Grado en Ingeniería de
Tecnologías de Telecomunicación

Trabajo Fin de Grado

“Diseño y desarrollo de un videojuego mediante Unity”

Jorge Vázquez García
Cuenca, Junio de 2017

Campus Universitario | 16071 Cuenca | Telf.: (+34) 969 179 100 (+34) 902 204 100 | Fax.: (+34) 969 179 119
politecnicacuenca.uclm.es | politecnica.cu@uclm.es

Agradecimientos

En primer lugar, quiero agradecerles a todas esas personas que han estado presentes dándome ánimo y fuerzas durante el desarrollo de este proyecto. En mayor medida a mis padres y mi hermana, los cuales han aportado plena confianza en mí y me han dado la oportunidad de estudiar el grado.

También darle las gracias a mis amigos y amigas, tanto de mi ciudad natal como los que he conocido aquí en Cuenca, su apoyo y haber estado tanto en los momentos buenos como en los malos. En especial a mis amigos Víctor Rueda y Jose Alcaraz y a mi amiga Sandra González por su ayuda en muchos momentos durante estos cuatro años.

Este trabajo se lo dedico en sí a todos ellos.

Como último, agradecer a Jose Manuel Pastor, mi tutor, por orientarme, apoyarme en este trabajo Fin de Grado y ayudarme a embarcarme en este proyecto.

Resumen

El objetivo de este Trabajo Fin de Grado es crear un videojuego 3D donde se desarrollen diferentes minijuegos. Se espera que resulte un videojuego entretenido y divertido, lo cual permite que el jugador lo use en su tiempo libre.

Con el fin de lograr ese objetivo se ha desarrollado un juego interactivo que sirva como pasatiempo y sea amigable para el usuario, consiguiendo así que el tiempo invertido estimule los reflejos.

El videojuego constará de una serie de minijuegos en los que a través de un escenario puede navegar e introducirse en la piel de un personaje peculiar, un alienígena. Hay cuatro minijuegos, un juego de tiros mezclado con el antiguo juego de los marcianitos, un Pacman infinito que puede matar a los enemigos y esquiva obstáculos, un pilla-pilla en el que hay que atrapar a un ladrón y un escondite inglés con un personaje amigable.

Abstract

The aim of this final degree project is create a 3D videogame where I develope differents minigames. I hope that this videogame will be entertained and funny, so the player can use it in his spare time.

In order to achieve this goal, It generates an interactive application, which is a friendly pastime to the user, in this way the spent time stimulates reflexes.

The video game will consist of a series of mini-games in which through a stage can navigate and enter the skin of a peculiar personage, an alien. There are four mini-games, a shooter mixed with the old Martian game, an infinite Pacman that can kill enemies and dodge obstacles, a game based in “pilla-pilla” game in which you have to catch a thief and an English hiding place with a friendly character.

Tabla de contenido

INDICE DE FIGURAS

I MEMORIA

CAPITULO 1: INTRODUCCIÓN

MOTIVACIÓN

Al cursar la asignatura “Animación Digital” vi la posibilidad de ampliar mis conocimientos sobre animaciones, en este caso empleado en animaciones sobre cualquier tipo de objeto, es decir, personajes. Es por eso, que el proyecto consta de un personaje al que he tenido que animar para que realice una serie de movimientos.

Debido al auge actual de los videojuegos, era interesante realizarlo además para las dos plataformas más usadas en el mundo: Ordenador y Android. Android me parece una plataforma muy interesante, ya que actualmente están potenciando la figura del programador Android, el cual desarrolla aplicaciones con diversas temáticas y/o utilidades y las mayores descargas son videojuegos hechos para esta plataforma, teniendo incluso grandes empresas desarrollando juegos, intentando introducirse en esta plataforma.

La motivación segunda de este proyecto era intentar unificar de alguna manera conceptos varios del grado. En un videojuego hay que saber conceptos sobre video, en cuanto a resolución y su adaptación a distintas resoluciones, para así poder jugar con una gran calidad. También hay que saber conceptos de audio, ya que de alguna forma, tiene que tener una ambientación sonora, en el proyecto se usa sobre todo el uso de filtros para una ambientación sonora única. Y obviamente, hay que saber conceptos de programación orientada a objetos, que es una habilidad que me gusta.

La última motivación es mi idea o concepto de un ingeniero. Creo que la figura de un ingeniero consiste en crear proyectos y trabajos que puedan ayudar o hacer algo que pueda repercutir de alguna forma tecnológica al mundo en el que vive. Habrá quien crea que un videojuego no aporta nada a la sociedad, pero yo creo todo lo contrario, ya que de primera, será una forma de ocio, y un pasatiempo único.

Cabe mencionar que el proyecto, para mí, es una carta de presentación de los valores que creo que tengo como profesional. Es un proyecto multidisciplinar, que conlleva mucho trabajo y en el que muestro mis habilidades creativas.

OBJETIVOS

El proyecto tiene como objetivo el desarrollo de cuatro minijuegos, con tratamiento gráfico en tres dimensiones. Se ha optado por minijuegos basados en reflejos, concretamente juegos hechos en épocas anteriores, como son el Space Invaders o el Pacman variándolos y mejorándolos, o juegos que jugamos de pequeños como son el “pilla-pilla” o el escondite inglés.

La implementación de estos minijuegos en dos plataformas nos obliga a usar diferentes formas de la interactividad entre usuario y juego. Aunque en sí, el proyecto lo que conlleva es el uso de diferentes algoritmos y filosofías para hacer los distintos minijuegos. Se quiere conseguir además de una aplicación visualmente atractiva, que incite a los diversos usuarios a hacer uso de ella.

Para alcanzar todos nuestros objetivos necesitaremos:

- Encontrar y estudiar las librerías gráficas adecuadas.
- Generar objetos en 2D y 3D.
- Desarrollar los algoritmos adecuados para todo el proceso.
- Construir una aplicación intuitiva y adecuada para el usuario.

CAPITULO 2: ESTADO DEL ARTE

Historia de los videojuegos

Esta información son las conclusiones extraídas en la Historia de los videojuegos [6].

En la actualidad los videojuegos son la entrada de niños y jóvenes en el universo de las TIC. Mediante el videojuego los niños adquieren capacidades, desarrollan habilidades diversas que pueden ir desde los reflejos hasta la creatividad.

El impacto que los videojuegos han tenido en la sociedad se ha desarrollado en los últimos treinta años, es un fenómeno que todavía no se ha estudiado en profundidad por los investigadores sociales. Los videojuegos pueden ser considerados como una forma de expresión artística contemporánea, aunque la sociedad todavía no es capaz de asimilarlas y "hacerlas suyas". Los videojuegos son un conjunto de varias artes, como son los gráficos, la música, la historia o trama, y sobre todo, el aspecto lúdico y la jugabilidad.

Durante bastante tiempo ha sido complicado señalar cual fue el primer videojuego, principalmente debido a las múltiples definiciones que de este se han ido estableciendo, pero se puede considerar como primer videojuego el OXO. El juego era una versión computerizada de un tres en raya, en el que un jugador se enfrentaba o jugaba contra la máquina.

Aunque ATARI fue considerado el padre del videojuego moderno, desarrollando el "Space World" con una máquina bastante peculiar. Pero la repercusión y por lo que fue conocido fue por crear el videojuego "Pong" en 1972. Tenía un objetivo claro, rebotar una pelota y en caso de no ser capaz de devolver conseguir un punto.

"Space Invaders" fue la piedra angular del videojuego como industria. Durante los años siguientes se implantaron numerosos avances técnicos en los videojuegos, apareciendo estos avances en los salones recreativos con juegos como el "Space Invaders" o "Asteroids".

Estos juegos tenían la misma temática, disparar a todo lo que encontraban a su paso. Fue entonces cuando apareció "Pac Man". Fue tal su repercusión que fue el primer videojuego que se convertía en una marca, apareciendo en todo tipo de productos.

Los videojuegos arcade invadieron todo tipo de lugares, desde bares y salones recreativos hasta funerarias y lugares insospechados. Había multitud de visiones, muchas de ellas eran críticas hablando de la repercusión en los jóvenes, pues se pensaba que enloquecerían y crearía enfermedades, además de que se gastaba mucho dinero en ellas.

El negocio asociado a esta nueva industria alcanzó en poco tiempo grandes cotas. Sin embargo, en 1983 comenzó la que se ha dado por llamar crisis del videojuego, la cual afectó principalmente a EEUU y Canadá. En el resto del mundo se produjo una polarización dentro de los sistemas de videojuegos, apostando por las consolas domésticas. El primero en lanzar fue Nintendo, la muy conocida "NES" mientras que Europa se decantaba por los microordenadores como el commodore o el Spectrum.

A principios de los 90 las videoconsolas dieron un importante salto técnico gracias a la competición de la llamada “generación de 16 bits”. Esta generación supuso un importante aumento de jugadores y la introducción de tecnologías como el CD-ROM, además de una importante evolución dentro de los diferentes géneros de videojuegos, principalmente gracias a las nuevas capacidades técnicas. Mientras tanto, diversas compañías habían comenzado a trabajar en videojuegos con entornos tridimensionales, principalmente en el campo de los PC.

Rápidamente los videojuegos en 3D fueron ocupando un importante lugar en el mercado, principalmente gracias a la llamada "generación de 32 bits" en las videoconsolas: Sony PlayStation y Sega Saturn (principalmente en Japón); y la “generación de 64 bits” en las videoconsolas: Nintendo 64 y Atari Jaguar. En cuanto a los ordenadores, se crearon las aceleradoras 3D.

Por su parte los videojuegos portátiles, producto de las nuevas tecnologías más poderosas, comenzaron su verdadero auge, uniéndose a la Game Boy máquinas como la Game Gear (Sega), Linx (Atari) o la Neo Geo Pocket (SNK), aunque ninguna pudo hacerle frente a la popularidad de la Game Boy, siendo esta y sus descendientes (Game Boy Pocket, Game Boy Color, Game Boy Advance, Game Boy Advance SP) las dominadoras del mercado.

Hacia finales de la década la consola más popular era la PlayStation con juegos como Final Fantasy VII (Square-Enix), Resident Evil (Capcom), Winning Eleven 4 (Konami), y Gran Turismo.

En PC eran muy populares los FPS (juegos de acción en primera persona) como Half-Life (Valve), y los RTS (juegos de estrategia en tiempo real) como Command & Conquer (Westwood) o Starcraft (Blizzard). Además, las conexiones entre ordenadores mediante internet facilitaron el juego multijugador, convirtiéndolo en la opción predilecta de muchos jugadores, y fueron las responsables del nacimiento de los MMORPG (juegos de rol multijugador online) como Ultima Online (Origin). Finalmente en 1998 apareció en Japón la Dreamcast (Sega) y daría comienzo a la “generación de los 128 bits”.

En el 2000 Sony lanzó la anticipada PlayStation 2 y Sega lanzó otra consola con las mismas características técnicas de la Dreamcast, nada más que venía con un monitor de 14 pulgadas, un teclado, altavoces y los mismos mandos llamados Dreamcast Drivers 2000 Series CX-1.

El ordenador personal PC es la plataforma más cara de juegos pero también la que permite mayor flexibilidad. Esta flexibilidad proviene del hecho de poder añadir al ordenador componentes que se pueden mejorar constantemente, como son tarjetas gráficas o de sonido y accesorios como volantes, pedales y mandos, etc. Otra ventaja es la posibilidad de actualizar los juegos con parches oficiales o con nuevos añadidos realizados por la compañía que creó el juego o por otros usuarios.

Motor de Videojuegos Unity

Esta información son las conclusiones extraídas en la página oficial de Unity [1].

Un motor de videojuegos es un conjunto de herramientas que realizan cálculos geométricos y físicos utilizados en los videojuegos. Este conjunto de utilidades representa un simulador ágil en tiempo real que reproduce las características de los mundos imaginarios en los que transcurren los videojuegos. El objetivo es permitir al equipo de desarrollo de un videojuego concentrarse sobre el contenido del juego y no sobre la resolución de problemas informáticos.

El fin que persiguen los motores de juego es que una vez incorporadas todas las funciones del motor, usuarios sin conocimientos de programación pueden manejar con facilidad el motor.

Puesto que las funcionalidades proporcionadas por un motor son limitadas, estos con frecuencia son dedicados a un tipo específico de juego. La manipulación del motor se hace generalmente a través de un lenguaje de script o por la interfaz.

Un motor de video juegos debe ofrece como mínimo las siguientes utilidades:

- Un motor 3D que permite la creación y visualización de un juego en un universo 3D.
- Un motor audio que permite la integración de elementos sonoros y música en el juego.
- Un motor físico que permita gestionar los comportamientos físicos de los objetos en un universo 3D, como la gravedad por ejemplo.
- Herramientas de gestión de red para añadir por ejemplo un componente multi-jugadores en red.

Unity presenta varias ventajas que hacen que sea uno de los motores de videojuego más potentes del mercado. En los siguientes párrafos se van a ir citando todas estas ventajas:

- Permite la importación de numerosos formatos 3D como 3ds Max, Maya, Cinema 4D, Cheetah3D y Softimage, Blender, Modo, ZBrush, FBX o recursos variados tales como texturas Photoshop, PNG, TIFF, audios y videos. Estos recursos se optimizan posteriormente mediante filtros.
- Es compatible con las API graficas de Direct3D, OpenGL y Wii. Además de ser compatible con QuickTime y utilizar internamente el formato Ogg Vorbis
- En Unity, el juego se construye mediante el editor y un lenguaje de scripts por lo cual el usuario no tiene que ser un experto en programación para usarlo. En efecto, este software tiene la particularidad de incluir la herramienta de desarrollo MonoDevelop con la que se pueden crear scripts en JavaScript, C# y un dialecto de Python llamado Boo con los que extender la funcionalidad del editor, utilizando las API que provee y la cual encontramos documentada junto a tutoriales y recursos en su web oficial.
- La estructura de los juegos creados por Unity viene definida mediante escenas que representan alguna parte del juego.
- Incluye un editor de terrenos que permite la creación de estos partiendo de cero. Este editor permite esculpir la geometría del terreno, su texturización y la inclusión de elementos 3D importados desde aplicaciones 3D o ya predefinidos en Unity.
- Si no se quiere modelar en 3D y se necesitan recursos para un videojuego, en la propia aplicación se puede acceder al Asset Store donde existe multitud de recursos gratuitos

y de pago. Incluso se puede extender la herramienta mediante plugins que se obtienen en esta misma tienda.

- En cuanto a los usuarios que no tienen ninguna noción de programación existen plugins como Playmaker que permiten "programar con cajitas" mediante máquinas de estados, de una forma visual. La utilización de estos plugins supone un coste añadido
- Dispone de una interfaz de desarrollo muy bien definida e intuitiva que permite crear rápidamente min-juegos.
- Existe en varias versiones en función de los módulos elegido, la versión más simple destinada a los amateurs es gratuita.
- Tal como se ha especificado antes es multiplataforma por lo cual permite la creación de juegos compatibles con distintas consolas (Para la mayoría de las plataformas citadas, se requiere una licencia adicional):
    - Microsoft Windows o Mac OS X ejecutable
    - Linux
    - En la web (a través del plugin Unity Web Player para Internet Explorer, Firefox, Safari, Mozilla, Netscape, Opera, Google Chrome y Camino) en Windows y OS X
    - En Mac OS X Dashboard widget.
    - Para Nintendo Wii
    - iPhone / iPad
    - Google Android
    - Google Native Client
    - Microsoft Xbox 360
    - Adobe Flash
    - Sony PlayStation 3
    - BlackBerry PlayBook

En definitiva la creación de videojuegos es mucho más sencilla y rápida con esta aplicación por lo cual el número de usuarios está en aumento constante.

Fundamentos de Unity

Esta información son las conclusiones extraídas en la página oficial de Unity [1].

GameObjects, Assets, Scene.

Unity se fundamenta en unos objetos llamados GameObjects. La mayoría de los elementos del juego, desde los scripts hasta elementos gráficos suelen depender de ellos.

Una Scene es un entorno donde se desarrolla una parte del juego, incluido dentro de este concepto los menús. Dentro de cada Scene se encuentran los gameObjects, que representarán a los diferentes elementos, desde personajes hasta la cámara. Cada vez que cambiemos de Scene cambiarán todos los gameObjects.

Un asset es una representación de cualquier item que puede ser utilizado en el juego o proyecto creado. Un asset podría ser un archivo creado fuera de Unity, tal como un modelo 3D, un archivo de audio, una imagen, o cualquiera de los otros tipos de archivos que Unity soporta. También hay otros tipos de asset que pueden ser creados dentro de Unity, tal como

un Animator Controller, un Audio Mixer o una Render Texture. Los scripts creados también se considerarían assets.

Los componentes

Como se ha mencionado cualquier gameObject puede llevar scripts que dependen de ellos. Los scripts son comportamientos que se dan a los gameObject, estos scripts Unity los llama componentes. Hay muchos tipos de componentes. Los componentes más utilizados para comportamientos de física son el componenente RigidBody, y el componente Collider.

Un collider es la propiedad de choque. Sin esta propiedad los personajes atravesarían cualquier superficie u objeto que esté presente en la escena y no habría interacción entre el universo que se diseñe y el personaje.

Dado que hablamos de videojuegos en 2D o 3D (El juego a desarrollar es en 3D), los vectores son muy importantes, una propiedad también muy necesaria y que todos los gameObjects en el momento de ser creados y puestos en el juego tienen es el componente Transform, el cual se encarga de la translación, rotación y escala de los gameObjects.

Los scripts que se crean por necesidad del desarrollador serán también componentes, y en el caso de necesitar que estos scripts dependan de otros componentes podrán llamarlos y ejecutar diversas propiedades de los componentes dependientes.

Inteligencia artificial

Un videojuego consta de muchas partes: un personaje, una cámara, un entorno o escena donde se desarrolle la acción y enemigos. Cuando se diseña y se desarrolla un videojuego se está dotando a un programa de cierta inteligencia como se le puede dar a un enemigo o incluso un gameObject vacío de renderización que solo se encarga de controlar el entorno y ver que todo sigue el ritmo y la historia del juego. Una inteligencia también puede controlar lo que el jugador puede o no hacer mediante las reglas que impone el desarrollador.

Commúnmente la inteligencia se le dota al enemigo. Dependiendo del tipo de juego y de las características que tenga este enemigo que se crea tendrá unos ciertos comportamientos, es decir, scripts que darán vida y sensación de inteligencia.

Por lo general, a la hora de desarrollar un videojuego, el diseñador y el desarrollador deben pensar y ser más inteligentes que el propio jugador cuando lo pruebe o ejecute. Aunque es cierto que siempre hay que dejarle alguna forma de superar el juego.

Durante años se ha visto, dependiendo del tipo de juego diferentes tipos de IA, como puede ser los fantasmas del PAC MAN, soldados en un FPS, o jugadores de un equipo en un juego de deportes.

Luces y cámaras.

Otra de las partes fundamentales de los videojuegos es el tipo de luz y la cámara por la que se enseña la acción al jugador.

El tipo de luz debe depender de la intención que se tenga en el juego, obviamente en un juego de survival terror, la luz será la mínima para que permita una sensación de oscuridad y pueda dar sustos.

El tipo de cámara también es importante, un juego en primera persona es un juego donde la cámara está colocada en la posición de la cabeza, simulando los ojos, es decir, permitiría la inmersividad del jugador. En tercera persona el jugador será un espectador de la acción. Es importante también pensar en la colocación de la cámara no solo como inmersión del jugador sino también a la dificultad que le puede añadir. Un juego en tercera persona adopta una posición donde el jugador puede tener todo bajo control. Mientras que con las primera persona se controla solo lo que vea el personaje.

El sistema de animación de Unity.

El sistema de Animación de Unity permite crear unas hermosas animaciones de personajes con Skinning. El Sistema de Animación soporta la fusión de animaciones (animation blending), mixing (mezcla), animaciones aditivas, sincronización walk cycle time, capas de animación, control sobre todos los aspectos de la reproducción de la animación (tiempo, velocidad, pesos de blending), mesh skinning con 1, 2, o 4 huesos por vertex (vértice) al igual que el soporte de rag-dolls, basados en físicas y animaciones procedimentales.

Crear un personaje animado implica dos cosas; *Moverlo* a través del mundo y *animarlo* como corresponde.

Audio en el videojuego.

Un juego estaría incompleto sin algún tipo de audio, ya sea la música de fondo o efectos de sonido. En la vida real, los sonidos son emitidos por objetos y escuchados por listeners. La manera en la que el sonido es percibido depende de una cantidad de factores. Un listener puede decir en qué dirección un sonido viene y también puede tener algún sentido de su distancia por su intensidad y calidad. Una fuente de audio que se mueve rápidamente (como una bomba que cae o un carro de policia que pasa) va a cambiar en el tono en la medida que se mueva debido al Efecto Doppler. Además, los alrededores van a afectar la manera en que el sonido se refleja, entonces una voz dentro de una cueva va a tener un echo, pero, la misma voz en el campo abierto no lo tendrá.

Para simular los efectos de la posición, Unity requiere que los sonidos sean originados de Audio Sources adjuntos a los objetos. Los sonidos emitidos luego son recogidos por un Audio Listener adjuntado a otro objeto, en la mayoría es la cámara principal. Unity puede simular los efectos de la distancia y la posición de una fuente del objeto listener y reproducirlos al usuario de acuerdo a esto. La velocidad relativa de los objetos fuente (source) y listener puede también ser utilizada para simular el Efecto Doppler para agregar realismo.

Lenguaje de programación: C#

Esta información son las conclusiones extraídas en la Wikipedia sobre C# [9].

C# es un "lenguaje de programación" orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA (ECMA-334) e ISO (ISO/IEC 23270). C# es uno de los lenguajes de programación diseñados para la infraestructura de lenguaje común.

Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET, similar al de Java, aunque incluye mejoras derivadas de otros lenguajes.

Aunque C# forma parte de la plataforma .NET, ésta es una API, mientras que C# es un lenguaje de programación independiente diseñado para generar programas sobre dicha plataforma. Ya existe un compilador implementado que provee el marco Mono - DotGNU, el cual genera programas para distintas plataformas como Windows, Unix, Android, iOS, Windows Phone, Mac OS y GNU/Linux.

Estudios de Videojuegos

(Esta información son las conclusiones extraídas en un vídeo del canal La Cueva de L Draven sobre organización de estudios de videojuegos, Youtube [3].)

Para el Diseño y Desarrollo de un videojuego en una empresa de videojuegos o estudio se distinguen miles de puestos de trabajo, desde el departamento de marketing hasta los distintos jefes.

