ANIMAL WORLD

VIDEOJUEGO EDUCATIVO EN UNITY
PARA EL DESARROLLO DE
VOCABULARIO ORIENTADO A PÚBLICO INFANTIL

JAVIER BAIZÁN FERNÁNDEZ

Universidad de Valladolid

UNIVERSIDAD DE VALLADOLID
ESCUELA DE INGENIERÍAS INDUSTRIALES

Grado en Ingeniería en Diseño Industrial y
Desarrollo de Producto

VIDEOJUEGO EDUCATIVO EN UNITY PARA EL DESARROLLO DEL VOCABULARIO ORIENTADO A PÚBLICO INFANTIL

Autor:
Baizán Fernández, Javier

Tutor:
Escudero Mancebo, David

Departamento:
Informática/Ciencia de la Computación

Septiembre 2020

ÍNDICE

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Este proyecto educativo toma su inspiración del clásico juguete para niños sobre los sonidos de los animales. En este caso, lo que se pretende es trasladar ese concepto al terreno de los videojuegos y la realidad virtual, pero de manera mucho más profunda gracias a las posibilidades que estas dos tecnologías nos ofrecen.

La intención es traer el mundo animal hacia los niños de una manera más inmersiva, de forma que estos puedan explorar los diferentes hábitats de los animales y sobre todo conocer más sobre ellos. Todo ello en un entorno 3D y con las ventajas que ofrece la realidad virtual.

Para la realización de este videojuego se ha empleado Unity, plataforma de la que hablaremos más adelante, junto con otros programas como Cinema 4D (para el modelado y las animaciones) y FL Studio (para el audio).

En el capítulo 2 hablaré brevemente del trasfondo de los videojuegos y la importancia que pueden tener en el plano educativo. En el 3 explicaré en qué consiste la realidad virtual y su viabilidad de uso en niños. En el cuarto expondré la idea del proyecto y su diseño. En el capitulo 5 explicaré en detalle el desarrollo e implementación de la idea introducida en el capítulo anterior. Por último en los capítulos posteriores (6 y 7) hablaré respectivamente de las pruebas realizadas y de las conclusiones personales que he podido sacar del desarrollo de este proyecto.

VIDEOJUEGOS

2.1 QUÉ SON LOS VIDEOJUEGOS

Podríamos definir al videojuego como un juego electrónico que necesita de un dispositivo de video y un controlador, mediante el cual el jugador podrá interactuar con lo que aparece en pantalla. El dispositivo electrónico que contiene el videojuego puede ser un ordenador, un smartphone o una videoconsola, entre otras. También se puede jugar a videojuegos contenidos en la red, con lo que solo necesitaríamos una conexión a internet.

El uso del controlador, también conocido como gamepad, es lo que hace posible que el usuario se involucre activamente con el contenido. Este envía órdenes a la videoconsola y estas se ven reflejadas en la pantalla.

Los videojuegos permiten recrear entornos y simular experiencias en los que el jugador puede controlar a uno o varios personajes, para conseguir uno o varios objetivos siguiendo unas reglas determinadas. En el caso de los videojuegos en realidad virtual, pasamos de controlar el personaje, a convertirnos en él. Esto nos permite interactuar de una forma mucho más directa e inmersiva con los diferentes elementos del juego. Trataremos el tema de la realidad virtual más adelante.

Para el desarrollo de un videojuego se necesitan conocimientos de informática, diseño, sonido e incluso actuación, entre otros. Por ello las grandes empresas desarrolladoras de videojuegos suelen trabajar con varios profesionales de cada campo. [1] [2]

2.2 LA RELEVANCIA DE LOS VIDEOJUEGOS EDUCATIVOS

Aunque en un inicio, los videojuegos fueron pensados para el entretenimiento, estos también pueden ser una fuerte herramienta de aprendizaje, y no solo para niños, si no para cualquier edad.

Un punto importante, y en el mejor de los casos, es que el conocimiento que se adquiere de manera implícita, o, en otras palabras, el usuario no se percata de que está aprendiendo a medida que avanza por el videojuego. Además, también permite desarrollar habilidades que quizás de otra forma serían más difíciles de adquirir.

Los videojuegos suelen recompensar de alguna manera al jugador cuando se resuelve un problema. Esto anterior, se puede comparar en cierto modo con el entorno del aula, donde a los estudiantes se les examina para luego obtener una calificación (recompensa).

Dentro de los videojuegos educativos, podemos hablar de los “serious games”. Se utilizan para educar acerca de una determinada materia o practicar una habilidad concreta. Dentro de este entorno podemos encontrar tanto videojuegos dedicados a entrenar bomberos o policías, como a enseñar matemáticas o idiomas, entre otros.

En el caso del aprendizaje de idiomas, los jugadores se ven obligados a aprender una determinada lengua para poder completar los diferentes niveles, o incluso para poder entender el menú. Por ello, se trata de uno de los temas más importantes.

Mediante los videojuegos educativos, podemos aprender de una forma novedosa y adaptada a nuestro tiempo. Son interesantes ya que, como muchos defienden, pueden ser una herramienta tan divertida como didáctica, lo que, por otro lado, concede una motivación extra que favorece que se juegue de manera regular.

Son muchos los beneficios que pueden proporcionar los videojuegos a nivel educativo. Enumerando algunos, podemos decir que:

- Favorecen nuestra capacidad de análisis y reflexión, al prestar atención a su historia y nuestro objetivo.
- Mejorar nuestra velocidad de respuesta y agilidad mental, si esta es un factor crítico dentro del juego.
- Fomentan el desarrollo de la imaginación.
- Favorecen el aprendizaje del trabajo en equipo, sobre todo si se trata de juegos online.

En resumen, es innegable que los videojuegos son una potente arma de aprendizaje, y, aunque un consumo excesivo de los mismos puede llegar a ser contraproducente, para eso están los educadores, que se encargarán de sacar el máximo partido a la capacidad educativa de los videojuegos. [3] [4]

2.3 UNITY

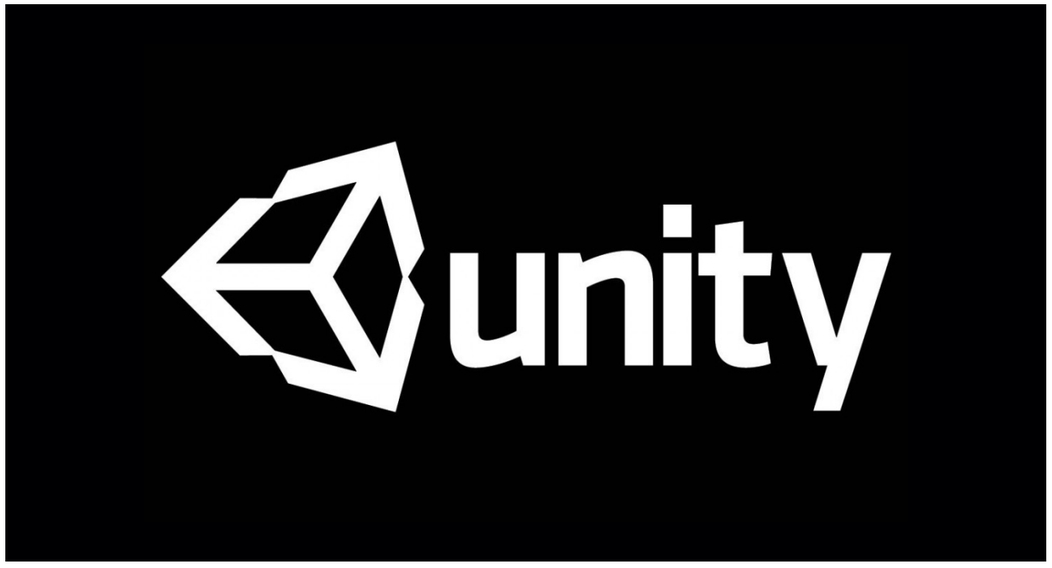
El motor utilizado en el desarrollo de este videojuego es Unity, que es una de las plataformas de creación a tiempo real más utilizadas en el desarrollo de videojuegos de todo el mundo.

Aparece en 2005 por parte de Unity Technologies. Aunque en un inicio se pensó para el desarrollo de videojuegos en Mac de forma exclusiva, su enorme éxito propició que se invirtiera más en su desarrollo y se expandiera a más plataformas.

La idea de la que se partió a la hora de crear Unity era la “democratización de los videojuegos”, o, en otras palabras, hacer más accesible el desarrollo de contenidos interactivos a personas de todo el mundo.

Dentro de Unity se pueden desarrollar tanto videojuegos en 2D como en 3D, además de que es compatible con la gran mayoría de plataformas existentes en el mercado. De esta manera es posible desarrollar proyectos para cualquier dispositivo móvil, videoconsola u ordenador. También ofrece soporte para las principales plataformas de realidad virtual existentes, tales como Oculus o GearVR.

Desde que se creó hasta ahora, Unity ha avanzado mucho con sus actualizaciones, situándose a la altura de otras plataformas líderes en el mercado. A destacar, lo que considero es su mayor virtud, es que sea accesible a todo el mundo gracias a su versión gratuita disponible en su página web, versión con la que se ha desarrollado este videojuego. [5] [6].



REALIDAD VIRTUAL

3.1 REALIDAD VIRTUAL

La realidad virtual (VR) consiste en la inmersión sensorial del usuario en un entorno virtual, como su propio nombre indica. Para ello, tenemos que utilizar las gafas de realidad virtual y sus accesorios (auriculares, guantes, etc.). Esta tecnología permite la creación de mundos ficticios de los que podrás formar parte.

La aparición de la realidad virtual supone uno de los cambios tecnológicos más importantes de los últimos años. Supondrá un antes y un después en la forma en que consumimos contenidos multimedia.

Sus aplicaciones varían desde su uso en el aprendizaje de operaciones quirúrgicas, hasta su uso en inmobiliarias para tours virtuales por sus inmuebles, pasando por los videojuegos. [7]

3.2 REALIDAD VIRTUAL EN NIÑOS

Una de las preguntas que deben hacerse a la hora de realizar un videojuego en realidad virtual para niños es, ¿puede un niño usar unas gafas de realidad virtual sin problema?

En una conferencia en 2013, cuando aún se desaconsejaba su uso por niños, Palmer Luckey, fundador de Oculus, una de las principales compañías dedicadas a la realidad virtual, aclaró que esto se debe a la diferencia de la separación de los ojos frente a la de las lentes y a la distancia a la que se encuentran de los ojos. Llegaron a la solución tiempo después mediante un ajuste manual por software de las lentes.

Hoy en día se pueden encontrar fácilmente gafas de VR que permitan un ajuste manual de las lentes, para poder acomodarlas a los ojos de quien las use, por lo que el problema mencionado en el párrafo anterior quedaría resuelto. Aun así, tampoco es recomendable un uso prolongado de las mismas.

La principal consecuencia de un uso prolongado por un niño es la miopía. Cuanto más tiempo use unas gafas VR, más problemas de vista podría causarle. Podríamos decir que, aunque no sean extremadamente peligrosas para los niños, cuanto menor sea su edad, menos tiempo deberían estar expuestos a ellas.

Como dato, Jeremy Bailenson, director del Laboratorio de Interacción Humano-Virtual de Stanford, solo ha permitido que su hija de 6 años use las gafas en 4 ocasiones, cada una de no más de 5 minutos de duración.

Por otro lado, sabemos que la realidad virtual ya se utiliza en las clases de algunos colegios, por lo que, contestando a la pregunta de si los niños pueden utilizar gafas de realidad virtual, podemos decir que sí, pero con un uso controlado. Siempre que los educadores sean conscientes de todo lo anterior, no supondrá ningún problema. [8] [9] [10]

DISEÑO DEL JUEGO

Como ya se ha mencionado anteriormente, este proyecto consiste en un videojuego educativo para niños sobre los animales. La mecánica básica del juego consistirá en responder a las preguntas que se nos planteen para así progresar por el juego.

Nuestro guía a través del juego y también quien nos planteará las preguntas que nos vayamos encontrando será un alienígena. Este nos dirá que necesita ayuda para saber más sobre la fauna de nuestro planeta y nos pedirá que recopilemos información sobre los diferentes animales que existen en cada hábitat.

El juego se divide en hábitats. En el proyecto base, hay un total de siete hábitats a explorar. Cada uno de ellos se divide en distintos niveles que consisten en responder a las preguntas que nos plantea el alienígena sobre los animales que encontremos.

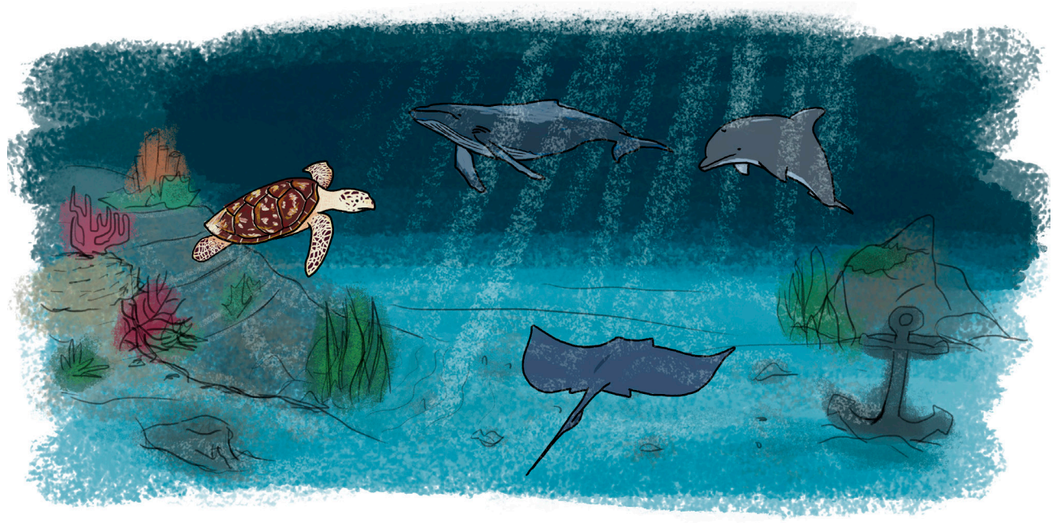
Lo interesante de este tipo de estructura de proyecto, es que es expandible, y si en un momento se llegara a desarrollar del todo, podrían añadirse más hábitats y animales por medio de actualizaciones. Incluso podría diseñarse un sistema que permitiera decidir al usuario qué hábitats instalar de entre los disponibles, como si se tratara de juego por bloques.

4.1 LOS HÁBITATS

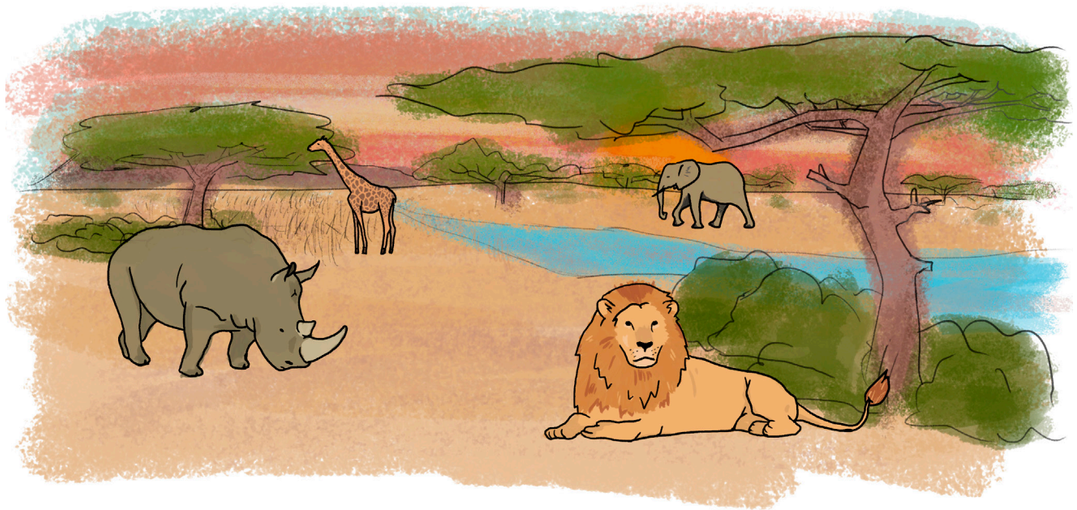
El bosque - Búho, ciervo, zorro y oso

El desierto - Coyote, camello, serpiente de cascabel y buitre

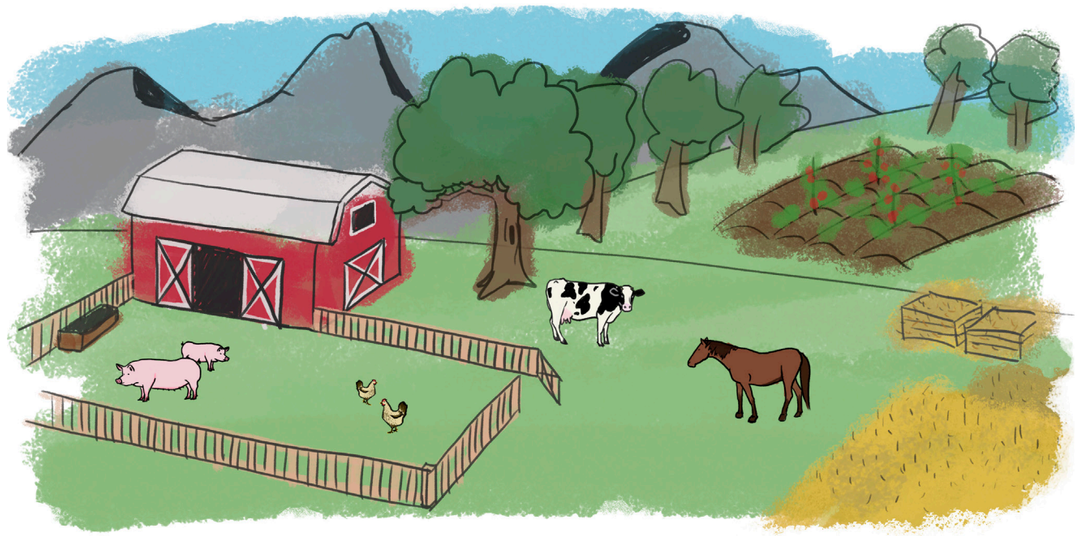
El océano - Tortuga, ballena, manta raya y delfín



