel, tal que se pueda jugar en su totalidad como si formara parte del producto final. Con esta técnica se fuerza al equipo de desarrollo a describir e implementar los sistemas base (sistemas de cámara, inteligencia artificial, diálogos, animaciones, entre otros), así como un marco operativo y funcional donde conectar dichos sistemas, de modo que, al finalizar este nivel parte de la *vertical slice*, la creación de los

[36] Más información en <https://en.wikipedia.org/w/index.php?title=Vertical_slice&oldid=962437839>

subsecuentes niveles se va a ver beneficiada de este trabajo existente y va a ver reducido el tiempo de desarrollo. Por estas razones se estima que el tiempo de desarrollo para terminar el producto, al finalizar el segundo MVP, va a ser de año y medio, y además durante este periodo el equipo de desarrollo va a ser más reducido, de tres personas. Al finalizar esta etapa, el desarrollo se puede dar como finalizado, por lo tanto, los trabajadores podrán ser recolocados en otros proyectos (que se encontrarán en la fase de preproducción o en proceso de salir de esta), tal como se puede apreciar a partir del tercer trimestre del segundo año.

Examinando con más detalle el desglose de gastos e ingresos mostrados, se puede comprobar que, durante el primer año, el trabajo se centra en el desarrollo del producto, por lo que los gastos son derivados de las remuneraciones de los empleados, en concreto de pagar tres empleados un sueldo de 2.000 €, y del coste de herramientas para control de versiones GitHub LFS[37] de 5 $ mensuales, necesario para la colaboración conjunta entre miembros del equipo de trabajo. De este modo, al finalizar el primer año se tendría un balance negativo de 72.052,92 € sin la inversión inicial de Kickstarter, suponiendo al pico de perdidas.

Durante el segundo año se empezarán a realizar los preparativos para llevar el juego a las plataformas de juego y tiendas digitales, en concreto se debería abonar la tasa única de entrada en la tienda de Steam de 100 $ (en el momento de redacción la conversión es a 88,20 €) extraído de (Steam, 2020). Con esta entrada temprana, se podría abrir un periodo de preventa en la que los potenciales consumidores pueden pagar para asegurarse la adquisición del producto una vez esté finalizado y disponible en la plataforma, de este modo se pueden conseguir ingresos adicionales de consumidores interesados que ahora están interesados en el producto o no pudieron participar en la campaña de Kickstarter. Con este método se estima obtener 3000 licencias adicionales, vendidas al precio habitual de 19,99 €. Además, la campaña de venta y preventa se verán ayudadas por la visibilidad producida por la campaña de mercadotecnia a la que se dedicará un total de 400 € repartidos sobre todo en los primeros trimestres: repartidos en la creación de videos promocionales y cumplir con las recompensas de los patrocinadores de la campaña de Kickstarter. A partir del tercer trimestre, el juego habría salido formalmente a la venta, con lo que se pronostica el mayor pico de ventas durante las primeras semanas, de unas 10000 licencias. Cabe señalar que se mantienen los tres empleados hasta final del año para cubrir el mantenimiento y resolución de los mayores problemas durante los primeros meses de lanzamiento. Adicionalmente, durante el cuarto trimestre se prevé la venta de unas 3000 licencias más.

[37] Servicio en línea para el guardado y control de versiones de grandes archivos para su favorecer su fácil uso por diferentes colaboradores. Especialmente útil para archivos de imágenes y modelados, comunes de proyectos de videojuegos.

Durante el tercer año, los trabajadores ya habrían sido reubicados a otros proyectos y el producto existente en el mercado habría alcanzado un nivel de madurez y estabilidad aceptable como para dejar de lado su mantenimiento. Por tanto, en este periodo solo habría ventas, ayudadas sobre todo por temporadas de rebajas y ofertas presentes en las plataformas de venta, las cuales suelen atraer un mayor número de usuarios activos y potenciales compradores que en otros periodos del año, aunque cabe destacar que las ventas trimestrales serían de menor cantidad y decrecientes debido a la alta competencia y a la pérdida de las cualidades novedosas del producto.

Como conclusión de la proyección económica deducimos que, dado la naturaleza del producto, solo generaría beneficios una vez sea lanzado al mercado, decreciendo con el tiempo el dinero que este puede producir, a diferencia de los juegos como servicio que generan ingresos a lo largo de su vida, dedicándose el equipo únicamente a sacar nuevo contenido, pero manteniendo un equipo de trabajo para poder producir nuevo contenido periódicamente y de forma estable. En nuestro caso, el lanzamiento de este producto, aunque no aporte un gran beneficio, supone la entrada del equipo al mercado y, por consiguiente, abre la posibilidad a la creación de otros productos de tamaño similar, que en conjunto sí puedan ofrecer una fuente de ingresos estable. Esta estrategia también serviría, en vistas a futuro y en base a sus ventas, para financiar nuevos proyectos de mayor envergadura que requieran de más tiempo y dedicación, dejando de depender de técnicas de microfinanciación colectiva como el mencionado Kickstarter.

3.4 Lean Canvas

El Lean Canvas es una herramienta que permite analizar el modelo de negocio escogido para aumentar las probabilidades de tener éxito. Esta técnica consiste en enumerar los valores de un producto o de una idea de negocio, tal que queden claros en una tabla que aspectos del producto son diferenciadores y únicos en el mercado en cuestión, así como las diferentes formas en la que se puede dar a conocer el producto. Además, permite diferenciar el público objetivo y el tipo de consumidor que pueden servir para probar el producto en sus fases más tempranas. La sección de costes e ingresos se puede observar como un resumen de los factores más relevantes destacados en el apartado de proyección económica. En definitiva, ofrece, de una forma general y compacta, las características clave del producto con el objetivo de compartir y difundir la idea de negocio a posibles interesados.

2 Problema

Se encontró en el mercado escasos juegos con temáticas éticas, así como universos que apuesten por una narrativa trabajada en este aspecto o que protagonistas no humanoides.

Cada vez hay más tendencia en la industria de hacer los juegos más largos y con más contenido, suponiendo un gran esfuerzo para el jugador terminar la historia.

Tendencia a realizar juegos como servicio a los que hay que dedicarlos mucho tiempo para progresar.

4 Solución

Historia ambientada en un mundo distópico, donde tu misión es paliar las consecuencias de la crisis climática.

Contenido cerrado y de corta duración, que se pueda finalizar de una sentada.

8 Métricas

- Valoraciones de la página del producto en sus plataformas de venta (Steam y itch.io).
- Formularios y cuestionarios realizados en ferias.
- Impacto social en redes sociales.

3 Proposición de valor

Videojuego narrativo con temática del cambio climático.

La historia busca concienciar y ridiculizar a partes iguales los vicios de nuestra sociedad acerca de la crisis climática, así como presentarnos una ciudad distópica habitada por helados.

Estilo artístico *low-poly*[38] y desenfadado.

Inclusión de mecánicas simples, pero con gran variedad de uso.

9 Ventaja competitiva

Entorno de juego y propuesta original, ya que se ambienta en un mundo habitado por helados.

5 Canales

- Portales de distribución de juegos (itch.io, Steam)
- Ferias y eventos (Feria de proyectos de estudiantes)
- Redes sociales (Instagram, Twitter)

1 Clientes

Público objetivo:
Se busca un perfil casual, es decir, gente interesada en videojuegos o en entrar en el medio, y por tanto no necesariamente muy habilidosos con los mismos.

Interesados en la problemática social del cambio climático.

Interesados en experiencias dirigidas a un solo jugador.

Early adopters[39]:
Familiares, amigos, patrocinadores relevantes y activos de la campaña de Kickstarter

7 Costes

Sueldos de trabajadores contratados por la empresa para finalizar el desarrollo.
Costes de publicidad para mejorar la visibilidad y aumentar la cantidad de potenciales compradores.
Licencias de herramientas de desarrollo como GitHub LFS.
Comisiones para la publicación de juegos en plataformas de venta en línea, concretamente Steam Direct.

6 Ingresos

Ventas, tanto provenientes de campañas de preventa como ya en su salida al mercado.
Ingresos derivados de la campaña de microfinanciación en Kickstarter.
Donaciones provenientes de la campaña de Kickstarter o posterior a ella.
Financiación o aportación adicional de los componentes del grupo.

Tabla 4. Lean Canvas realizado para el proyecto. Elaboración propia.

[38] Estilo o técnica de modelado basada en un bajo uso de polígonos a la hora de construir una figura. Más información en <https://en.wikipedia.org/w/index.php?title=Low_poly&oldid=973057569>

[39] Clientes que se espera sean los primeros en probar un nuevo producto

3.5 Conclusiones

Tras finalizar la evaluación de la idea original a través de diferentes técnicas, el equipo obtuvo las siguientes consideraciones: el equipo es consciente de los puntos fuertes del videojuego en cuestión, especialmente su historia y entorno únicos, por lo que serían aspectos destacables en las campañas de publicidad que se enfatizarían gracias a nuestro equipo de diseñadores, quienes se encargarían de que estos aspectos resalten frente a los de la competencia. Sin embargo, la baja experiencia en el mercado del equipo es también uno de los aspectos que más preocupan, ya que la curva de aprendizaje suele ser larga debido a la combinación e integración de herramientas de diferente ámbito.

Por otro lado, el género del juego a desarrollar no está entre los más rentables y la competencia para cualquier tipo de juego en la actualidad no permite tampoco a un estudio recién creado depender de su primer producto, por lo tanto, se deberá recurrir a la ayuda de campañas de financiación colectiva para poder cubrir los gastos necesarios para finalizar el producto. Además, será necesario, una vez sacado al mercado, la planificación de otros proyectos de similar envergadura con el objetivo de conseguir un grupo de productos que permitan: por una parte, sostener al equipo y poder contratar nuevo personal que ayude a realizar trabajos de mayor envergadura o con mejores acabados; y, por otro lado, subvencionar los subsecuentes proyectos, disminuyendo así la dependencia en la plataforma de financiación colectiva. En cualquier caso, el aumento de juegos realizados conseguirá que el estudio consolide un lugar en el mercado, lo que haría mucho más fácil el crecimiento de la empresa, pudiendo incluso llegar a realizar desarrollos más largos sin apenas percibir deudas y también atraer así a editoras con el porfolio de los productos creados pudiendo plantearnos la comercialización de ejemplares físicos gracias al apoyo de estas.

4. Desarrollo de la idea de negocio

El desarrollo de este proyecto ha tenido una duración de nueve meses, periodo de tiempo en el que también se incluyen las distintas pruebas con usuarios además del desarrollo de este. El cual se inició en la asignatura de Introducción a la Programación de Videojuegos; donde se nos proporcionó la oportunidad de juntarnos con otros alumnos del grado de Diseño y Tecnologías Creativas que cursaban la asignatura de Desarrollo de Videojuegos. Tras una sesión en la que se presentaron las distintas ideas propuestas por los alumnos de ambas asignaturas, cada uno eligió la que más le gustaba, para posteriormente tras los resultados unirse al equipo asignado para desarrollar la idea, con el objetivo de no solo ser evaluado en la asignatura sino también presentarlo a la Feria de proyectos de estudiantes de la ETSINF, donde tendríamos nuestra primera toma de contacto con posibles usuarios fuera del aula.

Tras reunirnos, el equipo resultante comenzó a idear que podría hacer un polo, quien sería nuestro protagonista; se comparó con distintos juegos para ver donde se podía sacar inspiración y además desarrollar elementos que lo diferenciaran del resto para hacerlo atractivo. Después de varias reuniones se asentaron algunas mecánicas básicas, cómo sería el mundo planteado y qué características tendría. Durante el primer mes se elaboró un Kanban[40] usando la herramienta colaborativa Trello[41] con las diferentes características a implementar en el primer MVP, de esa forma se podría realizar un seguimiento de las tareas a realizar, además se asignaron responsabilidades a cada integrante, ya que cada uno nos centraríamos en una cosa. El proyecto avanzó a buen ritmo aun estando con carga de trabajo en otras asignaturas a las que había que dedicarles tiempo.

Para la muestra en la feria se elaboró una encuesta que pudiera proporcionar datos significativos acerca de la experiencia de los usuarios con la demo desarrollada, sumado a los comentarios durante la misma feria; estos hicieron al equipo reflexionar acerca de la experiencia y los resultados, obteniendo como resultado la necesidad de replantear la disposición del mapa a mostrar al usuario y como este interactuaría con el entorno. Tras esto, se dejó un tiempo para reflexionar al equipo y decidir si seguir con el proyecto como trabajo final de grado.

Después del periodo de reflexión nos volvimos a reunir, con un integrante menos, para decidir qué cambios al diseño aplicábamos al nivel a desarrollar. Luego de una serie de reuniones, nos pusimos manos a la obra, sin embargo, dado a la pandemia del COVID-19[42], tuvimos que cambiar nuestro modo de trabajo y desde casa coordinarnos y hacer las reuniones con otra manera, realizando las sesiones de trabajo conjuntas a través de videollamada y resolviéndonos

[40] Más información en <https://www.atlassian.com/agile/kanban/boards>
[41] Más información en <https://es.wikipedia.org/w/index.php?title=Trello&oldid=121097153>
[42] Más información en <https://www.who.int/es/emergencies/diseases/novel-coronavirus-2019>

las dudas mutuamente como podíamos. Desde ese momento, día tras día trabajamos hasta conseguir una versión estable a compartir entre las personas de nuestro entorno a través de sistemas de mensajería, no pudiendo hacer una muestra al público en condiciones resolviendo dudas y explicando los posibles fallos que pudiesen ocurrir.

4.1 Mapa de características

Al inicio del proyecto, el equipo de trabajo discutió las características que iban a conformar el videojuego, con este propósito se elaboró una lista de unidades de trabajo[43], de ahora en adelante UT, tal que pudieran formar parte de la bolsa de trabajo[44] del proyecto, la cual se puede ver en la Figura 9. Basándonos en la jugabilidad deseada para el juego, se incluyeron varias UTs relacionadas con el movimiento y las mecánicas de juego (movimiento, salto, sigilo, forma de sorbete, cambio de palos para conseguir nuevas habilidades). También se incluyeron tareas relativas a los sistemas con el que el jugador podría interactuar dentro del entorno de juego (diálogos, inventario, misiones, cámara, patrullas y seguimiento de enemigos, zonas de calor y muerte), así como los menús que posibilitarían navegar hacia los niveles y controlar el juego (menú de pausa, menú principal, menús de configuraciones, guardado y cargado de partidas). Con el objetivo de hacer el juego más accesible y ayudar a nuevos jugadores se incluyeron diversas UTs: soporte para idiomas castellano e inglés, cinemáticas de introducción, cinemáticas de explicación, minimapa[45] y una flecha para indicar el siguiente objetivo.

El desarrollo de los niveles que compondrán el juego en un futuro se representó como las tareas de niveles exteriores y niveles interiores (entendidos en referencia a la nevera, un punto central en el juego donde el jugador podrá entrar), sin embargo, el desarrollo del primer nivel (exterior) ya se ha colocado como UT propia al ser el más relevante para la evaluación de la idea inicial, concretamente dividido en mapa nivel uno, eventos nivel uno y personajes nivel uno. Para tener en cuenta la creación de componentes artísticos y modelados genéricos, se incluyeron UTs para objetos que aparecerían en la mayoría de los niveles (objetos del entorno, objetos clave) y paralelamente, tareas para interactuar con los componentes del entorno de juego (interacción con personajes). Adicionalmente, se elaboraron UTs de componentes para establecer el tono del juego y que contribuyen a su aspecto artístico (posprocesado, animaciones, efectos del entorno, pisadas en la nieve, estilos de texto y voces de diálogo).

[43] Entendido como una parte del producto o servicio requerido. Característica de un producto cuya finalización aporta valor a quien al usuario.
[44] Conjunto de tareas que se pretende realizar durante el desarrollo de un proyecto.
[45] Mapa de pequeño tamaño empleado en videojuegos para situar al jugador en un espacio

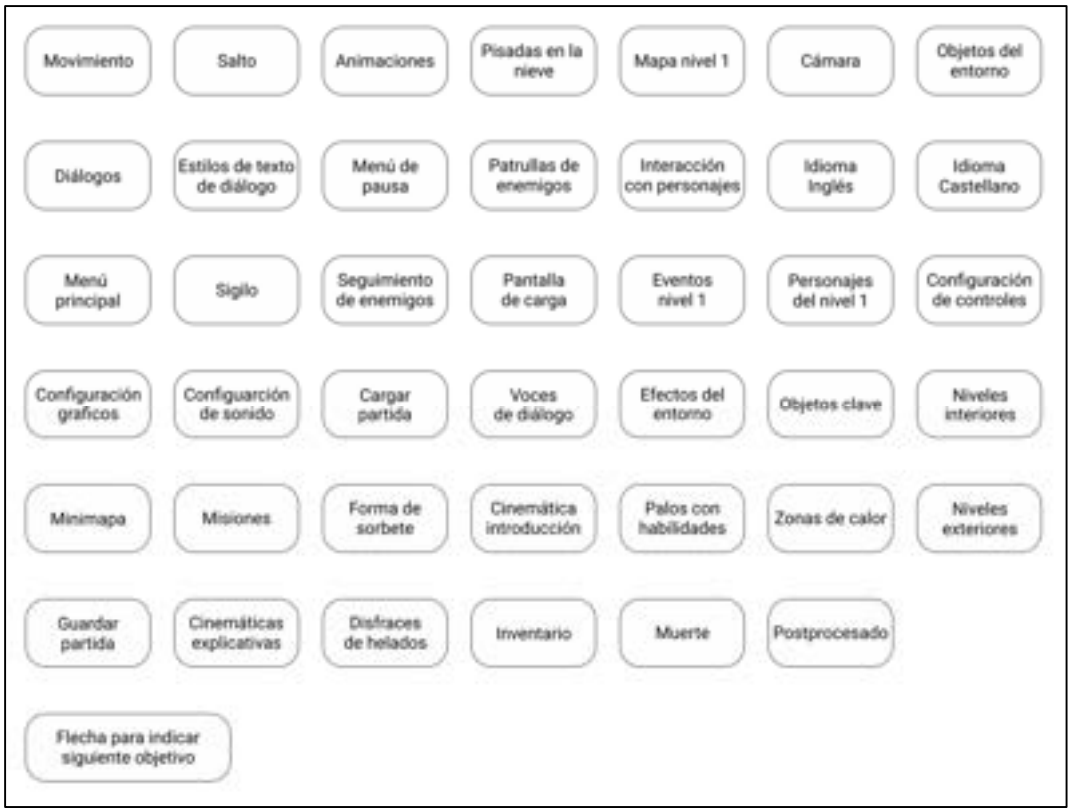

Figura 9. Mapa de características. Elaboración propia.

4.2 Primer Minimum Viable Product

Las características seleccionadas a desarrollar para el primer MVP fueron las mostradas en la Figura 10. Como se puede comprobar, aquí se incluyeron las mecánicas básicas, como andar, saltar y agacharse con las animaciones necesarias, así como la posibilidad de diálogo con los distintos personajes que aparecían en el entorno y el comportamiento de los enemigos. Por otro lado, también se desarrolló una primera versión de los menús de pausa e inicio, para que los usuarios que fueran a probar el producto pudieran hacer algún pequeño ajuste y acomodar los controles a sus necesidades, o pausar el juego para realizar alguna pregunta al equipo sin temer por que el juego terminase. Para evaluar la viabilidad de las características novedosas se incluyeron la creación de los personajes para el primer nivel, el estilo de texto y creación de objetos del entorno, con el objetivo de empezar a establecer el carisma del juego y comprobar el atractivo de este desde su inicio. También se incluyeron factores de riesgo como el efecto de pisadas en la nieve, el cual también contribuiría a la estética del juego, pero, por experiencia anterior de los componentes del equipo, podría suponer un problema en su implementación.

