UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería de Telecomunicación

BioPlay, Desarrollo de un videojuego 2D didáctico sobre biomedicina junto a su plan de marketing digital

Trabajo Fin de Grado

Grado en Tecnología Digital y Multimedia

AUTOR/A: Rodriguez Fernandez, Marcos

Tutor/a: Guerola Navarro, Vicente

CURSO ACADÉMICO: 2023/2024

Resumen

Este proyecto se centra principalmente en crear un videojuego 2D utilizando Unity como plataforma de desarrollo. Por otro lado, se tiene como objetivo transversal plasmar un plan de marketing digital con proyección a su futura salida al mercado.

Se busca incorporar en este Trabajo Final de Grado las etapas iniciales del videojuego, explicar el desarrollo necesario y por último mostrar su acabado final. A diferencia de diversos juegos de entretenimiento, "BioPlay" propone abrazar la diversión con el didactismo. En concreto, pretende enseñar conceptos básicos de biomedicina accesibles a cualquier tipo de públicos sin la necesidad de tener estudios técnicos en el campo. "BioPlay" aparte de ser didáctico se clasificaría en un videojuego de plataformas, disparos e incluso rompecabezas si nos centramos en su objetivo didáctico.

En cuanto al marketing digital, el planteamiento es el uso de las redes sociales y diversas técnicas para que el producto, tanto en su proceso como en su final, sea descubierto por diversos usuarios y ver resultados con la finalidad de enseñar de una manera sencilla la biomedicina al mundo.

Esta propuesta permite hacer uso de los conocimientos adquiridos en el grado de Tecnología Digital y Multimedia, puesto que se aplican técnicas de la asignatura de "Desarrollo de videojuegos" y "diseño gráfico" junto al plan de marketing perteneciente a la asignatura de "Modelos de negocio".

Palabras clave

Videojuego, Unity, 2D, Diseño, Programación, Biomedicina, Didactismo, Entretenimiento, Marketing digital.

Resum

Aquest projecte se centra principalment a crear un videojoc 2D utilitzant Unity com a plataforma de desenvolupament. D'altra banda, es té com a objectiu transversal plasmar un pla de màrqueting digital amb projecció a la seua futura eixida al mercat.

Es busca incorporar en aquest Treball Final de Grau les etapes inicials del videojoc, explicar el desenvolupament necessari i finalment mostrar el seu acabat final. A diferència de diversos jocs d'entreteniment, "BioPlay" proposa abraçar la diversió amb el didactisme. En concret, pretén ensenyar conceptes bàsics de biomedicina accessibles a qualsevol tipus de públics sense la necessitat de tindre estudis tècnics en el camp. "BioPlay" a part de ser didàctic es classificaria en un videojoc de plataformes, tirs i fins i tot trencaclosques si ens centrem en el seu objectiu didàctic.

Quant al màrqueting digital, el plantejament és l'ús de les xarxes socials i diverses tècniques perquè el producte, tant en el seu procés com en la seua final, siga descobert per diversos usuaris i veure resultats amb la finalitat d'ensenyar d'una manera senzilla la biomedicina al món. Aquesta

proposta permet fer ús dels coneixements adquirits en el grau de Tecnologia Digital i Multimèdia, ja que s'apliquen tècniques de l'assignatura de “Desenvolupament de videojocs” i “disseny gràfic” al costat del pla de màrqueting pertanyent a l'assignatura de “Models de negoci”.

Paraules clau

Videojoc, Unity, 2D, Diseny, Programació, Biomedicina, Didactisme, Entreteniment , Màrketing digital.

Abstract

This project focuses mainly on creating a 2D video game using Unity as a development platform. On the other hand, the transversal objective is to create a digital marketing plan with a projection for its future release to the market.

The aim is to incorporate in this Final Degree Project the initial stages of the video game, explain the necessary development and finally show its final finish. Unlike various entertainment games, “BioPlay” proposes to embrace fun with didacticism. Specifically, it aims to teach basic concepts of biomedicine accessible to any type of audience without the need to have technical studies in the field. “BioPlay”, apart from being educational, would be classified as a platform, shooting and even puzzle video game if we focus on its educational objective.

Regarding digital marketing, the approach is the use of social networks and various techniques so that the product, both in its process and its end, is discovered by various users and see results with the purpose of teaching in a simple way the biomedicine to the world. This proposal allows you to make use of the knowledge acquired in the degree of Digital and Multimedia Technology, since techniques from the subject of “Desarrollo de videojuegos” and “Diseño gráfico” are applied together with the marketing plan belonging to the subject of “Modelos de negocio".

Keywords

Videogame, Unity, 2D, Design, Programming, Biomedicine, Didacticism, Entertainment, Digital Marketing.

RESUMEN EJECUTIVO

La memoria del TFG del Grado en Tecnología Digital y Multimedia debe desarrollar en el texto los siguientes conceptos, debidamente justificados y discutidos, centrados en el ámbito de la tecnologías digitales y multimedia

CONCEPT (ABET)	**CONCEPTO (traducción)**	**¿Cumple? (S/N)**	**¿Dónde? (páginas)**
1. IDENTIFY:	1. IDENTIFICAR:		
1.1. Problem statement and opportunity	1.1. Planteamiento del problema y oportunidad	S	1-2, 6-7, 4, 55
1.2. Constraints (standards, codes, needs, requirements & specifications)	1.2. Toma en consideración de los condicionantes (normas técnicas y regulación, necesidades, requisitos y especificaciones)	S	2, 6-7, 8-12
1.3. Setting of goals	1.3. Establecimiento de objetivos	S	2
2. FORMULATE:	2. FORMULAR:		
2.1. Creative solution generation (analysis)	2.1. Generación de soluciones creativas (análisis)	S	13,15-17,21,27, 30,32, 33,35,42-44
2.2. Evaluation of multiple solutions and decision-making (synthesis)	2.2. Evaluación de múltiples soluciones y toma de decisiones (síntesis)	S	13-44
3. SOLVE:	3. RESOLVER:		
3.1. Fulfilment of goals	3.1. Evaluación del cumplimiento de objetivos	S	55
3.2. Overall impact and significance (contributions and practical recommendations)	3.2. Evaluación del impacto global y alcance (contribuciones y recomendaciones prácticas)	S	1-2,6-7,45,51,54,55

AGRADECIMIENTOS:

"En primer lugar, quiero expresar mi más sincero agradecimiento a mi madre, por recordarme cada día que era capaz de lograrlo. A mi hermana y a mi cuñado, gracias por comprender que no tenía tiempo ni siquiera para disfrutar del verano, y por confiar en mí en todo momento. A Marta y Maicas, os agradezco por brindarme el apoyo creativo que necesitaba para sacar adelante este proyecto. Pero, sobre todo, a ti, Diego: aún me cuesta creer que hayas tenido la paciencia para soportar mis enfados y mis agobios a lo largo de toda esta etapa. Sin tu paciencia y cariño, no habría logrado todo esto."

Índice

Capítulo 1. Introducción

1.1 Justificación

Los videojuegos, desde sus humildes comienzos en la era de los 70, hasta día de hoy, con los avances tecnológicos, nos han demostrado una progresión notable. Lo que comenzó como una simple forma de entretenimiento ha ido creciendo hasta convertirse en una industria influyente en numerosos aspectos de la sociedad moderna. Es por ello por lo que no se debe cometer el error de limitar (a pesar de no ser incorrecto) su uso a un mero pasatiempo.

Se pueden contemplar como poderosas herramientas para la educación y el desarrollo de habilidades. Es posible combinar la creatividad con el aprendizaje, haciendo que conceptos que a priori sean difíciles de digerir, mediante el entretenimiento podamos captar de una forma más sencilla e inteligible. Además, diversas investigaciones han demostrado que los videojuegos pueden mejorar el desarrollo cognitivo, fomentando habilidades como la resolución de problemas, el pensamiento crítico y la toma de decisiones.

Por otro lado, a pesar de que es aplicable con cualquier tipo de disciplina, la biomedicina es la protagonista de este proyecto, ya que es una disciplina fundamental en nuestro día a día que nos permite mejorar la comprensión y tratamiento de ciertas enfermedades, su relevancia se extiende a múltiples áreas de la salud. Es de vital importancia la creación de métodos que permitan a las futuras generaciones prepararse a afrontar ciertas cuestiones en este ámbito.

El desarrollo de plataformas de enseñanza como puede ser un videojuego no solo se debe enfocar en aquellos que particularmente quieren desempeñar esa profesión o sean estudiantes. También responde a la necesidad de aquellos curiosos que necesitan resolver ciertas cuestiones que pueden no ser tan accesibles en el día a día.

Para maximizar el alcance y la efectividad de estos videojuegos educativos, el marketing digital se presenta como una herramienta complementaria ideal. Las estrategias de marketing digital permiten segmentar audiencias específicas, garantizando que los juegos lleguen a aquellos que más pueden beneficiarse de ellos. Además, las campañas en redes sociales y el contenido viral pueden amplificar la visibilidad y el impacto de estos juegos, asegurando que los conocimientos que imparten se distribuyan de manera amplia y efectiva.

Incorporar estos elementos en el desarrollo de nuevos juegos educativos puede asegurar no solo el éxito del producto, sino también una contribución significativa a la educación y al avance del conocimiento en áreas cruciales como la biomedicina.

1.1.1 Justificación personal

La curiosidad de aprender nuevos conceptos de biomedicina siempre ha estado presente en mí. Aunque no deseo dedicarme profesionalmente a este campo, sí quiero seguir aprendiendo sobre él y combinarlo con otros intereses. En las redes sociales, ciertos divulgadores científicos me han inspirado a buscar soluciones a mis propias inquietudes. Por ello, veo en este Trabajo de Fin de Grado una oportunidad para aprovechar los conocimientos adquiridos en mi grado de Tecnología Digital y Multimedia.

La elección de desarrollar un videojuego se basa en mi percepción de que es una excelente propuesta para llegar a todo tipo de públicos. Personalmente, ha sido lo que más me ha motivado durante mi grado, me permite explotar mi parte creativa y es el área en la que quiero enfocar mi futuro por el momento. Concretamente, en la asignatura de Desarrollo de Videojuegos, me surgieron ganas de explorar los juegos de plataformas para la enseñanza, y en diversos proyectos realizados en asignaturas de marketing y plataformas de streaming, ya contemplaba el concepto de enseñanza en el mundo de la biomedicina.

Además, si considero fundamental incorporar la dimensión del marketing digital en los videojuegos es porque permite aumentar su visibilidad y alcance, llegar a un público más amplio y diverso, y generar una mayor interacción con los usuarios. De esta manera, se potencia tanto el impacto educativo del juego como su éxito comercial.

1.2 Objetivos

Este Trabajo Final de Grado tiene como objetivo principal plasmar de manera exhaustiva el desarrollo de un videojuego 2D mediante la herramienta de Unity. A lo largo del documento se presentarán y detallarán todas las fases fundamentales del proceso creativo y técnico, desde la conceptualización inicial hasta la culminación del proyecto con el producto final.

En el ámbito artístico, se profundizará en la creación de personajes y el diseño de escenario. Se describirán las técnicas y herramientas utilizadas para dar vida a los personajes, el mundo en el que se desarrolla la historia, así como todos los menús necesarios para navegar por el propio juego.

En cuanto al desarrollo técnico, se analizarán las distintas etapas de programación y desarrollo dentro de Unity, explicando cómo se implementaron las mecánicas de juego y las funcionalidades esenciales. Se prestará especial atención al desarrollo de los personajes y la metodología de enseñanza que se planteará a través de la jugabilidad.

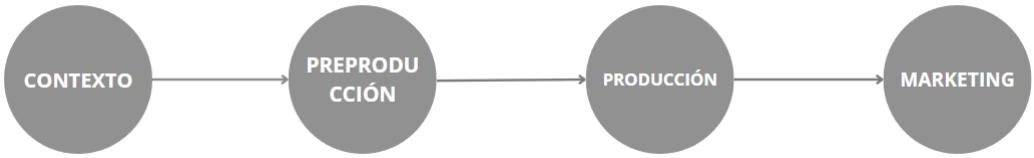
Además, se incluirán ejemplos prácticos, diagramas y capturas de pantalla que ilustrarán cada paso del proceso, proporcionando una visión clara y detallada del trabajo realizado. Este trabajo no solo pretende mostrar el producto final, sino también ofrecer una guía comprensiva sobre el desarrollo de videojuegos en 2D, destacando los desafíos enfrentados y las soluciones adoptadas a lo largo del proyecto.

Por último, referente a la parte de marketing se llevará a cabo lo siguiente:

- Un *user testing* para la detección de posibles problemas, la mejora de la funcionalidad del videojuego y el análisis de la experiencia de usuario.
- Utilización de la plataforma Ubersuggest para realizar un *keyword hunting* que identifique términos de búsqueda relevantes y analice la visibilidad del videojuego en motores de búsqueda y plataformas digitales.
- El uso de las redes sociales como ejemplo de la promoción del producto.

1.3 Metodología

El propósito de este Trabajo Final de grado seguirá una metodología que se divide en diversas etapas en base a los objetivos comentados anteriormente.



Cuadro 1. Metodología

Fuente: Elaboración propia.

1. Antes de introducirnos a lo que es el desarrollo de un videojuego se dará un contexto básico de los videojuegos y su relación con la enseñanza. Por otra parte, se aportará un análisis DAFO (Debilidades, Amenazas, Fortalezas y Oportunidades) que dará una comprensión clara de la situación actual de los videojuegos didácticos y de plataformas.
2. La preproducción resolverá las dudas relacionadas con el enfoque que se pretende dar con el desarrollo del proyecto, los múltiples propósitos e ideas y en definitiva definir el concepto y la visión general del videojuego.
3. Seguidamente, en este documento se explicará de manera detallada todos los procedimientos tanto técnicos como artísticos que lleven a dar el resultado final del videojuego.
4. Por último, se llevarán a cabo diversos recursos de marketing digital que completarán el propósito de este trabajo.

1.4 Relación con el grado en Tecnología Digital y Multimedia

Es de vital importancia relacionar este proyecto con lo aprendido en el grado, ya que permite demostrar la aplicación práctica de los conocimientos adquiridos a lo largo de los estudios.

A continuación, se detallan las principales áreas del grado que se conectan directamente con el desarrollo de este proyecto:

- **Área tecnológica**: Toda la base y conocimiento de programación adquirido para el desarrollo del videojuego (en este caso en C#) se ha estudiado en las asignaturas Desarrollo de Videojuegos y Programación.

- **Área artística**: En este proyecto se demuestran los conocimientos artísticos provenientes de la asignatura Diseño Gráfico en todo lo relacionado con los diseños de personajes. Talleres y Seminarios Emergentes II en lo que es la creación de una historia y un desarrollo de personajes. Edición y Postproducción Audiovisual para la animación de los personajes, pero sobre todo aquella asignatura que cierra el círculo artístico para darle un sentido al videojuego es Desarrollo de Videojuegos.

- **Marketing Digital**: Sin lugar a duda la asignatura de Modelos de Negocio queda plasmada al mostrar los recursos de Marketing Digital empleados en este trabajo. Organización Digital y Dirección de Proyectos son las asignaturas predecesoras a estas que también han sido fundamentales para esta área.

Cuadro 2. Relación del TFG con el grado.

Fuente: Elaboración propia.

Capítulo 2. Los juegos de plataformas y su enfoque didáctico

Antes de adentrarse en el desarrollo de un videojuego didáctico, es crucial entender el contexto en el que se sitúan tanto los juegos de plataformas como el enfoque educativo que algunos videojuegos han ofrecido a lo largo del tiempo. Esta perspectiva permitirá apreciar la evolución de estos juegos y su impacto en la educación. Para ello, se mostrarán una serie de ejemplos de videojuegos que han logrado combinar exitosamente el entretenimiento con la educación, proporcionando un marco de referencia para el desarrollo de nuestro propio juego.

Además, se llevará a cabo un análisis DAFO (Debilidades, Amenazas, Fortalezas y Oportunidades) para obtener una comprensión clara de la situación actual de los videojuegos didácticos y de plataformas, identificando factores internos y externos que pueden influir en nuestro proyecto.

2.1 La intersección de Juegos de Plataformas y Educación

Los juegos de plataformas son un subgénero de videojuegos que se centra en la navegación y la superación de obstáculos en niveles o escenarios verticales y horizontales. En estos juegos, el jugador controla a un personaje que debe saltar y moverse a través de plataformas, evitando obstáculos y enemigos mientras recolecta objetos o cumple objetivos.

Títulos icónicos como "Super Mario Bros." establecieron las bases del género con mecánicas de juego centradas en la acción y la exploración.

La relación entre videojuegos y educación ha crecido y cambiado mucho a lo largo de los años. En los inicios de los videojuegos, especialmente en las décadas de 1970 y 1980, los juegos estaban principalmente diseñados para entretener. Sin embargo, no pasó mucho tiempo antes de que los desarrolladores y educadores se dieran cuenta de que los videojuegos también podían ser herramientas útiles para enseñar y aprender.

A medida que la tecnología ha avanzado, también lo ha hecho la capacidad de los videojuegos para enseñar. Hoy en día, vemos cómo los juegos de plataformas y otros géneros se integran en las aulas y programas educativos, ayudando a los estudiantes a aprender de una manera interactiva y atractiva.

En resumen, lo que comenzó como simple entretenimiento ha evolucionado hasta convertirse en una herramienta poderosa para la educación. Con la tecnología avanzando continuamente, es probable que los videojuegos sigan desempeñando un papel importante en la educación del futuro.

2.1.1 Ejemplos

Una buena manera de refutar lo dicho es mostrando algunos ejemplos de aquellos videojuegos que hay en el mercado que han sido capaces de combinar didactismo con entretenimiento:

2.1.1.1 Zoombinis

Un juego de lógica y resolución de problemas en el que los jugadores guían a unas criaturas llamadas Zoombinis a través de una serie de desafíos.

Desarrolla habilidades de pensamiento crítico, lógica y matemáticas.

Fecha de lanzamiento: 1996

Cuadro 3. Captura del videojuego Zoombini.

Fuente: Steam [3]

2.1.1.2 JumpStart Adventures

Una serie de juegos educativos que combinan plataformas con actividades de aprendizaje en diversas áreas, como matemáticas, ciencia y lectura.

Fomenta el aprendizaje en una variedad de materias a través de un juego interactivo.

Fecha de lanzamiento: 1994

Cuadro 4. Captura del videojuego JumpStart Adventures.

Fuente: JumpStart Adventures Fandom [4]

2.1.1.3 Plague Inc.

Un simulador de pandemia donde los jugadores crean y evolucionan un patógeno para infectar y acabar con la población mundial.

Enseña sobre la propagación de enfermedades, la importancia de la salud pública y los esfuerzos necesarios para contener pandemias.

Fecha de lanzamiento: 2012

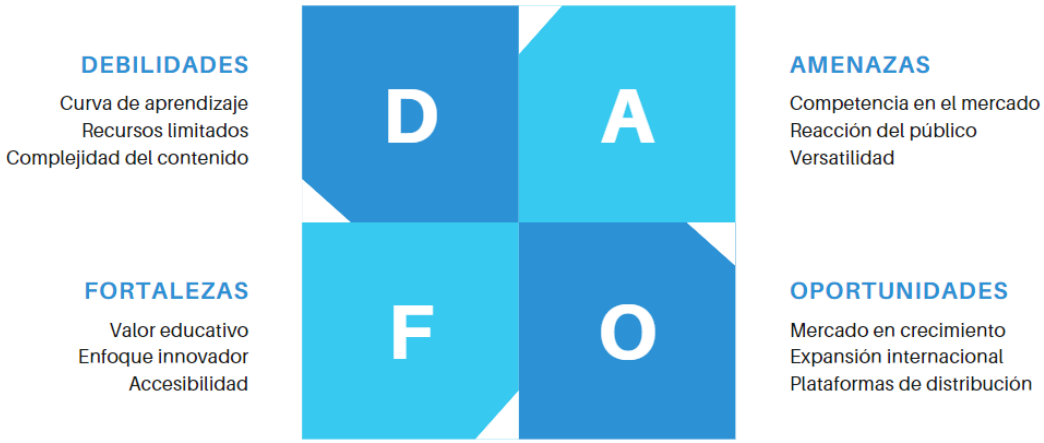
Cuadro 5. Captura del videojuego Plague Inc.