La sabana - Rinoceronte, jirafa, león y elefante

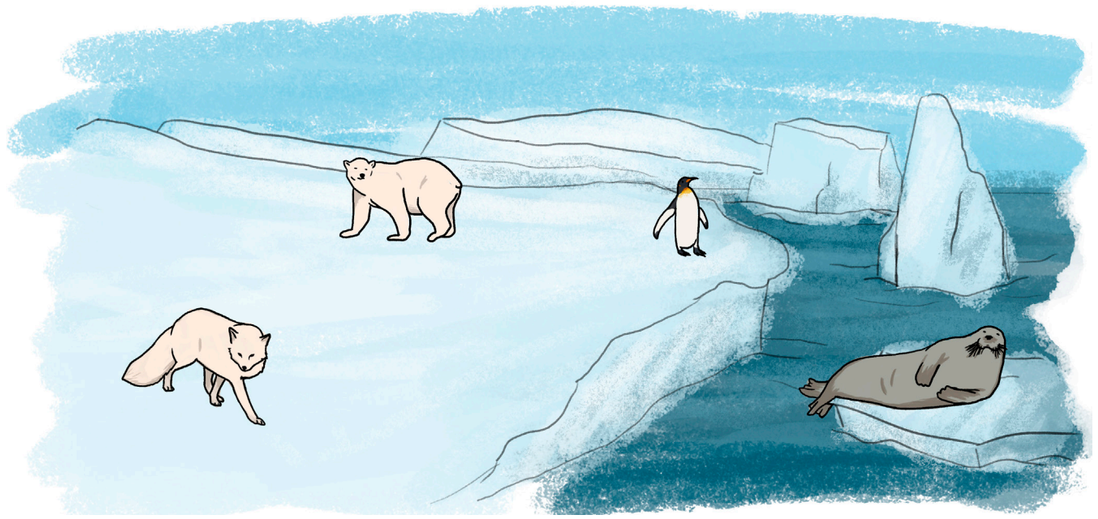


La granja - Cerdo, gallina, vaca y caballo



La jungla - Mono, tigre, anaconda y pantera

El polo - Zorro ártico, oso polar, pingüino y foca



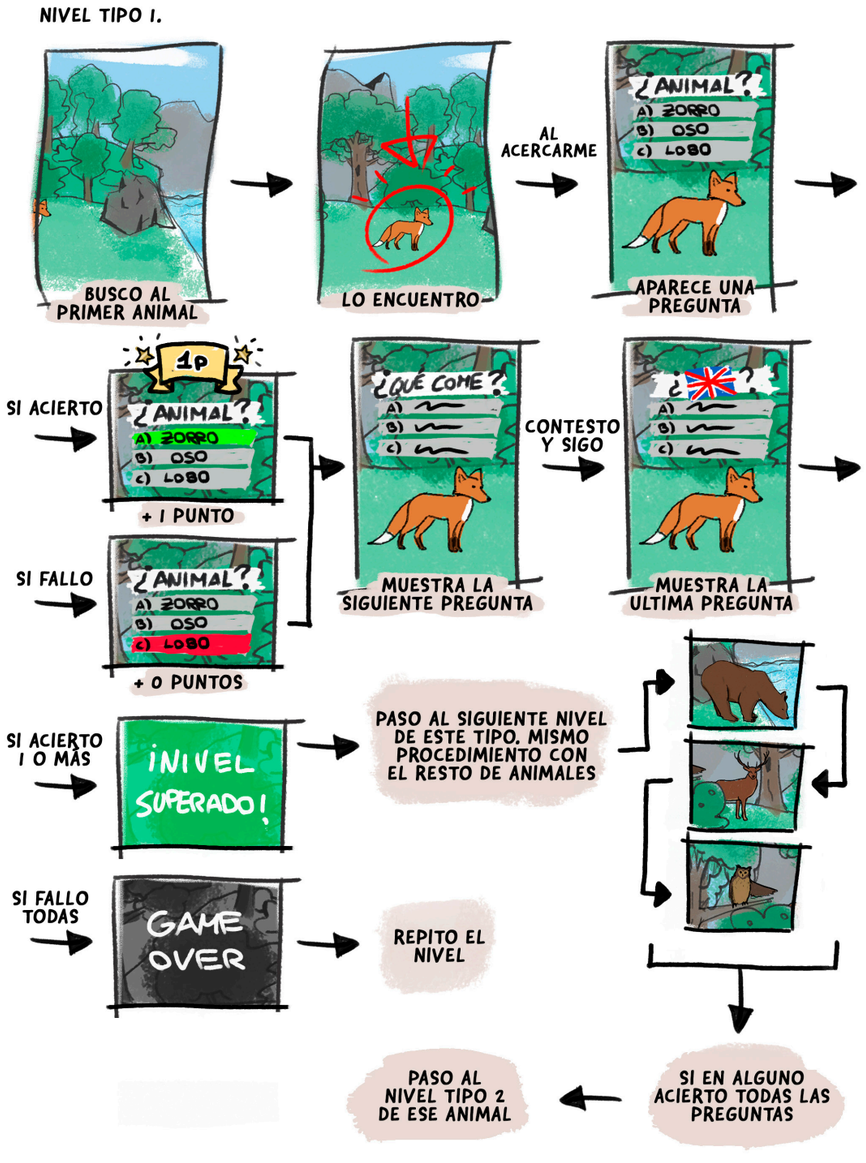
No existe un orden concreto en el que visitarlos ni será necesario desbloquear ninguno de ellos, por lo que el jugador podrá decidir qué hábitat visitar en cada momento.

4.2 LOS NIVELES

Hay tres tipos de niveles diferentes. Cada uno se repetirá varias veces dentro del mismo hábitat, pero con preguntas distintas. A continuación, se expondrá en qué consiste cada uno de ellos con el apoyo de unas ilustraciones a modo de storyboard.

4.2.1 TIPO 1

El primer nivel de cada hábitat sería así:



Aparece un animal en el escenario que tenemos que encontrar. Cuando lo hacemos se nos plantearán tres cuestiones sobre este. A modo de test, se nos propondrán tres opciones, de las cuales tendremos que escoger una.

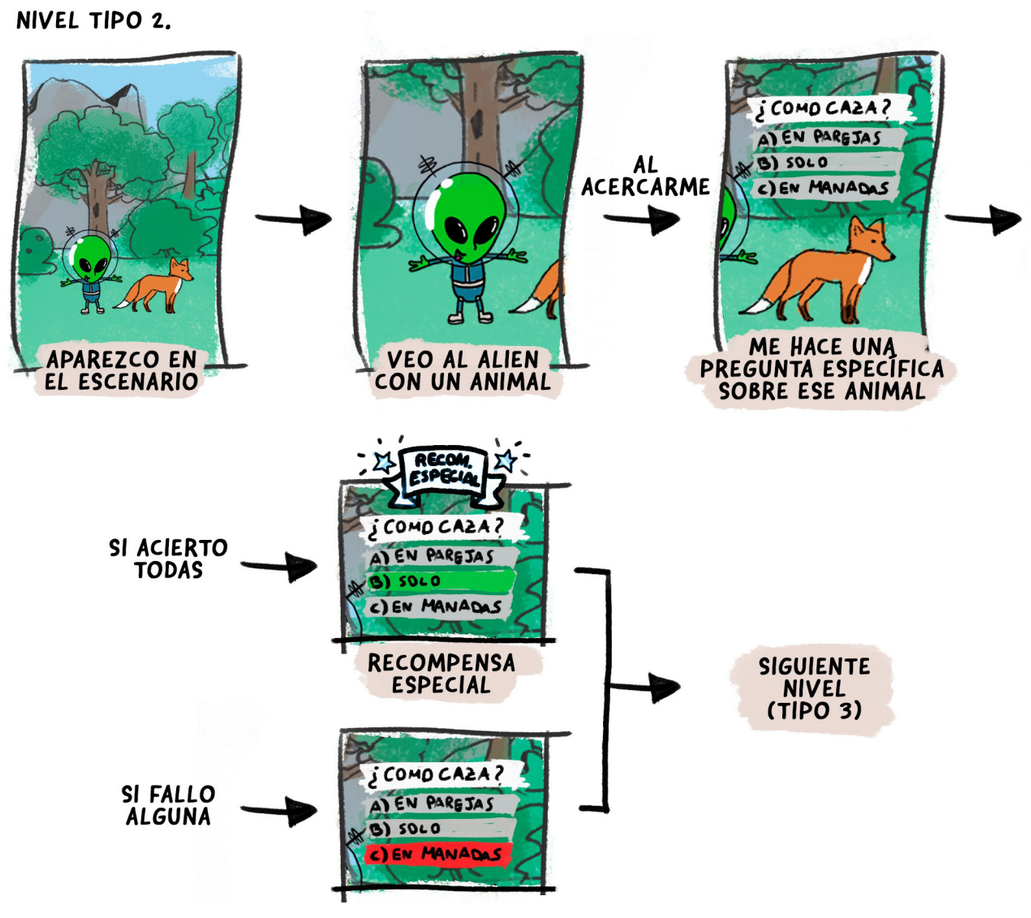
Cuantas más preguntas acertemos, más puntuación. Mínimo tenemos que acertar una. Cuando fallamos una pregunta, pasa a la siguiente.

Preguntas tipo

¿Qué animal es este?	¿Cómo se dice su nombre en inglés/alemán/…?
¿De qué se alimenta?	¿A qué grupo pertenece (mamífero, reptil…)?
¿Es un depredador?	¿Cómo se reproduce?

4.2.2 TIPO 2

El alienígena quiere saber más sobre algún animal ya visto en los niveles anteriores. Se nos plantearán tres preguntas específicas sobre ese animal. Del mismo modo que en los niveles anteriores tendremos 3 opciones para responder. Estos niveles irán intercalados con los del tipo 1 a modo de nivel de bonificación.



Solo se podrá superar este nivel si no se comete ningún error. Si lo hacemos, obtendremos una recompensa especial. Si fallamos, pasaremos de nivel sin ninguna recompensa (aunque podremos volver a intentarlo una vez superemos el hábitat). [11] [12] [13] [14] [15]

Preguntas tipo

¿En qué estación del año pierde el ciervo sus cuernos?

¿Cuánto tiempo aguantan los delfines bajo el agua?

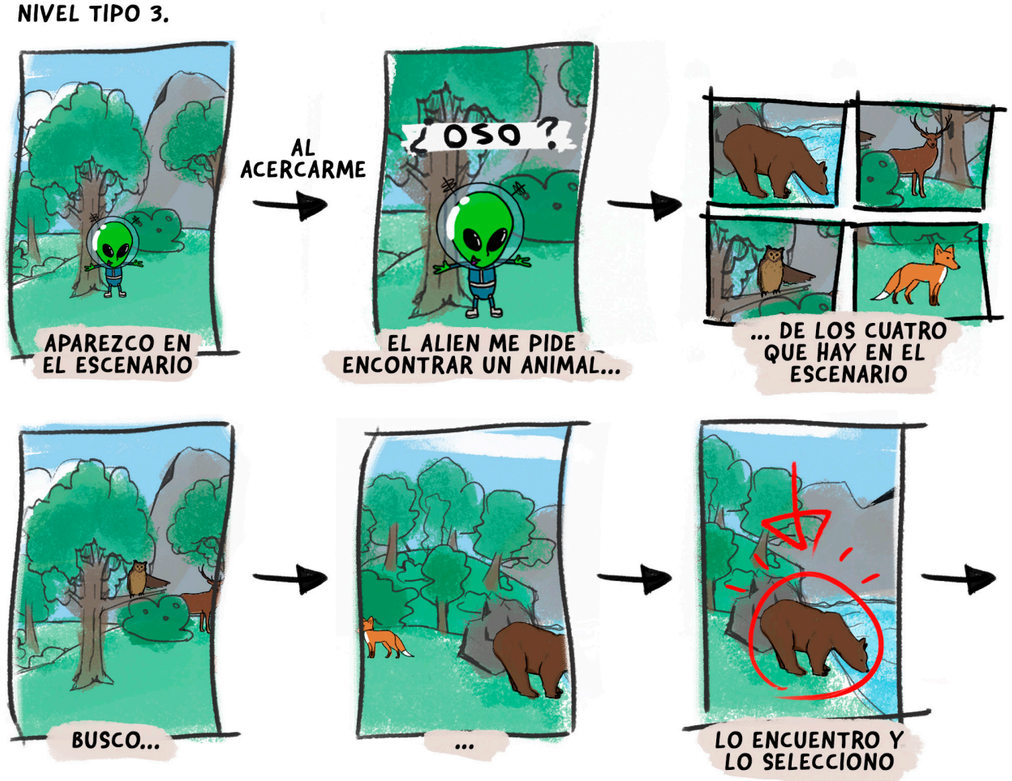
¿Cuánto dura la gestación de los elefantes?

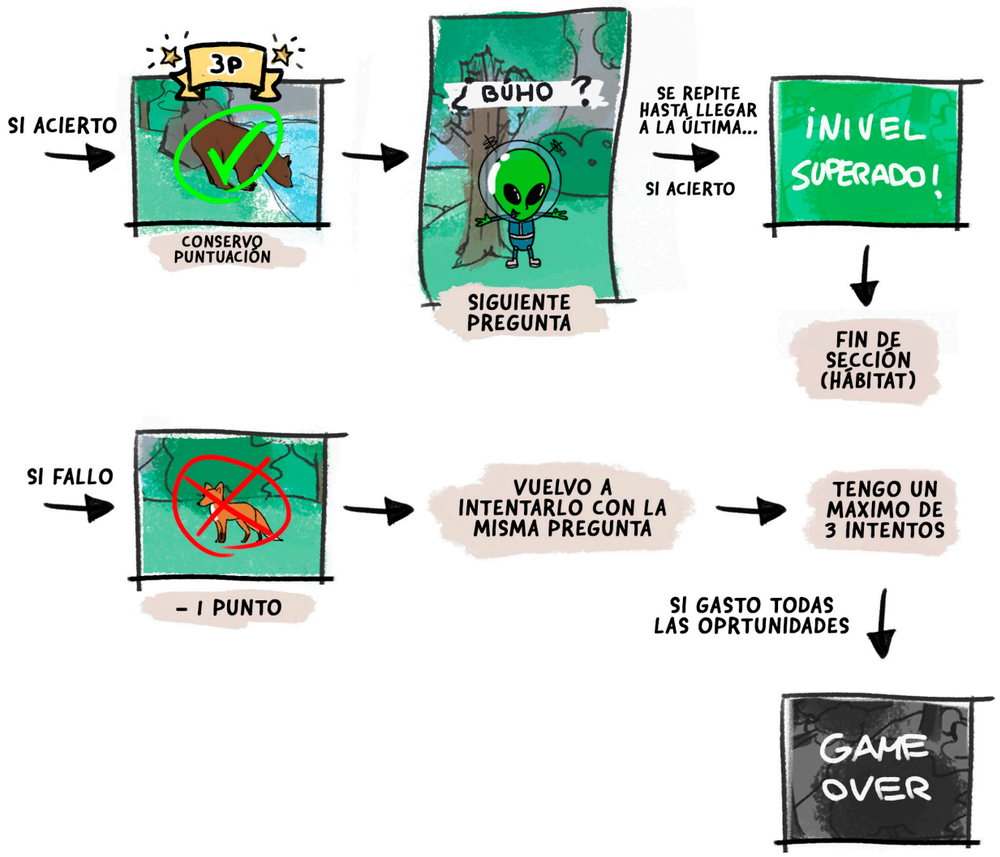
4.2.3 TIPO 3

Al alienígena se le escapan los animales que estaba estudiando y nos pide que encontremos o atrapemos a uno de ellos en concreto cada vez, hasta un total de 3 preguntas distintas. Nos lo pedirá de varias maneras, de forma que tendremos que encontrar y escoger al adecuado según la pregunta que nos hace.

Sería el último de los niveles de un hábitat de manera que los animales que se le escapen sean animales que ya habíamos encontrado.

Si acertamos, pasamos a la siguiente pregunta. Si fallamos, tendremos que volver a repetir la pregunta. Tendremos un máximo de 3 oportunidades. Cuantas menos veces fallemos, más puntuación obtendremos.





Preguntas tipo

¿Podrías traerme al animal herbívoro/carnívoro/omnívoro?