Los videojuegos son tratados como diferentes proyectos en los que colabora cada departamento y cada persona especializada en un trabajo específico. Es por ello, que al final los elementos que más intervienen en la producción de videojuegos son tres áreas:

- El departamento de Arte; Aquí entran los artistas que se dedican a componer y crear los distintos objetos que tienen lugar en las diferentes escenas del videojuego.
- El departamento de Diseño: Son los encargados del diseño del videojuego en sí, es decir establecer cómo será el videojuego (Game Designer) y las diferentes escenas o niveles(Level Designer) para estar bien balanceado, ser un juego único y causar una gran sensación.
- Programadores: Son los que realizan la programación para que exista la interactividad, que es la propiedad más importante en un videojuego, tanto entre objetos como entre objeto y jugador y las características necesarias para hacer que el juego sea único.
- Sonido: Hoy en día es un departamento muy importante, dado que un videojuego con una banda sonora y sonidos únicos pueden ser más atractivo que uno que no tiene. El uso de música y sonidos ayuda al jugador a dar pistas al jugador de lo que tiene e incluso disfrutar y meterle en la historia de una forma más inmersiva.
- El departamento de Calidad; son los testers. Una vez hay una parte hecha del videojuego ellos se dedican a probarlo, una y otra vez, buscando fallos al videojuego para que éste hecho lo mejor posible.

En la siguiente figura se puede apreciar los distintos departamentos que tiene una empresa de videojuegos y que se centran en el diseño y desarrollo de un videojuego.

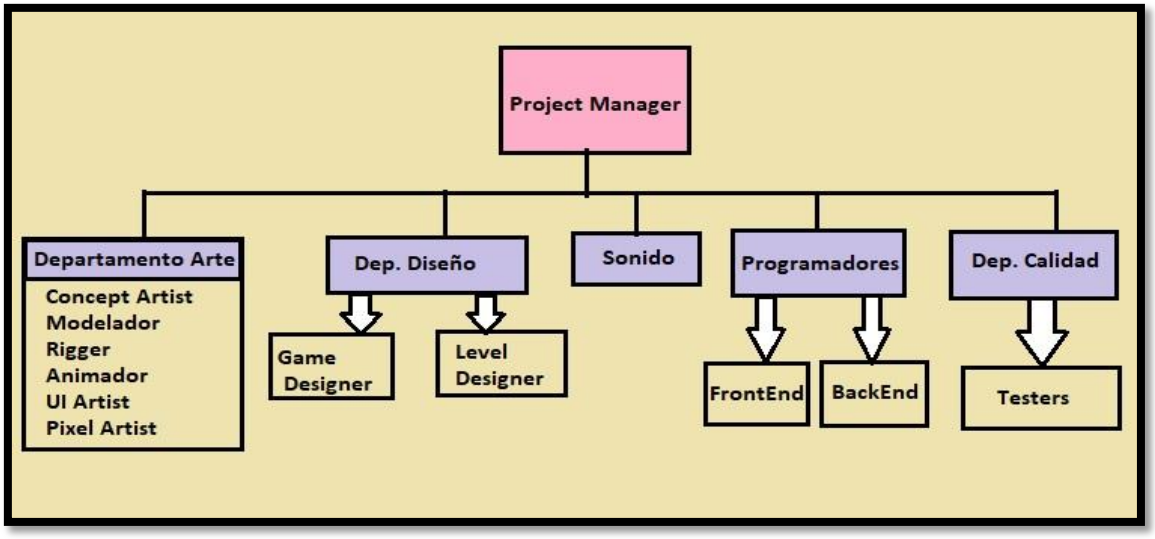


Figura 1: Departamentos implicados en un videojuego

Dependiendo del tipo de estudio estos departamentos están más separados y cada persona tiene una o varias tareas asignadas, así pues, hay estudios de una persona o un conjunto pequeño de personas llamado "Estudios Indie", en los que una persona se dedica a varias tareas o a todas.

Es por ello que los videojuegos Indie tardan mucho más en hacerse, ya que los estudios hacen videojuegos únicos y se tratan de pocas personas.

Unas herramientas que usan en todos los estudios de videojuegos son dos tipos de documentos:

- Documento de Diseño del Videojuego (GDD): En él se diseña las mecánicas del juego, se crean los puzles del juego. Define los comportamientos de personajes, ítems, etc… Es decir, cómo va a ser el juego en sí.
- Documento de Desarrollo del Videojuego: En él se especifica todo el proceso que han seguido los distintos departamentos encargados de la producción del videojuego para realizar el videojuego. Es una forma de hacer una guía y poder comparar lo diseñado con lo desarrollado. También se tratan las diferentes incidencias que se tienen al probarlo. Además es un documento abierto, ya que ante posibles mejoras y actualizaciones, se sigue completando.

En este proyecto se realizan los dos documentos. Así especifico tanto lo diseñado como lo desarrollado en el videojuego.

Para la realización de este proyecto Fin de Grado en el que estoy involucrado he tenido que realizar una serie de tareas que aquí se verán descritas.

1- Planteamiento: Se fijan los objetivos y el objetivo principal y secundario del videojuego.
2- Documentación: Es una de las partes más importantes del proyecto, pues es necesario entender todo lo que voy a hacer y qué cosas puedo realizar y qué cosas no puedo hacer.

3- Diseño: Realización del GDD. Se especificarán todas las mecánicas y comportamientos de los objetos y enemigos, además de los personajes.
4- Producto Mínimo con Valor: Es el videojuego hecho con cubos, esferas, etc. Los elementos primitivos. Sirve para crear los códigos que se usarán.
5- Creación de los objetos y gráficos de las escenas.
6- Creación de la música y sonidos del videojuego.
7- Creación de las diferentes escenas.
8- Creación y desarrollo del videojuego.
9- Pruebas del videojuego. Una vez esté el juego lo bastante pulido se
10- Realización del Documento de Desarrollo del videojuego.
11- Exportación Multiplataforma.

Desde el punto 4 hasta el punto 8 se indica que al ser escenas separadas, que no tienen nada que ver una escena con otra, es decir independientes, su realización ha sido de forma secuencial estos puntos, pero todas las escenas a la vez.

DOCUMENTO DE DISEÑO DEL JUEGO

Capítulo 3: Documento de Diseño de Topi's World

Topi's world es el videojuego de este trabajo fin de grado. Se trata de un videojuego diseñado y desarrollado por mí totalmente, desde cada gameObject, escenario con sus correspondientes componentes y propiedades hasta el audio, pasando por las luces y cámara empleada.

Topi's World consiste en un videojuego compuesto por varios mini juegos conocidos y adaptados al ambiente e historia del videojuego. Se centra en un mundo nuevo, un planeta desconocido en el que viven unos seres llamados Topis, estos seres tienen ciertas inquietudes y problemas que el personaje que movemos resolverá. Todos viven en un pueblo que es la columna del videojuego, pues es el escenario que tiene mayor libertad y que consigue que podamos jugar a cada uno de los mini juegos.

Los mini juegos planteados en el videojuego son todos de conseguir puntuación por lo que se podrán jugar una y otra vez. Para conseguir esta puntuación se debe seguir unas reglas o mecánicas para que el juego se produzca con éxito.

A continuación se explica la mecánica de los minijuegos que se contienen aquí.

Mecánicas

Alien Invasion

Es una recreación del antiguo juego Space Invaders pero en primera persona persona. Para ello el escenario cuenta con los siguientes elementos:

- El personaje principal que dispara a través de su embudo. Disparará unas bolas rojas a gran velocidad a través de su arma que tiene incorporado en su boca-nariz en forma de embudo. Se moverá libremente por el terreno del escenario.
- El terreno por el que se mueve el personaje y que si se cae el personaje muere. Este terreno contiene unas piedras enormes en las que se puede ocultar de las naves. Este terreno tiene límites por los que no se caerá el personaje debido a que si no fuera así sería muy difícil para el jugador.
- Los invasores o naves, que son el enemigo del juego. Para conseguir la puntuación hay que destruir mediante disparos las naves. El juego empezará con enemigos que no hacen nada, solo avanzar. Aparecerán en sitios aleatorios, aunque siempre sea delante del personaje. A medida que consiga puntuación el número y nivel de las naves será mayor haciendo el juego muy difícil.
  Hay tres tipos de naves, con lo cual, por destruir las naves tendrá distinta puntuación dependiendo del tipo de nave. Cada nave también hará un daño mayor.
  El juego termina si las naves le quitan toda la vida al personaje principal o si alguna de las naves consigue llegar al terreno del personaje.

- Además para ambientar el juego, el juego tiene una música enérgica con temática galáctica y electrónica que dejará de sonar si el personaje muere, para que así suene el sonido del personaje si muere.

El atractivo de este mini juego es la recuperación de un juego antiguo muy conocido por todos y adaptándolo para un tipo de juego muy famoso y seguido hoy en día como son los shooter por juegos como Call Of Duty o Battlefield.

Recolección en el Laberinto

Es una recreación del famoso PACMAN. Se trata de un laberinto en 3D que consistirá en ir a recoger monedas pero con vista 2D. Para ello el escenario cuenta con los siguientes elementos y mecánicas:

- El personaje principal; se moverá de forma continua por el laberinto. Su misión será recoger las monedas distribuidas que hay en el laberinto. Los enemigos perseguirán al personaje y para defenderse podrá disparar balas que salen de su boca-nariz a los enemigos.
- Los enemigos; se trata de unas "abejas-mono" de gran tamaño.
  Las abejas-mono amarillas perseguirán al personaje principal.
  En caso de que el personaje principal mate a algún enemigo éste será enviado a su jaula para volver a ser generado.
  En el momento que un enemigo toque al personaje principal, morirá el personaje y se pierde la partida del mini juego. Cada cierto tiempo aparecerán más abejas. Para ir complicando la partida.
- El personaje recolectará además de monedas unas frutas que le darán más puntuación.
- La cámara empleada para este mini juego será con una vista ortográfica a media distancia para que sea más fácil para el jugador poder controlar toda la situación y que el jugador pueda tomar decisiones.
- La música será una música inquietante pero tampoco muy tensa, ya que se busca la atención del jugador pero sin estresarlo con demasiada información.

El atractivo de este mini juego es la recuperación también de un juego antiguo adaptándolo a una vista 3D, con ello se conseguir devolver la niñez de adultos con un juego moderno.

Escondite Inglés

El siguiente mini juego consiste en el juego del escondite inglés, un juego al que de pequeño todas las personas hemos jugado y consiste en una persona que cuenta y al darse la vuelta tiene que fijarse si se mueven las personas a las que vigila. Para ello se cuenta con los siguientes elementos:

- El personaje principal que será el que tiene que estar quieto o moverse dependiendo del momento del juego.
- Un personaje al que llamaremos "el enemigo" pues tratará de fijarse en si se mueve o no para así seguir de vigilante.

- El mini juego acabará únicamente con la victoria del personaje principal. Cada vez que es detectado el movimiento del personaje principal se volverá a empezar. También se podrá salir del mini juego y volver al Pueblo.
- El juego será en tercera persona, con una cámara cercana al personaje con el objetivo de inmersión en el personaje.

El atractivo del mini juego es debido a que nunca se ha hecho virtualmente este juego tan típico de la calle al que de niños solemos jugar.

Pilla al Ladrón.

Este mini juego hace referencia al típico pilla-pilla. Un personaje se esconde y otro tiene que buscarle por el campo o escenario acordado. Los elementos y reglas que se desarrollan aquí son los siguientes:

- El escenario; Se trata de un bosque solo con árboles y rocas con algún que otro arbusto.
- El personaje objetivo, que sería nuestro enemigo al que tenemos que encontrar y tocar. Es decir para ganar en este juego hay que colisionar o tocar al enemigo. El personaje objetivo se moverá aleatoriamente más rápido o más lento que el jugador con la intención de ponerle un reto o motivación. Si el personaje principal se encuentra cerca éste saldrá moviéndose a algún otro lugar con la intención de no ser pillado.
- El personaje principal, que se moverá libremente por el escenario hasta los límites de éste. El personaje siempre se mueve a la misma velocidad. Tiene como objetivo atrapar al personaje objetivo que se encuentra escondido por algún lugar del escenario.
- Aquí es muy importante la ambientación sonora, ya que la música será baja, ya que importa mucho los sonidos, cuando el personaje objetivo se encuentre cerca del personaje principal se oirán pasos y si cada vez se encuentra más cerca más fuerte se oirán los pasos.

El atractivo del mini juego es al igual que antes, hacer referencia a un juego típico de la niñez. Además de estar pendiente del sonido del juego.

En sí, con estos juegos se busca el captar la atención tanto de pequeños como de adultos debido a que son juegos inocentes y con una dificultad mínima, con los que se entretendrán y divertirán a la vez que desarrollen capacidades de atención y habilidad motriz e incluso reacciones a estímulos tipo disparos, enemigos cercanos.

Escenas

Como se ha explicado los videojuegos en Unity se organizan mediante escenas que pueden denominarse como los niveles establecidos en este. Dependiendo del tipo de juego podemos diferenciar un nivel de otro. Por ejemplo, los niveles en Super Mario están diferenciados por sus niveles y se relacionan de la forma en que si superas un nivel puedes acceder al siguiente, incrementando también su dificultad. Otros, cada vez que tengamos un ambiente o escena diferente, tendremos una escena o nivel diferente.

En este caso, las escenas son escenas independientes ya que son diferentes mini juegos y se puede jugar a uno o a otro indistintamente de cual juegues primero. En sí, solo hay una escena o nivel más importante ya que será "la columna" del juego, pues a través de este nivel accedemos a los mini juegos y cuando termine se redirige a ese nivel también. El menú será el inicio del juego y aparecerá cada vez que se ejecute el juego.

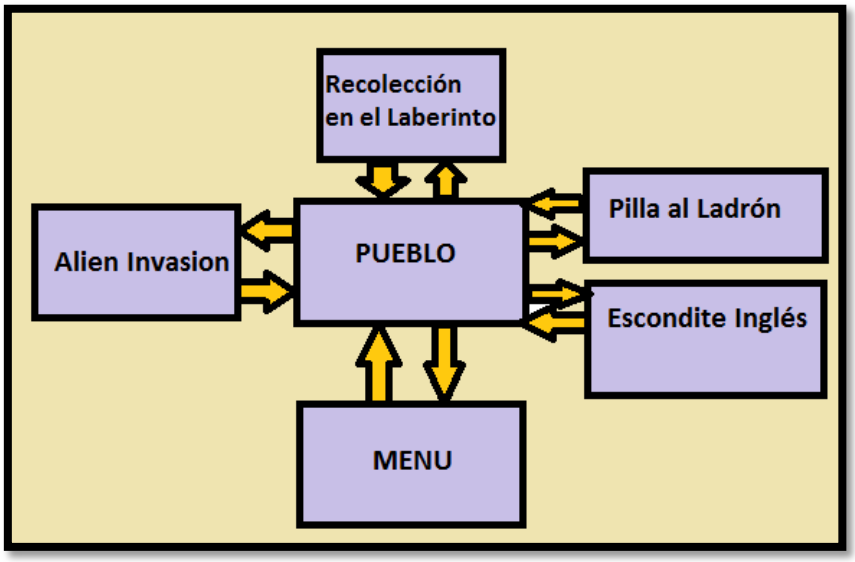


Figura 2: Acceso a las escenas del videojuego

Al igual que las mecánicas de la escena, que correspondería con el equipo de desarrollo, en un videojuego obviamente es importante el equipo de arte, el cual se encarga de la parte gráfica, como el modelado, texturizado y la parte de animaciones. Al cual le encargaremos estas tareas.

Menú

El menú de Inicio del juego se compone principalmente de un plano al que se le aplica un Image Texture y un canvas. En el canvas tendrá dos canvas interiores: Uno el Inicio, y otro que será el canvas de créditos.

En el canvas de inicio hay tres botones: Play, Créditos y Quit. Y en el canvas de créditos hay un botón para volver al canvas de inicio y los créditos.

La cámara visualizará una imagen 2D junto con los denominados canvas. La interacción y sus consecuencias en el Canvas del menú se refleja en la siguiente figura:

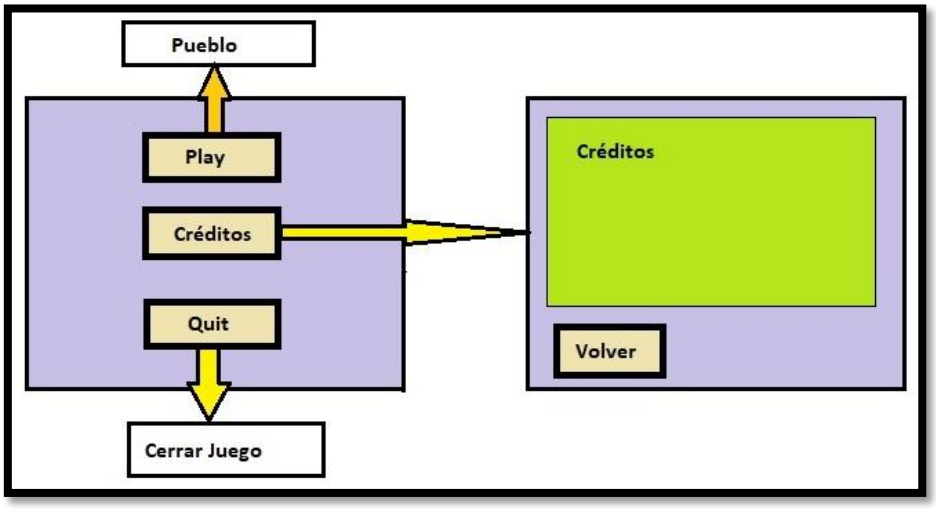


Figura 3: Interfaz de la escena Menú

Pueblo

Se trata de la columna vertical del videojuego pues aquí se puede navegar entre un minijuego y otro, además hay interacción con los elementos mostrados, proporcionando pistas de cómo ganar más puntos en los mini juegos.

Se trata de una escena basada en un terreno con casas, árboles, señales y una fuente en medio de la escena. Las casas y árboles de esta escena son bastante más grandes que los habitantes de este pueblo.

Los habitantes de este pueblo, serán de dos formas: Hay habitantes que al acercarse el protagonista del juego, lo saludan y le muestran un mensaje con pistas para superar los minijuegos con más puntos, o para entender cosas sobre su mundo creado.

Las siguientes escenas tratan de los minijuegos que contiene este videojuego, con tal de describir no sólo sus mecánicas sino también su estructura, se adjuntan una serie de bocetos hechos a mano correspondientes cada uno al nivel diseñado.

Alien Invasion

Es el minijuego estilo FPS, del género shooter con la intención de destruir el mayor número de naves. El player se encuentra en un trozo pequeño de tierra con dos rocas donde se puede ocultar. El skybox será oscuro, y de la distancia aparecerán naves. Hay tres tipos de naves. Tanto las naves como el personaje se dispararán balas. El juego termina cuando alguna nave toca el terreno o matan al personaje.

Mapa:

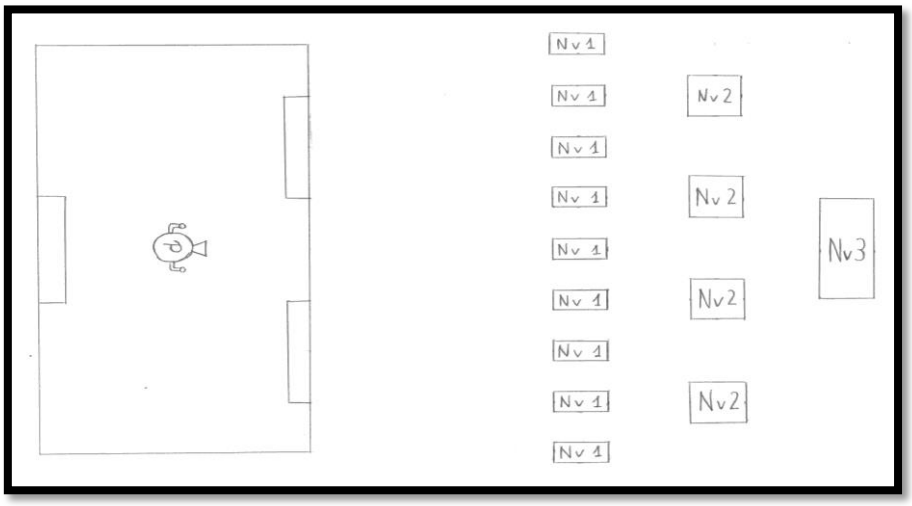


Figura 4: Mapa del minijuego Alien Invasion

Recolección en el Laberinto

Consiste en recolectar monedas y frutas por un laberinto sencillo. Los elementos a esquivar son pinchos asesinos que van apareciendo cada vez más, una bola asesina que al ser tocado también te elimina y los enemigos que tienen una forma rara. El juego será en tercera persona.

Mapa:

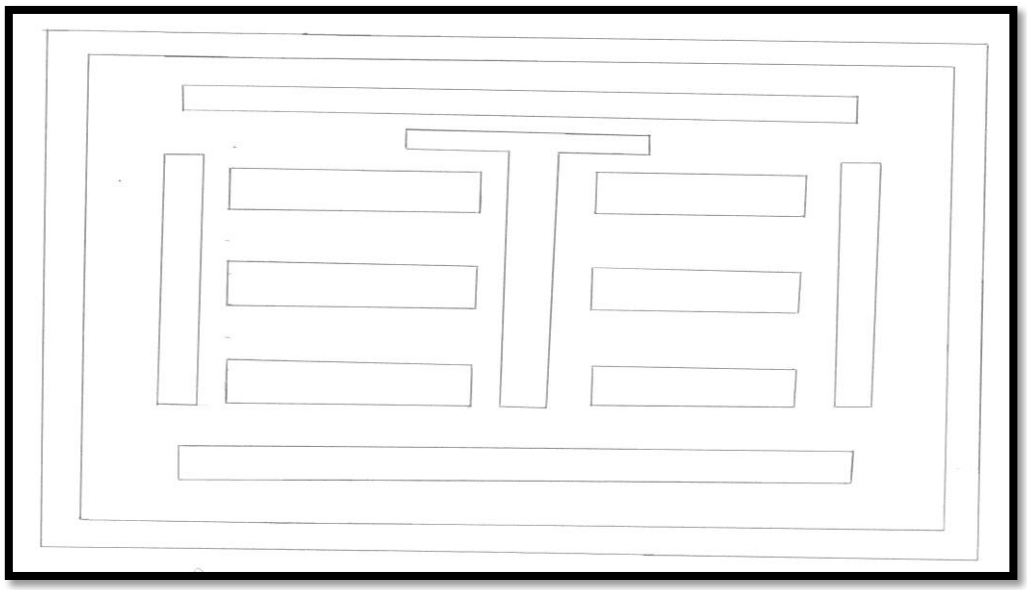


Figura 5: Mapa del minijuego Recolección en el Laberinto

Pilla al Ladrón

Consiste en el juego del pilla-pilla por el bosque. Hay que encontrar al personaje enemigo. Se trata de un minijuego en tercera persona.

La escena transcurre en un bosque donde se usa mucho el tema de sombras y silencio para hacer que el jugador se concentre en encontrarlo. Los elementos son el terreno lleno de árboles y los dos personajes, es decir, player y enemigo.