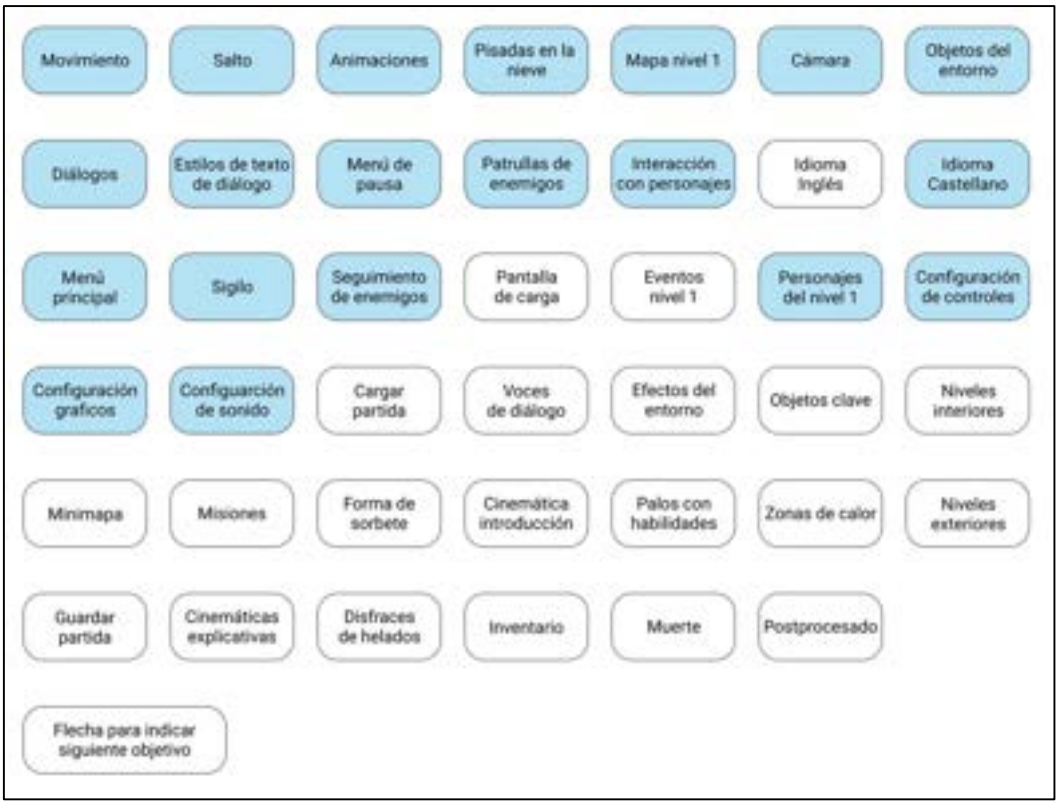

Figura 10. Mapa de características seleccionadas para el primer MVP (marcadas en azul). Elaboración propia.

4.2.1 Desarrollo del MVP

El nivel pensado para este periodo de desarrollo se iba a situar en un entorno exterior, tomando como punto de referencia del entorno la nevera, ya que es donde el inicio de la historia iba a tener lugar. En la Figura 11 se puede comprobar el aspecto que tendría el nivel, introduciendo la nevera, la ciudad exterior donde el juego iba a tener lugar, así como un vistazo a la ciudad abandonada, el que se consideraría el mapa para el segundo nivel. Una vez terminado el nivel, se pensó en reutilizar el mapa para crear un corto tutorial para la feria.

Figura 11. Arte conceptual para el nivel exterior. Elaboración propia.

En lo referente a la jugabilidad, se discutió como debería ser el movimiento del personaje y la libertad que tendría para moverse por el entorno, que animaciones serían necesarias para acompañarlo y como sería la cámara del juego. Decantándonos porque el jugador tuviera la posibilidad de moverse por todo el entorno y con una cámara libre en tercera persona que siguiera al jugador, solventando el problema de que la cámara atravesase objetos del entorno acercándola al jugador según la distancia del choque. Por otro lado, pese a que el usuario tiene total libertad de movimiento, estos tienen que ser delimitados de alguna forma para no romper los sistemas o vaya por lugares no diseñados para ello, entre otras cosas. Por ello, como se puede ver en la Figura 12, la zona se delimitó con tuberías y muros que dan forma al entorno y personajes no jugables (PNJ), los cuales, marcados con STOP, permanecerán bloqueando caminos de forma indefinida o hasta que ciertos eventos ocurran.

Figura 12. Mockup del mapa inicial con la distribución de enemigos, patrullas y puntos clave para el jugador. Elaboración propia.

Como se ha descrito, el efecto de la nieve era clave para transmitir una sensación de frio y, en general, marcar la estética del entorno. Para ello, se recurrió a uso de un sombreador (*shader*)[46] para aplicar el efecto de una capa de nieve que se acumula en el suelo[47], así como las pisadas en el suelo disminuyendo esta capa y creando un rastro por donde pasan los personajes. Este efecto era eficaz en terrenos pequeños, sin embargo, para el mapa de este nivel el efecto era pobre e imperceptible, por lo que la demo compilada no tuvo este efecto y esta característica se mantuvo para el trabajo del siguiente MVP con el objetivo de finalizar su implementación.

En el apartado de los diálogos, se investigaron herramientas existentes y probadas que se habían usado para desarrollar juegos y con posibilidad de tener un guion independiente que se pudiera modificar por personas sin conocimiento necesario de programación. Finalmente se decantó por

[46] Código ejecutado en la unidad gráfica de una computadora que modifica la imagen procesada según se especifique en este. Más información en <https://es.wikipedia.org/w/index.php?title=Sombreador&oldid=122810336>
[47] El tutorial seguido por el equipo para introducirse en este concepto fue el disponible en <https://www.youtube.com/playlist?list=PL3POsQzaCw53KM74XVRXv2xyesLjngfbG>

Yarn Spinner[48], ya que los juegos creados con él tenían un estilo y mecánicas similares al nuestro. La idea original para la presentación del texto se puede comprobar en la Figura 13, donde el texto aparece como bocadillos flotantes sobre el personaje. Además, también se puede ver una idea inicial del indicador de diálogos, colocando una imagen sobre ellos por determinar para señalar al jugador que un personaje tiene una conversación disponible se debía. Estas ideas se refinaron en un segundo mockup (Figura 14), donde el texto se presentaría en una clásica caja de diálogos, a modo de centralizar la atención del jugador en un mismo punto; también se especificó el indicador para que contuviera tres puntos.

Figura 13. Primer mockup de indicadores de diálogo y presentación de diálogo con bocadillos. Elaboración propia.

Figura 14. Segundo mockup de indicadores de diálogo y presentación de diálogo en una caja de texto. Elaboración propia.

Por otra parte, para el menú de inicio (Figura 15), al ser una primera versión de prueba a enseñar al público, se optó por un estilo básico, situando de forma simple las funcionalidades que tenía, como ajustes de sonido, video, controles y juego sin más detalle, encapsulados dentro de opciones y la posibilidad de empezar una partida. De la misma forma, el menú de pausa (Figura 16) solamente contenía las opciones de reanudar partida, guardar y cargar partida, aunque no estaban desarrolladas, la opción de repetir el nivel y la de salir al menú de inicio.

[48] Herramienta disponible para Unity que ofrece un sistema de guardado y proporción de líneas de texto en base a un guion almacenado en un archivo externo. Más información en <https://yarnspinner.dev/>

Figura 15. Primera versión del menú de inicio. Elaboración propia.

Figura 16. Menú de pausa del juego. Elaboración propia.

4.2.2 Experimento

Durante la «Feria de proyectos de estudiantes» se compartió una encuesta con ocho preguntas y tres sugerencias a los que probaron el juego. Están preguntas estaban dirigidas a resolver dudas sobre los aspectos que creíamos eran los más atractivos (estética, temática, historia) y a poner énfasis en aspectos que podían haber resultado problemáticos (objetivos, duración, repetitividad). Las sugerencias planteadas permitían a los usuarios escribir de forma libre aspectos relacionados con aquello que más y menos había gustado. A continuación, se pueden ver los resultados de la encuesta, la cual fue contestada por ocho personas.

Pregunta 1.- ***¿Crees que te ha quedado clara la idea del juego?***

Dado que la temática del juego iba vinculada con un mensaje acerca del cambio climático, se elaboró la pregunta para comprobar si los jugadores la percibían o había que realizar algún cambio en el diseño del nivel y como la idea era representada en el entorno y los diálogos. Como se puede observar en la Figura 17, el 62,5 % comprendía la idea, sin embargo, quedaba claro que aun podía mejorar dado que un 37,5 % de los encuestados habían seleccionado la opción «no del todo».

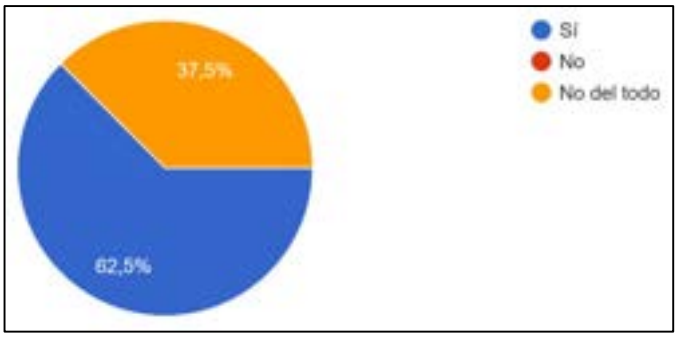

Figura 17. Encuesta MVP1, pregunta 1. Elaboración propia.

Pregunta 2.- ***¿Te ha resultado fácil comenzar a jugar o cambiar las opciones de juego?***

La segunda pregunta, relacionada con el menú de inicio y el de opciones, fue elaborada para comprobar si los usuarios podían navegar correctamente por estos, comprobando si era intuitivo su uso. Como se puede observar en la Figura 18, en este caso se obtienen resultados muy positivos, pues al 87,5 % de los usuarios les resultó fácil, en cambio a un 12,5 % no, lo que indicaba que estos cumplían su función.

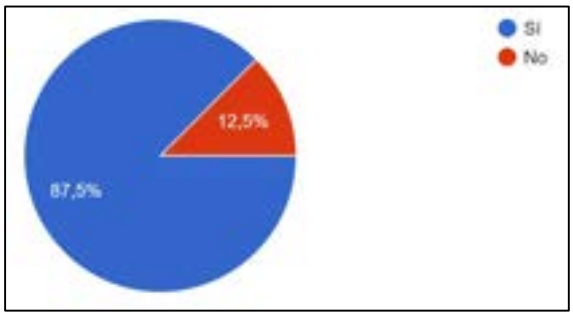

Figura 18. Encuesta MVP1, pregunta 2. Elaboración propia.

Pregunta 3.- ***¿Te has quedado con ganas de jugar al resto de niveles del juego?***

La tercera pregunta estaba relacionada con el interés general del producto, comprobando si los encuestados quisieran jugar al resto de niveles del juego. Así podríamos comprobar si la idea podía mantener su frescura y mantener al jugador interesado. Como se puede observar en la Figura 19, todos los encuestados se quedaron con ganas de probar más niveles del juego, resultado que el equipo recibió como un éxito.

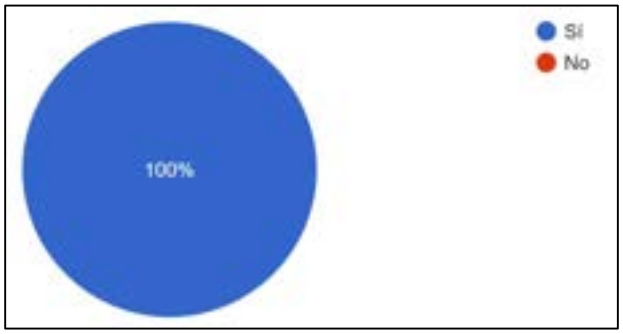

Figura 19. Encuesta MVP1, pregunta 3. Elaboración propia.

Pregunta 4.- ***¿Cuántas horas estarías dispuesto a invertir en un posible juego final?***

La cuarta pregunta estaba relacionada con la duración del juego, preguntando cuantas horas estarían dispuestas a jugar a la versión final del producto. Como se puede observar en la Figura 20, esta pregunta fue más diversa, ya que se ofrecían diez opciones. La mayoría de los encuestados preferirían jugar entre una y cuatro horas, mientras que una minoría eligieron una duración más larga (ocho y diez horas, respectivamente). Con esto podemos comprobar que el juego tendría éxito como una experiencia corta, aunque puede que la gente acostumbrada a jugar a estos juegos preferiría una mayor duración.

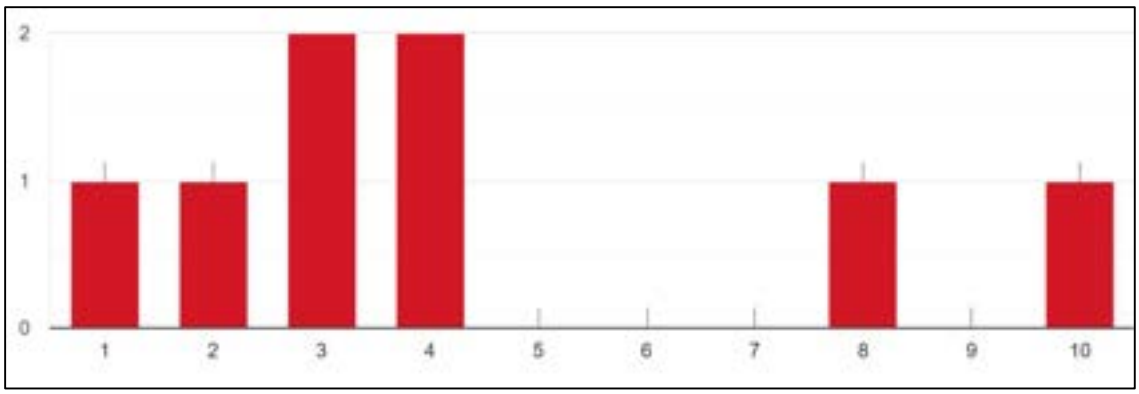

Figura 20. Encuesta MVP1, pregunta 4. Elaboración propia.

Pregunta 5.- ***¿Crees que se puede hacer repetitivo/aburrido al poco tiempo?***

Esta pregunta fue realizada para comprobar si el conjunto de las mecánicas elegidas para el juego junto con los diálogos, eran lo suficientemente atractivas para que el jugador siguiera jugando durante un intervalo prolongado de tiempo. Como se puede observar en la Figura 21, esta pregunta presenta diversidad en sus respuestas, la mayoría de los encuestados seleccionaron la opción de que podría hacerse repetitivo o aburrido al cabo del tiempo aunque comprendiendo que se trata de un juego de corta duración, por lo que en caso de aumentar el número de zonas, se tendrá que tener en cuenta que no pueden tener una gran extensión que alarguen más de la cuenta la duración del juego.

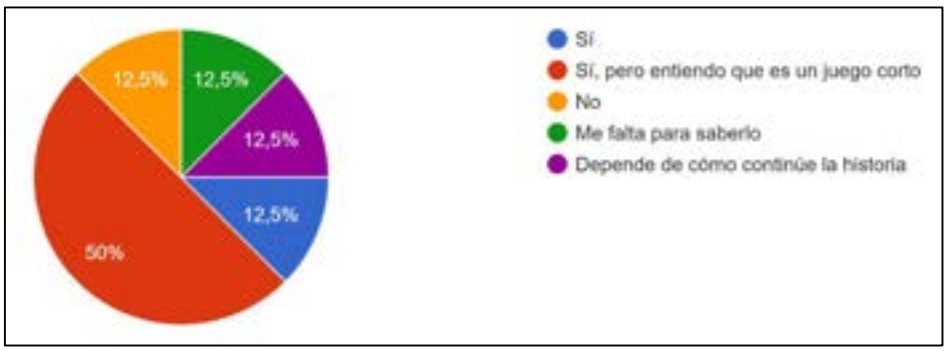

Figura 21. Encuesta MVP1, pregunta 5. Elaboración propia.

Pregunta 6.- ***¿Estarías dispuesto a terminar el juego solo por continuar con la historia (conspiración de la nevera, enfrentamiento con el alcalde, …)?***

La sexta pregunta estaba relacionada con la historia. Dado que la demo disponible solo mostraba una pequeña porción de la historia en comparación con los planes previstos, se tuvo que explicar a los participantes los antecedentes del personaje y que tipo de historia era esperable en los siguientes niveles. De modo que se preguntó si estarían dispuestos a terminar el juego solo por continuar la historia, para comprobar el atractivo de la historia. Como se puede observar en la Figura 22, en este caso se obtuvieron resultados muy positivos, dado que el 87,5 % de encuestados indicaron que sí estarían dispuestos a seguir y terminar el relato, reforzando el encanto y originalidad de la historia.

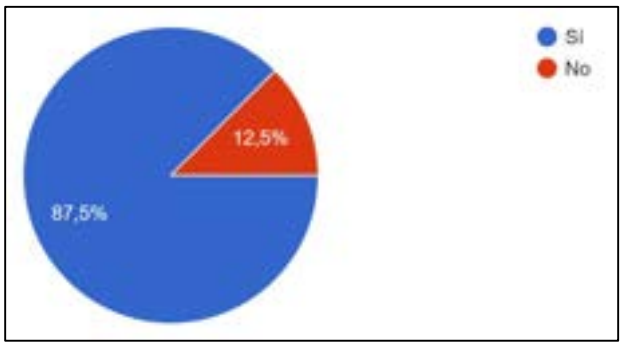

Figura 22. Encuesta MVP1, pregunta 6. Elaboración propia.

Pregunta 7.- ***¿Recomendarías a otra gente jugar a este juego?***

La séptima pregunta era sobre si recomendarían el juego a otra gente, poniendo a prueba el factor de la difusión del juego, el cual mejoraría la base de posibles interesados sin invertir en recursos para su difusión. Como se puede observar en la Figura 23, el 62,5 % de los encuestados recomendaría el juego, aunque el 25 % solo se lo recomendaría a fans del género, de modo que, en general, se demuestra que el producto podría gustar a bastante gente y ser fácilmente comunicado a otros interesados, aunque algunos consideran que solo interesaría a gente acostumbrada al género y juegos similares.

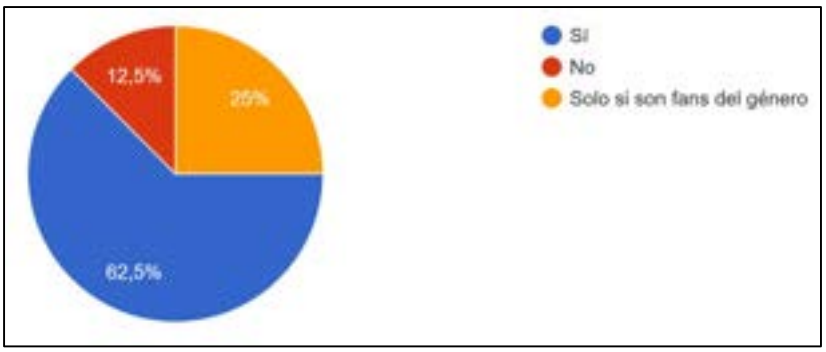

Figura 23. Encuesta MVP1, pregunta 7. Elaboración propia.

Pregunta 8.- ***¿Has jugado a otros juegos como este?***

Uno de nuestros puntos fuertes es la originalidad del producto y su estética, sin embargo, las mecánicas son conocidas por jugadores del género de puzle y acción, por lo que con esta pregunta quisimos conocer como de similar podían ver los usuarios nuestro producto con otros juegos del mismo género. Como se puede observar en la Figura 24, el 75 % de los encuestados han jugado a juegos similares o algo relacionados, esto nos muestra que no es un juego nicho y puede interesar a una gran cantidad de público.

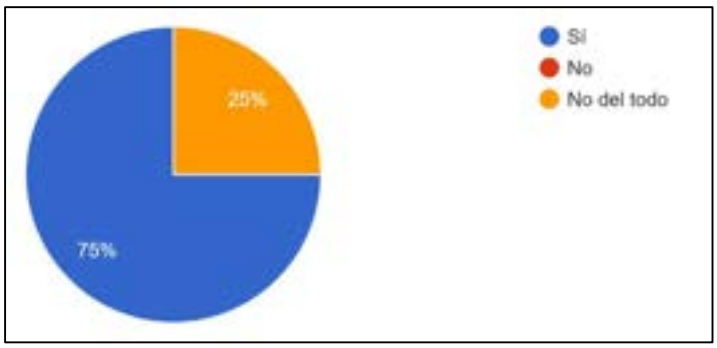

Figura 24. Encuesta MVP1, pregunta 8. Elaboración propia.