¿Podrías traerme al animal que **no** es herbívoro/carnívoro/omnívoro?
(enunciada de forma negativa)

¿Podrías traerme al depredador?

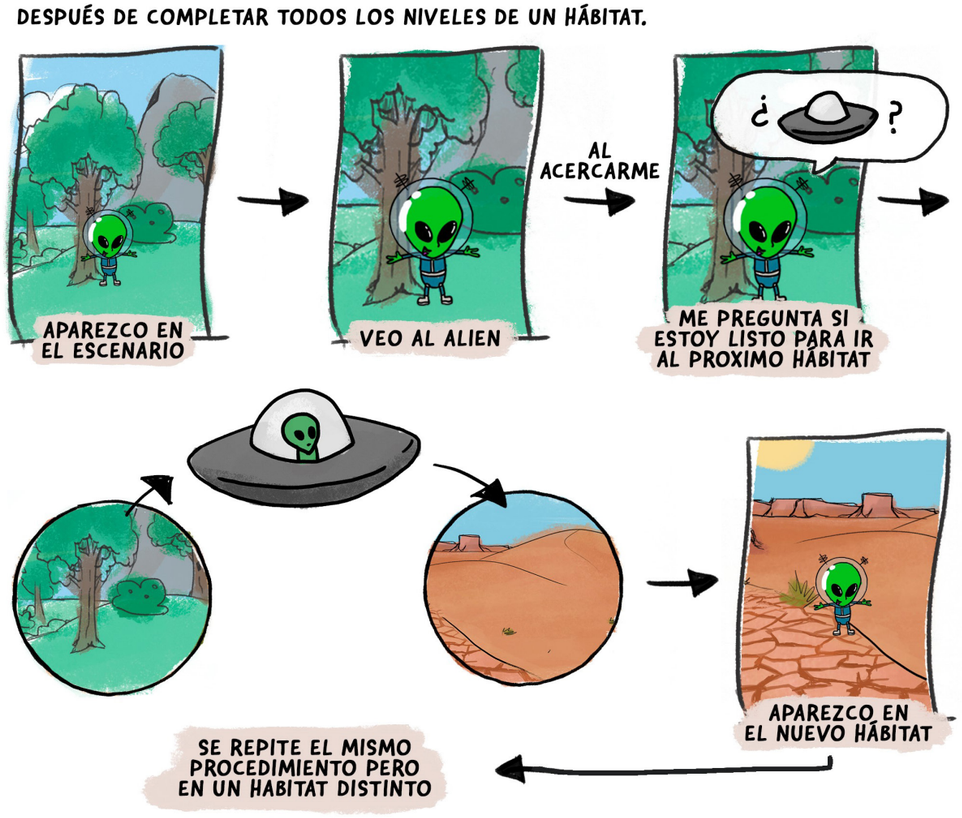
¿Podrías traerme al reptil/anfibio/ave/pez/mamífero?

¿Podrías traerme al animal con colmillos de marfil?
(más específicas con algunos animales)

¿Podrías traerme al rey de la selva?

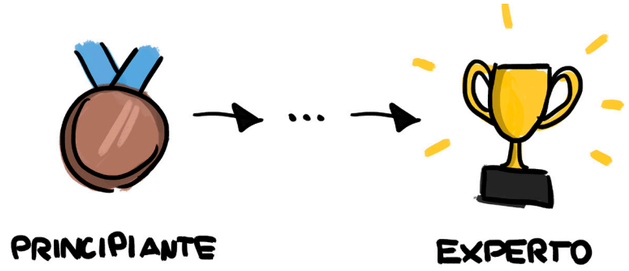
Al superar todos los niveles de un hábitat, el alienígena nos llevará en su nave hasta el siguiente, donde se repite la misma mecánica de juego.

Al volver a un hábitat ya completado, podremos interactuar con el alienígena, que nos ofrecerá repetir (con una aleatoriedad limitada en las preguntas) los niveles. Sirve para repasar, para superar tu puntuación u obtener nueva información sobre los animales que puede no haberse descubierto en niveles anteriores, debido a que las preguntas son aleatorias.



Al completar cada nivel, recibimos una recompensa de puntos en forma de estrellas. La cantidad obtenida dependerá del número de preguntas acertadas o falladas (dependiendo del tipo de nivel).

Estos puntos se irán acumulando para recibir otro tipo de recompensa, en forma de trofeo. Es una especie de sistema de rangos (explorador novato, principiante, avanzado, experto…).



La información que recopilamos de cada animal se va almacenando en un catálogo que podremos consultar cuando no estemos jugando. Es una forma de dejar constancia de lo que se va aprendiendo a lo largo del juego.

Como evento o nivel especial, se podrían añadir animales diferentes de los principales, como animales extintos. La forma de descubrirlos sería volver a los hábitats ya completados y activar algún tipo de evento que podría variar, como, por ejemplo, hablar con el alienígena, interactuar con cierta zona del escenario, mirar hacia cierto punto, etc. [16] [17] [18] [19]

4.3 ESTÉTICA DEL JUEGO

Al ser un juego educativo para niños, se ha optado por darle una estética basada en polígonos simples, lo que se conoce en inglés como *low poly*. Esta apariencia irreal del mundo es muy agradable a la vista debido a sus formas poligonales y sus colores puros, cosa que también atrae a los niños. A continuación, se muestra una imagen del menú de inicio del videojuego, donde se puede ver esa estética, seguida de otras varias capturas “in-game“. Como hasta ahora solo se ha desarrollado el escenario del bosque, las imágenes adjuntas únicamente presentarán este hábitat.


Pregunta 1
¿Qué animal es este?
Búho
Zorro
Oso


Pregunta 1
Tráeme al animal que no come carne
Es este
No es este, seguiré buscando


Hay un animal por aquí cerca...


Has fallado...
Pregunta 1
¿Qué animal es este?
Búho
Zorro
Oso

IMPLEMENTACIÓN

En este capítulo se explicará cómo se han realizado los diferentes elementos presentes en el proyecto y de qué forma se han implementado para el correcto funcionamiento del videojuego.

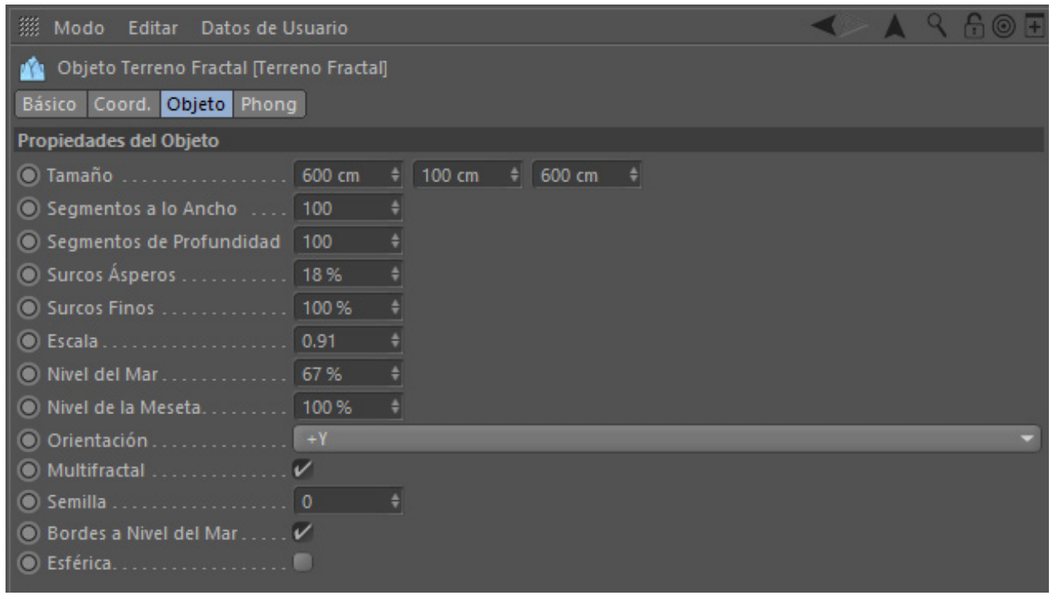
5.1 DISEÑO Y ANIMACIÓN EN CINEMA 4D

Para el desarrollo de este videojuego se han utilizado tanto modelos 3D descargados de internet, como otros originales creados por mí. Nos centraremos en los segundos, aunque cabe mencionar que algunos de los obtenidos de la red han sido modificados para adecuarse aún más a la estética deseada para el juego o corregir pequeños errores. Entre ellos, las setas, los arbustos y las flores. Todos los modelos originales han sido realizados en su totalidad con Cinema 4D. [20]

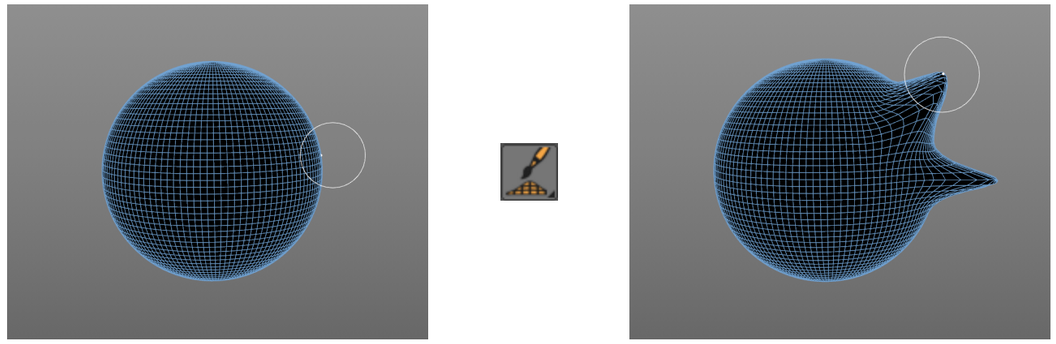
5.1.1 EL ESCENARIO GENERAL

Para empezar, hablaremos de la construcción de la zona jugable o explorable, las montañas y el entorno en general. Todas ellas han sido realizadas partiendo de un terreno fractal, uno de los objetos básicos que incluye Cinema 4D. Funciona muy bien para crear montañas.

Este permite generar una superficie irregular aleatoria modificando parámetros como los segmentos que componen la superficie, la escala, el nivel del mar o el nivel de meseta. También dispone del parámetro semilla, con el que podremos escoger entre todas las formas diferentes de las que se parte, un total de 100000. Debido a este alto número y al resto de configuraciones posibles, se podría decir que permite crear formas aleatorias.

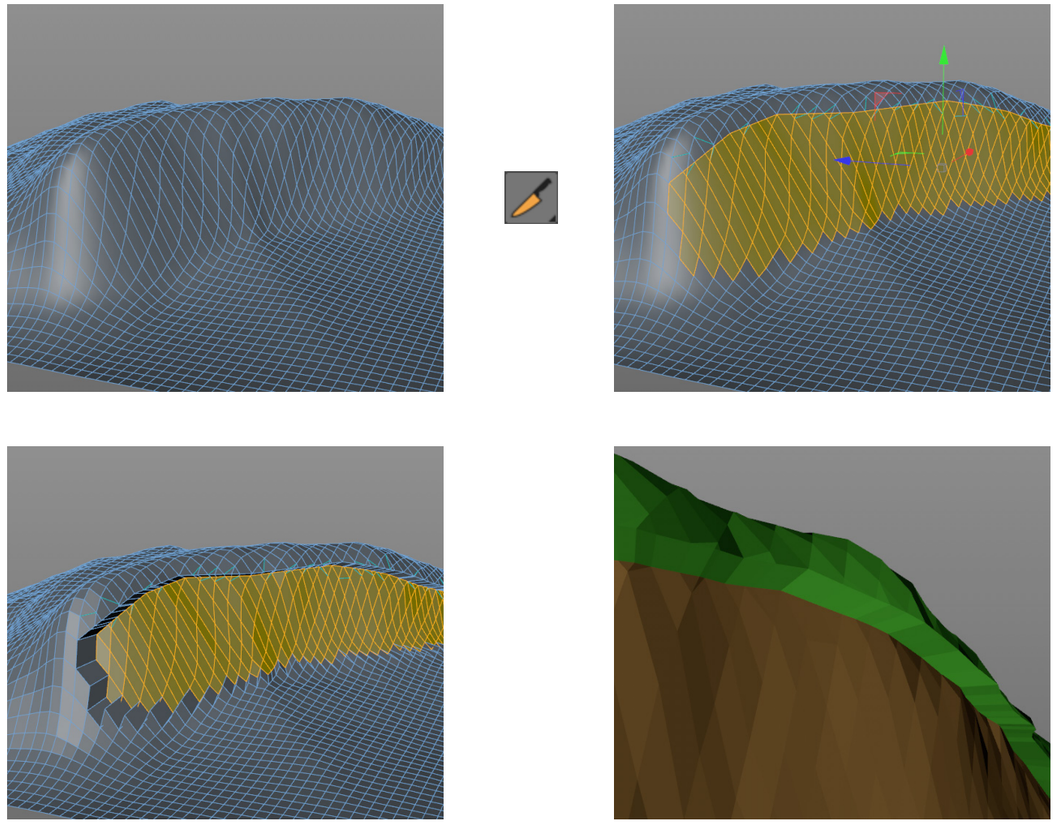


Partiendo de esta base, las formas también han sido modificadas mediante la herramienta *Pincel*, que permite esculpir la malla de un objeto a nuestro gusto. Para ello el objeto debe ser editable.



Para crear los desniveles del terreno (las zonas en las que se ve la tierra), después de utilizar las anteriores herramientas, se ha modificado directamente la malla. Para ello se ha utilizado la herramienta *Cortar línea*, para seccionar de la forma original el perfil deseado.

Una vez hecha la sección, se han seleccionado las caras del polígono en las que se quiere que se vea la tierra, para, manteniendo CTRL, extruir esa forma hacia dentro, escalarla y ajustarla. De esta manera queda resaltada la separación entre la tierra y la hierba del terreno.

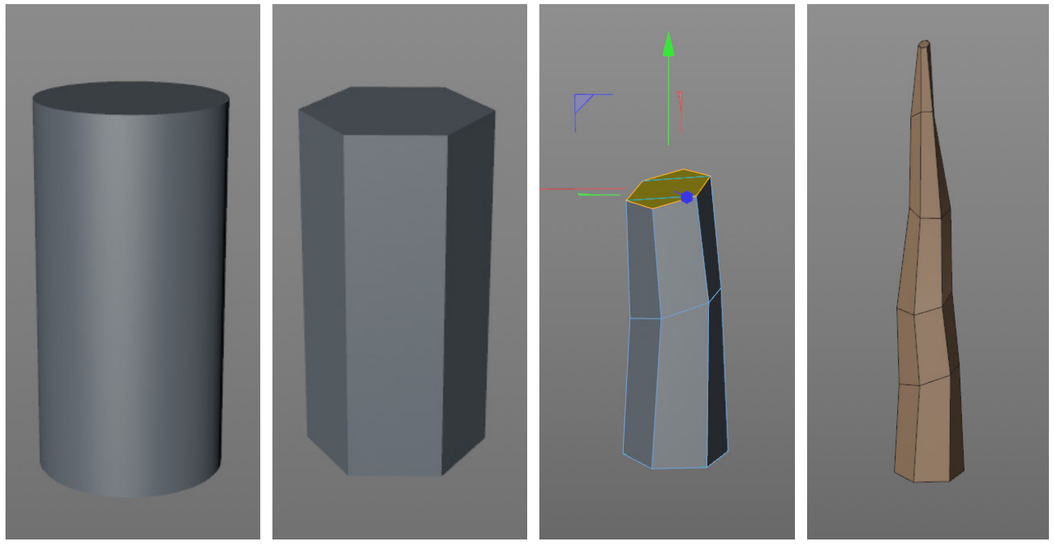


5.1.2 LOS ÁRBOLES DEL BOSQUE

Para los principales árboles del bosque se han realizado tres modelos diferentes con sutiles variaciones en color y forma. Esto se ha hecho para dotar de cierta aleatoriedad al entorno al no ser todos los árboles iguales. Las tres formas se construyen de la misma manera.

Partiendo del objeto cilindro de Cinema 4D se crea el tronco del árbol. Como la estética del juego es "low-poly", se han reducido los segmentos de rotación a 6. De esta manera ya no se puede decir que sea un cilindro, si no un prisma hexagonal.

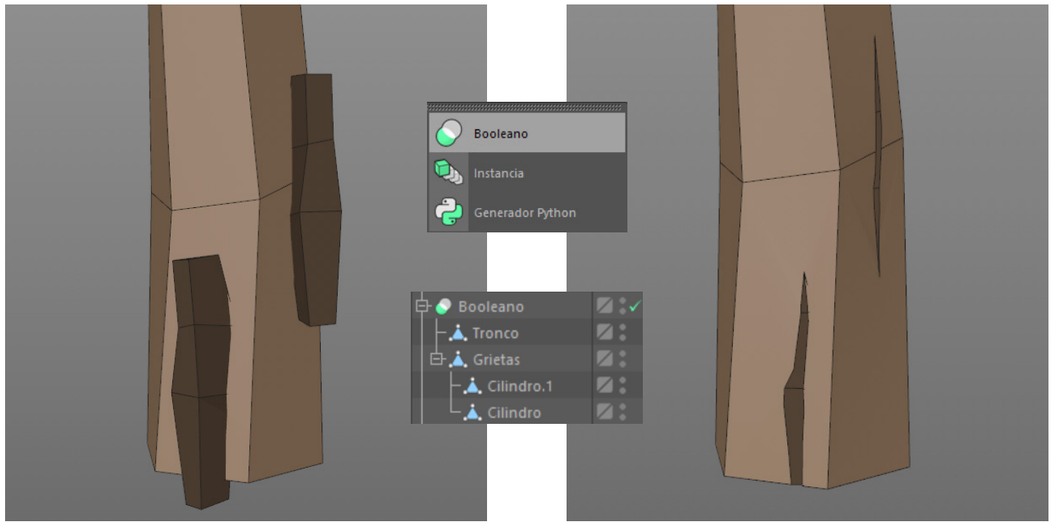
Una vez se ha hecho editable, se ha modificado la malla del prisma para generar una forma irregular estirada hacia arriba. Para ello se han extruido, escalado y rotado repetidas veces las tapas del prisma, haciéndolas cada vez más pequeñas hasta llegar a la punta.