Fuente: Steam [5]

Estos ejemplos representan solo una pequeña fracción de la vasta cantidad de videojuegos diseñados para ofrecer conocimientos, ya sea en el ámbito de la biomedicina o en otras disciplinas. La intersección de la educación y los videojuegos ha creado un panorama dinámico y en constante crecimiento.

2.2 Análisis DAFO

Como bien se ha comentado anteriormente, el análisis DAFO (Debilidades, Amenazas, Fortalezas, Oportunidades) es de vital importancia, sobre todo a la hora de sacar al mercado un videojuego porque permite evaluar y entender claramente la posición estratégica del producto.



Cuadro 6. Análisis DAFO.

Fuente: Elaboración propia.

- **DEBILIDADES**: Los jugadores pueden encontrar difícil equilibrar la diversión del juego con la comprensión de conceptos biomédicos avanzados. También puede ser costoso crear contenido educativo profundo y atractivo. Además, teniendo en cuenta que la

biomedicina es un campo complejo que podría dificultar la creación de niveles y desafíos que sean a la vez educativos y entretenidos.

- **AMENAZAS**: Presencia de otros juegos educativos y de entretenimiento que pueden captar la atención de los consumidores. Posible resistencia de los jugadores que prefieren juegos más tradicionales y menos educativos.

- **FORTALEZAS**: Combina entretenimiento con educación, lo que puede atraer tanto a jugadores como a educadores. Uso de una temática biomédica en un formato de juego de plataformas 2D, lo que es relativamente único y puede captar la atención del mercado. Los juegos 2D suelen ser más accesibles y menos exigentes en términos de hardware, permitiendo un alcance más amplio.

- **OPORTUNIDADES**: Aumento de la demanda de juegos educativos y de plataformas de aprendizaje interactivo. Potencial para adaptar el juego a diferentes idiomas y mercados globales. Uso de plataformas digitales como *Steam, App Store y Google Play* para una amplia distribución y alcance.

Capítulo 3. Preproducción de un videojuego 2D

La preproducción e ideación son fases críticas en el desarrollo de un videojuego, sentando las bases para su éxito final. Estas etapas previas son fundamentales porque permiten definir y refinar la visión del juego antes de que comience la producción en sí.

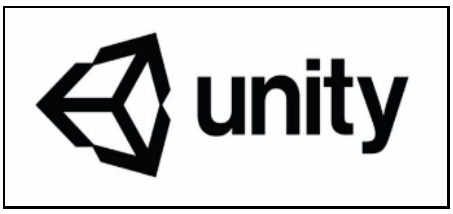
Durante la preproducción, se abordan aspectos importantes como la conceptualización del juego, la creación del guion y la narrativa, el diseño de personajes y escenarios, así como la definición de las mecánicas y la jugabilidad.

3.1 Herramientas de desarrollo

Un aspecto crucial en estas etapas iniciales es la elección y familiarización con las herramientas de desarrollo que se utilizarán a lo largo del proyecto. Es esencial identificar y dominar el software y las tecnologías que facilitarán la creación del juego.

3.1.1 Unity

Unity es uno de los motores de desarrollo de videojuegos más populares y versátiles del mercado. Se pueden utilizar varias herramientas de desarrollo para crear un videojuego, sin embargo, yo me he decantado por Unity por diversas razones.



Cuadro 7. Logo de Unity.

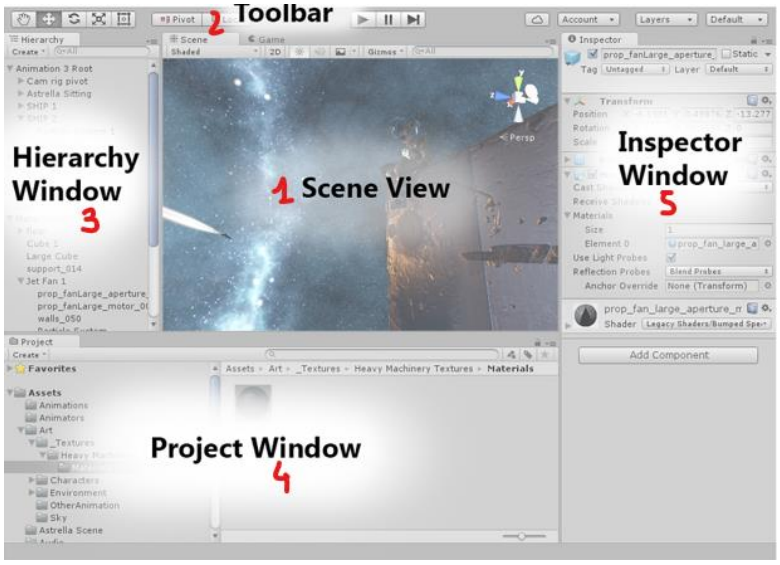
Fuente: Wikimedia Commons [6]

La principal razón es que Unity proporciona una interfaz intuitiva y accesible para desarrolladores de todos los niveles, desde principiantes hasta expertos. Su sistema de arrastrar y soltar simplifica el diseño e integración de los elementos del juego, agilizando considerablemente el proceso de desarrollo.

Los elementos que se pueden encontrar en esta aplicación son los siguientes:

1. **Vista de Escena**: Aquí puedes ver y editar los elementos en tu escena. Puedes mover, rotar y escalar objetos, ajustar la cámara y navegar por el entorno de juego. Es una representación visual de todo lo que está en la escena actual.
2. **Barra de Herramientas**: Contiene herramientas para manipular objetos en la Vista de Escena, opciones de reproducción y pausa del juego, y accesos a las funciones de Unity como guardar el proyecto, cambiar la configuración de la escena, entre otros.
3. **Jerarquía**: Muestra una lista jerárquica de todos los objetos en la escena actual. Aquí puedes seleccionar, organizar, y agrupar *GameObjects* (objetos del juego). Es donde administras la estructura de tu escena, con cada objeto representado como un nodo en la jerarquía.
4. **Ventana del proyecto**: Esta ventana contiene todos los archivos y recursos que forman parte del proyecto, organizados en carpetas.
5. **Inspector**: Muestra las propiedades del objeto seleccionado. Aquí puedes modificar las características de los componentes de los *GameObjects*, como sus transformaciones,

scripts asociados, y otros componentes. Es la principal herramienta para ajustar y personalizar los comportamientos y características de los objetos.



Cuadro 8. Logo de Unity.

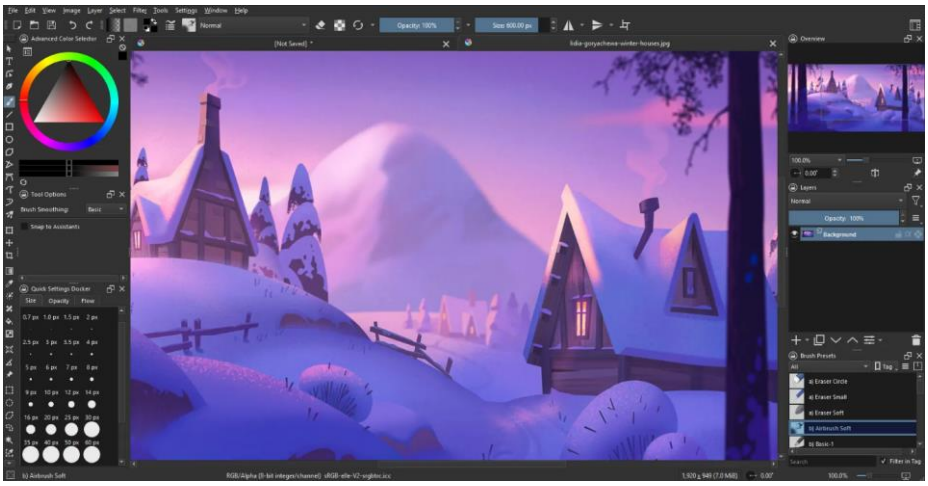
Fuente: Unity Documentation [7]

Y eso se añade con que su sistema de scripting, basado en C# es robusto y permite una gran flexibilidad en la creación de funcionalidades personalizadas.

Por último, si debo dar una justificación personal, como bien se ha comentado anteriormente, en el grado se ha impartido una asignatura específica para todo lo que es el desarrollo de un videojuego en el que se ha hecho uso de esta herramienta. Por lo tanto, es lógico entender que aquello que se está aplicando tiene mucho que ver con las técnicas enseñadas en dicha asignatura.

3.1.2 Krita

Krita ha sido una pieza fundamental en todo lo relacionado con la parte artística de este proyecto. Por experiencia propia, esta herramienta siempre ha demostrado ser útil para el diseño de personajes y otros elementos visuales del juego.



Cuadro 9. Logo de Unity.

Fuente: NOTILINUX [8]

Dado que el videojuego es en 2D, *Krita* se adapta perfectamente a nuestras necesidades, ofreciendo una amplia gama de funciones y capacidades que facilitan el proceso creativo. El uso de los pinceles, diversas herramientas y toda su interfaz intuitiva son suficientes motivos para escoger esta aplicación.

Al principio del desarrollo también se planteó usar *Gimp*, ya que fue la herramienta protagonista de la asignatura *Diseño Gráfico*. Sin embargo, a gusto personal disfruto más trabajando con *Krita*. Su flujo de trabajo se siente más natural y eficiente para mí, lo que ha permitido que el proceso de diseño sea más fluido y agradable.

3.2 Historia

En todos los videojuegos, una buena historia es fundamental para que el jugador se sumerja por completo en la experiencia. Permite a los jugadores conectarse emocionalmente con los personajes y el mundo del juego, haciendo que la experiencia sea mucho más memorable y significativa.

En este caso partimos de la base de que el juego pretende enseñar biomedicina. Intenté para ello visualizar una historia futurista en el que se consiguiera diseñar robots capaces de reducir su tamaño hasta tal punto en el que pudiesen llegar hasta las células afectadas por enfermedades y combatirlas para el bien del organismo. Esto proporciona al videojuego un contexto que justifica el que puedan hacerse preguntas de biomedicina y así también conocer a los personajes principales.

En el juego nos pondremos en la piel de Nano, un robot que está en sus primeras clases de *Introducción a la célula y mantenimiento*. Este personaje deberá combatir los males que acechan la célula para sanarla. Pero para conseguir superar su misión, deberá contestar preguntas de biomedicina, escogiendo el camino correcto en su aventura.

En su camino, Nano estará acompañado por Proteína, una proteína inteligente y simpática que lo guiará y asistirá a lo largo de su misión. Proteína proporcionará valiosos consejos y explicaciones sobre los desafíos que enfrentan.

3.3 Personajes

Una vez pensada la historia del juego, el siguiente paso era diseñar unos personajes que dieran sentido y vida a esta narrativa. Los personajes no solo debían ser visualmente atractivos, sino que también necesitaban tener personalidades y roles que enriquecieran la experiencia del jugador y facilitaran la transmisión del contenido educativo.

Los personajes ideados son los siguientes:

- **Nano-BOT**: el protagonista de esta historia, un robot del futuro diseñado para adentrarse en las células. Creado con la más avanzada nanotecnología, Nano tiene la capacidad de explorar y reparar estructuras celulares a niveles inimaginables para la medicina contemporánea. Es un ser entusiasta, lleno de curiosidad y determinación, siempre dispuesto a enfrentar nuevos desafíos y superar cualquier obstáculo. Su objetivo es claro: convertirse en el mejor robot biomédico del mundo.

- **Proteína**: como su nombre indica, es una proteína de la célula a la que Nano ha llegado. Su diseño refleja las largas estructuras enredantes y complejas, características de muchas proteínas. Las proteínas son fundamentales en el organismo, desempeñando roles cruciales como catalizar reacciones bioquímicas, estructurar tejidos, transportar moléculas y regular procesos celulares. Sin embargo, en esta aventura, la proteína le advertirá a Nano del peligro que está ocurriendo y le aconsejará y guiará en su camino.

- **Bacteria**: una de las antagonistas de esta aventura, peligrosas criaturas que reflejan, como su nombre indica, las bacterias. Las bacterias son microorganismos unicelulares que

pueden causar infecciones y enfermedades en los organismos multicelulares, incluidos los humanos. Estas entidades hostiles en el juego representan las amenazas bacterianas que Nano debe enfrentar y superar.

- **Virus:** otro enemigo al que Nano tendrá que hacer frente. Esta criatura está diseñada pensando en el diseño del coronavirus, con su distintiva forma esférica y sus espigas que sobresalen de la superficie. El virus tiene la habilidad de escupir toxinas que causan daño a Nano, dificultando su misión.

3.4 Controles y Mecánicas

Después de conseguir visualizar cuáles van a ser los personajes de nuestro videojuego, es importante determinar las mecánicas que definirán cómo los jugadores interactuarán con ellos y con el entorno. Los controles y las mecánicas son el núcleo de la experiencia de juego, ya que establecen las reglas y formas en que los jugadores pueden moverse, luchar, explorar y resolver desafíos dentro del mundo del juego.

- Empezaremos con lo que serían los controles del personaje:

Tecla pulsada	Función
A, D / Flecha izquierda, Flecha derecha	Correr en dos direcciones horizontalmente (izquierda, derecha)
Espacio / W / Flecha superior	Saltar
S / Flecha inferior	Agacharse
E	Interactuar con los personajes

Tabla 1. Controles del Personaje Principal.

Fuente: Elaboración Propia.

Se da la posibilidad de jugar tanto con WASD como con las flechas del teclado. Se puede observar que no son mecánicas complejas, pero se pretende así aportar comodidad a la hora de jugar sin dificultad, para todos los públicos.

- Sistema de barra de salud. El personaje principal podrá recibir daño y morir.

La idea es conseguir visualizar una barra coloreada la cual equivale a 100 puntos, estos puntos bajarán debido al daño de los enemigos.

- Las bacterias infligirán 10 de daño al personaje principal si lo tocan. Estos tendrán movimientos horizontales aleatorios.
- Los virus permanecerán estáticos y dispararán proyectiles que infligirán 20 de daño a nuestro personaje.
- Plataformas con movimiento para complicar el paso del personaje.
- Recolección de antibióticos y conteo de estos.
- Sistema de diálogo con la Proteína.

3.5 Metodología de aprendizaje

A lo largo de los años, el sistema académico en España nos ha llevado a memorizar largos contenidos de diversas materias. Con este proyecto, pretendía ofrecer una alternativa más entretenida y efectiva, creando una metodología de aprendizaje basada en lo que comúnmente conocemos como tipo test.

- **Interacción y Desafíos:** El jugador se enfrentará a diversas adversidades típicas de los juegos de plataformas, como saltar, esquivar enemigos y superar obstáculos. Además, se integrarán elementos educativos en la jugabilidad.
- **Preguntas y Caminos:** La proteína que acompaña a nuestro protagonista Nano no solo actúa como guía, sino que también plantea preguntas básicas de biomedicina. Estas preguntas aparecen en momentos clave del juego, y las respuestas determinan los caminos que el jugador puede tomar.
- **Caminos Correctos:** De las múltiples opciones de respuesta, solo una será correcta. Elegir la respuesta correcta permitirá al jugador avanzar en el juego, mientras que elegir incorrectamente presentará obstáculos adicionales hasta la muerte, fomentando así el aprendizaje a través del ensayo y error.

3.6 Dificultad y sistema de recompensas

Es necesario aclarar que la dificultad no se basa en niveles ya que se pretende recoger los contenidos en un solo nivel continuo. Este enfoque permite una experiencia de aprendizaje sin interrupciones, donde el jugador enfrenta un flujo constante de preguntas y desafíos mientras avanza en el juego.

Sin embargo, se plantea previamente desarrollar un sistema de recolección de recompensas para incentivar la participación y el progreso dentro del juego. Así se consigue no solo mantener el interés y la motivación del jugador, sino también reforzar el aprendizaje mediante una mecánica de juego que ofrece recompensas tangibles por el esfuerzo y la habilidad demostrados.

Este enfoque asegura una experiencia de juego más envolvente y gratificante, lo que facilita la retención de conocimientos y el compromiso continuo con el contenido educativo.

3.7 Mapa de color

La paleta de colores en un videojuego no solo define su apariencia visual, sino que también influye significativamente en la experiencia del jugador. La selección cuidadosa de los tonos puede establecer el tono emocional del juego, mejorar la legibilidad y guiar la atención del usuario hacia elementos clave.

En el desarrollo de un videojuego didáctico, la elección de los colores no se basa únicamente en preferencias estéticas, sino también en su capacidad para facilitar la comprensión del contenido educativo. Los colores deben ser elegidos para destacar información importante, mantener el interés y crear un entorno que refuerce el aprendizaje.

Los colores están clasificados según los requerimientos del propio videojuego:

Cuadro 10. Mapa de color.

Fuente: Elaboración Propia.

Capítulo 4. Producción de un videojuego 2D en Unity

Una vez se han tenido en cuenta y agrupado todas las ideas iniciales del videojuego, en este capítulo se reflejará todo su desarrollo desde su parte creativa hasta la parte de programación, explicando el porqué de cada decisión tomada en el proceso.

4.1 Menú e interfaz

Un menú inicial bien diseñado es tan importante como el propio videojuego, ya que orienta y guía al jugador desde el primer momento. Este componente crucial no solo establece el tono del juego, sino que también proporciona a los jugadores una forma clara y accesible de navegar por él.

Cuando se pensó en el diseño de la pantalla principal, se pretendía que los botones estuviesen postrados encima de una libreta. Los botones y otros elementos interactivos se integrarían como partes de esta libreta, contribuyendo a una estética llamativa.

Cuadro 11. Menú inicial.

Fuente: Elaboración Propia.

Por otra parte, el fondo tiene elementos típicos de la biomedicina, que es la temática principal. Se añadió cadenas de ADN, neuronas, bacterias, virus, y el personaje principal, entre otros (todo lo que es la parte artística ha sido cosecha propia, dibujada a mano con la tableta gráfica en *Krita*)

Estos elementos en Unity son sprites/png's que contienen dos scripts diferentes para conseguir movimiento y dinamismo.

Los códigos son los siguientes:

- **Script para simular movimientos respiratorios:** Este script se llama *BreathingMovement.cs* ya que pretende simular movimientos de contracción y expansión similares a la respiración. En él se han generado unas variables públicas que se muestran posteriormente en Unity que nos permiten modificar la velocidad y la amplitud de contracción. Con una ecuación sencilla conseguimos este efecto.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class BreathingMovement : MonoBehaviour
{
    public float speed = 1f; // Velocidad del movimiento de respiración
    public float amplitude = 0.1f; // Amplitud del movimiento de respiración

    private Vector3 startPosition;

    void Start()
    {
        startPosition = transform.position; // Guardar la posición inicial
    }

    void Update()
    {
        // Movimiento suave de expansión y contracción similar a la respiración
        transform.position = startPosition + Vector3.up * Mathf.Sin(Time.time * speed) * amplitude;
    }
}
```

Cuadro 12. BreathingMovement.cs.

Fuente: Elaboración Propia.

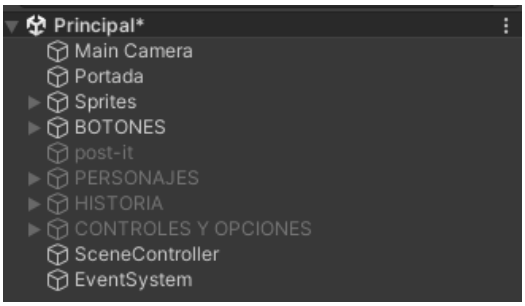
- **Script para simular rotación constante**: Al igual que el anterior script, con una simple ecuación, todos los .png con este código rotaran con una velocidad determinada. Por defecto está a 100 la velocidad, pero en Unity se puede modificar.