Sugerencia 1.- ***¿Qué aspectos te han gustado del juego?***

En la primera de las preguntas pidiendo sugerencias, queríamos conocer que aspectos eran los que más habían gustado tras haber probado la primera versión del juego en aspectos generales. Así pues, se obtuvo como resultado que lo que más destacado fue la originalidad de la historia y la estética del juego, con frases como «La ambientación e historia» o «El diseño y los controles».

Sugerencia 2.- ***¿Qué aspectos no te han gustado y cómo crees que se pueden mejorar?***

La segunda sugerencia, en contraposición, hacía referencia a los aspectos negativos. El mapa obtuvo críticas por su extensión y la falta de contenido en él («El mapa es demasiado grande o confuso» o «Falta contenido, mapa muy grande»). La jugabilidad también estuvo presente con comentarios como «Los controles pueden dar fallos» o «lento de andar». De modo que se tuvo en cuenta para la siguiente iteración de desarrollo una revisión a la jugabilidad y una recreación del mapa para que fuera más pequeño y sus componentes más fácilmente accesibles.

Sugerencia 3.- ***Sugerencias y otras consideraciones.***

Para cerrar la sección de sugerencias, se preguntó a los usuarios que dieran su libre opinión acerca del juego sin ninguna pregunta en específico, para ver así que otros aspectos no incluidos en el juego echaban a faltar o que les sobraba. Como respuesta, se obtuvieron distintas sugerencias, principalmente destacando que la apuesta era interesante, pero que hacía falta más contenido y que aspectos de la interfaz, como que el texto en ocasiones no se ajustaba al cuadro de diálogos («El layout de los diálogos es confuso»).

4.3 Segundo Minimum Viable Product

Las características seleccionadas a desarrollar para este segundo MVP fueron las mostradas en la Figura 25. Para este periodo se planeó la inclusión de voces para los diálogos, la traducción de los textos al inglés, la introducción de los diversos eventos en el mapa, la creación de la zona de calor para delimitar la zona jugable y mejoras de rendimiento. En cuestión de mecánicas se añadiría un inventario al jugador, así como los objetos clave, la forma de sorbete del jugador y la posibilidad de morir para reiniciar el nivel. Para mejorar la comprensión también se seleccionaron la creación de una pantalla de carga y la inclusión de diferentes cinemáticas, tanto explicativas como relacionadas a eventos. Con el objetivo de mejorar la estética del juego, un aspecto apreciado en el anterior MVP, se consideró la inclusión de más efectos de entorno, como las partículas y el posprocesado, y actualizar el menú principal y el de pausa. Por otro lado, dado los resultados de la encuesta, se planificó modificar aspectos como el movimiento, el salto y el sigilo, que formarían parte de una reestructuración del movimiento. Se incluyeron también retoques al efecto de pisadas en la nieve, el cual no pudo ser finalizado y debería ser revisado para este periodo. Con el objetivo de atajar los problemas relacionados con la falta de dirección y magnitud del mapa, se planeó el rediseño de este para que no fuera tan amplio, así como adaptar la cámara al nuevo entorno y añadir nuevos personajes al nivel. Además, se modificaron las tareas de cargado y guardado de partidas para reflejar mejor el estilo de juego, de modo que se reemplazaron por una tarea de sistema de puntos de control[49], más conocidos como *checkpoints*, liberando al jugador de las tareas de control de guardados para tener una gestión más concisa de estos.

[49] Zonas en las que se guardará automáticamente la situación del jugador la primera vez que esta es alcanzada, sirviendo como punto de aparición en caso de cerrar el juego o perder la partida.

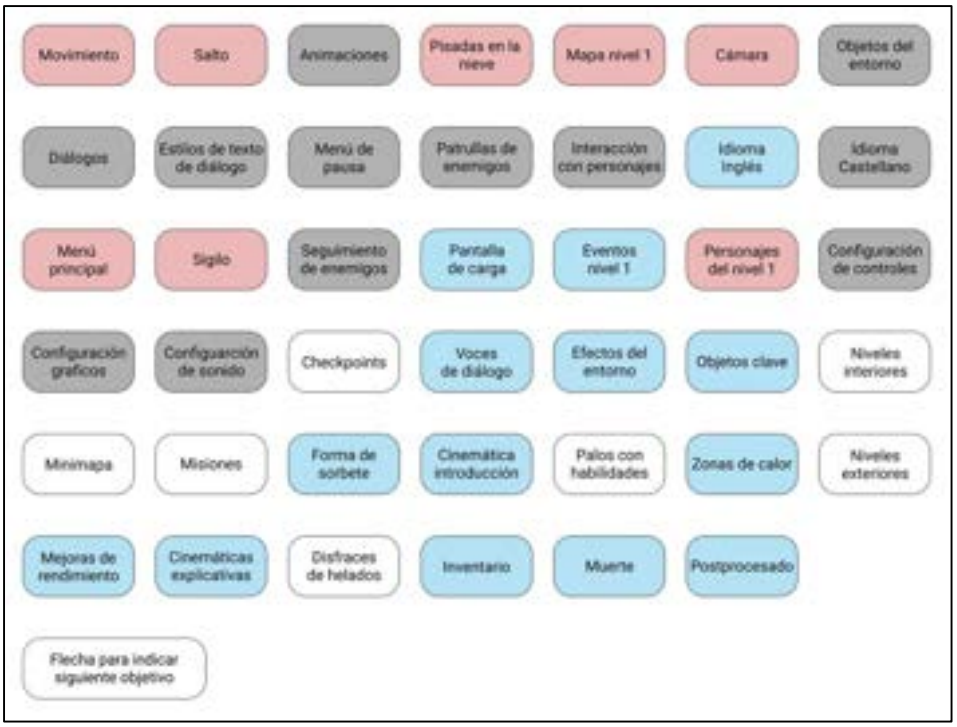

Figura 25. Mapa de características seleccionadas para el segundo MVP (en azul las nuevas características; en rojo las características a modificar o arreglar en base a los comentarios recibidos; en gris las características realizadas en el MVP anterior). Elaboración propia.

4.3.1 Desarrollo del MVP

Tras la feria de diciembre y los exámenes del mes de enero, no fue hasta casi mitad de febrero que empezamos a planear la remodelación del nivel que presentamos en la feria, como el movimiento, que es lo que había dado más problemas. Junto a esto, se pensó como se podrían adaptar los sistemas que ya se habían implementado en el juego y como se podrían incluir nuevos, añadiendo el inventario que, junto a los diálogos, serían el núcleo de la jugabilidad.

Figura 26. Mockup de la segunda versión del mapa. Elaboración propia.

Como se pude ver en la Figura 26, se planeó que el mapa dejara de ser una vasta extensión, sustituyéndolo por un mapa con una estructura de anillos separados por paredes verticales, que irían disminuyendo conforme la altura del mapa aumentara y acabando en la nevera. Cada anillo no sería perfectamente circular, ya que se situaron grandes surcos hacia el interior para dotar de irregularidad al entorno y darle profundidad, haciendo que no solo se pueda mover el jugador

por la zona exterior de los anillos. Debido a esto, el lado contrario al mapa quedaría vacío, visible en la zona marrón vacía de la Figura 27, dejando la cámara en tercera persona inservible, por lo que se decidió usar Cinemachine[50], una paquete opcional incluido en Unity que servía como extensión para la cámara, y las herramientas que nos proporcionaba, creando así un sistema basado en raíles por los que interpolará la cámara, estando situados en la zona exterior de cada anillo.

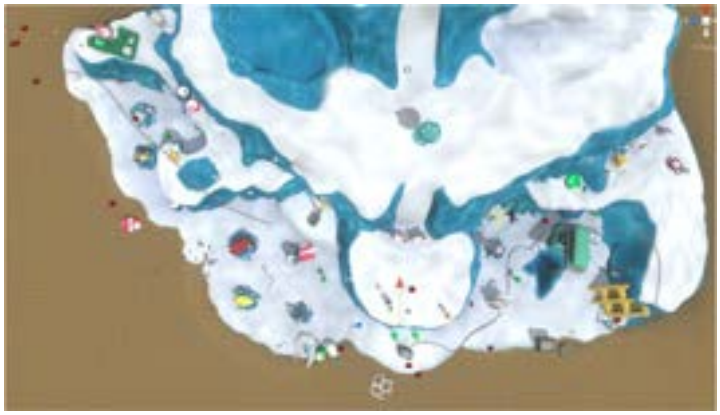

Figura 27. Zonas del mapa vistas desde arriba, incluyendo el yermo (zona marrón). Elaboración propia.

En cuanto al movimiento, dado que la primera versión estaba desarrollada a partir de un componente que no tenía en cuenta las físicas y no se tuvo en mente la habilidad de cambio de forma del jugador, nos vimos obligados a cambiarlo por completo de forma interna, separándolo además en diversos *scripts* y empleando un componente diferente que si las empleara. Además, se modificaron animaciones y se añadieron sonidos a algunas de estas.

Sobre la inteligencia artificial (IA), para este segundo MVP se mejoró la respuesta de las patrullas, evitando algunos fallos relacionados con las animaciones; se preparó un sistema de detección completo similar al de juegos de sigilo tradicionales con detección, distintos campos de visión y distintos tipos de patrullas y pausas en estas. Junto a esto, también se desarrolló un editor propio para su manejo y configuración.

Uno de los aspectos que iba a mejorar la interacción con el nivel y favorecer la exploración era la adición de ítems y un inventario del personaje. Tal como se puede ver en la Figura 28, este inventario sería presentado de una forma sencilla, con cada ítem representado por un icono. Además, a través de este menú también se podrían equipar los ítems que el personaje pudiera llevar (para este nivel el pico y la cuchara). Solo un ítem podría estar equipado a la vez, y en el menú del inventario sería marcado con un color azul.

[50] Más información en <https://unity.com/es/unity/features/editor/art-and-design/cinemachine>

Figura 28. Mockup de la interfaz para el inventario. Elaboración propia.

Con relación a las mejoras de rendimiento, por una parte, se disminuyó todo lo posible el número de polígonos de los elementos que conformaban el mapa, eliminando además las colisiones no necesarias y sustituyendo las que se podían por unas más simples, disminuyendo así la carga. Por otro lado, se reestructuró el código para que la comunicación entre componentes fuera más eficiente, además de hacer más fácil la creación de nuevos niveles desde cero, en cuanto a sistemas de juego se refiere. Para acomodarse a este nuevo diseño interno, se migraron las funcionalidades de los diálogos, y se retocaron sus componentes de interfaz para tener un estilo consistente con el resto del juego. El soporte para otros idiomas, concretamente el inglés, se llevó a cabo a través de una librería desarrollada por Unity (Unity Localizations[51]) que se encontraba en estado beta, pero totalmente funcional. Esta librería permitía actualizar el texto mostrado en pantalla en base al idioma seleccionado. Para localización del diálogo, sin embargo, se usaron las opciones internas de Yarn Spinner, de modo que se focalizara la edición del guion. Para acomodarse al estilo del videojuego, las voces de los personajes fueron sintéticas, asemejándose al estilo de voz de los personajes de *Animal Crossing*[52].

En cuanto al menú principal, este fue sustituido completamente por uno nuevo (Figura 29) en el que por un lado se mostraba en una pantalla las opciones principales y los ajustes en un dibujo pegado a la nevera, creando una transición entre estos y añadiendo efectos de distorsión de imagen a la primera, haciendo que estuviera integrado con la nevera. En cuanto al efecto de las pisadas en la nieve, también se mejoró para soportar terrenos grandes, de modo que el efecto conseguido durante el primer MVP fuese visible en el mapa actual. Cabe destacar, sin embargo, que este efecto no está presente en la demo compilada de este MVP debido a problemas técnicos con la exportación de Unity.

[51] Disponible en beta como paquete de Unity <https://docs.unity3d.com/Packages/com.unity.localization@0.6/manual/index.html>
[52] Más información en <https://es.wikipedia.org/w/index.php?title=Animal_Crossing_(serie)&oldid=128824132>

Figura 29. Menú principal de Frozen Out tras su rediseño. Elaboración propia.

4.3.2 Experimento

Al finalizar este MVP se realizó una prueba digital enviando un ejecutable del juego a distintos jugadores. Decidimos también incluir un enlace a una guía para que supieran qué hacer si se encontraban perdidos, así como un enlace a los controles por conveniencia. Para evaluar su experiencia, se les adjuntó nuevamente una encuesta de diez preguntas y tres sugerencias. Están preguntas estaban dirigidas a resolver dudas sobre los aspectos que habían sido revisados, como la reducción del mapa e historia más concisa, así como el movimiento y la cámara. También se planeaba evaluar los aspectos que habían tenido éxito (humor, estética) permanecían interesantes al público. En cuanto a las sugerencias, similarmente a la encuesta anterior, se permitió compartir libremente qué aspectos habían sido de su agrado y cuáles no. A continuación, se describen los resultados de la encuesta, la cual fue contestada por trece personas.

Pregunta 1.- ***¿Has podido saber cuál era el objetivo por cumplir en cada momento?***

Debido al rediseño del mapa, se preguntó a los encuestados si, gracias a los diálogos con los diferentes PNJs y las diferentes cinemáticas, les resultaba intuitivo conocer cuál debía ser su siguiente objetivo conforme los iban completando. Obteniendo con un 53,8 % de los votos que no habían podido saber el objetivo a seguir, como se puede ver en la Figura 30. Lo que significa que deberemos hacer énfasis en este aspecto y reforzarlo para mejorar la experiencia.

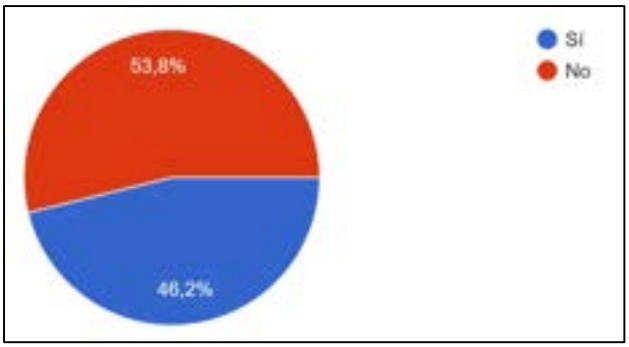

Figura 30. Encuesta MVP2, pregunta 1. Elaboración propia.

Pregunta 2.- ***¿Crees que ha sabido transmitir correctamente su idea y critica medioambiental?***

La segunda pregunta se hizo para comprobar si la crítica medioambiental había quedado clara durante el juego, la cual había sido insinuada en conversaciones de personajes y el entrono en sí. Como podemos observar en la Figura 31 más de la mitad de los encuestados considera que no se transmite del todo su idea y critica ambiental, de modo que el equipo decidió mejorar la representación de estos aspectos en más mecánicas, como cinemáticas o efectos.

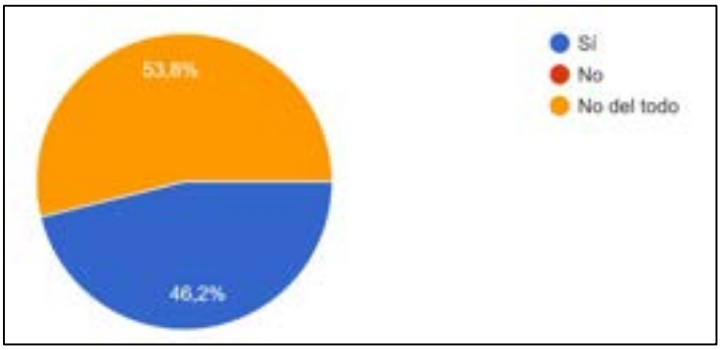

Figura 31. Encuesta MVP2, pregunta 2. Elaboración propia.

Pregunta 3.- ***¿Cómo de satisfactoria te ha resultado la partida?***

Otra vez, debido al cambio tan drástico en el mapa y los nuevos objetivos incluidos, se pidió a los encuestados que puntuarán del 1 al 10, su satisfacción tras haber completado todos los objetivos y haber acabado la partida. Como se puede ver en la Figura 32, había diferencia de opiniones, obteniendo una media de 6. De modo que, pese a que sabemos que no es una nota muy destacable, por comentarios de las sugerencias, conocemos que el principal problema era debido a la ausencia de puntos de control durante la demo.

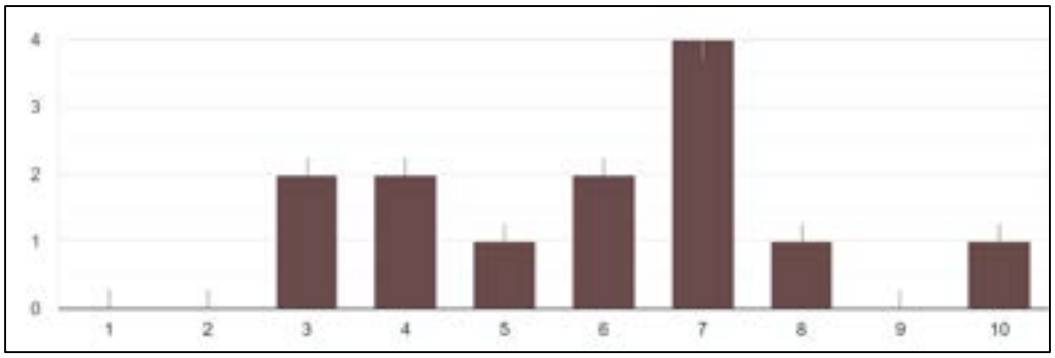

Figura 32. Encuesta MVP2, pregunta 3. Elaboración propia.

Pregunta 4.- ***¿Estarías dispuesto a terminar el juego solo por continuar la historia?***

La cuarta pregunta tenía el objetivo de conocer si se mantenía el atractivo de la historia y sus personajes. Como podemos observar en la Figura 33, el 86,4 % de la gente estaría dispuesta a terminar el juego por continuar la historia, de modo que se reafirma el interés del público en el entorno y narrativa presentados y demuestra que el esfuerzo en estos aspectos no debe ser mermado.

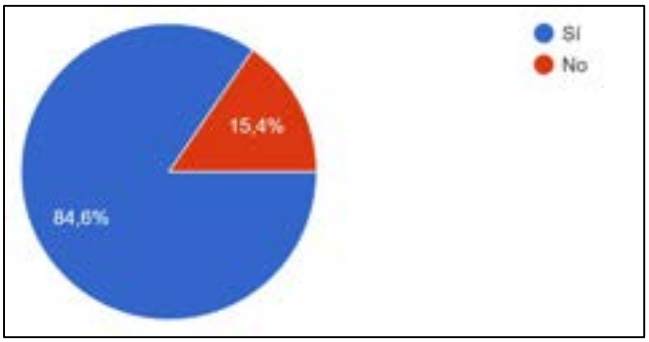

Figura 33. Encuesta MVP2, pregunta 4. Elaboración propia.

Pregunta 5.- ***Si probaste el juego en la feria de diciembre, ¿crees que se han resuelto los problemas que encontraste?***

Dado que se realizó un primer experimento y con el objetivo de intentar que alguna de las personas que participaron en el probaran la nueva versión del producto, realizamos esta pregunta para comprobar si desde su punto de vista, los problemas que percibieron se habían solucionado, obteniendo una respuesta afirmativa de todos los encuestados que participaron en el experimento anterior, tal y como se aprecia en la Figura 34.

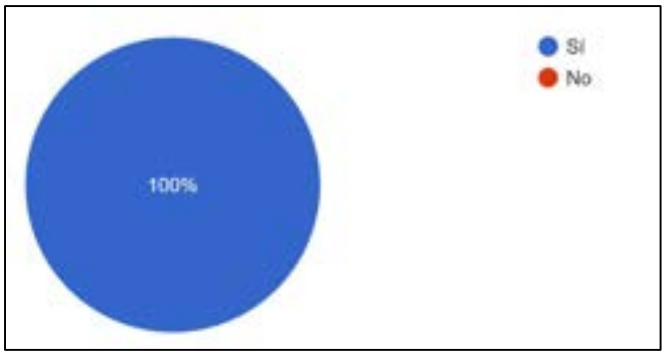

Figura 34. Encuesta MVP2, pregunta 5. Elaboración propia.

Pregunta 6.- ***¿Te ha parecido claro y funcional el diseño de los menús?***

La sexta pregunta se realizó para asegurarse de que los menús eran claros y sencillos de utilizar, ya que se habían realizado cambios a los mismo y se habían agregado menús para el control del inventario. Como se puede observar en la Figura 35, en este caso las respuestas fueron positivas, con un 92,3 % de los encuestados que han considerado el diseño de los menús claro y funcional. Solo un 7,7 % ha indicado que tuvo una mala experiencia. Estos datos demuestran que la presentación es correcta, pero tiene aspectos que se pueden mejorar.

