TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA EN SISTEMAS DE TELECOMUNICACIÓN

VIRTUAL BIKE: UN SIMULADOR DE BICICLETA EN REALIDAD VIRTUAL

AUTOR: **D. VICTOR MANUEL BUENDÍA EGEA**

DIRECTOR: **Dña. MARIA FRANCISCA ROSIQUE CONTRERAS**

Título:	**Virtual Bike: un simulador de bicicleta en realidad virtual**
Autor:	**D. Víctor Manuel Buendía Egea**
Director:	**Dña. María Francisca Rosique Contreras**
Codirector:	**D. Fernando Losilla López**
Departamento:	**Tecnologías de la información y las comunicaciones**

ÍNDICE

INDICE DE ILUSTRACIONES

1. INTRODUCCIÓN

A continuación se realizará una pequeña introducción del trabajo, en la cual se van a tratar temas como el objetivo principal del trabajo, las motivaciones personales que nos han llevado a su realización, así como la estructura seguida en el trabajo y un pequeño resumen del mismo.

1.1. Distribución de la memoria

La memoria de este trabajo está distribuida en varias partes cómo podemos ver a continuación.

En esta primera sección estamos viendo una pequeña introducción al trabajo, la cual está distribuida en varias subsecciones como son los objetivos y motivación, fases que se han llevado a cabo en el trabajo y un pequeño resumen de la aplicación.

En el segundo apartado que tiene por nombre, 'Estado del arte', vamos a ver temas relacionados con que es y en que consiste la realidad virtual, veremos una pequeña introducción sobre las herramientas utilizadas en el trabajo, también veremos aplicaciones en las que se utiliza realidad virtual para otros usos y por último veremos algún ejemplo de bicicleta virtual en la que comentaremos las diferencias que hay con respecto a nuestro trabajo.

En el tercer punto entramos ya en lo que sería el trabajo, se explica cómo se puede realizar un entorno en Unity de forma muy guiada tal y como si fuera una práctica además de explicar algunas herramientas básicas del programa. También se explicarán los Scripts, Assets y otros componentes usados en la escena del trabajo.

Un cuarto punto explica el circuito de arduino que se ha montado, además se muestran capturas del entorno ya terminado en Unity y del circuito montado. En este apartado también se comentará el código que se ha utilizado para la conexión entre arduino y Unity.

En un quinto punto veremos la ejecución de la escena en una aplicación creada para un sistema Android. Además se verá cómo se debe de configurar Unity para lograr la implementación del trabajo en Android.

Tendremos un sexto apartado en el cual se expondrá la conclusión del trabajo así como mi opinión personal acerca de todo el proceso que me ha llevado a crear este trabajo y lo que me ha podido aportar y podrá aportar en mi enseñanza.

Por último se expondrá la bibliografía utilizada en el trabajo.

1.2. Objetivo y motivación

El uso de la realidad virtual se ha ido aumentando considerablemente en estos últimos años debido al desarrollo de nuevas tecnologías. La realidad virtual se usa hoy en día en

aplicaciones tales como la medicina, (operaciones quirúrgicas), simuladores de pilotaje de vehículos, (coches, aviones, cohetes espaciales…), y otras aplicaciones que veremos más adelante.

El objetivo principal de este trabajo en el momento de su concesión fue la de crear una aplicación mediante un entorno gráfico en la cual el usuario pudiera manejar una bicicleta por un entorno simulado mediante realidad virtual, este entorno elaborado mediante la herramienta de trabajo Unity. La idea era que el usuario pudiera interactuar con la aplicación a través de una bicicleta estática en la que el usuario vaya pedaleando y avanzando a través del entorno virtual.

La idea principal fue la de poder utilizar la aplicación en hospitales con personas mayores aquejadas de enfermedades cerebrales y con niños más pequeños los cuales por distintas causas deben estar en un hospital y a los que con nuestra aplicación podríamos sacarlos un poco de ese mundo. Una vez realizada las pruebas con estas personas la idea era realizar unos test con los que nos contarán como se sentían antes y después de utilizar la aplicación. Teníamos ya contactos para poder probar en el hospital Virgen de La Arrixaca de Murcia pero finalmente por el tema de la pandemia que está asolando el mundo no hemos podido llevarlos a cabo, aunque puede que cuando pase todo este podamos realizarlos.

La motivación principal de este trabajo fue la de poder llevar la aplicación a estas personas y abstraerlos unas horas de la situación que por desgracia les ha tocado vivir. Ahora frente a esta pandemia se nos habré una nueva posibilidad porque en el tiempo que hemos estado encerrados a cualquiera le hubiera gustado poder dar un paseo aunque sea sin salir de casa y esto lo podríamos hacer mediante nuestra aplicación.

Otra motivación muy importante a la hora de llevar a cabo el proyecto fue la de aprender a realizar entornos gráficos de realidad virtual ya que pensamos que en el futuro va a ser una de las principales herramientas que se utilizarán en el mundo y será necesario saber implementar entornos virtuales.

1.3. Fases del TFG

En esta parte veremos las principales fases que se han llevado a cabo a la hora de realizar este trabajo.

- Elección del TFG a realizar y decisión del tipo de simulador que íbamos a realizar en nuestro trabajo.
- Estudios de los distintos tipos de herramientas gráficas entre las que se podía elegir para realizar el entorno gráfico y tomar la decisión sobre cual utilizar.
- Visualización de varios tutoriales para aprender a trabajar con la herramienta elegida, ya que este trabajo se empezaba sin ningún conocimiento de la herramienta.

- Realización de varias pruebas en la herramienta elegida para la consecución de un entorno gráfico que nos gustará para el trabajo.
- Estudio y elección de otras herramientas utilizadas en el trabajo, como por ejemplo la placa de arduino con los módulos necesarios para darle movilidad a la bicicleta.
- Programación y montaje de la placa arduino para darle la utilidad deseada en los pedales de la bicicleta.
- Unión de la parte física, placa arduino, con la parte virtual, entorno gráfico Unity, mediante puerto serie.
- Inclusión de las gafas Oculus en nuestra aplicación, a través de las cuales tendremos la experiencia en realidad virtual.
- Realización de diferentes pruebas para ver y mejorar el funcionamiento de la aplicación.
- Realización de la memoria, la cual se ha ido realizando poco a poco a lo largo del trabajo.

1.4. Resumen

Desde el primer momento que tuvimos la idea de realizar este trabajo el objetivo siempre fue el de crear una aplicación de realidad virtual la cual simule la conducción de una bicicleta por un parque u otro entorno virtual realizado a través de la herramienta Unity. Para la comunicación entre la bicicleta estática o pedales, nos basamos en la utilización de una placa arduino con la cual a través de un módulo incorporado de detección a través de infrarrojos, el cual detectará cada paso del pedal lo cual indica que la bicicleta debe avanzar. Por último también se ha utilizado las gafas de realidad virtual 'Oculus' las cuales nos permitirán reproducir la aplicación cuando el usuario se monte en la bicicleta.

2. ESTADO DEL ARTE

En esta sección veremos varios puntos, en los cuales se explica que es la realidad virtual, se explicará un poco de historia sobre las herramientas utilizada para realizar el trabajo, también se verán algunos ejemplos de otros usos que tiene la realidad virtual en la vida cotidiana y por último veremos otros simuladores de bicicleta en realidad virtual que hayan sido implementados explicando las diferencias con el nuestro.

2.1. ¿Qué es la realidad virtual?

A continuación vamos a tratar de responder esta pregunta, que resulta básica y transcendental para el transcurso del trabajo y la memoria, ya que nuestro simulador está basado en esta tecnología.

La definición que la RAE da a este concepto es la siguiente: representación de escena o imágenes de objetos producida por un sistema informático, que da la sensación de existencia real **[1]**. Sin embargo, desde nuestro punto de vista se puede decir que la realidad virtual consiste en la inmersión sensorial en un mundo nuevo, basado en entornos reales o no, que ha sido generado de forma artificial y que podemos percibir gracias a unas gafas de realidad virtual. El objetivo de esta tecnología es crear un mundo ficticio del que puedes formar parte e incluso ser el protagonista: viendo un coche en un concesionario virtual, siendo protagonista de un videojuego o bien practicando como hacer una operación a corazón abierto en caso de ser estudiante de medicina o médico.

Como pasó con los teléfonos móviles o internet, la aparición de la realidad virtual supone uno de los cambios tecnológicos más importantes de los últimos tiempos. Aunque aún no seamos demasiado conscientes, por la falta de medios para probarlo y también por la escasez actual de aplicaciones desarrolladas, (las cuales son cada vez más numerosas y más complejas), la realidad virtual y su adaptación a nivel de usuario supondrá un antes y un después en la forma en la que consumimos contenidos multimedia: videojuegos, cine, eventos deportivos, conciertos, documentales, etc…**[2]**

La realidad virtual comprende dos componentes principales: el entorno del usuario y el entorno virtual. Mientras que el usuario interactúa con el sistema de realidad virtual, los dos entornos se comunican e intercambian información a través de una barrera llamada interfaz. La interfaz puede considerarse como un traductor entre el usuario y el sistema de realidad virtual. Cuando el usuario aplica acciones de entrada, (por ejemplo, movimiento, generación de fuerza, voz, etc…), la interfaz traduce estas acciones en señales digitales, que pueden ser procesadas e interpretadas por el sistema. Por otro lado, las reacciones calculadas del sistema también se traducen por la interfaz en cantidades físicas, que el usuario puede percibir mediante el uso de diferentes tecnologías de pantalla y actuador, (por ejemplo, imágenes, sonidos, olores, etc…). Finalmente, el usuario interpreta esta información y reacciona al sistema en consecuencia **[3]**.

2.2. Herramientas utilizadas

Ahora veremos que son y para qué sirven cada una de las herramientas utilizadas en el trabajo, tanto el programa informático utilizado para realizar el entorno virtual como los componentes físicos que se utilizarán.

2.2.1. Unity 3D

Unity es un motor de videojuegos multiplataforma creado por Unity Technologies. Unity es una plataforma líder en la industria del desarrollo de videojuegos y entornos gráficos. Esta plataforma está disponible para distintos sistemas informáticos como Microsoft Windows, OSX o Linux. La plataforma de desarrollo tiene soporte de compilación con diferentes tipos de plataformas.

Ilustración 1: Logo de Unity

La empresa Unity Technologies fue fundada en 2004 por David Helgason (CEO), Nicholas Francis (CCO) y Joachim Ante (CTO) en Copenhague. La primera versión de Unity se lanzó en la Conferencia Mundial de Desarrolladores de Apple en 2005. Fue construido exclusivamente para funcionar y generar proyectos en los equipos de la plataforma Mac y obtuvo el éxito suficiente como para continuar con el desarrollo del motor y sus herramientas. Unity 3 fue lanzado en Septiembre de 2010 y se centró en empezar a introducir más herramientas que los estudios de alta gama por lo general tienen a su disposición, con el fin de captar el interés de los desarrolladores más grandes, mientras que proporciona herramientas para equipos independientes y más pequeñas que normalmente serían difíciles de conseguir en un paquete asequible. La última versión de Unity, Unity 5, lanzada a principios de 2015, se anunció en Game Developers e incluye añadidos como Mecanim animation, soporte para DirectX 11 y soporte para juegos en Linux y arreglo de bugs y texturas. Desarrollado por creadores de juegos para mayor expectativa **[4]**.

La principal razón que se tuvo en cuenta a la hora de elegir este motor de videojuegos en el trabajo en contraposición de otros programas disponibles son las características que nos ofrece Unity como, su fácil manejo y la gran cantidad de tutoriales y documentación que hay para trabajar con esta herramienta sin haber trabajado antes con ningún otro motor de videojuegos. Algunas de las principales características de Unity son:

- **Optimización de tiempo y características multiplataforma:** con Unity 3D 5, (la última versión), puedes exportar tu entorno gráfico a más de 20 plataformas en un solo clic, (dispositivos móviles, consolas, ordenadores, televisores, web, gafas de realidad virtual, etc…).

- **Store de assets:** Los assets son los elementos que componen el videojuego, tales como animaciones, modelos, sonidos, etc. Unity 3D dispone de un store en el que se puede encontrar prácticamente cualquier cosa que se necesite para desarrollar el entorno gráfico, desde proyectos completos a funcionalidades concretas.
- **Potencia en todos los entornos:** Unity 3D se caracteriza por su potencia tanto en entornos 2D como en entornos 3D.
- **Interfaz sencilla y fácil manejo:** Cuenta con una interfaz muy sencilla que ayuda a los desarrolladores a probar sus creaciones directamente en el programa pudiendo ver el resultado final de su entorno y modificarlo en el propio motor **[5]**.

2.2.2. Arduino

Arduino es una plataforma de creación electrónica de código abierto, la cual está basada en hardware y software libre, flexible y fácil de utilizar para los creadores y desarrolladores. Esta plataforma permite crear diferentes tipos de microordenadores de una sola placa a los que la comunidad de creadores puede darles diferentes tipos de uso. El hardware libre son los dispositivos cuyas especificaciones y diagramas son de acceso público, de manera que cualquiera puede replicarlos. El software libre son los programas informáticos cuyo código es accesible por cualquiera para que quien quiera pueda utilizarlo y modificarlo.

Ilustración 2: Logo Arduino

Arduino ofrece la plataforma Arduino IDE, (Entorno de Desarrollo Integrado), que es un entorno de programación con el que cualquiera puede crear aplicaciones para las placas Arduino, de manera que se les puede dar todo tipo de utilidades.

El proyecto nació en 2003, cuando varios estudiantes del Instituto de Diseño Interactivo de Ivrea, Italia, lo llevaron a cabo con el fin de facilitar el acceso y uso de la electrónica y programación. Lo hicieron para que los estudiantes de electrónica tuviesen una alternativa más económica a las populares BASIC Stamp, unas placas que por aquel entonces valían más de cien dólares, y que no todos se podían permitir. El resultado fue Arduino, una placa con todos los elementos necesarios para conectar periféricos a las entradas y salidas de un microcontrolador y que puede ser programada tanto en Windows como macOS y GNU/Linux.

Los microcontroladores son circuitos integrados en los que se pueden grabar instrucciones, las cuales las escribes con el lenguaje de programación que puedes utilizar en el entorno Arduino IDE. Estas instrucciones permiten crear programas que interactúan con los circuitos de la placa.

El microcontrolador de Arduino posee lo que se llama una interfaz de entrada, que es una conexión en la que podemos conectar en la placa diferentes tipos de periféricos. La información de estos periféricos que se conecten se trasladará al microcontrolador, el cual se encargará de procesar los datos que le lleguen **[6]**.

En nuestro proyecto la funcionalidad principal que va a tener la placa de Arduino va a ser la hacer que la bicicleta virtual se mueva a la vez que nosotros movamos los pedales en la realidad. Para ello hemos necesitado incluir un periférico al Arduino como detector infrarrojo, el cual colocado en la parte inferior del pedal, detectará cada vez que pase el pedal por encima y se enviará a Unity para que la bicicleta virtual avance, aunque esto se explicará detenidamente más adelante.

2.2.3. Oculus Rift

Oculus Rift es un casco de realidad virtual creado por la compañía estadounidense Oculus VR, esta empresa ha invertido unos 91 millones de dólares para el desarrollo de Oculus Rift. La versión de Oculus Rift se lanzó al mercado entre los meses de Marzo y Abril de 2016 por un precio que rondaba los 600 dólares en Estados Unidos.

Ilustración 3: Logo de Oculus

Oculus VR es una empresa estadounidense que desarrolla tecnología de realidad virtual, fundada por Palmer Luckey, Brendan Iribe, Michael Antonov, Jack McCauley y Nate Mitchell en Julio de 2012 en Irvine, California. En Abril de 2012 Palmer Luckey anuncio su primer producto que sería Oculus Rift el cual no llegó hasta el consumidor hasta el 28 de Marzo de 2016.

En Noviembre de 2015 la compañía se asoció con Samsung para desarrollar Samsung Gear VR para el móvil Samsung Galaxy 6. En Octubre de 2017 la compañía agregó dos nuevos productos a los ya existentes como son Oculus Go y Oculus Quest. La última versión de Oculus Rift se sacó a la venta en el segundo trimestre de 2019 por un precio que rondaba los 400 dólares. Esta empresa solo se dedica al desarrollo de gafas de realidad virtual, no hace móviles ni ningún otro tipo de equipamiento tecnológico **[7]**.

A continuación veremos algunas de las principales características de las Oculus Rift:

- Resolución de 800x1280 píxeles en cada ojo y un ángulo de visión de 100º por ojo.

- Se conecta mediante cable HDMI, USB o DVI al ordenador.
- Cada movimiento sutil de la cabeza se realiza un seguimiento en tiempo real creando una experiencia natural e intuitiva. El campo de visión es de más de 90º horizontales y de 110º de visión diagonal.
- Acelerómetros, magnetómetros y giroscopio de 3 ejes.
- La pantalla tiene un tamaño de 5,6 pulgadas.
- Gracias a una cámara externa se percibe el movimiento de todo el cuerpo. Es decir, puedes inclinarte para acercarte a los paneles de control o echarte a un lado para mirar hacia una esquina **[8]**.

2.3. Ejemplos de simuladores de realidad virtual

A continuación veremos algunos ejemplos en los que la realidad virtual se ha usado en simuladores que podemos tener presentes en la vida cotidiana, o que nos podríamos encontrar según el tipo de trabajo al que nos dedicáramos.

2.3.1. Marion Surgical

Esta es una compañía surgió en Septiembre de 2016 en Toronto, Canadá. Esta empresa nace por la colaboración de Ben Sainsbury (CEO) y Rajiv K Singal (CMO), Ben es un desarrollador en realidad virtual con un doctorado en ciencias de la computación y Rajiv es cirujano Urológico en el hospital Michael Garron y profesor asistente en la universidad de Toronto.

El objetivo de la compañía fue el de crear un simulador quirúrgico de realidad virtual diseñador para ayudar a los cirujanos, o futuros cirujanos a practicar operaciones. El simulador quirúrgico realizado por la compañía utiliza la realidad virtual en el dispositivo para colocar al usuario frente a una mesa de operaciones simulada, esto permite a los cirujanos o estudiantes una formación virtual antes de hacerlo en la realidad **[9]**.

En la Web de la empresa podemos ver como expresan lo que sienten los cirujanos cuando están estudiando. Los cirujanos dicen que están frustrados con las prácticas de formación quirúrgica que reciben, ya que esperan que sus experiencias sean dinámicas, innovadoras y realistas y esto no era así. Marion Surgical trabaja con cirujanos de todo el mundo para construir un conjunto de simuladores quirúrgicos avanzados.

Marion Surgical pretende permitir a los cirujanos a través de la realidad virtual aprender, colaborar, practicar y compartir procedimientos en un entorno realista, seguro y alojado en la nube.

En la actualidad la empresa solo dispone del simulador llamado K181, este es un simulador quirúrgico de realidad virtual para entrenar procedimientos de acceso percutáneo realizados bajo fluoroscopia en tiempo real. Estas son algunas de las cosas que se pueden simular:

- Predicción del acceso calicial para nefrolitotomía percutánea con tomografía computarizada.
- Determinación del ángulo, la profundidad y la punción para el acceso renal percutáneo guiado por fluoroscopia.
- Técnicas de punción en ojo de buey y triangulación.
- Navegación de imágenes de ARM y fluoroscopia.
- Manipulación de herramientas y cables guía.
- Medición y puntuación basada en protocolos manuales y de toma de decisiones.

Ilustración 4: Simulador K181

Próximamente estará disponible el simulador de PTE, este será un simulador interactivo para el entrenamiento de la tromboendarterectomía pulmonar, una cirugía que elimina los coágulos de sangre en las arterias pulmonares. Algunas de las características del mismo serán:

- Un simulador interactivo para ayudar a enseñar y practicar la cirugía PEA de forma segura.
- Usando un estereoscopio y dispositivos hápticos, el simulador PTE proporciona visión 3D y fuerza de retroalimentación para sumergir a los usuarios.
- Las fuerzas y deformaciones debidas a las interacciones del usuario se calculan en tiempo real utilizando GPU.
- Las métricas de rendimiento se proporcionan al usuario y al instructor **[10]**.

2.3.2. Simuladores Vesaro

Vesaro es una empresa británica de diseño y fabricación de simuladores con sede en Kent, Inglaterra. Los simuladores que construye Vesaro son de los más avanzados del mundo. Vesaro diseña y construye simuladores de vuelo, simuladores de helicópteros y simuladores de Fórmula 1.

El sistema de Vesaro se basa en un núcleo central ultrafuerte hecho a mano que se puede adaptar desde el sistema central hasta un simulador de física de movimiento de pantalla completa. Hay miles de configuraciones posibles que hacen del sistema de Vesaro el primer sistema simulador de grado comercial verdaderamente modular del mundo **[11]**.

A continuación vamos a ver dos de sus principales simuladores, el simulador de vuelo Vesaro 195 Flight y el simulador de carreras de Vesaro que tienen varios pero veremos el más básico y económico.

Vesaro 195 Flight: Es un simulador altamente sofisticado para el entrenamiento de simulación profesional, utilizando un conjunto de sistemas de control de vuelo que se pueden aplicar a múltiples aeronaves. El realismo y la inmersión se logran mediante un sistema de entrega sensorial multidimensional. Un sistema de audio de sonido envolvente proporciona la recreación de los sonidos de la aeronave y del entorno, mientras que un sistema de retroalimentación táctil proporciona la vibración física que se siente durante el vuelo. Las imágenes son proporcionadas por una pantalla envolvente de 195 pulgadas de triple curva inmersiva que brinda al usuario un entorno visual envolvente. El movimiento de todo el simulador al completo se logra utilizando la tecnología D-BOX para una recreación precisa del movimiento en 3 tipos de movimiento inteligente, cabeceo sutil, balanceo y elevación, proporcionando señales cinestésicas para desarrollar reflejos y preparándolos para el vuelo en la vida real. Vesaro 195 Flight es el primer y único simulador de vuelo verdaderamente modular del mundo.

Este simulador cuesta aproximadamente 55 mil Libras, ya que es un simulador pensado para que grandes compañías entrenen a sus pilotos. Algunas de sus características más destacadas son:

- **Sistema Yoke:** está altamente diseñado para satisfacer las exigentes necesidades de los pilotos profesionales, proporcionando una experiencia de vuelo suave, preciso y muy realista. El Yoke tiene un giro completo de 90 grados desde el centro, lo que brinda una mayor precisión al volar.
- **Multipanel:** Se puede controlar el tren de aterrizaje, la potencia del motor, las luces de aterrizaje y otras 11 funciones importantes de la aeronave desde una unidad compacta.
- **TPM:** el Pro Flight Throttle, replica la superficie de control que se encuentra en los aviones ligeros. Tres varillas de aluminio permiten a los usuarios alterar con precisión los niveles de acelerador, la hélice y la mezcla de combustible.

- **Pantallas LED del panel de radio:** La pantalla del piloto automático funciona en tiempo real, eliminando la necesidad de utilizar controles complicados en la pantalla.
- **Panel de instrumentos:** Con una pantalla LCD en color de 3,5 pulgadas, el panel de instrumentos puede mostrar 6 de los paneles de instrumentos de la cabina principal del software de simulación de vuelo.
- **Controles de vuelo hotas warthog:** Paquete de joystick de réplica del avión de ataque Hotas de la fuerza aérea de Estados Unidos **[12]**.

Ilustración 5: Simulador de vuelo Vesaro 195 Flight

Simulador de carreras de Vesaro: Simulador de realidad virtual profesional con la flexibilidad de una gran pantalla curva que se puede utilizar como pantalla principal de carreras cuando no se usan las gafas de realidad virtual. Es un simulador diseñado para clientes privados que requieren componentes de grado Motorsport de alga gama para entrenamiento personal de pilotos de carreras o entusiastas del hogar que requieren de los mejores simuladores, ya que el simulador más básico ronda las 23 mil Libras de precio.

El simulador viene ensamblado de fábrica y todos los sistemas están completamente integrados a bordo para crear un aspecto compacto y elegante con un funcionamiento sencillo. El simulador incluye un PC a bordo construido a medida que utiliza los mejores componentes seleccionados para un rendimiento de simulación óptimo.

De forma predeterminada, el audio lo proporciona el casco de realidad virtual integrado que ofrece una experiencia de audio discreta pero envolvente, además de hacer partícipe al usuario de estar dentro de la carrera. A continuación veremos dos ejemplos de estos simuladores el más básico y el más alto que puede rondar las 48 mil Libras **[13]**.

Ilustración 6: Simulador básico Vesaro Stage 1

Ilustración 7: Simulador avanzado Vesaro Stage 5

2.3.3. Birdly VR

Por último veremos este simulador de vuelo en primera persona, porque todos hemos tenido el sueño de poder volar como si fuéramos un ave. Birdly nació en 2013 como un proyecto de investigación de diseño en la universidad de las artes de Zurich, Suiza. Los primeros prototipos se demostraron en eventos de exhibición como el festival de cine de Sundance obteniendo innumerables premios de realidad virtual.

