PROYECTO FINAL DE INGENIERÍA

REALIDAD VIRTUAL EN BICICLETA FIJA

Ares Villafañe, Franco Oscar – LU1018999

Mohovich, Ivanna Sofía – LU1020533

Ingeniería en Informática

Tutor/es:

Ronconi, Francisco, UADE

Octubre 30, 2017

UNIVERSIDAD ARGENTINA DE LA EMPRESA

FACULTAD DE INGENIERÍA Y CIENCIAS EXACTAS

Resumen

El proyecto consiste en el diseño y desarrollo de un prototipo para una aplicación móvil que permite al usuario desplazarse en un mundo de realidad virtual utilizando una bicicleta fija, un dispositivo móvil y un casco de realidad virtual sin utilizar hardware adicional.

Basándose solamente en los sensores que posee cualquier dispositivo móvil, la aplicación ofrece al usuario libertad de movimiento en un entorno preseleccionado de realidad virtual. El usuario puede seleccionar distintos tipos de entrenamiento así como también la dificultad, la sensibilidad del dispositivo móvil, el escenario en donde entrenar y si la dirección varía por la rotación de la cabeza o es fija. También muestra el tiempo transcurrido y la cantidad de metros recorridos en el entrenamiento.

Toda la navegabilidad de la aplicación se realiza con el casco puesto pudiendo seleccionar la opción deseada manteniendo la vista durante 1.5 segundos sobre ella. De este modo, se logra prescindir de cualquier control adicional que pueda molestar al usuario mientras entrena.

Se registran los entrenamientos realizados guardando la dificultad seleccionada, el tiempo transcurrido y los metros recorridos, pudiendo ver desde la aplicación los últimos cinco entrenamientos.

El trabajo abarca las etapas principales del proceso de desarrollo de software, es decir, la Definición de los Requerimientos, el Análisis, el Diseño, el Desarrollo y el Testing, las cuales se desarrollaron siguiendo el paradigma de metodologías ágiles y utilizando las técnicas de Scrum y de XP Programming para el desarrollo del prototipo funcional. En lo que concierne a la aplicación se utilizó Unity3d como herramienta utilizando C# como lenguaje de programación.

Abstract

The project consisted in the design and development of a prototype for a mobile application to allow the user to move in a virtual reality world using a stationary bike, a mobile device and a virtual reality headset without the need of any extra hardware.

Using just the sensors built in the mobile device, the app brings freedom of movement to the user under a virtual reality environment. The user may choose between several types of training. Furthermore the user may change the difficulty of the training, the sensibility of the mobile device, the landscape in which the user would navigate and if the direction of movement is set by rotating the wearer's head or just a fixed direction.

The user should look at the different options to for 1.5 seconds in order to navigate the user interface. This way the user may dispense with any external control that might bother him while training.

All trainings are saved, the information that is stored is the pre selected difficulty, the time the training took and the distance covered. The user can see his five last trainings on the main menu.

The assignment covered the first four principal steps of the software development process, namely, Requirements Definition and Analysis, Design, Development and Testing, which were developed following the paradigm of the agile methodologies with the techniques of Scrum and XP Programming for the development phase of the functional prototype. Unity3d was the framework chosen to develop the prototype as well as C# was the language to develop it.

Introducción

El objetivo de este proyecto es desarrollar un prototipo de aplicación móvil completamente funcional en donde una persona andando en una bicicleta fija con un casco de realidad virtual pueda moverse libremente en un mundo virtual.

Si bien se analizan distintas alternativas, la meta del proyecto es captar el movimiento de la persona e interpretar esta información para prescindir completamente de elementos que no sean la bicicleta fija y el casco de realidad virtual, así como también se analizan distintas alternativas de cascos con la finalidad de hacerlo más accesible a cualquier usuario.

El presente trabajo se compone de las siguientes secciones:

- Antecedentes: En esta sección se comentan los distintos usos de realidad virtual con algunos ejemplos, así como también las distintas modalidades de movimiento dentro de cada aplicación. Se nombran, por otro lado, aquellas aplicaciones existentes, virtuales y no virtuales, similares a la temática de este proyecto marcando las diferencias de cada una con el prototipo a realizar.
- Descripción: Esta sección es el cuerpo principal del proyecto en donde se describen las primeras etapas del proceso de desarrollo de software desde la definición de los requerimientos hasta las pruebas llevadas a cabo para el desarrollo del prototipo. Por su extensión, se ha dividido en las siguientes subsecciones:
    - Definición de Requerimientos Funcionales
    - Atributos de Calidad
    - Arquitectura
    - Análisis General
    - Diseño Funcional
    - Diseño Lógico y Físico
    - Desarrollo y Codificación del Prototipo
    - Pruebas
- Metodología de Desarrollo: En esta sección se describe la metodología de desarrollo empleada para llevar a cabo el proyecto, así como también las técnicas

utilizadas, con una breve explicación sobre cómo fueron adaptadas a nuestras necesidades y por qué se decidió hacer uso de ellas.