Mapa:

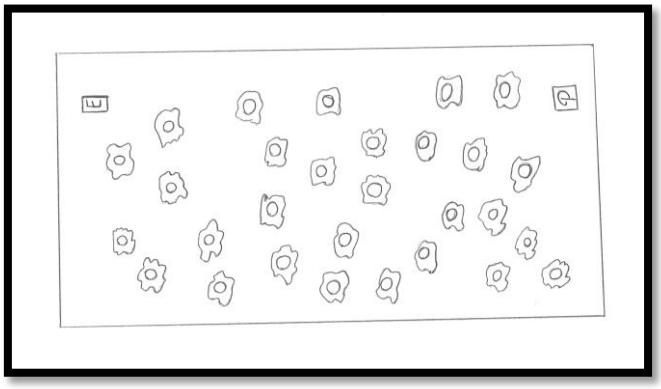


Figura 6: Mapa del minijuego Pilla al Ladrón

Escondite Inglés

Consiste en el juego del escondite inglés. El objetivo es llegar hasta la casa y que no te pillen en movimiento. Es un minijuego en tercera persona.

La escena transcurre en un camino hacia una casa. Los elementos son una casa y árboles que se han usado en el pueblo. Otros elementos que se usan son vallas, que indican el camino, y las paredes usadas.

Mapa:

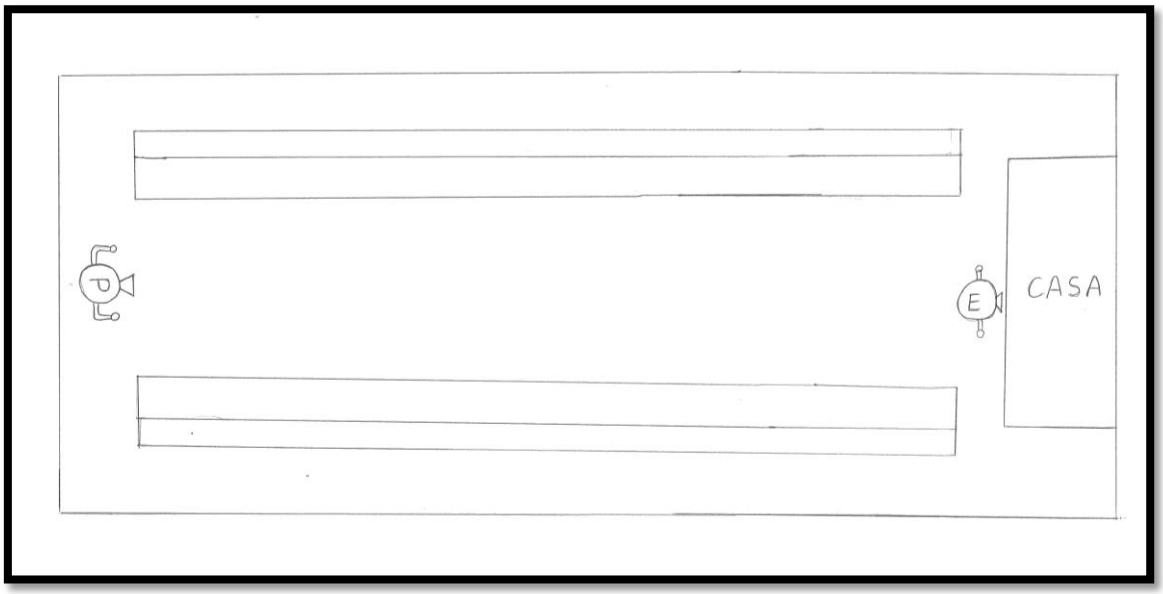


Figura 7: Mapa del minijuego Escondite Inglés

Canvas de los minijuegos y el Pueblo

Tendremos tres canvas para los minijuegos. El primero será el de Información y el activo durante la partida de los minijuegos. El segundo canvas confirmará el terminar el juego antes de perder o ganar. El tercer canvas es el de reinicio o salida del minijuego.

La estructura quedaría así:

PLATAFORMA ORDENADOR

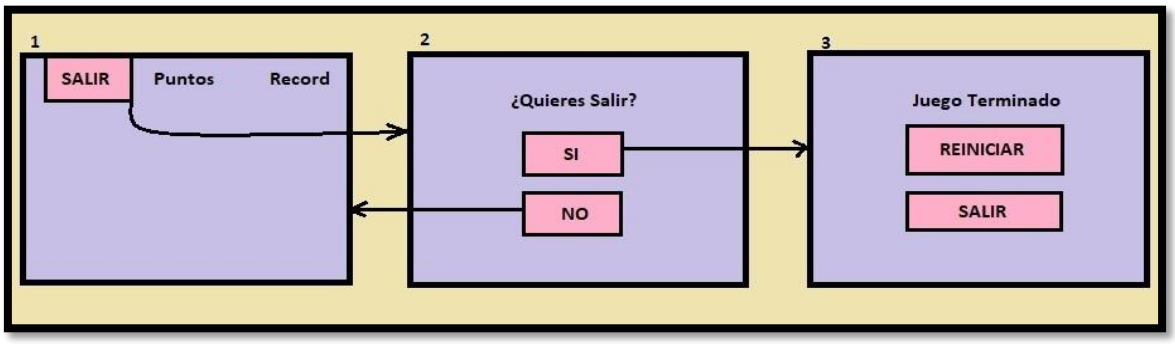


Figura 8: Interfaz en la plataforma Ordenador

PLATAFORMA ANDROID

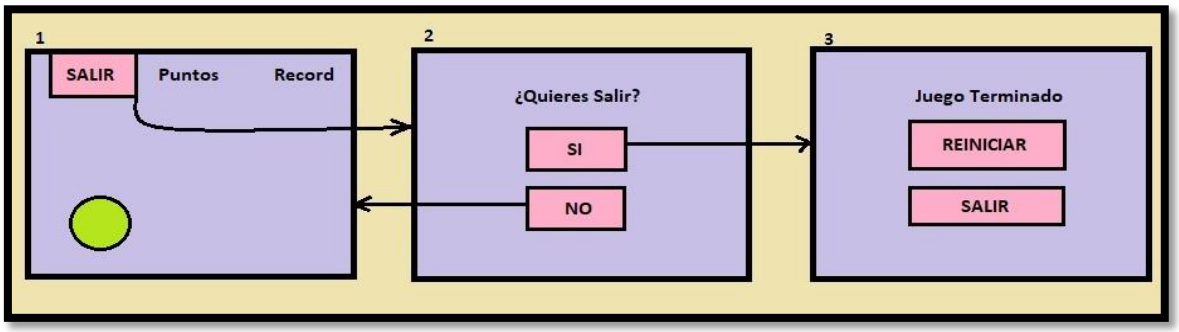


Figura 9: Canvas de la plataforma Android

DOCUMENTO DE DESARROLLO DEL JUEGO

Capítulo 4: Documento de Desarrollo de Topi's World

Para el desarrollo del videojuego se han adquirido los conocimientos proporcionados en los siguientes apartados de la bibliografía: [2, 4, 5, 7, 8]

ALIEN INVASION

Modelado

Como se trata de un juego de tiros los modelos de bala serán igual pero con distinto color para diferenciar y el jugador se puede cubrir detrás de dos rocas como este modelo.

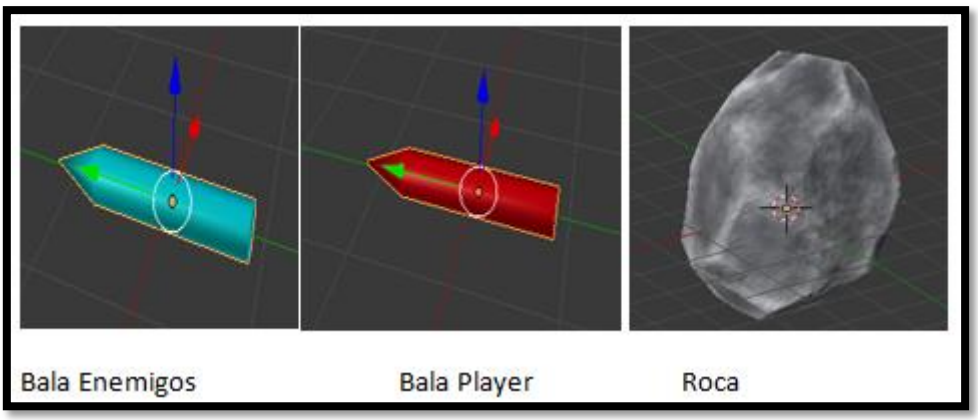


Figura 10: Gráficos del minijuego Alien Invasion

Los enemigos son estas tres naves creadas a partir del modelo de la nave 1. También se les cambian sus colores para hacerlas diferentes.

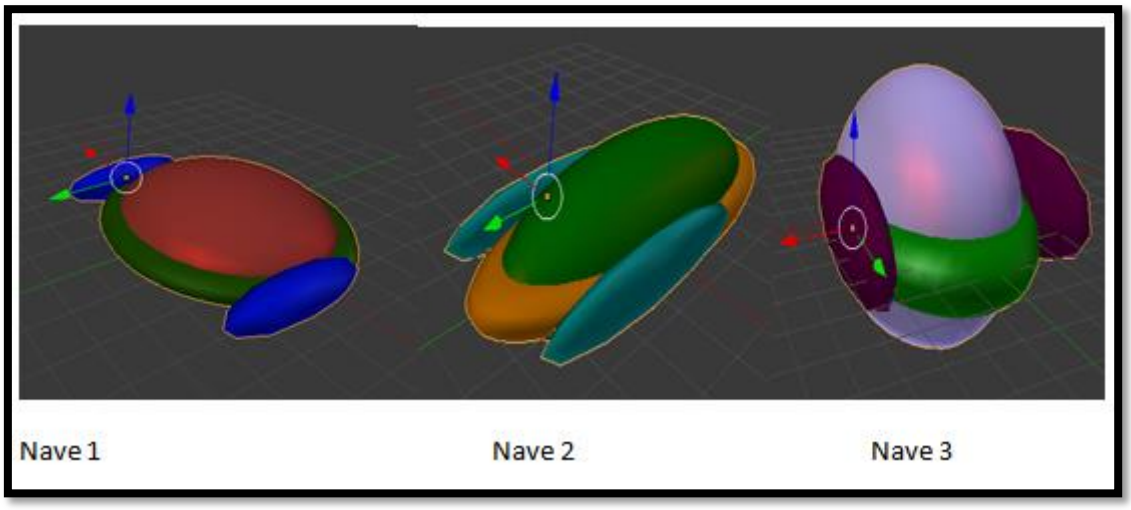


Figura 11: Naves del minijuego Alien Invasion

Programación

El primer paso es un script que consiste en un disparador. Será directamente para el jugador, ya que pulsando un botón se instanciará el objeto "bala" que se lanzará con cierta velocidad y se destruirá este objeto después de cierto tiempo.

Al igual que se ha creado un disparador para el player, se establece un disparador para los enemigos, será un script donde todas las variables serán públicas, para poder rectificar los datos desde la interfaz de Unity, ya que, hay que ajustar la dificultad de cada cuanto tiempo se dispara. Siempre disparará con la misma dirección, teniendo en cuenta el vector local de los enemigos, en este caso se trata del eje Z negativo.

Para la salud de los enemigos, el script contemplado se llama "EnemyHealth", que tratará de solucionar este problema. Para ello, se usará la función de colisiones "OnTriggerEnter", la cual detectará un objeto con el componente Rigidbody que comparará si se trata de una bala del player, mediante los tags. En caso negativo no ocurrirá nada, en cambio, si se cumple esta condición recibirá un daño de salud. Cuando se quede en 0 la salud del enemigo se activará el flag "isDead" que se comprueba todo el rato en la función "Update". Cuando se evalua que está activado este booleano se destruirá el objeto enemigo y añadirá una puntuación de 10 puntos al marcador.

También el player tiene una salud que los enemigos pueden disminuir a través del script "PlayerHealth". Esto lo hace de la misma forma que con los enemigos pero en vez del tag de la bala player se trata de la bala del enemigo. Como consecuencia cuando se activa la variable booleana "IsDead" del player, el player morirá, y se reiniciará el juego o se saldrá del minijuego.

Cuando algún enemigo llega al terreno de juego, también se termina el juego. El script se llama "Meta". Esto se gestiona a través de un cubo que será un collider que tendrá un flag que active el isDead del player.

El movimiento de los enemigos será tratado a través del script "EnemyMovement", lo único que hace es en el método "Update" actualizar la posición del enemigo adelantándolo siguiendo la dirección del vector local del objeto. Al igual que antes al hablar del vector local del enemigo será la dirección del eje Z negativo.

Como queremos que se vaya aumentando la dificultad, usaremos un objeto vacío que a través de un script llamado "Invocador" instanciando cada cierto tiempo los diferentes tipos de enemigos, haciendo que aparezcan de forma aleatoria.

Como último punto tenemos un script que trata de saber la puntuación que vamos obteniendo al destruir a los enemigos. Esto se evalúa mediante el script "PlayerScore".

Para reiniciar y quitar la escena se escribe el script "BotonesMarcianitos", donde se especifica a la aplicación que cargue de nuevo el nivel si se trata del reinicio del nivel o que vuelva a la escena del pueblo si se trata del botón salir.

A continuación se muestra un Plano de interacción entre Scripts.

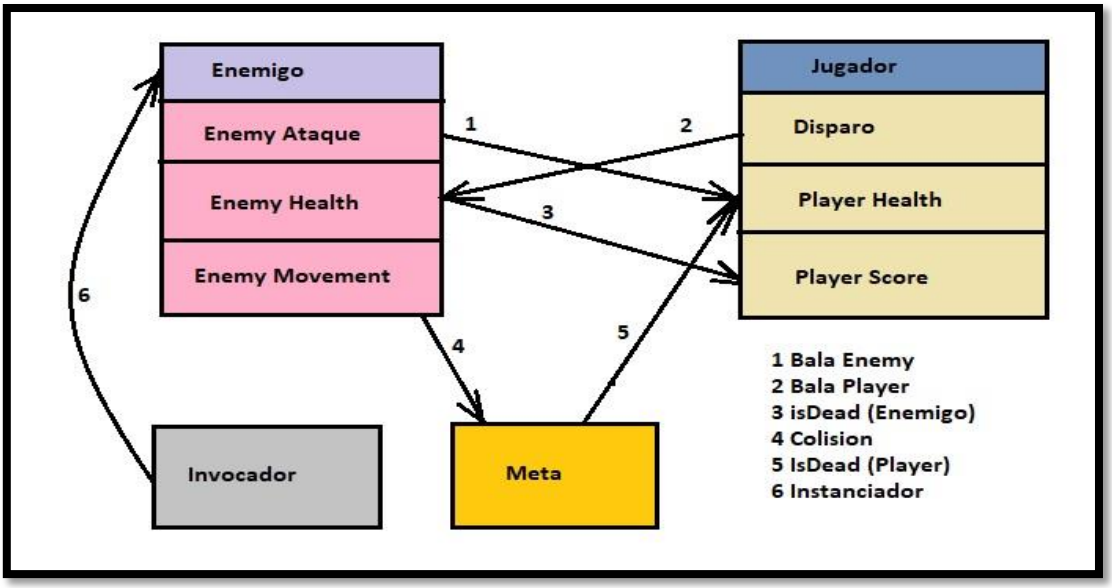


Figura 12: Diagrama de interacción entre scripts de Alien Invasion

Creación de la Escena

Para la escena de este minijuego no tenemos en sí muchos objetos que montar, se trata de un escenario que consta de un terreno con una anchura de 20 unidades y 50 unidades de longitud.

Lo siguiente será introducir un controlador en primera persona, cogeremos un modelo ya creado (un prefab) por Unity que importaremos. Este modelo es una cámara que está dentro del jugador haciendo la función de estar en primera persona, además ya viene incluso con el componente ”AudioSource” que luego se cambiará su sonido por las pisadas del personaje. Le introducimos también el componente CapsuleCollider con el booleano “Istrigger” activado, para así no dar presencia física, sino poder efectuar colisiones que detectaremos.

Lo siguiente es poner las rocas donde se podrá ocultar el player para que no le maten ni perder salud. Tendrán una presencia física y hará que las balas enemigas que choquen contra ella desaparezcan.

Ahora iremos enseñando la configuración de las naves:

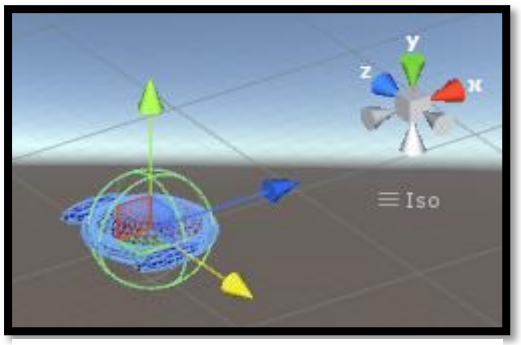


Figura 13: Nave. Vectores Locales y Globales

Lleva los componentes Rigidbody pero solo le afecta su cinemática, un collider esférico y luego los scripts “Enemy Health”, “EnemyMovement” y “EnemyAtaque”.

Como se observa en la imagen no tiene los mismos vectores, es decir los vectores locales (correspondientes al objeto) no se corresponden con los vectores espaciales (correspondientes a la escena).

Al igual pasa con los otros dos tipos de nave. Como se demuestra en la figura 14.

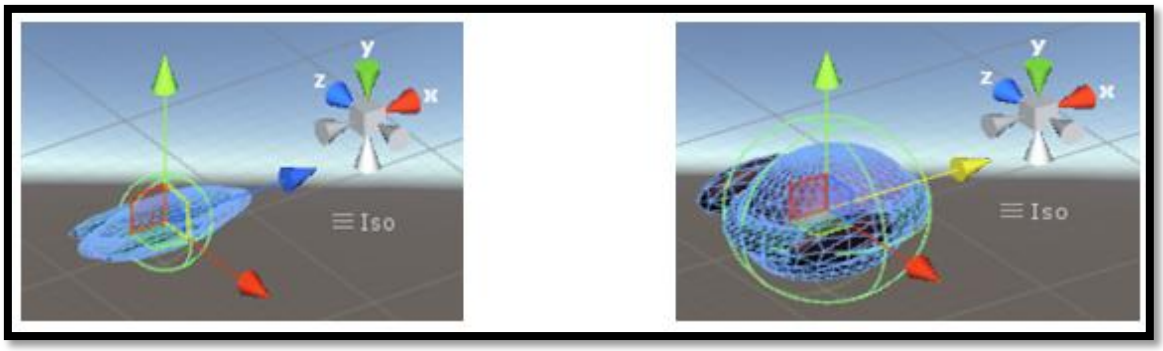


Figura 14: Naves 2 y 3. Vectores Locales y Globales

Los puntos de invocación para instanciar las naves es mejor verlo a través del Plano de Escena. Corresponden 9 puntos para la nave 1, 4 puntos para la nave 2 y 1 punto para la nave 3.

Se configura la interfaz de usuario que tendrá la escena. En él se ha configurado tal cual se pide en el documento de diseño, es decir 3 canvas unos dependientes de otros, además en caso de perder toda la vida el personaje directamente saltará el canvas de GameOver. Se puede navegar por los 3 tipos de canvas.

Los puntos obtenidos se obtienen de la destrucción de naves, cada vez que se destruya una nave se añadirá una puntuación. Esta puntuación se verá reflejada en la interfaz de usuario (canvas) destinada a Información del minijuego.

Música

Para la creación de la música usada aquí, se usará el programa Fl Studio. Para ello se creará una composición propia y única.

Se empezará hablando de la generación de los sonidos. Para ello se usará el clip de un golpe de kick para dar ritmo, el generador 3xOsc que generará un sonido que será la melodía en sí y el generador BooBass que sirve para los bajos de la melodía.

Figura 15: Patrón de la melodía

Figura 16: Plugin 3xOSC. Funcionamiento

El generador 3xOsc son 3 osciladores que para el sonido que se quiere lograr queda de esta forma configurado. El primer oscilador usará una señal senoidal. El segundo usa una señal cuadrada que será modificado su CRS (course Turning) con el objetivo de dar importancia a los graves de esta señal cuadrada. El tercer oscilador será también senoidal y al igual que antes se modifica su CRS con la misma intención, este último tiene menor volumen que los otros dos. Las melodías creadas con este sonido estarán en su quinta tonalidad.

Con esto conseguimos que al sumar las tres señales suene de una forma única. Se comprueba también el tiempo que suena un impulso de su señal.

Figura 17: Plugin 3xOSC. Señal en el Tiempo

El generador BooBass se dejará por defecto. Es encargado de hacer el bajo que queremos para añadir refuerzo a la música.

Figura 18: Plugin BooBass

La canción tendrá este aspecto, siempre el mismo nivel de volumen.

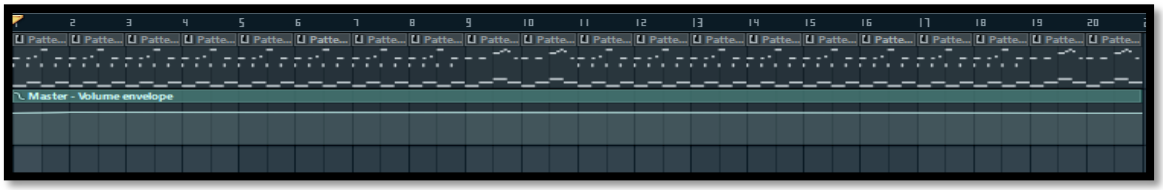


Figura 19: Canción del minijuego Alien Invasion

Al sonido que se usa por la melodía se le aplican una serie de cambios, una ecualización, un flanger y una reverberación. En la ecualización se observa la predominación de las frecuencias medias y más altas con respecto a las bajas, dado que de las notas bajas se ocupará el bajo. El flanger será un conjunto de filtros con delays, al que no se le aplican muchas de sus opciones. La reverberación será para dar mayor predominancia a la melodía y darle espacio musical, aunque la reverberación será la justa y será aplicado desde 75 Hz hasta 7.6 KHz.

Figura 20: Efectos Aplicados

Una vez hecho todo esto. Se exportará y se usará como música del minijuego.

En el caso del juego Alien Invasion existe un controlador en primera persona, es decir la cámara en primera persona tiene que tener el componente Audio Listener y se añade además a este, un componente AudioSource al que se le pone en bucle la canción creada para ello.

También se le aplican sonidos. En el script “PlayerHealth” se pone unas líneas que harán que al morir el jugador o perder la partida suene un sonido de pérdida indicando sonoramente y visualmente que ha perdido el minijuego.

RECOLECCIÓN EN EL LABERINTO

Modelado

Esta es la escena que tiene más gráficos incorporados más diferentes. El jugador tiene que evitar esa bola marrón, los pinchos y el enemigo abeja-mono. Tiene que conseguir las monedas y frutas que aquí se muestran.