La compañía Somniacs fue fundada para apoyar el despliegue comercial de Birdly, así como para desarrollar la próxima generación de realidad virtual, realidad aumentada y experiencias interactivas. La empresa D3D Cinema especializada en cine inmersivo y experiencias de realidad virtual es la distribuidora oficial de Birdly en América del Norte.

A diferencia de un simulador de vuelo común que requiere de un joystick o numerosos botones, Birdly comanda todo el cuerpo por completo con una experiencia de vuelo en realidad virtual instintivamente con los brazos y las manos movimientos correlacionados con el aleteo de las alas. Inmersos visualmente a través de una pantalla de realidad virtual montada en la cabeza, los volantes están envueltos en un paisaje virtual de alta resolución cargado de zonas interactivas.

Desde una vista de pájaro cobre las grandes ciudades y monumentos del mundo, a una vista de abeja del funcionamiento interno de un motor de combustión, a una vista de pterosaurio de un paisaje jurásico hay varios entornos en los que se puede disfrutar de la experiencia.

Birdly Serial Edition es un paquete de realidad virtual comercial confiable, de bajo mantenimiento y fácil de usar, que incorpora diseño de precisión suiza e ingeniería alemana. La unidad integra todos los componentes técnicos desde actuadores y sensores de simulación hasta CPU de alto rendimiento en un chasis metálico compacto y elegante. El precio de Birdly según su distribuidor D3D Cinema está en torno a 135 mil dólares más una serie de aranceles si tiene que ser transportado, un coste inalcanzable para gente humilde **[14]**.

Ilustración 8: simulador de vuelo Birdly

2.4. Bicicleta en realidad virtual

A continuación veremos un proyecto realizado por Paul Yan en el cual también implemento un simulador de bicicleta en realidad virtual a través de Unity y Arduino. Paul a diferencia de nuestro trabajo hizo una bicicleta en realidad virtual con una bicicleta real suspendida en el aire en la cual bajo la rueda trasera colocó la placa arduino con un detector infrarrojo distinto al nuestro para detectar cada paso de la rueda por él y así hacer que la bici avance.

El entorno creado por Paul en Unity también es distinto del nuestro ya que el creo un entorno mucho más de animación, como si estuviéramos dentro de un juego o una serie de dibujos animados. Paul para lograr reproducir su aplicación utiliza unas gafas virtuales mucho más sencillas que las Oculus Rift que utilizamos nosotros en las cuales debe de cargar su entorno en un móvil y después meter el móvil dentro de las gafas utilizadas para reproducirlo.

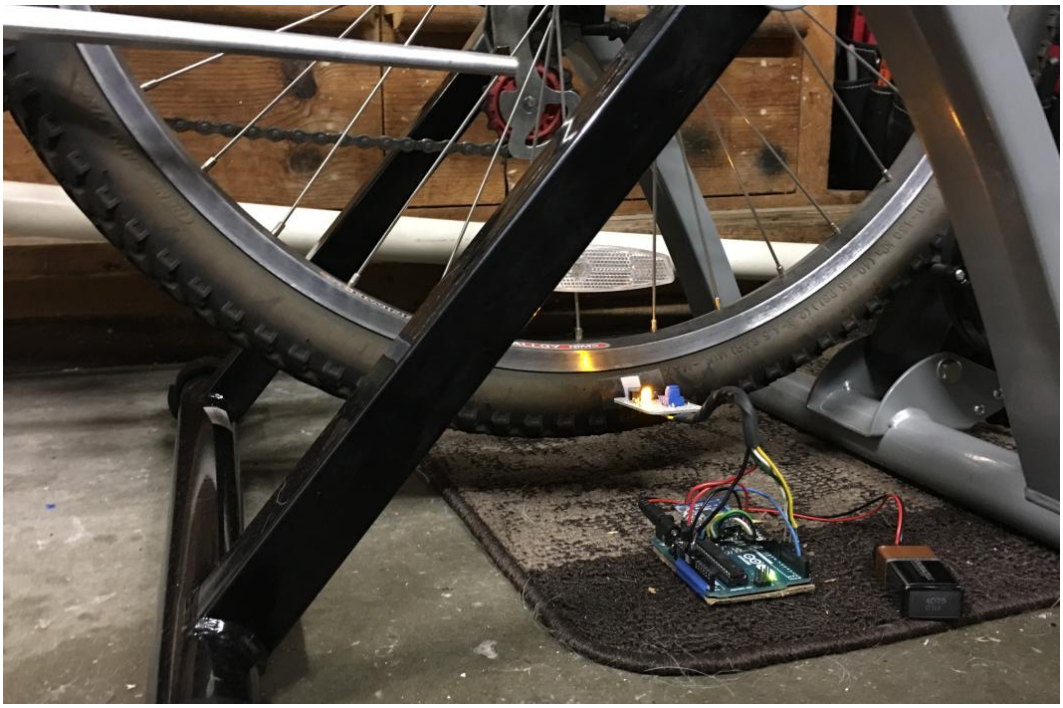

Ilustración 9: Placa de arduino con sensor infrarrojo colocada en la rueda

En la imagen anterior podemos ver como Paul colocó la placa de arduino con el sensor infrarrojo bajo la rueda de la bicicleta para detectar cada vuelta de la rueda, a diferencia del nuestro el cual colocaremos un sensor infrarrojo diferente bajo el pedal.

En la siguiente imagen podremos observar el entorno gráfico creado por Paul en Unity que cual se parece a una ciudad de dibujos animados.

Ilustración 10: Entorno de Unity creado por Paul

Paul en su trabajo tuvo que establecer una conexión entre Arduino, Unity y su móvil ya que como se ha dicho, en sus gafas iría insertado el móvil con la aplicación a reproducir, a diferencia del nuestro que solo debemos de conectar las gafas Oculus al ordenador y reproducir nuestra aplicación. Ahora podemos ver unas imágenes de cómo se ve en sus gafas la aplicación creada además de las gafas que utilizó.

Ilustración 11: Gafas de realidad virtual utilizadas por Paul

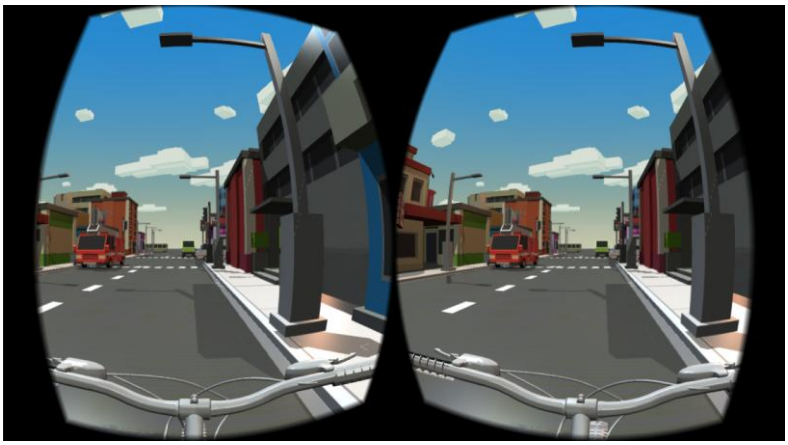

Ilustración 12: Vista de la aplicación de Paul a través de las gafas

Por último decir que la aplicación de Paul aunque utilice los mismos o muy parecidos elementos electrónicos que en nuestro trabajo se diferencia claramente del mismo, ya que nosotros utilizamos distinto casco de realidad virtual, utilizaremos un entorno creado en Unity muy diferente al de Paul, además de que los periféricos usados en la placa de arduino serán diferentes a los utilizados por Paul. En esta última imagen veremos a Paul utilizando su aplicación **[15]**.

Ilustración 13: Paul utilizando su aplicación

3. DESARROLLO DEL SIMULADOR

En esta sección veremos cómo se ha desarrollado todo el trabajo realizado para conseguir finalmente el simulador de bicicleta en realidad virtual. Vamos a explicar cómo funciona Unity y sus herramientas más utilizadas, también veremos cómo hemos utilizado Arduino en nuestro trabajo y mostraremos imágenes de todo el proceso de elaboración.

3.1. Desarrollo de la Práctica en Unity

La principal idea del proyecto era poder probar nuestra aplicación con gente que estaba en hospitales, ya hubieran sido personas mayores las cuales podrían probar la aplicación directamente, o bien personas más jóvenes. Con las personas más jóvenes aparte de probar nuestra aplicación también nuestra idea era enseñarlas a crear un entorno en Unity desde cero, por tanto en esta sección vamos a explicar cómo se puede crear un primer entorno sencillo en Unity como si fuera un pequeño tutorial, con varías capturas para poder ir viendo el funcionamiento de Unity. Si algún día se pudiera nos gustaría poder probarlo con dichas personas y que esto les sirva como una guía práctica además se podría ampliar para crear entornos más complejos en varias sesiones.

3.1.1. Descarga de Unity 5

Para empezar a trabajar con Unity, lo primero que se debe de hacer es descargar Unity 5 en la página oficial de Unity, ya que es una descarga gratuita, y crear una cuenta registrándose, en esa cuenta quedarán guardados todos los proyectos que hagamos y solo iniciando sesión podremos acceder al que queramos. A continuación se muestra cómo se puede descargar Unity de la página oficial de Unity, lo primero que debemos hacer es acceder a la página oficial de Unity y darle a la opción de Elige tu Unity + descargar. Una vez realizado esto se nos abrirá una nueva ventana donde tendremos que darle a la pestaña personal y elegir la opción comencemos. Una vez realizado esto podremos descargar Unity y crearemos una cuenta a través de nuestro correo electrónico **[16]**.

Ilustración 14: Descarga de Unity

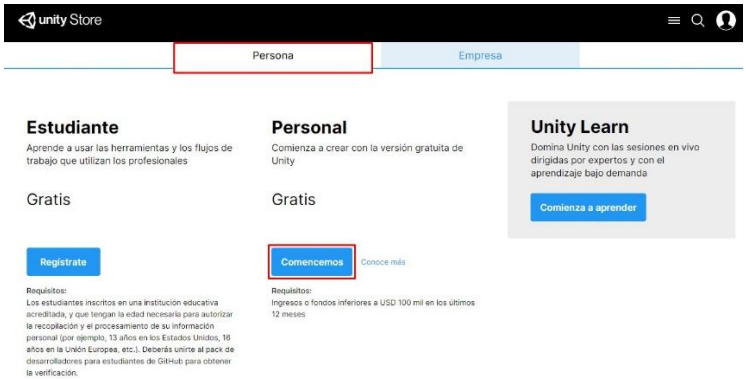

Ilustración 15: Descargar Unity de manera personal

Por último cuando ya tengamos descargado Unity crearemos una cuenta a partir de nuestro correo electrónico simplemente pulsando en regístrate, en esta cuenta quedarán disponibles todos los proyectos que realicemos.

3.1.2. Nuevo Proyecto

Una vez abramos sesión en Unity con nuestro usuario, podemos tener proyectos ya creados, si no es la primera vez que usamos Unity, los cuales podemos modificar o seguir trabajando sobre ellos o bien podemos crear un nuevo proyecto. Este último sería nuestro caso por tanto para crear un nuevo proyecto hacemos clic sobre 'New', como se puede observar en mi cuenta ya tengo varios proyectos que yo he creado anteriormente.

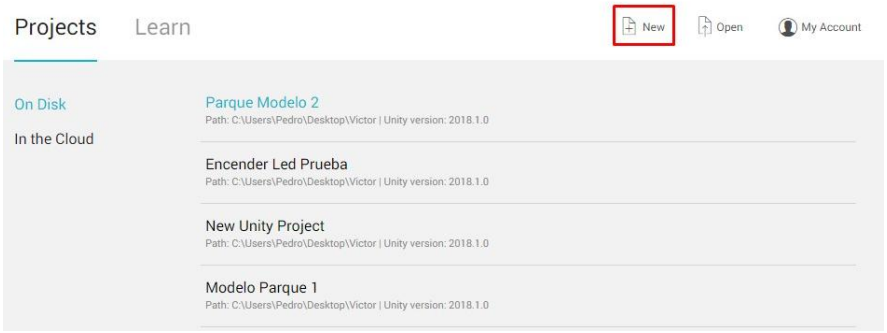

Ilustración 16: Nuevo proyecto Unity

Ahora se nos abrirá una nueva ventana donde tenemos que indicar el nombre del nuevo proyecto, la localización donde queremos que se guarde este proyecto y si queremos un proyecto en 3D, 2D, 3D With Extras… En nuestro caso elegimos 3D.

Ilustración 17: Selección de plantilla del nuevo proyecto

También podemos añadir Assets que podemos tener descargados de proyectos anteriores. En este caso yo me voy a importar todos los Assets que ya tengo de otros proyectos aunque no haría falta para este primer entorno. Para importar los Assets pulsamos sobre 'Add Asset Package' y se abrirán todos los paquetes de Assets que ya teníamos descargados. En este caso como los quiero importar todos marco la opción 'Select All' y hago clic sobre 'Done'.

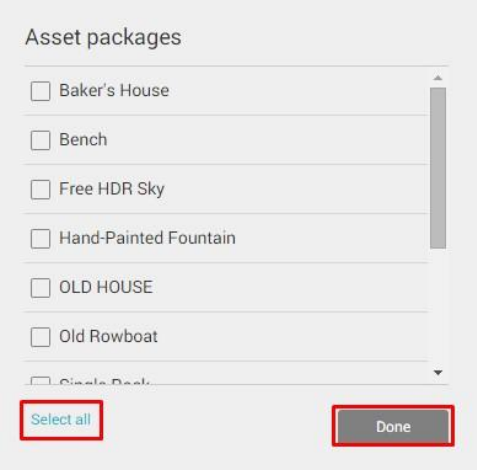

Ilustración 18: Selección de Assets a importar

Por último hacemos clic sobre 'Create Proyect' y se creará nuestro nuevo proyecto en el cual empezaremos a ver cómo crear un entorno en Unity desde cero.

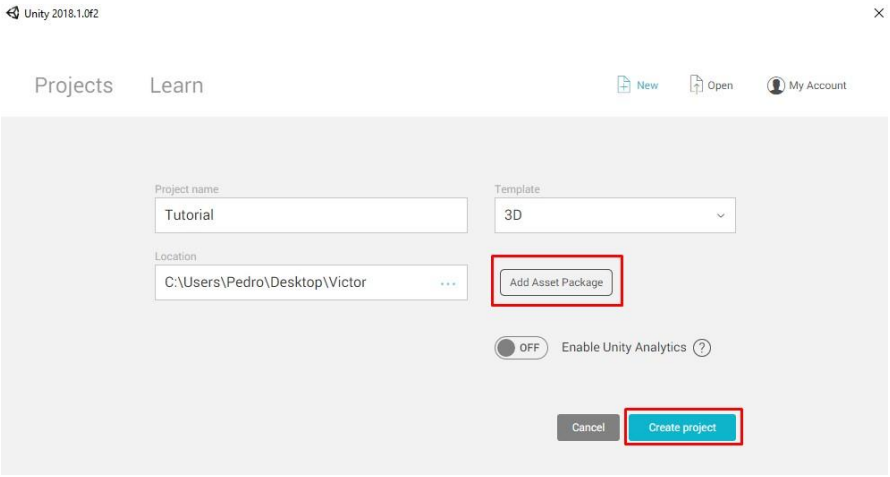

Ilustración 19: Crear proyecto

3.1.3. Herramientas Unity

A continuación veremos el nuevo proyecto que hemos creado en Unity vacío y vamos a explicar que son y para qué sirven cada una de las ventanas y herramientas que tenemos. Cada una de estas ventanas o herramientas están marcadas con colores para poder distinguirlas e ir explicándolas a continuación.

En azul podemos ver las pestañas, Scene, Game, Asset Store y Animator. Según la pestaña que marquemos en la ventana principal nos aparecerá una cosa u otra. Si marcamos Scene tendremos la escena en 2D del entorno en el que estamos trabajando en nuestro caso aparece vacía ya que aún no hemos colocado nada. Si marcamos Game lo que se verá será el entorno en una vista 3D como si estuviéramos ya con las gafas

puestas. En la pestaña Asset Store entraremos en la tienda de Assets donde podremos buscar y filtrar Assets que nos hagan falta para nuestro proyecto y por último en la pestaña Animator podemos configurar un Assets de nuestro entorno para darle movimiento siguiendo una serie de pasos. Por último justo encima de estas pestañas podemos ver la botonera típica de Play y Pause, cuando estemos realizando nuestro entorno estos botones nos ayudaran a reproducir nuestra escena como si estuviéramos viéndolo dentro las gafas y así nos irá ayudando para corregir errores.

En rojo podemos ver las opciones que tenemos para desplazarnos por nuestra escena aunque esto es ir cogiendo práctica y al final usar el que más nos guste. Según la opción que tengamos podremos seleccionar Assets y moverlos de una manera u otra.

En lila podemos ver la ventana Hierarchy, En esta ventana es donde vamos a tener todos los elementos o Assets que vayamos añadiendo en nuestro entorno, es aconsejable crear carpetas para ir organizando nuestros Assets. Cada vez que queramos crear una carpeta es tan fácil como clicar con el botón derecho del ratón dentro de la ventana Hierarchy y pulsa sobre la opción Create Empty, después podemos cambiar el nombre de esta carpeta y arrastrar todos los Assets que queramos meter en ella.

En verde tenemos una ventana donde podemos abrir como en este caso todos Assets que tenemos importados a nuestro proyecto o bien también podríamos abrir la consola o el animation para nuestro proyecto aunque estás dos opciones no se verán de momento.

En color naranja tenemos la ventana Inspector, esta es una de las ventanas más importante de Unity ya que va ser donde tenemos que configurar nuestros Assets para que hagan lo que nosotros queramos o añadirles cosas que necesitamos, aunque ahora mismo aparece vacía ya que no tenemos ningún Assets seleccionado.

Por último arriba en amarillo tenemos una tabla de herramientas con varias pestañas de las cuales aparece un menú desplegable y que vamos a explicar a continuación.

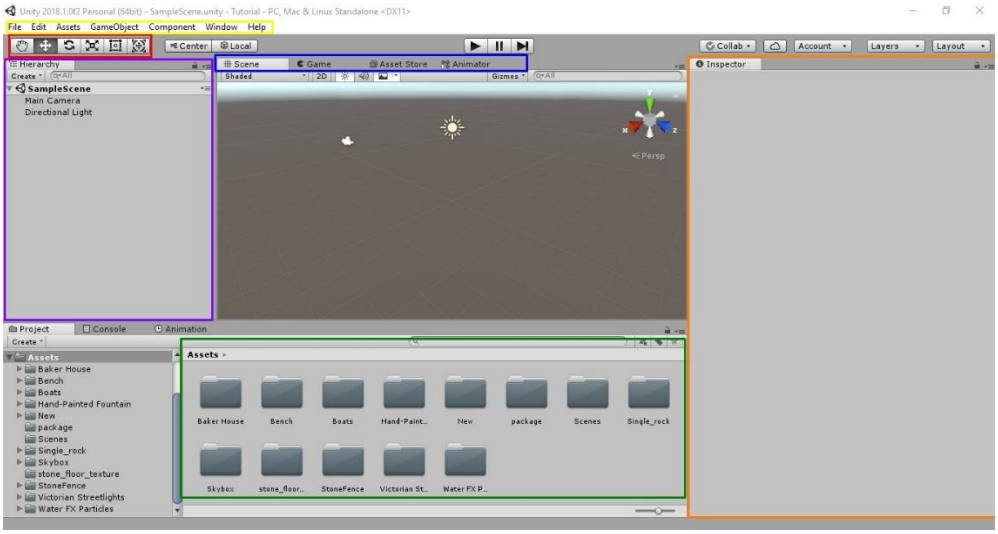

Ilustración 20: Escena vacía nuevo proyecto

Solo vamos a ver las primeras 4 pestañas de la barra de herramientas ya que van a ser las que vamos a usar en este primer entorno. La pestaña File es la típica pestaña de archivo que tenemos en casi todos los programas usados en los ordenadores. En esta pestaña podremos crear una nueva escena o abrir una nueva escena, guardar la escena, abrir un proyecto o crear uno nuevo y guardarlo, podemos compilar la escena y también podremos salir.

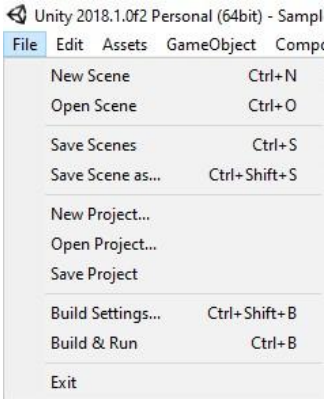

Ilustración 21: Pestaña File

En la pestaña Edit podemos editar la escena, podemos copiar, cortar o pegar elementos, podemos duplicar o eliminar elementos, podemos seleccionar todo o buscar algún elemento que no encontramos, también desde aquí podemos darle a Play o Pause de la escena y también podríamos modificar la configuración de varios elementos del programa.

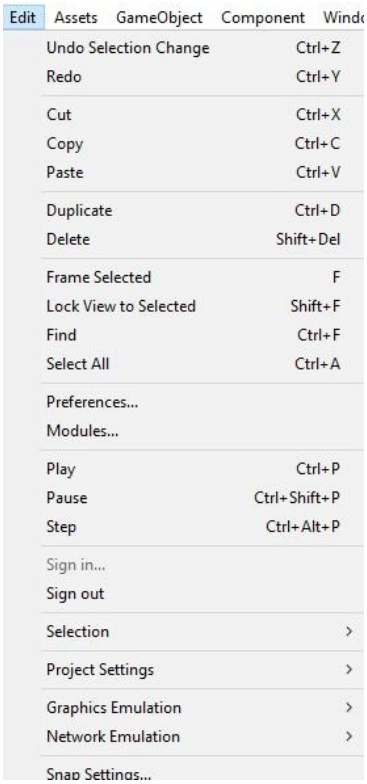

Ilustración 22: Pestaña Edit

La pestaña Assets es una de las que más vamos a utilizar, en esta pestaña podemos crear un nuevo Asset, podemos eliminar, abrir o renombrar un Asset, también podemos importar o exportar un Asset o un paquete de Asset, refrescar o actualizar y reimportar todos los Assets.

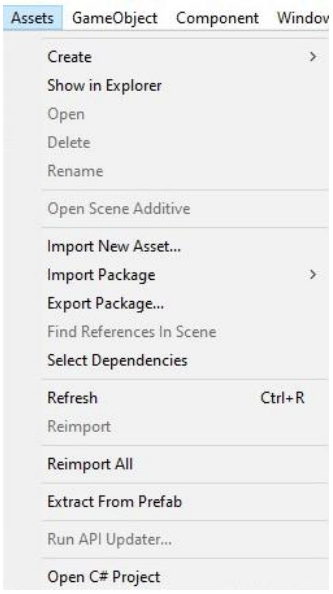

Ilustración 23: Pestaña Assets

Por último vamos a ver la pestaña GameObjects, en esta pestaña vamos a poder crear un nuevo objeto, también podemos crear objetos 3D o 2D, podemos crear objetos de cámara, de sonido, efectos… En el siguiente punto donde crearemos el terreno de nuestro entorno vamos a utilizar dicha pestaña.

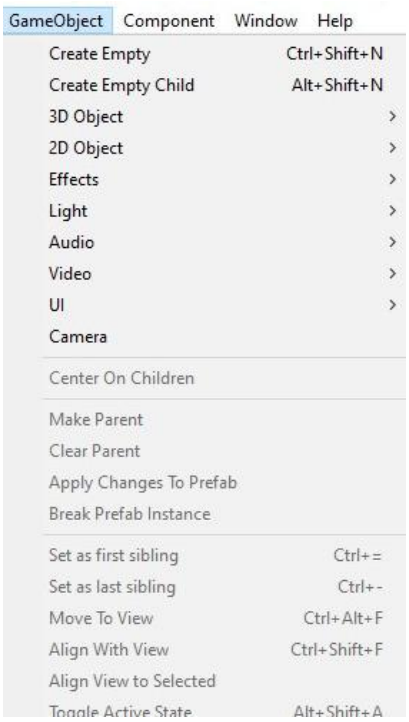

Ilustración 24: Pestaña GameObjects

3.1.4. Crear Terreno

A continuación vamos a ver cómo crear un terreno en Unity. Un terreno es la parte fundamental de todo proyecto que vayamos a crear, ya que un terreno es la superficie donde vamos a crear nuestro entorno y en la cual vamos a tener que situar todos elementos que necesitemos para que se convierta en la escena deseada.

Para crear un terreno tenemos que entrar en la pestaña desplegable de GameObjects y clicar sobre el desplegable 3D Objects, aquí se nos abrirá un nuevo desplegable con los objetos 3D que podemos crear y seleccionamos la opción de Terrain.