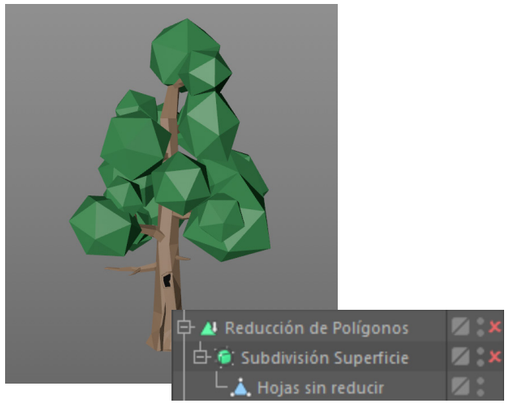
De manera análoga se han realizado las ramas del árbol. Se han construido de manera independiente, por lo que no parten de la forma del tronco sino que se han añadido a este posteriormente.

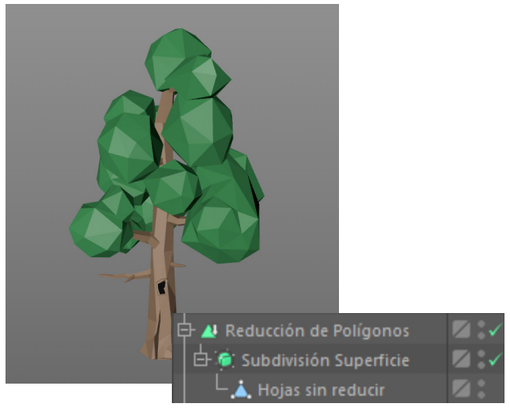
Para el modelado de las raíces superficiales, se ha empleado un método similar al descrito para la construcción de los desniveles del escenario. Primero se ha utilizado la herramienta Cortar línea para luego tomar la sección resultante y extruirla y escalarla repetidamente para darle la forma deseada.

En cuanto a las muescas presentes en el tronco, se ha utilizado la herramienta *Booleano*, que permite sustraer una forma de otra, entre otras cosas. Para ello, primero se ha creado una forma acorde a la muesca que se desea y después se ha superpuesto con el tronco. El último paso es aplicar un Booleano y configurarlo para que modifique el tronco en función de cómo interseccione con la muesca.



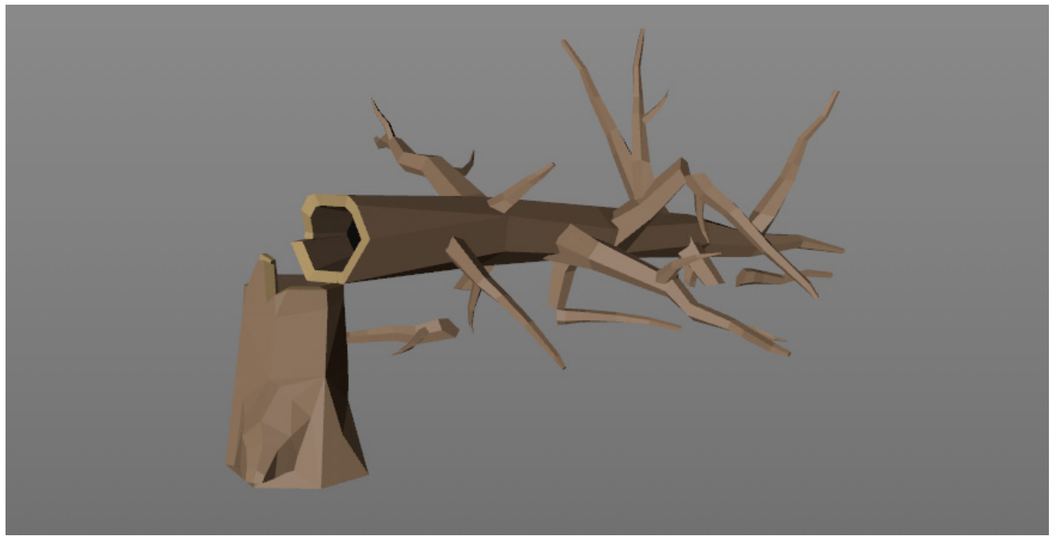
Las hojas de los árboles se han generado a partir de un conjunto de icosaedros regulares e irregulares que han sido unidos mediante un Booleano. Una vez conectados todos en un único objeto, se le aplica una *Subdivisión de Superficie* y un *Reductor de polígonos*, en ese orden. Con esto se consigue, además de reducir los polígonos del objeto, conseguir un conjunto más homogéneo. También se ha utilizado la herramienta *Pincel* para corregir pequeños errores.





Con objeto de dar más variedad de elementos al entorno, se ha creado un tronco roto a partir de uno de los tres construidos. Para ello se ha cortado el objeto con la herramienta *Corta línea* por la zona y con la forma deseadas.

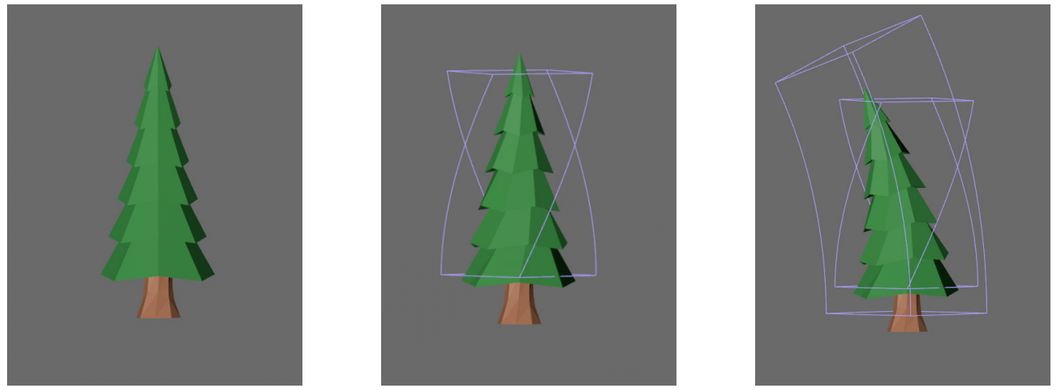
El tronco se ha tumbado en el suelo con el extremo roto apoyado en la base del árbol, como si lo hubieran derribado. Las ramas que colisionan con el plano han sido cortadas de forma parecida al tronco.



5.1.3 LOS PINOS

Aunque el modelo original del pino ha sido descargado de internet, se ha modificado lo suficiente como para poder hablar de su construcción en este apartado. [21]

Se ha modificado su forma de diferentes maneras para dar variedad al escenario. Para ello se han empleado los efectos de *Doblar* y *Enroscar.* En alguno de los modelos se han encadenado varias veces estos dos efectos para crear deformaciones más interesantes. También se han aplicado colores ligeramente diferentes a cada uno. Aquí podemos ver las tres versiones de pinos.



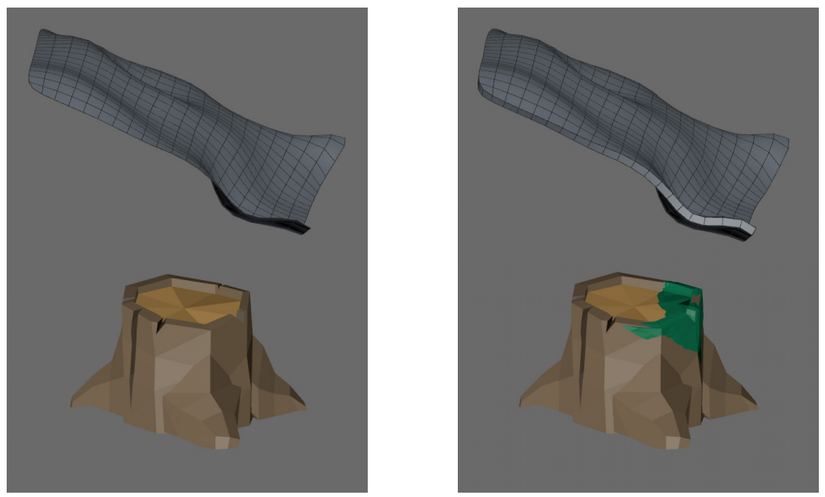
En cuanto al tronco, ha sido reemplazado por otro más detallado, que encajara mejor con la estética del juego. Su construcción es similar a la del tronco de los árboles descritos en el apartado anterior.

En la versión inicial del escenario del bosque la presencia de pinos era mucho mayor que en la versión final, donde apenas hay unos pocos. Por esto, en esta última, la variación carece de tanta importancia, pero al tratarse de una modificación posterior del proyecto, el trabajo ya estaba hecho y cabe mencionarlo.

5.1.4 LOS TOCONES

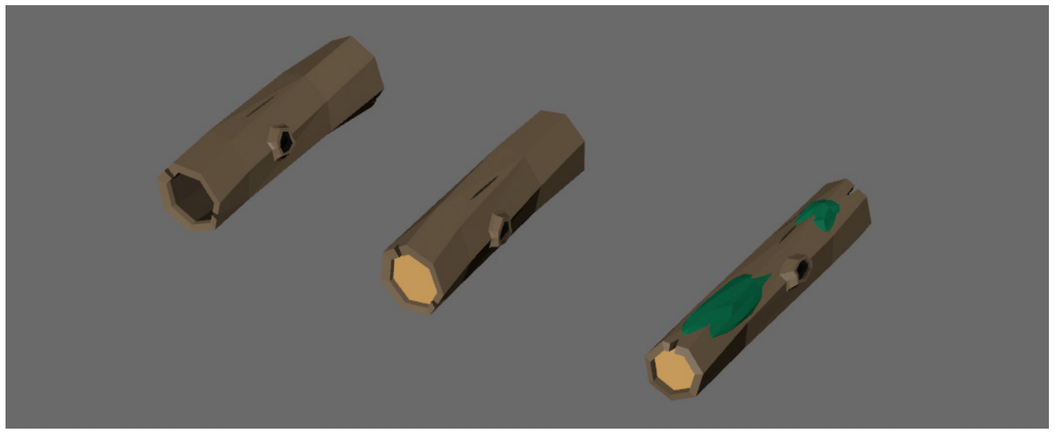
También se ha diseñado un tocón, para utilizar como elemento decorativo. Para su construcción se han utilizado las herramientas ya descritas de *Cortar línea, extruir,* y *booleano* para las rajas. Su proceso de modelado es igual al de los troncos de los árboles del bosque.

La única peculiaridad que presenta con respecto al resto de modelos es el musgo, que ha sido generado a partir de extraer una sección del tocón y aplicarle la herramienta de simulación S*uperficie Cloth.* Esta herramienta permite darle volumen o grosor a un cuerpo plano. Así el musgo parece estar cubriendo el objeto. También se ha aplicado la herramienta *Pincel* para hacerlo más irregular.



5.1.5 LOS TRONCOS CAÍDOS

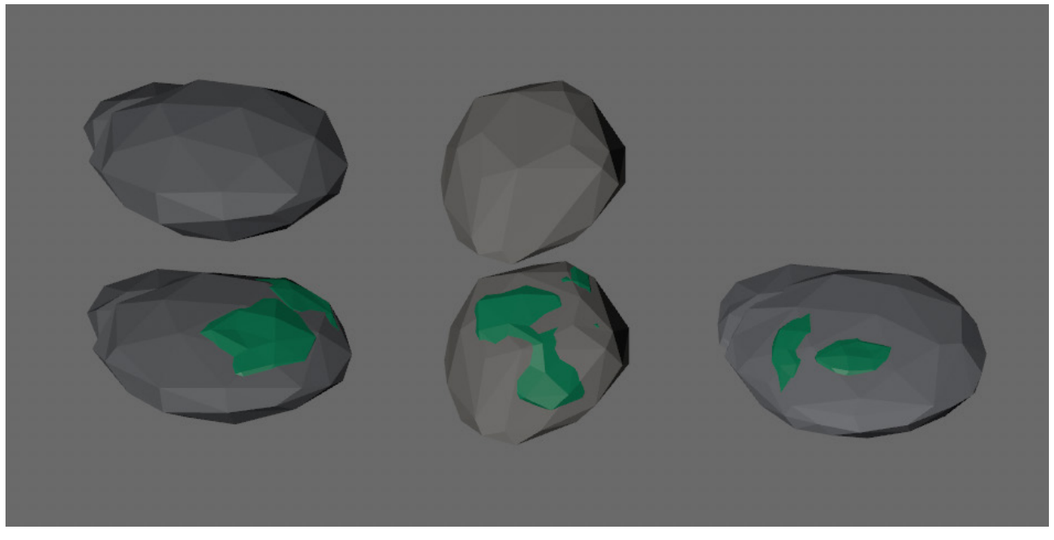
Sobre el proceso de modelado de estos troncos no hay nada nuevo que contar, se han desarrollado de manera parecida a lo descrito anteriormente. Se han creado 3 versiones del mismo modelo: el original, otro con musco y otro completamente hueco. Los tres han sido utilizados en el escenario.



5.1.6 LAS ROCAS

El modelado de las rocas es de los más sencillos. Se ha partido de una esfera formada de pocos polígonos, la cual se ha deformado y escalado hasta conseguir la forma deseada. A algunas rocas también se les ha añadido musgo, pero esta vez no ha sido utilizando la herramienta S*uperficie Cloth.* Simplemente se ha duplicado la roca y se ha superpuesto a ella, con el material del musgo.

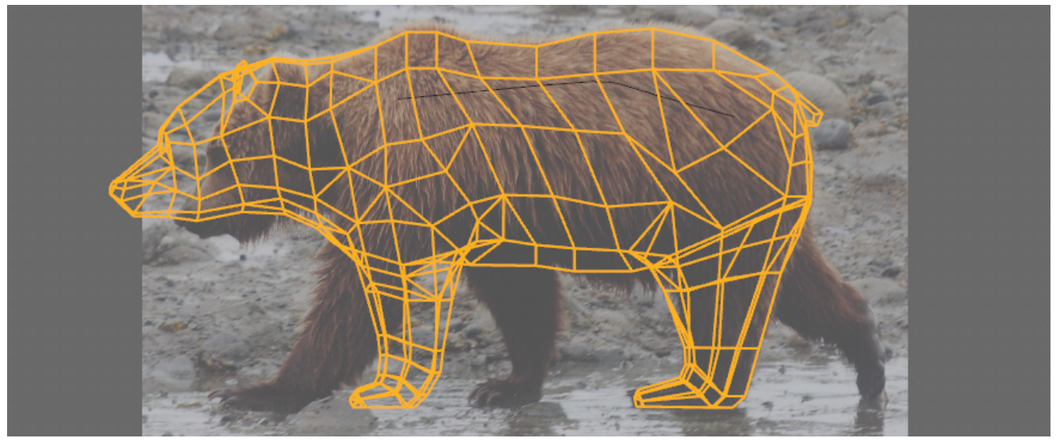
Realmente la forma de las rocas no se ha repetido en el escenario como pasa con los árboles, ya que, al tratarse de rocas, no existe ninguna proporción que conservar y se pueden escalar de la manera que queramos.



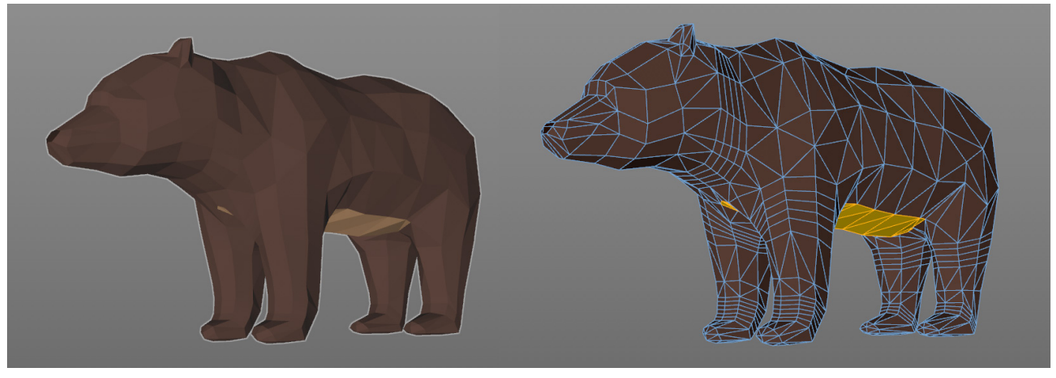
5.1.7 LOS ANIMALES

Aunque el modelo original del zorro, del ciervo y del búho han sido sacados de internet, se han modificado ligeramente para adaptarlos a su animación correspondiente y corregir pequeños errores que se podrían producir al animarlo. [22] [23] [24]

En cuanto al modelo del oso, este se ha generado desde cero. Para ello, se ha partido de una imagen del perfil de un oso. Esta se ha usado como referencia dentro de Cinema 4D para, partiendo de polígonos básicos, construir la silueta del animal. Una vez conseguido el perfil básico, se ha ido añadiendo detalle al modelo hasta lograr el aspecto deseado.