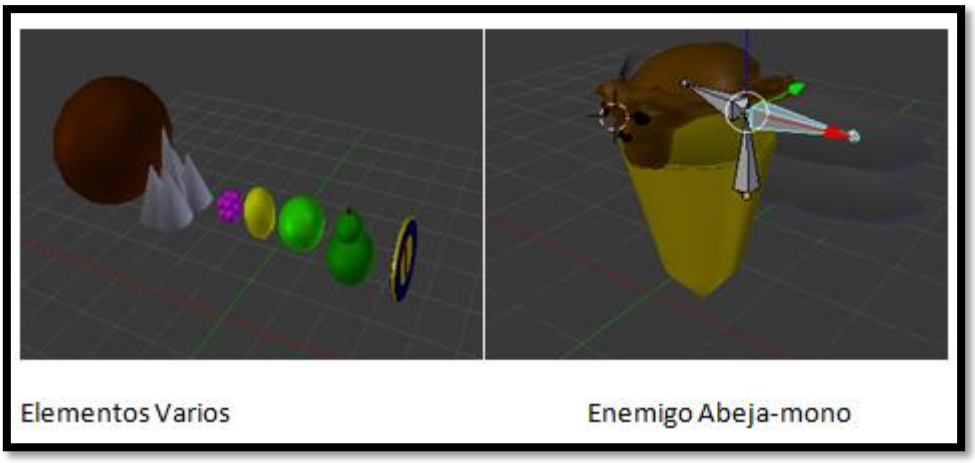


Figura 21: Gráficos del minijuego Recolección en el Laberinto

Programación

Este minijuego tiene bastantes scripts parecidos, en él se van a especificar varios métodos interesantes:

Empezando por el player, en este caso tenemos un script que trata de su movimiento, a este le llamaremos “movimientoTopi”, aquí la única variable pública se trata de la velocidad de movimiento que va a tener, ésta será necesaria para poder hacer más fácil o difícil de jugar. Se trata la programación de la animación de andando a través del método “Anim”. Para ello existen varios métodos, entre ellos está el movimiento y el giro del personaje, “void move” y “void turning”, al que se le pasan los valores de teclado o el joystick virtual para android recogidos en el método general FixedUpdate.

Otro script que tiene el player es el “ComportamientoTopi”, en este se especifica exactamente qué es lo que ocurre en las colisiones con otros objetos como pueden ser los pinchos, los enemigos, o la bola asesina que directamente te eliminan del minijuego y el jugador tendría que empezar la partida de nuevo. O también el caso de colisionar con frutas o monedas que le dan puntuación. Además al ser eliminado comprueba el record de puntuación para siempre intentar superar este record. Es decir le damos al jugador la motivación de superarse a sí mismo.

El script “DisparadorLaberinto”, al igual que en el minijuego de las naves, nos permite disparar objetos arrojadizos por la nariz. Además, da lo mismo sigue los vectores locales del objeto player, con lo cual resulta fácil la configuración de disparar para las 4 direcciones posibles.

Un script bastante importante está en la cámara, la cual va a seguir al jugador durante la partida, dejando además al player centrado en la escena. Para ello este script llamado “CamaraFollow” realiza lo siguiente: mediante un vector público y el componente transform del player se ajusta siempre a través del método FixedUpdate.

Ahora hablaremos primero de los enemigos, estos se instancian a través de SpawnPoints como en el anterior minijuego de las naves mediante el script “CreadorEnemigos”.

Los enemigos llevan dos scripts. El primero se trata del movimiento de los enemigos, este script llamado “movimientoEnemigo”, a través del componente NavMeshAgent se mueve por la escena buscando siempre la posición del player, como el player está siempre moviéndose, el enemigo cuenta entonces con una inteligencia artificial que conoce siempre la ruta más corta para llegar al player. El otro script detecta cuando el player le lanza un objeto que le mata. En caso de que se detecte la colisión con el objeto con el tag “BalaPlayer”, entonces se destruye el objeto Enemigo.

Al principio del juego, se ejecutan tres scripts que tienen la misma estructura. Se tratan de los instanciadores de monedas, frutas y pinchos asesinos. Estos contienen la función InvokeRepeating, el cual ejecutará cada cierto tiempo la instanciación de estos objetos. La monedas y las frutas se instancian además con cierto giro random, además de estar siempre instanciándose de forma aleatoria por toda la escena del player.

Como últimos scripts a nombrar tenemos los referentes a la bola asesina. El primero se trata de un objeto vacío que será un SpawnPoint que instanciará la bola cada cierto tiempo. El segundo se trata del movimiento y comportamiento de esta bola. La bola se moverá en dirección del eje X positivo y se destruirá cuando encuentre el objeto vacío rompeObjetos, que tiene el tag “rompebolas”.

Como flujo del juego tenemos el siguiente.

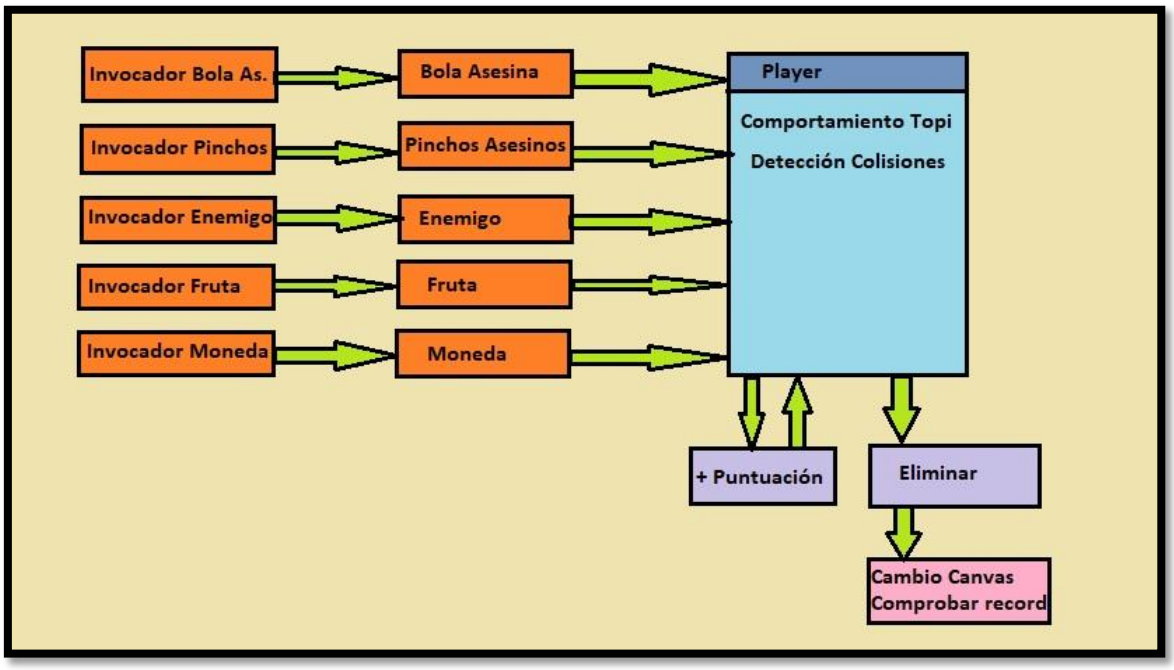


Figura 22: Diagrama de Interacción entre scripts del Laberinto

Creación de la escena

Lo primero que será necesario es tener un terreno. En el collider que viene incorporado le damos la física de zeroFriction. Con esta física ayudamos a que el personaje o cualquier objeto que tenga el componente rigidbody no traspasen el terreno. A continuación se le añade a este terreno el laberinto creado. Consistirá en una serie de bloques colocados de acuerdo al mapa del laberinto especificado en el documento de diseño.

Para los instanciadores usaremos objetos vacíos. Los dos instanciadores de enemigos serán colocados en los lugares especificados según el documento de diseño. Al igual pasa con el instanciador de la bola Asesina. En cambio los instanciadores de pinchos asesinos, frutas y monedas, da lo mismo donde se coloquen, ya que según el código, estos se irán moviendo por todo el escenario colocando sus objetos correspondientes.

Para el player se colocan sus scripts correspondientes, también le corresponden los componentes “animator” y Rigidbody. El primero será encargado de la animación del player, el segundo aunque da presencia física para las colisiones, en este es para ayudar en los scripts en los que se especifica su uso. Además se le ponen dos componentes colliders, uno con trigger activado y el otro sin él. El que no lleva da presencia física con respecto a otros objetos para no traspasar los objetos. En cambio el que lleva el trigger activado es para la ayuda del script “ComportamientoTopi”. Ambos tienen el mismo tamaño de radio y centro.

La cámara usa el script “CamaraFollow”, además se configura con una proyección Ortográfica, así obtendremos una vista parecida al 2D, que al estar hecho en 3D el nivel dará profundidad al juego.

Para el enemigo al conocer sus dos scripts que se han creado, hay que adecuarlo para hacer efectivo estas mecánicas. Al igual que el player, el enemigo tiene los componentes Rigidbody y animator y sus colliders propios (en este caso, debido a la forma del enemigo se usa un capsule collider). El enemigo tiene una componente especial y es el “Nav Mesh Agent”. Este componente es importante ya que es el encargado de conocer el terreno por el que se puede mover. Para ello antes hay que hacer un bakeo (conocimiento del terreno a través de su malla). Para ello se habilita la ventana Navigation y se consulta la ventana del agente de bakeo. Una vez hecho el bakeo el terreno se ve de la siguiente forma:

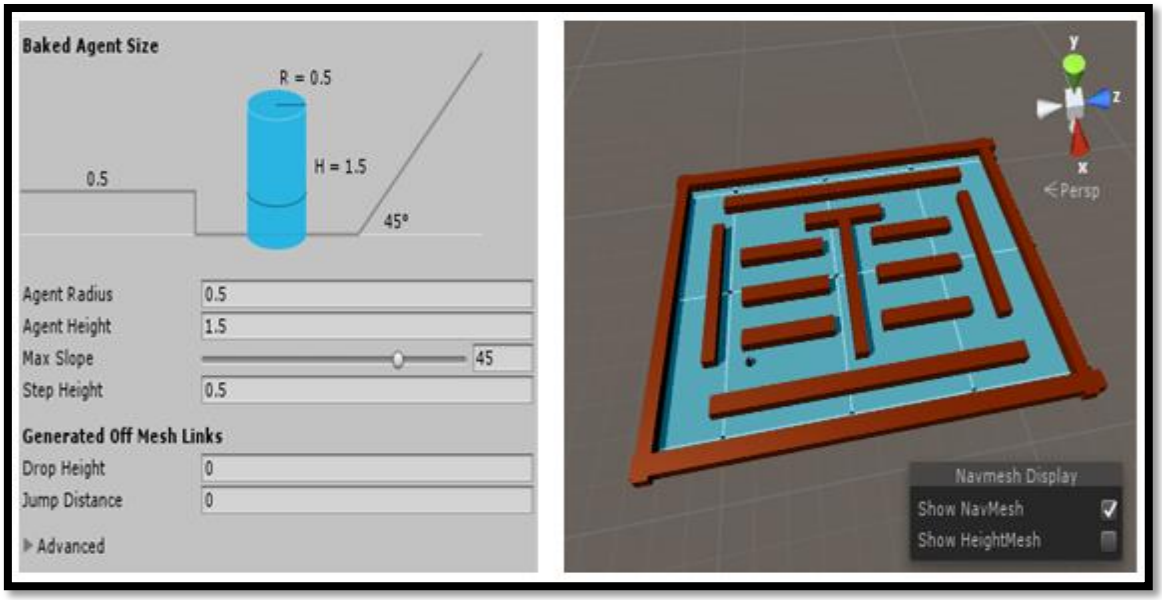


Figura 23: Bakeo del Laberinto

Se establecen los tres canvas que lleva el minijuego.

El primer canvas corresponde al de información del minijuego, en él se muestra los puntos que lleva y el record. Tiene un botón que conecta al segundo canvas, y dependiendo de si estamos en plataforma Ordenador o Android, un joystick y botón para los movimientos y disparo que se ejecutan sobre el player (son para la versión en Android).

El segundo canvas se corresponde con la confirmación de querer salir del juego, este es necesario para salir del minijuego. Se constituye con dos botones que dan la opción de sí o no. Si pulsa en Si, se sale al tercer canvas, y si es que no, se vuelve a activar el primer canvas y se sigue el minijuego.

El tercer canvas se corresponde con el reinicio y salida del minijuego. Dependiendo de dónde se pulse se cargará una escena u otra. Coge un objeto vacío llamado Auxiliar al que le aplicamos el componente “BotonesLaberinto”.

Música

Para la creación de la música usada aquí, se usará el programa Fl Studio. Para ello se creará una composición propia y única.

Lo primero es hablar de los sonidos a utilizar. Se usará un clip de palmada, un clip de caja y un oscilador llamado “Toxic Biohazard” que lo configuramos con un sonido de secuencia. Esto quiere decir que los sonidos serán casi por defecto y no hay que hacer muchas modificaciones en el plugin del oscilador. La melodía será una nota larga para que así el oscilador haga sonar el sonido de secuencia completamente entero.

La canción o música quedaría de la siguiente forma:

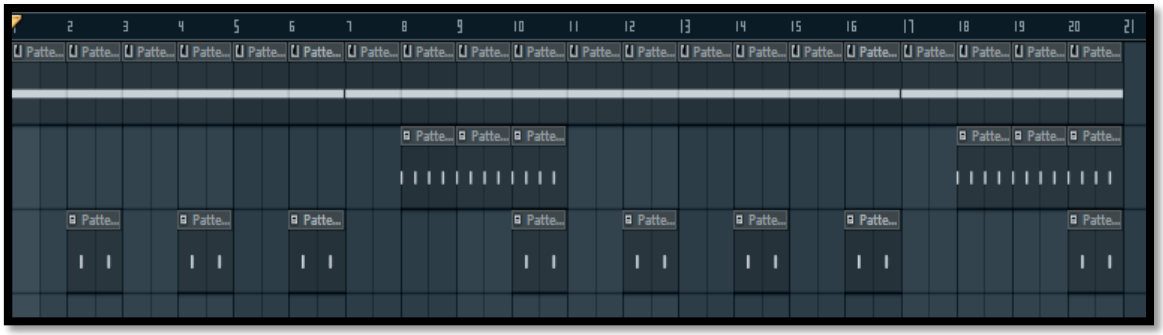


Figura 24: Canción del minijuego Recolección en el Laberinto

Cabe mencionar que la melodía lleva una serie de arreglos como son un flangus, un ecualizador y una reverberación. El flangus será una serie de filtros que a lo largo del sonido se intercambian, en el ecualizador vemos que las frecuencias medias además de tener la reverb son las que predominan sobre las bajas frecuencias.

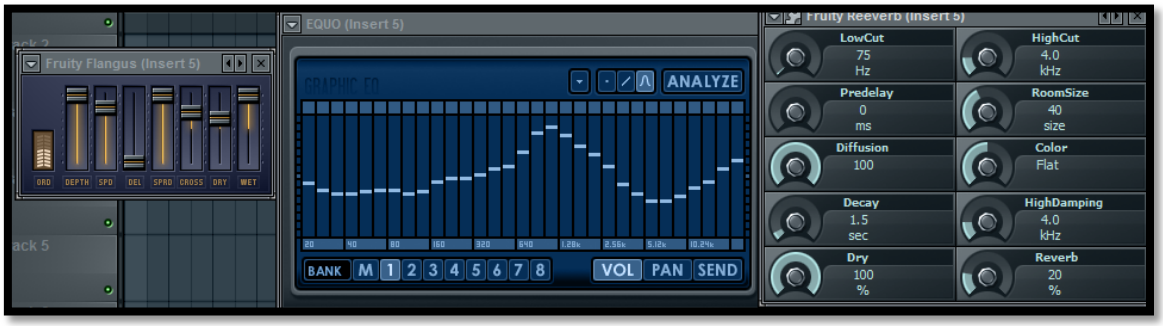


Figura 25: Efectos aplicados

Cuando se obtiene la melodía o canción requerida para este caso, se exporta y se usaría en Unity.

En el caso del laberinto la cámara que sigue al player tiene el componente Audio Listener y se añade al player, un componente AudioSource al que se le pone en bucle la canción creada para el laberinto.

También se le aplican sonidos. En el script “ComportamientoTopi” se escriben unas líneas que harán que al morir el jugador o perder la partida suene un sonido de pérdida indicando sonoramente y visualmente que ha perdido el minijuego.

PILLA AL LADRÓN

Modelado

En este no se han utilizado modelos creados por Blender. Todo lo necesario está en Unity para crear el escenario.

Programación

Esta escena va a reutilizar varios scripts. Estos son “movimientoTopi” y “CamaraFollow”. Estos son necesarios para el movimiento a través del terreno y su inspección.

El script ComportamientoTopiP se comporta en sí como el manager de la partida, a través de sus variables públicas que se establecen a través de la interfaz de Unity se establecen las relaciones. Su método “calcularTiempo” realiza cálculos para determinar los segundos que lleva buscando al enemigo, esto se muestra en el canvas de Información del juego. Ante la colisión con este objeto con el enemigo, (el cual tiene el tag “Enemy”) se evalúa el método “ComprobarTiempos”, el cual compara el record actual con el tiempo realizado, cuanto menos tiempo se tarde en pillar al ladrón mayor será el record. Para saber el record establecido se evalúa a través de la función Start con una función secundaria llamada “ComprobarRecord”.

El enemigo tiene varios scripts. “ControladorNavMesh” usa el NavMeshAgent y establece una serie de funciones públicas que se usarán en el script “ComportamientoEnemigo”. El comportamiento del Enemigo se basa en usar puntos aleatorios del terreno para ir allí como si fuera su meta. Después de ir a un punto éste es cambiado y así constantemente, sería como intentar huir del player.

El enemigo además lleva un objeto vacío que lleva un script llamado Detector Distancia que establece que el NavMeshAgent del enemigo vaya más rápido o más lento por el mapa, depende de si está dentro o fuera del collider establecido para este script.

Al igual que en los anteriores minijuegos tendremos un script para salir de este minijuego llamado “BotonesJuego”, cargará una escena u otra depende de si eligen reiniciar o salir del nivel. En caso de salir, entonces se cargará la escena del pueblo.

El flujo del Minijuego es el siguiente:

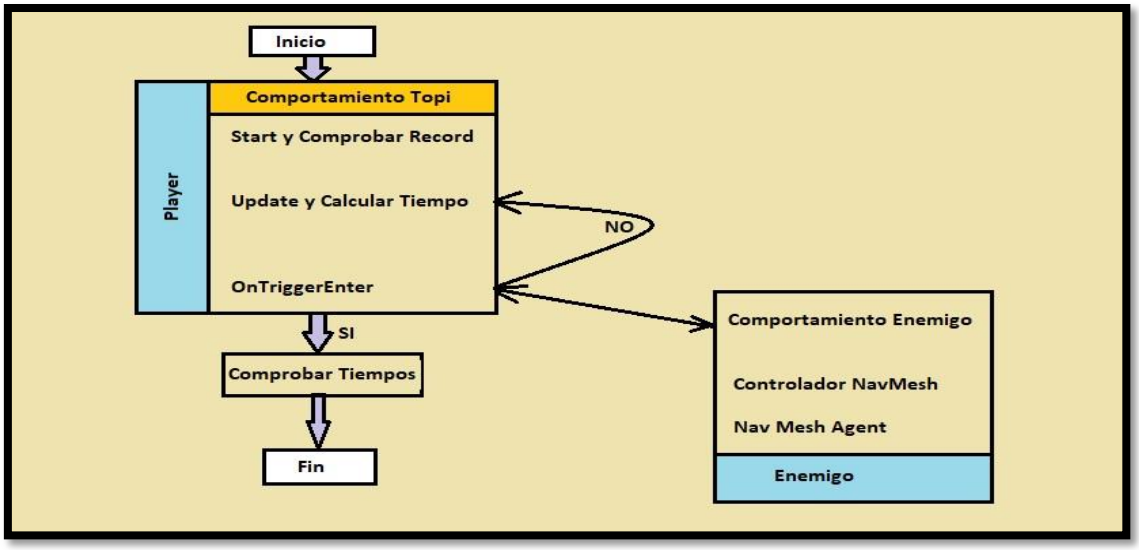


Figura 26: Diagrama de Interacción entre scripts del minijuego Pilla al Ladrón

Creación de la Escena

Para la creación de este nivel o escena donde se realiza el minijuego de pillar al ladrón se tiene en cuenta como en los anteriores, el mapa mostrado en el documento de diseño.

Para ello, se pone un terreno al cual su collider le ponemos la física "ZeroFriction". Dentro de la opción Terrain, se elevan los extremos del mapa para imposibilitar la subida tanto al enemigo como al player. Se establece una textura de césped y en el apartado "Posicion Árboles" se colocan muchos árboles iguales que es un Asset descargado del Standards Assets con la estructura especificada en el documento de diseño.

La luz se pone vertical apuntando al terreno. Así tendremos la menor sombra posible. Se establecen igualmente que las sombras que se generen sean suaves. Para poder distinguir debajo del árbol lo que se vea en la pantalla.

Se crean cuatro objetos vacíos que serán los puntos o metas a los que llega el enemigo. No importa donde estén situados, ya que el minijuego establecerá de forma aleatoria dónde aparecerán.

El enemigo tendrá los componentes Animator (con el controlador de animación "TopiProta") y NavMeshAgent. Al igual que antes será necesario el bakeo del terreno para que éste conozca el terreno.

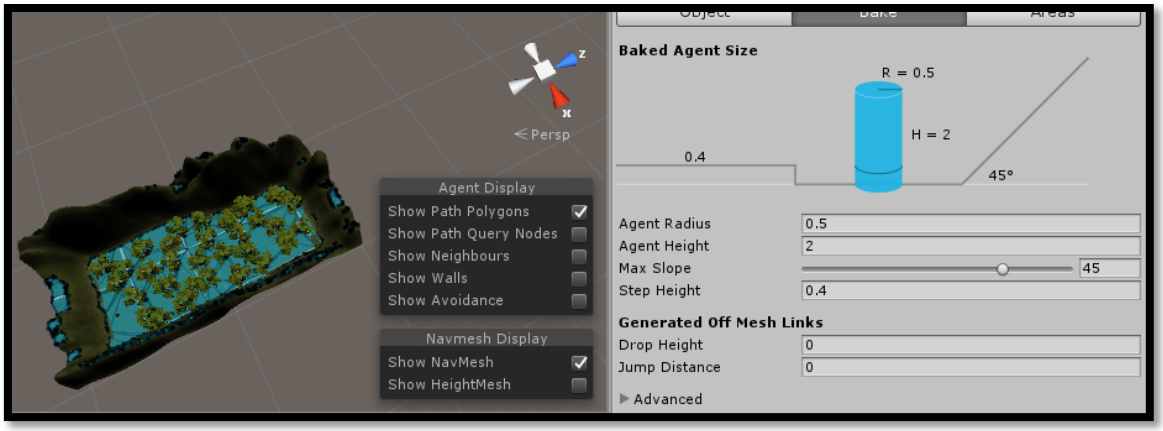


Figura 27: Bakeo del Bosque

Se le incluirá un colisionador esférico sin el trigger para darle presencia física, y así evitar que traspase los árboles. Además los scripts correspondientes para el enemigo ya comentados anteriormente. Dentro del enemigo en su centro existe un objeto vacío que evalúa mediante un collider con trigger cuando el player está fuera o dentro de este collider.