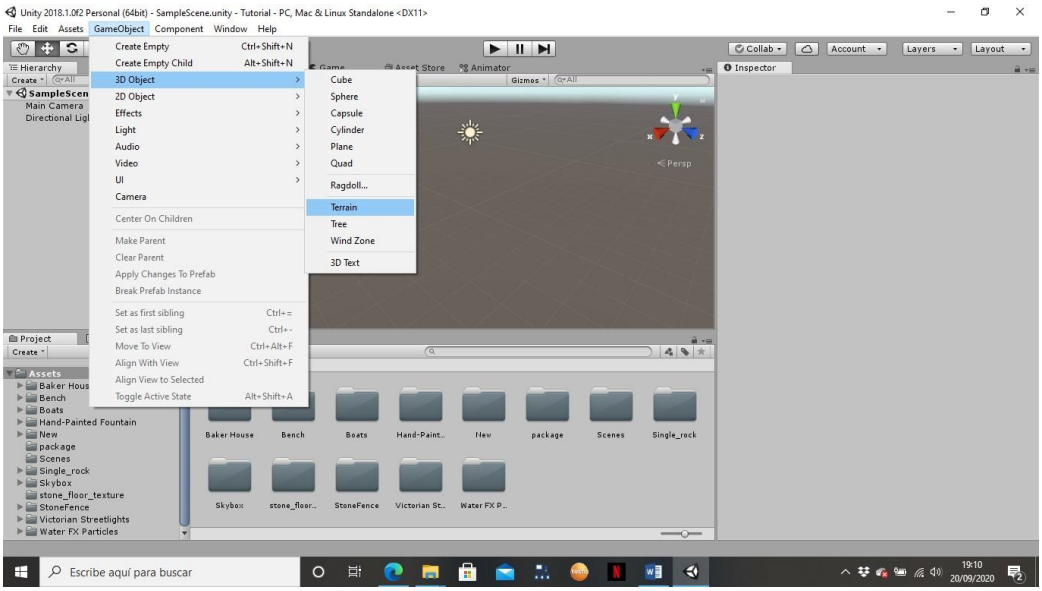

Ilustración 25: Crear terreno

Una vez realizado esto veremos cómo se sitúa en nuestra escena el terreno vacío que hemos creado, en este terreno como veremos en los pasos siguientes vamos añadir texturas, montañas, árboles y todo lo que haga falta para crear la escena. Sin embargo ahora mismo solo veremos un terreno blanco sin nada.

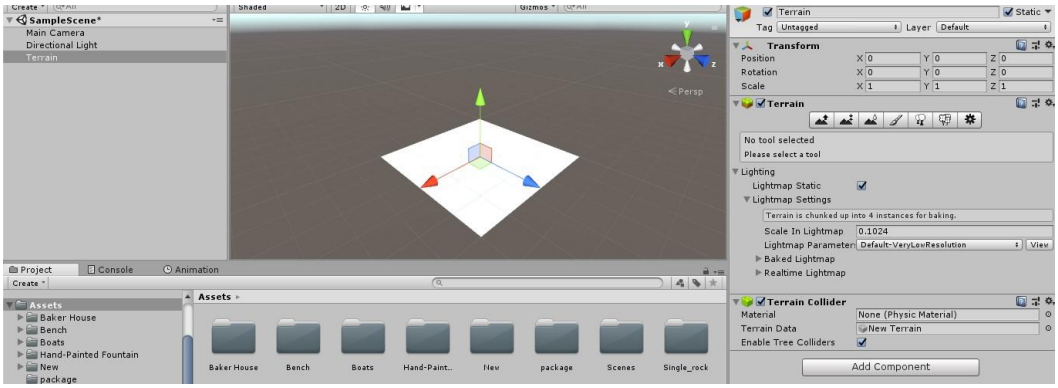

Ilustración 26: Terreno vacío

Una vez añadido el terreno en el inspector podemos observar la configuración de varios parámetros de nuestro terreno. En la siguiente ilustración podemos observar dicho inspector en el cual podemos ver el menú Transform donde podemos ajustar la escala posición y rotación de nuestro terreno y podemos ver el menú Terrain donde vamos a

poder añadir texturas a nuestro terreno, añadir montañas, árboles… En esta última parte va ser donde nos centraremos en la siguiente sección ya que vamos añadir texturas a nuestro terreno y vamos a crear las montañas de nuestro terreno.

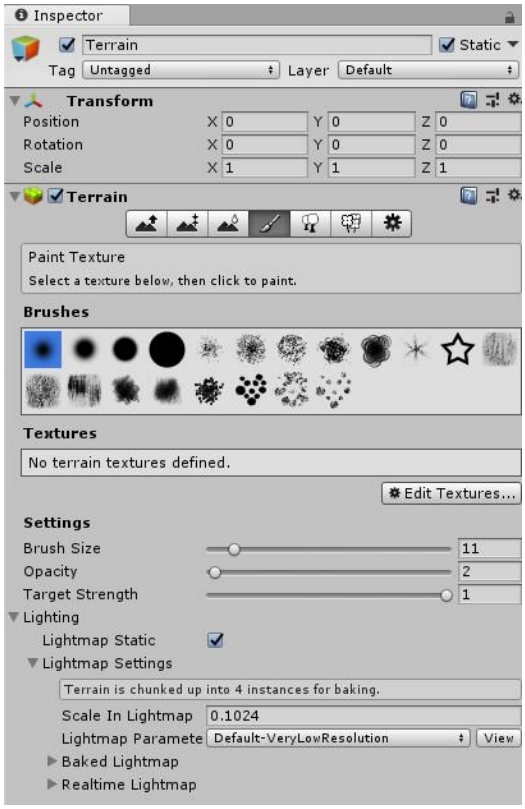

Ilustración 27: Inspector del terreno creado

3.1.5. Añadir Texturas y Montañas

Como hemos dicho anteriormente, nos vamos a centrar en el menú Terrain del inspector de nuestro nuevo terreno. A continuación veremos marcado en colores lo que hace cada parte de este menú. Antes de nada tenemos que descargar un paquete muy importante a la hora de realizar una escena ya que lleva muchas texturas esenciales además de otros assets que nos pueden ayudar mucho. Debemos descargar el paquete Enviroment. Para ello vamos a los menú Assets de la barra de tareas hacemos click sobre Import Package y seleccionamos Enviroment. Assets->Import Package->Enviroment.

En rojo podemos ver la parte que sirve para crear montañas, igualar el terreno con una misma altura, rebajar la altura de las montañas, en esta parte también podemos añadir árboles u otras plantas como veremos más adelante. En amarillo tenemos los punteros que podemos seleccionar para pintar nuevas texturas, montañas o árboles sobre el terreno, según vayamos aprendiendo veremos cuando podemos dar uso a unos u otros. Por último en verde podemos ver la parte de Edit Textures en la cual vamos a poder añadir texturas a nuestro terreno que será lo que veamos a continuación. Abajo podemos observar las configuraciones, (settings), en la cual podemos ajustar el grosor de los punteros, la densidad con la que queremos que pinten…

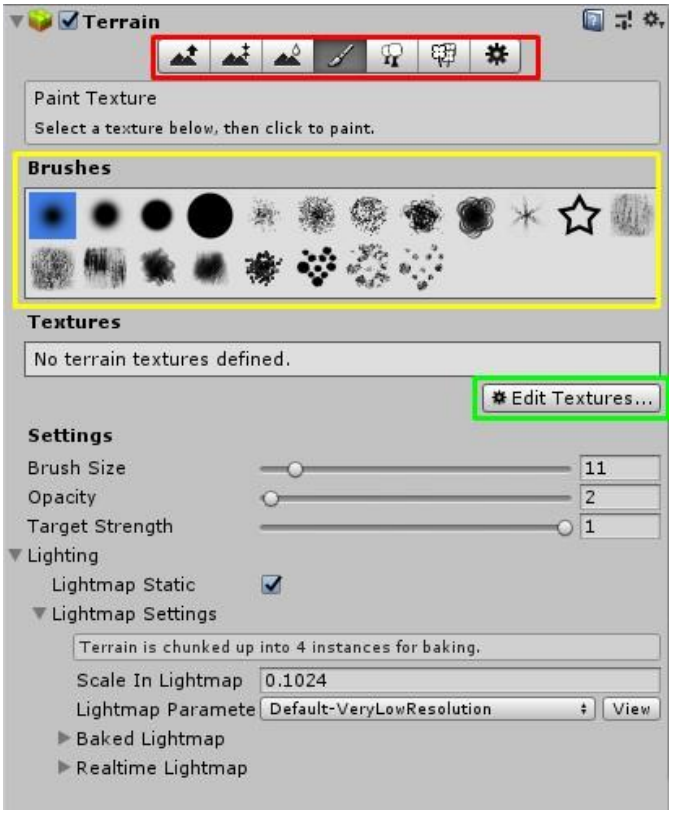

Ilustración 28: Menú Terrain del inspector

Ahora vamos añadir una textura a nuestro terreno, si pinchamos sobre el menú Edit Textures, y le damos a Add Texture se nos abre el menú Add Terrain Texture en el cual vamos a seleccionar la textura que nos gustaría que tuviera nuestro terreno. Pulsamos sobre el botón Select del primer recuadro que nos aparece.

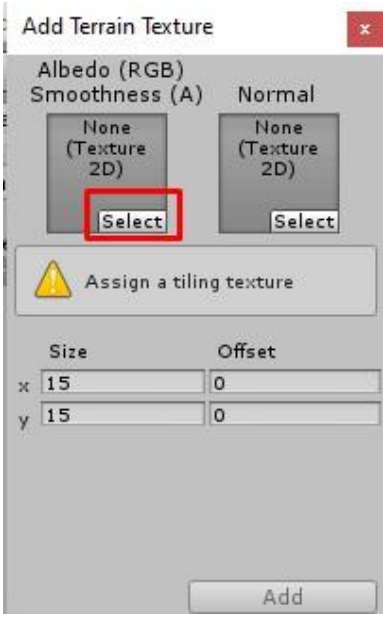

Ilustración 29: Menú Add Terrain Texture

Se nos abre a continuación el menú Select Texture2D en el cual vamos a elegir de entre las diversas texturas que nos aparecen las que queremos para nuestro terreno. En este caso vamos a elegir la textura Hand-Paint que es como un tipo de hierba que puede quedar muy bien en nuestra escena. Para escogerla hacemos doble click sobre la textura que queremos coger. Y volveremos al menú Add Terrain Texture donde nos aparece la textura seleccionada y hacemos click en Add.

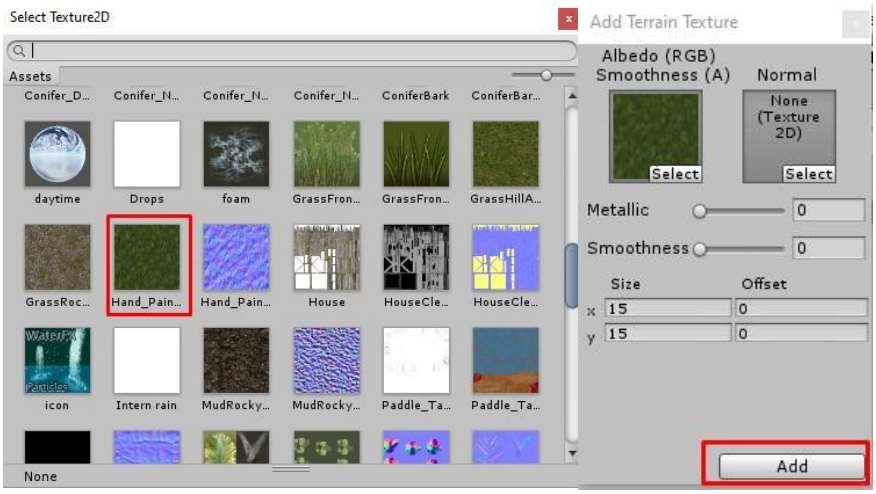

Ilustración 30: Añadir textura Hand-Paint

Una vez añadida la textura nuestro terreno que anteriormente era totalmente blanco, nos aparece con la nueva textura, como si estuviera todo lleno de esta hierba que hemos añadido.

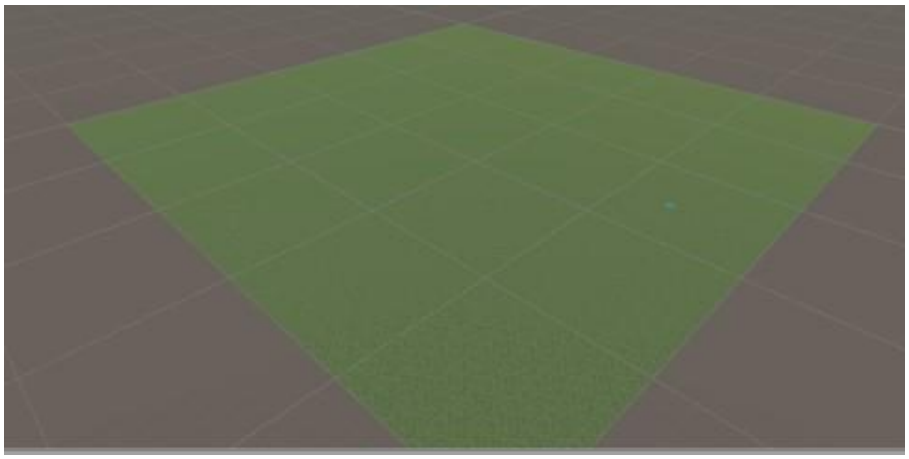

Ilustración 31: Terreno con la textura añadida

La idea principal de nuestra escena es realizar un paisaje rodeado de montañas, en el cual podamos añadir un lago central para ver también como se añade agua a nuestro terreno. En cualquier escena en la que queramos añadir un lago o algún lugar parecido en el que tengamos que hacer uso de agua lo primero que debemos hacer es adaptar nuestro terreno para poder añadir el agua.

Para adaptar el terreno a un lugar donde queremos añadir agua, debemos de igualar toda la superficie excepto el lugar donde vayamos a añadir el agua. Por así decirlo es

como dejar un hoyo en el terreno. Para realizar este hoyo debemos de seleccionar el terreno y en el inspector en el menú Terrain seleccionar la opción de igualar terreno para elevar nuestro terreno y ponerlo todo a la misma altura. A continuación vemos como debemos de configurar el menú Terrain para conseguir nuestro propósito.

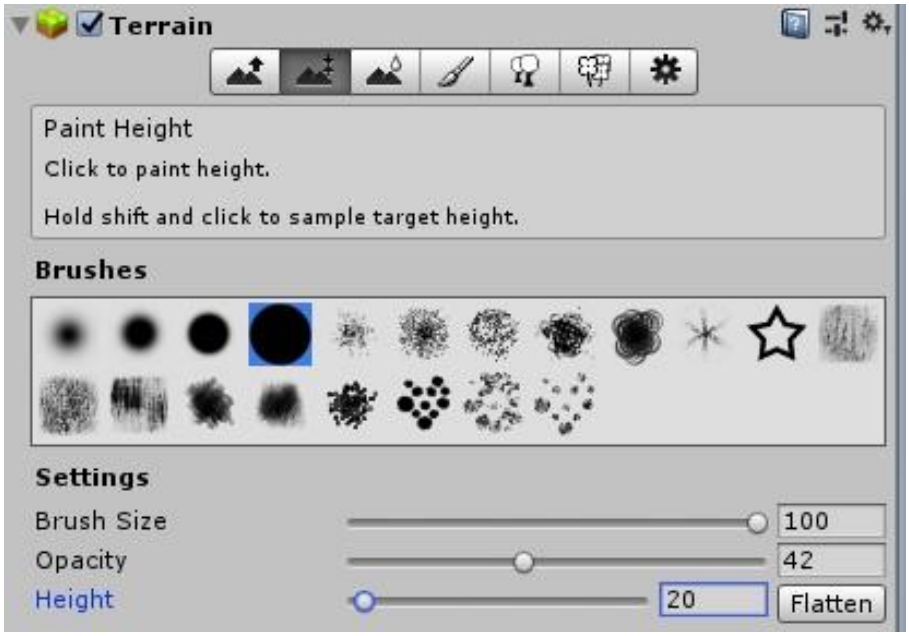

Ilustración 32: Igualar terreno

Una vez configurado nuestro inspector simplemente debemos de ir dibujándolo sobre nuestro terreno manteniendo pulsado el botón izquierdo del ratón por toda la superficie que queremos que suba y dejándolo sin pasar por la superficie donde va a ir nuestro lago. Cuando terminemos nos tiene que quedar algo parecido a la siguiente ilustración.

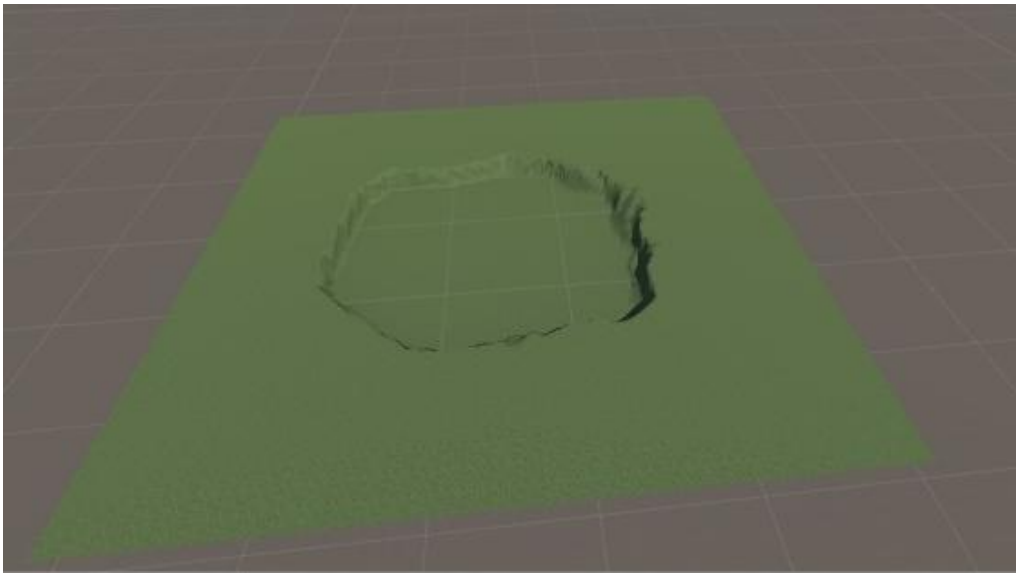

Ilustración 33: Terreno preparado para insertar un lago

A continuación vamos a añadir unas montañas para que nuestra escena sea como hemos dicho anteriormente un lago rodeado de montañas. Para añadir las montañas volvemos al menú Terrain del inspector de nuestro terreno. Aquí seleccionamos la opción de crear montañas, cogeremos un puntero más pequeño ya que ahora no queremos pintar en todo el terreno montañas sino solo en los bordes y también ajustamos la configuración tal y como vemos en la siguiente ilustración.

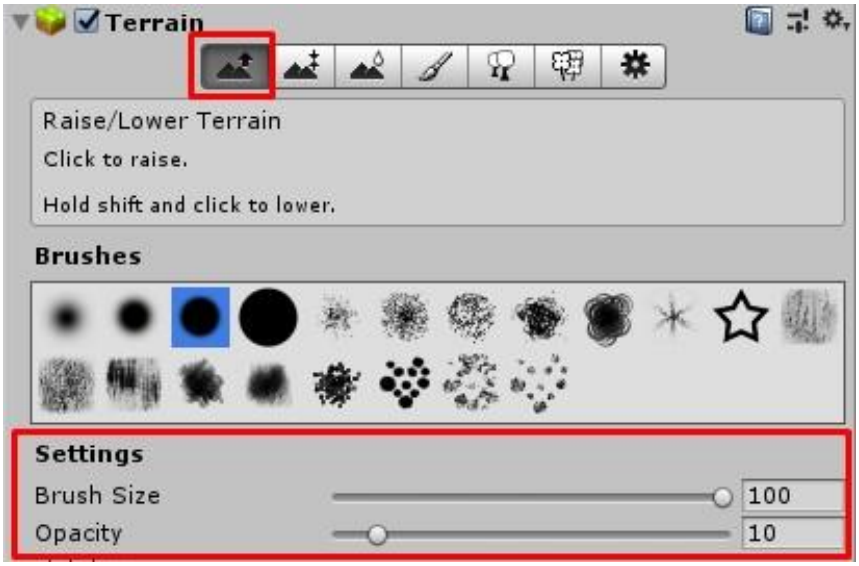

Ilustración 34: Crear montañas

Para crear montañas una vez tenemos configurado el menú también las vamos a ir creando con el botón izquierdo de nuestro ratón. En este caso cuanto más tiempo tengamos el click pulsado sobre un mismo lugar más alta será la montaña por eso una vez se considere que se ha llegado a la altura deseada soltamos el click. Si queremos hacer varias montañas como es nuestro caso pues vamos con el botón izquierdo pulsado por donde queremos crearlas y teniendo en cuenta la altura. Finalmente nos tiene que quedar algo parecido a la siguiente ilustración.

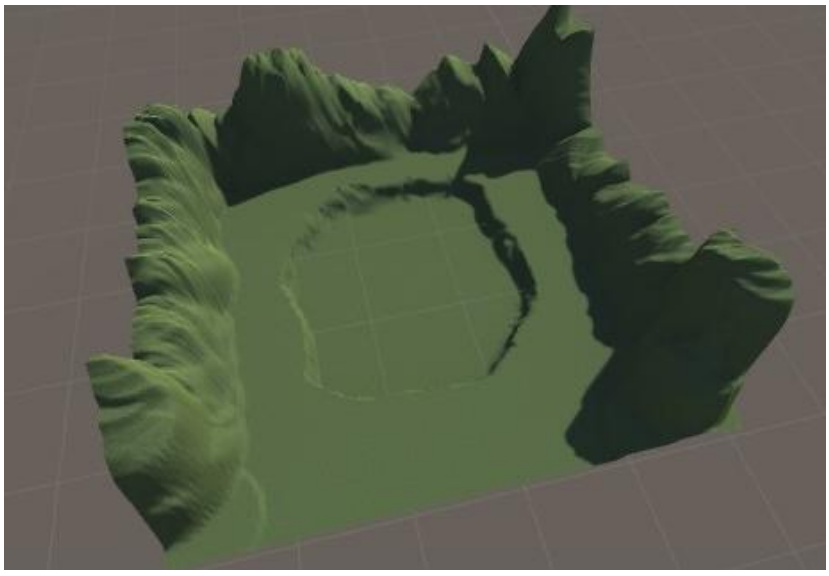

Ilustración 35: Terreno con montañas

Una vez creadas las montañas vamos a ver cómo podemos redondear las cimas y el borde del lago para que no parezca algo muy abrupto ya que las montañas normalmente son más redondeadas en la cima al igual que los bordes de los lagos. Para ello seleccionando el terreno vamos al menú Terrain de su inspector. Seleccionamos la opción de redondear y rebajar la altura de las montañas y configuramos el menú tal y como vemos a continuación.

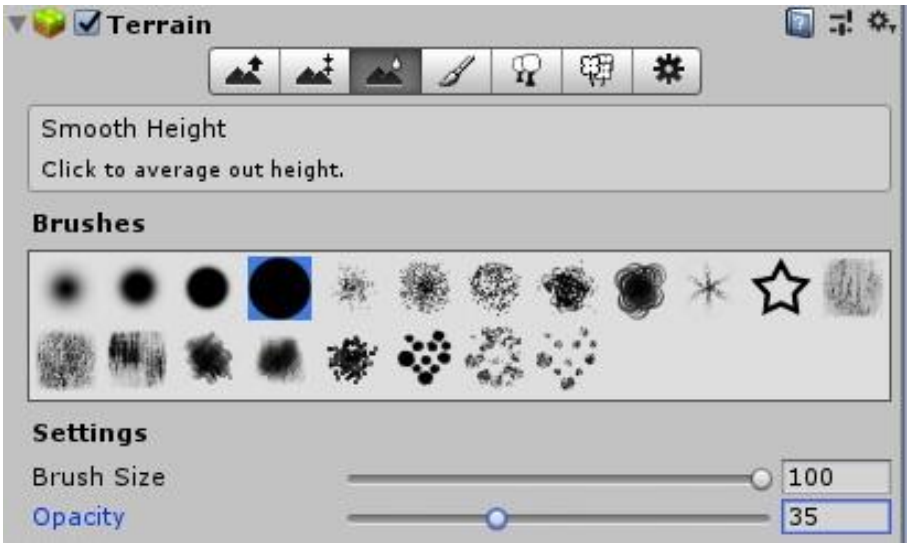

Ilustración 36: Redondear picos de montaña

Para aplicar este punto es igual que los anteriores con el click izquierdo de nuestro ratón vamos pulsando sobre todo aquello que queremos redondear. Vamos pulsando sobre los picos de las montañas y alrededor del borde donde irá nuestro lago. Esta parece una modificación que no influye mucho en nuestro terreno pero cuando se quiere hacer una escena en realidad virtual queda mucho más bonito ver las cimas redondeadas que ver una cima muy abrupta.