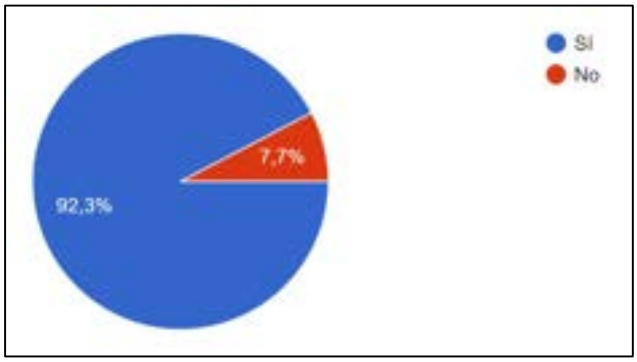

Figura 35. Encuesta MVP2, pregunta 6. Elaboración propia.

Pregunta 7.- ***¿Has tenido algún problema al interactuar con el inventario?***

Debido a la inclusión del inventario en el juego, decidimos preguntar más específicamente acerca de su interfaz y uso, con el objetivo de comprobar si el diseño y funcionamiento elegido era del agrado de los usuarios. Sin embargo, como podemos observar en la Figura 36, el 69,2 % de los encuestados no ha tenido problemas para interactuar con el inventario, con un 30,8 % teniendo dificultades, indicando que este sistema es funcional, pero debería ser revisado para pulir los aspectos que pueden haber resultado molestos.

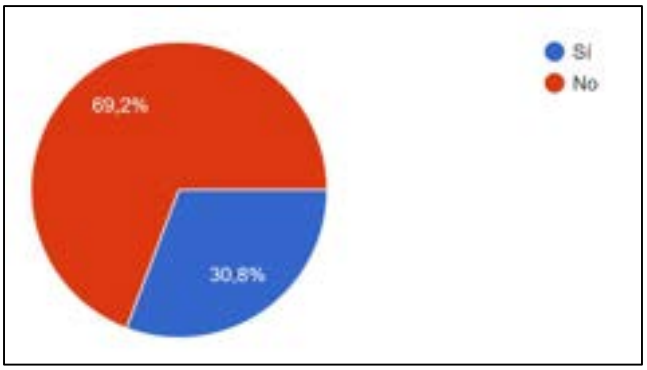

Figura 36. Encuesta MVP2, pregunta 7. Elaboración propia.

Pregunta 8.- ***¿Has encontrado el texto fácil de leer?***

Como pregunta acerca de la accesibilidad a la hora de leer los diálogos del juego, preguntamos acerca de su facilidad de lectura, ya que optamos por un color para la caja de diálogos y la letra de un tono similar y con un bordeado negro. Donde, según las respuestas de la encuesta (Figura 37), un 7,7 % de los encuestados tuvo problemas ocasionales al momento de leer alguno de los textos, frente a los 92,3 % que no encontró problemas para ello.

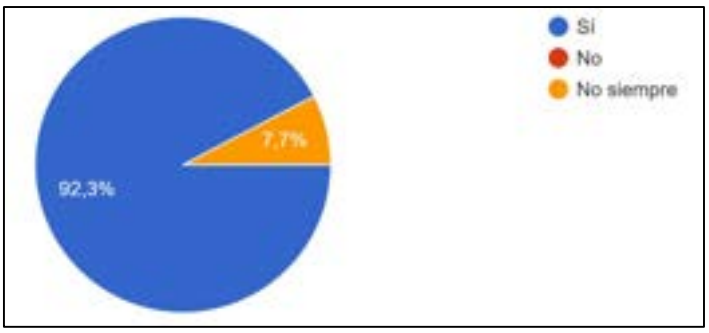

Figura 37. Encuesta MVP2, pregunta 8. Elaboración propia.

Pregunta 9.- ***¿Si se abriera una campaña de Kickstarter (micromecenazgo), estarías dispuesto a contribuir para terminar el desarrollo?***

La novena pregunta hacía referencia al futuro del proyecto y la disposición de los usuarios a ayudar en la finalización del proyecto, en concreto a su posible participación con la hipotética campaña de microfinanciación en Kickstarter. Como se puede observar en la Figura 38, un 69,2 % de los encuestados estrían dispuestos a participar en esta campaña de Kickstarter para contribuir al desarrollo, poniendo de manifiesto una vez más que la idea tiene potencial, pero posiblemente su ejecución es mejorable.

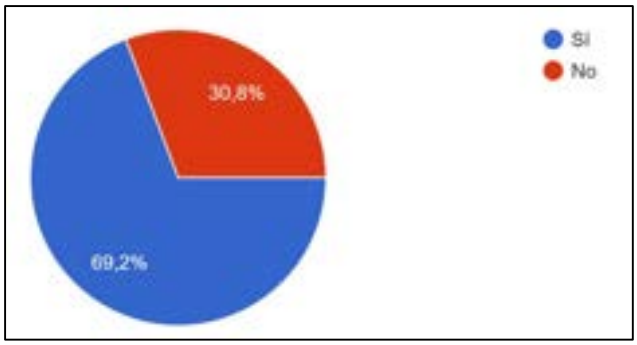

Figura 38. Encuesta MVP2, pregunta 9. Elaboración propia.

Pregunta 10.- ***En caso afirmativo, ¿cuánto estarías dispuesto a contribuir (en €)?***
Como continuación de la pregunta anterior, se preguntó sobre la cantidad de dinero con la que estarían dispuestos a contribuir en caso de abrirse la campaña, sin embargo, al momento de la elaboración de esta encuesta, se dejó la pregunta como obligatoria de contestar, obteniendo resultados como los de entre de 0,01 a 2 €. Pese a esto, debido al reducido número de encuestados, los resultados (Figura 39) son bastante prometedores, ya que de los 13 encuestados, 8 estarían dispuestos a contribuir con 5 € o más, por lo que, a una escala considerablemente mayor, se podría obtener beneficio.

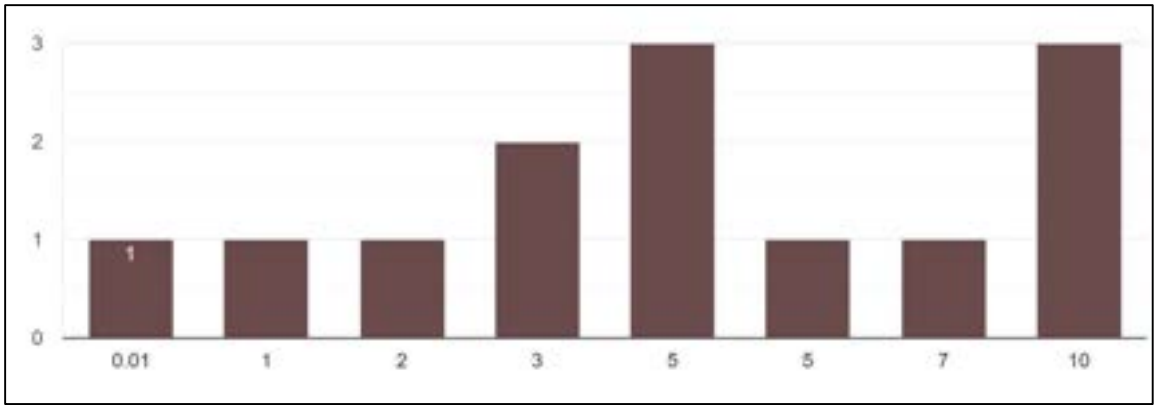

Figura 39. Encuesta MVP2, pregunta 10. Elaboración propia.

Sugerencia 1.- ***¿Cuáles han sido los aspectos que más te han gustado?***
La primera sugerencia estaba relacionada con los puntos positivos del proyecto, preguntando qué aspectos habían gustado más. Cabe destacar que las respuestas recibidas en esta ocasión para las sugerencias han sido más completas y concretas que en la ocasión anterior, hecho que ha mejorado la retroalimentación con el equipo. En la mayoría de las respuestas se hace mención a la estética y diseño del entorno, resaltando su originalidad («El diseño de los personajes y el mundo tienen mucha personalidad.») y detalles («Las animaciones de los helados y el movimiento» o «La adaptación de elementos del mundo real al mundo de los polos»). También se hace hincapié en su aspecto narrativo, destacando las diferentes conversaciones con los personajes («Los diálogos eran graciosos, y la historia interesante»).

Sugerencia 2.- ***¿Cuáles han sido los aspectos que no te han gustado o podrían mejorarse?***
Dado el mayor alcance conseguido con este experimento, era de esperar nuevos puntos de vista a la hora de probar el producto, encontrando fallos de diseño y de seguimiento de objetivos, como se pudo ver en la pregunta 1. Así pues, los aspectos que menos gustaron fueron los de tener que repetir todo el nivel tras morir, debido a la ausencia de puntos de control («Me gustaría haber tenido algún tipo de *checkpoint* o guardado de partida, aunque fuera manual»); el comportamiento de la cámara, que en ocasiones no era el correcto («La cámara es liosa») y debido a no poder controlarla generaba situaciones en las que no se podía jugar al juego («La cámara no ayudaba: esto era especialmente problemático en algunos saltos, ya que era difícil saber dónde ibas a caer»); y la interacción con el inventario, cuyos controles no quedaron claros a algunos usuarios («La interfaz de los menús es clara excepto el menú de objetos. A lo mejor indicar las teclas de cómo utilizarlo»). Con estas sugerencias, está claro que nos falta mucho por mejorar en general y mucho que pulir en cuanto al diseño y la implementación de mecánicas y sistema se refiere.

Sugerencia 3.- ***Sugerencias y otras consideraciones.***
La tercera sugerencia serviría para incluir conclusiones que no habían tenido cabida en los otros apartados o mejoras que podían haber considerado. En este caso nos indicaron posibles mejoras para conocer el objetivo o la misión actual en todo momento, ya que muchos de ellos habían usado la guía para avanzar en el juego («Estaría bien que de alguna forma se indicará lo que debes de hacer» o «No se sabe que hay que hacer […] sin la guía»). En cuanto a la jugabilidad se proponen más indicadores para conocer que partes del mapa son accesibles («delimitaría las zonas accesibles en el mapa»). También hay mejoras para la cámara, se sugirió el uso de una «[cámara] estática un poco desde arriba» para aliviar los problemas de visibilidad. Finalmente, como consideración para los menús se recomienda ofrecer soporte para «navegar [los menús] usando el teclado».

4.4 Mercadotecnia y difusión
Dado que este trabajo se realizó dentro de un marco universitario, existieron actividades de control por parte del profesorado de las asignaturas para controlar y evaluar el desarrollo de cada grupo de trabajo. Estas evaluaciones se realizaban en un estilo de presentación, donde cada grupo de trabajo debía defender delante del resto de alumnos las adiciones que se habían realizado a su proyecto. De este modo, se podía aprovechar para mostrar el juego y sus atractivos al resto de compañeros, los cuales podían difundir el proyecto dentro de la universidad en preparación para futuros eventos estudiantiles.

Como tarea final para la asignatura, se nos presentó la oportunidad de mostrar el proyecto realizado en la «Feria de proyectos de estudiantes» de la escuela (Figura 40), realizada en diciembre, en la que pudimos tener un contacto más cercano con gente ajena al proyecto y

mostrarles las características principales de nuestro proyecto, qué idea quería representar y en qué destacaba. Durante esta puesta en escena, la gente interesada pudo probar el juego y realizar al final una pequeña encuesta que nos sirvió como retroalimentación al equipo.

Figura 40. Equipo inicial de desarrollo de Frozen Out durante la feria. De izquierda a derecha: Vicent Pla Madrid, Adrián Reina Sáez, Alejandro Jiménez Carrasco, Alejandro Gómez Noe, Tomás Ruiz Martín y Pablo López Orrios

En preparación para la salida de la segunda demo al público (relacionada con el desarrollo del segundo MVP) se abrió una cuenta de Instagram para compartir imágenes y novedades sobre el juego. Se eligió esta red social debido a la importancia que tienen las imágenes en sus publicaciones, una característica ideal para la difusión del apartado artístico, uno de los atractivos del juego.

La difusión de la segunda demo y la prueba por parte de sus usuarios estuvo mermada por la evolución de la COVID-19, de modo que se centraron los esfuerzos proveer de mecanismos para permitir una difusión totalmente digital. Es por ello por lo que se preparó la página de GitHub del proyecto para albergar un espacio donde informar los usuarios de los enlaces para descargar la demo y para responder la encuesta, así como información adicional sobre los controles y una pequeña guía para poder terminar la demo en caso de confusión.

PARTE II: Desarrollo de elementos jugables

5. Aspectos técnicos

En este punto se enumerarán las diferentes herramientas empleadas tanto para el desarrollo, como para la organización del equipo, acompañados de una pequeña explicación acerca de su elección para el proyecto

Git

Software de control de versiones que nos permite mantener un registro y coordinar el trabajo de diferentes personas mediante la fusión de archivos, lo cual nos permite tenerlos actualizados si nos es necesario al estar trabajando en la misma rama o tener una versión parcialmente diferente donde desarrollar nuevas funcionalidades sin comprometer el resto del proyecto.

GitHub

Plataforma que nos permite usar Git de forma sencilla y visual, proporcionándonos gráficas para ver la información como el flujo de trabajo o el historial de cambios realizado. Además, nos permite ver directamente el estado del proyecto en versiones antiguas, pudiendo recuperar los datos de ser necesario. En nuestro caso (Figura 41), hemos realizado un equipo, del cual somos miembros, de esta forma el proyecto no está alojado en el perfil de uno de nosotros, sino en un lugar común, evitando así que los pagos o penalizaciones por espacio o ancho de banda perjudiquen al perfil de uno de nosotros directamente.

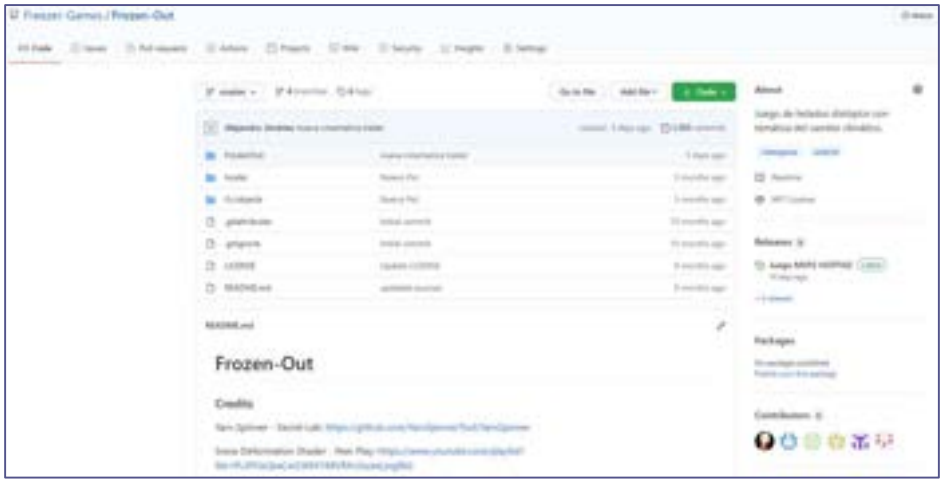

Figura 41. Proyecto en GitHub. Elaboración propia.

Trello

Como herramienta empleada para la planificación y organización del quipo se usó Trello, la cual, mediante el uso de listas a rellenar con diferentes tareas, ya sean de planificación, desarrollo, revisión o corrección, permite tener un control sobre que debería estar haciendo cada integrante del equipo según la tarea que se le asigne. Pudiendo además clasificarlas por prioridad dentro de cada lista, cada tarea dispondrá de un plazo en el que se deberá cumplir para el correcto avance del proyecto. En la Figura 61 se puede ver como quedaron las tareas tras acabar el segundo MVP.

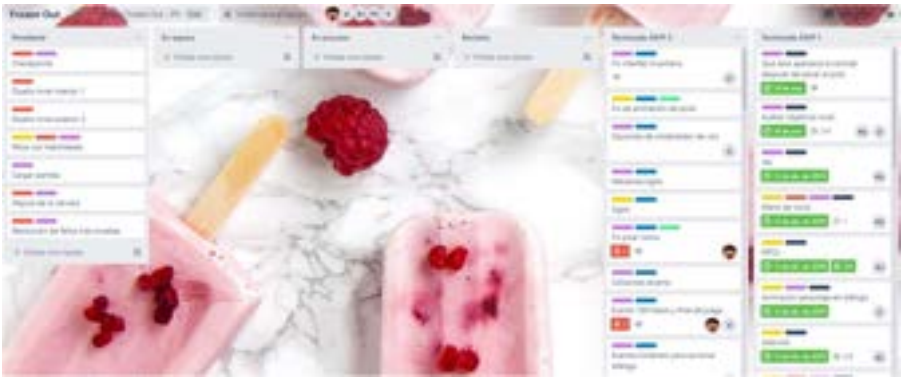

Figura 42.Página de Trello del proyecto al final del segundo MVP. Elaboración propia.

Unity

Como motor gráfico para el desarrollo del videojuego empleamos Unity, aunque también pudimos emplear otros como Unreal Engine o Godot, estos fueron descartados principalmente por la curva de dificultad en el uso de Unreal y el desconocimiento del uso del motor Godot. En cambio, en *Unity*, al haber sido utilizado por el equipo tanto en proyectos de clase, como en otros proyectos ya se poseía conocimiento de este. Además de que era un requisito hacer uso de este dado que el proyecto fue iniciado en la asignatura de Introducción a la Programación de Videojuegos, donde nos enseñaban su funcionamiento. En la Figura 43 se puede ver como tenía organizada la interfaz del programa para trabajar en el proyecto.

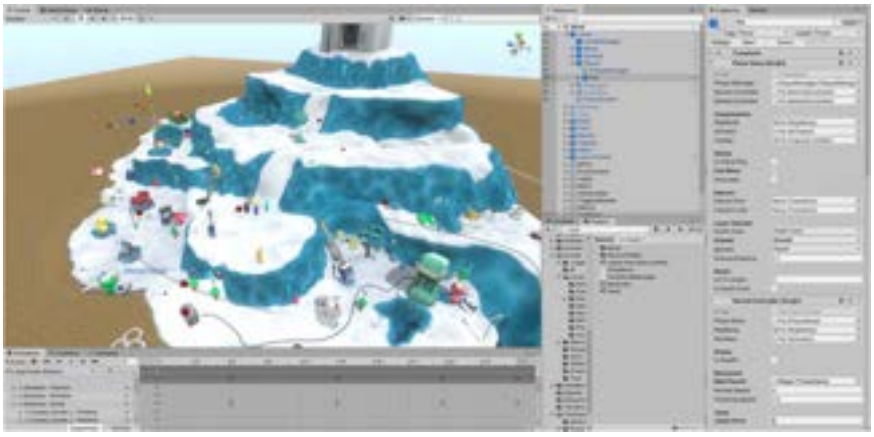

Figura 43. Proyecto abierto en Unity. Elaboración propia.

Visual Studio

Como entorno de desarrollo integrado[53] (IDE) se ha empleado principalmente Visual Studio (Figura 44), utilizado en otros proyectos y con conocimiento acerca de cómo emplear algunas de sus herramientas. Además, dada su potencia y versatilidad, junto la posibilidad de vincular puntos de parada al código, entre otras cosas, junto a la ejecución de las escenas en Unity para depurar y comprobar fallos en el código eran aspectos interesantes que nos ahorrarían mucho tiempo.