```
public class ConstantRotation : MonoBehaviour
{
    public float rotationSpeed = 100f;

    void Update()
    {
        transform.Rotate(Vector3.forward * rotationSpeed * Time.deltaTime);
    }
}
```

Cuadro 13. ConstantRotation.cs.

Fuente: Elaboración Propia.

Después se ha jerarquizado cada una de las vistas y botones en Unity de la siguiente manera para un correcto funcionamiento:



Cuadro 14. Jerarquía del menú principal.

Fuente: Elaboración Propia.

El elemento *SceneController* es aquel elemento vacío que contiene el script que permite hacer el cambio de escenas dependiendo de los botones que se aprieten.

4.2 Menús

Una vez se ha hablado de la vista principal y su diseño, ahora se indagará en cada uno de los menús a los que podrá acceder el usuario con cada uno de los botones.

4.2.1 Menú historia

Para sumergir a los jugadores en el contexto y la trama del juego de manera efectiva, aproveché el diseño de una libreta como menú principal, la cual me permitió simular que la historia estaba escrita dentro del libro de la siguiente manera:

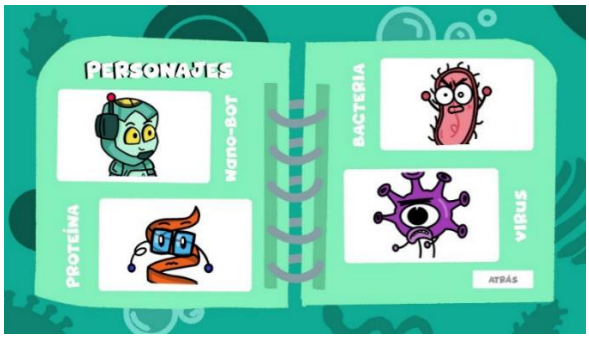
Cuadro 15. Menú historia.

Fuente: Elaboración Propia.

Puede que sea sencillo, pero a la vez es conciso y conveniente para los usuarios. También observaremos en esta figura que, tanto en este menú como en los otros, se reutiliza un fondo con diversos elementos que le da personalidad a la pantalla.

4.2.2 Menú Personajes

Para conseguir que el usuario entienda quiénes son los personajes involucrados en la historia, decidí crear adicionalmente un menú dedicado exclusivamente a los personajes. Este menú tiene como objetivo no solo presentar a los personajes, sino también explicar su importancia y lo que representan en el mundo de la biomedicina.



Cuadro 16. Menú Personajes.

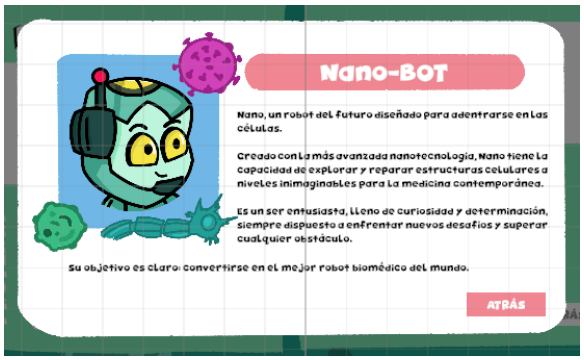
Fuente: Elaboración Propia.

En la primera pantalla, podemos observar que están cada uno de los personajes con su nombre al lado. Estas viñetas son, en realidad, botones interactivos que activan paneles explicativos sobre cada uno de ellos.

Al hacer clic en estos botones, se despliegan paneles detallados que explican qué son tanto los virus, como las proteínas, y las bacterias en el mundo de la biomedicina.

Los paneles son los siguientes:

- **Panel de Nano-BOT:** expone la historia del jugador y su propósito principal.

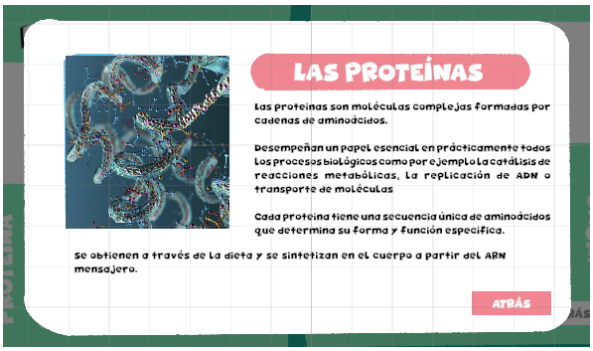


Cuadro 17. Panel informativo Nano-BOT.

Fuente: Elaboración Propia.

Al ser este un personaje ficticio, sencillamente se ha contado la historia del personaje, a diferencia de los otros personajes que tienen cartas descriptivas para así participar en el didactismo de este proyecto.

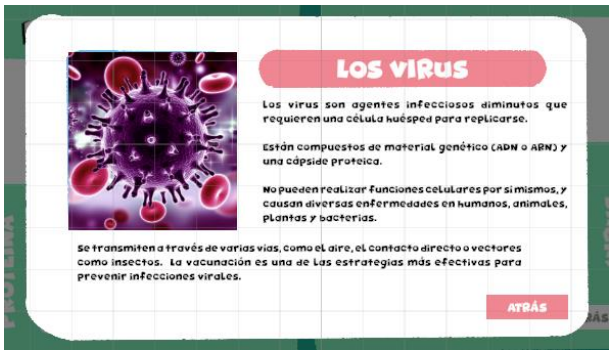
- **Panel de Proteína:** define lo que es una proteína, el papel que desempeña y sus características.



Cuadro 18. Panel informativo Proteínas.

Fuente: Elaboración Propia.

- **Panel de Virus:** define lo que es un virus, sus características y su método de transmisión.



Cuadro 19. Panel informativo Virus.

Fuente: Elaboración Propia.

- **Panel de Bacteria:** define lo que es una bacteria y sus características principales.

Cuadro 20. Panel informativo Bacterias.

Fuente: Elaboración Propia.

4.2.3 Menú Opciones y Controles

Es imprescindible contar con un menú que explique los controles al jugador para que pueda desenvolverse con facilidad durante el juego. Por lo tanto, he incorporado un menú de controles que proporciona una guía clara sobre cómo manejar el juego. Este menú detalla las acciones básicas que el jugador debe conocer: Andar, Saltar y Agacharse.

Simultáneamente, he añadido opciones básicas y necesarias para ajustar la experiencia del juego según las preferencias del usuario. Estas opciones incluyen:

- **Control de Volumen**: Un Slider que permite ajustar el volumen general del juego, para que el jugador pueda personalizar la experiencia auditiva según sus preferencias.
- **Regulación de Brillo**: Un Slider que permite ajustar el brillo del juego, asegurando que los jugadores puedan adaptarlo a sus condiciones de iluminación y garantizar una visualización óptima.

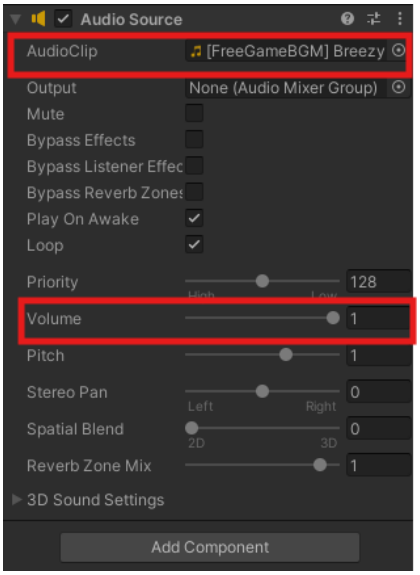
Cuadro 21. Panel informativo Bacterias.

Fuente: Elaboración Propia.

4.2.3.1 Regulador de Volumen

En la escena principal se ha añadido una música para darle ambientación y sumergir al jugador en el entorno del juego. En Unity los sonidos se introducen por lo general con un elemento que contenga una fuente de sonido.

La idea es que para controlar el volumen con la barra es relacionar con un código directamente el Slider con el parámetro que contiene esta fuente de sonido llamado *Volume*, el cual va de 0 a 1.



Cuadro 22. Fuente de Sonido en Unity.

Fuente: Elaboración Propia.

El código utilizado se llama *LogicaVolumen.cs*, el cual coge como parámetros la referencia del Slider, guarda el valor actual que refresenta el volumen y la imagen que se utilizará cuando el Volumen llegue a 0, ya que se pretende activar esta imagen al llegar a ese valor para avisar al jugador que no hay ningún sonido transmitiéndose.

```
public class LogicaVolumen : MonoBehaviour
{
    public Slider slider;
    public float sliderValue;
    public Image imagenMute;
    // Start is called before the first frame update
    void Start()
    {
        slider.value = PlayerPrefs.GetFloat("volumenAudio",0.5f);
        AudioListener.volume = slider.value;
        RevisarSiEstoyMuted();
    }

    public void ChangeSlider(float valor)
    {
        sliderValue = valor;
        PlayerPrefs.SetFloat("volumenAudio",sliderValue);
        AudioListener.volume = slider.value;
        RevisarSiEstoyMuted();
    }

    public void RevisarSiEstoyMuted()
    {
        if(sliderValue == 0)
        {
            imagenMute.enabled = true;
        }
        else
        {
            imagenMute.enabled = false;
        }
    }
}
```

Cuadro 23. Panel informativo Bacterias.

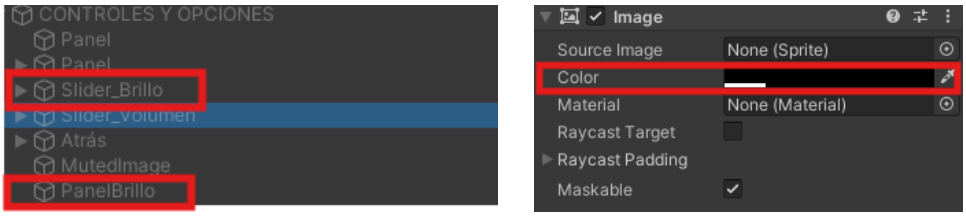
Fuente: Elaboración Propia.

Concretamente, en el **método Start()**, el cual se llama al inicio de la escena, recupera el valor del volumen guardado previamente. Si no se ha guardado ningún valor, se utiliza el valor predeterminado 0.5 sobre 1. Ajusta el volumen del sistema de audio al valor recuperado del Slider, y llama al método **RevisarSiEstoyMuted()**, el cual vemos al final que sirve para activar la imagen mencionada anteriormente.

Y por último el **método ChangeSlider(),** que actualiza como bien dice su nombre el valor del slider, lo guarda en las preferencias y ajusta el volumen del sistema de audio.

4.2.3.2 Regulador de Brillo

Para ajustar el brillo de la pantalla en el juego, se ha implementado una técnica que consiste en insertar un panel semitransparente en la vista principal, similar a otros elementos que actúan como fondos de escenas. Este panel cuenta con un canal alfa, el cual controla su nivel de transparencia. Al regular este canal alfa, podemos simular un ajuste de brillo en la pantalla.



Cuadro 24. Sliders y canal alfa para el brillo en Unity.

Fuente: Elaboración Propia.

El control de este ajuste se realiza mediante el Slider, que permite al usuario aumentar o disminuir la transparencia del panel. Al mover el Slider, se cambia el valor del canal alfa del panel, haciéndolo más transparente para aumentar el brillo o más opaco para reducirlo.

El código implementado para el Slider se denomina *LogicaBrillo.cs*. Las referencias o variables que requiere son:

- El Slider que controla el brillo
- El valor actual del Slider
- El panel

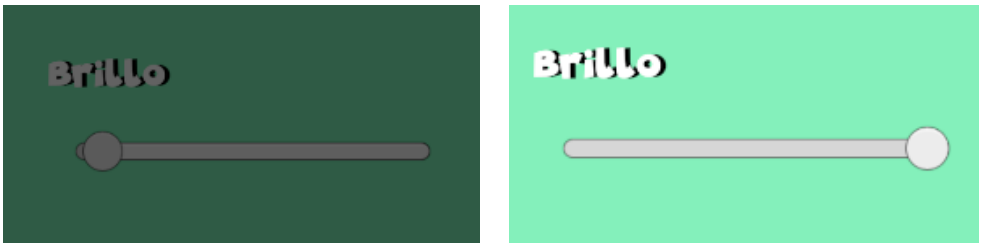
En este código, cuando se le llama al inicio del juego con el **método Start()**, recupera el valor de brillo guardado en las preferencias del usuario y ajusta la transparencia del panel con el valor del slider. Por otro lado, lo que es el **método** llamado **ChangeSlider()**, se llama cuando el usuario mueve el Slider, este actualiza su valor y se guarda en las preferencias. Por último, se cambia la transparencia del panel según el nuevo valor.

```
public class LogicaBrillo : MonoBehaviour
{
    public Slider slider;
    public float sliderValue;
    public Image panelBrillo;
    // Start is called before the first frame update
    void Start()
    {
        slider.value = PlayerPrefs.GetFloat("brillo", 0.5f);

        panelBrillo.color = new Color(panelBrillo.color.r,panelBrillo.color.g,panelBrillo.color.b,slider.value);
    }