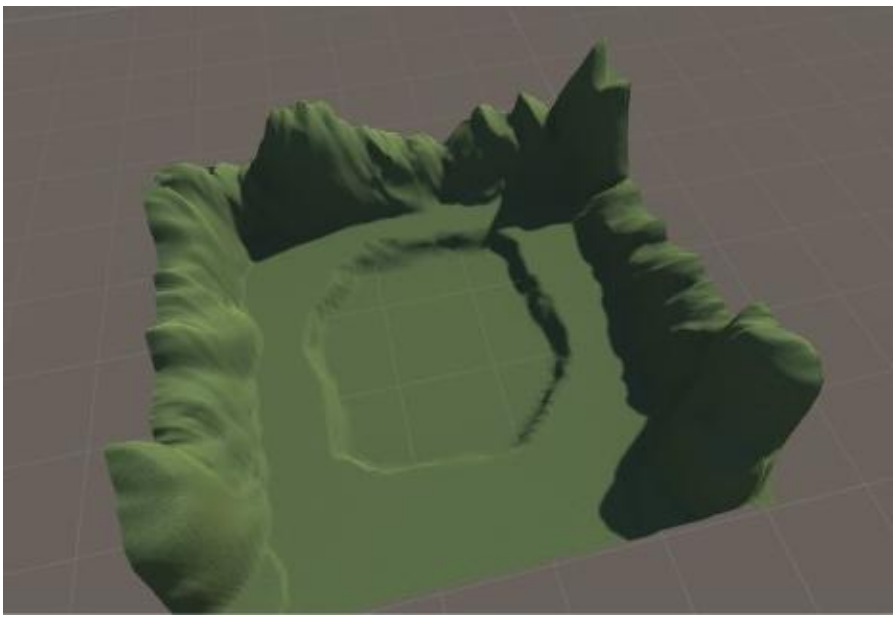

Ilustración 37: Terreno con las cimas redondeadas

Como podemos ver ahora mismo se ve todo muy verde en nuestro terreno debido al a textura de hierba que hemos usado. Ahora por último en este punto vamos a ver cómo podemos añadir otra textura diferente para las montañas porque queremos que las montañas tengan otra textura diferente a la del suelo de nuestro terreno.

Para añadir otra textura vamos al inspector de nuestro terreno y pulsamos en el menú Terrain sobre la opción Edit Textures tal y como hemos hecho para añadir la primera textura y después pulsamos sobre Add Texture. Una vez hecho esto es igual que anteriormente pulsamos sobre Select y seleccionamos la textura que queremos añadir y hacemos click en Add. En este caso como queremos añadir textura a las montañas queremos una textura más rocosa por eso elegimos la textura GrassRock que es como de roca con hierba incrustada.

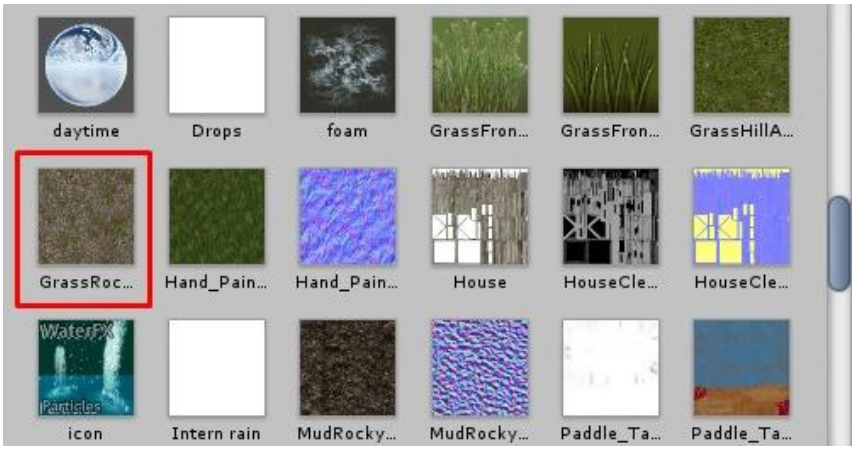

Ilustración 38: Textura para las montañas

Una vez hecho esto no apreciaremos ningún cambio en nuestro terreno, lo único que veremos es que se nos ha añadido la textura en la sección Textures del menú Terrain. Para aplicar esta textura nueva lo que haremos será seleccionar el pincel que nos aparece y elegir el tipo de puntero deseando y con el click izquierdo de nuestro ratón iremos pintando dicha textura por donde queramos, en este caso por las montañas.

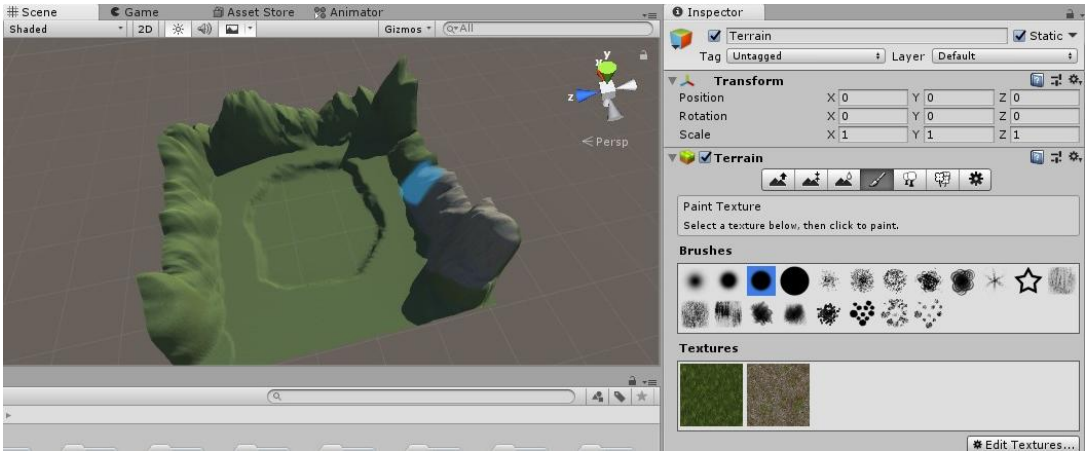

Ilustración 39: Pintar textura en las montañas

3.1.6. Añadir Skybox y Primera Persona

En este punto vamos a ver como añadir dos Assets que son fundamentales a la hora de desarrollar una escena. El Skybox básicamente es añadir como un fondo a nuestra escena en esta caso vamos añadir el cielo ya que ahora mismo no se ve nada si miramos arriba dentro de nuestra escena. También vamos añadir una cámara en primera persona para que podamos desplazarnos pos nuestra escena como si fuéramos los protagonistas de la aplicación que estamos creando.

Vamos a ver como añadir el Skybox de nuestra escena. En nuestro entorno de trabajo tenemos abajo la carpeta Assets donde tenemos todos los Assets que tenemos importados en este proyecto, a mí me aparecen más porque tengo otros descargados de otras escenas que he ido creando.

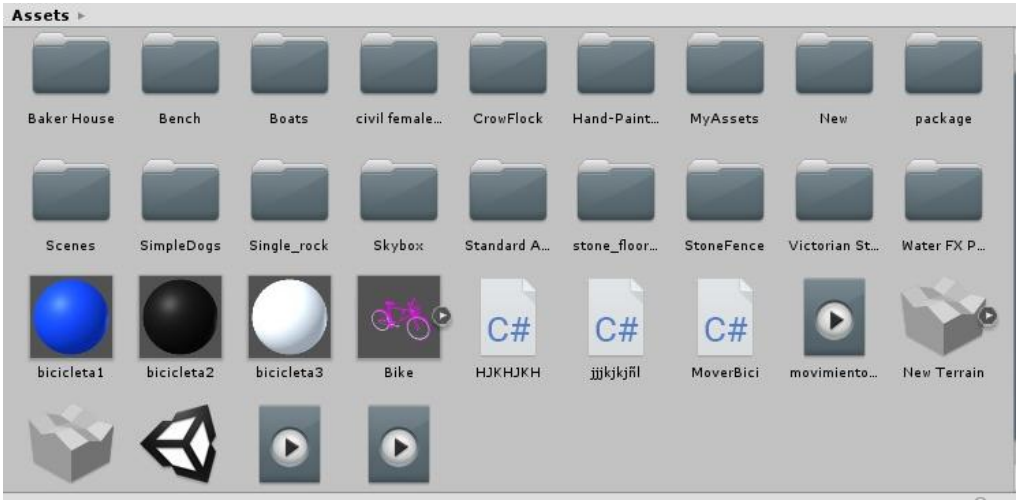

Ilustración 40: Carpeta Assets

En este caso a nosotros nos hace falta seleccionar la carpeta Skybox, abrimos esta carpeta y nos encontramos otras tres carpetas aunque a nosotros de momento solo nos interesa la carpeta donde pone Materials.

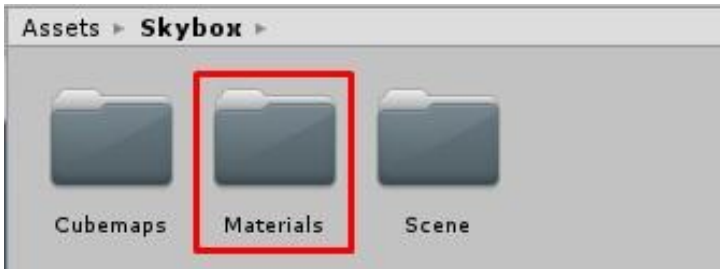

Ilustración 41: Carpeta Skybox

Si pinchamos sobre la carpeta Materials se nos abrirán dos modelos de Skybox, Skybox_Day y Skybox_Sunset. La diferencia entre cada uno es que en el primero será un cielo de día soleado y en el segundo será el cielo de un atardecer. Nosotros para nuestra escena vamos a elegir el primero ya que dará más luminosidad a nuestro paisaje. Para añadirlo a nuestra escena simplemente tenemos que hacer click sobre el que queramos y arrastrarlo al fondo de nuestra escena.

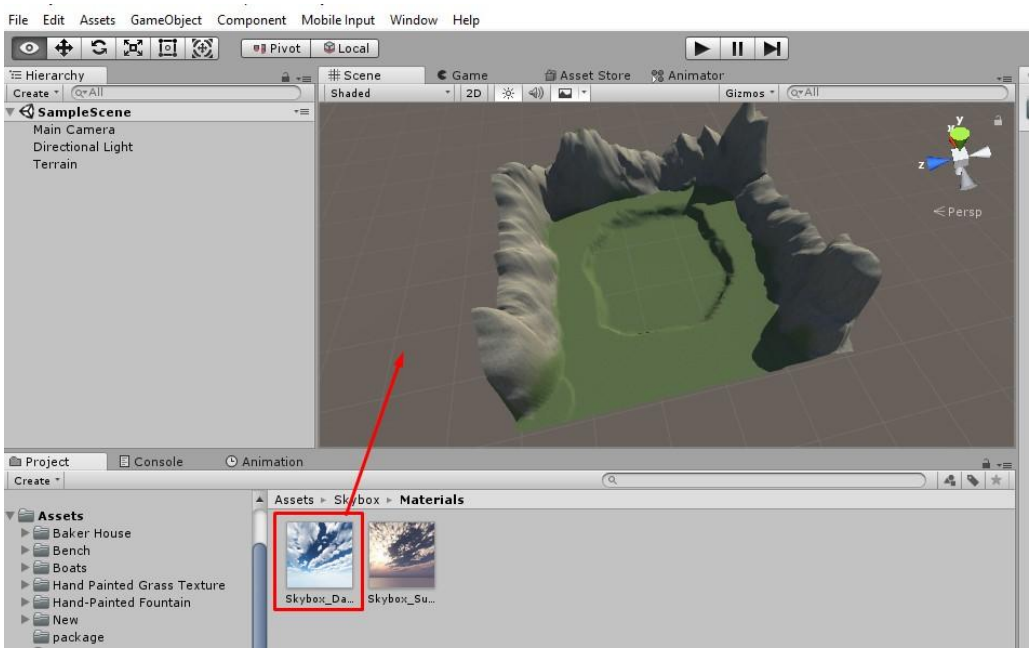

Ilustración 42: Arrastras Skybox

Una vez realizado podremos ver como el fondo se nos ha puesto de un color azul cielo sustituyendo al gris inicial.

Ahora vamos a insertar una cámara en primera persona, lo primero que tenemos que hacer es eliminar la cámara principal que hay ahora mismo. Para eso vamos al menú Hierarchy de la izquierda y hacemos click derecho sobre Main Camera y le damos a eliminar.

Para añadir la cámara en primera persona tenemos que importar un paquete de Assets igual que hemos importado anteriormente el paquete Enviroment. Ahora necesitamos el paquete Characters. Assets->Import Package->Characters. Una vez realizada la importación ya podemos añadir nuestra cámara en primera persona.

Vamos a la carpeta Assets, después hacemos click en la carpeta Standard Assets, abrimos la carpeta FirstPersonCharacter y por último la carpeta Prefabs aquí ya tendremos nuestra cámara en primera persona llamada FPSControls. Assets->Standard Assets->Characters->FirstPersonCharacter->Prefabs. Para añadirla seleccionamos la cámara FPSControls y la arrastramos a nuestro terreno donde queramos que aparezca cuando se inicie la aplicación, en este caso yo la pongo al principio.

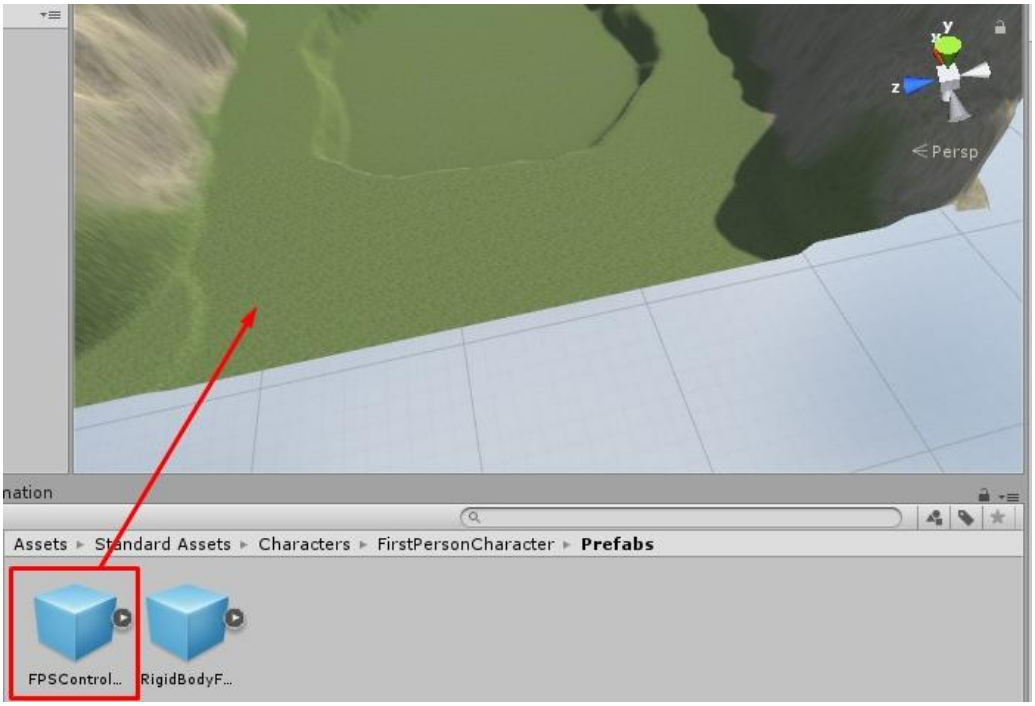

Ilustración 43: Añadir cámara en primera persona

A continuación vamos a ver como se observa nuestra escena hasta ahora como si estuviéramos reproduciendo la aplicación. Vemos como se ve ya la cámara en primera persona, se pueden observar las distintas texturas que hemos añadido y también se puede ver el cielo con el Skybox que hemos añadido.

Ilustración 44: Escena vista en primera persona

Podemos ver como nuestra escena va cogiendo forma, aunque aún nos queda añadir el agua al lago o añadir árboles lo cual veremos en los siguientes puntos.

3.1.7. Añadir Agua y Árboles

Por último en para terminar nuestra escena vamos añadir el agua al lago y terminaremos añadiendo unos árboles para aportar más realidad al paisaje.

Para añadir del agua tenemos que seguir los mismos pasos que anteriormente para añadir la cámara en primera persona. Vamos a la carpeta Assets, dentro de esta abrimos la carpeta Standard Assets, en este caso abrimos la carpeta Enviroment, después abrimos la carpeta Water (Basic) y por último la carpeta Prefabs. En esta última carpeta solo tendremos que arrastrar el modelo de agua deseado hasta el lugar donde queremos situar el lago. Assets->Standard Assets->Enviroment->Water (Basic)->Prefabs.

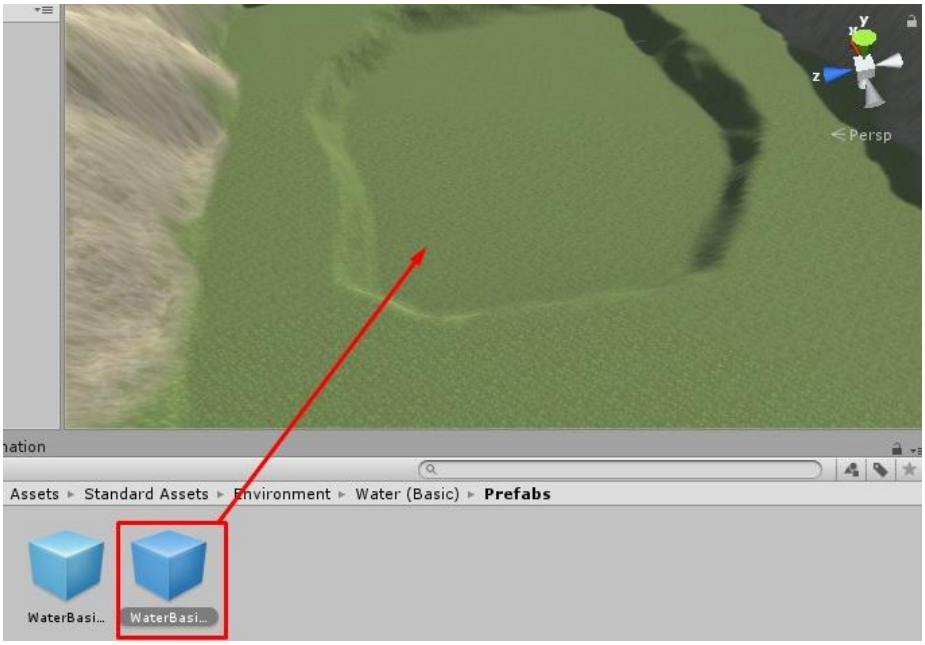

Ilustración 45: Añadir agua al lago

Una vez añadida el agua en el hueco del lago no veremos nada ya que debemos ajustar la escala del agua para que sea visible y se sitúe en el lugar adecuado. Para ajustar el agua debemos de seleccionar el agua que hemos añadido y en el inspector ajustar los ejes X, Y, Z para que se cree el agua del tamaño adecuado. En mi caso lo ajusto según la siguiente ilustración pero dependerá del tamaño del hueco donde queremos situar el agua.

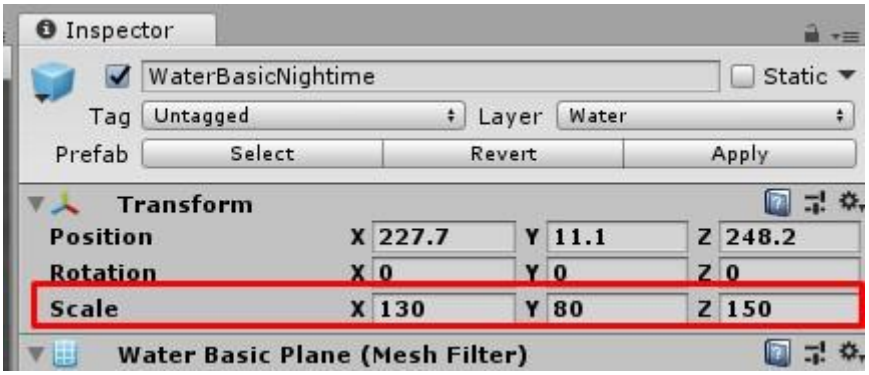

Ilustración 46: Ajustar la escala del agua

Por último vamos a ver como añadir árboles a nuestra escena y así aportar más realidad al paisaje. Para añadir árboles debemos de seleccionar el terreno y en el menú Terrain del inspector vamos a la parte de añadir árboles. Ahora pulsamos el botón Edit Trees y después Add Tree. En la configuración podemos elegir la densidad de árboles que se va a dibujar por cada click o el grosor del puntero con el que se dibujan los árboles.

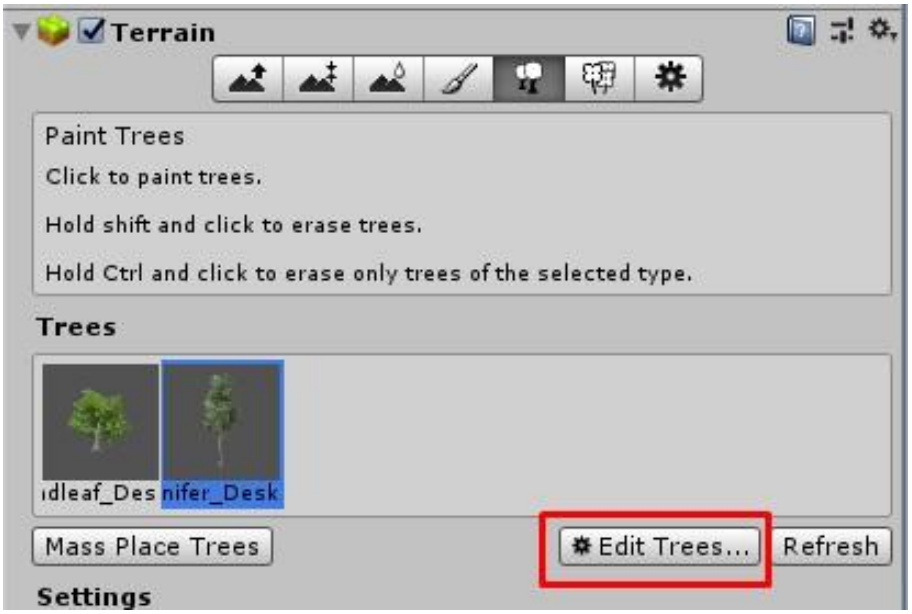

Ilustración 47: Añadir árboles

Ahora se nos abre la ventana Add Tree en la cual debemos pulsar sobre el círculo que aparece marcado en la siguiente ilustración.

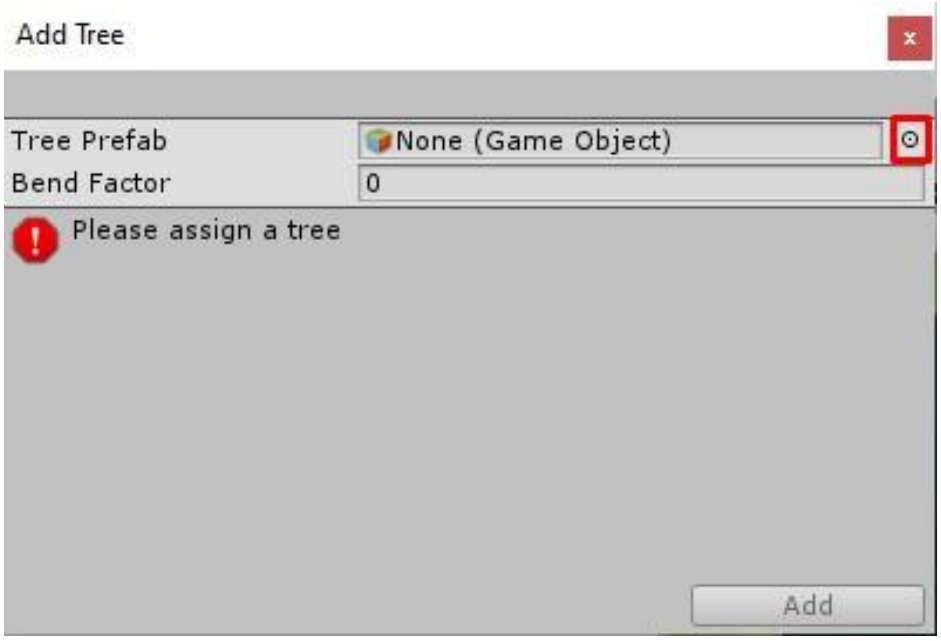

Ilustración 48: Menú Add Tree

Ahora se abrirá el menú Select GameObject en el cual vamos a poder elegir el tipo de árbol que queremos añadir para dibujar, aunque también se podría elegir otro tipo de Asset que tuviéramos descargado para añadirlo a nuestra escena. En mi escena he añadido los tres tipos de árboles que vemos marcados en la ilustración, para añadir cada uno de los tipos hay que hacer el mismo procedimiento para cada tipo. Para seleccionar el tipo de árbol que queremos añadir hacemos doble click y volveremos al menú Add Tree.