El player usará los componentes Animator (con el controlador de animación "TopiProta"), RigidBody como siempre hemos mostrado. Lleva dos colliders. El primero es esférico para darle la presencia física y no pueda traspasar los árboles. El segundo se trata de un collider en forma de cubo que se pone delante del player, su función es obtener la colisión con el enemigo para así terminar el juego. El aspecto es el siguiente:

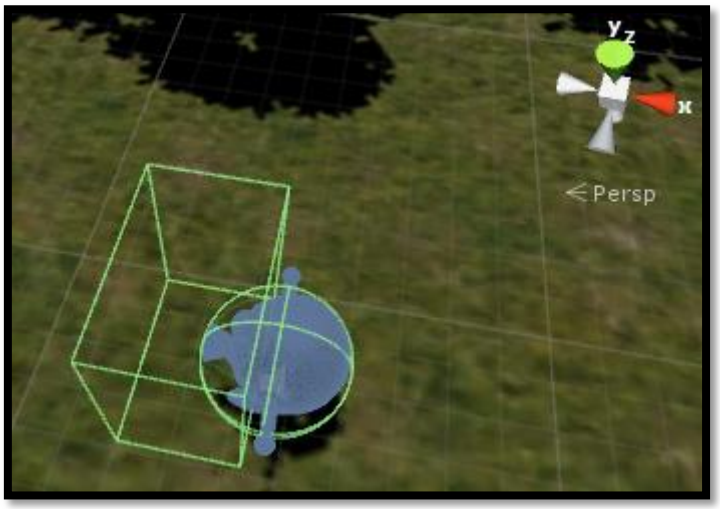


Figura 28: Colliders del Player

Se le incluye sus scripts correspondientes y se establecen sus relaciones a través de las variables públicas que se pueden configurar.

La cámara llevará el script "CamaraFollow" ya usado anteriormente, se ajusta a la altura y distancia necesaria para un juego de primera persona que parezca inmersivo, es decir, estar en la piel del player, pero siendo tercera persona, tal y como se especifica en el documento de diseño.

Al igual que antes se van a poner los canvas de Información, mostrando tanto el tiempo de ejecución del nivel en segundos, el tiempo record en el que ha sido pillado y un botón de salida del nivel que lleva al canvas de Confirmación. Dependiendo de la plataforma a usar tendremos joystick virtual o no. En caso cierto se trata de plataforma android, en caso negativo, Ordenador.

El segundo canvas es el de Confirmación de salida del juego, éste relaciona el primer y el tercer canvas. Y el tercero es el cargar una escena u otra. En caso de reinicio se carga la escena del bosque, es decir, en la que nos encontramos. En caso de salir se carga la escena del pueblo.

Música

Para la creación de la música usada aquí, se usará el programa Fl Studio. Para ello se creará una composición propia y única, aunque más bien, en realidad no se trataría de una canción sino más bien de una composición de sonidos.

Como primer paso especificar los sonidos que tendremos: Sonido de piano a través del plugin "FL Keys", el sonido de alguien chistando y como último sonido ambiente de un bosque. Para la melodía será una composición con mucho silencio y las notas dadas están en la quinta tonalidad, es decir frecuencias medias-altas.

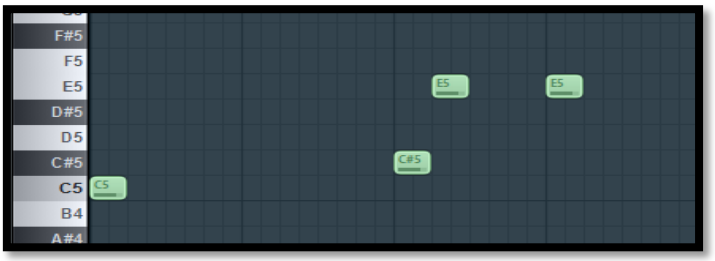


Figura 29: Melodía Piano para la ambientación sonora

El ritmo del ambiente sonoro sería el siguiente:

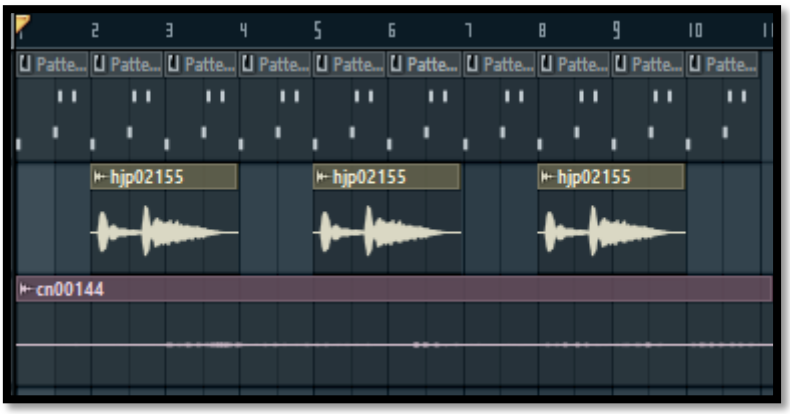


Figura 30: Ambientación sonora

Al tratarse de un ambiente sonoro no se le aplica ningún efecto de retoque como podría ser un ecualizador. Se supone que se tienen los sonidos que se quieren. Se exportaría y ya estaría listo para su uso en el videojuego.

En el caso del juego en el bosque la cámara que sigue al player tiene el componente “Audio Listener” y se añade al player, un componente AudioSource al que se le pone en bucle el ambiente sonoro que se ha creado para ello.

También se le aplican sonidos. En el script “ComportamientoTopiP” se escriben unas líneas que harán que al pillar el jugador al enemigo, en la partida suene un sonido de victoria indicando sonoramente y visualmente que ha terminado el minijuego.

ESCONDITE INGLES

Modelado

Excepto la valla de madera, estos son modelos reutilizados del pueblo.

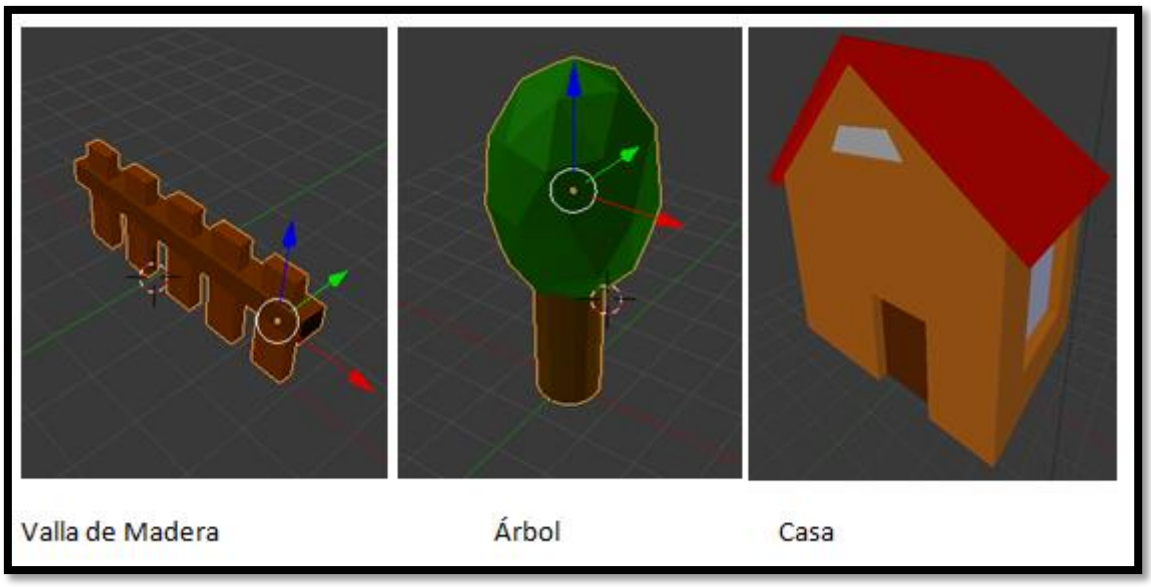


Figura 31: Gráficos del minijuego Escondite Inglés

Programación

Al igual que en el anterior minijuego, se reutilizan scripts con la intención de disminuir la cantidad de scripts o código que ejecutar y guardar dentro del juego. Los scripts usados serán el movimiento del player y "CamaraFollow".

El objetivo del escondite inglés es llegar a una meta. Para ello se escribe un script llamado "managerPartida". Este script tiene dos funciones, por un lado habilitar y deshabilitar el canvas en función de si se ha llegado o no a la meta. Si llega a la meta salta el canvas de juego terminado. Para ello se hace a través del método "OnTriggerEnter" que evalúa si la colisión es debida a un objeto con el tag "player". Por el otro lado se encarga de calcular el tiempo que lleva jugando en esa escena, cada vez que se reinicia empieza la cuenta de tiempo.

El enemigo aquí tiene varios scripts relacionados que forman una máquina de estados. A continuación se explica detalladamente.

El script "Maquina de Estados" es el script general que relaciona los scripts referentes a los estados y comportamientos de esta inteligencia artificial creada.

Se usarán variables públicas heredadas del Monobehaviour (Clase de todos los scripts y componentes) que serán los diferentes scripts, dos variables booleanas correspondientes a sus estados para evaluar si ha ocurrido o no y poder establecer condiciones y una variable pública que al igual que los estados determinará el estado inicial. Como última variable pública es el componente "Mesh Renderer" que se usa para establecer el color a un cubo. Además existe una variable privada llamada Estado Actual, también heredada del Monobehaviour que corresponde con el estado que se está ejecutando en ese momento.

En el método “Start” se especifica que estadoActual tiene que ser el estado inicial y queriendo establecer el estado desde el principio se definen las variables booleanas, también con respecto al estado inicial. Se crea un método o función pública llamada “activar estado” para poder hacer las transiciones entre un estado y otro.

El primer estado a comentar es el estado Cuenta correspondiente al script “EstadoCuenta”. Este script lo que hace es contar hasta un número que ha sido definido de forma aleatoria. Este será el momento en el que el player se podrá mover para intentar llegar a la meta. Además para darle mayor realismo al juego y poder hacer que el jugador lo vea se establecen dos métodos. Uno se dedica a que el personaje se encuentre girado mientras cuenta. El otro es dar el color Verde a un cubo que se encuentra encima del enemigo.

El segundo estado es más complicado de entender pero es lo que realmente haríamos en la vida real cuando jugamos a este juego.

Se establecen una serie de variables públicas, como son el manager de la partida, para poder interactuar los objetos y darse la instrucción de terminar la partida, y dos variables “transform” que se encargan de la comparación entre posiciones. Al habilitar este script se ejecuta lo siguiente:

- Una serie de variables privadas correspondientes a tiempo que se inicializan a 0 o al valor aleatorio.
- El guardado de posición del jugador.
- La determinación del color rojo al cubo que se encuentra encima del enemigo.
- La inicialización de la corrutina de girar hasta ponerse mirando al jugador.

Entonces en el método Update ocurre lo siguiente: Se mira durante el tiempo que se haya fijado aleatoriamente. Una vez hecho esto, la inteligencia evalúa si la posición del componente “transform” del jugador es igual que la guardada por el transform auxiliar. En caso correcto, se volverá al estado Cuenta. En caso contrario se termina la partida.

Se puede observar el flujo del juego de la siguiente forma:

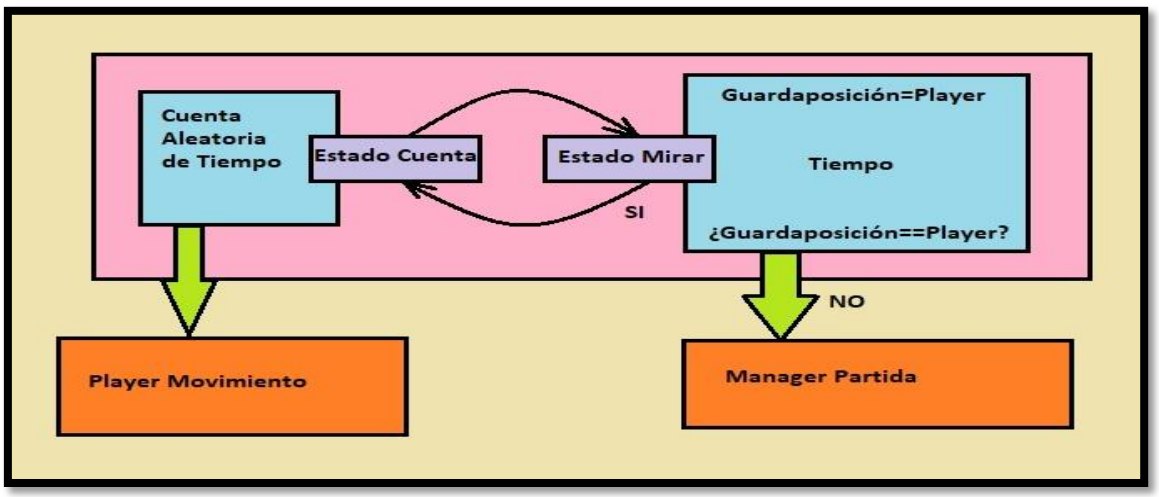


Figura 32: Diagrama de interacción entre scripts del minijuego Escondite Inglés

Además tenemos un script, que al igual que en los niveles anteriores corresponde a los botones que muestra en el juego terminado. Es decir, Reiniciar (carga el nivel cargado) o Salir (carga el nivel del pueblo).

Creación de la Escena

Lo primero es la creación del mapa. Para ello como en los anteriores niveles o minijuegos, se crea un terreno al que se le aplica la física de ZeroFriction y se le da una textura de césped. Se reaprovechan los modelos de casa y árboles usados para el pueblo. Se introduce además el modelo valla, haciendo que quede un camino único hacia la casa. El objetivo es dejarlo al montar el escenario es tenerlo igual que viene especificado en el documento de diseño.

En este minijuego para que el personaje no se caiga, por si no quiere seguir el camino y se mueve por el escenario, se ponen paredes con alguna imagen de un paisaje. Como último, se observa que al encerrar el terreno entre cuatro paredes, es necesario tener más luces. Se incorporan 3 luces a este escenario, de tal manera que exista luz en todo el escenario y sobre todo en el centro, que será el camino para llegar a la meta.

El player tendrá los componentes Animator (con el controlador de animación “TopiProta”) y RigidBody. También tiene un collider Esférico para dar presencia física al player. Como scripts desarrollados usa el “movimientoTopi”.

La configuración de la cámara que observa el juego está detrás del player y sigue al player con el script “CamaraFollow”.

El enemigo en este caso, sólo usa el componente Rigidbody. Lleva los scripts correspondientes a la máquina de estado y sus estados. Lleva un colisionador en forma de cubo con la intención de dar presencia física.

También se crea un cubo que será escalado para que se vea desde lo más lejos posible. Este será el indicador de cuando hay que moverse. Se colocará encima del enemigo.

Se configura un objeto que será un cubo al que le deshabilitaremos el Mesh Renderer, ya que lo que nos importa es su collider con el tamaño, este objeto será la meta, y le incluiremos el script “manager de la partida”.

Otro objeto vacío llamado “Auxiliar” será el que lleve el script de los “botones del juego”.

Como último al igual que antes se van a poner los canvas de Información, mostrando tanto el tiempo de ejecución del nivel en segundos, el tiempo record en el que ha sido pillado y un botón de salida del nivel que lleva al canvas de Confirmación. Dependiendo de la plataforma a usar tendremos joystick virtual o no. En caso cierto se trata de plataforma android, en caso negativo, Ordenador.

El segundo canvas es el de Confirmación de salida del juego, éste relaciona el primer y el tercer canvas. Y el tercero es el cargar una escena u otra. En caso de reinicio se carga la escena del bosque, es decir, en la que nos encontramos. En caso de salir se carga la escena del pueblo.

Música

La música empleada es una melodía que se ha descargado, es libre de autoría y se puede usar libremente para cualquier uso, es además una canción de 8 bits, a la que no se le ha aplicado ningún efecto.

En el caso del juego del escondite Ingles, la cámara que sigue al jugador tiene el componente “Audio Listener” y se añade además al player, un componente AudioSource al que se le pone en bucle la canción creada para ello.

También se le aplican sonidos. En el script “Estado Mira” están escritas unas líneas que harán que al ser visto el jugador y perder la partida suene un sonido de pérdida indicando sonoramente y visualmente que ha perdido el minijuego.

MENU

Programación

Sólo se usa un script aquí, es un script que carga el nivel del pueblo o quita el juego en sí. Se usa en el canvas principal.

Creación de la Escena

Tendremos un canvas que será el menú principal con tres botones: jugar, créditos y salir. El botón jugar tendrá el método que carga el pueblo. Salir tendrá la función de salir de la aplicación/videojuego. El botón créditos envía al canvas de créditos.

El canvas de créditos tendrá los créditos del videojuego y un botón para volver al canvas del menú principal.

Pondremos una imagen que estará detrás del canvas para darle un toque más animado. Será un dibujo del videojuego.

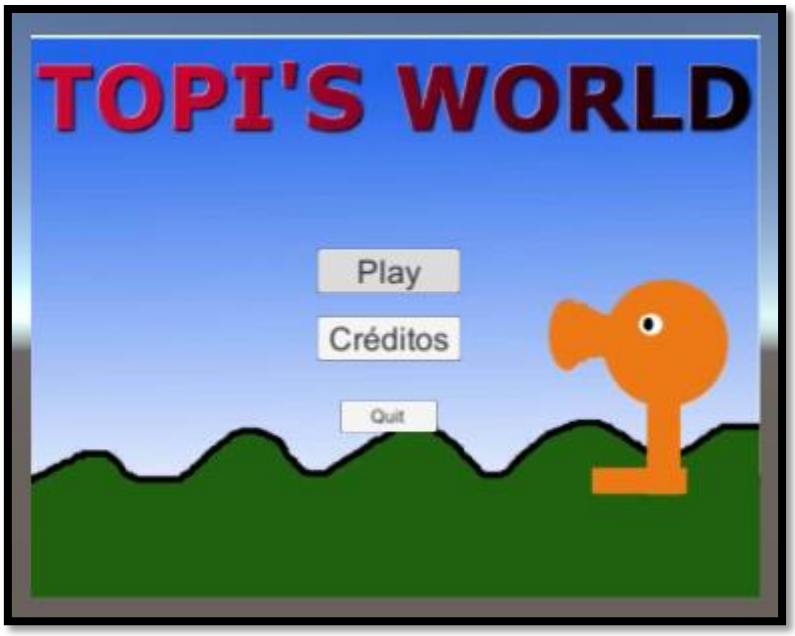


Figura 33: Menú Principal

Música

La música empleada es una melodía que se ha descargado, es libre de autoría y se puede usar libremente para cualquier uso.

El componente “AudioListener” se encuentra situado en la cámara de la escena. La canción para el menú se pone en el objeto “Scripts”, que es un objeto auxiliar para el menú. No se habilita su bucle. Es una canción que da la bienvenida al videojuego.

PUEBLO

Modelado

Estos son los modelos y objetos usados en la escena con los que compondremos un pueblo.

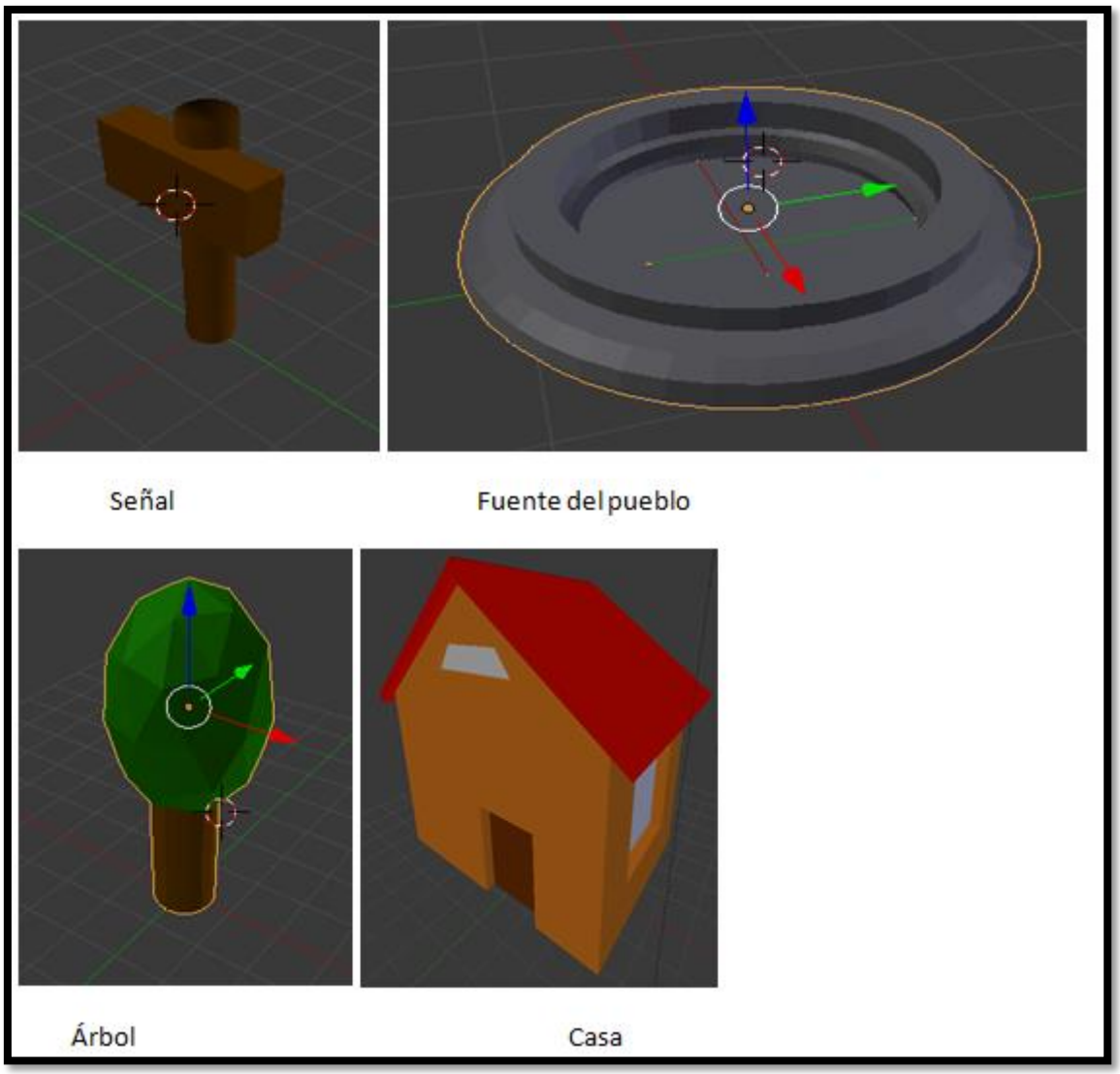


Figura 34: Gráficos del Pueblo

Programación

Se reutilizan scripts con la intención de disminuir la cantidad de scripts o código que ejecutar y guardar dentro del juego. Los scripts usados serán el movimiento del player y "CamaraFollow".

Como esta escena consiste en ser una escena para moverse entre los minijuegos e interactuar un poco con el videojuego es lógico un script que funcione para cargar los diferentes minijuegos, este script se llama "NavegarMapa".

Para poder cargar los minijuegos se recurre a usar diferentes canvas, habilitar o deshabilitar los canvas es la función del script "ActivarMinijuego". Mediante este script detecta si el player colisiona y está dentro o fuera de este. Cuando el player salga de la colisión entonces se deshabilitará el canvas del minijuego correspondiente.