    public void ChangeSlider(float valor)
    {
        sliderValue = valor;
        PlayerPrefs.SetFloat("brillo", sliderValue);
        panelBrillo.color = new Color(panelBrillo.color.r,panelBrillo.color.g,panelBrillo.color.b,slider.value);
    }
}
```

Cuadro 25. Panel informativo Bacterias.

Fuente: Elaboración Propia.

Dando como resultado lo siguiente:



Cuadro 26. Resultado del regulador de brillo.

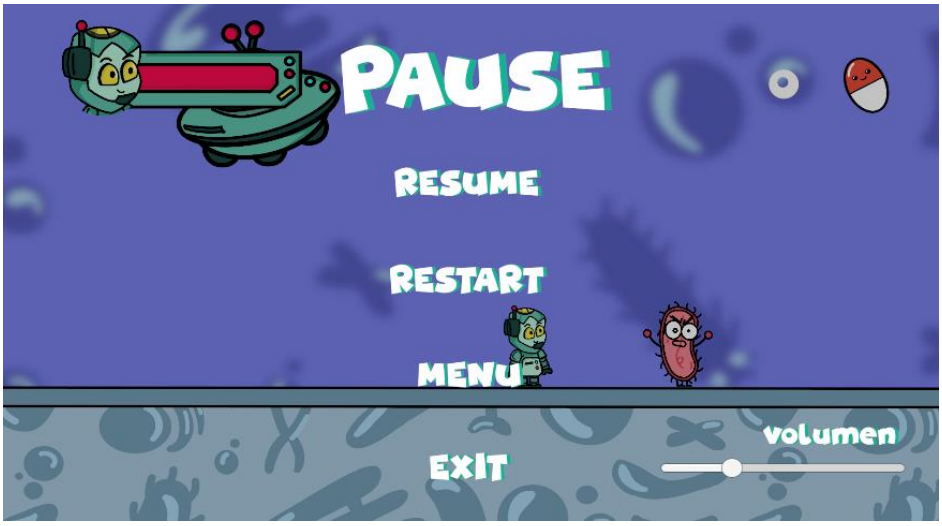
Fuente: Elaboración Propia.

4.2.4 Menú de Pausa

Para finalizar con los menús, a pesar de no estar presente en la pantalla de Inicio, he decidido plasmar el menú de Pausa que se activa con la tecla "**Esc**" cuando el usuario está jugando. Este menú permite al jugador tomar un descanso, acceder a opciones del juego o salir de la pantalla e incluso del juego.

Este menú ofrece las funcionalidades:

- **Resume:** Para seguir con el nivel
- **Restart:** Para reiniciar el nivel
- **Menu:** Para volver al menú principal
- **Exit:** Para salir del juego
- **Volumen:** Para controlar el volumen



Cuadro 27. Menú Pausa.

Fuente: Elaboración Propia.

4.3 Personaje principal

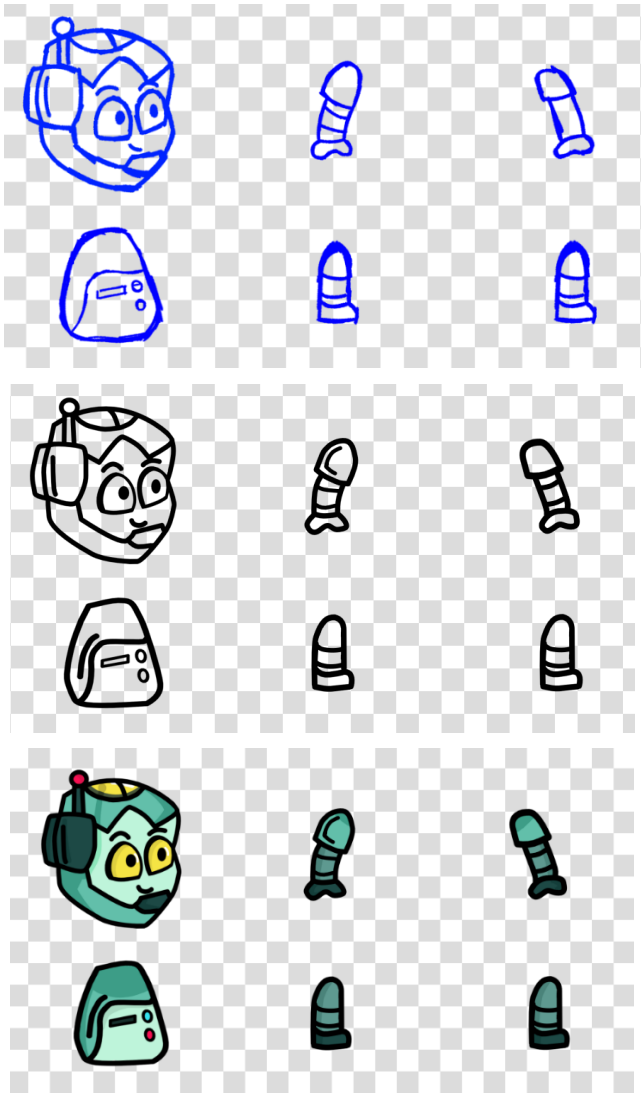
En todos los videojuegos de plataformas 2D, el diseño del personaje principal es de las partes más importantes a desarrollar. Un buen diseño de personaje debe ser visualmente atractivo, coherente con la temática del juego y capaz de transmitir emociones y personalidades a través de su apariencia y animaciones.

En esta memoria, se distingue el desarrollo de la parte artística y la programación para asegurar que cada proceso realizado quede bien documentado y comprendido.

4.3.1 Diseño

En general, cuando visualizaba este proyecto, pretendía mostrar un videojuego para todo tipo de edades. La manera más fácil de conseguir este acabado, bajo mi punto de vista, era aplicar en los diseños de los personajes matices infantiles, colores vivos y elementos entretenidos como se explica en el mapa de color.

Como se ha mencionado anteriormente, el diseño se ha realizado utilizando el programa *Krita* y una tableta gráfica.

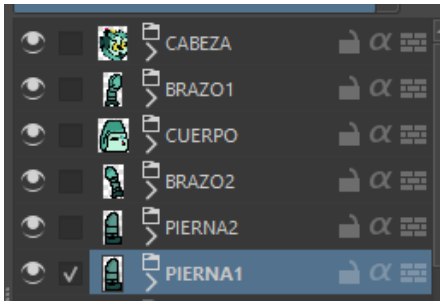


Cuadro 28. Sketching, inking y painting Personaje Principal.

Fuente: Elaboración Propia.

Se han seguido los pasos aprendidos en la asignatura de Diseño Gráfico en la que se explicaba que era fundamental antes de hacer el delineado y el acabado final hacer un boceto para tener estructurado todos los elementos del personaje.

Se puede observar en las capturas aportadas que las extremidades, cuerpo y cabeza están separadas entre sí. Esto se debe a que este proceso venía con la idea de posteriormente aportar un *rigging* para lo que es el movimiento del personaje en Unity (Esto se explicará adecuadamente en el punto 4.3.2).

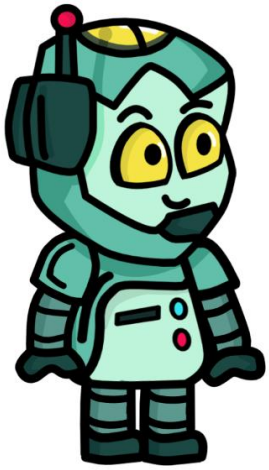


Cuadro 29. Segmentación Personaje Principal.

Fuente: Elaboración Propia.

Al estar cada una de las partes separadas, veremos que facilita mucho las complejidades que puedan aparecer para que los movimientos sean naturales.

El acabado final del personaje es el siguiente:



Cuadro 30. Diseño Final Personaje Principal.

Fuente: Elaboración Propia.

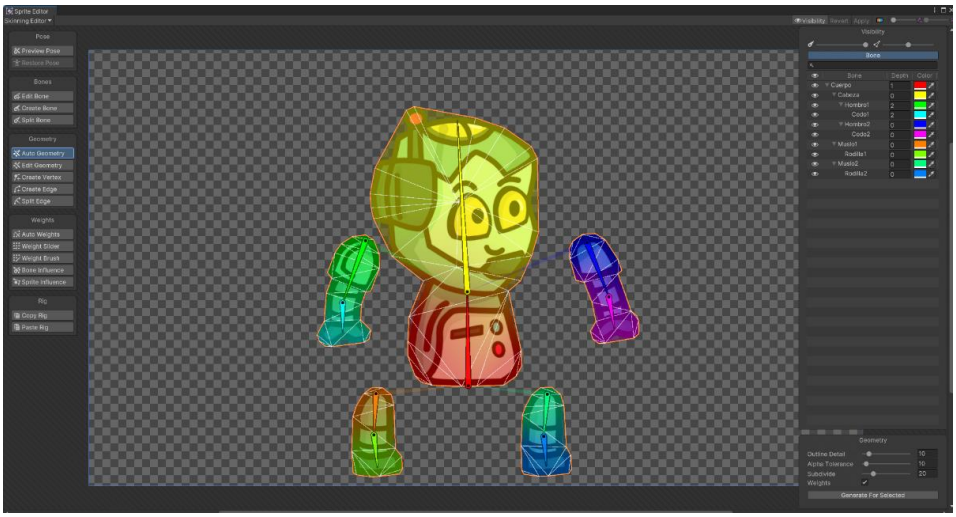
4.3.2 Rigging

El *rigging* se refiere a la creación de un sistema de esqueletos y controles para los personajes y objetos 2D, facilitando su animación de manera eficiente y realista.

En este caso la propia herramienta Unity proporciona a los usuarios una funcionalidad llamada *Sprite Editor*. En él nos permite añadirle los huesos necesarios al png de nuestro personaje para darle vida posteriormente en la animación.

Usé la herramienta que genera los huesos llamada *Create Bone*. Intenté simplificar cada uno de los huesos para que no fuesen demasiado complejos los movimientos. Añadí el tronco central que llega de pelvis a cuello, el cual hace de unión a las demás extremidades y cabeza.

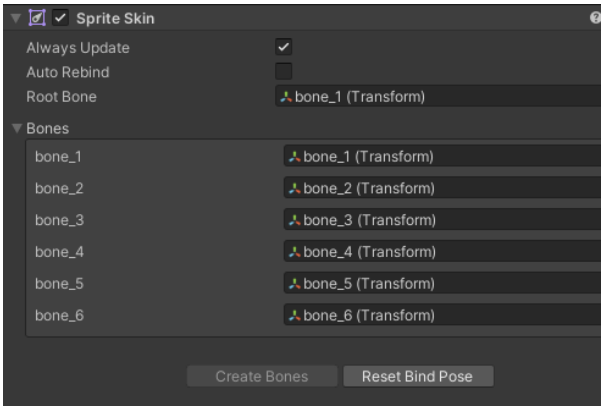
Gracias a que están separadas cada una de las partes, a la hora de ejecutar el proceso de unir los huesos con la imagen no hay ninguna interferencia en que detecte partes indeseadas.



Cuadro 31. Rigging Personaje Principal.

Fuente: Elaboración Propia.

Estos huesos se visualizarán cada vez que seleccionemos a nuestro protagonista. De hecho, en el inspector de Unity, en el apartado *Sprite Skin* nos aparecerá de la siguiente manera:



Cuadro 32. Huesos Personaje Principal.

Fuente: Elaboración Propia.

Como resultado final hemos conseguido un cuerpo con huesos que podremos modificar posteriormente a la hora de animar a nuestro personaje.

4.3.3 Animación

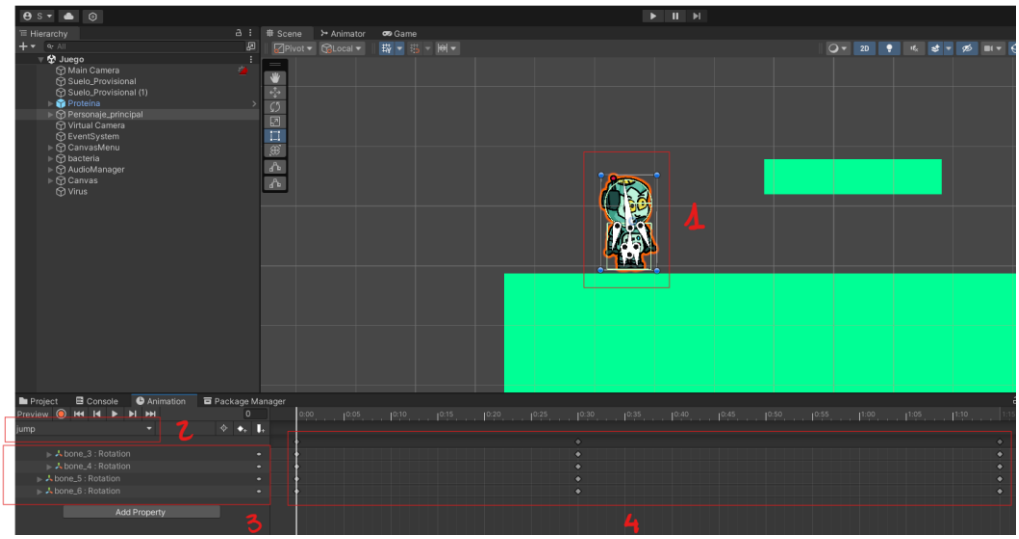
El siguiente paso para dar vida a Nano-BOT es diseñar las animaciones para los movimientos predefinidos, como saltar, caminar, agacharse, y la animación de espera cuando el personaje está quieto.

Escoger la herramienta de Unity una vez más es una excelente opción para este tipo de procedimientos, ya que viene con un sistema de animación integrado que facilita la creación y gestión de estas animaciones de manera eficiente.

Para ello, sencillamente se selecciona al personaje principal. Posteriormente se abre la ventana que nos ofrece Unity llamada *Animation*, la cual nos proporciona una línea de tiempo en la cual se puede grabar los movimientos colocando en cada instante de tiempo los huesos que deseamos que se muevan. Se puede comparar este proceso a la técnica *Stopmotion*, donde se usan objetos inanimados que son movidos por un animador en cada fotograma.

Si enumeramos junto a una imagen este proceso para aclarar cualquier duda sería de la siguiente manera:

1. Seleccionamos al personaje que lleva incorporado cada uno de los huesos y el cual moveremos en cada fotograma.
2. En esta ventana nos ofrece la posibilidad de crear las animaciones (recalco que estas animaciones quedan guardadas en nuestro proyecto)
3. Al presionar el botón de grabar, cada vez que movemos a nuestro personaje queda registrado los huesos que se están modificando.
4. Es la línea de tiempo, donde seleccionamos cuanta periodicidad habrá entre cambios de movimientos.



Cuadro 33. Proceso de animación Personaje Principal.

Fuente: Elaboración Propia.

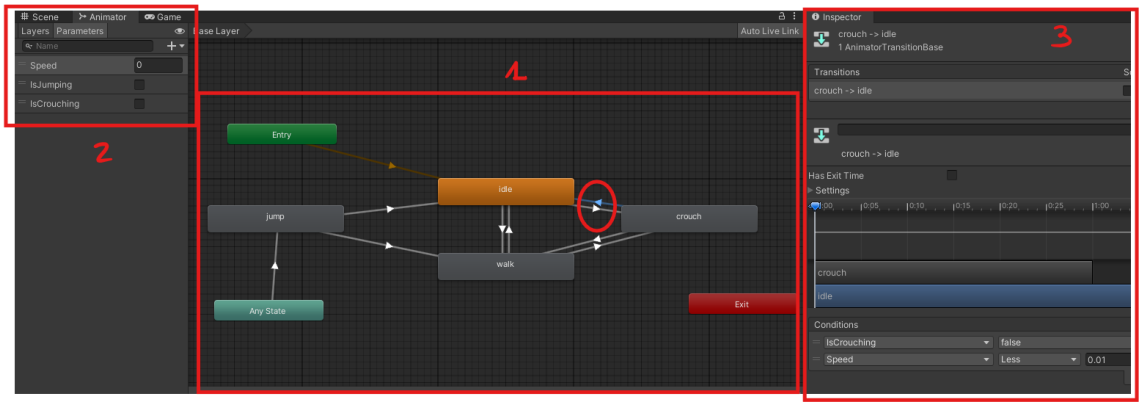
Este proceso se repite con cada una de las animaciones deseadas para que quede preparado para lo que es la programación posterior.

4.3.4 Programación

Las animaciones que tenemos guardadas no se llevarán a cabo hasta que programemos cuándo se deben ejecutar y en qué condiciones.

Estas condiciones se determinan en la ventana *Animator* (no confundir con la anterior que es *Animation*) de Unity. En esta ventana crearemos las transiciones entre los estados:

- Idle <-> Walk: Que de estar quieto pase a andar y viceversa.
- Any State -> Jump: Que de cualquier movimiento pase a saltar.
- Jump -> Idle: Que cuando termine la animación de saltar pase a estar quieto.
- Crouch<-> Idle: Que de estar quieto pase a agacharse y viceversa.
- Walk <-> Crouch: Que de estar agachado pase a andar y viceversa.



Cuadro 34. Transiciones Personaje Principal.

Fuente: Elaboración Propia.

En la figura X podemos observar 3 elementos clave en la vista:

1. Representa todas las transiciones que hemos generado y mencionado anteriormente.
2. Nos permite crear nuestras variables que servirán posteriormente cuando generemos el código y en la ventana número 3. Estas son *speed*, que modifica la velocidad a la que cambiaran las animaciones y dos booleanos llamados *IsJumping/IsCrouching* para detectar que está agachado/saltando o no.
3. En la ventana número 3 usamos las variables generadas para poner las condiciones.

Ahora bien, la parte fundamental de nuestro objetivo es la generación de un código que recoja estas condiciones, las lleve a cabo y se ejecute en nuestro personaje. Este código se llama *PlayerMovement.cs*.

```
public class PlayerMovement : MonoBehaviour
{

    public CharacterController2D controller;
    public Animator animator;

    public float runSpeed = 40f;

    float horizontalMove = 0f;
    bool jump = false;
    bool crouch = false;

    // Update is called once per frame
    void Update()
    {
        horizontalMove = Input.GetAxisRaw("Horizontal") * runSpeed;

        animator.SetFloat("Speed", Mathf.Abs(horizontalMove)); //Pa

        if (Input.GetButtonDown("Jump"))
        {
            jump = true;
            animator.SetBool("IsJumping",true);
        }

        if (Input.GetButtonDown("Crouch"))
        {
            crouch = true;
        } else if(Input.GetButtonUp("Crouch"))
        {
            crouch = false;
        }
    }

    public void OnLanding()
    {
        animator.SetBool("IsJumping", false);
    }

    public void OnCrouching (bool isCrouching)
    {
        animator.SetBool("IsCrouching",isCrouching);
    }
    void FixedUpdate ()
    {
        controller.Move(horizontalMove * Time.fixedDeltaTime, crouch, jump);
        jump = false;
    }
}
```

Cuadro 35. PlayerMovement.cs.

Fuente: Elaboración Propia.

Este código genera unas variables públicas para que en Unity se introduzca lo que son las condiciones anteriores y el personaje principal. Otros parámetros son la velocidad a la que queremos que se mueva el personaje e instanciar que inicialmente no está saltando ni agachándose.

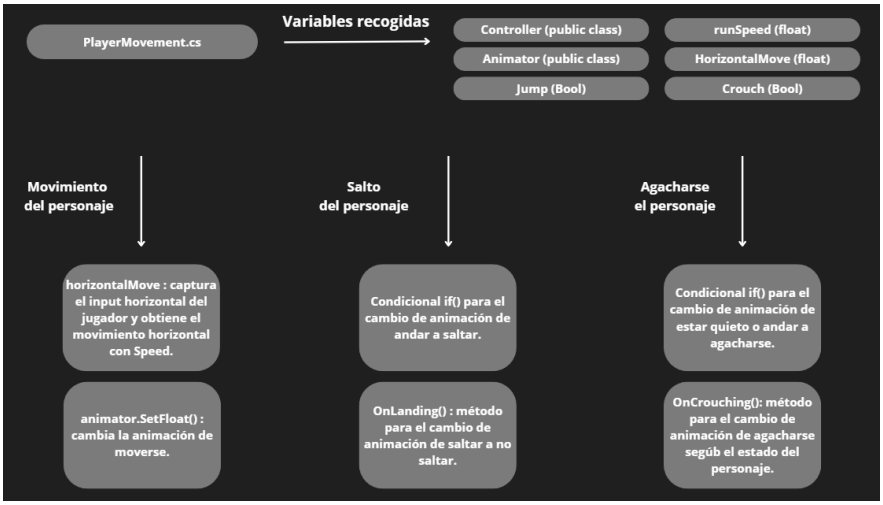
Seguidamente, captura el input horizontal del jugador (usando las teclas de flechas o 'A' y 'D') y lo multiplica por la velocidad para obtener el movimiento horizontal. Se ajusta el parámetro *Speed* en el Animator para cambiar la animación del personaje según su velocidad.

Por otro lado, se verifica si se presionó el botón de salto. Si es así, activa la variable *jump* y ajusta la animación de salto. Pasa igual con la variable para agacharse.

El método OnLanding() es llamado cuando el personaje aterriza después de un salto, desactivando la animación de salto. El método OnCrouching() ajusta la animación de agacharse según el estado del personaje.

Cuando comprobé todos estos procesos el personaje no hacía correctamente los movimientos cuando saltaba, es por ello que el método FixedUpdate() aplica las entradas del jugador al movimiento físico del personaje y asegura que el salto se realice solo una vez por entrada.

De todas formas, este código viene a continuación acompañado de un pseudocódigo que esquematiza correctamente los procesos tenidos en cuenta y aclare cualquier tipo de duda.



Cuadro 36. Pseudocódigo PlayerMovement.cs.

Fuente: Elaboración Propia

Y ya con todo esto se consigue tener un personaje que haga los movimientos que cabe esperar de un videojuego de plataformas.

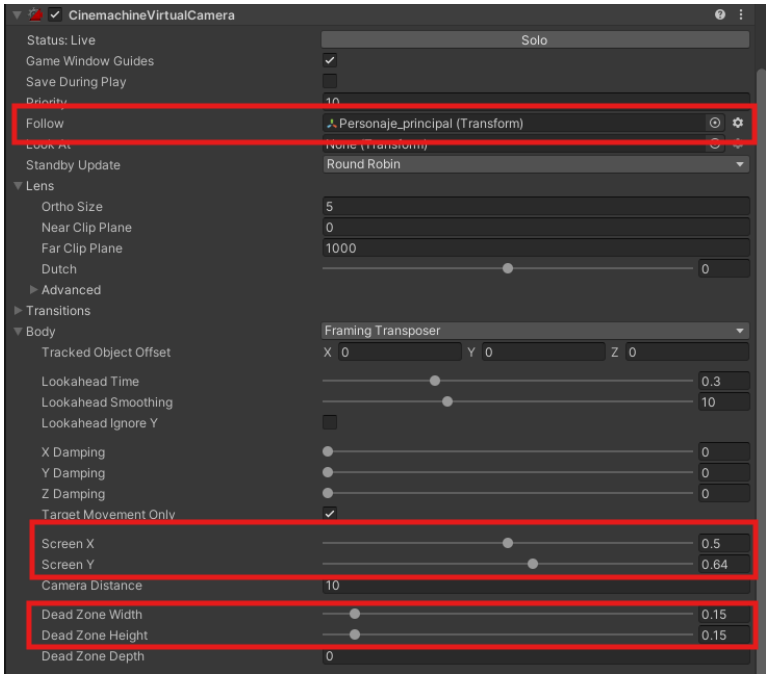
4.4 Cámara

Cuando se crea un proyecto en Unity, por defecto la cámara principal queda estática en un punto. Sin embargo, esto no debería ocurrir en un juego de plataformas, una de las características principales en este tipo de juegos es que la cámara siga constantemente el camino por donde va el personaje. Si esto no ocurriese, no podríamos contemplar el camino por donde se mueve Nano-BOT y lo perderíamos de vista.

Se ha añadido para conseguir este efecto un paquete a Unity llamado *Cinemachine*. El cual ofrece la opción de crear una cámara particular en 2D que se adhiere a nuestra cámara estática y tiene diversos parámetros específicos para estos casos.

En Unity se observa al seleccionar la cámara creada el parámetro *Follow*, para introducir a quién queremos que siga como objetivo, en este caso Nano-BOT. También nos permite regular la posición y ángulo de lo que queremos capturar.

Y, por último, hay unos parámetros muy interesantes que se llaman *Dead Zone*. Esto se utiliza para delimitar el movimiento que tendrá permitido nuestro personaje antes de que la cámara vuelva a perseguirlo.



Cuadro 37. Parámetros Cámara Principal.

Fuente: Elaboración Propia

Con todo esto ya tenemos como resultado un personaje con una cámara que sigue todos sus movimientos.

4.5 Salud

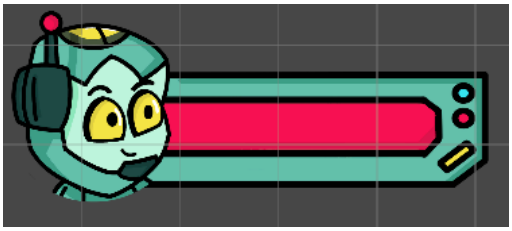
Se pretende que el usuario al experimentar el juego no solo aprenda conceptos de biomedicina, también se le agrega ese incentivo de superar el nivel con entretenimiento.

Es por ello por lo que en la vista del juego se ha agregado una barra de salud para que pueda morir el personaje, así intentará esquivar a los enemigos y proyectiles que vengan hacia él mientras va respondiendo cada pregunta.

Se va a distinguir el acabado de la barra salud en dos partes, por un lado, lo que ha sido el diseño tanto en Krita como en Unity y por otro lado lo que es la programación, similar a la metodología que se ha aplicado para explicar todo el desarrollo en el personaje.

4.5.1 Diseño

Nuevamente he vuelto a recurrir a Krita para dibujar el diseño que se verá por pantalla en la parte superior izquierda del jugador, tanto lo que es el contorno como la propia barra que bajará según sufra daño el personaje. El contorno pretende recrear lo que es la esencia del personaje.



Cuadro 38. Diseño Barra de Vida.

Fuente: Elaboración Propia

La barra roja está insertada como una imagen de tipo *Filled*, lo cual significa en Unity que actúa como si fuese un slider como en el menú de opciones.

Esta imagen lleva un parámetro llamado *Fill Amount*, que quiere decir que tiene una cantidad total según lo lleno que se encuentre esa imagen. Jugaremos en la parte programación con este parámetro para modificarlo.

4.5.2 Programación

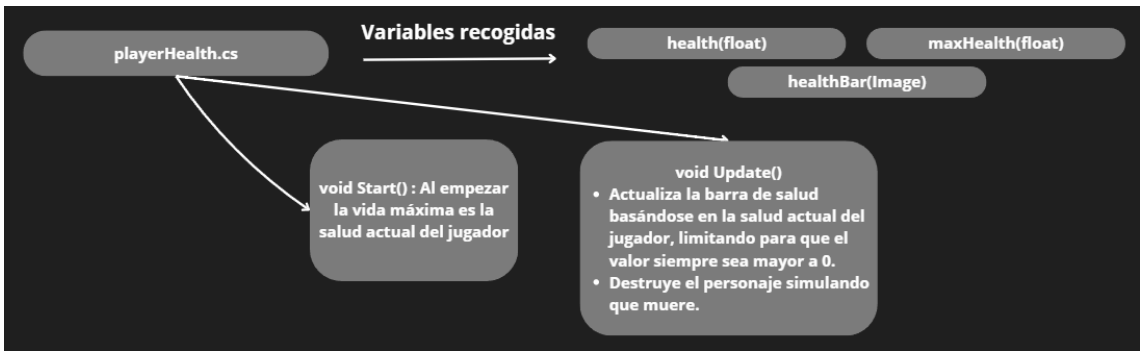
Queda claro que para que el personaje le baje la vida se debe distinguir en dos códigos, uno que gestione lo que es la vida del personaje (playerHealth.cs) y otro código que lleve los enemigos para infligir el daño (Damage.cs) y quede reflejado en la barra de salud.

- **playerHealth.cs:** Las variables públicas que recoge son la salud actual del jugador que se insta en Unity, su salud máxima y la imagen mencionada anteriormente que representa la salud. Se inicializa al empezar que la vida máxima es la salud actual del jugador (será 100). Actualiza la barra de salud basándose en la salud actual del jugador y lo limita para que no pueda ser valor negativo.