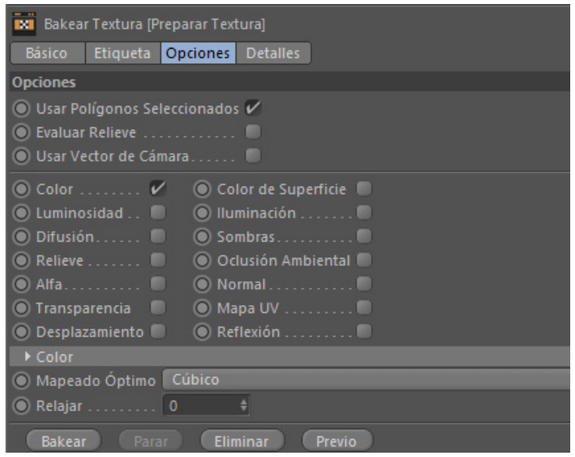


La apariencia de cada animal se ha logrado mediante la combinación de distintos materiales. Cada uno de estos materiales se corresponde con un color diferente y se establecen dentro del modelo por selección de polígonos.



Todas las texturas de los animales se han generado de forma automática dentro de este mismo programa a partir de los distintos materiales atribuidos a cada animal. Para esto se ha empleado el atributo “Bakear textura”. Lo que hace es hacer una lectura de todos los polígonos que componen el modelo en cuestión y crear una única imagen que contenga todos los materiales presentes en él en los polígonos correspondientes.

Una vez obtenida la imagen, se crea un nuevo material con esta textura. De esta forma los modelos exportados solo están compuestos por un material en vez de por varios. [25] [26] [27]



5.1.8 EL LOGO

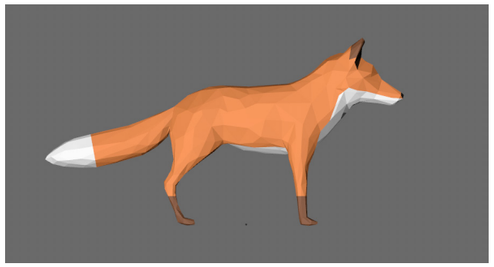
Para realizar el logo que aparece en la pantalla de inicio se ha seleccionado una tipografía que encajase con la estética del juego. Una vez escogida y editada dentro de *Illustrator,* se ha llevado a *Cinema 4D,* donde se ha extruido para darle un aspecto tridimensional.

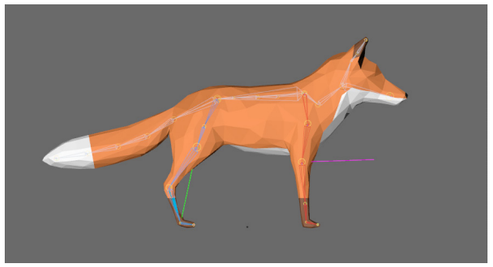
Por último, se han animado cada una de las letras de la palabra “animal” para darle dinamismo a la escena inicial. [28]

5.1.9 LA ANIMACIÓN

Para explicar el proceso de animación se utilizará el modelo del zorro.

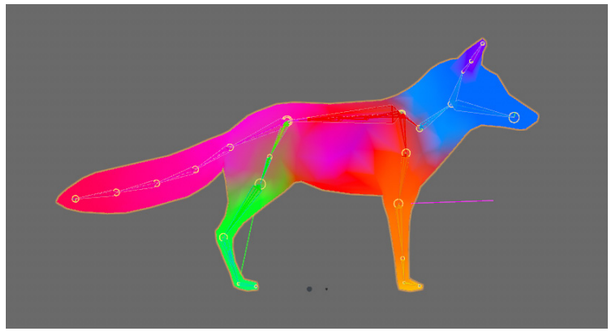
Se comienza con la creación de las juntas. Con ellas se establece lo que sería el esqueleto del modelo a animar y también son las responsables de que este último adquiera movimiento.



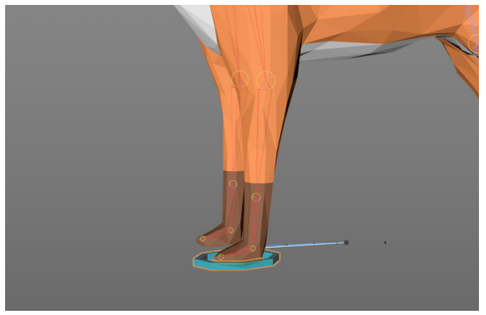


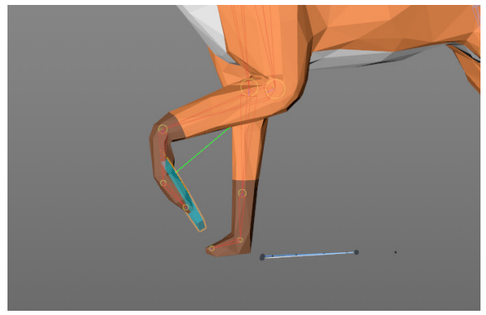
Una vez están colocadas, han de enlazarse con el modelo en cuestión. Para ello, primero se seleccionan todas las juntas y el modelo. Seguidamente, se aplica la acción *Enlazar,* dentro de la sección comandos del apartado *Character*. Con el modelo ya enlazado, se puede empezar a animar.

Una vez hecho esto, lo normal es encontrarse con que el modelo se deforma de manera extraña en algunos puntos al articular las juntas. Esto se soluciona con la herramienta *Peso*. Con ella se puede controlar qué sección del modelo se mueve con cada junta y en qué medida.



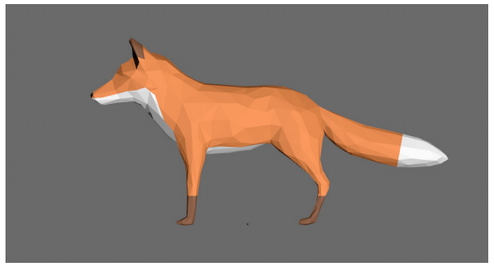
Hay dos tipos de animación: FK e IK. La primera hace referencia a “Forward Kinematics” y consiste en el movimiento de las juntas individualmente. IK corresponde a “Inverse Kinematics” y consiste en el movimiento de varias juntas al mismo tiempo a través de un manejador. Las animaciones se han realizado de forma mixta, aunque predomina el método IK.



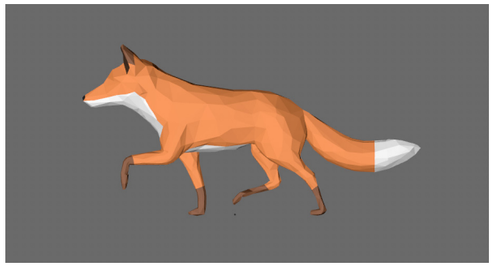


El siguiente paso es la propia animación del modelo. Para ello es esencial la línea de tiempo. En esta es donde ocurre la animación realmente. El proceso consiste en establecer fotogramas clave para que el programa genere las interpolaciones entre ellos.

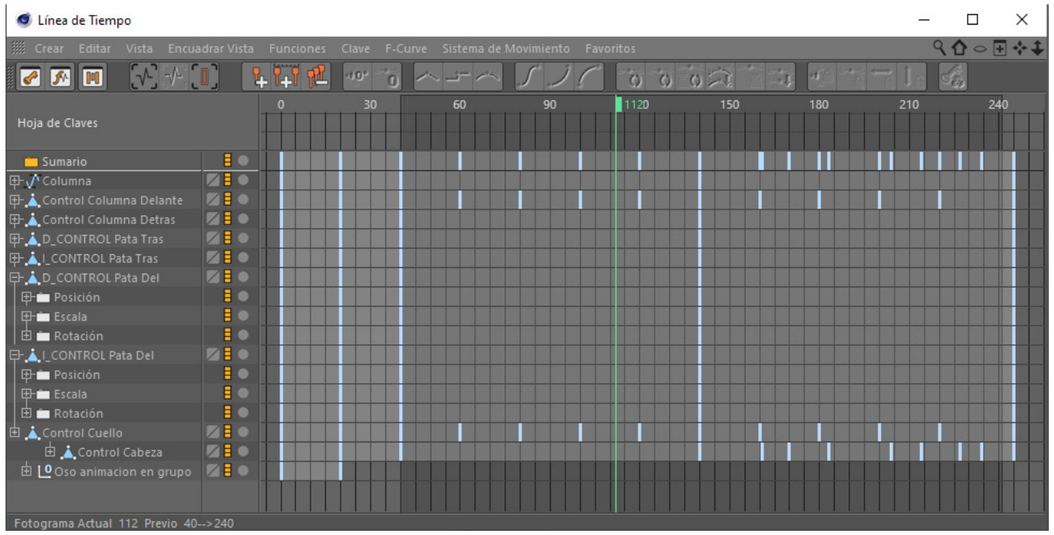
Un fotograma clave es en el que se establecen unos valores determinados de los parámetros involucrados. En este caso, posición y rotación. Estos fotogramas clave sirven para establecer las poses que, en su conjunto, componen el movimiento, para que luego el programa genere las interpolaciones oportunas entre estas. De esta manera se crea el movimiento sin necesidad de establecerlo fotograma a fotograma. [29] [30] [31] [32] [33] [34] [35]



Pose 1



Pose 2

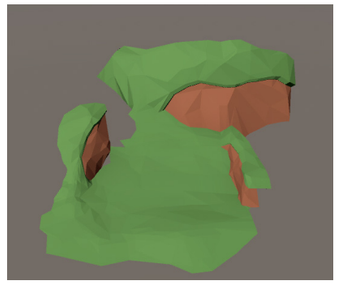


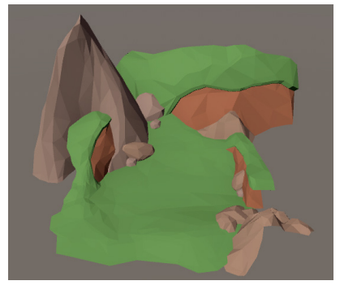
5.2 CONSTRUCCIÓN DEL ESCENARIO

En este apartado se explicará el proceso de construcción del escenario a partir de los modelos 3D disponibles.

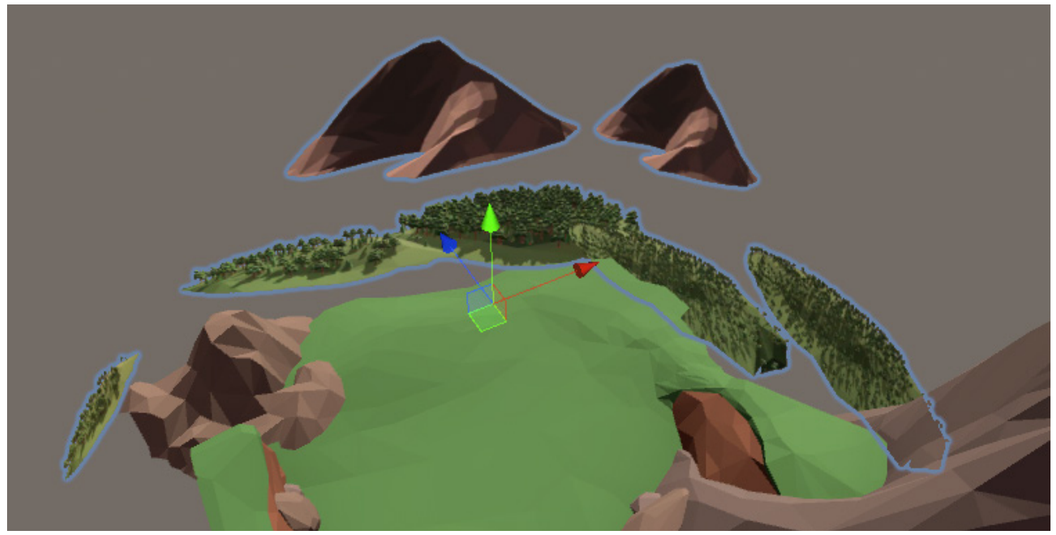
Una de las cosas más importantes a tener en cuenta a la hora de crear un escenario virtual, es que no se vea el horizonte del entorno 3D. Para ello es esencial generar irregularidades en el terreno o añadir elementos grandes como montañas para cerrar el espacio, de manera que el jugador solo pueda ver el escenario y el cielo.

Para empezar, se ha partido de la superficie donde se apoyará todo lo demás. Después se ha colocado la montaña y seguidamente las rocas que terminan de rellenar y dar variedad al entorno base.

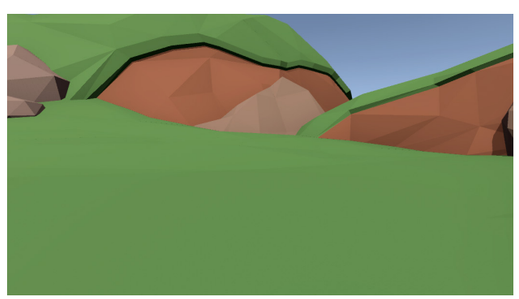


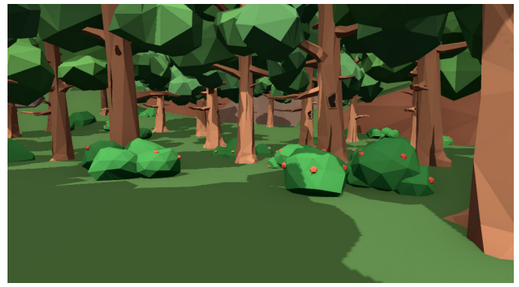


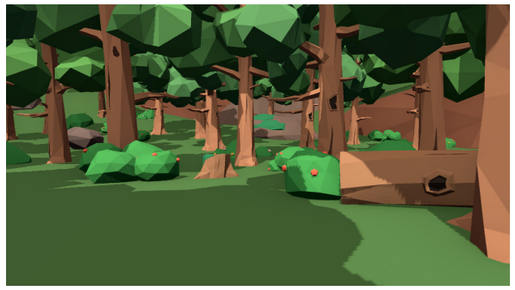
Para que el escenario dé la sensación de mundo extenso, se han colocado imágenes planas de monte con árboles (los mismos utilizados en el escenario principal) y de una montaña frente al entorno principal para producir ese efecto de profundidad y lejanía al mismo tiempo que se oculta el horizonte al jugador. En la siguiente imagen se puede apreciar.

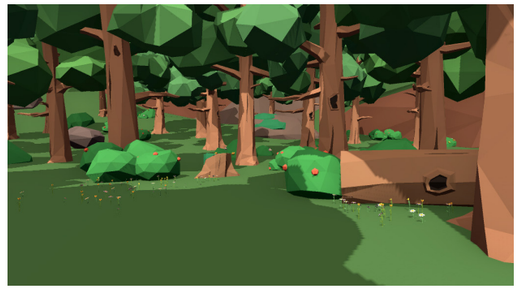


A continuación, se han colocado en este escenario el resto de elementos que componen el bosque: los árboles, los arbustos, los troncos, los tocones y más rocas. Asimismo, se han colocado flores y setas para dar más variedad al entorno más cercano al jugador.





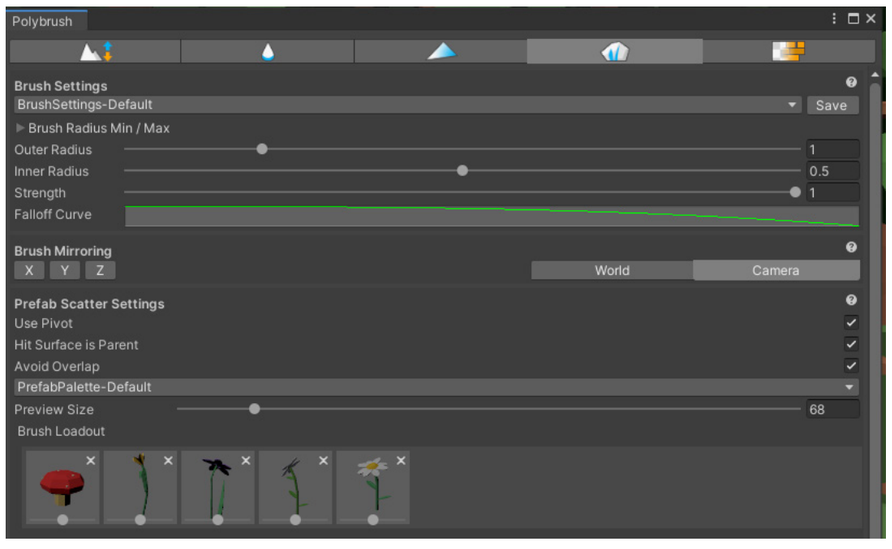




Para situar los elementos que más se repiten como los árboles, los arbustos o las flores y setas se ha utilizado la herramienta Polybrush, que se puede descargar de la Asset Store de Unity. Esta permite "pintar" en la superficie escogida los elementos que elijamos para distribuirlos aleatoriamente sobre ella utilizando como base el punto pivote de cada uno. [36]

Una vez colocados los elementos, se han tenido que corregir algunas posiciones que no estaban del todo bien. Esto se debe principalmente a que la superficie de esta zona es irregular, y al utilizar Polybrush algunos modelos (árboles y arbustos principalmente) no quedaron bien colocados del todo.