Al tener tantos canvas, se crea un script a propósito para la inicialización de un canvas u otro llamado “TopiCanvas”. En sí, solo se tiene al inicio el canvas general, es decir el que nos va a dar la información y las cosas que nos dicen los “Topi”, e incluso si tratamos de la plataforma Android el movimiento del Topi.

Como se menciona los personajes del pueblo dicen información. Esta información viene dada a través del script “Textos”. Dependiendo del color del personaje tendrá información distinta que decir, además dirá un texto aleatorio de los elegidos.

Otras cosas que hacen los personajes por el pueblo es aparecer en una casa y desaparecer en otra. Para ello se tienen dos scripts: “Mover De Un Sitio A Otro”, que se trata de un instanciador de personaje, “TopiSeMueve” que usando el componente navMeshAgent va hacia un destino.

Luego existen muchos personajes que se mueven por el mapa usando puntos de Meta, esto se ve en el “PuntosMeta”, será una copia parecida a un script ya usado anteriormente.

No existe un flujo del juego como tal, debido a que no hay un juego como tal aquí. Aquí solo sirve para poder moverse por el mapa e ir a los minijuegos.

Creación de la Escena

Lo primero como en todos los niveles será la creación de un terreno por el que el player y los personajes se muevan. Este terreno tendrá sus bordes elevados para poder hacer que ningún personaje, ni el player puedan salirse de lo diseñado. Se establece la física “ZeroFriction” y la textura de césped.

Una vez hecho esto se sacan un árbol y una casa, ya que son los elementos que más abundan y se hace su prefab. A ambos se le incluyen un collider. Al árbol se le incluye el capsule collider y a la casa el box collider. Después estos elementos se distribuyen de forma que se parezca lo más posible al mapa del documento de diseño. Al igual se hace con las señales. Se incluye en el centro del mapa una fuente, al que se le añade el agua. Como último paso se agrupan todos estos objetos en un objeto vacío para limpiar la ventana.

A continuación se ponen cuatro personajes, cada uno de un color diferente que serán los que indiquen el minijuego. Tendrán su collider, tanto el de presencia física como el de comprobación de colisión. Se atribuye a éstos el script “Activar Minijuego”.

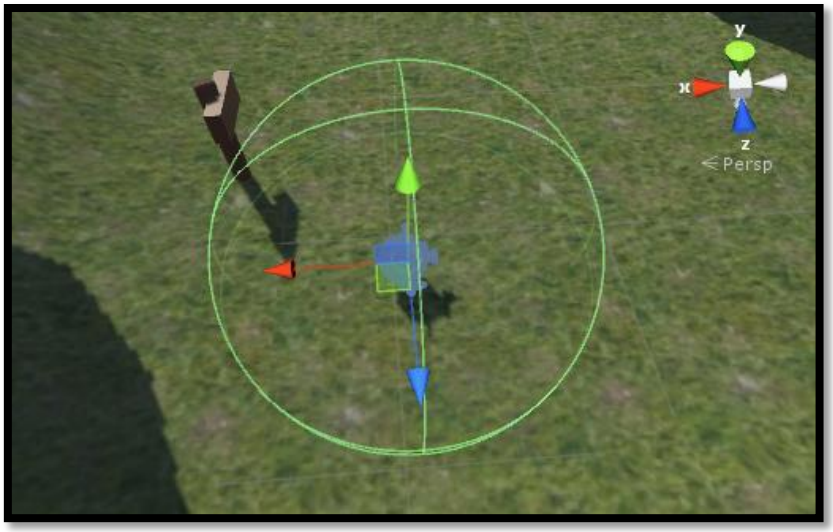


Figura 35: Collider de detección del player

Se crean los diferentes personajes, algunos cada uno con los scripts que necesiten. Es decir si hacemos los que van de una casa a otra, usarán “MoverDeUnSitioAOtro” y “TopiSeMueve”, en caso de solo mostrar diálogos usarán “Textos” y si lo que hacen es moverse por el mapa usarán el script “TopiCarrera”.

Los personajes que hablan tendrán dos colliders, uno de presencia fija y otro de comprobación de colisiones. Ambos serán “Sphere Colliders”. En caso de que se muevan tanto para un lugar como para otro necesitan el componente NavMeshAgent. Además hace falta hacer un bakeo de la escena.

Figura 36: Bakeo del pueblo

Como se observa en la imagen, hay unos cubos, o planos amarillos estos son indicadores de que hay están los minijuegos. Se usa una segunda cámara, ésta con inclinación cenital. Esta

cámara se configura para ser un minimapa que se muestra en la esquina inferior derecha de la pantalla. Al igual que la cámara que sigue de cerca al personaje, esta cámara 2, seguirá al personaje. Este es el aspecto que tiene en Android el usar dos cámaras y con el canvas general.

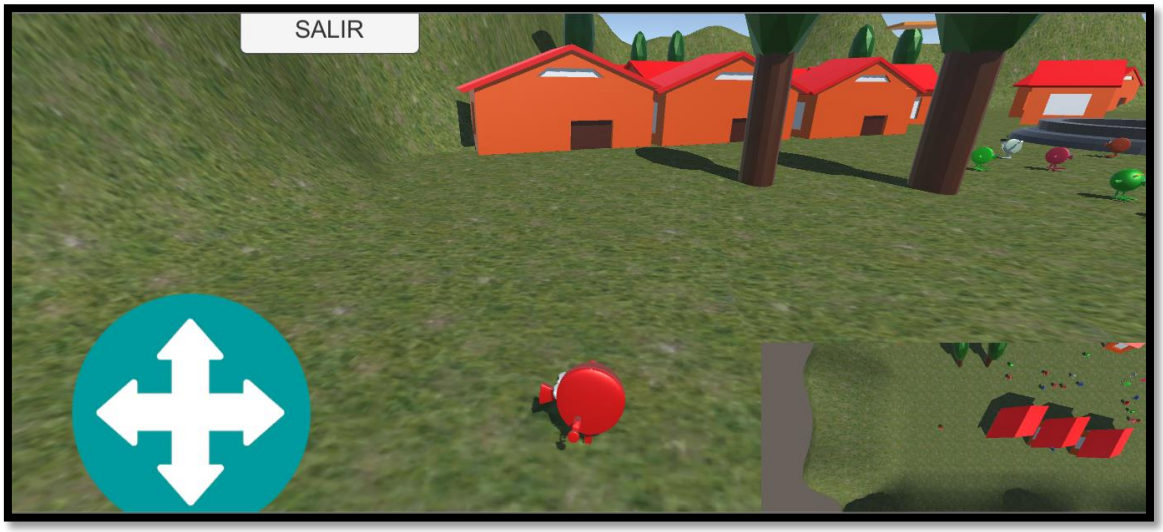


Figura 37: Vista Canvas General Android desde Unity

Se crean los diferentes canvas necesarios. Es decir, uno general que como se menciona permite el movimiento android, y permite ver los mensajes de los personajes del pueblo. Luego existen los canvas para cada minijuego, es decir ayudan a cargar el juego. Hay una por cada minijuego. El botón Salir que se observa en la pantalla carga el menú principal.

Como último, nombrar los múltiples objetos vacíos que se usarán, muchos de ellos serán Puntos de Meta o Instanciadores de personajes.

Música

La música empleada es una canción libre de derechos y que permite que al mezclar y realizar cambios en la canción se pueda usar con cualquier fin.

Los cambios realizados son los siguientes:

Gross Beat, ha permitido crear cambios de volumen casi inapreciables que al final cambian el ritmo y permiten parecer música como la creada hasta ahora. Se trata de la línea naranja.

Figura 38: Efecto GrossBeat

Al igual que en anteriores canciones, esta canción ha sido ecualizada también dándole predominancia a las frecuencias medias con respecto a las bajas.

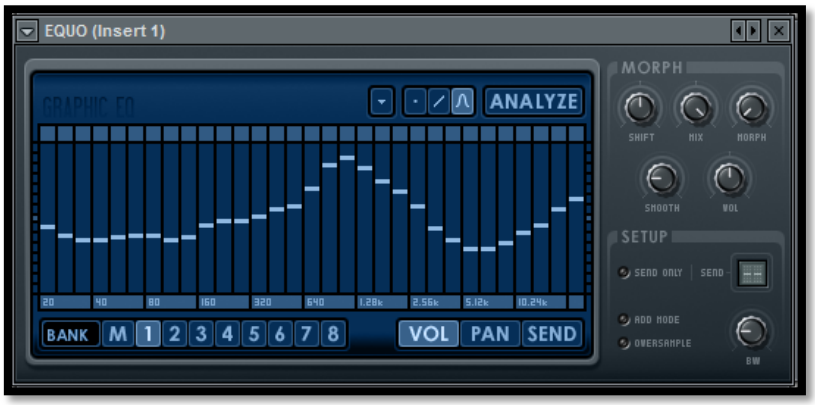


Figura 39: Efecto Ecualizador

Se le aplica un “flanger”, con este efecto conseguimos cambiar el sonido de la canción, es decir con sus múltiples cambios que realizamos cambiamos los filtros y los momentos a los que se realiza.

Figura 40: Efecto Flanger

Como último se le aplica un distorsionador llamado “Hardcore” que coge el sonido de la canción y lo distorsiona según frecuencias. Además para ello usa varios efectos como son el “chorus”, la ecualización, y la “reverb” en este caso.

Figura 41: Efecto Hardcore

Con todo ello cuando suene de una forma que se quiera, se exporta y se usa para el videojuego.

En el caso del pueblo existen dos cámaras. La cámara más cercana que sigue al personaje tiene que tener el componente Audio Listener y, se añade además al player un componente AudioSource al que se le pone en bucle la canción creada para ello.

También se le aplican sonidos. En el script “ActivarMinijuego” y “Textos” se pone unas líneas que harán que al hablar el personaje suene un sonido como si estuvieran hablando entre ellos indicando sonoramente y visualmente una conversación.

ANIMACION

Rigging

En este apartado se comenta el desarrollo de las animaciones y los controladores de animación empleados. Desde el momento en el que esté hecho tanto los personajes o modelos en Blender hasta su puesta en el videojuego.

Blender es el software usado para el diseño y desarrollo de los personajes y enemigos que tendrán animaciones en el videojuego. Para ello una vez diseñado, se incluirá un esqueleto al que se le ajusta el peso de los huesos a la malla del personaje.

Todos los personajes del pueblo tienen el mismo esqueleto. Se observa la estructura del esqueleto.

Topi Protagonista y sus personajes

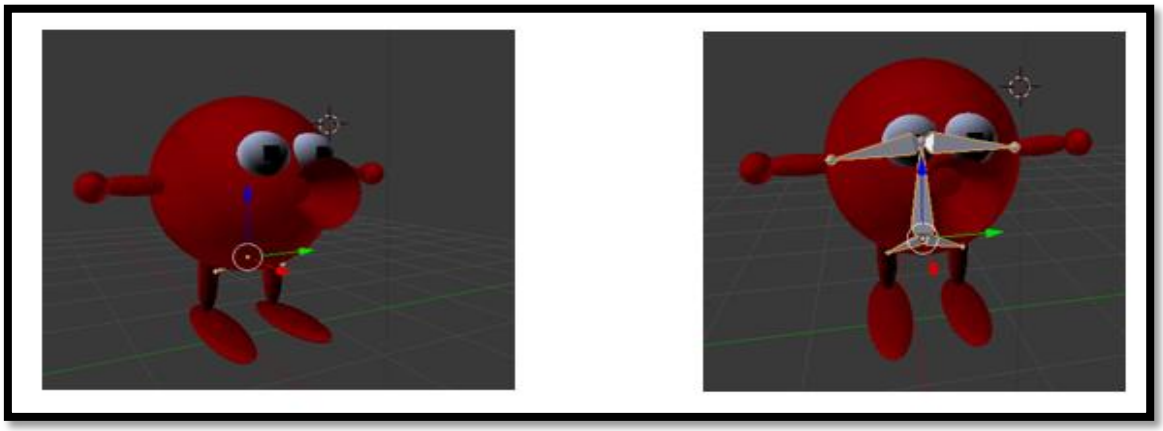


Figura 42: Rigging Personaje

A continuación se muestran el mapa de peso de cada uno de los huesos que se usan en las animaciones.

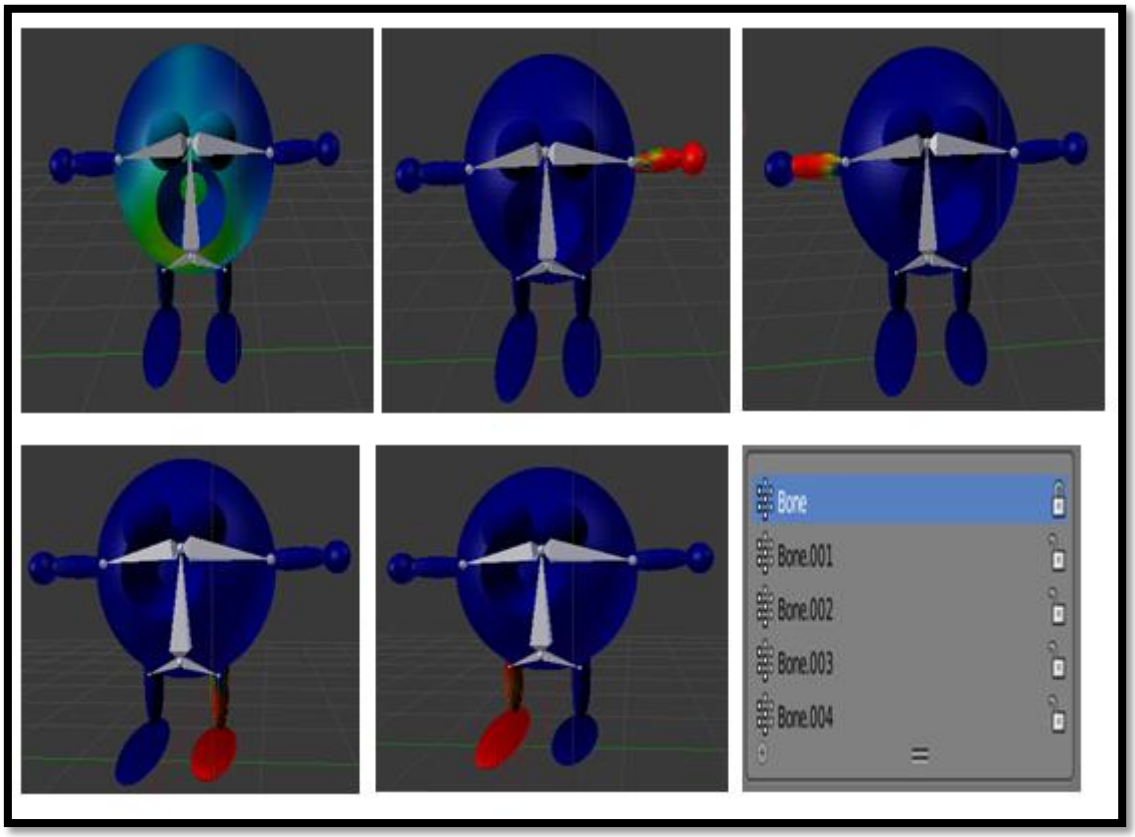


Figura 43: Mapa de peso de los huesos del Personaje

Enemigo del laberinto

Este objeto tiene un esqueleto más sencillo que la de los personajes, como se puede comprobar. El uso del esqueleto que se le incluye es para la animación de vuelo, es decir, como bate las alas.

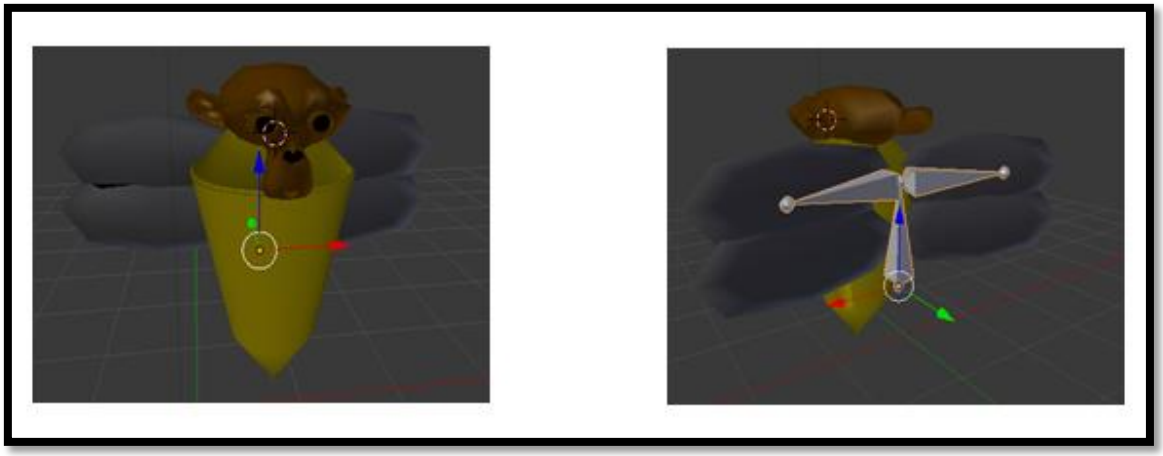


Figura 44: Rigging Enemigo Abeja-Mono

A continuación se muestran el mapa de peso de cada uno de los huesos que se usan en las animaciones.

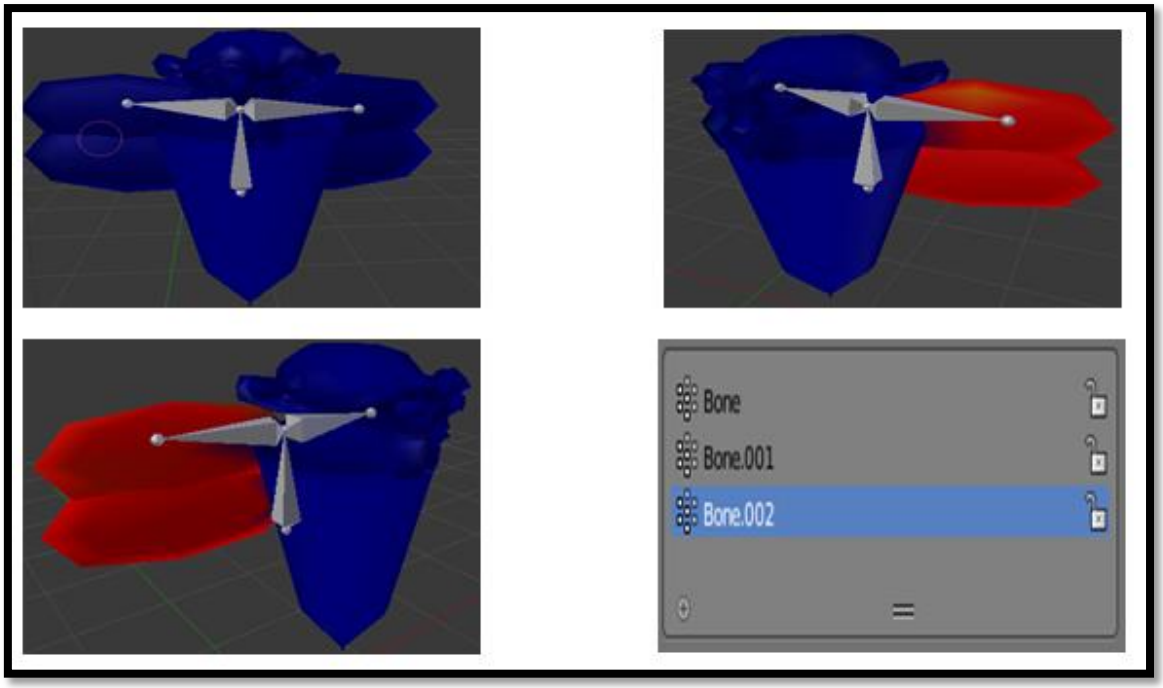


Figura 45: Mapa de peso de los huesos de Enemigo Abeja-Mono

Mecanim es la aplicación que se usa para el desarrollo de animaciones dentro de Unity. Está diseñado para hacer animaciones fluidas de personajes con una interfaz eficiente e intuitiva. Incluye herramientas de creación de máquinas de estados, árboles de mezcla, manipulación de los conocimientos nativos y retargeting automático de animaciones, desde el editor de Unity.

Animaciones

Las animaciones se han hecho a través de Mecanim. Cada imagen mostrada muestra el conjunto de modificaciones hechos, que son las estrellas. Es decir, la estrella en el creador de animaciones habla de modificaciones. Se han hecho en total 4 animaciones. Tres animaciones corresponden a los personajes, y una al enemigo que existe en el laberinto. Después servirán para hacer los controladores de animaciones.

Animación de correr:

Se usa los huesos 3 y 4 correspondientes a las piernas. La posición y la rotación será la contraria en cada pierna. Es decir, si la pierna derecha se encuentra adelantada, la pierna izquierda está atrasada, y viceversa.

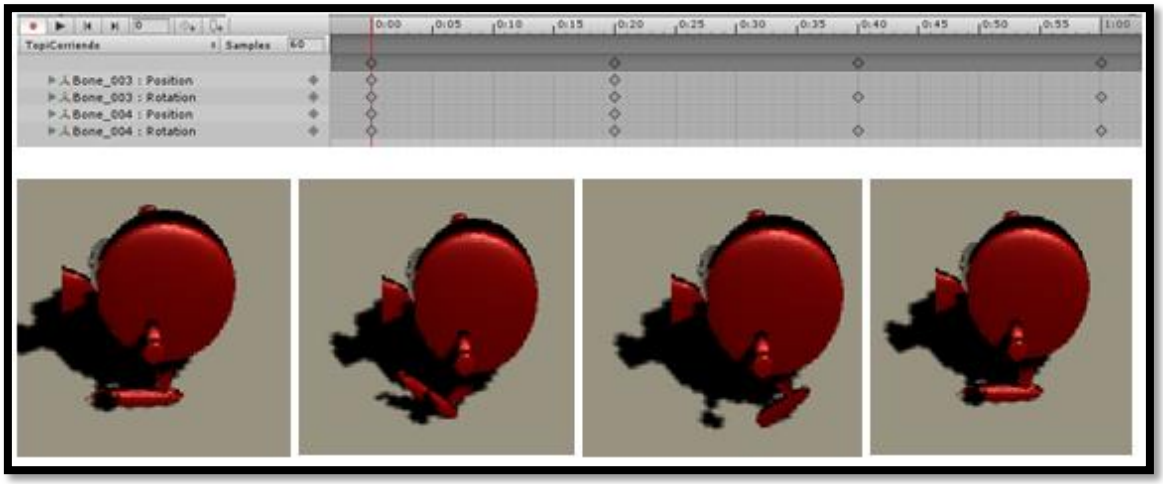


Figura 46: Animación de Correr

Animación de Parado:

Se aplica la rotación de la armadura para tener sin aplicar ningún cambio a ningún hueso ni el cuerpo. Esto lo hacemos para que la animación sea estar parado, sin hacer nada.