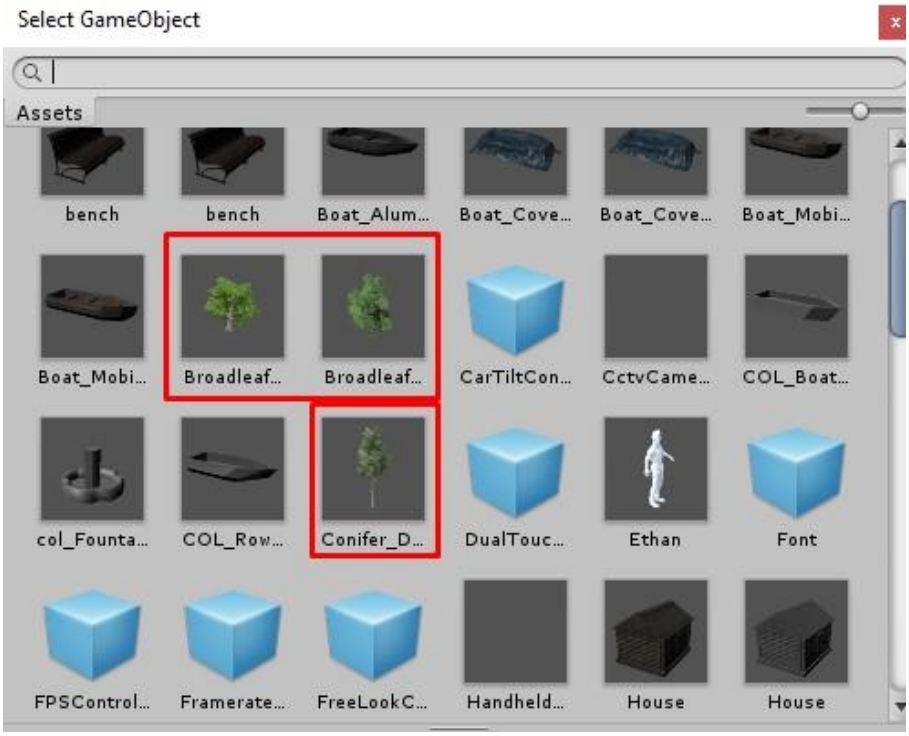

Ilustración 49: Menú Select GameObject

Una vez volvemos al menú Add Tree pulsamos sobre Add para que nos aparezca ya en el menú Terrain de nuestro terreno.

Ilustración 50: Botón Add

Una vez tenemos los árboles en el menú Terrain solamente tenemos que seleccionar el que queremos pintar y con el click izquierdo del ratón ir distribuyéndolos por las zonas de nuestro paisaje por donde queramos añadir árboles.

Ilustración 51: Pintar árboles

Cuando tengamos todos los árboles dibujados ya habremos terminado nuestro primer paisaje, si pulsamos sobre el botón Play podremos ver cómo quedaría nuestra escena en la aplicación. Podemos avanzar por nuestra escena con las flechas del teclado y así ir paseando por nuestro paisaje. A continuación en la siguiente ilustración vemos cómo quedaría nuestra escena en la aplicación con la cámara en primera persona.

Ilustración 52: Vista escena primera persona

3.2. Simulador de Bicicleta Virtual en Unity 5

En esta parte vamos a ver como se ha montado el entorno Unity para el simulador de bicicleta virtual en el cual consiste el trabajo. Vamos a ir viendo cada uno de los menús como se han configurado para conseguir el objetivo final. El entorno Unity creado para el simulador es muy parecido al creado en la práctica anterior, en este caso hemos querido elaborar un parque por el cual puedas dar un paseo en bicicleta de manera virtual, para este entorno se han creado Assets descargados de la Assets Store, además también se han configurado script para algunos de los Assets, se ha añadido sonido para que parezca más real y también se ha añadido viento que aporta más movilidad a la escena.

3.2.1. Animator y Animation

A diferencia de la escena que hemos creado anteriormente para la práctica, en la escena de este trabajo hemos añadido Assets descargados de la Assets store. Por ejemplo hemos descargado un paquete de Assets en el cual se encuentran una gran variedad de razas de perros, también hemos descargado el Asset de un humano que podría estar dando un paseo por el parque mientras nosotros vamos en nuestra bicicleta.

Una vez tenemos estos Assets descargados necesitamos darles una movilidad para que vayan paseando por el parque cuando ejecutamos la escena. Esta es una de las partes que más me ha costado en el trabajo ya que he tenido que leer mucha documentación y ver varios tutoriales para poder crear un Animator y un Animation para la persona y el perro. Antes de nada explicaremos que es un Animator y un Animation, un Animator es un diagrama de estados que tiene el Asset en el cual podemos hacer transiciones de un estado a otro y en el cual vamos a relacionar los Animations que tenga asignados este Asset, un Animation es directamente el movimiento grabado que tiene para una acción el Asset como por ejemplo podría ser andar.

En los Assets que hemos descargado tanto para los perros como para la persona, tenemos creados también distintos Animations que serán los que utilicemos para dar movimiento a los Assets. Pues bien una vez colocados lo Assets que queremos utilizar sobre nuestro terreno vamos a ver cómo darles movimiento.

Lo primero que vamos a ver es como hacerlo para el Asset de la persona, lo primero que tenemos que hacer es insertar este Asset en nuestra escena, por lo tanto lo arrastramos desde la carpeta donde está a la escena.

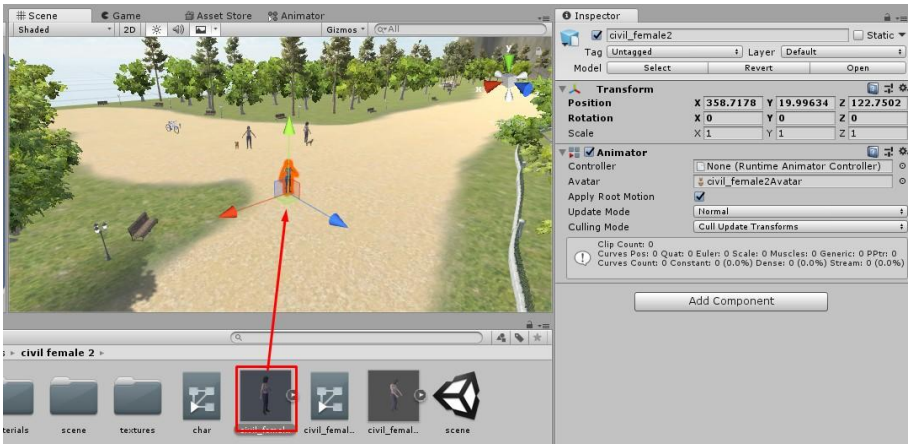

Ilustración 53: Colocar Asset persona

Como podemos observar en el menú inspector de la ilustración anterior, este Asset ya lleva su Animator creado, ahora lo vamos a configurar. Tenemos que asignar un Controller a este Animator, para ello pulsamos sobre el circulo que aparece a la derecha de la barra Controller y se nos abrirá la ventana Select RuntimeAnimatorController en la cual podremos elegir el Controller, en este caso será el civil_female2.

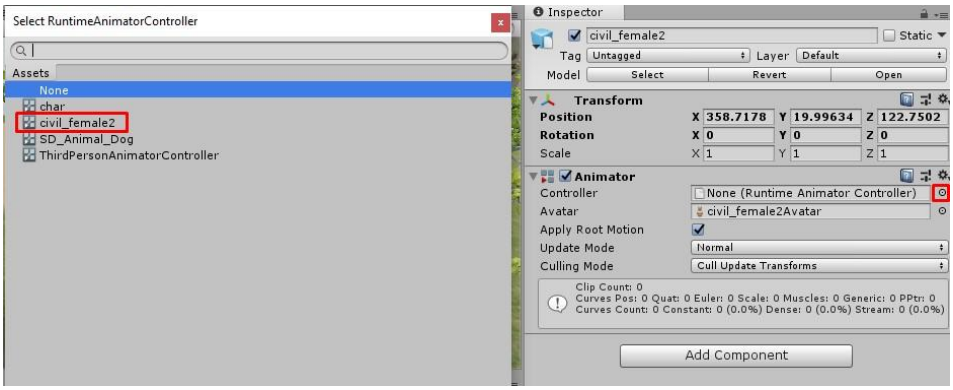

Ilustración 54: Agregar Controller

Para terminar de configurar el Animator tenemos que desmarcar la opción Apply Root Motion para que no mueva automáticamente el objeto, el parámetro Update Mode lo ponemos en Animate Physics y el parámetro Culling Mode lo ponemos en Always Animate con lo cual nos quedará el menú tal y como se observa en la siguiente ilustración.

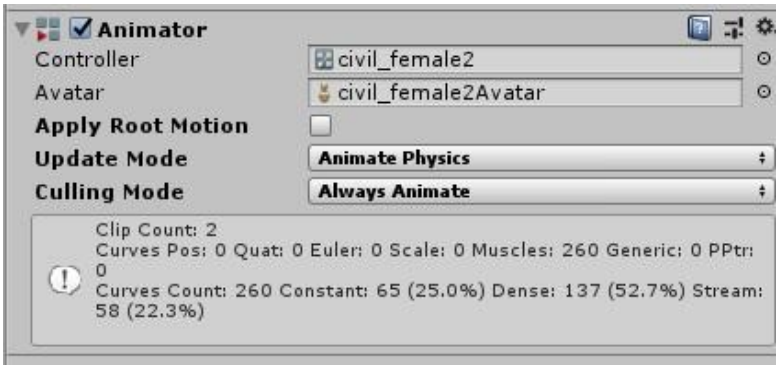

Ilustración 55: Menú Animator

Ahora tenemos que crear la Animation para este Assets, para ello dentro del menú Inspector del Asset vamos a pulsar sobre el botón Add Component y vamos a la opción Miscellaneous en la cual vamos a seleccionar Animation que es lo que queremos crear.

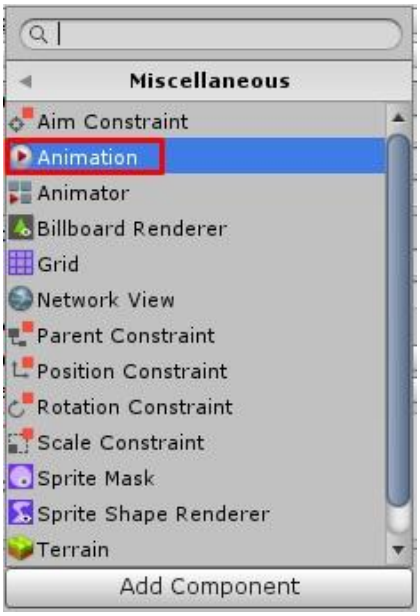

Ilustración 56: Añadir Animation

Una vez seleccionado se nos crea el menú Animation dentro del menú Inspector, lo siguiente que tenemos que hacer es configurar este nuevo menú. Lo primero que hacemos es añadir la Animation para lo cual debemos de pulsar sobre el círculo de la derecha de la línea de Animation y entonces se nos abrirá la ventana Select AnimationClip, en este caso nosotros elegimos la opción Walk ya que queremos que nuestro personaje tenga la acción de andar. Para terminar de configurar este menú marcamos la opción Animate Physics.

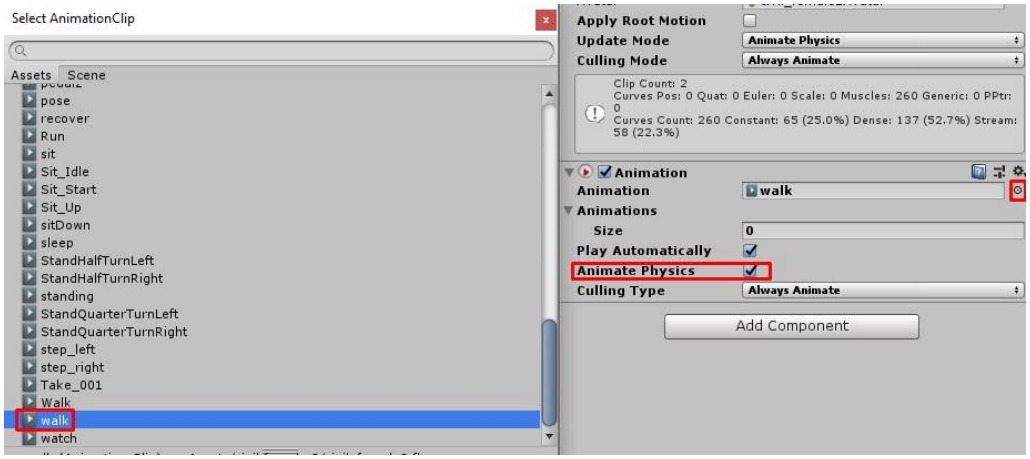

Ilustración 57: Menú Animation

Ahora ya tendríamos creados los menús de Animator y Animation para nuestra persona, ahora tenemos que crear su máquina de estados, para lo cual se utiliza el menú Animator que podemos ver junto a la Assets Store.

Pulsamos en el menú Animator y aquí crearemos nuestra máquina de estados, en el caso de nuestra persona solo vamos a utilizar el estado Walk y el estado Take_001. Para crear un estado es tan fácil como arrastrar la Animation que queremos usar, en este caso Walk y Take_001. Para crear las transiciones entre los distintos estados tenemos que hacer click derecho sobre el estado y le damos a Make Transition y pinchamos hasta el estado que vamos hacer esta transición.

Tenemos en este menú tres estados por defecto, los estados Any State y Exit que en este caso no utilizamos y el estado Entry que es cuando empieza la ejecución de la escena hacia qué estado debe transicionar, en este caso debe transicionar a Walk para que cuando se ejecute la escena directamente veamos a la persona andar. Por último he creado la variable booleana Caminar y la he marcado a true, para crear una variable debemos pulsar el botón de más que aparece en el cuadro indicado en la ilustración. Esta variable se utiliza como veremos un poco más adelante para la transición de un estado a otro.

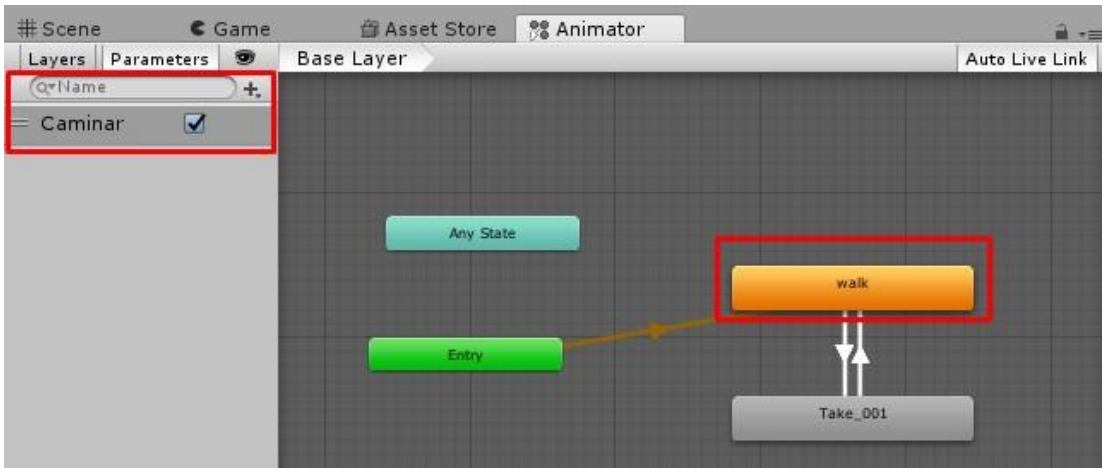

Ilustración 58: Animator Persona

A continuación vamos a ver el menú Inspector de la transición que va del estado Walk al estado Take_001, aquí podemos ver el diagrama de tiempos de la transición para ver cómo pasa de un estado al otro. Aquí debemos desmarcar el check de Has Exit Time, debajo del diagrama de tiempos podemos añadir condiciones para que la transición se ejecute, en este caso la condición que hará que pase del estado Walk al estado Take_001 es que la variable Caminar sea igual a false.

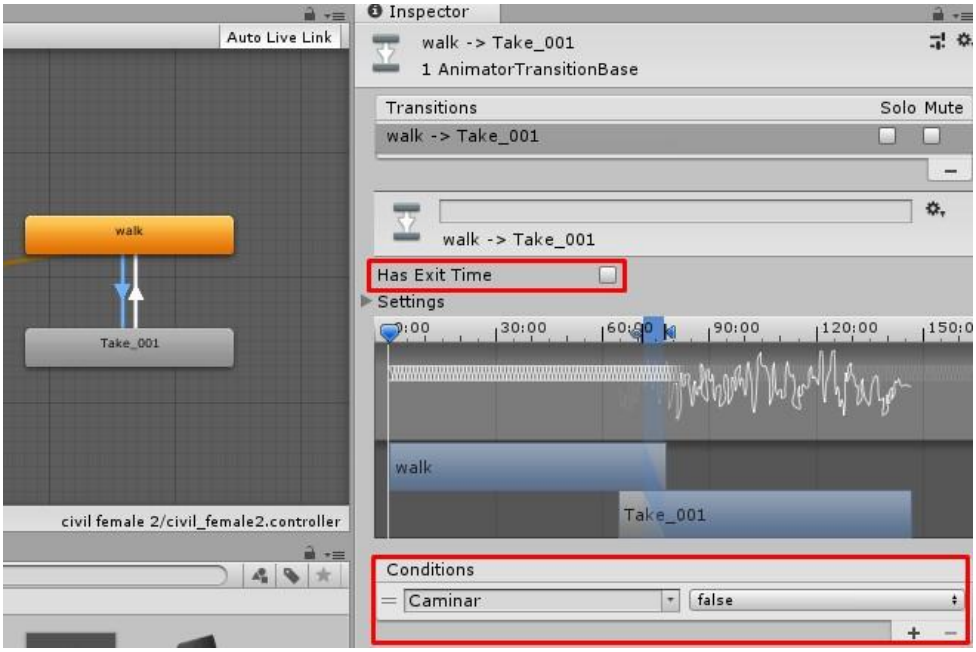

Ilustración 59: Transición Walk a Take_001

Por último para terminar de configurar nuestra persona debemos de asignarle un Script al Asset para indicar mediante código que debe desplazarse. Para añadir el Script le daremos a Add Component en el menú Inspector del Asset y seleccionaremos la opción Scripts y aquí seleccionaríamos el Script que queremos asignar. Si no tenemos ningún Script creado tendremos que darle a New Script y crear el Script para nuestro Asset.

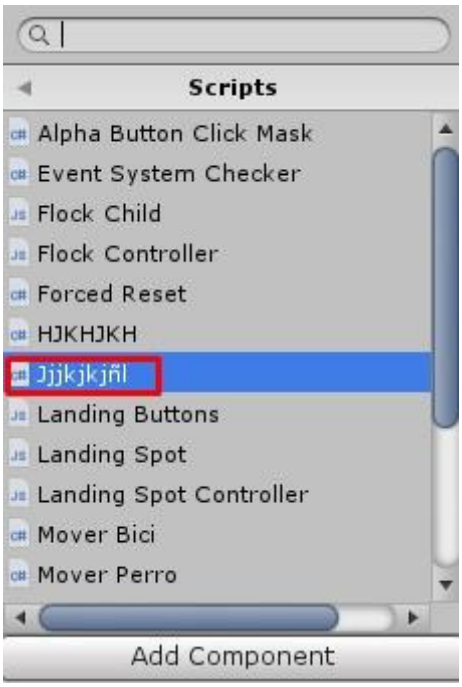

Ilustración 60: Añadir Script Persona

En el Script de nuestro Asset persona simplemente vamos a tener una clase dentro de la cual tendremos la función Start y la función Update. Antes de declarar la clase tenemos que insertar las bibliotecas que necesitamos para realizar nuestro código. Una vez dentro de la clase que tiene que ser pública, declaramos las variables que necesitamos, la variable animator del tipo Animator es para declarar nuestro Animator al igual que la variable myTransform del tipo Transform es para tener el Transform de nuestro Asset. También declaro la variable pública velocidad de tipo float la cual vamos a poder modificar en Unity en el menú Inspector del Asset y que nos va a servir para decir a qué velocidad se debe de desplazar la persona.

Dentro de la función Start simplemente obtengo el Animator y el Transform en las variables anteriormente indicadas y por último en la función Update vamos a indicar el desplazamiento que debe de hacer la persona para ello llamamos a la función Translate que está en la librerías y creamos un objeto Vector3 el cual es utilizado en los Script de Unity para desplazar los Assets a través de los ejes XYZ. En la componente Z de este objeto metemos la variable velocidad que configuramos desde Unity y multiplicamos el objeto por el tiempo.