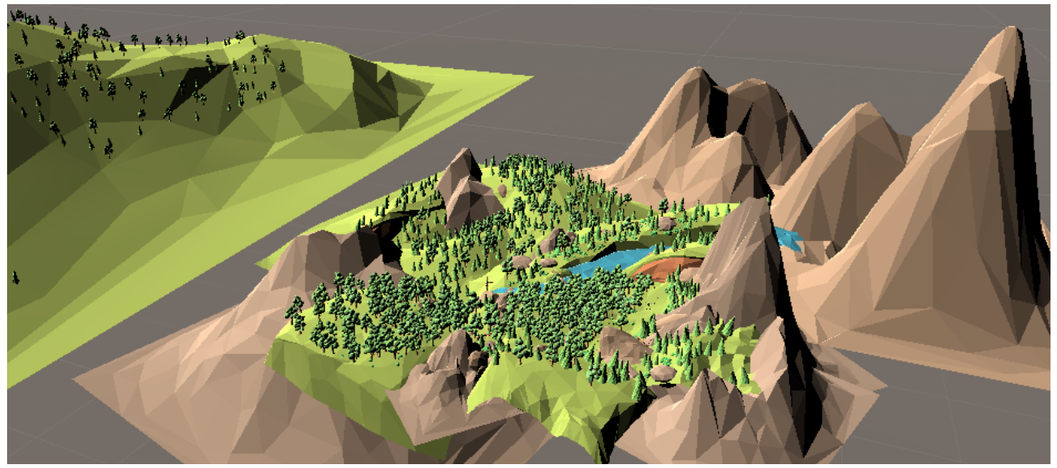
A parte de los estos árboles, también se han colocado algunos pinos de forma manual para dotar de algo más de variedad al escenario.



5.2.1 DISEÑO INICIAL DEL ESCENARIO

En un principio el juego se había planteado de forma que el jugador pudiera recorrer el escenario libremente para poder explorarlo en busca de cada animal. Por ello el escenario era más amplio y variado. Contaba con un río, montes llenos de árboles que se podían observar al asomarse a un precipicio por el que el río caía en forma de cascada y con una zona explorable limitada por paredes invisibles que impedían el paso al resto del escenario.

Se ha decidido cambiar este planteamiento por dos razones. Primero para que el juego fuera más ligero al tener un escenario mucho más reducido con menos elementos en él. Segundo para simplificar la jugabilidad, ya que para que el jugador pudiera moverse libremente se necesitaría un controlador o joystick. De esta forma con este nuevo enfoque, el jugador interactuará con los elementos del juego mediante un solo botón (presente en la propias gafas VR) y el propio uso de las gafas para poder observar el entorno en 360º.

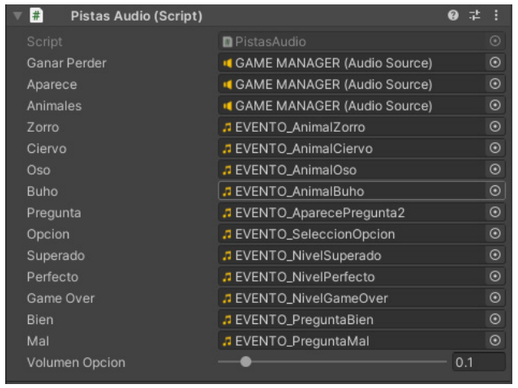


5.3 EL AUDIO

Tanto los sonidos utilizados en el juego como el hilo musical son originales y han sido producidos con el software de producción FL Studio.

Para que un audio se escuche en el juego se necesita un Audio Source. Este es un componente con el que debe de contar el objeto que va a reproducir el sonido. En el inspector, solo se puede asignar una pista al Audio Source, como se ha hecho en el caso de la música de fondo y el sonido ambiente.

Esto se ha hecho porque suenan en bucle, pero en el caso de que el audio a reproducir lo haga de forma puntual, una mejor opción es asignar la pista de audio al Audio Source mediante código. Esto se ha hecho con todos los sonidos para eventos (sonido al fallar, al acertar, etc.), que solo se reproducen una vez (por evento).



En este script se han creado unas variables públicas de audio donde poder asignar manualmente en el inspector cada uno de los sonidos de evento presentes en el juego. Dentro del código asignamos estas pistas a un Audio Source, que estará dentro del mismo objeto que este script.

Esto se realiza por medio de una función pública (más adelante se explicará en qué consiste), de manera que el resto de scripts puedan controlar qué pista se reproduce y en qué momento.

Estas funciones son llamadas desde un Event Trigger, componente que nos permite llamar funciones dentro de un script a elegir cuando se da una condición (al hacer click, al pasar el puntero por encima, etc.). Todos los objetos que reproduzcan un sonido al interactuar con ellos cuentan con uno de estos componentes.

Para tener un mejor control de los volúmenes de cada pista, a cada Audio Source se le asigna una salida hacia el Audio Mixer, donde se tendrá una visión general de todos los volúmenes de las pistas [37]

5.4 CÓDIGO E IMPLEMENTACIÓN EN UNITY

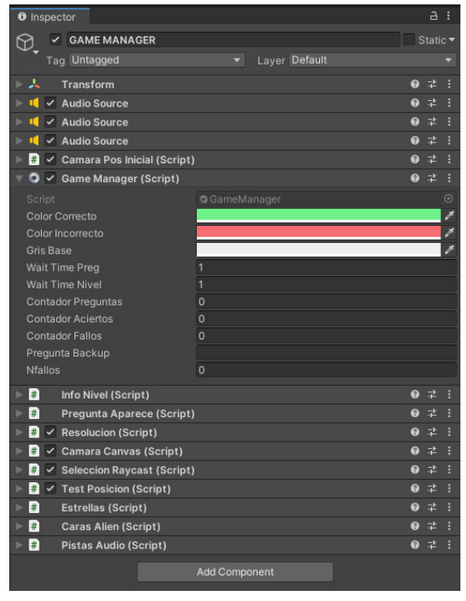
Todo el código es original y se ha escrito en C#. Los scripts mencionados estarán disponibles junto con esta memoria para su consulta. [38] [39] [40] [41] [42] [43] [44] [45] [46]

5.4.1 DISEÑO Y ESTRUCTURA BÁSICA DE UNITY

Para entender cómo se trabaja dentro de Unity es importante conocer lo que son las ventanas *Inspector* y *Hierarchy* (o jerarquía).

La primera está por defecto situada en la parte izquierda de la pantalla. Es aquí donde se introducen todas las escenas (desde la ventana Proyect, que es donde se almacenan por carpetas todos los archivos del proyecto), por lo que todos los elementos que aparecen en el juego se encuentran listados en esta ventana. Como su propio nombre indica, estos elementos se pueden introducir unos dentro de otros, formando una jerarquía.

La segunda se sitúa por defecto en la parte derecha de la pantalla. En esta es donde se muestran los componentes de cada elemento de la escena. Aquí es donde deben arrastrarse desde *Proyect* los scripts que hacen funcionar el juego. Es aquí también donde se podrán editar (en caso de que los haya) los parámetros de los scripts y los componentes empleados.



5.4.2 ANIMACIONES

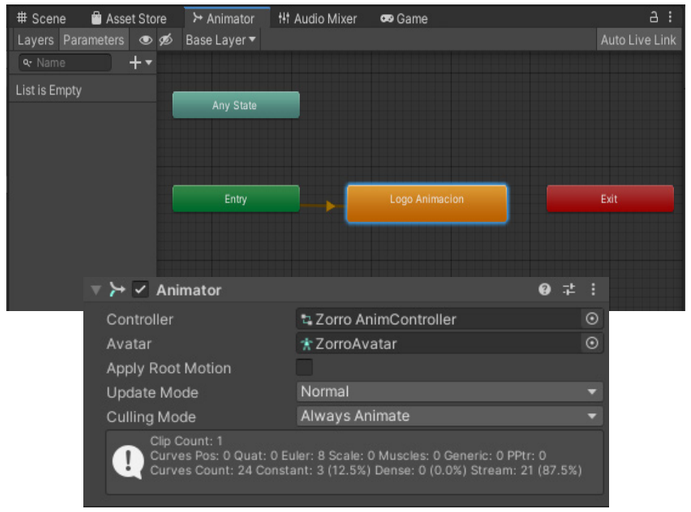
Para llevar las animaciones a Unity, primero tenemos que exportarlas junto con el modelo en cuestión en formato .fbx desde Cinema 4D, ya que es compatible con todos (o casi todos) los programas.

Una vez importado el archivo a Unity (dentro de la carpeta *Assets*), hacemos click en este y nos abrirá sus ajustes en la ventana inspector. Dentro de esta se encuentran 4 pestañas, pero la que nos interesa en este caso es la de *Animation.* Aquí se podrá establecer si la animación el cíclica y en que frames empieza y acaba.

El siguiente paso es crear un *Animator Controller* en la carpeta *Assets.* Este es necesario para gestionar y programar las animaciones y las transiciones entre ellas. En este caso solo se han utilizado una animación por animal, por lo que no se ha programado ninguna transición entre animaciones.

El *Animator Controller* se abre a través de la ventana *Animator* y es aquí donde se podrán introducir y gestionar todas las animaciones existentes en el archivo *.fbx.* Para introducirlas solo han de arrastrarse desde el *.fbx* a la ventana *Animator*.

La animación por defecto (en este caso la única existente) se debe conectar con el nodo *Entry*. El último paso será añadir un componente *Animator* al modelo en cuestión y referenciar en este al *Animator Controller* correspondiente a su vez que el *Avatar* del modelo en cuestión, es cual se genera automáticamente en los ajustes del *.fbx.* [47] [48]



5.4.3 FUNCIONES

Las funciones son una parte importante dentro de la programación. Consisten en una serie de líneas de código que desempeñan una tarea concreta, la cual puede devolver un valor o no. Dentro de estas se podrán emplear variables propias o las ya existentes dentro del código en el que están escritas (aunque en algunos casos no es la mejor opción).

La funciones pueden ser públicas o privadas. Las primeras podrán ser accedidas desde otros scripts mientras que las privadas solo podrán utilizarse dentro del programa en el que se encuentran. A continuación un ejemplo del aspecto básico de una función:

```
public void FuncionEjemplo() {
    […]
    [contenido o tarea]
    […]
}
```


Las funciones ayudan a facilitar la lectura del script por parte del usuario, ya que con ellas este queda fragmentado en bloques de código independientes que pueden ser llamados en cualquier momento para que desempeñen su función. Esto se hace de la siguiente forma: [49]

```
FuncionEjemplo();
```

5.4.4 TRIGGERS

Los triggers funcionan como interruptores. Se usan para activar un determinado evento o mecanismo (descrito en el código) al interactuar con el objeto que actúa como trigger. Para poder definir como trigger un objeto, debe hacerse a través del componente *Collider*. Este incluye entre sus ajustes una casilla para activar o desactivar el trigger. Solo los objetos con este componentes son interretados por el programa como elementos en los que se puede hacer click.

5.4.5 RAYCAST

El raycast se podría decir que es en esencia un rayo rectilíneo que sale desde el objeto escogido en una determinada dirección a lo largo de una distancia máxima establecida por el programador. Se utiliza en programación para obtener información acerca de cuándo un objeto colisiona con este rayo.

En el caso concreto de este juego, se utiliza para saber cuándo estoy mirando a un animal y cuándo no. Esto es necesario conocerlo para poder realizar acciones como: cambiar la posición de la UI del test (como se explicará más adelante), saber a qué animal estoy mirando con la ayuda de los Tags o activar la aparición de la silueta blanca al apuntar a uno de los animales.

Esto último se ha realizado gracias a una herramienta de la Asset Store llamado Quick Outline. Al añadir el script al objeto en cuestión, podemos controlar el ancho de la silueta y su color, entre otras cosas. [50]

Al tratarse de funciones públicas podemos acceder a este componente desde otro script. En este caso desde el script que utiliza el raycast para saber a dónde está mirando el jugador. De esta forma la silueta o “outline” solo aparece cuando se apunta a uno de los animales. [51] [52]

5.4.6 FUNCIONAMIENTO DEL TEST

Las preguntas tipo test son una de las partes más importantes del juego, ya que es a través de estas de donde se obtiene la mayor parte de información acerca de los animales.

Cada pregunta consta de tres opciones a elegir (excepto en los niveles de tipo 3, que se explicarán más adelante). Inmediatamente después de haber respondido, sabremos si hemos acertado o no por medio de información visual y auditiva.

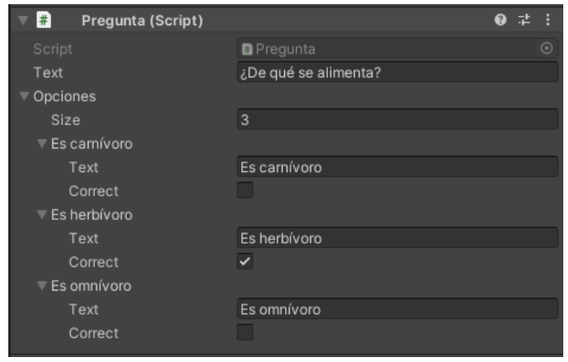
En lo referente a lo visual, el color de la opción elegida se teñirá de rojo en caso de fallar y de verde en caso de acertar. Esto además irá acompañado con un mensaje por parte del alien que nos informará de esto mismo pero de forma textual.

En cuanto al audio, los aciertos y los fallos tienen cada uno un sonido correspondiente. Estos dos sonidos se han tomado de base para crear también los que se reproducen al superar o no el nivel y al superarlo sin ningún fallo (perfecto).

Los scripts involucrados en el funcionamiento del test son: Pregunta.cs, Opciones.cs, TestDB.cs, TestUI.cs, BotonesOpciones.cs, GameManager.cs y Resolucion.cs. Seguidamente se explicará la función de cada uno de ellos.

Las preguntas junto con las opciones a elegir se almacenan en *prefavs*. Esto son objetos virtuales con unas características, unos componentes y un comportamiento predefinidos por el usuario. Son útiles para instanciar un objeto las veces que se requiera sin tener que crearlo de cero cada vez que se necesite.

Cada uno de estos *prefavs* almacenan el script *Pregunta.cs* que permite introducir una pregunta junto con las opciones que se requieran, de las cuales solo la correcta estará marcada como tal. Tanto el texto de las opciones como la casilla que permite seleccionar cuál es la correcta están contenidos en el script *Opciones.cs,* el cual esta referenciado en forma de lista dentro de *Pregunta.cs*. A partir de la información que aportan estos objetos el resto de scripts involucrados podrán mostrar al jugador las preguntas, las opciones y su resolución.



Las preguntas estarán referenciadas manualmente en el script *TestDB.cs* que estará contenido dentro de un game object que hará de base de datos para el funcionamiento del test. Este script es el encargado de escoger la pregunta que se va a mostrar en cada momento. Exceptuando la primera pregunta de los niveles de tipo 1 (que siempre será la misma), la elección de estas es aleatoria. Esto se hace con la función:

```
Random.Range(x, y)
```

El valor escogido será un número aleatorio entre *x* e *y.* El número corresponde al índice de la pregunta dentro de una lista.

Este mismo game object que contiene *TestDB.cs* también contará con el *TestUI.cs*, que se ocupará de establecer el texto de la pregunta y de comprobar si la opción escogida es correcta o no, al obtener dicha información de *BotonesOpciones.cs*. Esto se hace por medio de una función. Todos los botones donde se muestran las opciones también están referenciados manualmente en este script a través del inspector.

BotonesOpciones.cs se comunica con *Opcion.cs*, de donde obtiene la información para establecer el texto de cada opción y para saber si la escogida es correcta o no (esta es la información que le llega a *TestUI.cs*). *BotonesOpciones.cs* se añade a cada botón (opción), los cuales contienen como hijo un elemento texto, que es el que se modifica para mostrar cada opción.

GameManager.cs es el encargado de, entre otras cosas, decidir qué hacer a partir de la información aportada por el resto de scripts ya explicados. Dicho de otra forma, es el que decide tomar una acción u otra en función de la opción escogida. También lleva la cuenta del número de aciertos y fallos, lo que a su vez también afecta al comportamiento del test, concretamente a la resolución del nivel.

Para esto último entra en juego *Resolucion.cs.* Este script es el encargado de mostrar el resultado final (si se ha superado o no el nivel) y el mensaje de acierto/fallo además de cambiar o repetir el nivel (escena). Todo esto se hace por medio de funciones públicas que son llamadas desde *GameManager.cs.*

La fila superior de la caja que contiene la pregunta también aporta información. Al lado izquierdo se puede ver el número de pregunta actual. Al lado derecho, la puntuación obtenida hasta el momento (en forma de estrellas). Ambas partes están programadas en *Estrellas.cs*, que al igual que en *Resolucion.cs,* se controlan mediante funciones públicas a través, también en este caso, de *GameManager.cs.* Una de ellas edita el texto de la izquierda y la otra el color de los sprites (imágenes) de las estrellas.