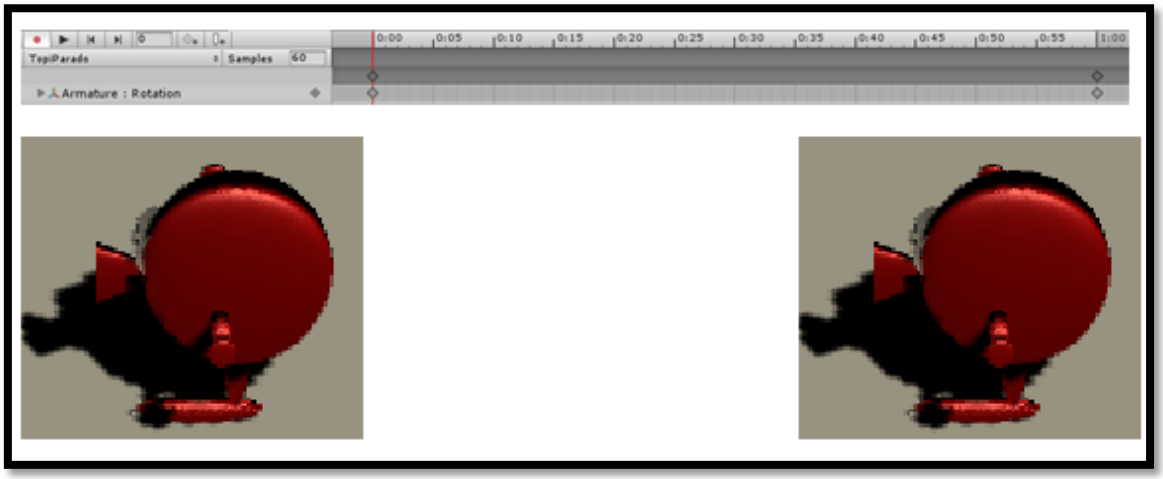


Figura 47: Animación de Parado

Animación de saludo:

Se usa el hueso 1 de la armadura del cuerpo del personaje. Corresponde con el brazo izquierdo. Moverá el brazo hacia arriba y hacia abajo.

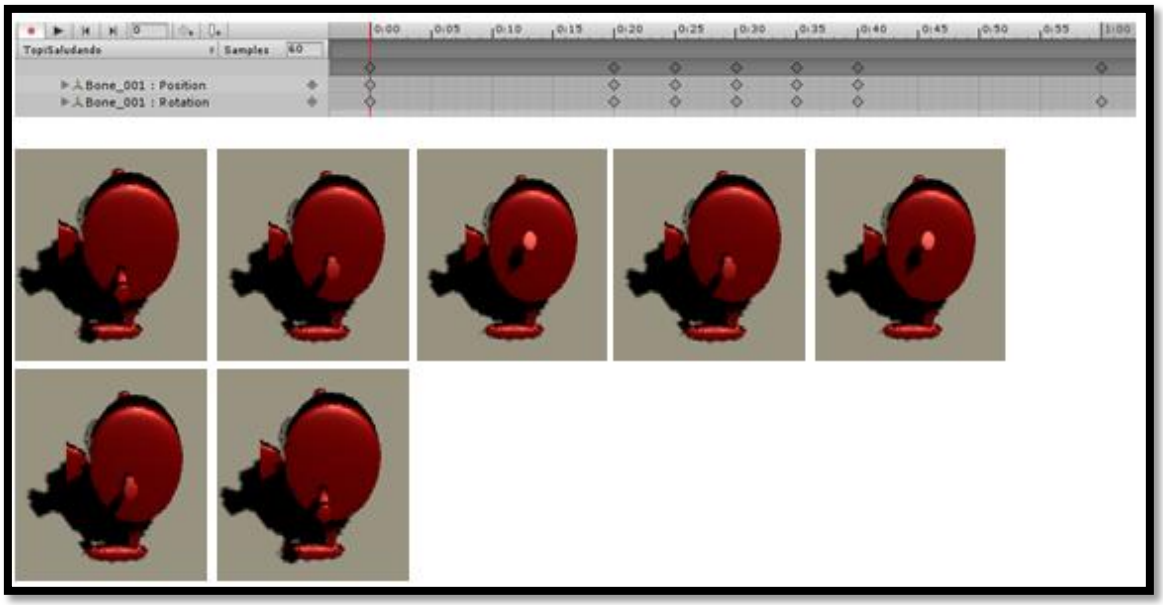


Figura 48: Animación de Saludo

Animación del enemigo:

Usa los huesos 1 y 2 de la armadura del cuerpo del enemigo. La animación consiste en que bata sus alas. Con esta animación y su movimiento se consigue la sensación de vuelo del enemigo.

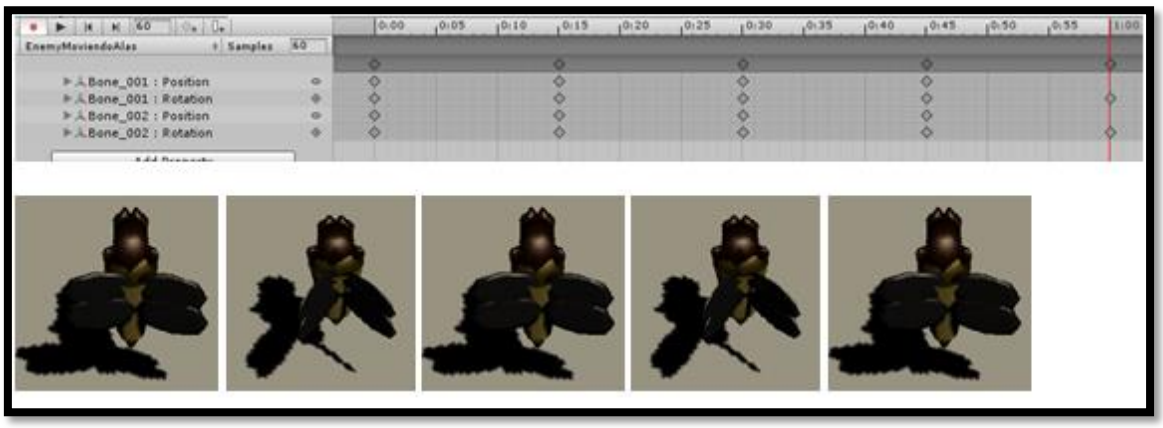


Figura 49: Animación del Enemigo

Una ventaja que tiene el Mecanim de Unity es la reproducción de una animación en cualquier objeto. Es decir, por ejemplo en el pueblo, tenemos distintos personajes, es decir distintos objetos, cada uno con su color, pero al tener el mismo esqueleto, las animaciones hechas para el player, también sirven para los personajes del pueblo, pero no sirve para el enemigo, ya que no tiene el mismo esqueleto.

Una vez hecho esto, hay que meter estas animaciones en controladores de animación, que al final son las que controlarán y ejecutarán las animaciones de los personajes y el enemigo.

Controladores Animación

Topi Prota

Controla la animación del player. Tiene 3 estados u animaciones. El estado parado será el estado por defecto. Para acceder al estado y se ejecute la animación TopiCorriendo tendrá que ser cierta la variable booleana “Running”. Al igual con la animación TopiSaludando pero con la variable “Saludo”. Estas variables se activarán mediante código. El script que lleva la activación y desactivación de las variables es “Player Movement”, el cual se utiliza en las escenas “Laberinto”, “Escondite-Bosque”, “Escondite Inglés” y “Pueblo”.

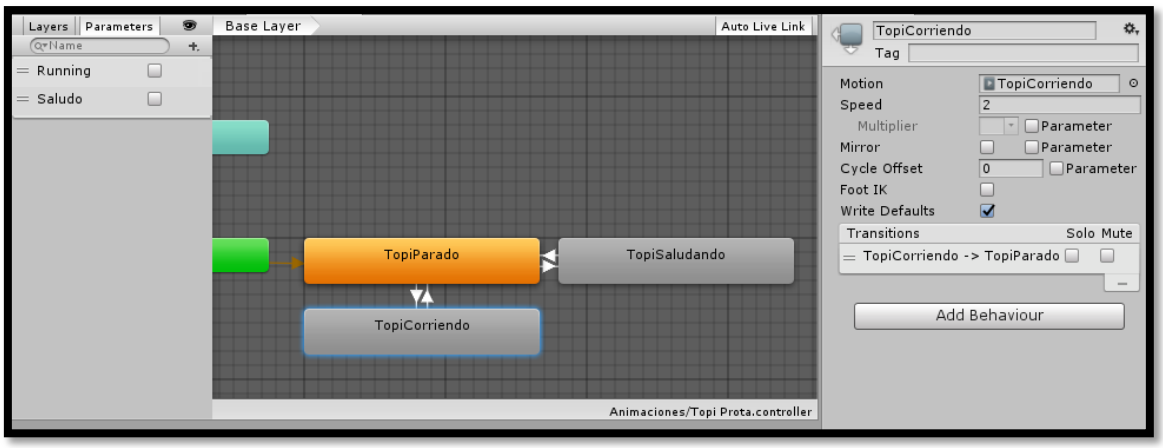


Figura 50: Controlador del Player

Topi Enemigo

Controla la animación del topi enemigo y que hay que atrapar en el minijuego del bosque, es decir el juego del pilla-pilla. Directamente está realizando la animación de correr. No necesita que se cumpla ninguna condición ni acceder mediante código a la animación.

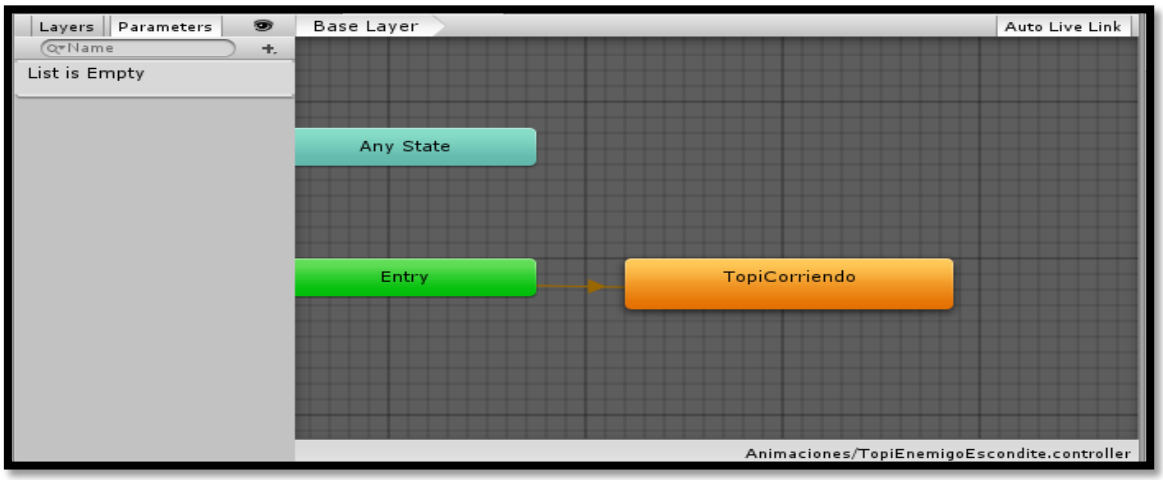


Figura 51: Controlador Topi Enemigo

Enemigo Abeja-Mono

Controla la animación del enemigo en el laberinto, es decir, el juego de "recolección en el laberinto". Directamente está realizando la animación de correr. No necesita que se cumpla ninguna condición ni acceder mediante código a la animación.

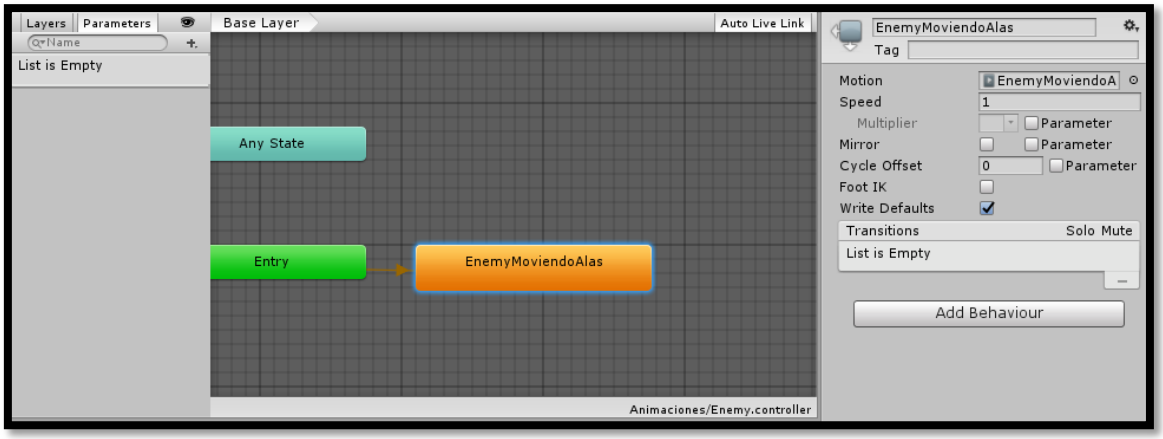


Figura 52: Controlador Enemigo moviendo Alas

Saludar

Controla la animación de los personajes topi que están en el pueblo parados y no se mueven de su posición. Directamente está realizando la animación de parado. Para realizar la acción de saludo usa una variable booleana que se activa mediante código en el script "Textos". No necesita que se cumpla ninguna condición ni acceder mediante código a la animación.

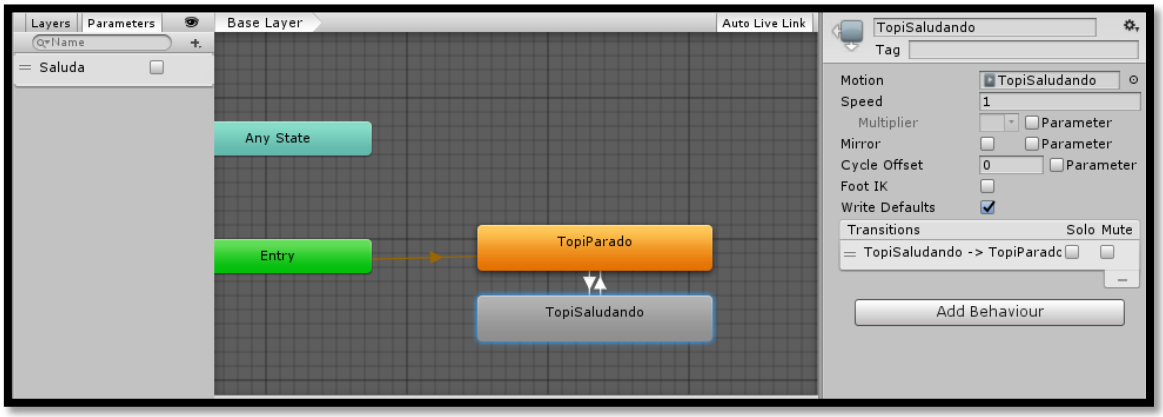


Figura 53: Controlador Saludar

Correr

Controla la animación de los personajes topi que están en el pueblo moviéndose y yendo de un lugar a otro. Directamente está realizando la animación de parado. Para realizar la acción de saludo usa una variable booleana que se activa mediante código en el script "TopiSeMueve". No necesita que se cumpla ninguna condición ni acceder mediante código a la animación.

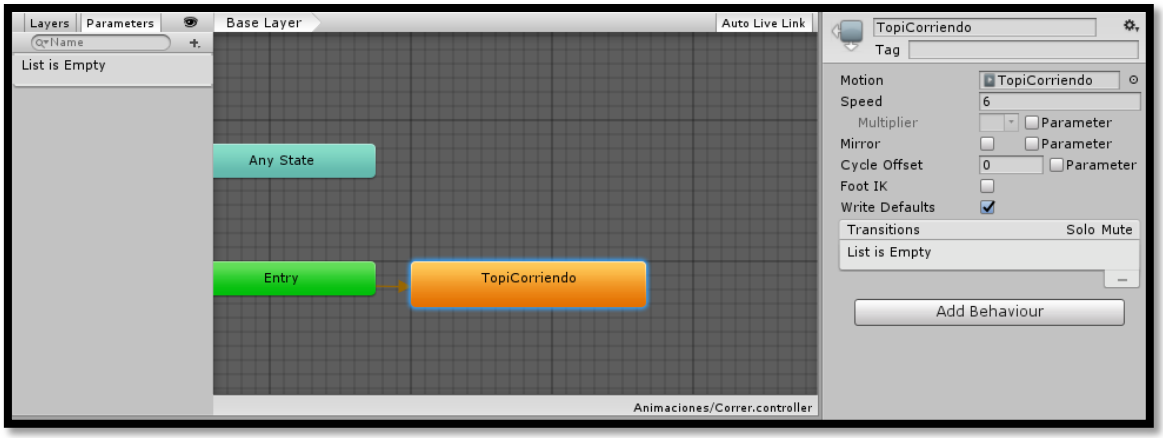


Figura 54: Controlador Correr

Capítulo 5: Aplicación. Manejo.

En esta sección se explica la funcionalidad de la aplicación, mostrándose todas las opciones que dispone y los resultados obtenidos en su ejecución.

Funcionamiento

En primer lugar, el usuario debe instalar la aplicación en su dispositivo móvil. Para ello, debe aceptar algunos permisos como por ejemplo el de almacenamiento, además tiene que tener activado el instalar y ejecutar aplicaciones de origen desconocido. Tras su instalación, se accede a la aplicación por medio del icono.

Figura 55: Icono Móvil

Una vez pulsado este logotipo, el usuario se encontrará con una imagen inicial que se muestra antes de la pantalla principal. Esta será la del motor con el que se ha hecho el videojuego, además saldrá la versión con el que se ha hecho.

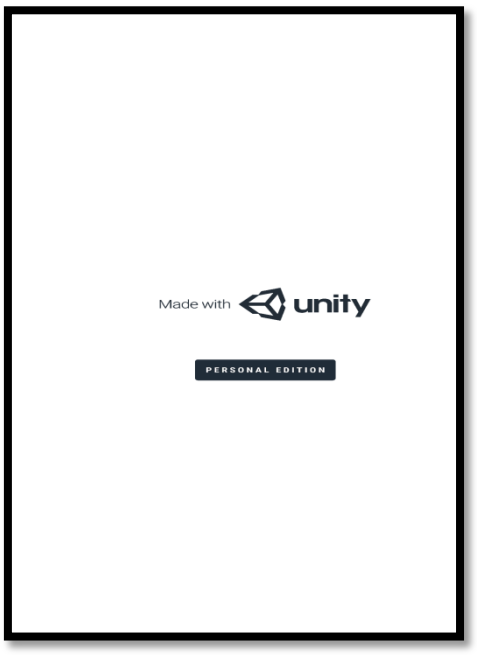


Figura 56: Imagen de Unity

Después nos encontramos con la primera escena ya del videojuego, que se trata del menú principal. En él dependiendo de cómo estemos jugando, si con orientación vertical u horizontal veremos las cosas, de una forma u otra. Se recomienda la orientación horizontal en la mayoría

de móviles y dispositivos, pues no gozan de la resolución óptima para ver todo en la orientación vertical.

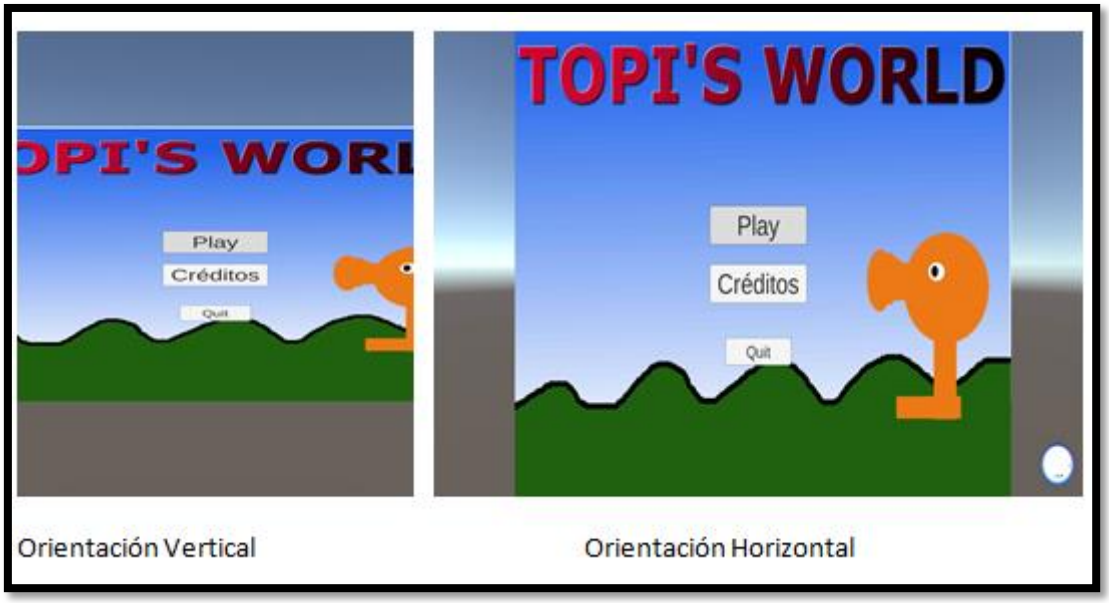


Figura 57: Orientación óptima del móvil

A partir de aquí, entramos en la escena del pueblo. En ella se puede mover al personaje a través de su joystick y navegar por todo el mapa de la escena explorando cada rincón. En esta escena se observan tanto los personajes que saludan y hablan como los que se mueven en alguna dirección o los que plantean mini juegos, estos últimos están situados en ciertas partes limítrofes del mapa, y son identificados a través de un indicador que se ve en la cámara inferior izquierda.

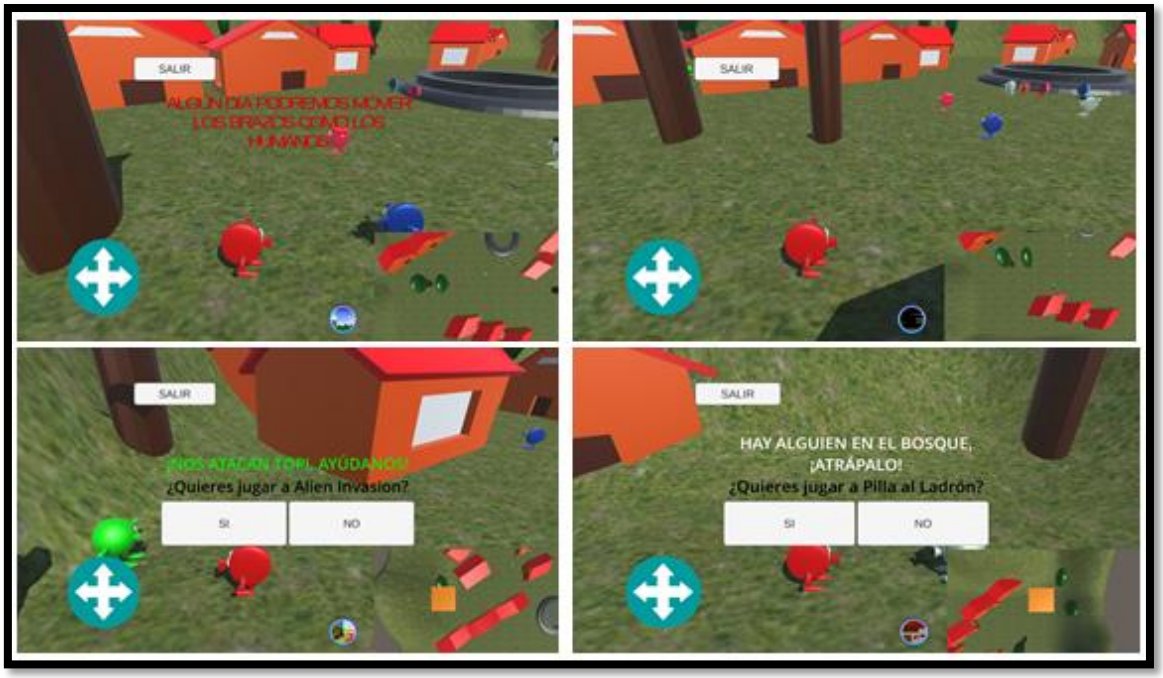


Figura 58: Vistas de Interacción del Pueblo

En la versión en Android se permiten los eventos multitouch, esto quiere decir que se puede mover al personaje, y disparar como se hace en el minijuego de Alien Invasion o en Recolección en el Laberinto.