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
5   public class jjjkjkjñl : MonoBehaviour {
6
7       //private Animator aimator;
8       // Use this for initialization
9       Animator animator;
10      Transform myTransform;
11      public float velocidad;
12      void Start () {
13          animator = this.GetComponent<Animator> ();
14          myTransform = this.GetComponent <Transform>();
15          //animator = GetComponent<Animator>();
16      }
17
18      // Update is called once per frame
19      void Update () {
20
21          transform.Translate(new Vector3 (0,0,velocidad) * Time.deltaTime) ;
```

Ilustración 61: Script Asset persona

A continuación podemos ver cómo quedaría el Script en el menú Inspector y cómo podemos modificar la variable velocidad como queramos.

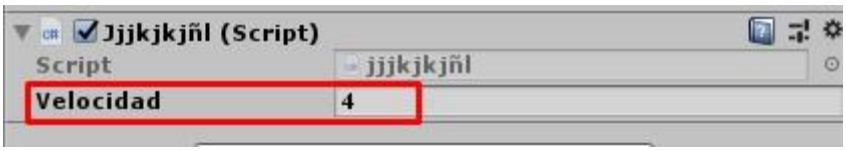

Ilustración 62: Variable Velocidad

Ahora vamos a ver como lo hemos hecho en este caso para el Asset del perro, los primeros pasos son iguales debemos de asignar el Animator, en este caso SD_Animal_Dog y las demás configuraciones tanto como para Animator como Animation serían las mismas, en este caso el Animation Walk que hemos añadido aunque se llame igual no es el mismo que para la persona, ya que en este caso es andar adaptado al perro. Por último también hemos añadido el Script MoverPerro en el cual definimos el comportamiento de este Asset, como podemos ver también tenemos una variable llamada Velocity para modificar la velocidad del perro.

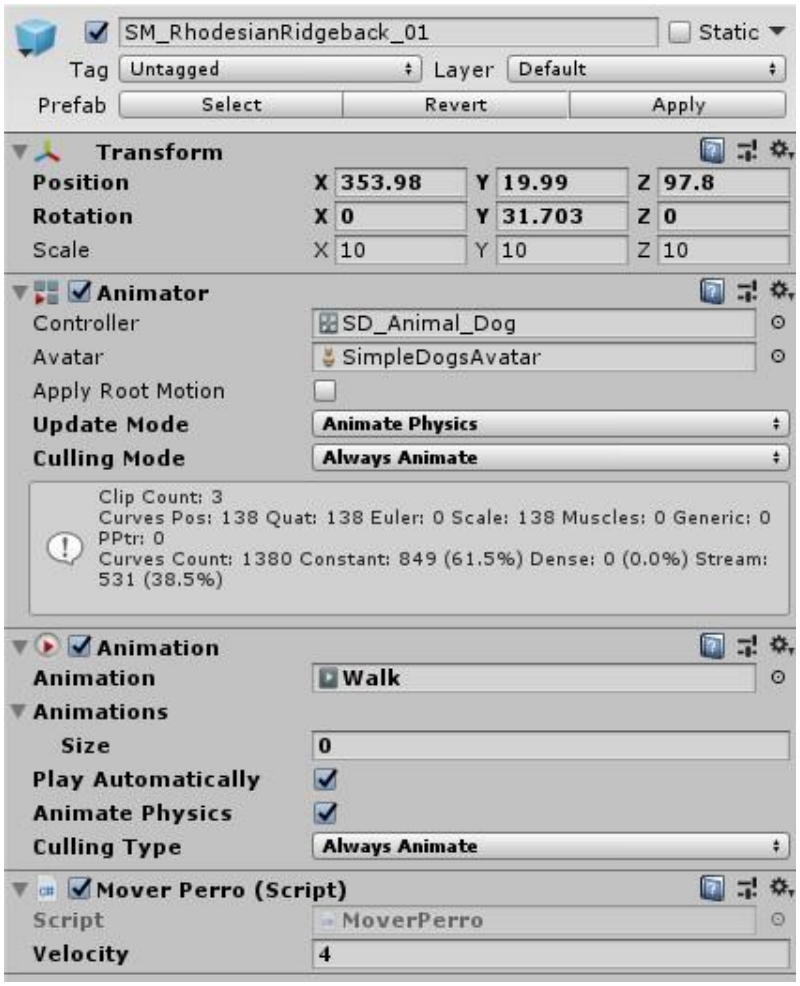

Ilustración 63: Inspector Asset perro

A continuación podemos observar también el menú Animator del perro en el cual tenemos los estados Walk y Run, además hemos añadido la variable booleana Andar con la cual podemos hacer la transición de un estado a otro. En un principio cuando se ejecute la escena se pasará directamente del estado Entry al estado Walk por lo tanto el perro comenzará a andar.

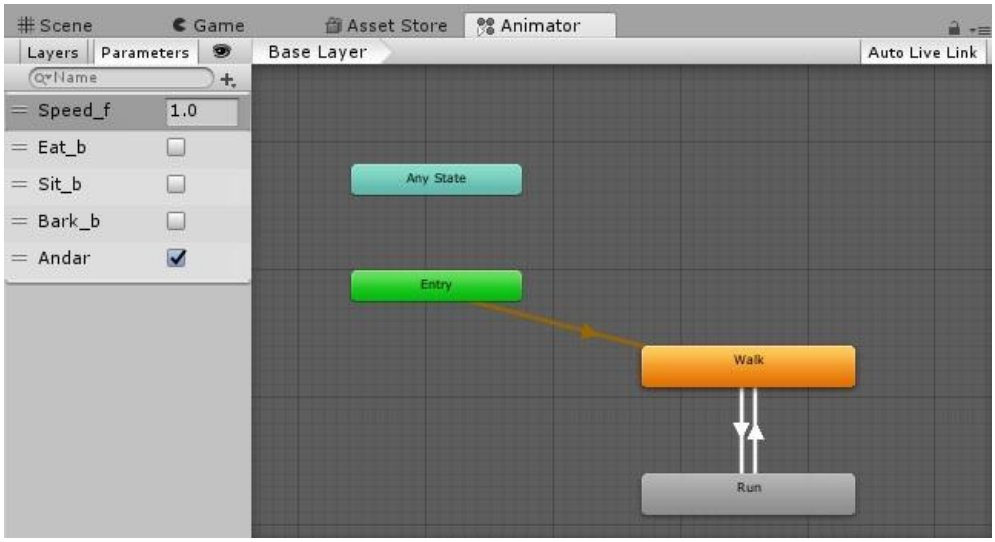

Ilustración 64: Animator Perro

Podemos ver la transición del estado Walk a Run, para esta transición tenemos la condición de que la variable booleana Andar sea igual a false por lo tanto cuando esto se cumpla el perro comenzara a realizar su Animation de correr.

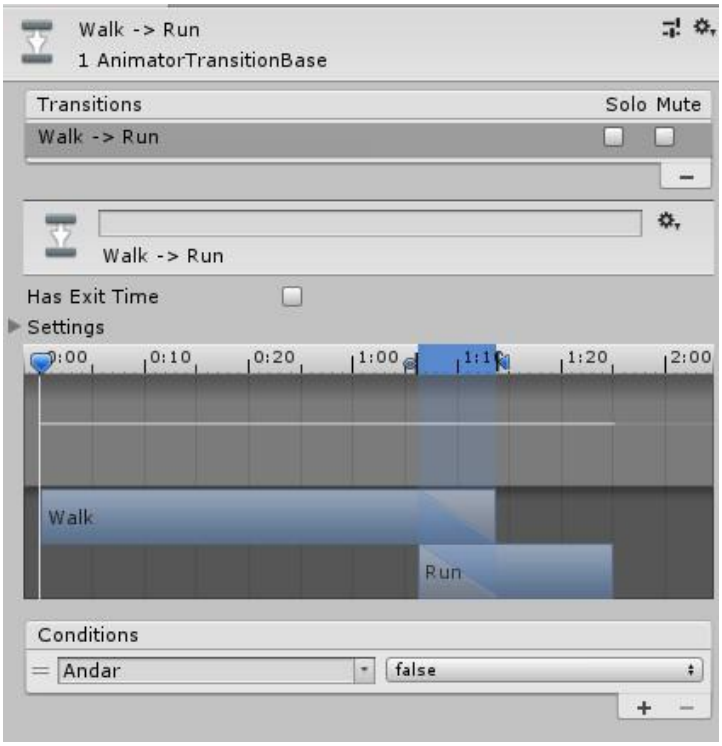

Ilustración 65: Transición Walk a Run

Por último vamos a ver el Script que hemos creado para realizar el movimiento del perro, es muy parecido al de la persona, sin embargo en la función Update hemos añadido la condición para que si se pulsa la letra 'O' del teclado el perro cambie su velocidad y haga la transición del estado Walk al estado Run ya que ponemos la variable Andar a false.

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

public class MoverPerro : MonoBehaviour {

    // Use this for initialization
    Animator animator;
    Transform myTranform;
    //private bool Andar = true;
    public float velocity;
    void Start () {
        animator = this.GetComponent<Animator> ();
        myTranform = this.GetComponent<Transform>();
    }

    // Update is called once per frame
    void Update () {
        transform.Translate(new Vector3 (0,0,velocity) * Time.deltaTime) ;

        if (Input.GetKey (KeyCode.O))
        {
            Andar = false;
            velocity = 5;
        }
```

Ilustración 66: Script MoverPerro

3.2.2. Explicación

A continuación vamos a ver algunas configuraciones que se han utilizado en la escena. En la siguiente ilustración podemos observar la escena cuando abrimos el proyecto.

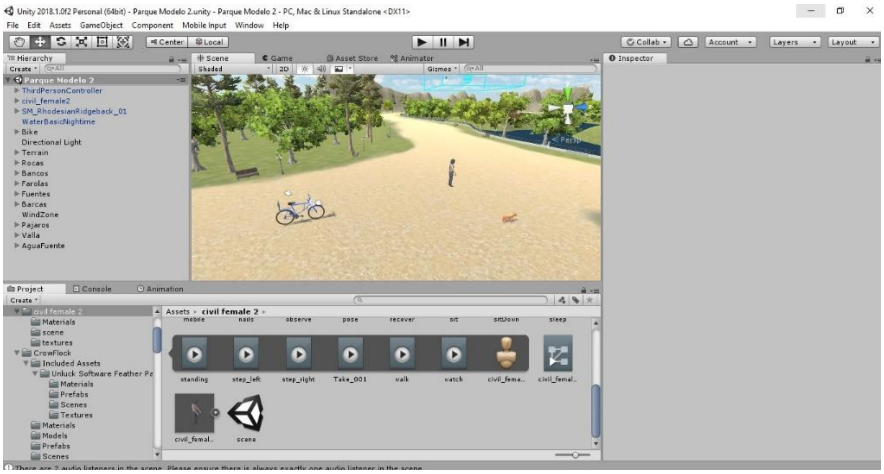

Ilustración 67: Escena al abrir el proyecto

En la izquierda podemos ver el menú Hierarchy en el cual tenemos todos los Assets empleados en la escena, como se puede observar están ordenados por carpetas. Se puede ver la carpeta de bancos donde están incluidos todos los bancos que se han puesto en la escena, en otra carpeta podemos ver las farolas añadidas en la escena…

Ilustración 68: Menú Hierarchy

A continuación podemos ver el Inspector del terreno, podemos observar como en este terreno hemos añadido un Script, Skybox y también el Audio Source en el cual hemos añadido el sonido de un parque con gente paseando, sonido de pájaros, niños jugando…

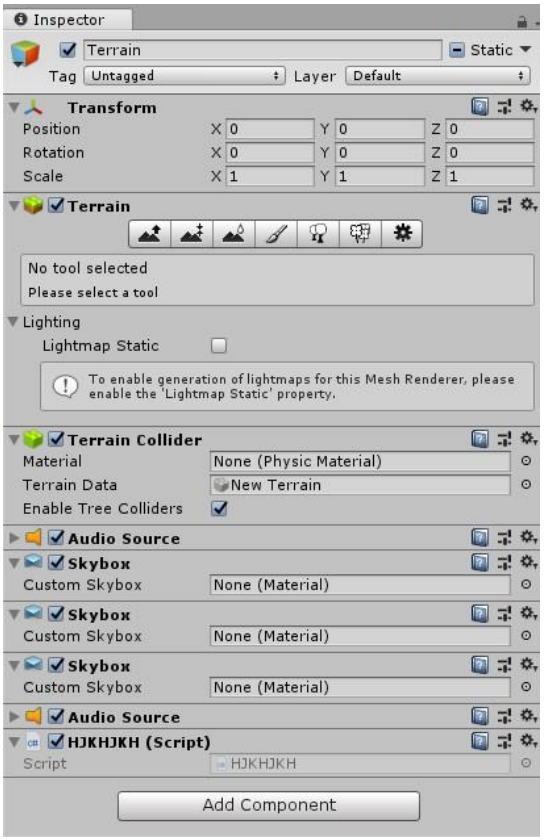

Ilustración 69: Inspector del terreno

Si vemos el menú Audio Source podemos observar el clip de audio que hemos empleado para añadir a la escena, así como también podemos configurar el volumen, la prioridad y otros ajustes del audio. Para añadir un clip de audio necesitamos un clip en formato MP3.

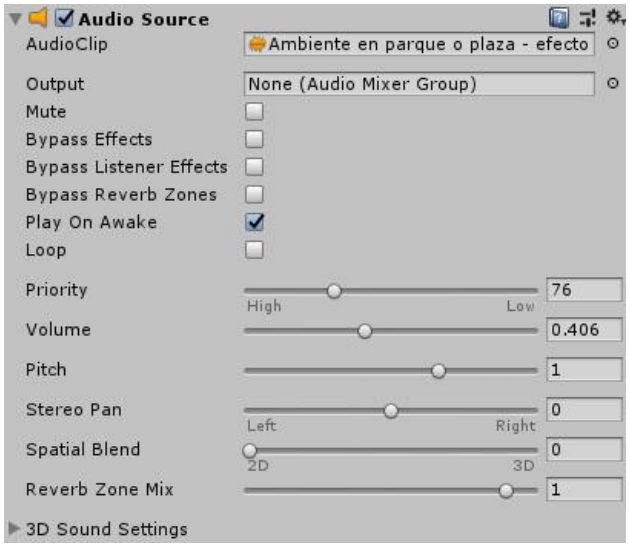

Ilustración 70: Menú Audio Source

En la escena también hemos añadido unas bandadas de pájaros que volarán por el cielo a la vez que reproducimos la escena. Este es un Assets descargado para el cual podemos ver su inspector a continuación en el cual se pueden configurar la cantidad de pájaros, la velocidad… Con este Asset conseguimos aportar un movimiento que queda muy bonito a la hora de reproducir la escena.

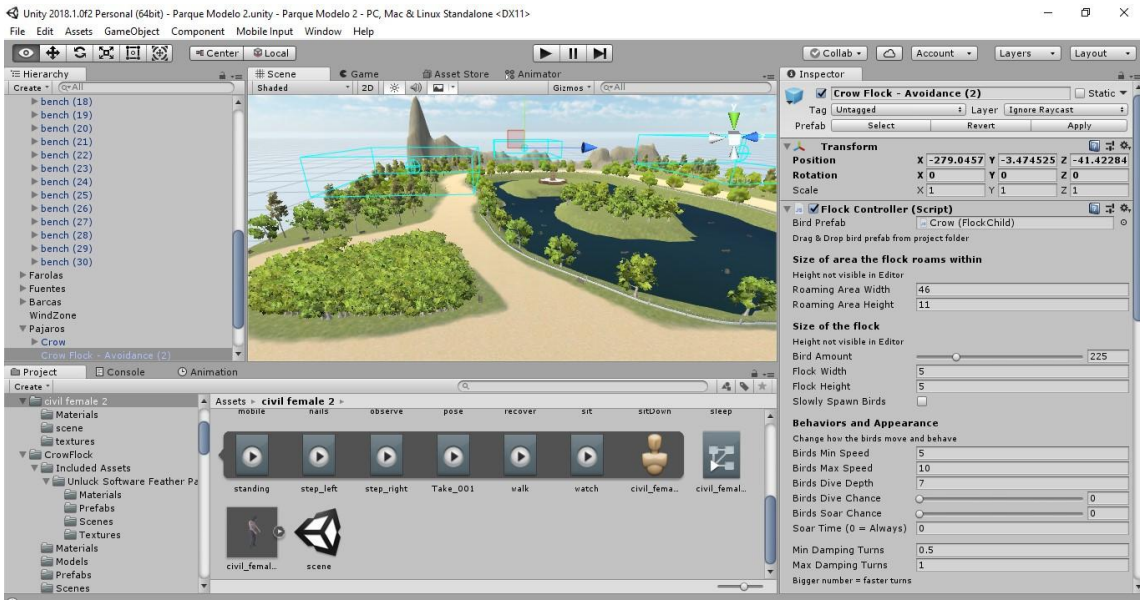

Ilustración 71: Escena con Assets de pájaros seleccionado

Ahora vamos a ver el Script MoverBici con el cual vamos a realizar el movimiento de la bicicleta, este Script que vamos a ver es una adaptación para el ordenador ya que la bicicleta se moverá mediante el teclado del ordenador. Declaramos las librerías necesarias y después empezamos nuestra clase pública. En esta clase vamos a declarar las variables públicas speed e yOffset las cuales controlaremos desde Unity y servirán para indicar la velocidad de la bicicleta y la posición del eje Y respectivamente. También necesitamos la variable myBici del tipo Transform donde obtendremos el Transform de

nuestra bicicleta. En la función Start simplemente obtendremos el Transform de nuestra bicicleta en la variable myBici.

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
5   public class MoverBici : MonoBehaviour {
6       Transform myBici;
7       public float speed;
8       public float yOffset;
9
10      // Use this for initialization
11      void Start () {
12          myBici = this.GetComponent<Transform>();
13      }
```

Ilustración 72: Script Bici declaración

En la función Update comprobaremos que letra del teclado hemos pulsado, si es una 'O', entonces pondremos la variable speed igual a 5, crearemos la variable directionO del tipo Vector3 y en este Vector 3 leeremos el Transform de cada eje y será multiplicado por la variable speed y por el tiempo. También creamos la variable rotationO del tipo Quaternion en la cual vamos a tener la rotación del eje Y. Ahora llamamos a la función Translate de la librería en la cual multiplicamos directionO y rotationO para que desplace la bicicleta. Por último modificamos la posición de nuestra bicicleta mediante lo que tengamos en el Transform de los ejes X y Z y con lo que tengamos en la variable yOffset. Hacemos lo mismo para las letras L, P y K solo jugando con las direcciones de los ejes para cambiar la dirección de la bicicleta. Si no pulsamos ninguna tecla lo que hacemos es poner la velocidad a cero para que no se mueva.

```
// Update is called once per frame
void Update () {
    if (Input.GetKey (KeyCode.O))
    {
        speed = 5;
        Vector3 directionO = new Vector3 (myBici.transform.forward.x, 0, myBici.transform.forward.z).normalized * speed *
        Quaternion rotationO = Quaternion.Euler(new Vector3(0, -transform.rotation.eulerAngles.y, 0));
        myBici.Translate(rotationO * directionO);

        myBici.position = new Vector3 (transform.position.x, yOffset, transform.position.z);
    }
    else if (Input.GetKey (KeyCode.L))
    {
        speed = 5;
        Vector3 directionL = new Vector3 (myBici.transform.forward.x, 0, -myBici.transform.forward.z).normalized * speed
        Quaternion rotationL = Quaternion.Euler(new Vector3(0, -transform.rotation.eulerAngles.y, 0));
        myBici.Translate(rotationL * directionL);

        myBici.position = new Vector3 (transform.position.x, yOffset, transform.position.z);
    }
```

Ilustración 73: Script bicicleta teclado ordenador parte 1

```
else if (Input.GetKey (KeyCode.K))
{
    speed = 5;
    Vector3 directionK = new Vector3 (-myBici.transform.forward.x, 0, myBici.transform.forward.z).normalized * speed * Time
    Quaternion rotationK = Quaternion.Euler(new Vector3(0, -transform.rotation.eulerAngles.y, 0));
    myBici.Translate(rotationK * directionK);

    myBici.position = new Vector3 (transform.position.x, yOffset, transform.position.z);
}
else if (Input.GetKey (KeyCode.P))
{
    speed = 5;
    Vector3 directionP = new Vector3 (-myBici.transform.forward.x, 0, -myBici.transform.forward.z).normalized * speed * Tim
    Quaternion rotationP = Quaternion.Euler(new Vector3(0, -transform.rotation.eulerAngles.y, 0));
    myBici.Translate(rotationP * directionP);

    myBici.position = new Vector3 (transform.position.x, yOffset, transform.position.z);
}else{
    speed = 0;
    Vector3 direction = new Vector3 (-myBici.transform.forward.x, 0, -myBici.transform.forward.z).normalized * speed * Time
    Quaternion rotation = Quaternion.Euler(new Vector3(0, -transform.rotation.eulerAngles.y, 0));
    myBici.Translate(rotation * direction);