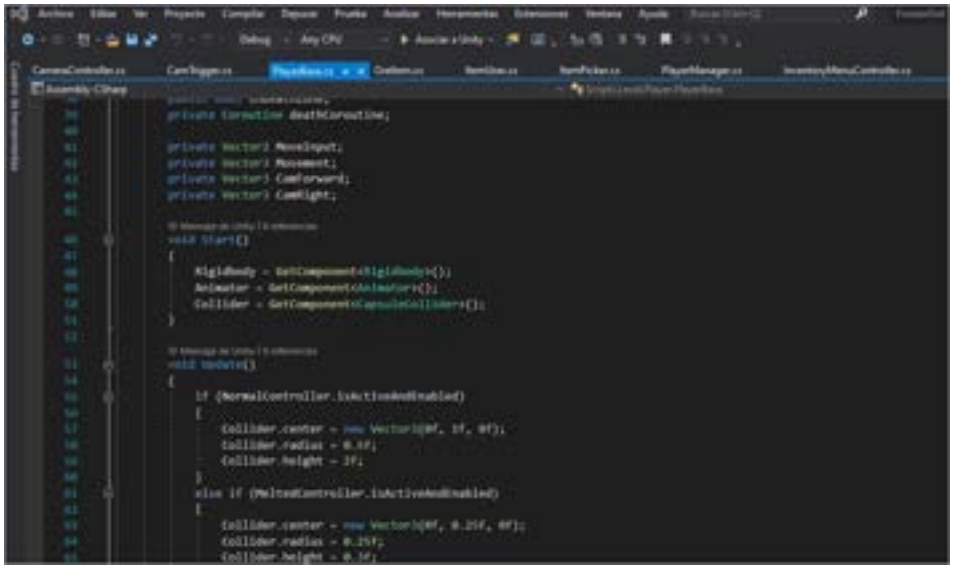

Figura 44. Script de uno de los componentes del proyecto abierto en Visual Studio. Elaboración propia.

[53] Programa que facilita el desarrollo de código con diferentes servicios integrados en él.

6. Desarrollo de mecánicas

Las mecánicas o habilidades dentro de un juego sirven como medio para que el jugador pueda desplazarse, interactuar y superar desafíos que se le presenten dentro del entorno virtual en el que se encuentra. Pudiendo ser más o menos sencillas, están deben cumplir con su cometido para que el jugador pueda usarlas cuando lo necesite y no se olvide de ellas y las posibilidades que le ofrecen. En *Frozen Out*, el jugador podrá moverse y saltar libremente, además a medida que avance el usuario, encontrará distintas herramientas, las cuales le permitirán interactuar con diferentes objetos según se requiera. En la Figura 45 se puede ver las mecánicas pensadas en la fase de ideación del juego.

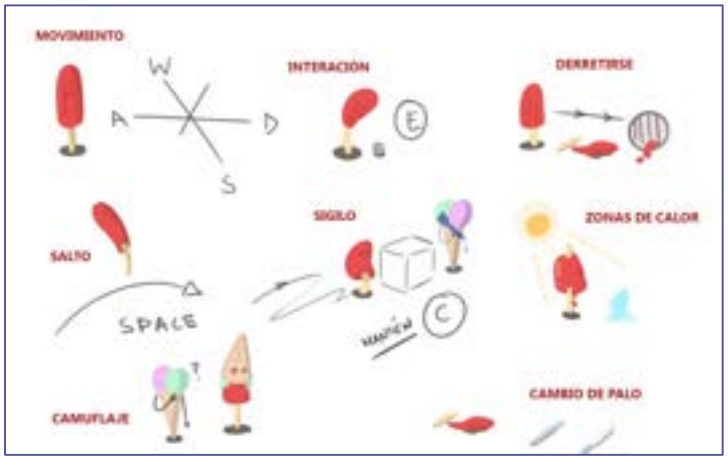

Figura 45. Concepto inicial de las mecánicas del juego. Elaboración propia.

6.1 Estructura del personaje

Unity presenta una estructura basa en *GameObjets* [54]con la que se construyen las escenas, que contienen a estos, constituyendo un entorno jugable en mayor o menor medida. Así mismo, los *GameObject* están constituidos por diferentes piezas, llamadas componentes, que dotan a los objetos del entorno un comportamiento o propiedad según las preferencias o el objetivo que el desarrollador tenga pensado para este. Pese a poder configurar el comportamiento de los objetos y decidir qué componentes van a formar parte de él, estos siempre serán creados con un componente que determinará su posición, rotación y escala, llamado *Transform*, que indicará estos valores de forma global, salvo en los casos en los que un objeto este contenido en otro objeto, haciendo que este componente vaya en función de su objeto padre.

Esto es importante debido a que, el avatar del jugador está constituido principalmente por un modelo 3D con su propia escala para adaptarse al tamaño del entorno y a otros objetos ligados a las funcionalidades desarrolladas. Por lo que, de elegir el modelo como el objeto padre, los *Transforms* de los objetos hijos darían problemas, en cuanto al tamaño o posición se refiere.

[54] Todo objeto situado en la ventana del editor de Unity y que compone una escena

Como solución, los objetos se encapsulan en su totalidad en un objeto vacío con sus valores de posición, escala y rotación por defecto, donde además se incluirán el resto de los compontes y sobre el cual se realizarán casi la totalidad de los cálculos necesarios.

Figura 46. Jerarquía de objetos del avatar y muestra en la escena. Elaboración propia.

Como se puede ver en la Figura 46, la jerarquía de objetos que componen al personaje jugable, Pol, está constituida por el esqueleto realizado por Adrián Reina Sáez; empleado por el animador Alejandro Jiménez Carrasco para crear las animaciones y dotarle de vida; un punto que servirá como objetivo para la cámara, denominado *CamPoint*; la geometría que constituye la parte visible de Pol, principalmente su cuerpo y palo que caracterizan a un polo, pero también apareciendo como desactivados en la imagen con un tono más claro, un objeto que sirve como estado intermedio, llamado *DerretirsePol,* que pese a no estar afectado por el esqueleto también forma parte de la geometría, su función se explicará más adelante; un objeto llamado *SorbetePol*, que sirve como cuerpo visible del personaje al activar la habilidad de derretirse; las herramientas básicas a utilizar, el pico y la cuchara; y un palo extra, que no pertenece a la geometría, pero que sirve como apoyo para la habilidad de derretirse, teniendo su propio comportamiento, además de incluir un objeto denominado *ItemPos*; por último, se encuentra un objeto vacío que contiene las partículas con las que se aplicarán efectos a ciertas acciones de Pol.

Por otro lado, tenemos los componentes que hacen posibles los distintos comportamientos desarrollados para Pol. A diferenciar entre los que podemos ver en la Figura 47, empezando por su *Transform*, explicado con anterioridad; la clase *PlayerBase* que recoge los *inputs* por parte del usuario y las operaciones que se le aplican a estos que son comunes en cualquiera de los estados posibles del avatar; las clases *NormalController* y *MeltedController*, los cuales están siempre activos de forma alterna, refiriéndose a las dos formas que puede adoptar el jugador, normal y derretido, utilizando los *inputs* proporcionados por el *script* mencionado al principio

en su provecho para transformarlo en el movimiento y acciones que caracteriza a cada forma; el cuarto *script* se refiere al componente encargado de reproducir los sonidos de Pol al realizar ciertas acciones; a continuación se encuentra el *Animator*, componente encargado de controlar la ejecución de las animaciones, situado en el objeto que encapsula al avatar para poder, no solo acceder al modelo y animarlo sino también, hacerlo lo propio con distintas variables de otros componentes; seguido de este se encuentra el *Rigidbody* y un *Capsule Collider*, el primero se encarga de aplicar físicas al objeto, cuya zona de colisión está delimitada por el segundo, quien además se encarga de detectar que está entrado en su campo y que está saliendo; por último nos encontramos con el *AudioSource*, cuya función es la de emitir los sonidos seleccionados para ello.

Figura 47 Componentes del avatar. Elaboración propia.

En cuanto al *Animator*, dada su importancia a la hora de la sincronización de distintos sonidos de Pol y de habilidades como la del cambio de forma, que como se ha mencionado en el párrafo anterior, se encarga del control de las animaciones de un objeto. Estas animaciones constituyen un estado sobre lo que se irá cambiando según diferentes parámetros (Figura 48) modificables desde el código, obteniendo de la unión de diferentes estados un grafo (Figura 49) en el que, partiendo del único estado vinculado a *Entry*, se podrá alcanzar cualquiera de los demás estados siguiendo las precondiciones y condiciones (Figura 50) necesarias para alcanzarlo, exceptuando los que únicamente poseen arcos de entrada del estado *Any State,* que pueden ser alcanzados en cualquier momento, siendo utilizados principalmente para animaciones vinculadas a acciones que se pueden realizar en casi cualquier momento, como el sigilo o el salto.

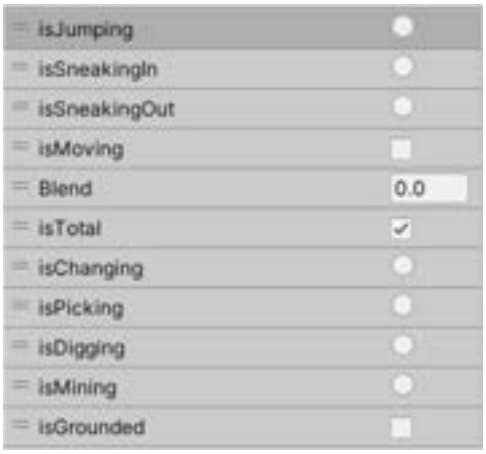

Figura 48. Parámetros del Animator de Pol. Elaboración propia.

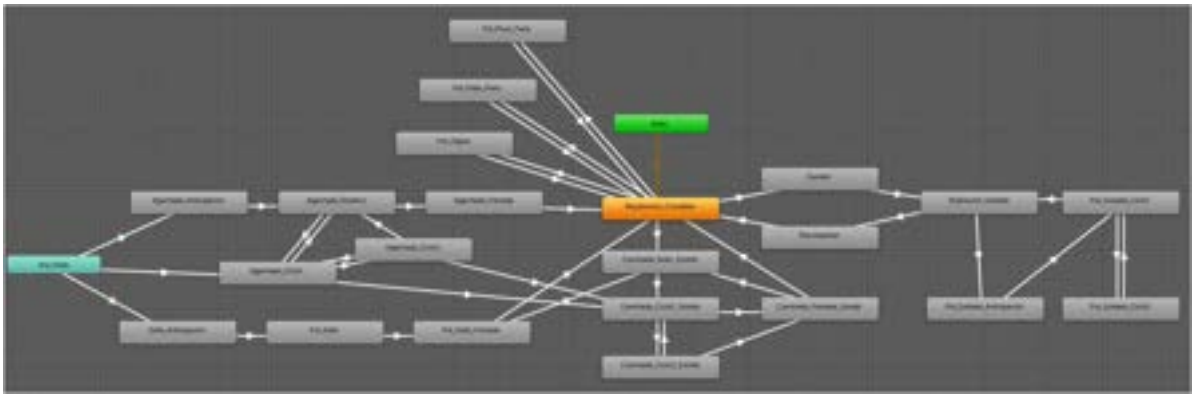

Figura 49. Grafo del Animator de Pol. Elaboración propia.

Figura 50. Condición para pasar de la animación de movimiento a la de frenada. Elaboración propia.

6.2 Movimiento

Al tratarse de un videojuego en 3D, donde una de sus principales características es la de posibilitar el desplazamiento del jugador en los tres ejes de coordenadas, se decidió añadir esta característica al juego dada la naturaleza de este y el entorno jugable desarrollado, una ciudad con distintos niveles de altura y de una extensión considerable.

Para su desarrollo se nos presentaron dos posibilidades con dos componentes diferentes, con sus pros y contras, teniendo que elegir entre el *Character Controller* (Figura 51) y el *Rigidbody* (Figura 52). El primero, el cual incluye su propia caja de colisiones incorporada, únicamente hace uso de la gravedad dentro del motor de físicas, siendo la única fuerza que se ejerce sobre el objeto, teniendo que programar como y cuando le afectaría la gravedad al objeto, además ofrece de base funciones tanto para mover al objeto que lo contiene en cualquiera de los ejes, como

para detectar el suelo, además de una forma propia de detectar las colisiones con otros objetos del entorno. El segundo, haciendo uso del motor de físicas, hace que el jugador se mueva en base a fuerzas a aplicar en distintas direcciones, teniendo como resultado el movimiento según los cálculos de la simulación del motor dependiendo de los valores y restricciones de fricción o rotación seleccionados. No nos facilitaba ninguna función a usar más allá de las que nos permitían aplicar fuerzas y obtener velocidades resultantes; al no tener una caja de colisiones integrada, se podía elegir la forma de esta además de poder incluir más de una, lo que detectaría a cualquier objeto saliente o entrante de esta en todo momento.

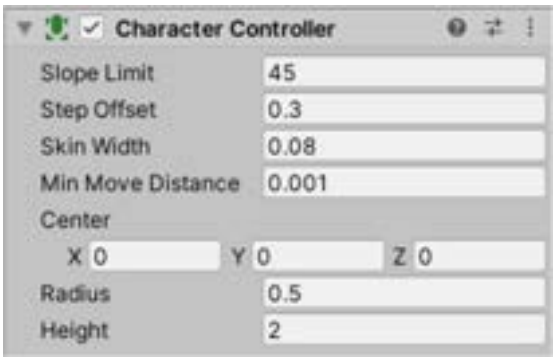

Figura 51. Character Controller en el inspector de Unity. Elaboración propia.

Figura 52 Rigidbody en el inspector de Unity. Elaboración propia.

Finalmente se eligió emplear el segundo componente, que sin ofrecer una forma directa de desplazar al jugador en la dirección deseada, teniendo que aplicar fuerzas en la dirección elegida, tenía en todo momento en cuenta el resto de colisiones del entorno, a diferencia del *Character Controller* que solo las detectaba al llamar a la función que realizaba el movimiento, lo cual era un problema para la introducción de plataformas móviles dentro del juego, detectar que el jugador ha entrado en una zona no permitida de forma involuntaria o la posibilidad de

interactuar con objetos. Siendo este el escenario actual, el movimiento del personaje se desarrolló para ser posible en las ocho direcciones del plano XZ, haciendo además que fuera relativo a la dirección de los mismos ejes a los que estaba mirando la cámara, ignorando su vector Y.

Para la implementación, dadas las dos formas de las que consta el jugador, el procesamiento de los *inputs* se dividió entre los tres componentes antes mencionados: *PlayerBase*, *NormalController* y *MeltedController*. El primero se encarga de capturar los *inputs* del usuario y normalizar el vector resultante en cada fotograma, además de capturar la dirección a la que la cámara está mirando. En el caso del controlador para la forma normal del avatar, se hace uso de los datos proporcionados por la cámara y las pulsaciones de teclas del usuario para calcular en qué dirección debe ir el avatar, aplicándole una fuerza en la dirección del vector resultante ejecutando y ejecutando la animación necesaria (Figura 53). También existe una mecánica adicional relacionada al movimiento, el sigilo, en la que el jugador reducirá su velocidad de movimiento a conveniencia, aplicando también su animación correspondiente, para no ser visto rápidamente por las patrullas enemigas. Dado que su comportamiento va vinculado a como la IA reacciona a este suceso, este aspecto es explicado en el trabajo de Pablo López Orrios.

Figura 53. Pol en mitad de la animación de andar. Elaboración propia.

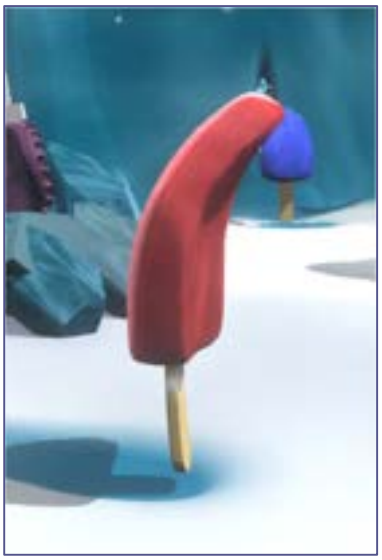

Figura 54. Pol en mitad de la animación de sigilo. Elaboración propia.

En esta forma también se hace uso de los eventos propios de las animaciones de Unity, a los que se les puede vincular cualquier función pública de los componentes que estén dentro del mismo objeto y nivel en el que se encuentra un componente *Animator*. Para este caso, el evento está vinculado a la función *SelectClipToPlay* de la clase *PlayerSound*, la cual hereda de la clase *SoundController*, que posee funciones tales como reproducir y parar un clip de audio, así como seleccionar uno de forma aleatoria desde una lista, evitando así que reproduzca el mismo constantemente.

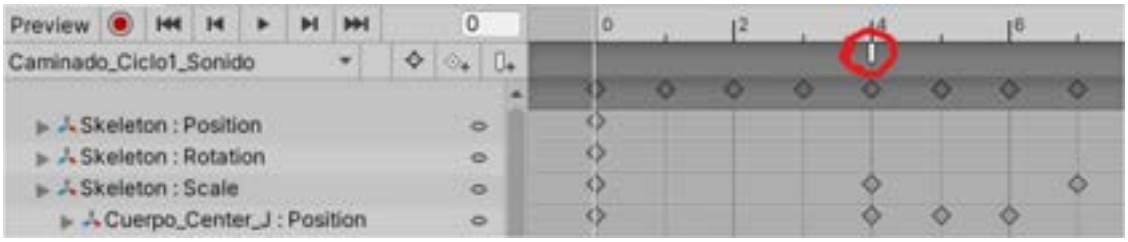

Figura 55. Animación de andar con evento de animación redondeado en rojo. Elaboración propia.

Figura 56. Fragmentos de código de PlayerSound y SoundController. Elaboración propia.

Por último, el tratamiento de los datos en el controlador de la forma sorbete o derretida requiere del componente *Animator* para sincronizar la velocidad que debe llevar el jugador con respecto a la animación (Figura 57) del ciclo de andado de la forma que se está reproduciendo en ese momento. Por lo que tal y como se puede ver en la Figura 58 que hace referencia a la animación de carga de la masa, el parámetro enmarcado en rojo se encuentra activado, reduciendo la velocidad aplicada al vector de movimiento resultante a una tercera parte del total. Por otro lado, en la Figura 59, el parámetro de carga está desactivado, lo que hará que al vector de movimiento resultante se le aplique el total de la velocidad. De esta forma, añadiendo una condición en la función de movimiento del controlador de la forma, se podrá cambiar sin complicaciones la velocidad, como se realiza en la Figura 60.

Figura 57. Pol andando por el mapa en su forma de sorbete. Elaboración propia.

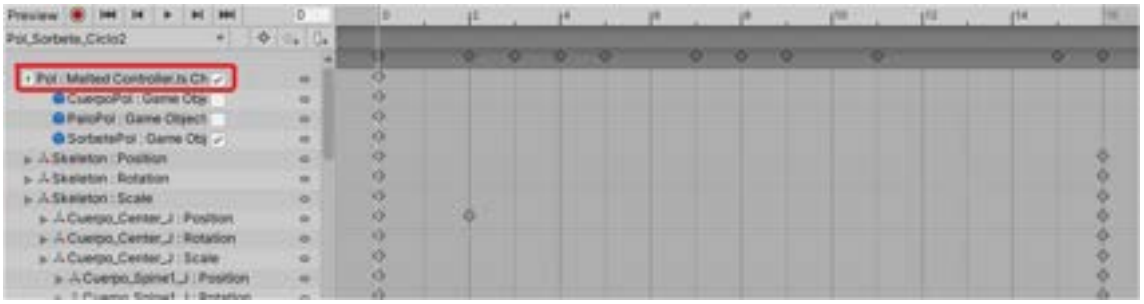

Figura 58. Ciclo de la animación del movimiento de la forma sorbete con el parámetro de carga activado. Elaboración propia.

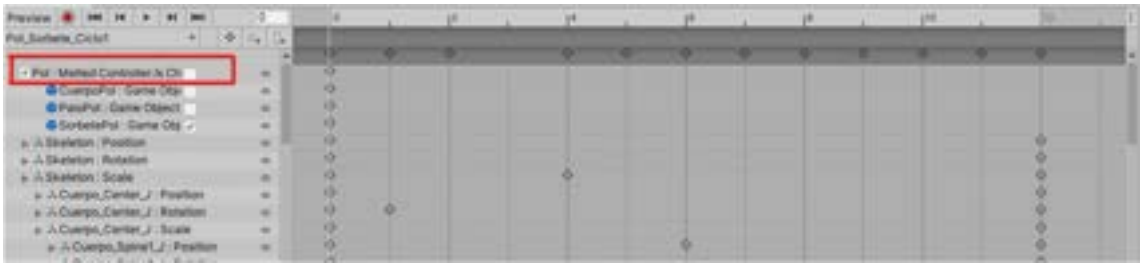

Figura 59. Ciclo de la animación del movimiento de la forma sorbete con el parámetro de carga desactivado. Elaboración propia.

```
protected void CalculeMove()
{
    MovementDir = PlayerBase.GetMovement();

    if (IsCharging)
    {
        MovementDir *= (MoveSpeed / 3f);
        Rigidbody.velocity = new Vector3(
            MovementDir.x,
            Rigidbody.velocity.y,
            MovementDir.z);
    }
    else
    {
        MovementDir *= MoveSpeed;
        Rigidbody.velocity = new Vector3(
            MovementDir.x,
            Rigidbody.velocity.y,
            MovementDir.z);
    }
}
```

Figura 60. Función que sincroniza la velocidad del jugador según el parámetro booleano IsCharging modificado mediante la animación de movimiento. Elaboración propia.