Figura 59: Vista de juego de Alien Invasion en Android

En todos los minijuegos al perder la partida aparece la siguiente interfaz, que permite reiniciar el minijuego (Reiniciar Partida) o volver al pueblo (Salir).

Figura 60: Vista de GameOver de Alien Invasion en Android

Al igual que Alien Invasion, tenemos las mismas interfaces en los otros minijuegos: Recolección en el Laberinto, Pilla al Ladrón y el Escondite Inglés.

Figura 61: Vistas de los otros minijuegos

Además se observa que el sistema de guardado de puntos funciona perfectamente, guardando la puntuación más alta que ha obtenido.

CONCLUSION Y LINEAS FUTURAS

Conclusión

Se ha conseguido realizar satisfactoriamente el desarrollo de un videojuego. El videojuego tiene una serie de características como son: el uso de librerías gráficas para 3D del motor Unity y algoritmos varios que plantean mecánicas de juego.

Además se ha incrementado por parte del alumno el aprendizaje acerca de la programación orientada a objetos.

El gran avance gráfico que se está produciendo en todos los entornos acoge también a Android. Los dispositivos móviles forman parte de nuestro día a día en mayor medida para realizar tareas de comunicación, pero también van adaptándose a las nuevas tecnologías que se presentan.

Durante el desarrollo de los diferentes mini juegos se han planteado retos como son la interactividad entre los distintos objetos, mandándose mensajes, la máquina de estados desarrollada en el escondite inglés. Para ello, hubo momentos de descanso para informarme y documentarme para poder solucionar esos problemas.

Al desarrollar este videojuego, se llega a la conclusión de que es ideal para entender los conceptos sobre programación orientada a objetos, y aplicar distintos áreas de ingeniería de telecomunicaciones. Además del interés aún mayor que ha generado en mí el aprender más acerca del área de los videojuegos y adentrarse más en este mundo.

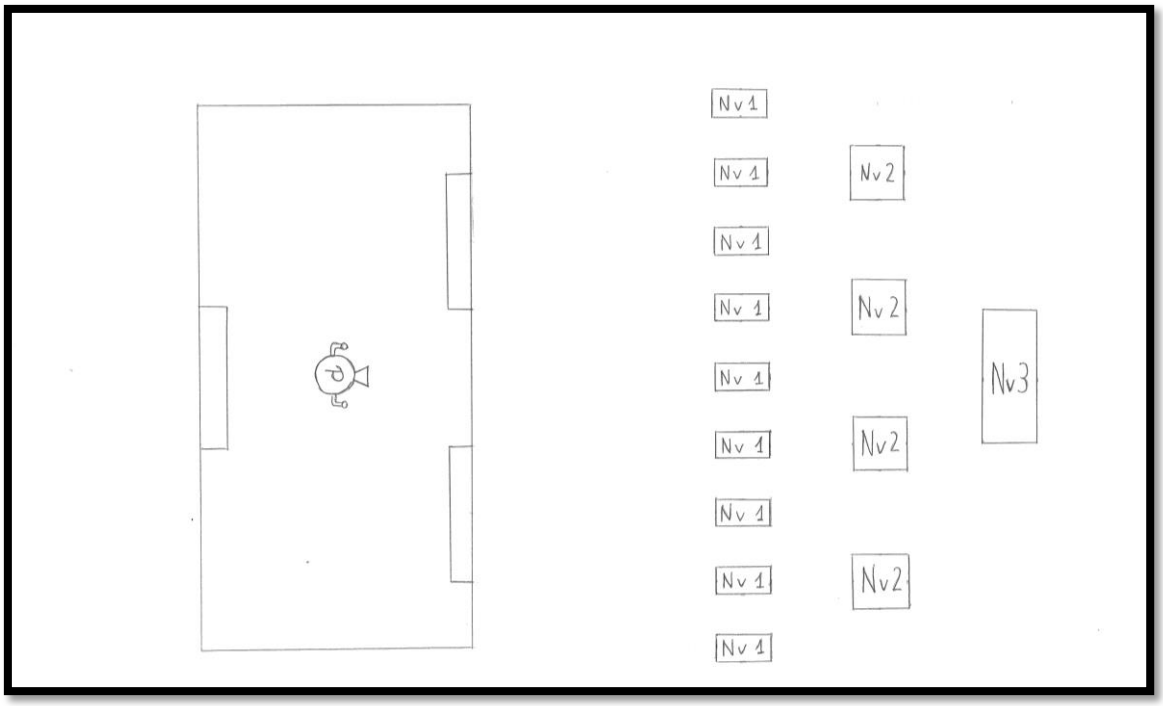
Líneas Futuras

En cuanto a las líneas futuras para este trabajo se puede tener en cuenta lo siguiente:

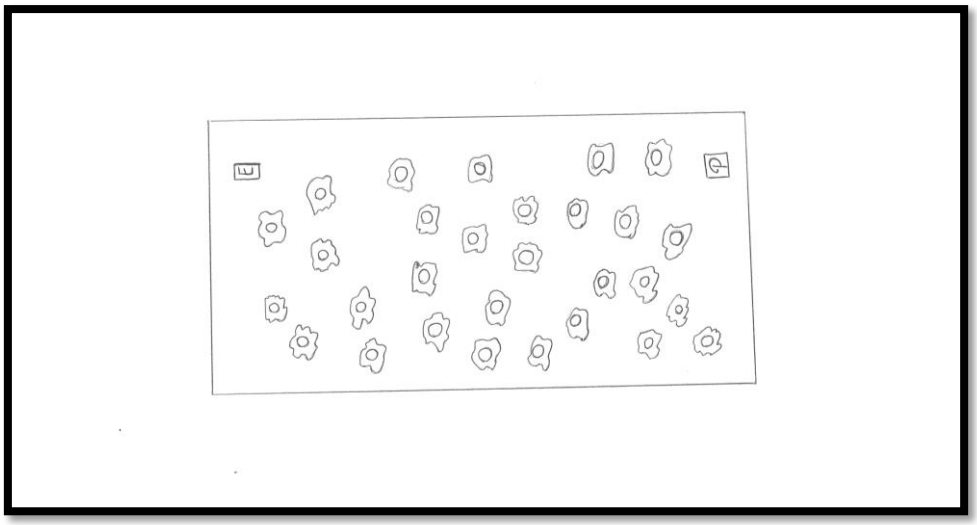
- Sacar más partido a los minijuegos, sobre todo para la plataforma móvil, en el que se le puede introducir un Player VS Player estableciendo redes de comunicación.
- Es posible que la aplicación en el pueblo vaya lento o con cierto retardo debido a las características de Unity, podría hacerse una versión que careciera de esa escena y enseñar solo los minijuegos para poder acceder a ellos fácilmente en equipos o móviles no tan potentes.
- Otra posible mejora para la escena del pueblo es añadir cosas que inspeccionar, ya que se trata de un mundo abierto.
- Introducir conceptos de customización de personajes.
- Introducir nuevos niveles o minijuegos.

II PLANOS

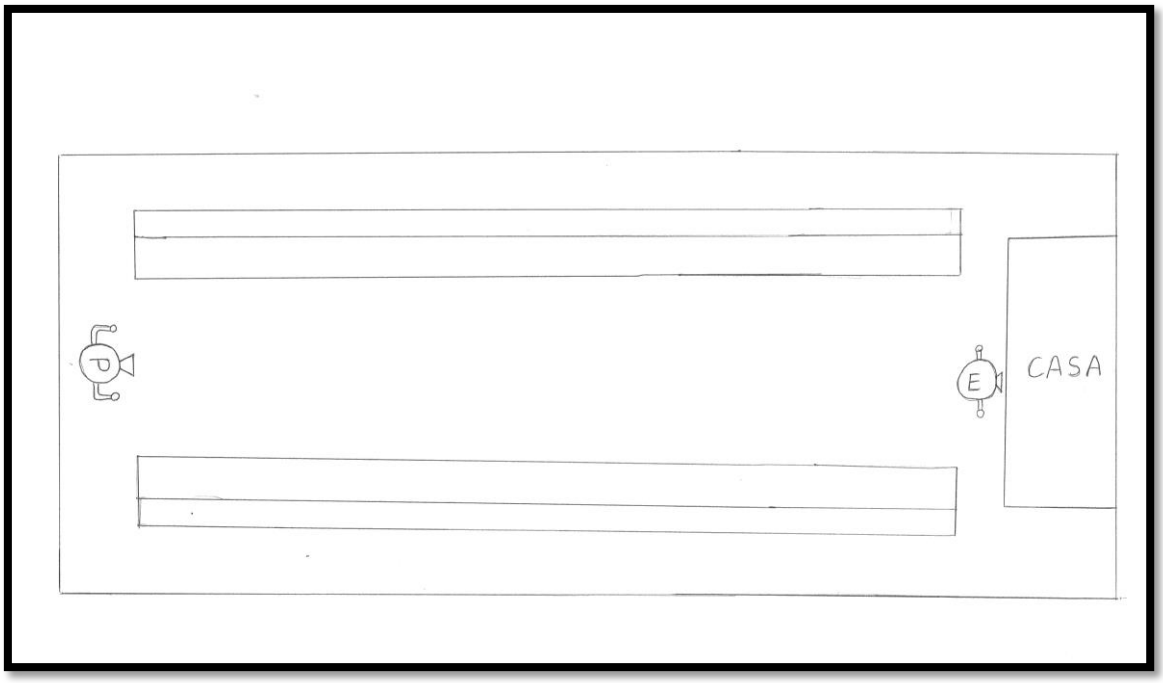
JUEGO ALIEN INVASION



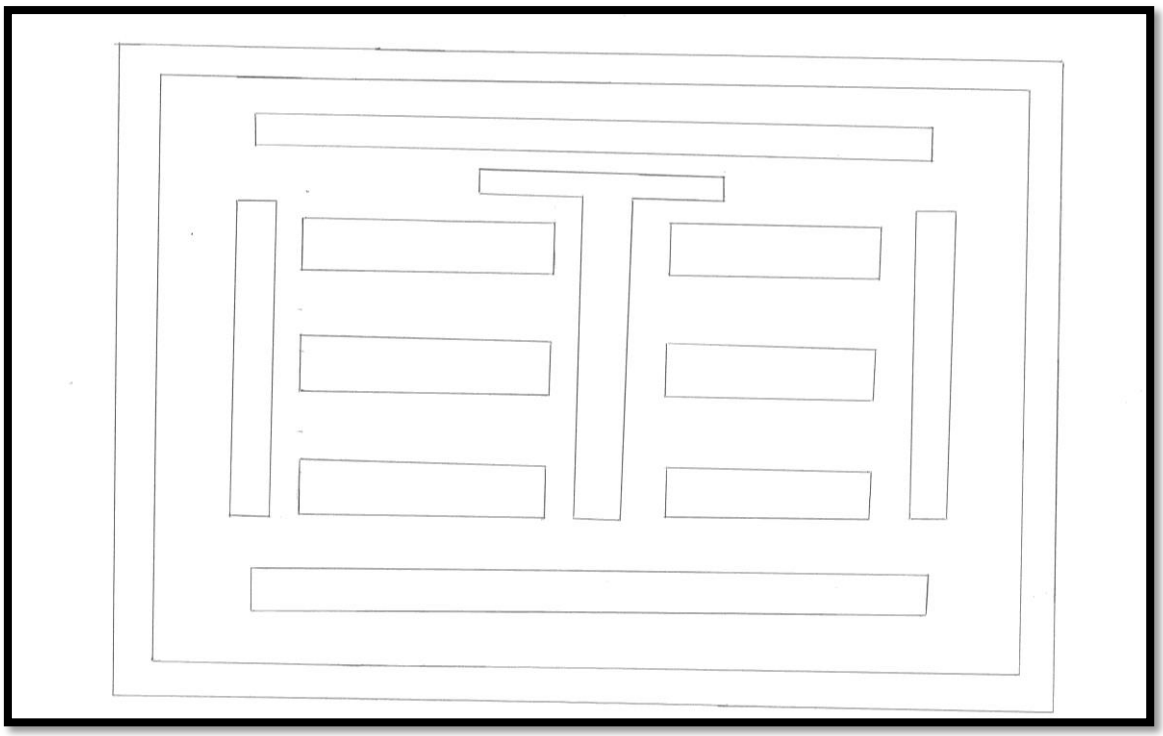
JUEGO PILLA AL LADRÓN



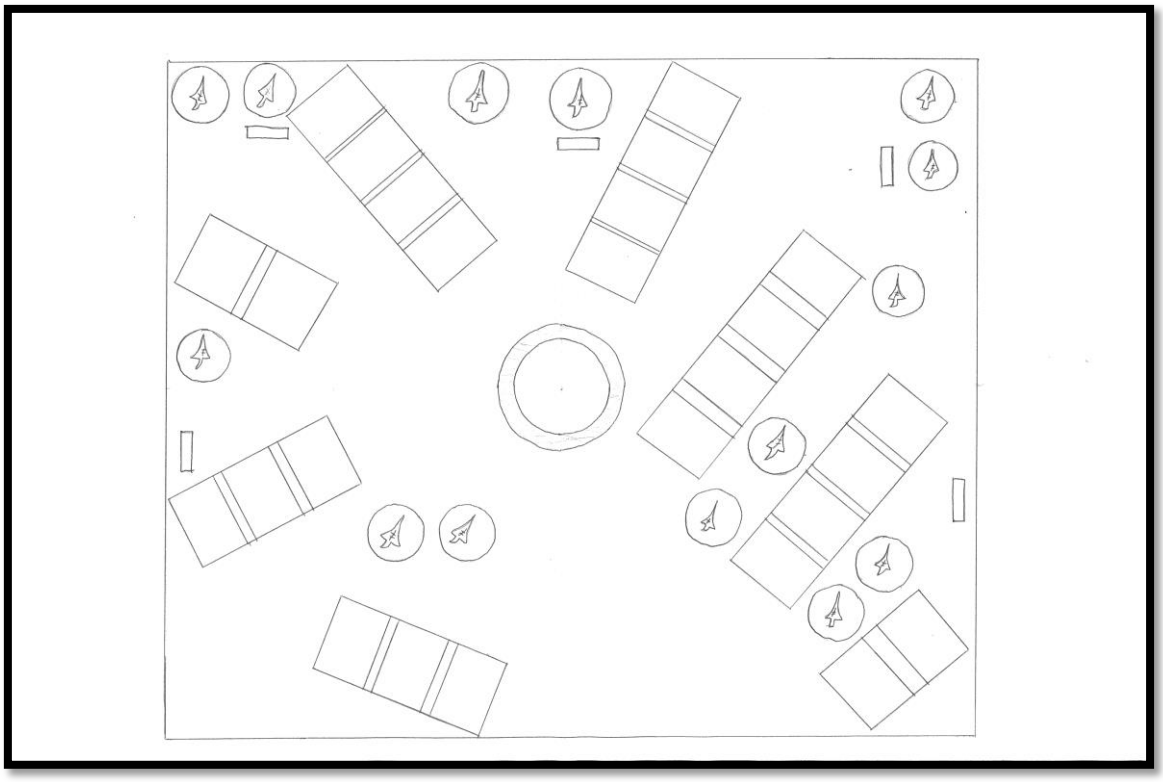
JUEGO ESCONDITE INGLÉS



JUEGO RECOLECCIÓN DE FRUTAS



PUEBLO



III PLIEGO DE CONDICIONES

PLIEGO DE CONDICIONES

En este apartado se muestra tanto el software como el hardware utilizados para llevar a cabo este proyecto.

HARDWARE

En cuanto a hardware se refiere se requieren un ordenador portátil y un dispositivo móvil. Vemos a continuación las características de cada uno.

Portátil

HP AM080 i5

Figura 62: HP AM080 i5

PROCESADOR

Core i5 de Intel de doble núcleo a 2,4 GHz.

MEMORIA

8 GB de Memoria RAM

SOPORTE GRAFICO Y VIDEO

HD Graphics 3500 de Intel con 1GB dedicado.

PANTALLA

Pantalla panorámica brillante retro iluminada por LED de 17 pulgadas.

TAMAÑO Y PESO

27,4 cm (largo) x 41,53 cm (ancho) x 3,18 cm (alto mín.) / 3,7 cm (alto máx.)

Peso: 3.72 kg

CONEXIONES Y EXPANSIONES

1xToma de corriente.

1xPuerto Gigabit Ethernet.

3x puerto USB 2.0 (hasta 480 Mb/s).

1xEntrada/salida de audio.

1xRanura para tarjetas SD.

SISTEMA OPERATIVO

Windows 10 Home

BATERIA

Batería integrada de polímeros de litio de 63,5 vatios por hora.

Toma de corriente.

Dispositivo Móvil

Sony Xperia M4 Aqua

Figura 63: Sony Xperia M4 Aqua

CÁMARA Y VIDEO

Cámara de 13 megapíxeles con enfoque automático

Zoom digital de 4 aumentos

Grabación de vídeo en Full HD (1080p)

HDR para fotos

Cámara frontal de 5 MP, HD 720p para las capturas de la cámara

Captura de imágenes (formato de archivo compatible: JPEG)

Reproducción de imágenes (formatos de archivo compatibles: BMP, GIF, JPEG, PNG y WebP)

Grabación de vídeos (formatos de archivo compatibles: 3GPP y MP4)

Reproducción de vídeo (formatos de archivo compatibles: 3GPP, MP4, Matroska, AVI, Xvid y WebM)

PANTALLA

Pantalla de 5" (1.280 x 720 píxeles)

PROCESADOR (CPU)

Qualcomm MSM8939 a 1,5 GHz

GPU Adreno 405

Ocho núcleos y 64 bits

MEMORIA Y ALMACENAMIENTO

4 GB de RAM

Memoria flash de hasta 8GB*****

Tarjeta microSD™ de hasta 200 GB

SONIDO

Tecnología de sonido envolvente 3D de Sony (VPT)

Clear Audio+: software de mejora del sonido

Clear Bass™

Ecualizador manual

Tecnología xLoud™

SOFTWARE

El software usado para el desarrollo del proyecto ha sido el siguiente:

UNITY

Figura 64: Logotipo Unity

Unity ha sido el programa y motor de videojuegos para videojuegos en 3D con el que he creado el videojuego. Tiene un gran potencial.

BLENDER

Figura 65: Logotipo Blender

Blender ha sido utilizado con el propósito de modelar los objetos que se iban a usar en el videojuego y hacer el rigging de los personajes.

FL STUDIO

Figura 66: Logotipo FL Studio

FL Studio ha sido el programa usado para la creación y modificación de la banda sonora creada y usada en el videojuego.

IV PRESUPUESTO

Presupuesto

Para determinar el presupuesto del videojuego se dividirá en las siguientes partes:

- Hardware: Costes referentes a los equipos utilizados.
- Software: Recoge los costes de los programas informáticos. Sin embargo, tenemos software utilizado totalmente gratuito, con lo cual ese software será nombrado pero no incluido en el presupuesto.
- Trabajo del ingeniero: Agrupa los honorarios y gastos del ingeniero.
- Presupuesto final: Se calcula a partir de los subtotales anteriores, incluye un 21% de IVA.

Los costes de Hardware, Software y equipamiento se calculan mediante la fórmula de amortización:

$$Costes = \frac{Meses\ de\ equipo}{Periodo\ de\ Amortización} * Coste\ del\ equipo * Porcentaje\ de\ Uso$$

Siendo:

- Meses de uso de equipo: El número de meses de uso del equipo durante el desarrollo del proyecto.
- Periodo de amortización: Vida útil del equipo. Normalmente en equipos informáticos y software es de aproximadamente 36 meses.
- Coste del equipo: Precio del equipo en Euros (€).
- Porcentaje de Uso: Porcentaje de uso con valores de 0 a 1 referido a la utilización del equipo especificado en relación al tiempo en el proyecto.

HARDWARE					
Hardware	Precio	Meses Uso	Amortización (meses)	Uso	Total
Ordenador	800	6	36	100	133,33
Disp. Móvil	240	6	36	100	40
Subtotal Hardware					173,33

SOFTWARE					
Software	Precio	Meses Uso	Amortización (meses)	Uso	Total
Unity	125	5	36	83	14,41
Fl Studio	200	1	36	17	0,94
Subtotal Software					15,35

Realización Del Proyecto	Horas
Planificación	20
Documentación	80
Diseño del proyecto	160
Desarrollo del proyecto	240
TOTAL	500

Costes Humanos			
Proyecto	Horas	Precio/Hora	Total
Realización del proyecto	500	20	10000

Presupuesto Total	Coste
Hardware	173,33
Software	15,35
Costes Humanos	10000
Subtotal	10188,69
Costes 21%	2139,624
Total	**12328,31**

APÉNDICES

BIBLIOGRAFÍA

[1] Manual de Unity:

https://docs.unity3d.com/Manual/index.html?_ga=2.76106589.936418997.1496332893-1010319241.1480270997

[2] Canal de Youtube, Nodician:

https://www.youtube.com/user/nodician/videos

[3] Canal de Youtube La Cueva de L Draven

https://www.youtube.com/channel/UC49ACIHmz1yvHqdDE4Y_foQ/videos

[4] Canal de Youtube Hagamos Videojuegos

https://www.youtube.com/user/juande/videos

[5] Canal de Youtube, game3dover

https://www.youtube.com/user/game3Dover/videos

[6] Historia de los videojuegos:

https://www.elotrolado.net/wiki/Historia_de_los_videojuegos

[7] Curso Udemy: Unity 5 Creando un juego para PC

https://www.udemy.com/curso-unity-5-creando-un-juego-para-pc/learn/v4/content

[8] Curso Udemy: Tu primer videojuego 2D multiplataforma en Unity 5

https://www.udemy.com/unity-5-primer-videojuego-2d-multiplataforma/learn/v4/content

[9] Wikipedia, Lenguaje C#

https://es.wikipedia.org/wiki/C_(lenguaje_de_programaci%C3%B3n)