    myBici.position = new Vector3 (transform.position.x, yOffset, transform.position.z);
}
```

Ilustración 74: Script bicicleta teclado ordenador parte 2

Por último podemos ver como se reproduciría el parque mediante las gafas de realidad virtual en el cual podemos ver en primera persona como si estuviéramos montados en la bicicleta virtual.

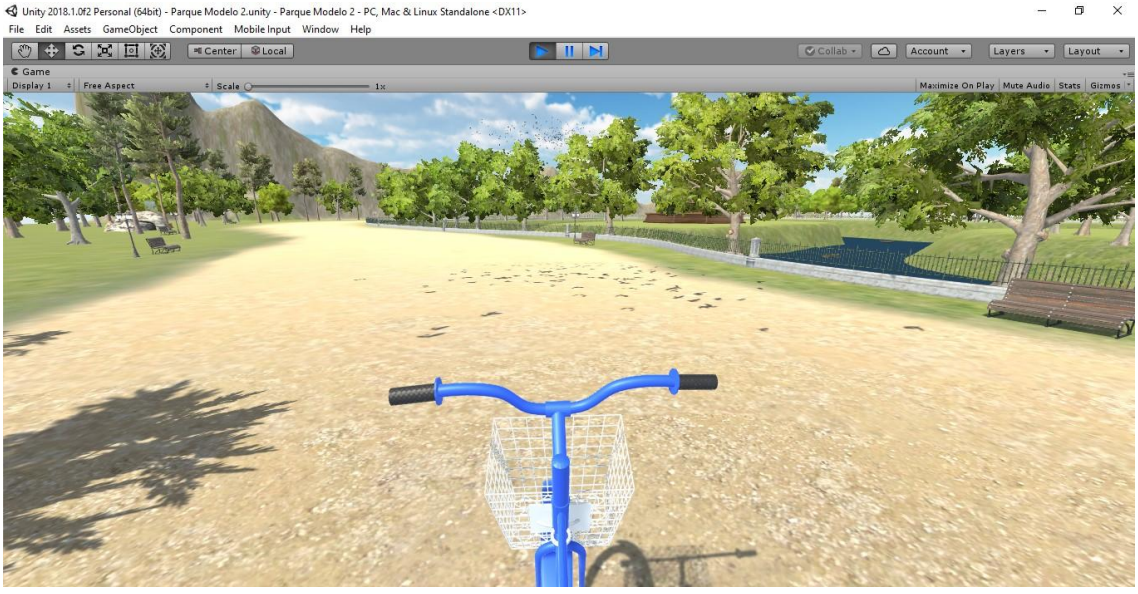

Ilustración 75: Parque con vista de ejecución de la aplicación

4. MICROCONTROLADOR ARDUINO

En esta sección vamos a ver como se ha desarrollado la parte del microcontrolador arduino para poder conseguir el objetivo de nuestro simulador. Veremos cómo se ha utilizado el microcontrolador arduino para conseguir que detecte cada paso de los pedales reales de nuestra bicicleta y como se lo comunica a Unity para que la bici avance. Veremos en esta sección el Kit de Arduino que hemos utilizado, también veremos la parte de montaje y programación de Arduino, y por último veremos cómo conectamos Arduino con Unity a través del puerto serie.

4.1. Kit de Arduino

Para el proyecto se ha utilizado un kit de Arduino básico el cual me cedió la Universidad Politécnica de Cartagena. Es un kit de Sparkfun en el cual simplemente hay un kit de Arduino básico el cual incluye cables, LED's, cable USB, placa Arduino, motor, resistencias, una placa protoboard donde montar los circuitos… Este kit puede rondar en el mercado los 60 euros y es un kit muy práctico para comenzar a realizar los primero montajes en una placa Arduino. No se necesita ninguna experiencia previa en programación o electrónica para empezar a este equipo, lo que es muy importante para cualquier persona que tenga curiosidad e interés en el manejo de Arduino. A continuación podemos observar el kit con el que hemos llevado a cabo el proyecto **[17]**.

Ilustración 76: Kit Arduino

En el caso de nuestro proyecto solo necesitamos utilizar del kit anterior para el montaje de nuestro circuito electrónico: la placa Arduino, la placa protoboard, el cable USB y unos cuantos cables para las conexiones. Por último solo necesitamos para el montaje un periférico de la placa Arduino, en este caso un sensor de infrarrojos con el que vamos a detectar cada uno de los pasos de los pedales. El módulo que he usado para el proyecto es el módulo FC-51 el cual tuve que comprar y que está valorado en aproximadamente

2 euros. Es un módulo muy económico y muy fácil de utilizar ya que con solo alimentarlo ya detecta el obstáculo en este caso nuestro pedal, y a partir del código de programación podemos hacer que cada vez que detecte el pedal le indique a Unity que la bicicleta debe de avanzar. A continuación podemos ver el módulo de sensor infrarrojo FC-51.

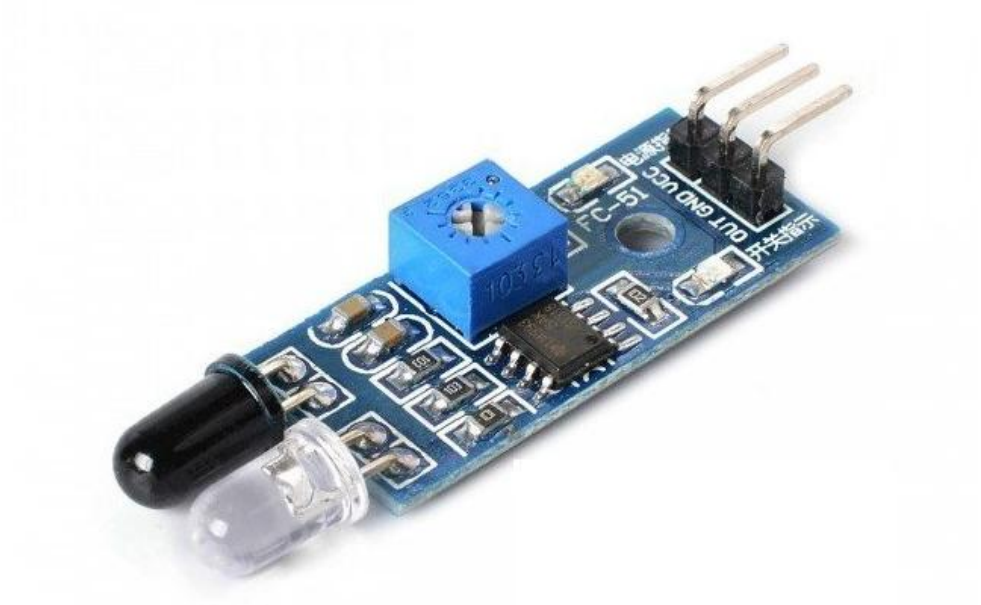

Ilustración 77: Módulo FC-51

Por último podemos ver una última ilustración en la cual podemos ver descrito cada una de las partes del módulo FC-51 y cual son sus funciones. Por ejemplo el LED negro es el recibidor de infrarrojo y el LED transparente es el emisor infrarrojo.

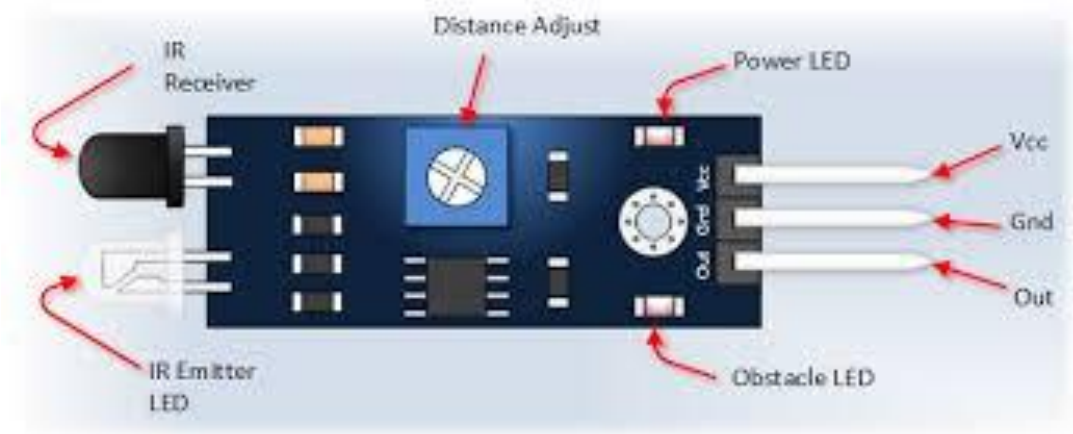

Ilustración 78: Sensor FC-51 funciones

4.2. Montaje y Programación

Ahora vamos a ver como se ha desarrollado el montaje electrónico del circuito para realizar el detector infrarrojo que necesitamos. Como he comentado anteriormente solo se necesita alimentar el sensor FC-51 para ello necesitamos conectar los 5 Voltios de la placa Arduino al puerto Vcc del sensor, después conectamos GND, (tierra), con el puerto GND del sensor y por último conectamos la salida OUT del sensor con el puerto A0 de la placa Arduino. Si queremos se puede conectar un LED a la placa Arduino en los puertos GND y 13, este LED se iluminará cada vez que el sensor FC-51 detecte el pedal.

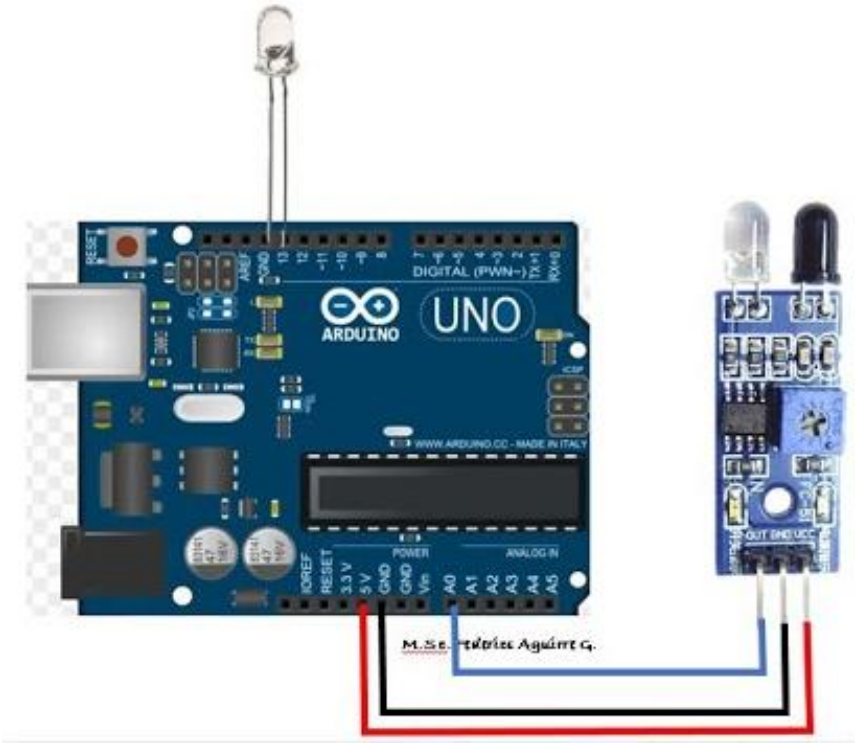

Ilustración 79: Circuito de sensor infrarrojo

Ahora vamos a ver el montaje realizado físicamente sobre la placaboard, he conectado un cable que sale del pin de 5V y que irá a la placa en una línea que consideramos de alimentación, después de esta línea sacamos otro cable que ira a la patilla VCC de nuestro módulo infrarrojo, esta línea de alimentación es la que tiene el cable amarillo. Después hemos cogido un cable negro para hacer la línea de GND, la cual irá desde el pin de GND hasta una línea que he considerado de tierra, y después de esta línea he sacado otro cable que irá a la patilla GND del módulo. Por último con un cable de color verde uniremos la patilla OUT del módulo con el pin 9 de tipo digital de nuestro Arduino y ya tendríamos el circuito necesario montado.

Para la alimentación simplemente hemos conectado el cable USB que lleva integrado el kit de Arduino al puerto USB del PC.

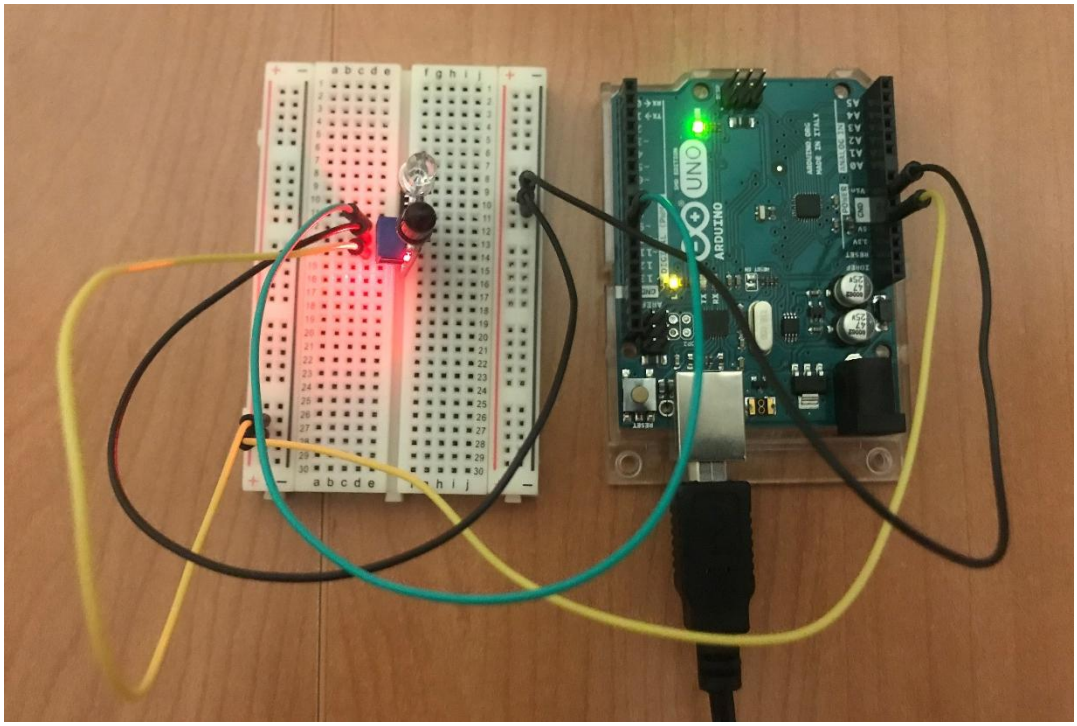

Ilustración 80: Montaje físico Arduino

Aquí podemos observar como cuando detecta un obstáculo por los LED's que lleva integrados el módulo en la parte superior se enciende el piloto izquierdo que indica que está pasando un obstáculo, sin embargo si no detecta nada entonces no se encenderá dicho LED y solo estará encendido el de la parte derecha.

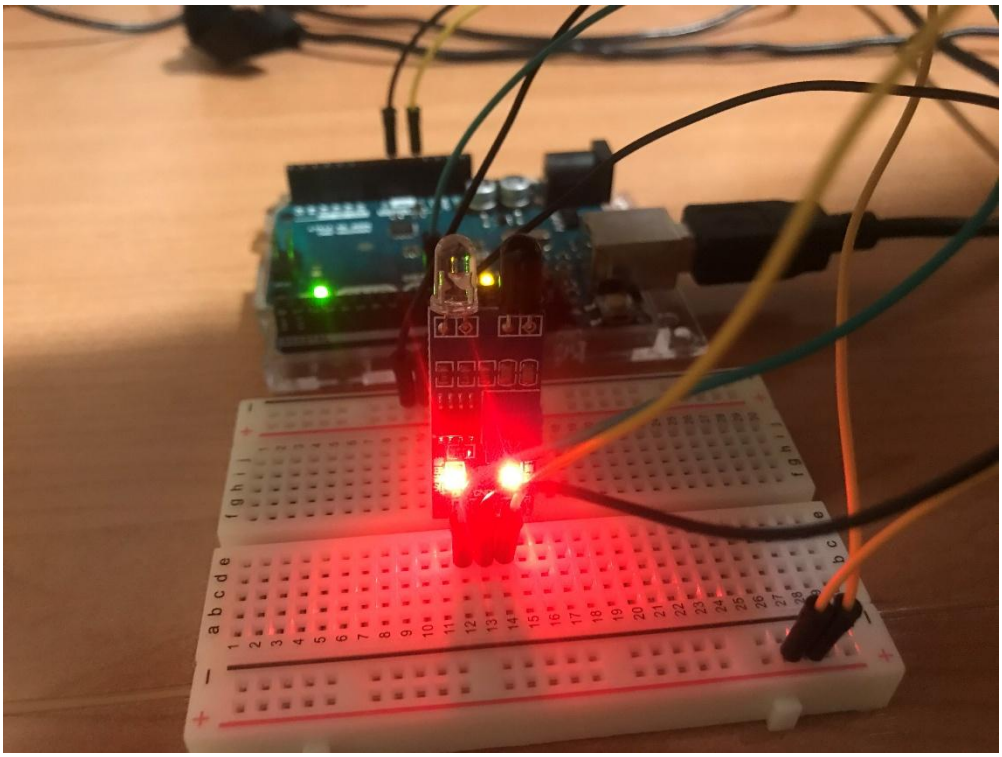

Ilustración 81: Módulo con detección de obstáculo

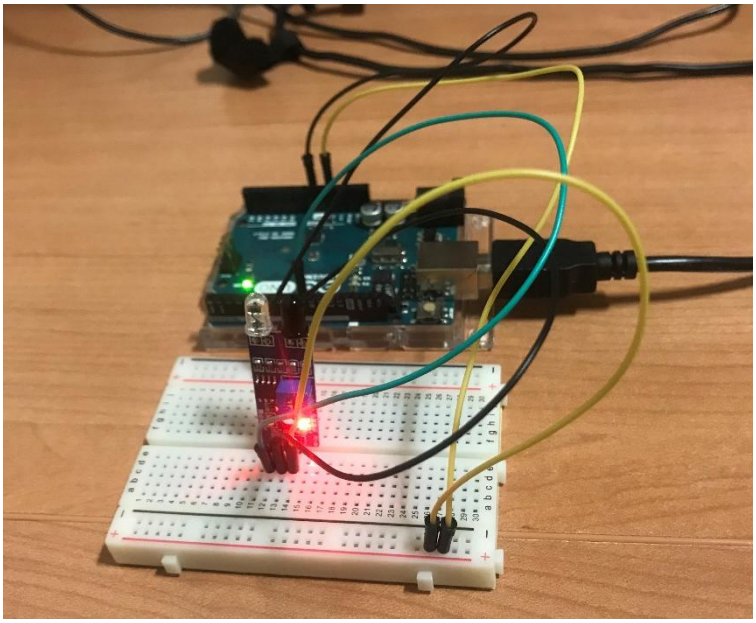

Ilustración 82: Módulo sin detección de obstáculo

Una vez realizado el montaje he realizado el código necesario tanto en Arduino como en el Script de Unity de la bicicleta para que avanzara cada vez que se detecta el obstáculo. Lo primero que necesitamos es descargar el programa para el código de Arduino el cual es gratis y lo podemos descargar de la página principal de Arduino.

El código de Arduino es muy sencillo, ya que solo necesitamos enviar a Unity un dato cada vez que nuestro módulo detecte el obstáculo o en este caso el pedal. Lo primero que hacemos es declarar una constante llamada InPin donde pondremos el número 9 que es el pin por el que estamos recibiendo datos del módulo. Ahora crearemos la función setup en la cual vamos a configurar los baudios en este caso 9600, y pondremos el pin 9 como entrada mediante la constante declarada anteriormente.

Por último creamos la función loop, en la cual declaramos la variable value de tipo int y con valor cero. En esta variable guardamos lo que leemos del pin 9, osea lo que está detectando el módulo. Si el valor de la variable es 0 escribimos el valor para que lo podamos leer en Unity y finalmente añadimos un retardo.

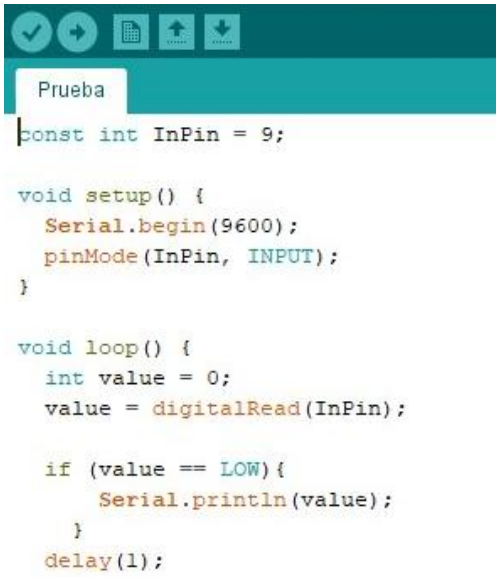

Ilustración 83: Código Arduino

A la hora de conectar Arduino con Unity hay que tener en cuenta la configuración de Arduino y en que placa de Arduino estamos trabajando. En este caso se está trabajando con la placa Arduino Uno por lo tanto seleccionamos dicha placa en el menú de Herramientas.

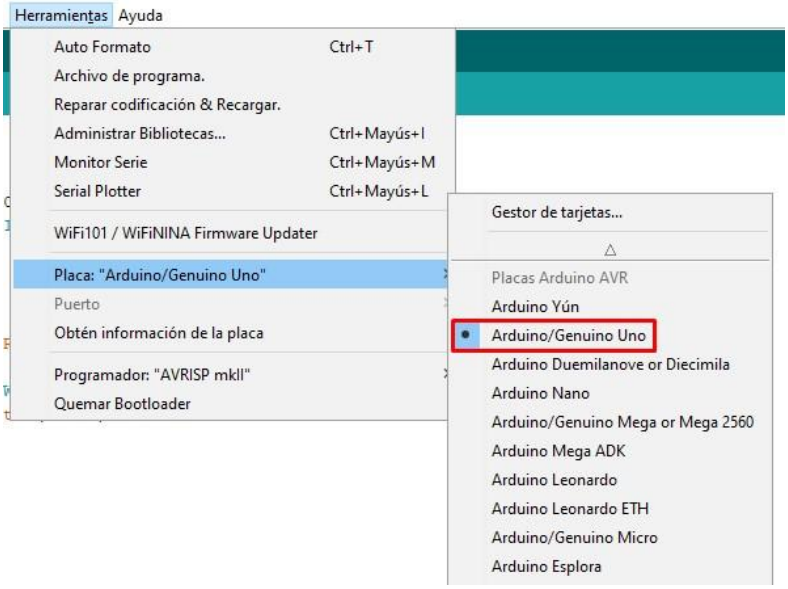

Ilustración 84: Configuración placa

También tenemos que tener en cuenta el puerto por el que vamos a transmitir, esto lo podemos localizar fácilmente en la parte de abajo del programa de Arduino, también lo podemos cambiar en Herramientas->Puerto. En este caso transmitiremos por el puerto COM3 ya que lo tenemos que tener en cuenta para el Script de Unity.

Ilustración 85: Puerto Arduino

A la hora de realizar el código para Unity tenemos que coger el Script asignado a la bicicleta y editarlo para poder obtener los datos que están llegando desde Arduino. Lo primero que se hace es añadir la librería System.IO.Ports esta librería la necesitaremos para poder realizar la conexión por puerto serie con Arduino. Dentro de la clase pública necesitamos declarar la variable myBici del tipo Transform ya que aquí vamos a obtener el Transform de nuestra bici. También declaramos las variable speed e yOffset del tipo float, estás dos variables se podrán configurar desde Unity y servirán para indicar la velocidad de la bicicleta y su posición en el eje Y. Por último declaramos la variable pública serialPort del tipo SerialPort la cual la inicializamos con los valores del puerto donde vamos a leer en Arduino 'COM3' y también los baudios a los que hemos configurado el arduino en este caso 9600 **[18]**.

En la función Start obtenemos el Transform de la bici en la variable myBici, abrimos el puerto serie mediante la llamada a la función Open integrada en la librería y añado un retardo.

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4   using System.IO.Ports;
5
6   public class MoverBici : MonoBehaviour {
7       Transform myBici;
8       public float speed;
9       public float yOffset;
10      public SerialPort serialPort = new SerialPort ("COM3", 9600);
11      // Use this for initialization
12      void Start () {
13          myBici = this.GetComponent<Transform>();
14          serialPort.Open ();
15          serialPort.ReadTimeout = 100;
16      }
```

Ilustración 86: Script bici para Arduino parte 1

Por último dentro de la función Update realizo un Try Catch para controlar las excepciones que pudieran ocurrir y lo que hago es comprobar que si el puerto serie está abierto, y si es así leo lo que nos llega en la consola y llamo a la función MoverArdu la cual recibe como parámetro un int. Si el parámetro recibido es igual a cero lo igualo a 1 y hago que la bicicleta se mueva poniendo la variable speed igual a cinco y moviendo la bicicleta mediante un Translate de la dirección obtenida, multiplicada por la rotación.

```
64          try{
65              if(serialPort.IsOpen){
66                  print(serialPort.ReadLine());
67                  MoverArdu(int.Parse(serialPort.ReadLine()));
68              }
69          }catch (System.Exception ex){
70              ex=new System.Exception();
71          }
72
73      }
74
75      void MoverArdu(int valor){
76          if (valor == 0){
77              valor = 1;
78              speed = 5;
79              Vector3 directionAr = new Vector3 (myBici.transform.forward.x, 0, myBici.transform.forward.z).normalized * spee
80              Quaternion rotationAr = Quaternion.Euler(new Vector3(0, -transform.rotation.eulerAngles.y, 0));
81              myBici.Translate(rotationAr * directionAr);
82
83              myBici.position = new Vector3 (transform.position.x, yOffset, transform.position.z);
84          }
85
86      }
```

Ilustración 87: Script bici para Arduino parte 2

5. EJECUCIÓN EN ANDROID

Al final debido a toda la situación actual causada por la pandemia mundial del Covid-19 no he podido conseguir la ejecución de mi trabajo en las gafas Oculus ya que son propiedad de la Universidad y no se podían sacar del centro y tampoco realizar la configuración allí debido a la pandemia. Por lo tanto hemos optado por realizar la ejecución de nuestra escena en un móvil con sistema Android, en este caso un móvil modelo Samsung Galaxy A8. En este punto vamos a ver como se han realizado todas las configuraciones necesarias para poder ejecutar la escena creada en Unity a través de una aplicación en el móvil. Para la ejecución tampoco hemos usado finalmente los pedales aunque tenemos toda la configuración realizada, debido a que necesitábamos un módulo bluetooth para comunicar el arduino con el móvil del que no disponíamos y tampoco teníamos bicicleta estática ni pedales a disposición para la ejecución.

5.1 Configuraciones

Lo primero que debemos hacer es descargar el módulo de Android para Unity, si utilizáramos las gafas Oculus tendríamos que instalar el módulo correspondiente. Para descargar el módulo de Android debemos ir al menú desplegable File y seleccionar la opción Build Settings, File->Build Settings. Aquí se nos abrirá la ventana Build Settings con las distintas plataformas donde podemos ejecutar la aplicación. Por defecto nos aparece seleccionada la opción PC, Mas & Linux Standalone, sin embargo a nosotros nos interesa la opción Android. Cuando pulsamos sobre Android si no tenemos descargado el módulo para Android nos pedirá que lo descarguemos, en la siguiente ilustración podemos ver un ejemplo de cómo nos saldría para IOS ya que el módulo de Android ya lo tengo descargado, simplemente tendríamos que pulsar sobre el botón Open Download Page y descargar el módulo.

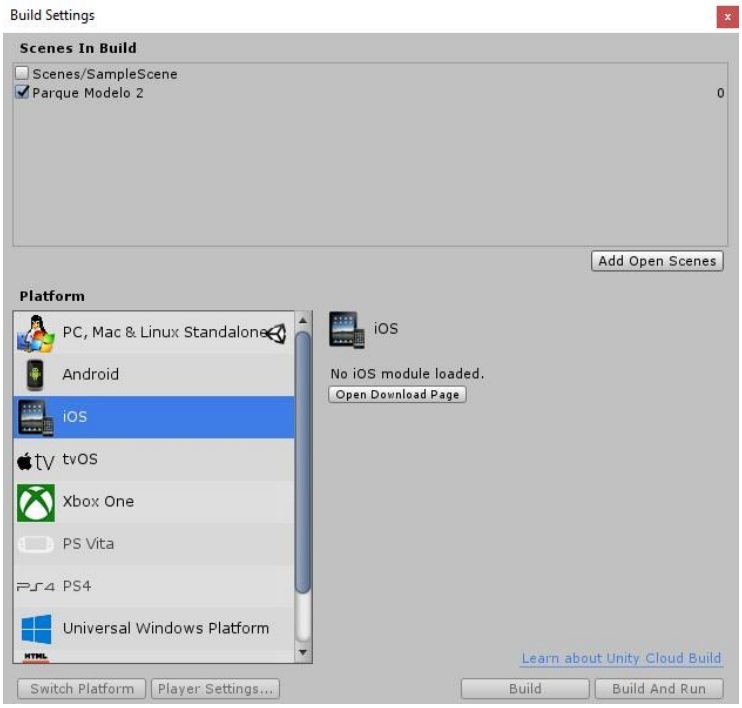

Ilustración 88: Ejemplo de descarga para módulo IOS

Una vez descargado el módulo de Android para Unity necesitamos descargar otros dos programas, el programa Android Studio y el programa para los JDK de Java. Para descargar estos dos programas tenemos que pulsar sobre el menú desplegable Edit y después seleccionar la opción Preferences, a continuación se nos abrirá la ventana Unity Preferences y aquí elegimos la opción External Tools, Edit->Preferences->External Tools. Abajo podemos ver que pone Android y una línea para SDK, otra línea para JDK y otra línea para NDK que en este caso no necesitamos. A la derecha de las líneas podemos ver los botones Download que nos llevarán directamente a la página de descargas tanto de SDK, (Android Studio), como de JDK, (Oracle), para esta última necesitaremos crear una cuenta con nuestros datos para poder descargar el programa. Una vez descargados solo tendremos que poner la ruta donde se hemos instalado nuestros programas y estará listo.

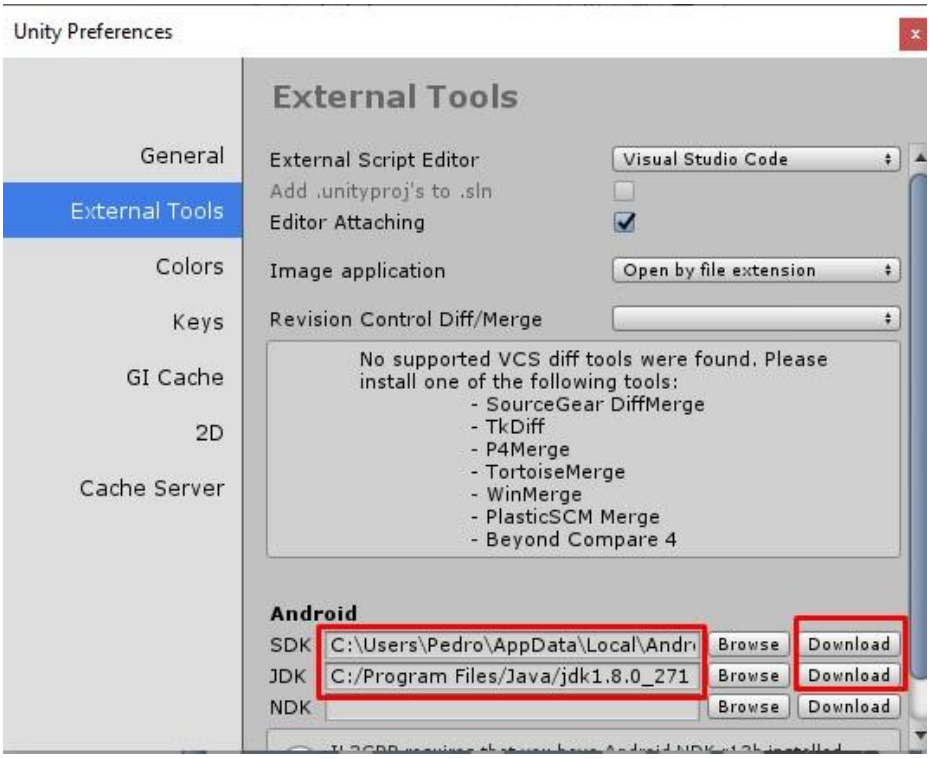

Ilustración 89: Ventana Unity Preferences

Ahora debemos de realizar una serie de descargas dentro del programa Android Studio antes de seguir con la configuración en Unity. Cuando abrimos el programa pulsamos sobre el botón de configuración abajo y elegimos la opción SDK Manager.