```
public class PlayerHealth : MonoBehaviour
{
    public float health;
    public float maxHealth;
    public Image healthBar;

    // Start is called before the first frame update
    void Start()
    {
        maxHealth = health;
    }

    // Update is called once per frame
    void Update()
    {
        healthBar.fillAmount = Mathf.Clamp(health / maxHealth, 0, 1);

        if(health <= 0)
        {
            Destroy(gameObject);
        }
    }
}
```

Cuadro 39. PlayerHealth.cs.

Fuente: Elaboración Propia



Cuadro 40. Pseudocódigo PlayerHealth.cs.

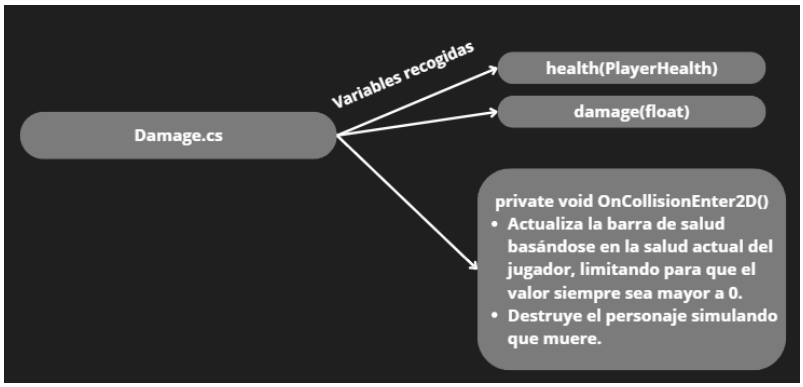
Fuente: Elaboración Propia

- **Damage.cs**: Las variables públicas que recoge son la salud del personaje y la cantidad de daño que infligirá al jugador, lo cual se insta también en Unity. Se crea un método llamado OnCollisionEnter2D() que se llama cuando el objeto colisiona con otro. Si el objeto colisionado tiene la etiqueta "Player" (la cual añadimos a Nano-BOT previamente), accede al componente de su vida y reduce su salud la cantidad de daño definida.

```
public class Damage : MonoBehaviour
{
    public PlayerHealth pHealth;
    public float damage;
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {

    }

    private void OnCollisionEnter2D(Collision2D other)
    {
        if(other.gameObject.CompareTag("Player"))
        {
            other.gameObject.GetComponent<PlayerHealth>().health -= damage;
        }
    }
}
```

Cuadro 41. Damage.cs.

Fuente: Elaboración Propia



Cuadro 42. Pseudocódigo Damage.cs.

Fuente: Elaboración Propia

Ya con esto hemos conseguido que el usuario calcule el daño que le están infligiendo los enemigos.

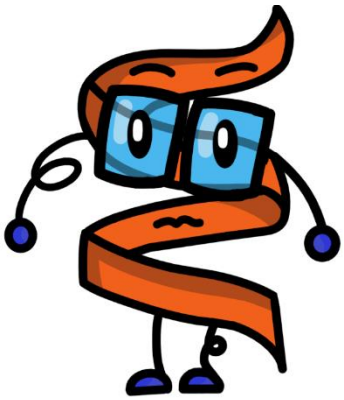
4.6 Personajes secundarios

En el punto 3.3 ya se han definido los personajes secundarios, por lo que no es necesario profundizar en sus descripciones aquí. Sin embargo, es importante destacar que cada uno de ellos tiene un propósito específico en el videojuego: algunos actúan como enemigos del protagonista, mientras que otros cumplen el rol de ayudantes y guías.

A continuación, se detallarán las mecánicas incorporadas en cada personaje, así como su diseño, para asegurar que cumplan eficazmente con su función y cómo interactúan con su entorno.

4.6.1 Proteína

El desarrollo de este personaje ha sido tan fundamental como el del personaje principal. Su contribución al Proyecto es insustituible y, de hecho, en el punto 4.9, al detallar la metodología de aprendizaje del juego, se demostrará por qué su funcionalidad es esencial.



Cuadro 43. Diseño Proteína

Fuente: Elaboración Propia

En cuanto al diseño de la Proteína, no se profundizará demasiado, ya que se ha optado por representarla de forma simplificada y con una apariencia infantil, al igual que los demás personajes. Aunque en la realidad una proteína es mucho más compleja que lo que muestra el dibujo, se ha buscado capturar su estructura enredada, añadiendo unas gafas como detalle distintivo que le confieren una personalidad intelectual.

La proteína es el personaje del videojuego que guía al jugador, explicándole el propósito del juego, advirtiéndole sobre las amenazas y evaluándolo en cuestiones de biotecnología. Para cumplir con este rol, se ha implementado un sistema de diálogo en Unity, que permite a la proteína interactuar de manera dinámica con el jugador, facilitando la comunicación y asegurando que reciba la información necesaria en cada etapa del juego.

4.6.1.1 Sistema de Diálogo

El objetivo es que cuando el personaje principal se acerque a la Proteína, al presionar el botón E, se active un panel en la parte inferior de la pantalla que muestre el mensaje o la información que la Proteína quiere transmitir.

Para lograr esto, es necesario diseñar un panel en la parte inferior de la pantalla donde se mostrará el diálogo. Además, se requiere un código que modele la información a transmitir y otro que gestione la lógica para que el diálogo aparezca solo cuando el personaje esté cerca de la Proteína y presione la tecla E.

En Unity, como se muestra en la Figura 46, el diseño del diálogo ha sido predefinido y está compuesto por los siguientes elementos:

1. Un icono representativo de la Proteína.
2. Una cabecera (header) que incluye el nombre del personaje.
3. Un cuerpo (body) que contiene un componente Text(TMP), donde se mostrará el mensaje que deberá actualizarse cada vez.
4. Un botón de "continuar" que permitirá avanzar a través de los diálogos.

Cuadro 44. Diseño diálogo Proteína.

Fuente: Elaboración Propia

El script que controla el sistema de diálogos es el Dialogue.cs. Gestiona la escritura progresiva del texto y la transición entre líneas de diálogo cuando el jugador presiona el botón de continuar.

```
public TextMeshProUGUI textComponent;
public string[] lines;
public float textSpeed;
public Button nextButton;

private int index;
private bool isTyping = false;

void Start()
{
    textComponent.text = string.Empty;
    nextButton.onClick.AddListener(OnNextButtonClick);
}

public void StartDialogue()
{
    index = 0;
    textComponent.text = string.Empty;
    StartCoroutine(TypeLine());
}

IEnumerator TypeLine()
{
    isTyping = true;
    foreach (char c in lines[index].ToCharArray())
    {
        textComponent.text += c;
        yield return new WaitForSeconds(textSpeed);
    }
    isTyping = false;
}
```

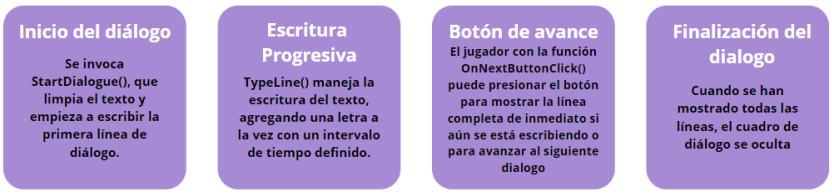
```
IEnumerator TypeLine()
{
    isTyping = true;
    foreach (char c in lines[index].ToCharArray())
    {
        textComponent.text += c;
        yield return new WaitForSeconds(textSpeed);
    }
    isTyping = false;
}

void OnNextButtonClick()
{
    if (isTyping)
    {
        StopAllCoroutines();
        textComponent.text = lines[index];
        isTyping = false;
    }
    else
    {
        NextLine();
    }
}

void NextLine()
{
    if (index < lines.Length - 1)
    {
        index++;
        textComponent.text = string.Empty;
        StartCoroutine(TypeLine());
    }
    else
    {
        gameObject.SetActive(false);
    }
}
```

Cuadro 45. Dialogue.cs.

Fuente: Elaboración Propia.

Este código tiene como entrada la variable del texto mencionado, el cual será modificado junto a la velocidad que se le quiere dar a la reproducción de texto, las líneas que se quieren introducir en Unity y el botón de continuar.

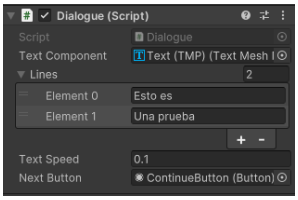
Puesto que es un desarrollo bastante extenso, se ha elaborado un pseudocódigo que esquematiza correctamente su funcionamiento:



Cuadro 46. Pseudocódigo Dialogue.cs.

Fuente: Elaboración Propia.

La manera en el que el desarrollador contempla esta modificación de los diálogos en Unity es un listado de líneas que se quieren añadir de la siguiente manera:



Cuadro 47. Dialogo en Unity.

Fuente: Elaboración Propia.

Por último, se tiene el código *NPCDialogueTrigger.cs*, el cual es el responsable de conseguir que el jugador al presionar la E y se encuentre cerca del personaje active el dialogo que hemos generado y editado.

```
public class NPCDialogueTrigger : MonoBehaviour
{
    public GameObject dialogueBox; // Asigna el DialogueBox en el Inspector

    private bool playerInRange = false;

    void Update()
    {
        // Inicia el diálogo al presionar "E" cuando el jugador está en rango
        if (playerInRange && Input.GetKeyDown(KeyCode.E))
        {
            if (!dialogueBox.activeSelf)
            {
                dialogueBox.SetActive(true); // Activa el DialogueBox
                dialogueBox.GetComponent<Dialogue>().StartDialogue(); // Inicia el diálogo
            }
        }
    }

    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.gameObject.CompareTag("Player"))
        {
            playerInRange = true;
            Debug.Log("Jugador en rango del NPC.");
        }
    }

    private void OnTriggerExit2D(Collider2D collision)
    {
        if (collision.gameObject.CompareTag("Player"))
        {
            playerInRange = false;
            Debug.Log("Jugador fuera del rango del NPC.");
        }
    }
}
```

Cuadro 48. NPCDialogueTrigger.cs.

Fuente: Elaboración Propia.

El script utiliza dos métodos, OnTriggerEnter2D() y OnTriggerExit2D(), para detectar cuándo el jugador entra y sale del área de colisión (o "rango") del NPC. Cuando el jugador entra en rango, se establece playerInRange() como true, y cuando sale, se cambia a false.

Por otro lado, en el método Update(), que se ejecuta en cada frame, el script verifica si el jugador está dentro del rango (playerInRange es true) y si ha presionado la tecla "E". Si ambas condiciones se cumplen, el cuadro de diálogo (dialogueBox) se activa y se llama al método StartDialogue() para iniciar el diálogo. Antes de activar el cuadro de diálogo, el script verifica si ya está activo para evitar reiniciarlo accidentalmente si el jugador sigue presionando "E".

4.6.1.2 Resultado

El resultado que conseguimos es el siguiente:

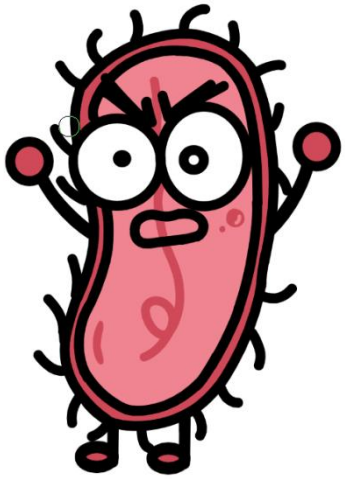
Cuadro 49. Resultado Diálogo.

Fuente: Elaboración Propia.

4.6.2 Bacteria

Teniendo en cuenta que el personaje de la bacteria es un microorganismo que puede presentarse de diversas formas, ya sean esferas, bastones, espirales o hélices, se ha decidido elegir una forma similar a ese estilo, sacando ciertas protuberancias a su alrededor. También se ha representado en su interior su nucleoide, dando como resultado la Figura 52.

Vuelve a ser un diseño simple pero que da ese tono infantil que se espera a la jugabilidad.



Cuadro 50. Diseño Bacteria

Fuente: Elaboración Propia

Antes de nada, cabe resaltar que previamente se ha utilizado una animación de andar que ha sido introducida al igual que se ha explicado con el personaje principal.

Por otro lado, en cuanto a las **mecánicas** se refiere, se pretendía que tuviese como se ha indicado en la preproducción del videojuego lo siguiente:

- Que inflija daño a Nano-BOT
- Que se mueva constantemente en movimientos horizontales, simulando el movimiento de un zombi.

La parte de infligir daño acabamos de ver cómo se desarrolla, simplemente se le ha añadido el código al png de Bacteria. Sin embargo, todavía faltaba representar ese movimiento del que estamos hablando.

4.6.2.1 EnemyPatrol.cs

Este código reflejará en Unity el movimiento comentado anteriormente para simular que patrulla.

```
public class EnemyPatrol : MonoBehaviour
{
    public GameObject pointA;
    public GameObject pointB;

    private Rigidbody2D rb;
    private Animator anim;
    private Transform currentPoint;
    public float speed;

    // Start is called before the first frame update
    void Start()
    {
        rb = GetComponent<Rigidbody2D>();
        anim = GetComponent<Animator>();

        // Cambiamos el punto inicial para que comience moviéndose hacia el punto A
        currentPoint = pointA.transform;

        // Iniciamos el movimiento hacia el punto A
        rb.velocity = new Vector2(-speed, 0); // Mover hacia la izquierda
        anim.SetBool("IsRunning", true);
    }
```

Cuadro 51. EnemyPatrol.cs.

Fuente: Elaboración Propia.

Las variables recogidas son los dos puntos entre los que el npc se moverá, el componente de Unity que controla la física de la Bacteria, el animator que controla la animación de la bacteria, la referencia del punto al que se está dirigiendo y la velocidad a la que se moverá.

En cuanto al método **Start():**

- Obtiene y almacena la referencia al componente RigidBody
- Obtiene y almacena la animación
- Se establece que el npc empiece a moverse al punto A
- Activa la animación de moverse

```
// Update is called once per frame
void Update()
{
    Vector2 point = currentPoint.position - transform.position;
    if (currentPoint == pointB.transform)
    {
        rb.velocity = new Vector2(speed, 0); // Mover hacia la derecha
    }
    else
    {
        rb.velocity = new Vector2(-speed, 0); // Mover hacia la izquierda
    }

    if (Vector2.Distance(transform.position, currentPoint.position) < 0.5f)
    {
        flip();

        // Alternar el punto de destino entre A y B
        currentPoint = (currentPoint == pointA.transform) ? pointB.transform : pointA.transform;
    }
}

private void flip()
{
    // Invertimos la escala X para hacer el flip correctamente
    Vector3 localScale = transform.localScale;
    localScale.x *= -1;
    transform.localScale = localScale;
}

private void OnDrawGizmos()
{
    Gizmos.DrawWireSphere(pointA.transform.position, 0.5f);
    Gizmos.DrawWireSphere(pointB.transform.position, 0.5f);
    Gizmos.DrawLine(pointA.transform.position, pointB.transform.position);
```

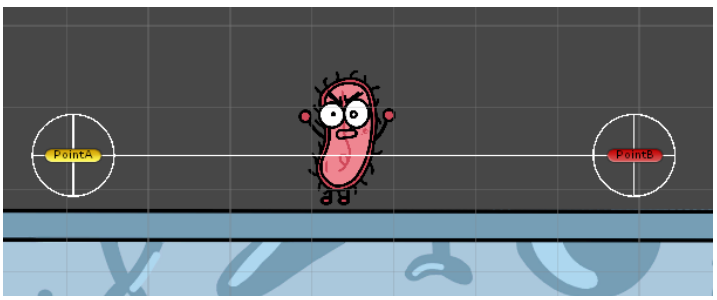
Cuadro 52. EnemyPatrol.cs parte 2.

Fuente: Elaboración Propia.

En cuanto al método **Update():**

- Calcula la dirección en la que debe moverse el NPC
- Dependiendo si el NPC está moviendo hacia un punto, se ajusta la velocidad hacia un lado u otro
- Si la distancia llega a 0.5, el NPC llama al método flip() para que cambie su orientación y cambia su objetivo al otro punto.

El método flip() invierte la escala para que mire de un lado a otro. Y el OnDrawGizmos() se usa como referencia visual en la ventana de Unity. Así el desarrollador puede visualizar una guía de movimientos.

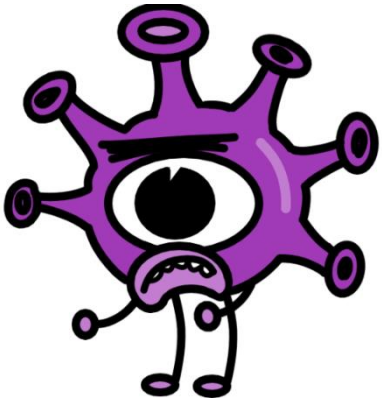


Cuadro 53. Resultado movimiento Proteína.

Fuente: Elaboración Propia.

4.6.3 Virus

El diseño del virus en este proyecto se inspiró en la estructura visual del COVID-19, un virus ampliamente reconocido en la actualidad y que claramente indica su papel como antagonista en el juego.



Cuadro 54. Diseño Virus

Fuente: Elaboración propia

Este diseño se puede observar en la Figura 56. Para darle funcionalidad, se ha integrado el código *Damage.cs*, lo que asegura que Nano-BOT no pueda atravesar al virus sin recibir daño, haciendo la interacción más realista y desafiante.

Además de estas mecánicas, el virus es un personaje estático que dispara proyectiles, los cuales infligen 20 puntos de daño a Nano-BOT al impactar.

Para implementar esta funcionalidad, se han desarrollado dos scripts: *EnemyShooting.cs*, que controla los disparos del virus hacia el jugador dentro de un rango específico, y *EnemyBulletScript.cs*, que gestiona el comportamiento de las balas, incluyendo su movimiento y destrucción tras el impacto.

4.6.3.1 EnemyShooting.cs

El primer código desarrollado es el siguiente:

```
public class EnemyShooting : MonoBehaviour
{
    public GameObject bullet;
    public Transform bulletPos;

    private float timer;
    private GameObject player;
    // Start is called before the first frame update
    void Start()
    {
        player = GameObject.FindGameObjectWithTag("Player");
    }

    // Update is called once per frame
    void Update()
    {
        timer += Time.deltaTime;

        float distance = Vector2.Distance(transform.position, player.transform.position);
        Debug.Log(distance);

        if(distance < 10)
        {
            timer += Time.deltaTime;

            if (timer > 4)
            {
                timer = 0;
                shoot();
            }
        }
    }

    void shoot()
    {
        Instantiate(bullet, bulletPos.position, Quaternion.identity);
    }
}
```

Cuadro 55. EnemyShooting.cs.

Fuente: Elaboración Propia.

En el caso del código *EnemyShooting.cs* la primera variable definida en el código es *bullet*, que representa la bala o proyectil que será disparado por el Virus en dirección al jugador. La posición desde la cual se disparará la bala se especifica mediante la variable *bulletPos*, la cual se asigna en Unity. Además, se configuran la frecuencia de disparo y el jugador objetivo al que se dirigirán las balas.

Por otro lado el **método Start():** busca al jugador en la escena utilizando la etiqueta "Player" y guarda una referncia de él.

Mientras que el **método Update():**

- Llama cada frame. Incrementa el temporizador con el tiempo transcurrido ("Time.deltaTime")
- Calcula la distancia entre el enemigo y el jugador utilizando Vector2.distance.
- Establece la lógica de que si la distancia entre Virus y Jugador entra dentro del rango de 10 unidades dispara.
- Establece la lógica de que si el temporizador supera los 4 segundos se reinicia el temporizador y llama al método shoot().

El **método shoot()** instancia una bala en la posición indicada del enemigo con una rotación predeterminada ("Quaternion.identity")

4.6.3.2 EnemyBulletScript.cs

El segundo código desarrollado es el siguiente:

```
public class EnemyBulletScript : MonoBehaviour
{
    private GameObject player;
    private Rigidbody2D rb;
    public float force;
    private float timer;

    // Start is called before the first frame update
    void Start()
    {
        rb = GetComponent<Rigidbody2D>();
        player = GameObject.FindGameObjectWithTag("Player");

        Vector3 direction = player.transform.position - transform.position;
        rb.velocity = new Vector2(direction.x, direction.y).normalized * force;

        float rot = Mathf.Atan2(-direction.y,-direction.x) * Mathf.Rad2Deg;
        transform.rotation = Quaternion.Euler(0, 0, rot + 90);
    }

    // Update is called once per frame
    void Update()
    {
        timer += Time.deltaTime;

        if(timer > 10)
        {
            Destroy(gameObject);
        }
    }

    private void OnTriggerEnter2D(Collider2D other)
    {
        if(other.gameObject.CompareTag("Player"))
        {
            other.gameObject.GetComponent<PlayerHealth>().health -= 20;
            Destroy(gameObject);
        }
    }
}
```

Cuadro 56. EnemyBulletScript.cs.

Fuente: Elaboración Propia.

Las variables utilizadas son la referencia del jugador en la escena, “rb” que representa el Rigidbody2D (La física que hemos comentado en otros puntos) de la bala, la fuerza con la que se moverá la bala y el temporizador que controla cuánto tiempo estará activa antes de ser destruida.

El **método Start()** por una parte:

- Obtiene la referencia al Rigidbody2D de la bala y se busca al jugador en la escena.
- Calcula la dirección hacia el jugador (direction) y se aplica una velocidad a la bala en esa dirección normalizada, multiplicada por la force.
- Finalmente, se calcula el ángulo de rotación para que la bala apunte hacia el jugador, y se aplica esa rotación.

El **método Update()** :

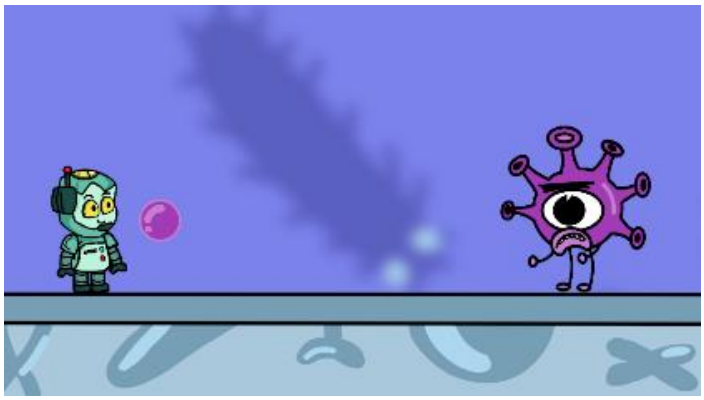
- El temporizador (timer) se incrementa en cada frame. Si el timer supera los 10 segundos, la bala se destruye automáticamente para evitar que permanezca indefinidamente en la escena.

El **método OnTriggerEnter2D(Collider2D other)** :

- Se activa cuando la bala colisiona con otro objeto. Si el objeto con el que colisiona tiene la etiqueta "Player" (Como es Nano-BOT), el script accede al componente PlayerHealth del jugador y reduce su salud en 20 unidades.
- Después de hacer daño al jugador, la bala se destruye.

4.6.3.3 Resultado

En resumen, se ha logrado que el personaje del virus dispare una bala cuando Nano-BOT se encuentra a una determinada distancia. Esta bala seguirá la posición de Nano-BOT y se destruirá al impactar, infligiendo daño, o al cumplirse un tiempo preestablecido.



Cuadro 57. Resultado disparo Virus.

Fuente: Elaboración Propia.

Por otro lado, para esclarecer las variables y elementos que han sido insertados, el primer código se ha aplicado al Virus y el segundo a la bala diseñada en Krita.

Cuadro 58. Resultado variables virus en Unity.

Fuente: Elaboración Propia.

4.7 Plataformas en movimiento

Las plataformas móviles son un pilar esencial en cualquier juego de plataformas, agregando dinamismo y desafiando la habilidad del jugador para coordinar saltos y movimientos precisos. Es por eso por lo que, para elevar el nivel de desafío y mantener al jugador constantemente comprometido, he incorporado plataformas en movimiento dentro del diseño del juego.

Estas plataformas se desplazan a lo largo de trayectorias predefinidas, obligando al jugador a calcular el tiempo de sus saltos y anticipar los movimientos del entorno, lo que añade una capa adicional de estrategia y dificultad a la experiencia de juego. También serán un elemento necesario para lo que será el desarrollo de entorno que comentaremos en el punto 4.9.

Para conseguir este elemento, primeramente, he diseñado lo que es la propia plataforma en *Krita* y exportándolo a formato png en Unity. Este png será el que llevará el código para que se comporte como una plataforma móvil.

El código que se le ha añadido se llama *MovingPlatform.cs*. Este código está desarrollado de tal manera que consigue que una plataforma se mueva de un punto a otro de manera suave, utilizando la interpolación lineal *Lerp*, una operación matemática que permite calcular un punto intermedio entre dos valores dados, a continuación, se explicará detalladamente:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class MovingPlatform : MonoBehaviour
{
    public Transform platform;
    public Transform startPoint;
    public Transform endPoint;
    public float speed = 1.5f;

    int direction = 1;

    private void Update()
    {
        Vector2 target = currentMovementTarget();

        platform.position = Vector2.Lerp(platform.position, target, speed * Time.deltaTime);

        float distance = (target - (Vector2)platform.position).magnitude;

        if (distance <= 0.1f)
        {
            direction*= -1;
        }
    }
```

Cuadro 59. MovingPlatform.cs.

Fuente: Elaboración Propia.

Las variables son:

- platform: de tipo Transform, representa la plataforma que se va a mover. Será el componente que almacena la posición, rotación y escala del objeto.
- startPoint y endPoint: de tipo Transform, los dos puntos de inicio y final que delimitan el movimiento de la plataforma.
- speed: de tipo float, controla la velocidad a la que la plataforma se mueve. Será configurable en el editor de Unity.
- direction: de tipo int, variable para controlar la dirección del movimiento, si es 1 la plataforma se mueve a startPoint, si es -1 la plataforma se mueve a endPoint.

En el inspector de Unity se ven estas variables de esta forma:

Cuadro 60. MovingPlatform.cs en Unity.

Fuente: Elaboración Propia.

Por otro lado, en el método Update(), el cual recordemos que es aquel que se ejecuta de manera continua, contiene lo siguiente:

1. Vector2 target= currentMovementTarget(): Llama al método currentMovementTarget() para obtener la posición del destino actual de la plataforma, ya sea el startPoint o el endPoint, dependiendo de la dirección.

2. platform.position= Vector2.Lerp(platform.position, target, speed * Time.deltaTime):
    a. Aquí se utiliza la función Vector2.Lerp para interpolar linealmente entre la posición actual de la plataforma y el objetivo (target).
    b. Lerp toma tres argumentos: la posición inicial (platform.position), la posición final (target), y un valor de interpolación que determina cuán lejos moverse hacia el objetivo en este frame.
    c. speed * Time.deltaTime se utiliza para asegurarse de que el movimiento sea suave y dependiente del tiempo.

3. float distance = (target - (Vector2)platform.position).magnitude: Calcula la distancia entre la posición actual de la plataforma y el objetivo. Se convierte platform.position a un Vector2 para que la operación de resta sea válida y el resultado sea la magnitud del vector, que representa la distancia.

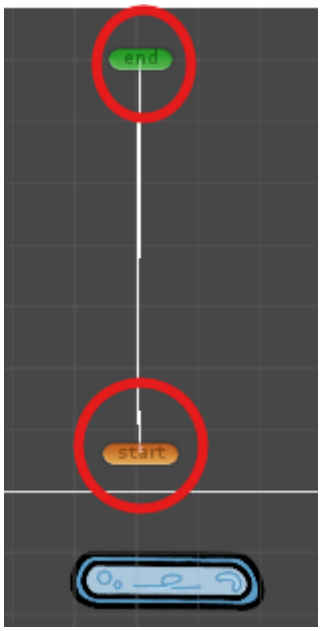
4. if (distance <= 0.1f):
    a. Verifica si la distancia entre la plataforma y el objetivo es menor o igual a 0.1f.
    b. Si es así, significa que la plataforma ha alcanzado (o está muy cerca de alcanzar) su objetivo, por lo que cambia la dirección del movimiento multiplicando direction por -1.

```
Vector2 currentMovementTarget()
{
    if(direction== 1)
    {
        return startPoint.position;
    }
    else{
        return endPoint.position;
    }
}

private void OnDrawGizmos(){

    if(platform!=null && startPoint!=null && endPoint != null){
        Gizmos.DrawLine(platform.transform.position, startPoint.position);
        Gizmos.DrawLine(platform.transform.position, endPoint.position);
    }
}
```

Cuadro 61. MovingPlatform.cs parte 2.

Fuente: Elaboración Propia.

En la segunda parte del código se desarrolla el método llamado en Update() que es currentMovementTarget() ya explicado y OnDrawGizmos(), un método que nos permite dibujar ayudas visuales que solo aparecen en el editor, así facilita la configuración de la plataforma.



Cuadro 62. Guizmos de la plataforma.

Fuente: Elaboración Propia.

El resultado final es el siguiente:

Cuadro 63. Resultado final plataforma movil.

Fuente: Elaboración Propia.

4.8 Recompensa

Al igual que las plataformas, como bien se había comentado en la introducción de este Trabajo Final de Grado, no podía faltar un sistema de recompensas. Para darle más emoción al videojuego y coherencia con la historia, a lo largo del nivel hay repartidos antibióticos que deberá encontrar el jugador.

La recolecta de antibióticos justificaría el intentar combatir las amenazas que invade a la célula a la que habita nuestro personaje.

Para conseguir estas recompensas por una parte se ha diseñado el dibujo de un medicamento que se distribuirá a lo largo del nivel. Por otra parte, debemos visualizar al igual que se ha hecho con la barra de salud, un contador que refleje cuántos antibióticos recoge el jugador.

Cuadro 64. Contador de antibióticos en Unity.

Fuente: Elaboración Propia.

Este contador se podrá observar en la parte superior de la derecha. La idea es que este contador iniciado en 0 vaya sumando 1 punto cuando Nano-BOT recoja dichos antibióticos, el cual llevará insertado el código llamado *ScoreScript.cs* consigue este objetivo.

```
public class ScoreScript : MonoBehaviour
{
    public TextMeshProUGUI MyscoreText;
    private int ScoreNum;

    // Start is called before the first frame update
    void Start()
    {
        ScoreNum = 0;
        MyscoreText.text = ScoreNum.ToString();
    }

    private void OnTriggerEnter2D(Collider2D antibiotico)
    {
        if(antibiotico.tag == "antibiotico")
        {
            ScoreNum += 1;
            Destroy(antibiotico.gameObject);
            MyscoreText.text = ScoreNum.ToString();
        }
    }
}
```

Cuadro 65. ScoreScript.cs.

Fuente: Elaboración Propia.

Las variables son:

- MyscoreText: de tipo TextMeshProUGUI, variable que recoge el texto que refleja el puntaje.
- Score Num: de tipo int, variable que almacena el puntaje. Este se inicializará en 0 e incrementará cuando el personaje interactúa con el objeto.

En el método Start() es donde inicializamos el puntaje en 0 al comenzar el juego, después MyscoreText.text = ScoreNum.ToString() convierte el puntaje en cadena de texto y lo actualiza para reflejar el puntuaje inicial en la pantalla.

Y por último el **método OnTriggerEnter2D(Collider2D antibiotico)** hace lo siguiente:

- Verifica si el objeto que ha colisionado tiene la etiqueta "antibiotico".
- Esta etiqueta debe ser asignada a los objetos relevantes (los "antibióticos") en el editor de Unity.
- Incrementa el puntaje en 1 si la colisión es con un objeto etiquetado como "antibiotico".
- Destruye el objeto "antibiotico" que ha sido tocado, removiéndolo de la escena.
- Actualiza el texto de la UI para mostrar el nuevo puntaje después de la colisión.

En el videojuego podemos encontrarnos el resultado de esta forma:

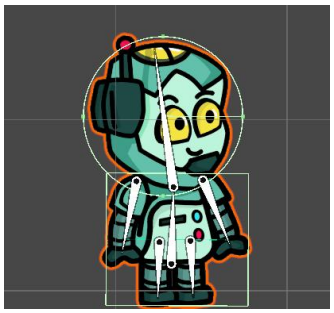
Cuadro 66. Resultado recompensa.

Fuente: Elaboración Propia.

4.9 Entorno y elaboración de nivel

Antes de continuar con lo que es el desarrollo de todo el entorno, cabe destacar una particularidad importante que creía coherente resaltar en este apartado. Y es que Nano-BOT para que esté controlado parcialmente por el motor de física y pueda interactuar con el entorno se le han añadido dos elementos:

- Un RigidBody 2D: esto hace que le afecte, por ejemplo, la gravedad.
- Dos Box Collider 2D: define la forma que representa el jugador en el motor de física, y la cual se usa para el cálculo de colisiones.



Cuadro 67. Físicas Nano-BOT.

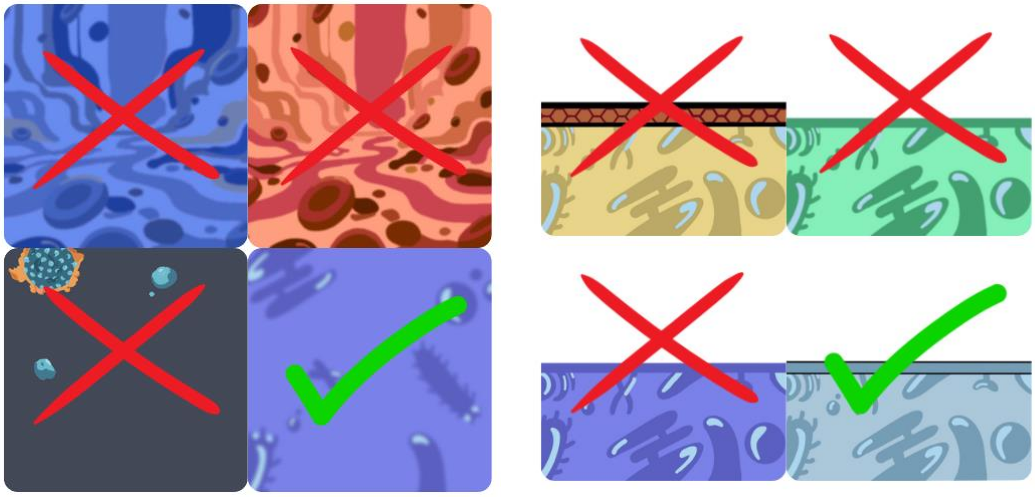
Fuente: Elaboración Propia.

Cada elemento del suelo y cada plataforma que forman el terreno sobre el que se desplaza el personaje cuenta con un Box Collider integrado. Estos colliders interactúan con el collider del personaje, permitiéndole mantenerse sobre las estructuras de manera estable.

El diseño de estos realmente ya se puede contemplar en otras figuras vistas anteriormente, sin embargo, en este apartado se expondrá el por qué del diseño. Y es que, en la historia, como bien se ha comentado en la preproducción, se pretende que Nano-BOT se sumerja en una misión microscópica.

Dado que mis diseños son de estilo cartoon, tuve muchas dudas sobre cómo quería enfocar la estética. La idea inicial era crear una superficie que imitara la membrana celular, ya que la historia gira en torno a Nano-BOT salvando a la célula. Sin embargo, el enfoque en la metodología de aprendizaje limitaba mi creatividad. Finalmente, opté por patrones que sumergieran al jugador en un mundo microscópico.

De hecho, se realizaron varias versiones de estos diseños para intentar lograr tanto una coherencia con la historia como el estilo visual que ya se refleja en la pantalla inicial:



Cuadro 68. Backgrounds.

Fuente: Elaboración Propia.

Los diseños que no fueron seleccionados se descartaron debido a su exceso de detalles, porque transmitían temáticas que no reflejaban de manera distintiva la biotecnología, o simplemente porque el escenario no contrastaba adecuadamente con los personajes. Esto ocurrió igual con los diversos elementos que también se añadieron para complementar la escena.

4.9.1.1 Aplicación de la metodología de aprendizaje

Para concluir, el mecanismo de aprendizaje, que es el propósito principal del videojuego, se logró con éxito tal y como se había planteado en la fase de preproducción.

1. El jugador se acerca a la Proteína y presiona la tecla E para que le haga preguntas de tipo test.
2. Los caminos estarán numerados en relación a las respuestas.
    a. Si el jugador se equivoca de respuesta se introducirá en un camino que le llevará a la muerte segura.
    b. Si el jugador acierta en su respuesta y por ende el camino, continuará.