6.3 Salto

De la misma forma que el movimiento nos permite movernos a través del plano formado por los ejes X y Z, el salto nos permite sortear obstáculos o acceder a nuevas zonas que estén a una altura superior o separadas por un agujero a saltar, a través del desplazamiento por el eje Y, pudiendo de esta forma elevar al jugador momentáneamente mediante el sistema de físicas de Unity.

La posibilidad de ejecutar el salto viene ligada a la restricción de que el jugador tiene que estar en contacto con el suelo. Pese a que esta comprobación solo sería necesaria en la forma normal dado que es la única que posee esta capacidad, el método empleado para ello sirve para otras funciones a explicar más adelante, por lo que las comprobaciones realizadas están contenidas dentro de la función *CheckWithRay* de *PlayerBase*. Como la comprobación necesaria es la de asegurarnos de que el jugador está en contacto con el suelo del nivel, una implementación básica sería la de ver si la caja de colisiones de este está en contacto con la del suelo, pero dado

el seccionamiento del mapa para la implementación del *shader* encargado de marcar un rastro por donde pasa el jugador era una solución inviable, ya que daba falsos negativos, teniendo la posibilidad de no poder saltar en ciertas zonas que lo requerían. Así pues, para hacer esta comprobación, se hizo uso de un *Raycast*[55], el cual detectaría lo que habría bajo los pies del jugador, pudiendo distinguir entre si algo es o no suelo u otro tipo de superficie, clasificado por un *layer*[56], debiendo hacer algo adicional al posarse sobre esa superficie o no.

La implementación constaría de la función *CheckWithRay* (Figura 61) antes mencionada, que se ejecutaría en cada fotograma comprobando si está situado en la superficie correcta, creando un rayo en la dirección negativa del eje Y de la posición del jugador en cada momento. Como nace desde el centro del objeto, para evitar posibles colisiones con objetos internos del mismo que componen al jugador, se añadió un parámetro adicional a la hora de crear el *Raycast*, de tal forma que todo lo que esté contenido dentro del *layer* designado como «Player», será ignorado. Tras la creación del rayo, se comprueba que este haya impactado con algún otro objeto, tras esto se comprobará a que conjunto de *layer* pertenece, siendo el de «Ground» el elegido para delimitar el suelo y el resto para delimitar otras zonas o aspectos del juego que no influyen en la jugabilidad. Así pues, si el jugador está bien situado, al presionar la tecla vinculada para el salto se le aplicará momentáneamente una velocidad en el eje Y de coordenadas, momento en el cual se reproducirá la animación correspondiente (Figura 62).

```
private void CheckWithRay()
{
    Debug.DrawRay(transform.position, -transform.up * groundDistance, Color.red, 1f);
    RaycastHit hit;

    if (Physics.Raycast(transform.position, -Vector3.up, out hit, groundDistance, ~Ignored))
    {
        if (Ground == (Ground | (1 << hit.transform.gameObject.layer)))
        {
            Grounded = true;
            Animator.SetBool("isGrounded", true);

            if (InDeathZone)
            {
                StopCoroutine(deathCoroutine);
                InDeathZone = false;
                PlayerManager.OnNormalZone();
            }
        }

        if (DeathZone == (DeathZone | (1 << hit.transform.gameObject.layer)))
        {
            if (!InDeathZone)
            {
                deathCoroutine = StartCoroutine(CountdownToDeath());
                PlayerManager.OnDeathZone();
            }
        }
    }
}
```

Figura 61. Implementación de la función CheckWithRay Elaboración propia.

[55] Rayo que se origina desde un punto en una dirección y que detecta las colisiones de los objetos que este atraviesa

[56] Forma de crear grupos de objetos a los que se referirse como una capa y que comparten alguna característica en particular

Figura 62. Pol saltando sobre un obstáculo. Elaboración propia.

6.4 Cambio de forma

La habilidad de cambio de forma se pensó para permitir al jugador pasar a través de zonas en las que dada su altura fuera imposible hacerlo o en las que, dado el estado sólido de este no pudiera pasar a menos que se convirtiera en algo entre medias, como un sorbete, pudiendo pasar a través de rejas u obstáculos similares que obligasen al jugador a separar su cuerpo mínimamente. La ejecución de esta da como resultado el cambio de forma del avatar del jugador, de un helado a una masa entre sólida y líquida dejando el palo atrás y viceversa, manteniendo este y la forma normal. Para volver a formarse se deberá o bien encontrar un nuevo palo o volver al anterior. Pese a que se ha desarrollado esta funcionalidad, esta no ha sido introducida como una mecánica a usar en el nivel desarrollado debido a decisiones de diseño para el nivel, ya que la habilidad se conseguiría en un momento más avanzado.

Así pues, para que el jugador pudiera hacer uso de esta habilidad, tendría que permanecer inmóvil y tocando el suelo para poder usarla. En caso de pasar del estado sólido al sorbete, dejaría el palo atrás para ser recogido, en adición, en ese momento este dejaría de formar parte del objeto que agrupa a los que compone al jugador, convirtiéndose en un objeto externo más con el que se podría interactuar, al que la afectarían las físicas de forma independiente del avatar. En el caso contrario, al situarse el jugador cerca del palo deseado a partir del cual se quiera reconstruir y presionar la tecla de interacción, este se movería automáticamente a la posición de *ItemPos* del objeto *Stick*. Una vez ahí, se ejecutará la animación necesaria para realizar la transición y volverá al estado normal, volviendo el palo a ser parte del conjunto de objetos que componen al personaje.

Para su implementación, se hizo uso de los eventos propios que ofrece Unity, los cuales se incluyen dentro de los *scripts*. Para hacer uso de ellos, es necesario declararlos objetos de tipo *UnityEvent* e inicializarlos en la función *Start* del *script*, pudiendo ser invocados mediante su función *Invoke*. El primero de ellos se situó en el controlador de la forma normal del jugador, teniendo una referencia al *script* controlador del palo en posesión del jugador. Debido a que el que posee actualmente es diferente al visible en la geometría, en el momento en el que la habilidad se accione, por una parte, el controlador normal se encargará de desactivar el palo de la geometría además de dar la señal para reproducir la animación que cambie la forma, mientras que el evento se encargará de llamar a la función del *script StickItem*, *Melting*, a la que hace referencia, activándolo y dejando de formar parte del conjunto. En este momento, como ya se ha mencionado, el palo pasará a ser parte de los objetos del entorno con los que se puede interactuar.

Figura 63. Ejemplo de UnityEvent en el inspector

El segundo evento se situó en el controlador de los palos, teniendo una referencia al script del controlador de la forma derretida del jugador. Para que el jugador pueda reconstruirse necesitará primero estar en contacto con el palo desde el que va a volver a formarse, de forma que *PlayerBase* pueda detectar el punto al que el jugador tiene que mirar y moverse de forma automática al interactuar, una vez detectado, en este caso si el usuario activa la habilidad de cambio de forma, este entrará en el estado de interacción, durante el cual el jugador perderá el control momentáneamente hasta que a través de la función *MoveToTarget* (Figura 64) llegue a su destino según el resultado obtenido al haberse acercado al palo, en este caso. Mientras el estado interacción esté activado, el cambio de forma se detiene, reanudándose cuando se detenga la ejecución de *MoveToTarget*, momento en el cual se activará la animación y los cambios consecuentes a las colisiones y objetos activados necesarios.

```
public void MoveToTarget(Transform targetPos,Transform targetLook, float distanceToStop, float speed)
{
    if (targetPos == null && targetLook == null)
    {
        PlayerManager.SetIsInteracting(false);
        PlayerManager.SetInteractiveItem(null, null);
    }
    else if (targetPos == null && targetLook != null)
    {
        Vector3 lookPos = targetLook.position;
        lookPos.y = transform.position.y;
        transform.LookAt(lookPos);

        PlayerManager.SetIsInteracting(false);
        PlayerManager.SetInteractiveItem(null, null);
    }
    else
    {
        Debug.Log("interaccion con dos puntos");
        Vector3 lookPos = targetLook.position;
        lookPos.y = transform.position.y;
        transform.LookAt(lookPos);

        Vector3 goPos = targetPos.position;
        goPos.y = transform.position.y;

        if (Vector3.Distance(goPos, transform.position) > distanceToStop)
        {
            Animator.SetBool("isMoving", true);

            transform.position =
                Vector3.MoveTowards(transform.position, goPos, speed);
        }
        else
        {
            Animator.SetBool("isMoving", false);
            PlayerManager.SetIsInteracting(false);
            PlayerManager.SetInteractiveItem(null, null);
        }
    }
}
```

Figura 64. Función MoveToTarget. Elaboración propia.

En cuanto a la ocultación de la activación y desactivación de partes del personaje que caracterizaban cada una de las dos formas, se hizo uso de un objeto intermedio que pudiera hacer transiciones entre los estados, un objeto deformable no animado por el esqueleto que permanecía desactivado siempre que el jugador mantuviera una de las formas pero que se activaba para realizar el cambio, ya que los dos estados de los que partía eran la forma normal y la forma sorbete, permitiendo cambiar valores de la deformación a través de animaciones para conseguir el resultado como se puede ver en la Figura 65.

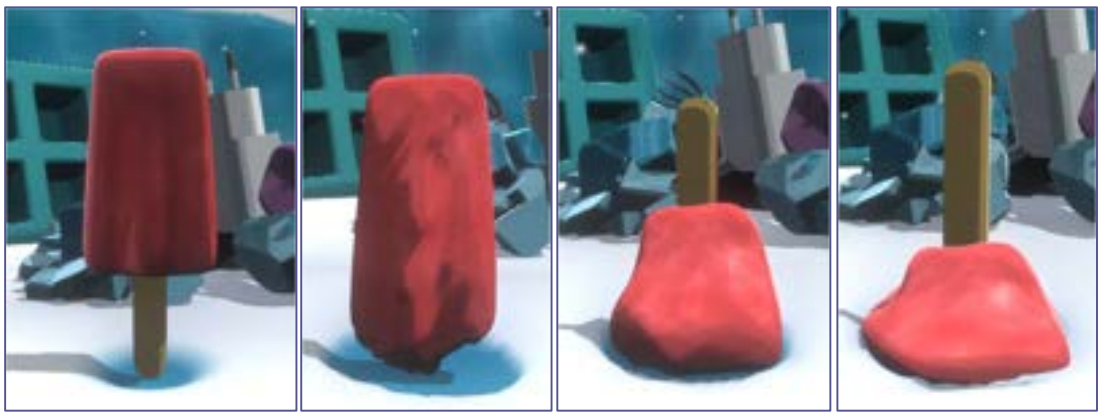

Figura 65. Proceso de cambio de forma de forma normal a sorbete. Elaboración propia.

Por último, con relación al palo extra situado en el jugador, tanto las físicas como las colisiones y la malla, se activan y desactivan a la par que lo hace el objeto antes mencionado, convirtiéndose en un objeto externo cuando, al estar completamente transformado en la forma sorbete, deja de estar en contacto con el cuerpo de Pol (), quedando tirado en el suelo para ser recogido posteriormente si así se desea. Además, he de destacar que, pese a tener un sistema de interacción e inventario, a explicar más adelante, en el cual se incluye la interacción con el palo, ya que su controlador deriva de ese sistema, las consecuencias de interactuar con este son gestionadas de forma diferente para una mejor gestión de su componente *Transform*.

Figura 66. Pol sorbete y su palo separados. Elaboración propia.

6.5 Derretirse (fin de la partida)

La muerte en un videojuego sirve principalmente para delimitar las zonas en las que puede moverse el jugador, devolviéndolo a un punto previo sin recibir mucho daño, o de forma más común, indicar que la partida ha terminado ya sea por culpa del enemigo o una zona peligrosa para el jugador. En nuestro juego, el fin de partida puede darse de tres maneras, la primera (Figura 67) sería caer en una zona en la que no puedes salir o peligrosa, acabando la partida al instante, como en la zona de la batidora; otra forma de que termine la partida sería mediante los enemigos hostiles, quienes al entrar en contacto contigo hacen que termine el juego, el proceso de detección de los enemigos está explicado en el trabajo de Pablo Lopez Orrios; la forma restante consiste en acceder a una zona fuera de los límites del mapa que, en nuestro caso, no están cubiertas de nieve, siendo indicador de que hace mucho más calor del que el jugador puede soportar.

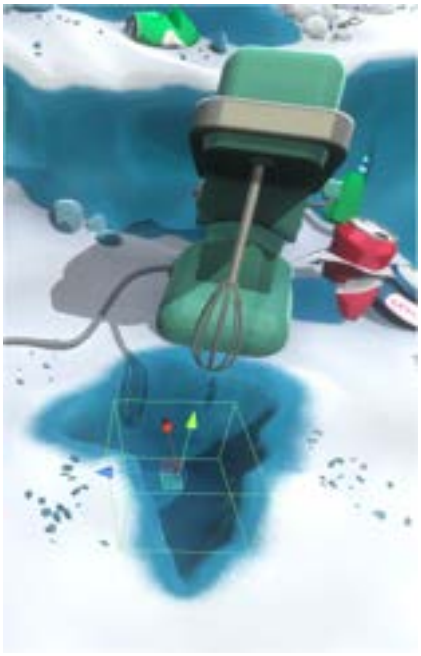

Figura 67. Batidora y caja de colisiones de muerte del agujero provocado por la excavación. Elaboración propia.

Para la implementación del primer método, al tener situada una caja de colisiones en la zona, el simple contacto con ella por parte del jugador ejecutaría la función *GameOver* del *LevelManager,* acabando así la partida y apareciendo el menú de muerte (Figura 68).

Figura 68. Menú de muerte. Elaboración propia.

Para lograr el tercer método, se hizo uso de la misma función que detecta si el jugador está tocando el suelo, *CheckWithRay*, con la diferencia de que en vez de buscar que el *layer* del suelo sea «Ground», sea «DeathZone». Una vez se ha detectado esto, significando que el jugador se encuentra en esa zona, la *corrutina*[57]*CountdownToDeath* se ejecutará una única vez, activando, por una parte, una cuenta atrás en la que si el jugador permanece en la zona durante el tiempo delimitado de tres segundos la partida acabará, llamando a la función *GameOver* de *LevelManager*. Para revertir esta situación y evitar perder, bastaría con entrar nuevamente a la

[57] Parte de la lógica que permite tratar un fragmento de código de forma que esté vinculado a una duración de tiempo en vez de a los fotogramas a los que se reproduce el juego.

zona designada como «Ground», lo que pararía las partículas, devolvería a la escena el color gélido y lo más importante, detendría por completo la *corrutina*.

Mientras tanto, para dar la sensación de que hace mucho calor en la zona, se cambiará la temperatura de los colores de la opción de posprocesado *Color Grading*[58], consiguiendo un tono anaranjado en la escena (Figura 69), a través de la modificación del parámetro *Temperature* de la propia opción.

Figura 69. Cambio de color de la escena al entrar en yermo. Elaboración propia.

A la vez que se activa el gradiente de color de la escena, se activarán también unas partículas con forma de gotas (Figura 70) que irán cayendo a medida que pasa el tiempo, dando la sensación de que Pol se está derritiendo.

Figura 70. Pol en el yermo empezando a gotear. Elaboración propia.

[58] Efecto que altera el color de la imagen final producida por Unity

6.6 Inventario e interacción con objetos

La interacción con objetos, junto con el inventario, se pensó como unos de los pilares del juego tanto para la progresión como para el diseño del mapa, ya que para poder avanzar de zona o poder acceder a un objeto clave, sería necesario tener un objeto específico o una cantidad fija de estos en el inventario. El funcionamiento básico de la interacción se podría dividir en tres tipos: la interacción con herramientas, que se añadirán al inventario para ser equipadas o desequipadas; la interacción con objetos que no requieren de herramienta para ser utilizados; y la interacción con objetos que requieren de una herramienta específica para poder ser utilizados. Por otro lado, debido a como el sistema ha sido construido, es necesario explicar ciertas cosas antes. Por último, he de destacar que la interacción con PNJ no entra dentro de este sistema, ya que requiere de otros componentes y sistema dedicado, explicado en el trabajo de Vicente Pla Madrid.

6.6.1 Estructura de la escena

Para que el inventario funcione y con ello se pueda interactuar con algunos de los objetos de la escena, es necesario que otros componentes y sistemas estén presentes en esta. Los objetos que contienen los engranajes para que el nivel pueda funcionar están agrupados dentro de un objeto llamado *Level* (Figura 71), que contiene el controlador general del nivel, el cual posee referencias a todos los subsistemas enumerados a continuación; el controlador de la música, que se encargaría de controlar los valores de reproducción de los sonidos de fondo y la música del juego; la cámara necesaria para la visualización del nivel; el objeto que contiene todo lo referente al jugador, cuya función ya ha sido explicada anteriormente; el sistema de diálogos, vinculado fuertemente al inventario; y por último el inventario. Estos sistemas, con sus parámetros y referencias a objetos independientes, tienen en común una única referencia, la del controlador del nivel.

Figura 71. Objeto Level desplegado. Elaboración propia.

Esta estructura viene dada debido a la elección de diseño del código, estando basado en controladores maestros cuya comunicación es únicamente de forma interna con los componentes que derivan de él y con el controlador del nivel de ese momento (Figura 72), de forma que, si precisa de algún dato externo, deberá hacer uso de las funciones pertinentes para el dato que requiera de un componente preciso. De esta forma, al momento de crear nuevos niveles y añadirles los sistemas desarrollados, en caso de faltar alguno, este no rompería el

funcionamiento del resto. De igual forma, se creó un objeto prefabricado que los contuviese a todos, así como las referencias base necesarias para agilizar el proceso de creación.

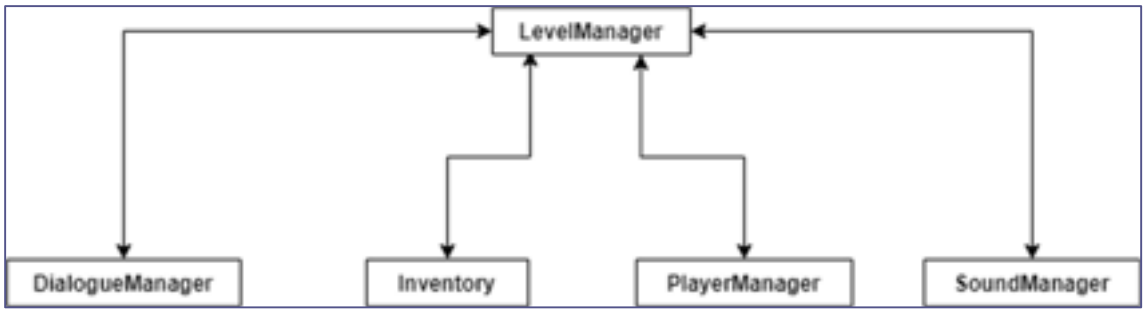

Figura 72. Flujo básico de información entre los controladores de un nivel. Elaboración propia.

6.6.2 Inventario

Principalmente, el inventario se encarga tanto de gestionar los objetos que pueden ser equipados encontrados en el mapa, como todos aquellos con los que se puede interactuar dentro de un nivel. Como ya se ha mencionado antes, este contiene una referencia al controlador del nivel, *LevelManager*, además de referencias a los controladores de los subsistemas (Figura 73) que él mismo contiene: uno encargado de gestionar el menú visible al abrirlo y el resto como conexión entre el inventario y los objetos interactivos.

Figura 73. Objeto Inventory desplegado. Elaboración propia.