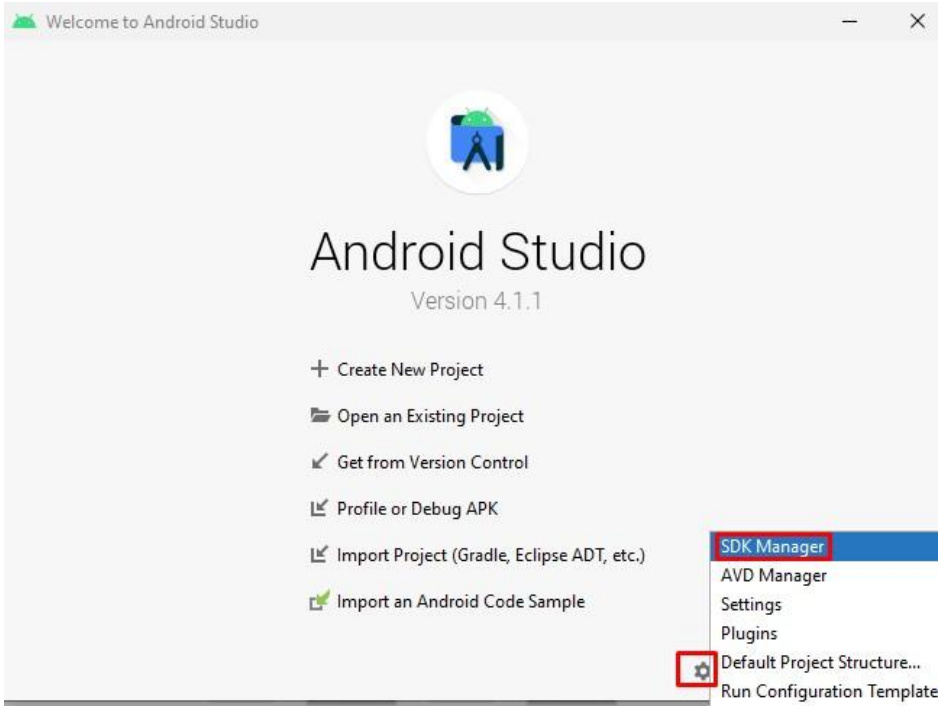

Ilustración 90: Configuración Android Studio

En la parte de System Settings, vamos al apartado Android SDK y elegimos la opción SDK Platforms aquí vamos a instalar todas la versiones de Android donde queremos que

funcione nuestra aplicación yo he instalado desde la más nuevas hasta la versión Android 4.4.

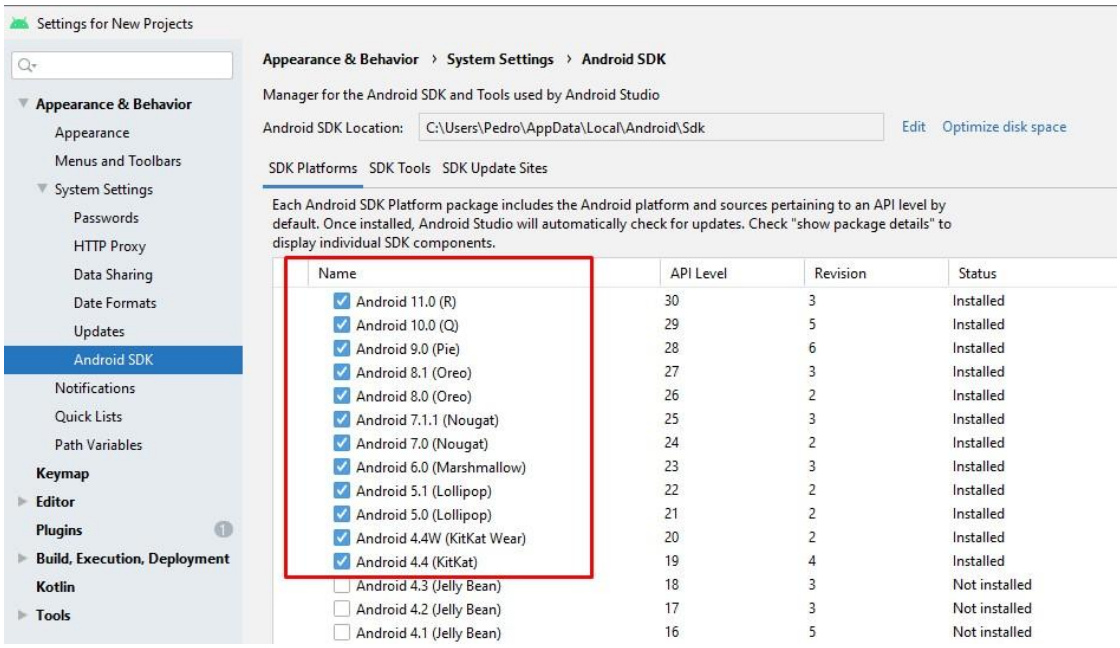

Ilustración 91: Versiones de Android a instalar

Por último seleccionamos la pestaña SDK Tools donde podemos instalar las herramientas que necesitaremos en nuestra aplicación, una vez seleccionadas tanto las herramientas como las versiones de Android, podemos descargarlas pulsando en Apply y después en OK.

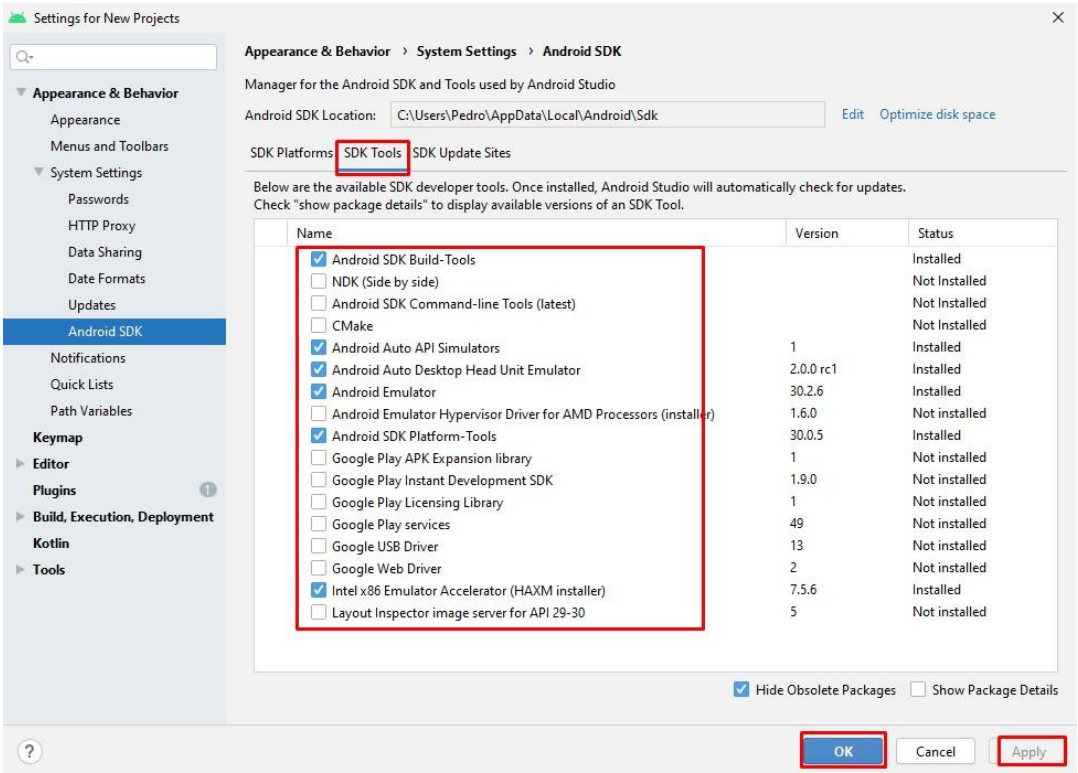

Ilustración 92: Herramientas necesarias para la aplicación

Antes de terminar la configuración en Unity debemos tener en cuenta que como no vamos a usar Arduino para mover la bicicleta debemos de cambiar la función Update del Script. Ahora cuando la aplicación se ejecute en el móvil, la bicicleta deberá de ponerse en movimiento, ya que al no tener el módulo de bluetooth para Arduino no podemos mandar la información al móvil, la función Update simplemente cambiará la velocidad y cambiará la dirección y la rotación de la bicicleta como ya hemos visto anteriormente.

```
18  // Update is called once per frame
19  void Update () {
20
21      speed = 5;
22      Vector3 direction = new Vector3 (myBici.transform.forward.x, 0, myBici.transform.forward.z).normalized * speed * Time.
23      Quaternion rotation = Quaternion.Euler(new Vector3(0, -transform.rotation.eulerAngles.y, 0));
24      myBici.Translate(rotation * direction);
25
26      myBici.position = new Vector3 (transform.position.x, yOffset, transform.position.z);
```

Ilustración 93: Script, función Update para bicicleta Android

Ahora sí vamos a terminar la configuración en Unity, debemos de ir a la ventana Build Settings, File->Build Settings, y elegir la plataforma Android de la cual ya tendremos el módulo descargado. Aquí vamos a pulsar sobre el botón Player Settings, aquí debemos de configurar todos los parámetros de nuestro proyecto para pasarlos al móvil con sistema Android y que no nos de ningún fallo.

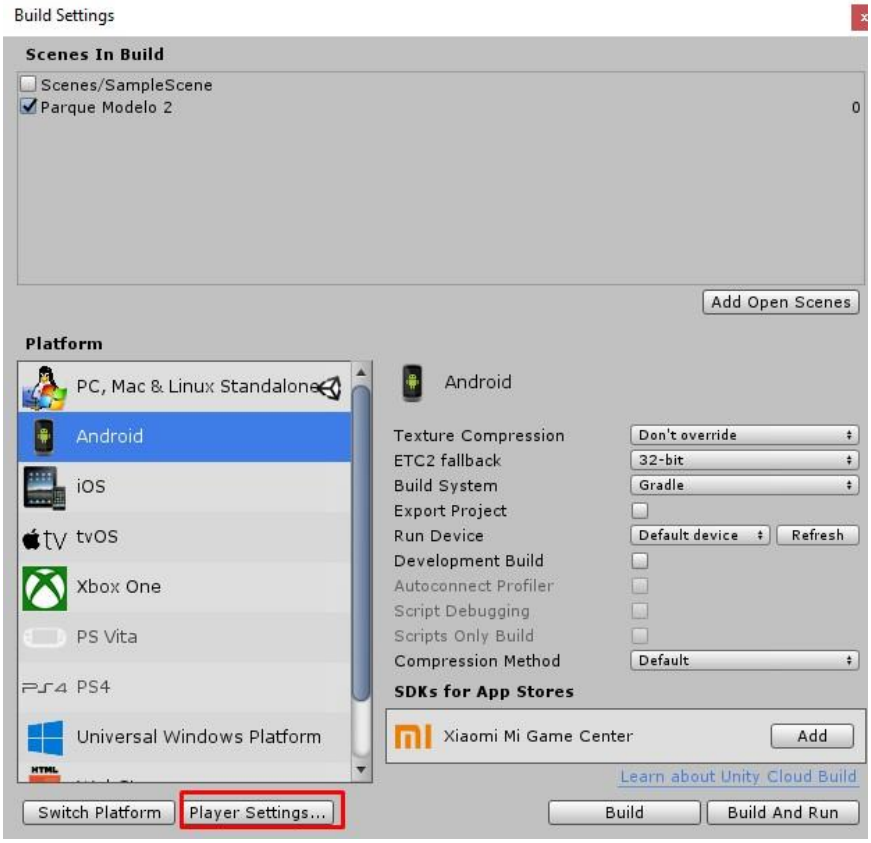

Ilustración 94: Botón Player Settings

El menú Player Settings se nos abrirá en el Inspector, lo primero que debemos hacer es poner el Company Name y el Product Name, que será el nombre de la empresa aunque en este caso no haya ninguna le tenemos que dar un nombre y el nombre de la

aplicación. También debemos elegir el icono que queremos para nuestra aplicación, en este caso yo he puesto el icono de la Universidad Politécnica de Cartagena, para poner el icono simplemente debemos arrastrarlo hasta el cuadro.

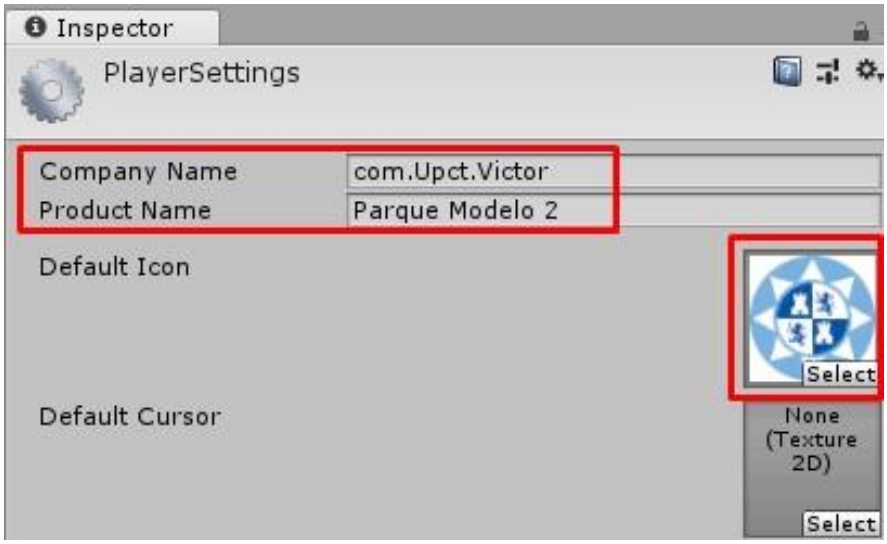

Ilustración 95: Player Settings configuración nombre e icono

Si seguimos bajando en el Inspector, otro punto que debemos de configurar es Other Settings, aquí en la parte Identification debemos de poner el nombre del paquete, la versión y también es importante el parámetro Minimum API Level que nos indica la versión mínima de Android que necesitamos para nuestra aplicación en este caso hemos puesto la versión 4.4. También debemos de configurar en la parte Configuration el parámetro API Compatibility Level, el cual tenemos que poner la opción .NET 2.0 ya que por defecto lleva la opción .NET 2.0 Subset que es una versión más reducida que nos puede dar errores al compilar.

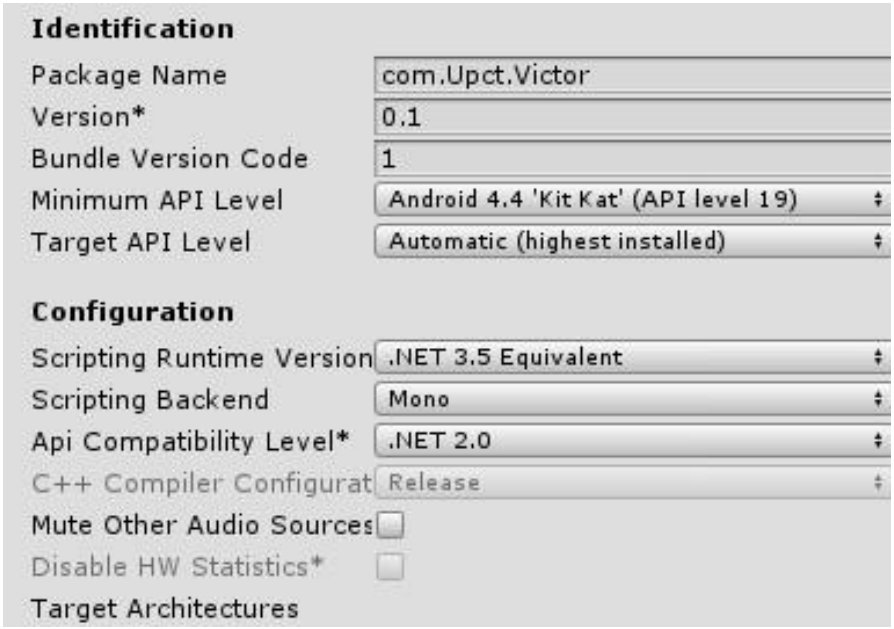

Ilustración 96: Configuración Other Settings

Ahora solo nos queda configurar la parte XR Settings, en esta parte debemos de marcar el parámetro Virtual Reality Supported, además debemos de añadir una Cardboard en la parte de Virtual Reality SDKs, por último el parámetro Stereo Rendering Method lo dejamos en Multi Pass.

Ilustración 97: Configuración XR Settings

Solo nos queda cargar nuestra aplicación en el móvil, para esto volvemos a la ventana Build Settings en la opción de Android. Si en el cuadro Scenes In Build no aparece nuestra escena pulsamos sobre el botón Add Open Scenes y seleccionamos la nuestra. Una vez tengamos nuestra escena solo tenemos que pulsar sobre el botón Build y elegir la ubicación de nuestro ordenador donde queremos que se genere nuestra aplicación en formato APK.

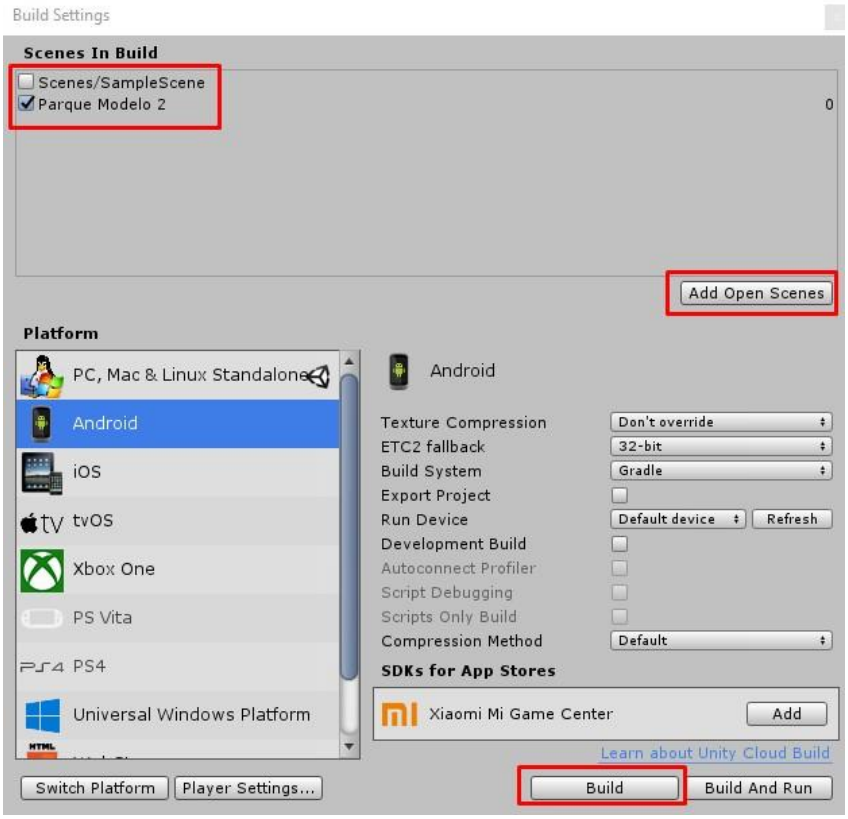

Ilustración 98: Generar aplicación

Ya solo nos queda una vez se haya generado nuestro archivo en formato APK importarlo a nuestro móvil e instalar la aplicación. Finalmente aquí podemos ver una ilustración de cómo queda la aplicación ejecutada a través de nuestro dispositivo Android.

Ilustración 99: Escena vista a través del dispositivo Android

6. CONCLUSIONES

Se ha cumplido con el objetivo de poder crear un entorno virtual que simule la estancia en un parque, además de la consecución de poder aprender a controlar el microcontrolador Arduino y poder interactuar con él desde Unity. Aunque esto último no se ha podido realizar con pedales tal y como se tenía previsto debido a que nos faltaba un módulo bluetooth para Arduino.

Por el tema de la pandemia del Covid-19 no se ha podido conseguir probar el trabajo con las gafas Oculus Rift, aunque si se ha podido utilizar Android para conseguir ejecutar el entorno virtual a través de un móvil con sistema Android.

Desde el momento en el que supe que llegaba al final de mi grado en la Universidad Politécnica de Cartagena y que tenía que elegir un trabajo para el final del grado tuve claro que quería hacer algo que sirviera para ayudar a otras personas. Al principio me surgieron otras opciones para trabajos con líneas de investigación sin embargo siempre tuve claro que quería algo más especial.

Cuando hable con Paqui la primera vez en su despacho, recuerdo que me propuso la maravillosa idea de este trabajo el cual aparte de poder aprender nuevas técnicas y programas, me daba la posibilidad de intentar llevarlo a un hospital y poder ayudar a la gente que peor lo está pasando. Ya habíamos tenido algún contacto con el hospital Virgen de la Arrixaca para poder llevar a cabo nuestro objetivo pero finalmente debido a la pandemia mundial que no ha tocado vivir no lo hemos logrado. La función principal era la llevar nuestra aplicación al hospital y que fuera testeada por los pacientes, o el de enseñar mediante diversas prácticas el manejo de Unity y otros programas a los mismos, simplemente con el afán de conseguir disuadir a los pacientes durante un rato de la situación que les está tocando vivir. No pierdo la esperanza de que algún día aunque no esté ya estudiando en la Universidad, pueda lograr probar la aplicación en algún hospital siempre considerando la opinión de Paqui que es la principal responsable de esta idea.

Este proyecto ha sido un gran reto para mí, desde el día que lo empecé he tenido que terminar algunas asignaturas que me quedaban pendientes, además de compaginarlo con mi trabajo. Cuando empecé con el trabajo no sabía nada de Unity o Arduino, así que he tenido que ir aprendiendo con la ayuda de Paqui y de muchos tutoriales a través de internet. Considero que este trabajo me puede ayudar a la hora de desenvolverme y aprender nuevos programas y herramientas de los que no conozco, sobre todo para mi trabajo en el día a día.

Simplemente me gustaría aprovechar para dar las gracias infinitas a Paqui por su infinita paciencia conmigo, por recibirme cada día que iba hacer preguntas sobre el trabajo con una sonrisa y sobre todo por esta maravillosa idea del trabajo fin de grado que era justo lo que yo quería aunque no se haya logrado de momento el objetivo.

7. BIBLIOGRAFÍA

[1] R.- ASALE y RAE, «realidad | Diccionario de la lengua española», *«Diccionario de la lengua española» - Edición del Tricentenario*. https://dle.rae.es/realidad

[2] « ¿Qué es la realidad virtual? - Historia, funcionamiento y gafas VR ¤¤ Mundo Virtual», *Mundo Virtual*. http://mundo-virtual.com/que-es-la-realidad-virtual/.

[3] «Realidad virtual», *Wikipedia, la enciclopedia libre*. dic. 06, 2020. Disponible en: https://es.wikipedia.org/w/index.php?title=Realidad_virtual&oldid=131498111.

[4] «Unity (motor de videojuego)», *Wikipedia, la enciclopedia libre*. nov. 17, 2020. Disponible en: https://es.wikipedia.org/w/index.php?title=Unity_(motor_de_videojuego)&oldid=130993146.

[5] M. L. González, « ¿Qué es Unity3D y por qué utilizarlo? analizamos sus ventajas», *Bravent*, oct. 28, 2016. https://www.bravent.net/que-es-unity3d-y-por-que-utilizarlo-analizamos-sus-ventajas

[6] «Qué es Arduino, cómo funciona y qué puedes hacer con uno». https://www.xataka.com/basics/que-arduino-como-funciona-que-puedes-hacer-uno

[7] «Oculus VR», *Wikipedia, la enciclopedia libre*. nov. 21, 2020. Disponible en: https://es.wikipedia.org/w/index.php?title=Oculus_VR&oldid=131107983.

[8] «Oculus Rift de realidad Virtual - Características», *Gafas Oculus*. https://www.gafasoculus.com/rift/

[9] «Marion Surgical Company Profile: Valuation & Investors | PitchBook». https://pitchbook.com/profiles/company/173452-51

[10] «Marion Surgical | Simulators». https://www.marionsurgical.com/SIMULATORS

[11] «About Us - Vesaro». https://www.vesaro.com/store/pc/viewcontent.asp?idpage=1

[12] «Vesaro 195 Flight», *Vesaro*. https:/www.vesaro.com/store/pc/viewPrd.asp?idproduct=315

[13] «Vesaro I Professional VR». https://www.vesaro.com/store/pc/viewCategories.asp?idCategory=641

[14] «Birdly VR | El último sueño de volar». https://birdlyvr.com/

[15] Paul, «A VR Cycling Experience for $40», *Yanman!*, ene. 24, 2016. https://pauldyan.wordpress.com/2016/01/24/my-vr-bike/

[16] U. Technologies, «Unity Store - Descargar». https://store.unity.com/es/download

[17] «SIK Experiment Guide for Arduino - V3.3 - learn.sparkfun.com». https://learn.sparkfun.com/tutorials/sik-experiment-guide-for-arduino---v33/all

[18] «Conectando Arduino y Unity a través del puerto serie | Sensorización». http://sensorizacion.blog.tartanga.eus/conectividad/conectando-arduino-y-unity-a-traves-del-puerto-serie/