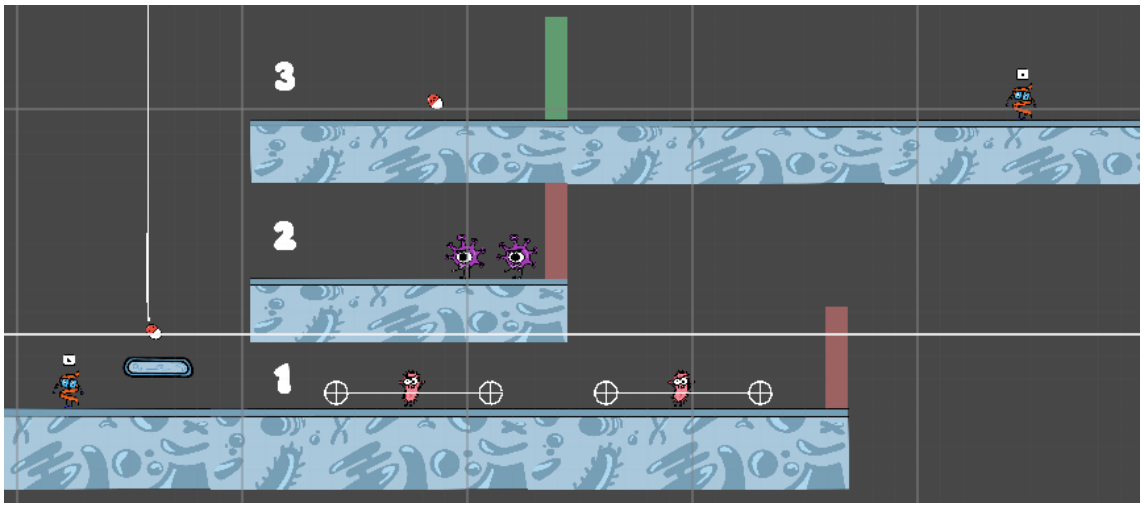
Este suceso ocurrirá sucesivamente hasta llegar al final. El personaje recolectará por el camino los antibióticos necesarios para salvar a la célula.

La justificación de esta metodología es darle un giro al concepto tradicional de memorizar contenidos y convertir el aprendizaje en una experiencia interactiva y dinámica. Al integrar preguntas de test en el contexto del juego, se fomenta la participación activa y la aplicación práctica de los conocimientos. Este enfoque no solo refuerza el aprendizaje mediante la repetición y la retroalimentación inmediata, sino que también mantiene al jugador motivado a medida que avanza y enfrenta desafíos relacionados con el contenido educativo. Al combinar la educación con la mecánica del juego, se busca mejorar la retención de la información y hacer el proceso de aprendizaje más atractivo y efectivo.

Si el personaje falla y muere, volverá al inicio. Esto es útil porque:

1. **Refuerza el Aprendizaje**: La repetición permite que el jugador revise y refuerce el contenido que ha fallado, ayudando a consolidar el conocimiento.
2. **Promueve la Reflexión**: Al regresar al inicio, el jugador tiene la oportunidad de reflexionar sobre sus errores y mejorar su comprensión antes de intentarlo nuevamente.
3. **Incrementa la Motivación**: Superar obstáculos y aprender de los errores fomenta la perseverancia y el sentido de logro, manteniendo al jugador motivado.
4. **Asegura la Comprensión Completa**: Repetir los desafíos asegura que el jugador cubra todos los aspectos del contenido educativo, maximizando la efectividad del aprendizaje.

En el videojuego se puede observar de esta manera:



Cuadro 69. Metodología de aprendizaje.

Fuente: Elaboración Propia.

Capítulo 5. Estrategias de marketing digital para un *serious game*

A continuación, se presentarán los tres métodos seleccionados para la promoción del videojuego antes de su fecha de lanzamiento. Estas propuestas son estrategias útiles que demuestran diversas formas de promocionar y analizar la salida de un videojuego u otro tipo de producto.

5.1 User Testing

El **user testing**, o prueba de usuario, consiste en observar y analizar cómo los usuarios interactúan con el producto para identificar problemas, recoger comentarios y obtener información valiosa que puede mejorar el producto. Esta técnica permite a los desarrolladores ajustar y perfeccionar su creación según la experiencia real de los usuarios.

Personalmente, he decidido emplear esta técnica en el desarrollo de "BioPlay" porque considero crucial mostrar la jugabilidad del videojuego antes de su lanzamiento oficial. Realizar pruebas con usuarios me brinda la oportunidad de recopilar opiniones directas de aquellos que están realmente interesados en el juego, permitiéndome ajustar y mejorar aspectos que quizás no había considerado. Escuchar a los jugadores ayuda a garantizar que el producto final sea lo más atractivo y funcional posible.

Esta técnica no es nueva para mí; ya la utilicé en el grado de Tecnología Digital y Multimedia, específicamente en el curso de Modelos de Negocio. En esa ocasión, presentábamos a nuestros compañeros el modelo de una página web en desarrollo. Después de la presentación, recogíamos sus opiniones y comentarios para realizar un análisis detallado del producto. Este proceso nos permitió identificar áreas de mejora y adaptar el diseño según las necesidades y preferencias del usuario.

El testing he decidido dividirlo en dos partes:

- La pantalla inicial
- La pantalla de juego

Ha habido un total de 14 usuarios a fecha 20/08/2024 que han hecho uso del videojuego en las primeras pruebas, pero no he querido plasmar todos los comentarios en la memoria para que no fuese demasiado extenso lo que quiero mostrar. Sin embargo, aquí dejo constancia de los comentarios de 3 usuarios para que se entienda adecuadamente este proceso.

5.1.1 Pantalla inicial

Al diseñar la pantalla inicial pretendía que los usuarios se sintieran cómodos en la bienvenida del videojuego. Que pudiesen localizar cada uno de los elementos a los que quería que accediesen fácilmente.

En base a los comentarios y cómo se desenvuelven podemos deducir si se ha conseguido este objetivo:

Usuario Anónimo 1

COMENTARIOS:

- El fondo lo ve muy original, nada más ver el fondo ya se sabe de qué va.
- La música hace notar que va a ser para todos los públicos, sensación de que el juego es didáctico

- El movimiento del fondo da dinamismo, visualmente le parece un acierto.

CÓMO SE DESENVUELVE:

- Lo primero que hace es entrar en el botón historia, la lee y vuelve atrás.
- Sigue por el botón de los personajes.
- No se da cuenta de que los botones de los personajes llevan consigo información descriptiva y vuelve atrás.
- Le da a jugar, no prueba el botón de opciones y controles.

Usuario Anónimo 2

COMENTARIOS:

- Según él es agradable el juego, visualmente le gusta
- La música no le transmite mucho
- Los botones de los personajes le incitan a conocer su información
- Esperaba la opción de cambiar de personaje

CÓMO SE DESENVUELVE:

- Primero le da al botón de personajes
- Le da directamente a jugar sin observar los demás botones

Usuario Anónimo 3

COMENTARIOS:

- Visualmente le parece colorido y llamativo.
- El sonido le parece agradable, acompaña bien los movimientos que hacen los elementos flotantes.
- Es intuitivo el tema.

CÓMO SE DESENVUELVE:

- Primero le da al botón de historia, le causa curiosidad.
- Aprieta a los botones de los personajes.
- Entra en el botón de opciones a regular el volumen.

5.1.2 Pantalla de juego

El objetivo principal al diseñar la pantalla de juego es garantizar que los usuarios se adapten fácilmente a la jugabilidad, comprendan claramente su objetivo en el videojuego y disfruten mientras aprenden. Observemos los comentarios de los usuarios:

Usuario Anónimo 1:

COMENTARIOS:

- Encuentra que la interfaz de la pantalla de juego es clara y funcional, sin elementos que lo distraigan del objetivo principal.
- Aprecia que los elementos interactivos estén bien diferenciados, lo que le facilita la comprensión de lo que debe hacer
- Entorno agradable

CÓMO SE DESENVUELVE:

- Comienza por explorar el entorno moviéndose hacia la izquierda y la derecha para familiarizarse con los controles.
- Prueba las interacciones disponibles, como recoger objetos y hablar con personajes.
- Explora los distintos caminos posibles

Usuario Anónimo 2:

COMENTARIOS:

- Le parece curioso que pueda usar tanto las flechas como las teclas WASD.
- Diseño de personajes “muy tierno”.
- Podría considerarlo como un juego indie.

CÓMO SE DESENVUELVE:

- Se toma unos segundos para explorar el entorno sin realizar acciones específicas.
- Avanza antes de entender el objetivo, lo que le lleva algo de tiempo.
- Regresa a leer el dialogo de la Proteína.
- Responde las respuestas adecuadamente.

Usuario Anónimo 3:

COMENTARIOS:

- Considera que la pantalla de juego es agradable, pero le gustaría más claridad en los objetivos iniciales.
- Cambiaría la música de fondo ya que es igual que la de la pantalla de inicio
- Como sugerencia le gustaría que hubiese más coleccionables y que se plasmase en una pantalla final.

CÓMO SE DESENVUELVE:

- Recorre el camino adecuadamente.
- Responde las preguntas, algunas veces se equivoca y vuelve a empezar.

5.1.3 Conclusiones

Ahora con todos los comentarios y gracias a ver cómo se desenvuelven cada uno de ellos podemos tener en cuenta varios puntos para mejorar el resultado del videojuego:

- Hacer más vistoso el botón de personajes que llevan consigo información. Los usuarios no se dan cuenta.
- Las opciones no llaman la atención a los usuarios, no suele ser muy usada esta ventana.
- Echan en falta la recolección de más objetos.
- Se puede contemplar más variedad de sonidos.
- Tener en cuenta desarrollar más niveles.

Como conclusión, la información recopilada muestra que esta estrategia es valiosa y beneficiosa para los desarrolladores. Es un claro ejemplo de cómo, con el tiempo, la técnica PULL—que se basa en escuchar a los usuarios y adaptar el producto a sus necesidades y deseos—ha demostrado ser más eficaz que la técnica PUSH. En el futuro se harían modificaciones del videojuego para incrementar el interés en el público.

5.2 Keyword Hunting

El *keyword hunting* consiste en investigar por qué términos de búsqueda o palabras clave puede ser útil que los contenidos de nuestro producto, en este caso el videojuego, se posicionen entre los primeros resultados de la página de respuestas de Google.

Se dividirá esta metodología en dos fases:

- Fase 1: Identificaremos las palabras clave vinculadas a la temática del videojuego que estamos promocionando, clasificaremos dichas palabras clave según la intención del usuario, analizaremos los contenidos que actualmente tienen mejor posicionamiento, y finalmente, elaboraremos una propuesta de tipos de contenido que el producto debería desarrollar para optimizar su visibilidad en los motores de búsqueda.

- Fase 2: Nos daremos de alta en la herramienta de análisis SEO llamada *Ubersuggest*, donde extraeremos los datos relacionados con cada expresión/palabra clave de búsqueda.

5.2.1 Fase 1: Palabras clave

Tendremos en cuenta que las palabras clave se clasifican en:

- Navegacional: El usuario tiene claro el destino al que quiere llegar en la web, pero desconoce la URL exacta o por pereza, lo busca desde Google.
- Informacional: Incluye consultas generales y la búsqueda de información sobre una amplia variedad de temas.
- Comercial: se busca información, pero el usuario está más cerca de realizar la compra.
- Transaccional: El usuario tiene la intención de realizar una acción concreta, como hacer una compra o suscribirse a un boletín informativo.

Palabra clave	Clasificación	URL 1er posicionado	Título 1er posicionado	Tipología 1er posicionado	Propuesta tipo de contenido
BioPlay	Navegacional	https://www.amazon.es/BIOPLAY%C2%AE-pl%C3%A1stico-org%C3%A1nico-sostenible-interior/dp/B0BQZFK51G	BIOPLAY Juguete de arena a partir de 1 año de plástico.	Página de comercio electrónico.	Amazon ofrece un juguete para niños de la marca BIOPLAY.
Videojuego de biomedicina	Comercial	https://www.cerebriti.com/juegos-de-ciencias/biomedicina-cuanto-sabes	Juego de Biomedicina ¿Cuánto sabes?	Página web con apartados y juegos.	Diversos juegos educativos de cualquier materia.
Descargar BioPlay gratis	Transaccional	https://play.google.com/store/apps/details?id=com.betplay.betplayapp&hl=es_EC&gl=CO&pli=1	BetPlay – Apps en Google Play.	Web de descarga de aplicaciones de Google.	Marca de apuestas Online para eventos del año en disciplinas deportivas.
Cómo jugar a BioPlay	Informacional	https://support.bang-olufsen.com/hc/es/articles/360018801597-Uso-de-Beoplay-Portal-para-JUGAR-POR-CABLE-minijack-de-3-5-mm	Uso de Beoplay Portal para JUGAR POR CABLE(minijack)	Página web con artículos.	Página web de una marca que ofrece servicio técnico y productos en línea.

Tabla 2. KeywordHunting.

Fuente: Elaboración Propia.

Según el *keyword hunting* que hemos realizado hay tres conclusiones clave a tener en cuenta:

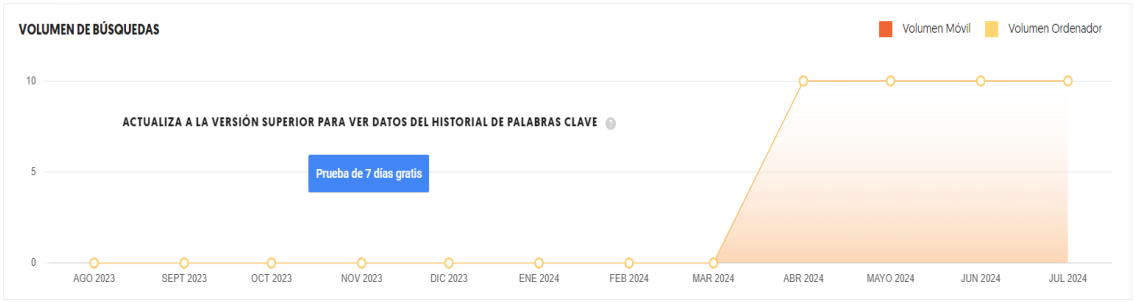
1. La marca Amazon, la cual es una de las empresas más grandes del mundo en ventas online, pone como primeras búsquedas la marca BIOPLAY, la cual se dedica a vender juguetes para niños. Eso puede dificultar con grandes creces la promoción de nuestro producto.
2. A parte de la empresa de juguetes, como competidores no tenemos grandes referentes que tengan por nombre *BioPlay*. Podemos conseguir que nuestro videojuego tenga exclusividad y entidad propia.
3. En las búsquedas de los navegadores hay variantes de la palabra *BioPlay* vendiendo productos totalmente diferentes, quizá eso dificulte que la gente encuentre nuestro videojuego. De hecho, el propio navegador de Google al escribir *BioPlay*, te corrige para que escribas otras palabras.

5.2.2 Fase 2: Ubersuggest

A continuación, usaremos la página web Ubersuggest, una página web muy útil que nos va a aportar un análisis de dominios, seguimiento de rankings y sugerencias para mejorar la visibilidad de nuestro sitio web en los motores de búsqueda como Google.

Podemos observarlo en las siguientes imágenes, teniendo en cuenta las tres primeras palabras clave escogidas anteriormente como ejemplo. La aplicación ofrece una versión gratuita en la que este proceso tan solo se puede hacer un total de 3 veces al día.

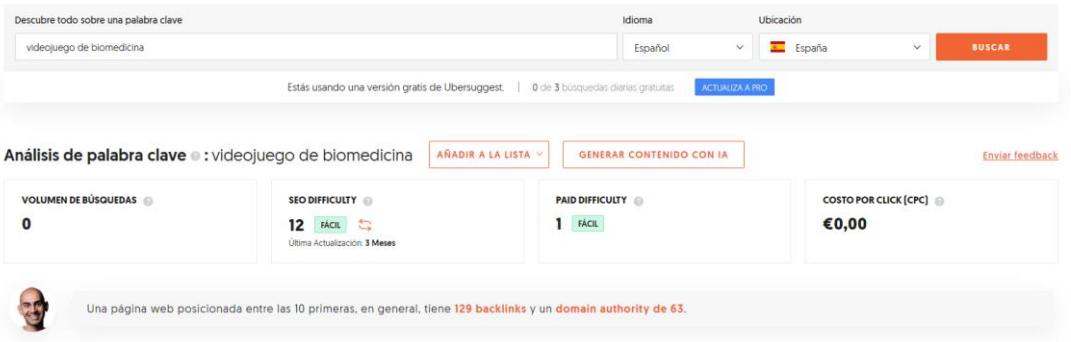
- BioPlay (navegacional)



Cuadro 70. Ubersuggest de la primera palabra clave.

Fuente: Ubersuggest[9].

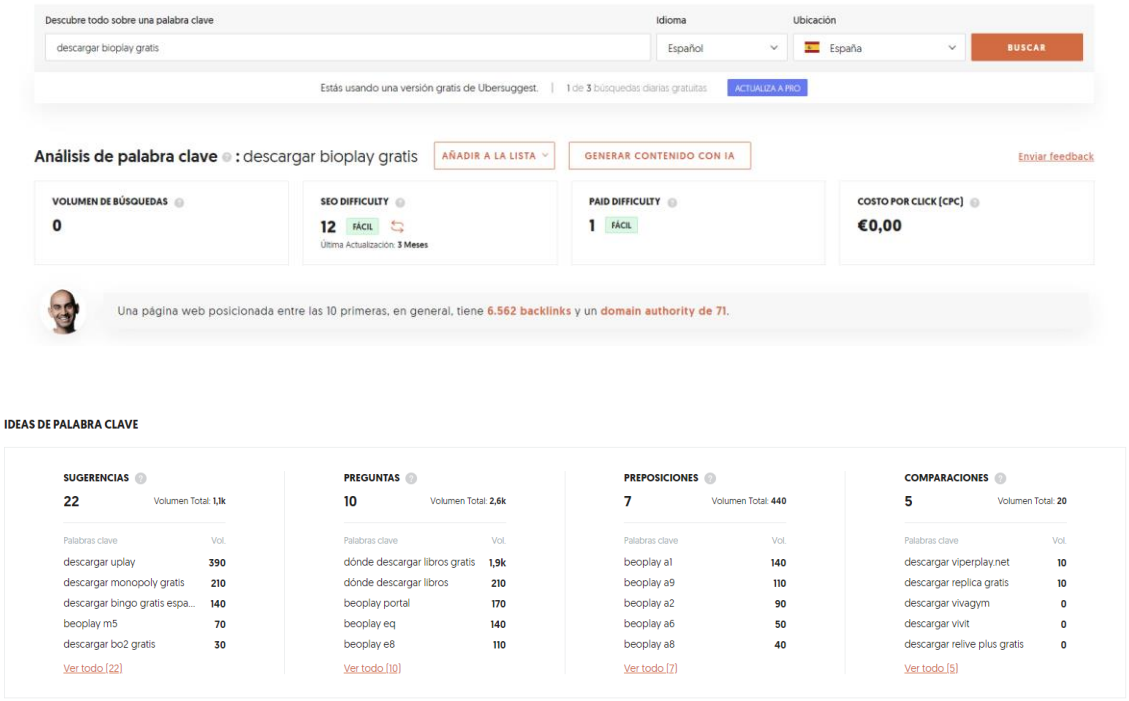
- Videojuego de biomedicina (comercial)



Cuadro 71. Ubersuggest de la segunda palabra clave.

Fuente: Ubersuggest[9].

- Descargar BioPlay gratis (transaccional)



Cuadro 72. Ubersuggest de la tercera palabra clave.

Fuente: Ubersuggest[9].

Con esta página web se consiguen datos tan interesantes como el volumen de búsquedas, sugerencias relacionadas con nuestras palabras clave, comparaciones en los motores de búsqueda y mucho más.

Como conclusión personal, esta herramienta me ha parecido bastante curiosa y útil, es una lástima que sea de pago para poder hacer más uso de ella que simplemente 3 búsquedas al día. En definitiva, este enfoque en las búsquedas me parece un buen punto de vista para posicionar nuestro servicio en el mercado.