Dado que el inventario es un objeto reutilizable y a usar en los distintos niveles, siendo algo independiente entre niveles. Este puede configurarse para que, dependiendo del nivel, aparezcan más o menos objetos, declarándolos previamente a través del inspector. Para ello, el inventario mantiene una lista de estos (Figura 74), cada uno de los cuales poseerá: el nombre de variable que se le ha dado a dicho objeto, el hecho de si se puede o no equipar, el nombre de la animación que empleará el avatar del jugador cuando use dicho objeto, la cantidad de estos que se pueden poseer, si ha sido usado o no, y las dos imágenes con las que representarán dentro de la interfaz de inventario en caso de ser necesario, representando si el objeto/herramienta está equipada o no.

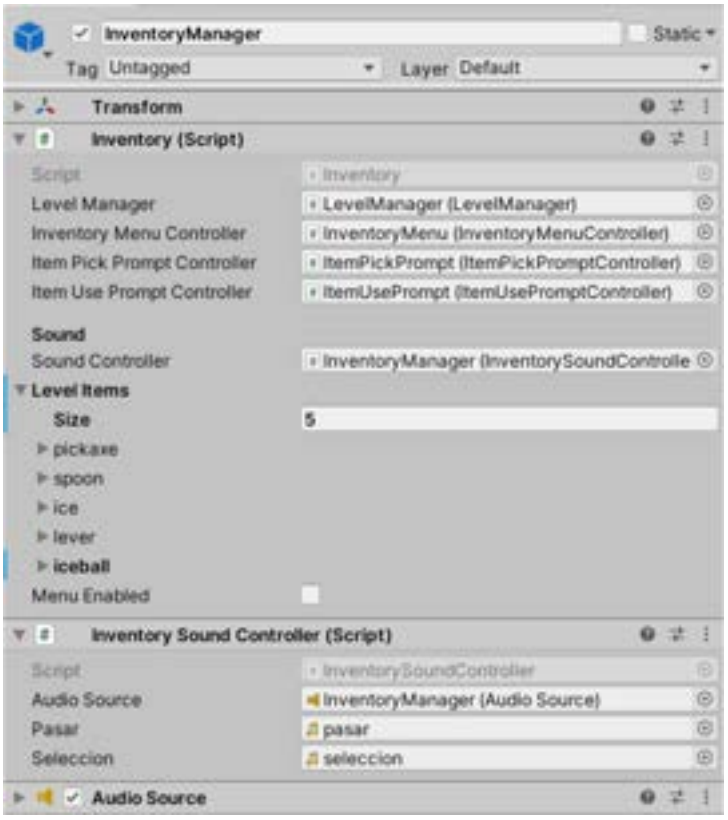

Figura 74. Objeto InventoryManager desde el inspector. Elaboración propia.

La lista antes mencionada, de nombre *LevelItems*, está compuesta por objetos de tipo *ItemInfo*, con la información listada anteriormente. De la misma forma, existe una segunda lista, *Items*, encargada de gestionar los objetos en posesión del jugador, también de tipo *ItemInfo*. A la que se irán añadiendo o quitando los objetos a medida que vayamos interactuando con ellos o usándolos, quedando marcados como usados en el inventario o añadiéndose más cantidad a otros.

Una vez en la lista, se podrán consultar: la cantidad que se posee de cada uno, normalmente cero, aunque en ocasiones se requerirá de cierta cantidad para cumplir ciertos objetivos; si ha sido equipado o usado; y si está en el inventario, además de poder recuperarlos para gestionarlos. Para vincular estas operaciones con los elementos relacionados con el inventario y viceversa, el modelo del jugador, las animaciones de interacción y la propia interfaz del usuario, se ha hecho uso de los eventos propios que el lenguaje empleado, en este caso C#, ofrece. En el caso de la interfaz, está suscrita a tres eventos, a diferenciar entre *ItemPicked*, el cual se invoca desde el *script Inventory* cuando el jugador coge un objeto, añadiendo la imagen del objeto correspondiente a la interfaz; el evento *ItemRemoved*, el cual quitará la imagen del objeto pasado como argumento; y, por último, *ItemUpdated*, que actualizará el contador de los objetos de los cuales se pueda tener más de uno al interactuar con un objeto del mismo tipo. En segundo lugar, para que al equipar o desequipar un objeto este aparezca o desaparezca del modelo del jugador y haga la animación de coger un objeto, el *script* de *LevelManager* está suscrito a los

eventos: *ItemEquipped* e *ItemUnequipped*, encargados de, al recibir la señal, darle la tarea al *PlayerManager* de activar y desactivar las herramientas (Figura 75) incluidas en el modelo de Pol, el pico y la pala. Además, también está suscrito a *ItemPicked*, encargado también de decirle al *PlayerManager* que reproduzca la animación de coger un objeto.

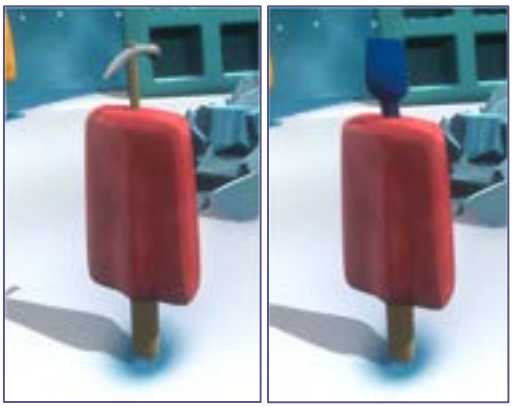

Figura 75. Pol con las herramientas disponibles equipadas. Elaboración propia.

Interfaz de usuario

Como ya se ha mencionado en el apartado anterior, la interfaz del inventario posee imágenes de los objetos almacenados en la lista de objetos, *Items* del jugador, de tal forma que podemos navegar por esta y seleccionar para equiparnos (Figura 76) un objeto en caso de ser posible. Esta consiste en las imágenes de los objetos dispuestos en forma de fila, cada una de las cuales posee dos estados, equipado o no, tal y como se puede ver en la Figura 77, y de su nombre y una pequeña descripción.

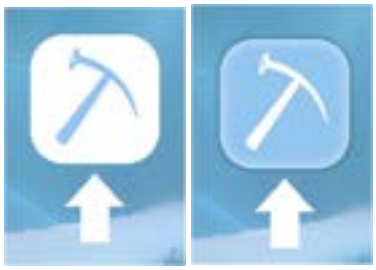

Figura 76. Estados del pico en el inventario (desequipado y equipado). Elaboración propia.

Figura 77. Inventario del jugador con 3 objetos. Elaboración propia.

Tanto el funcionamiento como la disposición han sido inspiradas en el inventario del videojuego *A Short Hike (*Figura 78*)*, que al igual que en nuestro juego, presenta las imágenes de los objetos que se poseen a seleccionar mediante una flecha en la parte inferior. Así pues, cada vez que se quiera hacer uso de una herramienta o comprobar si se posee un objeto o cierta cantidad de estos, será necesario abrir el menú del inventario, haciendo obligatorio el uso de este.

Figura 78. Inventario del videojuego A Short Hike. A Short Hike (2019)

En cuanto a la implementación y el funcionamiento, el *script* que gestiona la interfaz, *InventoyMenuController*, al añadirse un objeto mediante la escucha del evento *ItemPicked* del controlador del inventario, se añade un objeto prefabricado, al cual se le cambiará la imagen por la del objeto correspondiente, además, este objeto también incluye la flecha de selección de objeto, debido a que al navegar por el menú, en lugar de desplazar la flecha, de forma interna se recorre la lista de objetos con cada pulsación de las teclas que lo controlan y se desactivan y activa las flechas correspondientes, de forma que cuadrar que esta esté en el lugar indicado se convierte en una tarea simple, ya que no hace falta actualizar la posición relativa cada vez que la lista visible en la interfaz crezca lo que hace que se reajuste la posición del resto de elementos. Una vez clara la navegabilidad por este, al situarnos sobre la herramienta a equipar deseada y seleccionarla, desde el controlador de la interfaz se realizarán las siguientes acciones: comprobar si ya se tiene alguna otra herramienta; cambiar el estado de la imagen seleccionada por la de seleccionado y en caso de tener otra equipada, cambiar su imagen por la de desactivado; y por último invocar al evento *ItemEquipped* mediante la función *EquipItem* del controlador del inventario, que como se mencionó en el apartado anterior, hará que desde el *LevelManager*, el controlador del nivel, comunique al controlador general del jugador, que active y desactive el modelo de la herramienta correspondiente.

6.6.3 Objetos interactivos

Los objetos con los que se puede interactuar u objetos clave, son objetos que se irán añadiendo al inventario conforme los vayamos recogiendo, ya sea para usarlos activamente o para mantenerlos en el inventario hasta que algún PNJ nos lo requiera para desbloquear alguna zona u acción especial, otros, en cambio, formarán parte del entorno con los que podremos interactuar si poseemos la herramienta necesaria. Así pues, como ya se mencionó al principio de este apartado, la interacción con objetos consiste en tres tipos diferentes: la interacción con herramientas, la interacción con objetos que no requieren de ninguna herramienta para poder interactuar y por último los que sí que requieren de una. Para su implementación, de estas tres opciones, se hizo una separación en dos grupos, los objetos que al interactuar solamente serían recogidos por el jugador, denominados *ItemPicker* (Figura 79), como las herramientas, que se añadirán al inventario y los objetos utilizables del entorno, denominados *ItemUser* (Figura 80), como los montículos de nieve, que previo a obtener el resultado de la interacción, le precede una animación o un posicionamiento específico de forma automática.

Figura 79. Ejemplos de ItemPicker situados en el nivel. Elaboración propia.

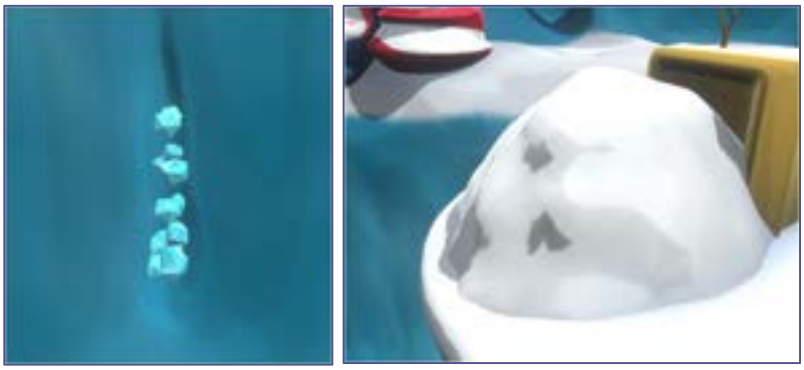

Figura 80. Ejemplos de ItemUser situados en el nivel. Elaboración propia.

Para su implementación, del controlador de cada uno de los dos tipos de objetos, se le extrajo a un *script* diferente la detección de colisiones y de *triggers*[59], ya que, para interactuar, el inventario debe saber también si se está cerca o no de un objeto antes de poder realizar alguna gestión, de esta manera tanto el inventario, de forma visual, como el jugador de forma visual,

[59] Colisiones sin cuerpo que no bloquean el paso del jugador a no ser que se especifique vía código.

mediante la habilitación del parámetro *Selected Color* del material[60], saben que la interacción es posible (Figura 81).

Figura 81. Cambio de color de un objeto al acercarse a él. Elaboración propia.

En cuanto a la estructura del controlador de los objetos, aunque en scripts diferentes, comparten las funcionalidades de: detectar si el jugador está entrando o saliendo del radio de acción del objeto, siendo llamadas mediante sus respectivos scripts de detección (*TriggerUseItem* y *TriggerPickItem);* la función que activa el efecto que sombrea los objetos interactivos con un color anaranjado; y por último la función que principalmente deshabilita el objeto con el que se ha interactuado, habiendo diferencias clave en su ejecución según el tipo y siendo invocadas siempre al ser ejecutadas desde el inventario cuando la tecla de ejecución es oprimida.

Estando presente la diferencia antes mencionada, esta se basa en que, al interactuar con un *ItemPicker*, este es añadido al inventario directamente, sin importar si el avatar debe hacer alguna animación en algún punto en específico. Sin embargo, para los *ItemUsers* esto es diferente, ya que siempre se interactúa con algún objeto integrado en el entorno de forma que para, por ejemplo, picar una mena, es necesario que la animación correspondiente sea efectuada en una posición y orientación específica, de forma que esta no atraviese el objeto o parte del mapa más de lo necesario o realice la acción mirando en la dirección opuesta.

Así pues, parecido a lo que ocurría cuando se interactuaba con el palo para volver a la forma normal, al interactuar con un objeto primero se comprobará si se requiere de alguna herramienta y si se tiene equipada. En caso de no tenerla aparecerá un mensaje sugiriendo que quizás necesites de algo para poder realizar la acción, de no tenerla equipada o no requerir de ninguna, se entrará en el modo interacción. De cualquier modo, tendremos disponibles las posiciones de *ItemPos* e *ItemLook* a través de las funciones destinadas a la recuperación de estos datos del *script ItemUser*, encargadas de proporcionárselas al *PlayerManager* a través del inventario cuando el jugador se acerque a un objeto. Al entrar en el modo interacción, se perderá el control

[60] Componente de un objeto que define como debe ser renderizado en caso de tener una malla o similar que le de forma.

del personaje hasta que esté posicionado y orientado respecto a las posiciones antes obtenidas. Al mismo tiempo, se iniciará un *corrutina*, que se quedará esperando hasta que se salga del modo interacción, es decir cuando el avatar del jugador esté correctamente posicionado mediante el movimiento automático proporcionado por la función *MoveToTarget* de *PlayerBase* las posiciones relativas del objeto. Una vez la *corrutina* salga del bucle en el que estaba esperando, se ejecutará la función equivalente a la que se ejecutaba cuando se recogía un *ItemPicker*, pero añadiendo los efectos de sonido y visuales característicos de cada objeto. Por otro lado, pensando en que, para avanzar por uno de los futuros niveles, sería posible que un PNJ encomendase la tarea de interactuar con cierto número de objetos del mismo tipo, se decidió primero hacer la clase padre, *ItemUser*, abstracta dejando únicamente la implementación de las funcionalidades comunes (detección del jugador, sombreado del objeto y destrucción) y realizando para cada tipo de objeto una clase que heredara de objeto, con la implementación de la función *OnUse* independiente en cada una, pudiendo añadir los efectos y las animaciones extras deseadas.

Por último, hay que destacar que, al interactuar con un objeto, no tiene por qué añadirse este al inventario o algún objeto derivado, estos simplemente pueden ser destruidos para abrir el camino, como ya se mencionó, o desencadenar secuencias de eventos en las que diversas cosas suceden a la vez mediante la ejecución de una *Timeline*[61], como se puede ver diversas veces dentro del nivel desarrollado.

6.7 Cámara del juego

La cámara en un videojuego es un elemento importante debido a que puede llegar a definir como se juega un videojuego o como los niveles de este son diseñados. Como ejemplo tenemos el desarrollo de este juego mismo, ya que, en un principio, la cámara iba a ser controlable por el jugador, permitiendo diversos ángulos, pensando así en un entorno extenso, sin embargo, dados los problemas de diseño que eso supuso y las complicaciones en cuanto a la implementación que suponía que la cámara realizara cambios de puntos de vista o seguimientos de objetos más allá del jugador, se decidió empezar a utilizar la herramienta *Cinemachine* que Unity proporcionaba como paquete opcional descargable. Basando su funcionamiento en situar cámaras virtuales sobre las que se podrá ir desplazando la cámara principal de la escena en base a prioridades modificables vía código, se eligió una de las posibilidades que ofrecía, que consistía en situar la cámara virtual sobre un carril como si de un carro se tratara. Así pues, dado el diseño del mapa de pirámide de anillos, donde cada nivel podría estar dividido en subzonas, se situó un carril principal en el anillo inferior, el cual no estaba subdividido, y uno por cada segmento del nivel superior, que constaba de tres zonas, adicionalmente se situó alguna cámara extra para hacer énfasis en esa zona en concreto. Como se puede apreciar en la Figura 82, cada

[61] Herramienta para la secuenciación y reproducción de animaciones de más de un elemento a la vez.

carril hace referencia a la zona en la que está situada según su nombre, en verde están los planos que sirven de colisión para los cambios de zona, los símbolos de cámara son los puntos (cámaras virtuales) sobre los que pivotará la cámara del juego.

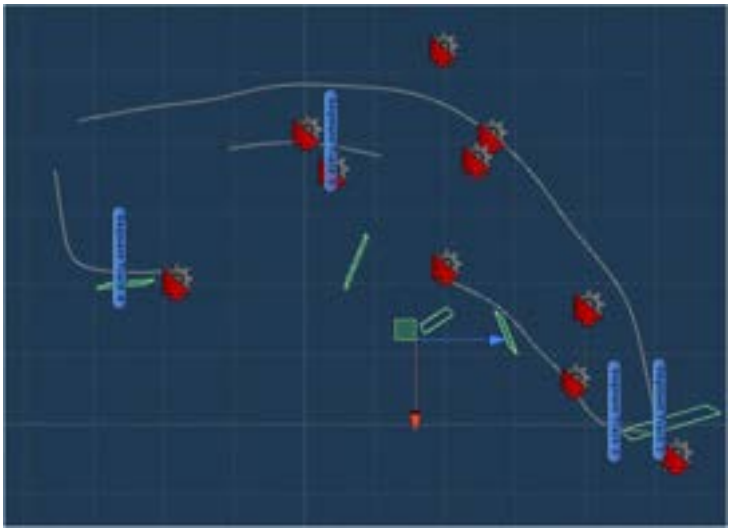

Figura 82. Sistema de cámaras visto desde arriba y sin el nivel. Elaboración propia.

El funcionamiento de las cámaras, al estar situadas sobre carriles, consiste en seguir de forma automática al jugador habilitando la opción desde el componente que nos ofrece *Cinemachine* al crear una cámara virtual, realizando el cambio de carril según una serie de colisiones situadas en lugares de transición entre zonas del mapa de un nivel inferior a uno superior, o según la distancia del jugador con respecto al carril de la cámara activa en ese momento.

El carril de la zona principal del nivel, la zona baja, contiene tres cámaras virtuales sobre las que se irá cambiando en todo momento según la distancia, situando una de ellas sobre el propio carril (Figura 83), que se activará a una distancia media de este; otra más elevada y hacia delante (Figura 84), de forma que se tiene una vista más general del terreno, la cual se activará cuando haya una distancia considerable entre jugador y carril; y la última alejada del carril (Figura 85) y casi al nivel de los pies del jugador, la cual se activará cuando se esté muy cerca.

Figura 83. Cámara principal del carril, zona principal. Elaboración propia.

Figura 84. Cámara desde arriba del carril, zona principal. Elaboración propia.

Figura 85. Cámara desde abajo del carril, zona principal. Elaboración propia.

Por otro lado, las zonas superiores no disponen de tanta profundidad como la zona baja, así que solo poseen una cámara (Figura 86). Para realizar el cambio de cámara de la zona baja a una superior, se hará según entremos a la colisión (*trigger*) situada al principio de la zona, mientras que, en el caso contrario, esta cambiará según la distancia entre el jugador y el carril activo en ese momento, pero solo teniendo en cuenta la distancia entre las posiciones del eje Y de coordenadas.

Figura 86. Cámara del segmento dos del mapa. Elaboración propia.

En cuanto al control general de las cámaras y su implementación, se agruparon las diferentes cámaras virtuales y sus colisiones de cambio de zona, junto con otras cámaras que permanecen

dentro de la estructura, las cuales son estáticas, obteniendo como resultado el objeto *LevelVCams*. Este objeto, al contener el controlador, posee una referencia a cada una de las cámaras virtuales principales de los carriles, gestionando el cambio de cámara de los niveles superior al inferior mediante el cálculo de la distancia entre las posiciones del eje Y de coordenadas. Teniendo esta que ser menor que las señaladas desde el inspector (variables que empiezan por *Diff)* según la cámara activada en ese momento (Figura 87).