El componente *Button,* presente en todos los botones, cuenta con una opción que funciona como un Event Trigger, solo que únicamente dispone de la condición de hacer click (OnClick()). Aquí se puede enlazar una función de un script para que se accione al pulsar el botón. Esto se ha utilizado principalmente en los botones de siguiente nivel y repetir nivel al finalizar, enlazando la función correspondiente dentro de *Resolucion.cs.*

Normalmente esta referencia a la función se realiza de forma manual desde el inspector, pero también puede hacerse por código, como en el caso de los niveles de tipo 3. Esto se ha hecho de esta forma porque normalmente el botón de Siguiente Nivel está enlazado con la función que te lleva al siguiente nivel, pero al ser este el último de todos, esta se ha cambiado por otra que te devuelve al menú. A continuación un ejemplo de cómo se vería este trozo de código

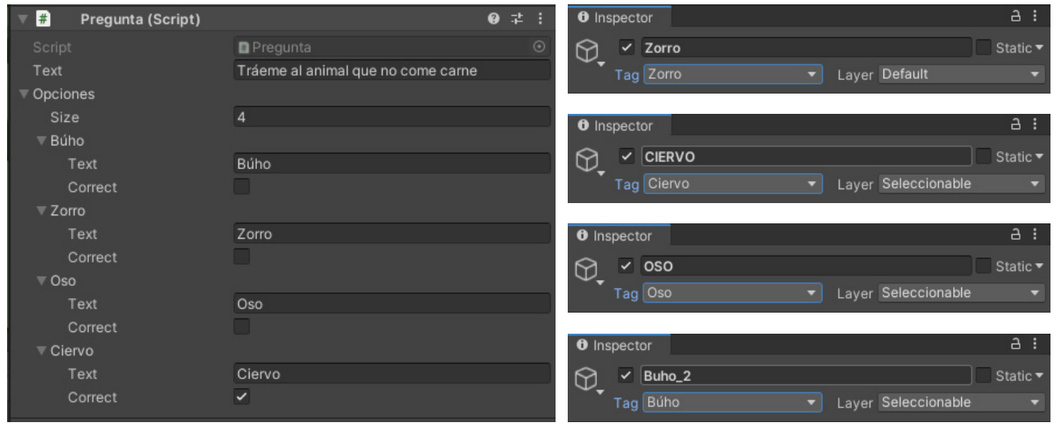
```
botonEjemplo.onClick.RemoveAllListeners();
UnityEventTools.AddPersistentListener(botonEjemplo.onClick, FuncionEjempo);
```

A parte de esto, también mediante código, se ha cambiado el texto del botón de “Siguiente nivel” a “Volver al menú”:

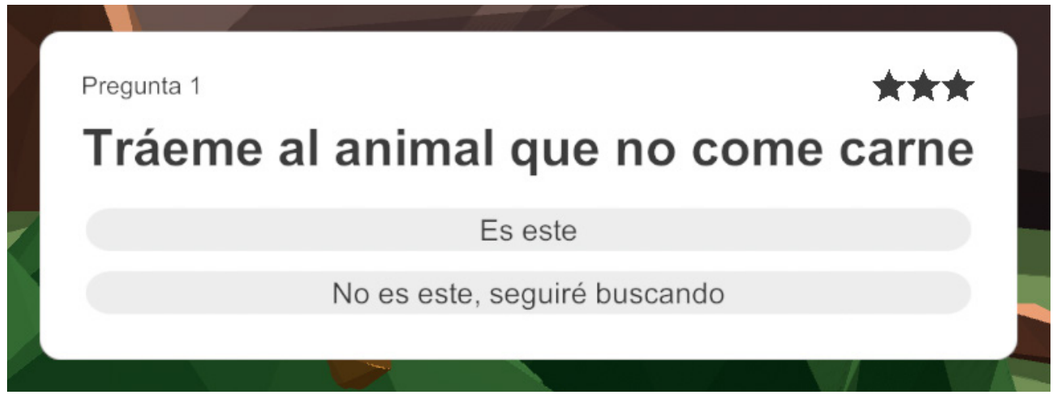
```
Text txt = botonEjemplo.transform.Find(“NombreObjetoEjemplo”).GetComponent<Text>();
txt.text = “Volver al menú”;
```

A parte de esto, los niveles de tipo 3 presentan otra diferencia en su funcionamiento con respecto al resto de niveles, en concreto en las preguntas del test. En este caso las opciones por pregunta irán descendiendo de uno en uno empezando por cuatro. Esto es porque cada pregunta te hará elegir entre uno de los animales presentes en el entorno, que son cuatro, y cada vez que aciertes se retirará a ese animal, quedando uno menos cada vez.

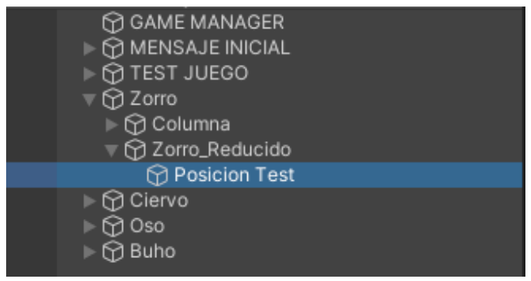
Aquí las opciones son los propios animales. Se seleccionan a través del Raycast y se comprueba la coincidencia del tag de cada animal (Game Object) con la respuesta correcta.



Por ello la forma en que se muestran aquí las preguntas también será distinta. La caja no contará con un botón por opción, si no que tendrá solo dos, uno para confirmar que el animal al que estamos mirando es la respuesta correcta, y otro para cerrar el cuadro de texto y seguir buscando.



También en lo referente al test, por el funcionamiento de este nivel, el cuadro de texto tiene que cambiar de posición según el animal al que se esté mirando. Por esto se ha escrito el script *TestPosicion.cs*. Para ello se han creado unos objetos vacíos (llamados “Posición Test”) en la jerarquía como hijos de los animales, de manera que mediante código se acceda a la posición de estos puntos (previamente colocados en el lugar más conveniente cerca de cada animal) e iguale la posición del test con la de estos puntos. [53] [54]

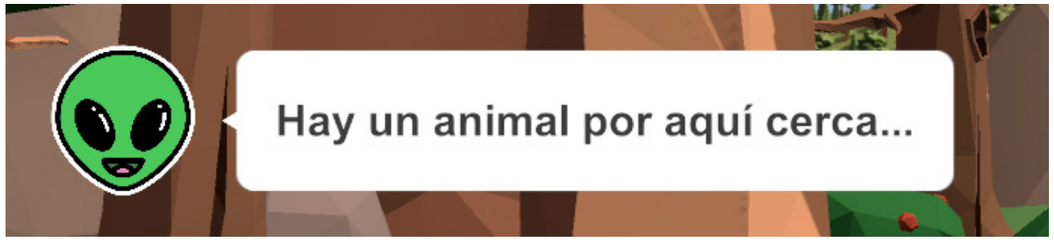


5.4.7 INTERFAZ DE USUARIO (UI)

La Interfaz de Usuario aparece directamente en el escenario y no en un menú aparte. Este entorno en sí mismo ya aporta bastante información visual y es lo suficientemente colorido como para añadir más detalle por encima.

Debido a esto, para la UI se ha elegido un diseño con un estilo limpio y sencillo, que presente únicamente la información necesaria. Por lo cual, los únicos dos colores que presenta son el blanco y el negro, a excepción del feedback recibido al acertar o fallar una pregunta (la opción escogida se tiñe de verde o rojo respectivamente) y de la ilustración del alien, que también es a color.

En cuanto a la forma, la caja que contiene la información referente al test tiene las esquinas redondeadas. Esto aporta una experiencia visual más agradable que si fuera rectangular. Asimismo, el mensaje del alien que aparece al responder una pregunta conserva este estilo, aunque en este caso lo haría en forma de bocadillo de texto, como si él mismo te lo estuviera diciendo. Esta será la forma en que se mostrarán todas las interacciones de este personaje.



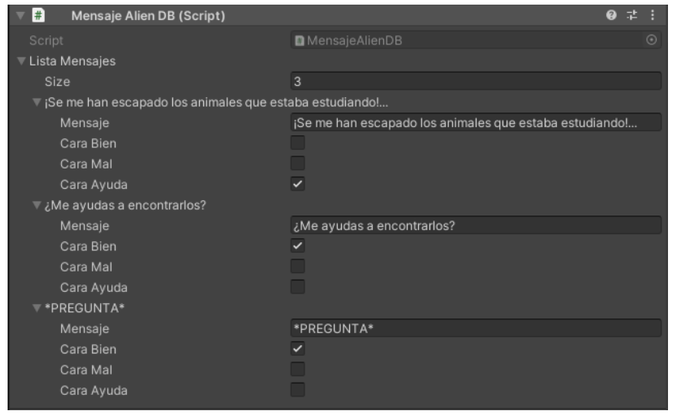
En esto último, la expresión del alien al interactuar variará en función de lo que te esté diciendo. Cuenta con un total de 3 expresiones faciales.

La primera, y la empleada por defecto, tendría un aspecto amigable y aparecería acompañada tanto con mensajes positivos como neutrales. La segunda mostraría un tono de decepción y será la indicada para cuando se cometa algún error. La tercera y última presenta una expresión de preocupación, utilizada cuando el mensaje transmitido corresponda a este sentimiento, como cuando en los niveles de tipo 3 este te dice que ha perdido a todos los animales que estaba estudiando y te pide que lo ayudes a recuperarlos.

Al inicio de cada nivel, el alien te pide algo. Esto será de lo que trate el nivel. En los de tipo 1, te pedirá que encuentres a un animal que anda cerca. En los de tipo 2 te pedirá que le ayudes a conocer mejor al animal en cuestión mediante preguntas más específicas. En los de tipo 3 te dirá lo mencionado en el párrafo anterior.

Para crear esta mecánica se han escrito unos códigos (*MensajeAlien.cs, MensajeAlienDB.cs y MensajeAlienFuncionamiento.cs*) que funcionan de la siguiente forma.

Para empezar, se ha creado un código que usaremos como contenedor de mensajes. En *MensajeAlienDB.cs* será donde escribamos cada mensaje que dirá el alien al mismo tiempo que la expresión facial que mostrará. Esto se hace por medio de variables públicas (que podemos editar en el inspector), una de tipo string para el texto y tres de tipo booleano para marcar la expresión a utilizar.



Para que estos mensajes se muestren como deben, se ha creado *MensajeAlien Funcionamiento.cs*. En este se ha programado todo su comportamiento. Se han creado funciones para que el mensaje siga a la cámara en todo momento, para que se muestre el siguiente mensaje al hacer click y que desaparezca tras mostrar el último, para ocultar o mostrar el mensaje y lo mismo para los animales presentes en el entorno (cuando se muestra el mensaje no se muestran los animales). También se ha creado una función pública que permite editar el texto mostrado desde otro programa.

Como ya se ha mencionado, todos estos scripts estarán disponibles para su consulta en los archivos aportados junto con la memoria.

Toda la UI se adapta a la longitud del texto mostrado, ya sea una pregunta o un mensaje. Esto se ha realizado de dos formas distintas: en el caso del test a través del código y en el caso del mensaje inicial del alien a través del inspector mediante componentes de disposición o “layout”.

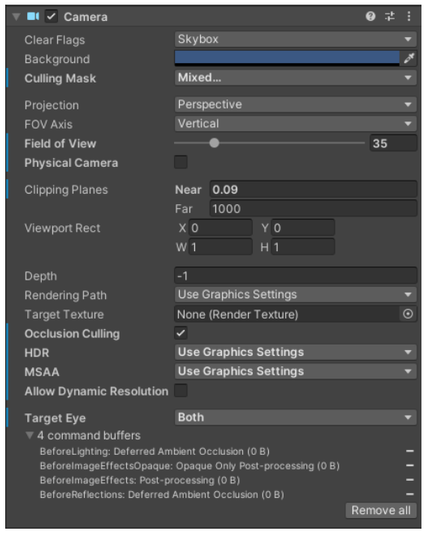
El método por código funciona de forma sencilla, simplemente obtiene información sobre el ancho y alto de la caja del conjunto de elementos y del ancho de la caja que contiene la pregunta. Cuando el ancho de la pregunta supera al del conjunto, este último se iguala con el primero y viceversa cuando es al contrario. En cuanto a la altura, está predefinida en funciones que se utilizan en función del número de opciones que haya.

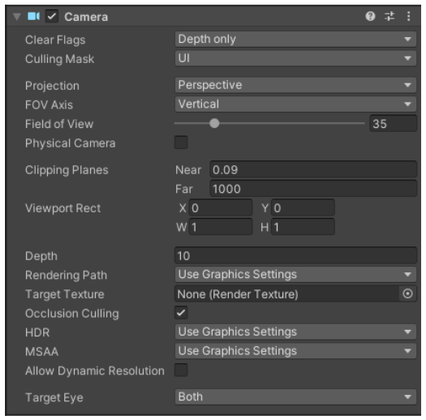
La UI también cuenta con el script *MirarACamara.cs* que como su propio nombre indica se encarga de que la información mostrada esté enfocada siempre en dirección a la cámara.

5.4.8 VISUALIZACIÓN DE MENSAJES

En el juego se utilizan dos cámaras al mismo tiempo. Una para la UI y otra para todo lo demás, lo cual es posible gracias a que el componente cámara ofrece la opción de filtrar lo que muestra. Esto se

ha hecho para que el mensaje inicial del alien se pueda ver mires a donde mires, incluso cuando se encuentre físicamente detrás de otro objeto. A parte del filtro, tienen algunos otros ajustes distintos.



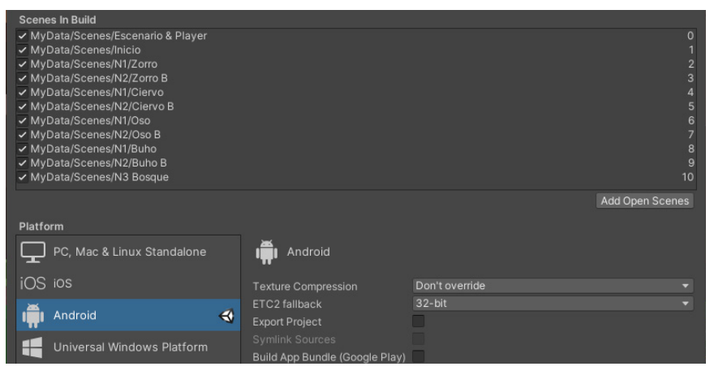


Aunque la caja del test también forma parte de la UI, se ha excluido de esta categoría dentro de Unity ya que se ha preferido que esta diera la sensación de que está físicamente en el escenario, siendo en ocasiones tapado parcialmente por otro objeto. [55]

5.4.9 ESCENAS

Los diferentes niveles del juego se han estructurado por escenas independientes dentro de Unity. Dichas escenas, para que estén presentes en el archivo final exportado, deben añadirse a la lista dentro de los Build Settings. Dentro de esta lista cada una de ellas tiene asociado un número o índice que determina el orden en el que aparecen.

En este caso, la escena número cero se ha utilizado para alojar únicamente el escenario del bosque. Al ser un elemento constante en todo el juego (la parte desarrollada al menos), esta escena se ha programado para que no se destruya (mediante *DontDestroy.cs*) al cambiar de nivel y además no tenga que volver a cargarse, lo que ralentizaría la experiencia.



5.4.10 GOOGLE CARBOARD VR

Esta es una herramienta gratuita desarrollada por Google que permite añadir de forma rápida al proyecto un controlador VR junto con su retícula (punto centrado en la pantalla que indica la posición del puntero). Esta retícula viene con la funcionalidad incorporada de expandirse al apuntar a un objeto interactuable (que contenga un trigger).

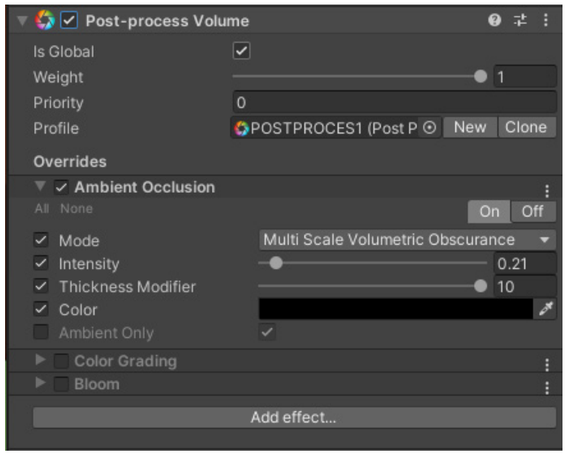
Para utilizarla solo es necesario añadir a la escena una serie de scripts y prefavs que incluye. A la cámara debe añadírsele *GvrPointerPhysicsRaycaster.cs* y como hijo de esta, debe añadirse a la escena el prefav *GvrReticlePointer.* Por último, solo es necesario añadir los prefavs *GvrEventSystem* y *GvrEditorEmulator* y ya estaría todo listo para que funcione. [56] [57] [58]

5.4.11 POST PROCESSING

El post processing es, como su propio nombre indica, un proceso de mejora de calidad de imagen que se aplica de forma posterior al renderizado original en tiempo real. Esto es un método que se emplea tanto en videojuegos como en cualquier otro tipo de contenido audiovisual, como películas o fotos.

En este caso, se ha utilizado únicamente para añadir un efecto de oclusión ambiental. Este favorece a una iluminación más realista al añadir sombras en las zonas de contacto de dos superficies colindantes. A continuacíon se puede ver una comparativa del antes y el después. La diferencia aunque es sutil, mejora mucho la experiencia visual.