5.3 Uso de redes sociales

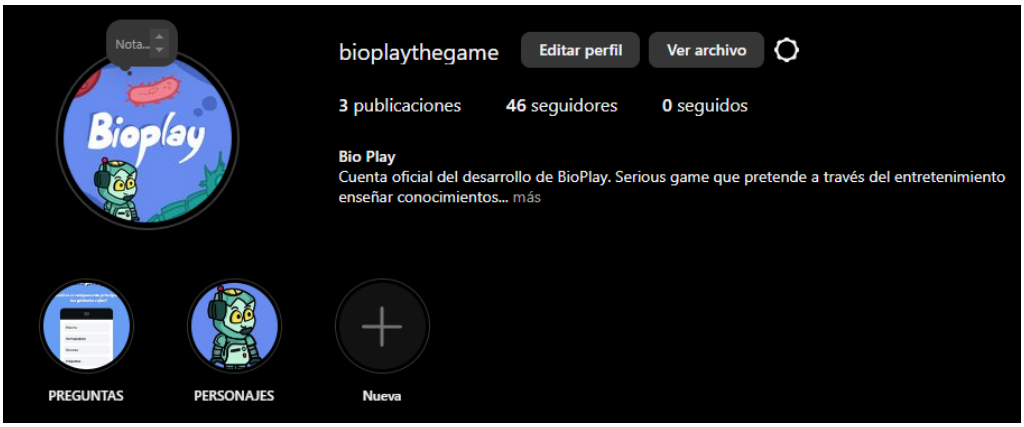
El uso de las redes sociales hoy en día es inevitable, y no se puede subestimar la importancia que tienen para la promoción de un producto. Las plataformas como Facebook, Instagram, Twitter, LinkedIn y TikTok, entre otras, ofrecen a las empresas la oportunidad de llegar a un público amplio y diverso de manera rápida y efectiva.

Además, las redes sociales permiten a las marcas interactuar directamente con sus clientes, recibir retroalimentación en tiempo real y ajustar sus estrategias de marketing en consecuencia.

En particular, he escogido hacer uso de Instagram por los siguientes motivos:

- **Plataforma muy visual**: perfecta para mostrar los diseños de nuestro videojuego. Publicar imágenes atractivas y videos cortos puede captar la atención de los usuarios y generar interés rápidamente.
- **Alta tasa de participación**: Instagram tiene una de las tasas de participación más altas entre las redes sociales. Los usuarios de Instagram están más dispuestos a interactuar con el contenido.
- **Historias de Instagram**: Las llamadas "historias" permiten compartir contenido efímero que desaparece en 24 horas, ideal para mostrar avances, actualizaciones, y contenido.
- **Uso de Hashtags**: Los hashtags son herramientas poderosas en Instagram para aumentar la visibilidad de nuestras publicaciones. Utilizando hashtags populares y específicos del nicho de videojuegos podemos llegar a una audiencia más amplia que busca contenido relacionado.

La cuenta usada se puede encontrar con el siguiente nombre @bioplaythegame.



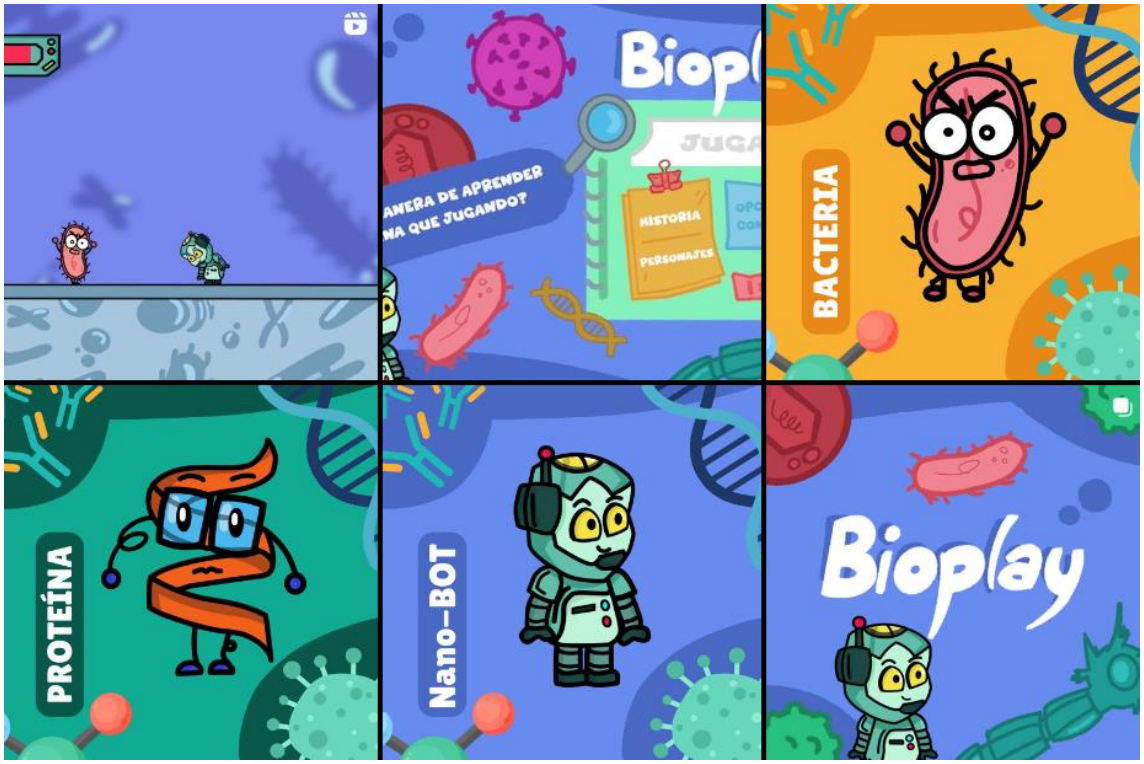
Cuadro 73. Cuenta de Instagram de bioplaythegame.

Fuente: Instagram [10].

5.3.1 Contenido publicado en Instagram

Una vez explicados los motivos esenciales de usar las redes sociales y en concreto Instagram, se mostrarán algunos ejemplos de las publicaciones que han dado pie a la participación de los usuarios.

- Publicaciones con contenido explícito del videojuego, muestro a los usuarios los personajes, algunos detalles del desarrollo, etc.



Cuadro 74. Cuenta de Instagram de bioplaythegame ejemplos.

Fuente: Instagram [10].

Al abrir la cuenta, la primera publicación da la bienvenida a los usuarios con el logo del videojuego y una descripción detallada de lo que pueden esperar en la cuenta, incluyendo la iniciativa BioPlay. La respuesta de los usuarios es entusiasta, marcando un excelente comienzo para generar expectación:

Cuadro 75. Ejemplo respuesta usuarios.

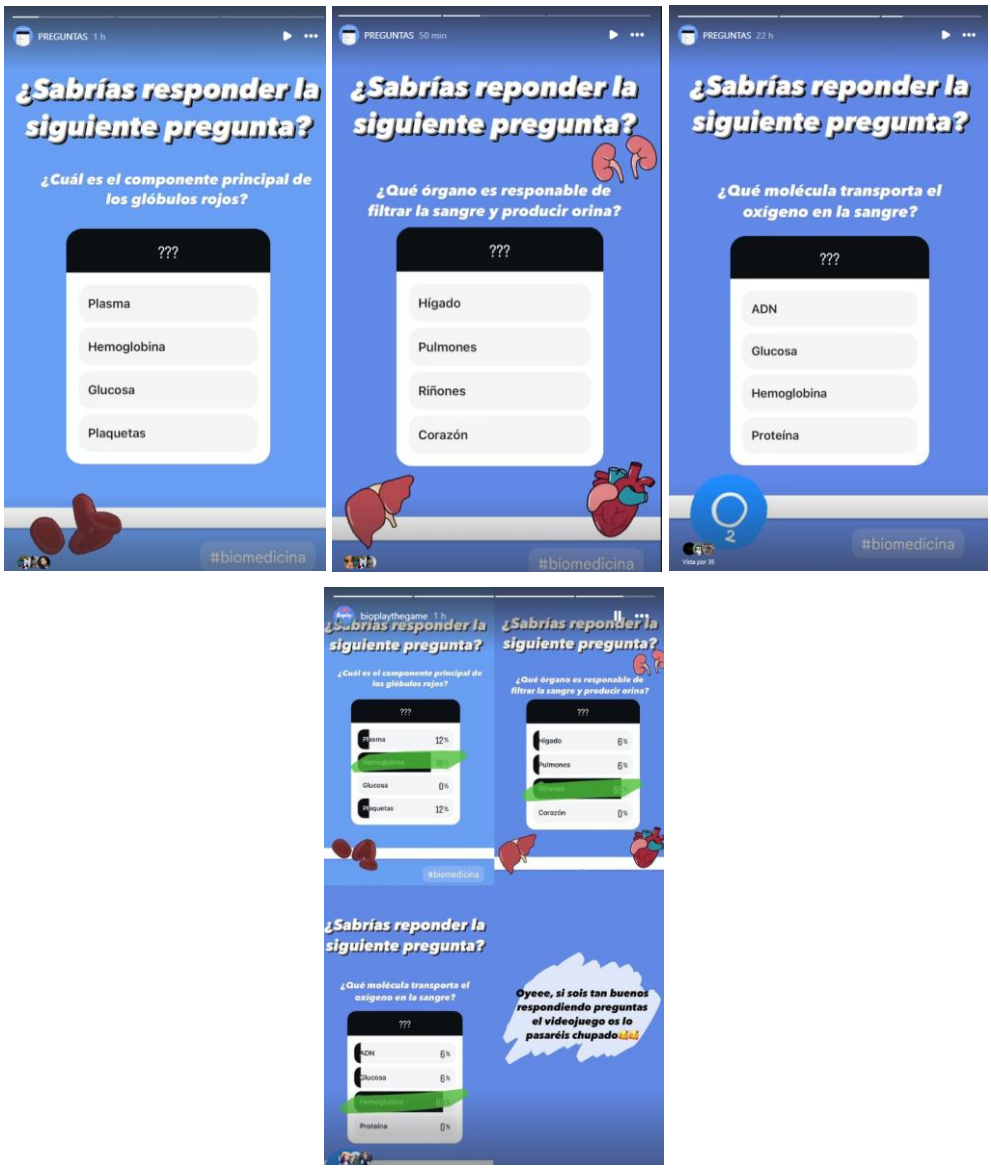
Fuente: Instagram [10].

Periódicamente, se han ido publicando los personajes que forman parte de la aventura BioPlay. De este modo, aunque nuestro videojuego aún no ha sido lanzado al mercado, los usuarios comienzan a familiarizarse mejor con ellos y a conocer más de cerca el universo del juego.

Lo mismo ocurre con los videos publicados que muestran los movimientos e interacciones de los personajes dentro del juego. Estos videos no solo permiten a los usuarios ver en acción a los personajes, sino que también les dan un adelanto de las mecánicas de juego y el estilo visual.

Además, los videos generan un mayor interés y expectativa, ayudando a crear una comunidad más comprometida y ansiosa por experimentar el juego completo.

- Publicaciones con la intención de fomentar el aprendizaje en la biomedicina. Por ejemplo, un listado de preguntas para que contesten los usuarios y después mostrándoles los resultados. Así se genera competitividad e interés.



Cuadro 76. Historias de Instagram.

Fuente: Instagram [10].

Publicar contenido con la intención de fomentar el aprendizaje en biomedicina es una estrategia eficaz para captar la atención de los usuarios de BioPlay. Por ejemplo, un listado de preguntas sobre temas biológicos y médicos invita a los usuarios a poner a prueba sus conocimientos, creando un ambiente de competitividad sana.

Al mostrar los resultados después de que los usuarios respondan, se genera no solo interés, sino también una oportunidad para el aprendizaje y la discusión. Esto no solo aumenta el compromiso de la comunidad, sino que también refuerza el propósito educativo del juego, haciendo que los usuarios se sientan más conectados y motivados a seguir participando.

Capítulo 6. Conclusiones y propuesta de trabajo futuro

A lo largo de este trabajo se ha logrado plasmar con éxito el desarrollo de *Bioplay*, un videojuego de plataformas con un enfoque didáctico orientado a la enseñanza de conceptos de biomedicina.

El proceso de desarrollo, desde la concepción de la idea hasta la implementación final, ha estado guiado por los objetivos planteados al inicio de esta memoria, los cuales se han alcanzado satisfactoriamente. Este proyecto deja un trabajo redondo, integrando diferentes disciplinas que van más allá del desarrollo del videojuego en sí como la programación y el diseño. Se han aportado técnicas de marketing digital, lo que ha permitido crear algunas estrategias y factores a tener en cuenta para *Bioplay*,

Se espera que esta idea contribuya positivamente al mundo de los videojuegos, ofreciendo una herramienta atractiva y accesible para facilitar el aprendizaje a aquellos que encuentran dificultades en métodos tradicionales. La combinación de entretenimiento y educación tiene el potencial de motivar a más personas a aprender de manera lúdica y efectiva, especialmente a aquellas personas que están interesadas en aprender de manera autodidacta conceptos que, hasta ahora, podían parecerles inaccesibles o complejos.

Por otro lado, las técnicas de marketing implementadas en este proyecto reflejan un área clave con margen de mejora para el videojuego. Sin embargo, el tiempo limitado para entregar este trabajo ha supuesto restricciones significativas. En condiciones ideales, el desarrollo de un videojuego puede llevar años, permitiendo a los creadores refinar cada aspecto, desde la jugabilidad hasta la promoción. Esta diferencia de tiempo remarca la dificultad y el desafío de desarrollar y comercializar un videojuego en un marco temporal reducido. Es por ello por lo que el juego no saldrá a la venta hasta que se considere oportuno.

Como reflexión personal, para futuros proyectos me doy cuenta de la importancia de ser más consciente de los límites y de hasta dónde puedo abarcar. Desarrollar un videojuego es un proceso complejo que requiere la colaboración de programadores, diseñadores y expertos en diversas áreas. Además, gestionar la promoción del juego a través de una cuenta de Instagram es una tarea que consume mucho tiempo y energía. Intentar manejar todo esto por mi cuenta ha sido una experiencia enriquecedora, pero también me ha demostrado lo exigente que puede ser. Para próximas iniciativas, sería más prudente en buscar trabajar en equipo, lo que permitiría una gestión más equilibrada y menos presión sobre una sola persona.

Este proyecto también me ha hecho reflexionar sobre la necesidad de apoyar más iniciativas que combinen educación y entretenimiento. Creo firmemente que los videojuegos educativos como Bioplay pueden desempeñar un papel crucial en el futuro de la educación, haciendo que el aprendizaje sea más accesible y atractivo para las nuevas generaciones.

Finalmente, aunque este es solo el comienzo de lo que espero que sea una larga trayectoria en el desarrollo de videojuegos educativos, estoy satisfecho con lo que se ha logrado hasta ahora. Espero que Bioplay no solo sirva como una herramienta útil para el aprendizaje, sino que también inspire a otros a explorar el vasto potencial de los videojuegos como medios de educación y desarrollo personal. Con esta memoria, cierro un capítulo importante en mi formación, pero también dejo abierta la puerta a futuras oportunidades y desafíos en este campo emocionante y en constante evolución. Invito a otros estudiantes y profesionales a unirse a esta misión de integrar la tecnología y la educación de manera creativa y efectiva.

Capítulo 7. Bibliografía

[1] ScienceDaily. (2017, June 22). *Video games can improve cognitive abilities*. ScienceDaily. Obtenido de: https://www.sciencedaily.com/releases/2017/06/170622103824.htm#:~:text=There%20is%20also%20evidence%20that,a%20video%20game%20training%20program.

[2] Edutopia. (n.d.). *4 ways to use games for learning*. Edutopia. Obtenido de: https://www.edutopia.org/video/4-ways-use-games-learning/#:~:text=Research%20shows%20that%20playing%20games,motivate%20students%20to%20take%20risks.

[3] Steam. (n.d.). *Zoombinis*. Steam. Obtenido de: https://store.steampowered.com/app/397430/Zoombinis/?l=spanish

[4] JumpStart Wiki. (n.d.). *JumpStart Adventures 3rd Grade: Mystery Mountain*. JumpStart Wiki. Obtenido de: https://jstart.fandom.com/wiki/JumpStart_Adventures_3rd_Grade:_Mystery_Mountain

[5] Steam. (n.d.). *Plague Inc: Evolved*. Steam. Obtenido de: https://store.steampowered.com/app/246620/Plague_Inc_Evolved/?l=latam

[6] Wikimedia Commons. (n.d.). *Unity Technologies logo*. Wikimedia Commons. Obtenido de: https://commons.wikimedia.org/wiki/File:Unity_Technologies_logo.svg

[7] Unity Technologies. (n.d.). *Learning the Interface*. Unity Documentation. Obtenido de: https://docs.unity3d.com/es/530/Manual/LearningtheInterface.html

[8] NotiLinux. (n.d.). *Krita: Una impresionante herramienta de diseño gráfico y arte digital para Linux*. NotiLinux. Obtenido de: https://notilinux.com/krita-una-impresionante-herramienta-de-diseno-grafico-y-arte-digital-para-linux/

[9] Patel, N. (n.d.). *Neil Patel's Ubersuggest*. Neil Patel. Obtenido de: https://app.neilpatel.com/es/login?next=https%3A%2F%2Fapp.neilpatel.com%2Fes%2Fubersuggest%2Foverview

[10] Instagram. (n.d.). *Bioplay the Game*. Instagram. Obtenido de: https://www.instagram.com/bioplaythegame/?hl=es

[11] [@bioplaythegame]. (2023, April 26). *[Image accompanying post about Bioplay]* [Instagram photo]. Instagram. Obtenido de: https://www.instagram.com/p/C-qMLZZRsrj/?hl=es

[12] [@bioplaythegame]. (2023, April 28). *[Image accompanying post about Bioplay]* [Instagram photo]. Instagram. Obtenido de: https://www.instagram.com/p/C-czWkvtyra/?hl=es

[13] [@bioplaythegame]. (2023, May 1). *[Image accompanying post about Bioplay]* [Instagram photo]. Instagram. Obtenido de: https://www.instagram.com/p/C-TL71atuO-/?hl=es

[14] [@bioplaythegame]. (2023, April 20). *[Image accompanying post about Bioplay]* [Instagram photo]. Instagram. Obtenido de: https://www.instagram.com/p/C-GDJVGN8P_/?hl=es

[15] [@bioplaythegame]. (2023, April 18). *[Image accompanying post about Bioplay]* [Instagram photo]. Instagram. Obtenido de: https://www.instagram.com/p/C-A2NbytHKh/?hl=es

[16] [@bioplaythegame]. (2023, April 14). *[Image accompanying post about Bioplay]* [Instagram photo]. Instagram. Obtenido de: https://www.instagram.com/p/C9wsAFKtUpk/?hl=es&img_index=1

[17] Martra. (n.d.). *Cómo crear un sistema de diálogos para Unity*. Obtenido de: https://martra.uadla.com/como-crear-un-sistema-de-dialogos-para-unity/

[18] Unity Community. (n.d.). *Best 2D movement practices* [Forum post]. Obtenido de: https://discussions.unity.com/t/best-2d-movement-practices/845125

[19] Game Artist 86. (2021, June 15). *Simple player movement in Unity 2D*. Medium. Obtenido de: https://gameartist86.medium.com/simple-player-movement-in-unity-2d-c9bfd46f8837

[20] Unity Community. (n.d.). *Health & damage in a 2D game* [Forum post]. Obtenido de: https://discussions.unity.com/t/health-damage-in-a-2d-game/860851

[21] Unity Community. (n.d.). *How to make player take damage* [Forum post]. Obtenido de: https://discussions.unity.com/t/how-to-make-player-take-damage/910951

[22] Unity Community. (n.d.). *How to make player take damage* [Forum post]. Obtenido de: https://discussions.unity.com/t/how-to-make-player-take-damage/910953