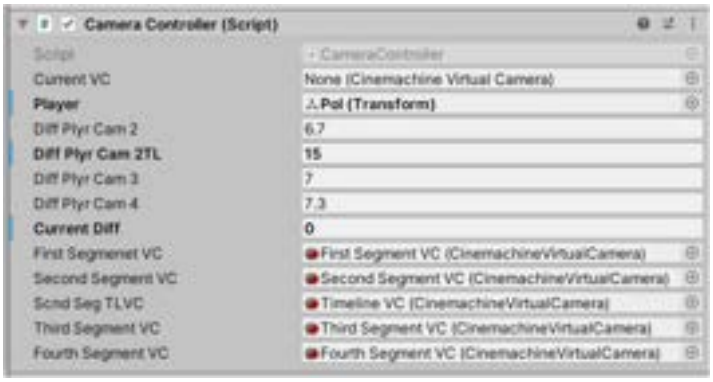

Figura 87. CameraController desde el inspector

Por último, para saber qué cámara principal está activada en cada momento, se añadió a cada detector de colisiones de zona, un controlador (Figura 88) encargado de realizar el cambio de prioridad correspondiente a las cámaras, teniendo en cuenta el valor de la variable *IsMain* que, al marcarse, se comunica con el controlador de las cámaras para dejar constancia de cual está activa. Adicionalmente, existe la posibilidad de que el detector funcione una únicamente vez, marcando la opción *Disable*, obligando a que se desactive una vez se pase por él, pudiendo añadir cámaras temporales para destacar ciertas zonas o eventos.

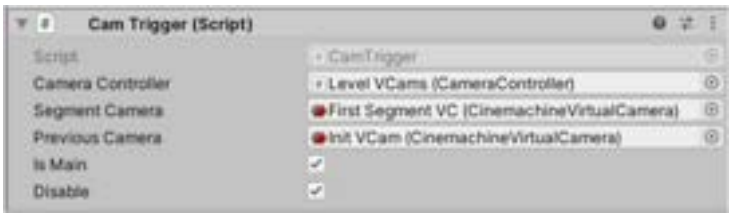

Figura 88. CamTrigger desde el inspector

7. Conclusiones

Como se puede apreciar, como equipo, hemos conseguido materializar una serie de ideas y convertirlas en el prototipo de *Frozen Out*, ya que partiendo de una experiencia casi nula en el desarrollo de proyectos de una envergadura destacable dados nuestros medios, hemos sabido organizarnos para sacar las cosas adelante. Se ha conseguido construir un entorno cómodo en el que trabajar, en el que se tiene en cuenta las opiniones de cada uno de los integrantes para obtener un resultado en el que todos en mayor o menor medida estemos conformes. Un entorno en el que la colaboración a la hora de sacar tareas adelante es un punto clave, dada la experiencia previa que cada uno pudiera tener o los diferentes puntos de vista dado el carácter multidisciplinar del proyecto, ya que al haber tareas que requieren de ambos lados, arte y programación, no habría sido posible, siendo además la organización un punto clave para poder tener características desarrolladas o diseñadas para que el equipo avanzara de forma conjunta.

En cuanto a los contenidos de esta memoria, se ha conseguido encontrar los puntos fuertes en los que el producto en desarrollo podrá destacar en el mercado, teniendo en cuenta las debilidades frente a sus competidores y aun sabiendo que pocos proyectos de carácter indie tienen un éxito abrumador que permita a los desarrolladores vivir plenamente de él. Por otro lado, se ha conseguido que la construcción de nuevos niveles sea sencillo dada la modularidad de los componentes clave que componen la escena, habiendo mejorado en referencia a las primeras versiones donde encontrar las referencias y objetos pertinentes era una tarea tediosa, además con el desarrollo de las mecánicas del juego y el apoyo de las herramientas que Unity pone a disposición de los desarrolladores, así como con la implementación de sistemas para el control del entorno, he conseguido asentar mis conocimientos en el uso motor, adquiridos tanto en clase como de forma autónoma.

Pese a lo anterior mencionado, tanto como equipo, como yo personalmente, somos conscientes de que hay muchas cosas que mejorar o pulir, pero que no dejan de ser consecuencias de decisiones que hemos tomado como equipo y que tendremos que solucionar como tal, siendo esto resultado de la colaboración durante tanto tiempo y que pasará a formar parte de los primeros pasos en la industria y de la historia de Frozen Out.

8. Trabajo futuro

Dado las respuestas obtenidas en la encuesta y las cosas a mejorar tras revisiones, es innegable que hay que pulir ciertos aspectos referentes a la jugabilidad y la idea que se quiere transmitir, no solo por el entorno sino también por el aspecto jugable, ya que moverse por ciertas zonas es un tanto molesto y la disposición de las cámaras no ayudan a ello. Por lo que, además de arreglar ciertos bugs encontrados tras la fase de pruebas, se tendrán que volver a definir aspectos antes mencionados y encontrar alguna mecánica que conecte la jugabilidad con la idea a transmitir.

Por otro lado, dadas las tareas que quedaron pendientes en el *backlog*, como los puntos de control y el cargado de la partida, estas se realizarán a la par que el rediseño de los aspectos antes comentados adaptando los necesarios.

Por último, se empezará a diseñar el siguiente nivel y una introducción para situar en contexto a los jugadores, ya que el nivel desarrollado no corresponde con el principio de la historia y puede dejar descolocados a los usuarios si no poseen algún texto explicativo o a los desarrolladores guiándolos mientras juegan.

9. Referencias bibliográficas

AEVI. (14 de abril de 2019). *La industria del videojuego en España.* Recuperado el 29 de junio de 2020, de http://www.aevi.org.es/web/wp-content/uploads/2020/04/AEVI-ANUARIO-2019.pdf

Carrasco, A. J. (2020). *FROZEN OUT (I): Diseño y desarrollo artístico de un videojuego crítico 3D.*

itch.io. (2020). *Open revenue sharing*. Recuperado el 29 de junio de 2020, de itch.io: https://itch.io/docs/general/about

Kickstarter. (20). *Estadísitcas de Kickstarter*. Recuperado el 27 de agosto de 2020, de Kickstarter: https://www.kickstarter.com/help/stats?lang=es

Kickstarter. (2020). *Temtem - Massively multiplayer creature-collection adventure*. Obtenido de Kickstarter: https://www.kickstarter.com/projects/cremagames/temtem-massively-multiplayer-creature-collection-a/description

Madrid, V. P. (2020). *"Frozen Out", videojuego de aventura gráfica 3D. Diseño y uso de los diálogos.*

Newzoo. (2020). *Top 10 Countries/Markets by Game Revenues*. Recuperado el 29 de junio de 2020, de Newzoo: https://newzoo.com/insights/rankings/top-10-countries-by-game-revenues/

Orrios, P. L. (2020). *Desarrollo de un videojuego con Unity: implementación de la inteligencia artificial y funciones back-end.*

Pearson, D. (14 de febrero de 2011). *Steam controls 50-70% of PC download market*. Recuperado el 29 de junio de 2020, de GamesIndustry.biz: https://www.gamesindustry.biz/articles/2011-02-14-forbes-steam-controls-50-70-percent-of-pc-download-market

Sáez, A. R. (2020). *FROZEN OUT (II): Diseño de interacción y animación de caracteres de un prototipo de videojuego crítico 3D.* .

Steam. (2020). *Joining The Steamworks Distribution Program*. Recuperado el 29 de junio de 2020, de Steamworks: https://partner.steamgames.com/steamdirect

Valentine, R. (1 de diciembre de 2018). *Valve adds revenue share tiers for developers*. Recuperado el 29 de junio de 2020, de GamesIndustry.biz: https://www.gamesindustry.biz/articles/2018-12-01-valve-adds-revenue-share-tiers-for-developers

10. Índice de ilustraciones

11. Anexos

Guía del prototipo de Frozen Out

A continuación, explicaré cómo sería una partida al nivel de Frozen Out, desarrollado como prototipo, en su última versión, disponible en: <https://github.com/Freezer-Games/Frozen-Out/wiki>

Al iniciar el juego, lo primero que veremos será el menú de inicio, donde podremos empezar directamente la partida o cambiar algún ajuste antes desde de empezar desde el menú de opciones.

Figura Anexo 1. Menú principal. Elaboración propia.

Si elegimos entrar al menú de opciones, se nos mostrará el siguiente menú, donde podremos cambiar los ajustes del juego, audio, gráficos y controles.

Figura Anexo 2. Menú de opciones. Elaboración propia.

Una vez elegimos la opción de «Comenzar», el nivel empezará y apareceremos al principio de este, como se puede ver en la imagen de abajo. A partir de aquí podremos movernos libremente por el mapa.

Figura Anexo 3. Principio del prototipo de Frozen Out. Elaboración propia.

Conforme avanzamos, podremos encontrarnos con un pico en el suelo el cual al acercarnos se iluminará y podremos coger.

Figura Anexo 4. Pol cerca del pico iluminado. Elaboración propia.

Al coger el pico, este aparecerá en nuestro inventario, pudiendo ser equipada para ser usada más adelante. Al equiparlo, se añadirá al personaje, permaneciendo sobre su cabeza mientras lo lleve.

Figura Anexo 5. Inventario tras coger el pico. Elaboración propia.

A su vez, cerca del pico encontraremos a un polo trabajando, con el que podremos hablar para recibir información valiosa sobre los problemas que nos podremos encontrar más adelante y como evitarlos.

Figura Anexo 6. Pol hablando con un polo trabajando. Elaboración propia.

En la misma zona podremos encontrarnos al primer enemigo y un cubo con hielo, el cual será un objeto clave para más adelante. Dado que el cubo está bloqueado por la patrulla, resultará inaccesible por el momento, ya que de acercarnos acabará la partida.

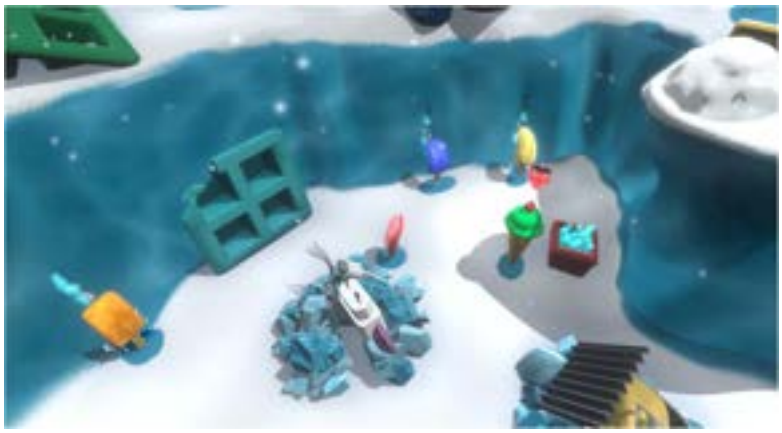

Figura Anexo 7. Pol cerca de un enemigo a punto de atraparle. Elaboración propia.

Al entrar a la siguiente zona, veremos un pedal de bicicleta convertido en ascensor. Al acercarnos, el trabajador encargado de hacerlo girar nos dirá que no puede ayudarnos a subir porque ha perdido la manivela, por lo que hasta que no se la entreguemos no podremos avanzar. Si elegimos obviar el problema, no podremos avanzar, ya que el acceso a la siguiente zona está bloqueado por dos guardias.

Figura Anexo 8. Ascensor de la segunda zona inferior del mapa. Elaboración propia.

Figura Anexo 9. Segunda ruta de la zona dos cortadas por dos enemigos. Elaboración propia.

Al entregar la manivela, la cual se encuentra por la zona, y entregársela a su dueño, podremos elegir cuál de las dos zonas superiores adyacentes visitar. Dado que el objetivo principal es salir de la zona para llegar a la nevera, elegiremos la zona de la derecha, en la que nos encontraremos dos cortes de helado, que nos comunicarán que, para poder seguir, necesitamos darles cien trozos de hielo. Además, nos encontraremos a un polo preocupado por una máquina que está provocando derrumbamientos.

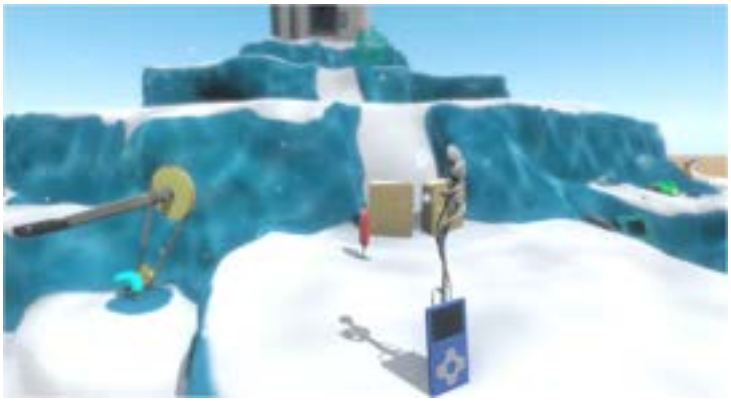

Figura Anexo 10. Segunda zona superior y los cortes de helado custodiando la salida. Elaboración propia.

Con la tarea encomendada por los cortes y con el cubo de hielo visto en la zona anterior en mente, podremos hacer dos cosas, intentar avanzar por la zona cortada por los guardias desde la zona en la que nos encontramos o buscar la manera de conseguir el cubo custodiado por el guardia del principio.

Si intentamos conseguir el cubo de hielo primero, necesitaremos acceder a la primera zona superior mediante el ascenso. Al llegar, veremos una mena de hielo en la que podremos picar, por lo que, si interactuamos con el pico equipado, podremos recoger cierta cantidad de hielo, aunque no será suficiente para los 100.

Figura Anexo 11. Pol picando en la primera zona superior. Elaboración propia.

No muy lejos de ahí, podremos encontrar un montículo de nieve, que al interactuar con él y no llevar la herramienta indicada, nos dirá que es lo que nos hace falta, siendo en este caso la cuchara, la cual podremos obtener si sorteamos los obstáculos de la zona, pasando las dos latas de refresco.

Figura Anexo 12. Pol interactuando con un montón de nieve. Elaboración propia.

Figura Anexo 13. Pol consiguiendo la pala. Elaboración propia.

Una vez tengamos la pala, si regresamos al montón de nieve podremos usarla sobre él, lo que hará que se reproduzca una animación en la que empujaremos la nieve para tirársela al guardia

de la zona inferior, enterrándolo y dejando de ser un problema, pudiendo acceder sin problemas al cubo de hielo que custodiaba.

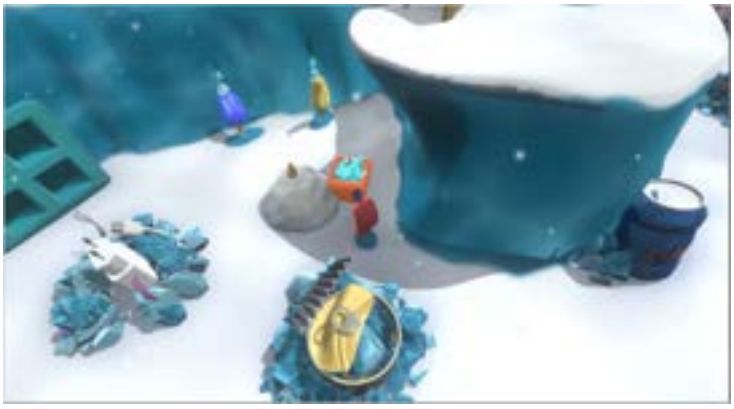

Figura Anexo 14. Pol recogiendo el hielo del cubo tras haber enterrado al guardia. Elaboración propia.

Nuestro siguiente paso será el de alcanzar la tercera zona inferior mediante la segunda zona superior, por lo que volveremos allí para esta vez saltar. Una vez allí veremos una batidora, la máquina que nombró un polo anteriormente, por la que podremos subir para alcanzar la tercera zona superior, el único camino disponible que nos queda.

Figura Anexo 15. Tercera zona inferior y la batidora. Elaboración propia.

Una vez en esta zona, si exploramos un poco, podremos ver que hay unas latas cerca de la batidora de forma similar que en la primera zona superior. Así pues, si conseguimos subir encima de la batidora, podremos acceder a la tercera zona superior.

Figura Anexo 16. Pol saltando desde la batidora a la tercera zona superior. Elaboración propia.

Una vez en esta zona, si avanzamos, podremos ver un polo encerrado dentro de un montón de nieve a casa de los temblores de la batidora. Para rescatarlo, necesitaremos tener la pala equipada.

Figura Anexo 17. Pol hablando con un polo tras rescatarlo. Elaboración propia.

Al rescatarlo y hablar con él, nos ofrecerá la posibilidad de sacar una bola de nieve de la caja que estaba usando antes de quedarse atrapado. Además, nos explicará que si la hacemos rodar aumentará de tamaño. Si cuando obtenga su tamaño máximo, la acercamos a la cubitera situada en forma de rampa, se activará una cinemática en la que la bola de nieve arrasará con lo que tenga a su paso, ahuyentando así a los guardias que bloqueaban el camino de la zona inferior 2 a la 3.

Figura Anexo 18. Bola de nieve arroyándolo todo a su paso. Elaboración propia.

Ahora, si pasamos por donde los helados cortaban el camino, observaremos dos cubos llenos de hielo, con los cuales, al cogerlos conseguiremos la suma de hielo total necesaria, pudiendo tener más si hemos ido picando en los diferentes rincones habilitados para ello en el mapa.

Figura Anexo 19. Pol cogiendo el hielo de los cubos. Elaboración propia.

Con la cantidad obtenida, si regresamos a la salida custodiada por los cortes de helado y volvemos a hablar con ellos, al ver la suma de hielo nos habilitarán el paso, pudiendo así salir de la mina para ir a la ciudad.

Figura Anexo 20. Cortes de helado permitiendo a Pol pasar tras cumplir su petición. Elaboración propia.

Tras dejarlos atrás, conseguiremos salir y una cinemática se activará, llevando la cámara hasta la nevera, donde tras pasar por la entrada del alcalde mientras se acerca a esta, podremos ver los créditos en la pantalla que tiene incorporada.

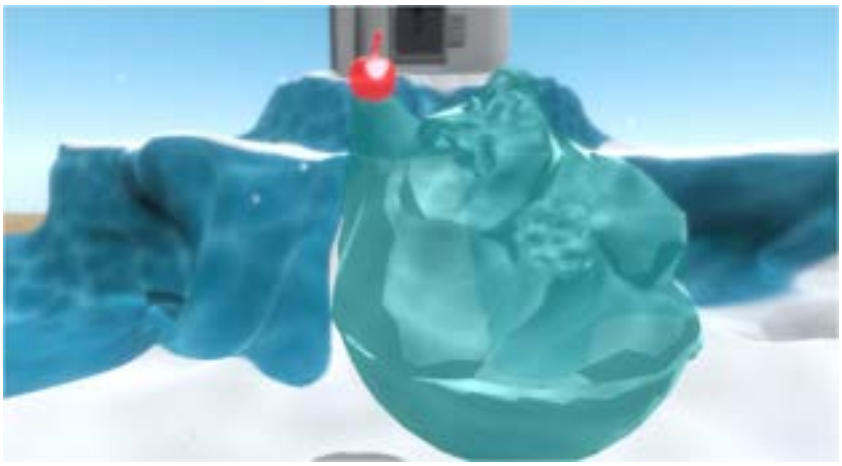

Figura Anexo 21. Estatua del alcalde McTopping en la transición de la cinemática final. Elaboración propia.

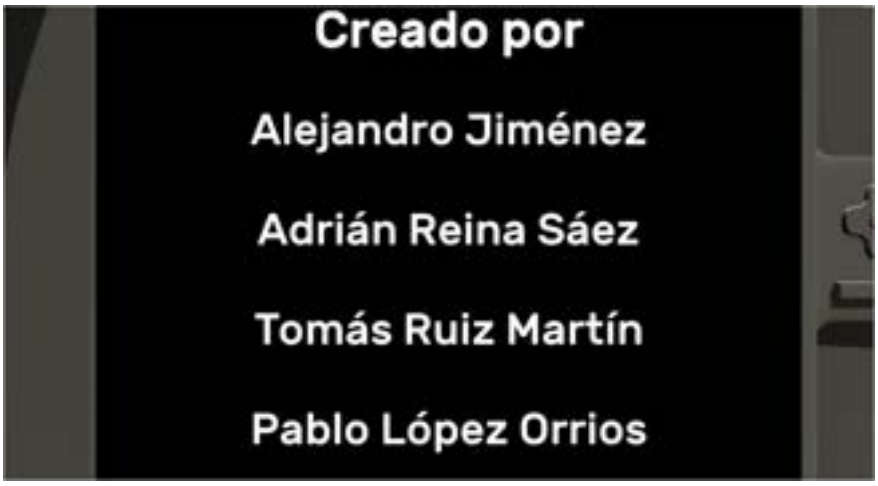

Figura Anexo 22. Fragmento de los créditos tras haber acabado el juego. Elaboración propia.