Para aplicarlo solo es necesario crear un *Post Process Profile*. Tras ello y para que este se aplique a la escena, debe añadirse el componente *Post-process Volume* a uno de los objetos. Una vez hecho esto, dentro del inspector en el componente ya mencionado se pueden aplicar los efectos que se deseen y ajustarlos como se quiera. [59]



PRUEBAS

En el desarrollo de cualquier proyecto que involucre escribir código, es necesario pasar por un proceso de ensayo – error. Este proceso se da de forma constante durante todo su desarrollo, ya que cada vez que se crea o se añade algo nuevo al código hay que asegurarse de que no haya errores de compilación ni de funcionamiento.

Si uno de los scripts involucrados contiene errores de compilación, el juego no puede iniciarse, por lo que estos son los primeros que se deben corregir. Pero que no haya este tipo de errores no quiere decir que el juego funcione bien, ya que puede que la sintaxis empleada no sea del todo correcta o que el algoritmo no esté bien planteado. Esto provoca que el juego no funcione de la manera que se espera.

Es por esto que todo el desarrollo del videojuego se ha visto interrumpido por estos tipos de errores, pero es algo inevitable que siempre acompaña a los programadores.

Para ilustrar esto anterior, se puede usar como ejemplo el mensaje del alien que se muestra al iniciar cada nivel. En este caso se quería que el mensaje se mostrara siempre delante de la cámara independientemente de la dirección en la que mirara el jugador.

Para llevarlo a cabo era necesario crear una función que se ocupe de que el mensaje se desplace constantemente a un punto delante de la cámara a una distancia fijada. Una vez escrita y aplicada en el objeto a través del inspector, se podía observar que el mensaje cumplía el cometido pero presentaba dos problemas. El primero, que aunque el mensaje se mostraba siempre centrado en la vista del jugador, no rotaba en dirección a la cámara, por lo que el mensaje no se podía leer desde cualquier posición. El otro problema es que el desplazamiento lo hacía a tirones, no era nada fluido.

El primer problema se solucionó con un script (que ya había sido escrito para otro objeto) que hacía que el objeto que lo contuviera mirara en todo momento a la cámara. Para solucionar el segundo, tras investigar un poco, se descubrió que podría ser útil una función integrada de unity llamada “Lerp”. Esta lo que hace es crear una interpolación entre dos posiciones contando con una variable que se podría decir que establece la suavidad con la que se crea dicha interpolación.

Una vez aplicado esto el mensaje debería mostrarse como se había planteado, pero al ejecutarlo comprobamos que no es así, ya que aunque s**í** que se desplaza de forma más fluida que antes, sigue haciéndolo de forma algo atropellada. De nuevo se ha investigado el problema, y se ha encontrado que, en lugar de ejecutarlo en la función *Update()* (que se ejecuta una vez por frame, o sea, constantemente), es mejor hacerlo en la función *FixedUpdate().* Esta se ejecuta justo despúes de *Update()* y por alguna razón favorece a que el movimiento del mensaje sea más fluido.

Una vez finalizado el juego, éste ha sido probado por otros seis usuarios (familiares y amigos). Estas pruebas han servido para comprobar que no existen errores en la ejecución del programa. Estos usuarios han reportado altas tasas de satisfacción además de varios comentarios positivos como “los gráficos son espectaculares” o “me gusta cómo se adaptan los menús”.

COCLUSIONES

7.1 CONCLUSIONES GENERALES

Mi idea inicial para este proyecto parte, como ya comenté en la introducción, de la idea de trasladar el concepto del juguete clásico de los animales al terreno audiovisual, concretamente el de los videojuegos. Fue mi tutor quien me propuso la idea de desarrollarla en un entorno de realidad virtual empleando Unity, además de darme pautas sobre desarrollo de videojuegos. Al plantearse como una aplicación de este tipo, me parecía buena idea ampliar el concepto más allá del simple "¿qué sonido hace la vaca?", permitiendo al niño que jugase explorar cada hábitat y conocer en mayor profundidad a cada animal.

Empecé a trabajar en el proyecto con ganas, aunque en ocasiones era muy frustrante. Esto era debido principalmente al desconocimiento de Unity en particular y del mundo de la programación en general. Por ello tuve que invertir mucho tiempo en mi aprendizaje, documentándome acerca del funcionamiento de esta herramienta y de la programación en C#, apoyándome tanto en libros como en la búsqueda por internet y viendo tutoriales en plataformas como Youtube. A todo esto se debe sumar el tiempo destinado a la búsqueda de la propia información inherente al desarrollo de un videojuego educativo como este, como la referente a los animales o al enfoque que debería tener una herramienta interactiva orientada al aprendizaje.

En un principio la intención de ofrecer la posibilidad de explorar el entorno era literal, ya que la primera versión permitía moverse por un escenario mucho más amplio que el empleado finalmente. La razón principal por la que se descartó esto era para que cualquier persona con unas gafas de realidad virtual simples pudiera jugar sin necesidad de tener que adquirir un controlador o joystick, el cual sería necesario para una exploración inmersiva. De esta forma, solo se requeriría un botón con el que interactuar, el cual viene integrado hasta en las gafas más asequibles.

Contrario a lo que pensaba en un primer momento, este nuevo enfoque no quiere decir que el planteamiento haya empeorado, sino más bien al contrario. La realidad virtual es una herramienta de inmersión en sí misma y el hecho de no poder desplazarte, ciertamente favorece a dicha inmersión, ya que tú en la realidad tampoco estarías haciéndolo.

Como colofón y por todo lo anterior, puedo decir que el desarrollo de este proyecto ha resultado ser una experiencia muy autodidacta en lo que respecta a desarrollo de videojuegos. También ha sido muy enriquecedora por todo el conocimiento que he adquirido además de permitirme demostrar y desarrollar aún más mis aptitudes creativas y de diseño.

7.2 TRABAJO FUTURO

Aunque estoy satisfecho con el proyecto presentado, sé que se puede mejorar. Con más medios y tiempo el resultado podría estar más pulido y aproximarse más al estándar de calidad que debería presentar un videojuego. Asimismo, aún quedarían cosas por hacer. Faltaría la realización de test unitarios y de conjunto de forma sistemática y también un test de usabilidad riguroso con usuarios reales.

La idea detrás de este es mucho más amplia y su desarrollo completo implicaría mucho más trabajo, por no contar con las posibilidades que ofrece a mayores de las presentadas. Creo que este proyecto puede tener una viabilidad real, ya que lo presentado aquí podría entenderse como la idea inicial de lo que podría llegar a ser si se continuara con su desarrollo.

Introducir más hábitats y animales. Añadir una biblioteca interactiva a modo de enciclopedia cuyas entradas iríamos desbloqueando al jugar. Desarrollarlo de forma que permitiera al usuario decidir qué bloques (hábitats) quiere instalar en su dispositivo, de manera que no ocupara espacio de más si no te interesan todos. Crear modos de juego más inmersivos o incluso expandir este concepto a otras áreas de conocimiento como la física o la historia. Estas son algunas de las ideas que se podrían llegar a hacer realidad de plantearse de forma seria su completo desarrollo.

BIBLIOGRAFÍA

[1] Wikipedia. *Wikipedia.* 8 de mayo de 2020. https://es.wikipedia.org/wiki/Videojuego.

[2] Gardey, Julián Pérez Porto y Ana. *Definicion.de.* 2013. https://definicion.de/videojuego/.

[3] Wikipedia. *Wikipedia.* 23 de mayo de 2020. https://es.wikipedia.org/wiki/Videojuego_educativo.

[4] Gimeno, Bárbara. *Bloglenovo.* 20 de febrero de 2018. https://www.bloglenovo.es/los-videojuegos-herramienta-educativa/.

[5] Unity. *Unity.* 2020. https://unity.com/products/unity-platform.

[6] Wikipedia. *Wikipedia.* 11 de septiembre de 2020. https://es.wikipedia.org/wiki/Unity_(motor_de_juego).

[7] Mundo Virtual. *MundoVirtual.* s.f. http://mundo-virtual.com/que-es-la-realidad-virtual/.

[8] Figueroa, Sergio. *Xatakandroid.* 4 de marzo de 2016. https://www.xatakandroid.com/moviles-android/es-recomendable-la-realidad-virtual-para-ninos-dudas-sobre-gafas-vr-de-mcdonalds.

[9] Amazon. *Amazon.* 2019. https://www.amazon.es/Tepoinn-Realidad-Virtual-ajustable-correa/dp/B018YRNMCS.

[10] Micó, Josep Lluís. *La Vanguardia.* 5 de agosto de 2017. https://www.lavanguardia.com/tecnologia/20170805/43352687744/realidad-virtual-educacion-vr-clases-profesores.html.

[11] CMM Castilla-La Mancha Media. *YouTube.* 19 de marzo de 2019. https://www.youtube.com/watch?v=j5Pedi9H8ak&ab_channel=CMMCastilla-LaManchaMedia.

[12] Jara y Sedal. *Revistajaraysedal.* 25 de enero de 2016. https://revistajaraysedal.es/9-cosas-que-no-sabias-de-los-ciervos/.

[13] Top Fauna Salvaje. *YouTube.* 17 de noviembre de 2017. https://www.youtube.com/watch?v=wtCmVe2kXR8&ab_channel=TopFaunaSalvaje.

[14] Romero, Sarah. *MuyInteresante.* s.f. https://www.muyinteresante.es/naturaleza/fotos/curiosidades-sobre-los-buhos/20.

[15] Valenzuela, Isabel. *VIx.* s.f. https://www.vix.com/es/btg/curiosidades/7348/9-datos-fascinantes-sobre-los-zorros.

[16] Martín Rodríguez, Iván. *Análisis narrativo del guion de videojuego.* Madrid: Síntesis, 2015.

[17] Gros, Begoña, y Antònia Bernat. *Videojuegos y aprendizaje.* Barcelona: Graó, 2008.

[18] Marín Díaz, Verónica. *Los videojuegos y los juegos digitales como materiales educativos.* Madrid: Síntesis, 2012.

[19] Iglesias Gorrón, Rodrigo, y Santiago Blanco Suárez. *Videojuegos, un recurso didáctico para nuestro sistema educativo.* Valladolid, 2018.

[20] jprinsloo. *Free3d.* 2 de diciembre de 2017. https://free3d.com/3d-model/free-low-poly-nature-pack-16603.html.

[21] deimosyt. *Free3d.* 21 de agosto de 2016. https://free3d.com/3d-model/blender-lowpoly-nature-assets-pack-36502.html.

[22] CG Cookie. *YouTube.* 4 de junio de 2019. https://www.youtube.com/watch?v=OVbIOHAI3iY&ab_channel=CGCookie.

[23] Erikjuh. *Thingiverse.* 28 de diciembre de 2014. https://www.thingiverse.com/thing:613160.

[24] snippysnappets. *Free3d.* 25 de febrero de 2017. https://free3d.com/3d-model/low-poly-deer-72513.html.

[25] Abbitt, Grant. *YouTube.* 26 de marzo de 2019. https://www.youtube.com/watch?v=6mT4XF-JYq-4&t=654s&ab_channel=GrantAbbitt.

[26] THExPo Graphics. *YouTube.* 24 de mayo de 2018. https://www.youtube.com/watch?v=o0Ur-99kzWnU&ab_channel=THExPoGraphics.

[27] Lundskow, Rick. *YouTube.* 13 de octubre de 2017. https://www.youtube.com/watch?v=kZaw-97yODJ0&ab_channel=RickLundskow.

[28] thepixellab2011. *YouTube.* 27 de octubre de 2016. https://www.youtube.com/watch?v=-UdC-JEXCm6c&ab_channel=thepixellab2011.

[29] aceh3d. *YouTube.* 20 de octubre de 2017. https://www.youtube.com/watch?v=tZiUWMijl-CA&ab_channel=aceh3d.

[30] Bays, Bret. *Cineversity.* 2015. https://www.cineversity.com/vidplaytut/siggraph_2015_rewind_bret_bays_weighting_workflows_for_cinema_4d_character.

[31] CGDreamsTutorials. *YouTube.* 15 de septiembre de 2017. https://www.youtube.com/watch?-v=mH93MdWGCnE&ab_channel=CGDreamsTutorials.

[32] Cineversity. *YouTube.* 3 de agosto de 2017. https://www.youtube.com/watch?v=gnTElmPV-JAs&t=83s&ab_channel=Cineversity.

[33] Megasteakman. *YouTube.* 3 de noviembre de 2012. https://www.youtube.com/watch?v=Hs8A-fIIy6HU&ab_channel=Megasteakman.

[34] Estudio Motovisual. *YouTube.* 23 de julio de 2013. https://www.youtube.com/watch?v=Zsf-bZezhJpE&t=0s&ab_channel=EstudioMotovisual.

[35] Voznesenski, Aleksey. *YouTube.* 17 de noviembre de 2018. https://www.youtube.com/watch?-v=0r-Dkl2ubdU&ab_channel=AlekseyVoznesenski.

[36] Unity Technologies. *assetstore.unity.* s.f. https://assetstore.unity.com/packages/tools/modeling/polybrush-beta-111427.

[37] Unity. *YouTube.* 27 de octubre de 2014. https://www.youtube.com/watch?v=vOaQp2x-io0&ab_channel=Unity.

[38] Unity. *docs.unity3d.* s.f. https://docs.unity3d.com/Manual/index.html.

[39] Unity. *docs.unity3d.* s.f. https://docs.unity3d.com/ScriptReference/index.html.

[40] Varios Autores. *answers.unity.* 12 de septiembre de 2012. https://answers.unity.com/questions/316805/unityeditor-namespace-not-found.html?_ga=2.188379112.1309908389.1599422755-1139412042.1594914754.

[41] Varios Autores. *answers.unity.* 9 de enero de 2018. https://answers.unity.com/questions/1452254/how-to-make-an-object-follow-the-camera-orientatio.html

[42] Varios Autores. *answers.unity.* 21 de noviembre de 2014. https://answers.unity.com/questions/837627/get-width-and-height-of-image-in-unity-46.html?_ga=2.193851115.754646340.1595695116-1139412042.1594914754

[43] Varios Autores. *answers.unity.* 21 de agosto de 2014. https://answers.unity.com/questions/775779/change-size-of-the-new-ui-rect-transform-using-scr.html?_ga=2.193851115.754646340.1595695116-1139412042.1594914754

[44] Varios Autores. *answers.unity.* 31 de octubre de 2017. https://answers.unity.com/questions/1428613/how-to-check-if-gameobject-is-missing.html

[45] Varios Autores. *answers.unity.* 23 de noviembre de 2012. https://answers.unity.com/questions/352942/gameobjectname-to-string.html

[46] Varios Autores. *forum.unity.* 23 de julio de 2019. https://forum.unity.com/threads/no-option-to-create-hdrp-asset.714863/.

[47] Balfaqih, Omar. *YouTube.* 13 de septiembre de 2019. https://www.youtube.com/watch?v=U0d-lWhB_e0E&ab_channel=OmarBalfaqih.

[48] New Tech Tutorials. *YouTube.* 12 de diciembre de 2018. https://www.youtube.com/watch?v=-2DO7rqyiek4&ab_channel=NewTechTutorials.

[49] Palomares, Kiko. *Kikopalomares.* 18 de abril de 2019. https://kikopalomares.com/%F0%-9F%91%BB-que-es-una-funcion-en-programacion-diccionario-del-programador/.

[50] Nolet, Chris. *assetstore.unity.* 16 de julio de 2018. https://assetstore.unity.com/packages/tools/particles-effects/quick-outline-115488.

[51] Infalible Code. *YouTube.* 27 de marzo de 2019. https://www.youtube.com/watch?v=_yf5vzZ2s-YE&t=232s&ab_channel=InfallibleCode.

[52] Seco, Jose. *YouTube.* 26 de enero de 2018. https://www.youtube.com/watch?v=VLslOhPdNrs&-t=187s&ab_channel=JoseSeco.

[53] Unity ClassRoom. *YouTube.* 4 de julio de 2018. https://www.youtube.com/watch?v=eKzKntYG-7Pc&t=1124s&ab_channel=UnityClassRoom.

[54] Unity ClassRoom. *YouTube.* 4 de julio de 2018. https://www.youtube.com/watch?v=5dkKm-6jt-do&ab_channel=UnityClassRoom.

[55] Blackthornprod. *YouTube.* 1 de junio de 2018. https://www.youtube.com/watch?v=ou8VkQB-2sos&t=396s&ab_channel=Blackthornprod.

[56] Android Authority. *YouTube.* 21 de junio de 2018. https://www.youtube.com/watch?v=3gQym-6mF2Jw&t=395s&ab_channel=AndroidAuthority.

[57] Valem. *YouTube.* 30 de julio de 2019. https://www.youtube.com/watch?v=qZzhXHqXM-g&ab_channel=Valem.

[58] Xlaugts. *YouTube.* 27 de mayo de 2018. https://www.youtube.com/watch?v=OEP7sMwfZ-nE&ab_channel=Xlaugts.

[59] KevlarOxy [] The ART of Video Games. *YouTube.* 8 de diciembre de 2019. https://www.youtube.com/watch?v=2AGRMoIXrKo&t=511s&ab_channel=KevlarOxy%5B%5DTheARTofVideoGames.

UVa Universidad de Valladolid

ESCUELA DE INGENIERIAS INDUSTRIALES