- Recomendaciones para próximas etapas: En este apartado se sugieren diferentes enfoques para llevar a una versión productiva el prototipo desarrollado, pasando por los métodos de monetización posibles de emplear hasta la generación de terreno aleatorio, incluyendo formas de mejorar la experiencia del usuario ya sea agregando competencia entre jugadores o registros de entrenamiento, entre otras cosas.
- Análisis Final: En esta sección se hace un análisis sobre los resultados de las pruebas llevadas a cabo y se nombran las dificultades con las que nos encontramos en la realización del prototipo.
- Conclusiones: En esta sección final se sacan algunas conclusiones sobre los resultados de la ejecución del proyecto y la utilidad del prototipo, entre otras cosas.
- Bibliografía: En esta sección se definen los vínculos y referencias hacia todos los contenidos bibliográficos citados y utilizados durante el transcurso del desarrollo del proyecto.

Antecedentes

En cuanto a los antecedentes relacionados con la funcionalidad del proyecto, existen tanto aplicaciones como juegos con realidad virtual y cabe destacar que es una tecnología que está en los inicios de un crecimiento exponencial.

Existen aplicaciones utilizadas con fines profesionales, como aquellas relacionadas al área militar, medicinal, de la construcción, entre otras cosas. A modo de ejemplo la aplicación OramaVR sirve para practicar cirugías, StoryBoardVR para mostrar ideas, Fuzor para Arquitectura.

Otras aplicaciones son utilizadas con fines educativos en donde el usuario puede ingresar a una situación particular de la historia por ejemplo, o bien conocer un museo. Algunos ejemplos de estas aplicaciones son: BLVRD para recorrer un museo de arte,

Unimersiv para viajar en el tiempo y conocer la historia de Roma, Atenas, el Titanic y los Dinosaurios, así como también conocer una estación espacial.

Por otro lado, existen las aplicaciones utilizadas exclusivamente para el entretenimiento, como son los juegos. Ejemplos de estos hay varios, tanto para Smartphone como para Oculus, HTC Vive y otros cascos. Para Smartphones podemos encontrar entre los más populares el InCell VR y el Lamper VR. Para Oculus y HTC Vive están Rick and Morty: Virtual Rick-ality y Lone Echo entre muchos otros.

Finalmente, están aquellas utilizadas para practicar un deporte. Entrando más en la temática de nuestro proyecto encontramos algunas aplicaciones relacionadas al entretenimiento durante un entrenamiento en bicicleta fija.

VESCAPE es una aplicación de celular no virtual para monitorear el entrenamiento sobre una bicicleta fija, que toma la información que transmite la bicicleta mediante Bluetooth para obtener la velocidad y la resistencia del pedaleo. Dentro de VESCAPE se creó el mini juego Greedy Rabbit que trata de un conejo intentando escapar de los erizos.

BitGym es otra aplicación no virtual que muestra videos en primera persona de recorridos fijos por los Alpes Suizos, Venecia de Italia y El Gran Cañón de Estados Unidos entre tantos otros. Esta aplicación utiliza la cámara frontal del dispositivo móvil para calcular la velocidad con la que se mueve el usuario.

VirZOOM es una bicicleta fija que se vende en conjunto con un pack de juegos de realidad virtual. La bicicleta fija tiene las manijas adaptadas a un control de videojuego y se conectan vía Bluetooth.

CycleVR es una aplicación realizada por Aaron Puzey cuya meta era recorrer el Reino Unido en bicicleta sin salir de su casa. Para ello desarrolló una aplicación que utiliza las imágenes tomadas por Google’s Streetview para realizar el recorrido. El recorrido fue previamente fijado por él desde GoogleMaps desde un punto inicial hacia un punto final pasando por puntos que él quería conocer. Utilizó un sensor magnético que se conecta vía Bluetooth con el Smartphone para captar la velocidad con la que pedaleaba.

Por otro lado, podemos clasificar en 4 grandes grupos de aplicaciones de Realidad Virtual según la movilidad.

La más simple implica la transmisión de un video, en 360º, en la dirección en que el usuario mueve la cabeza. Por ejemplo, GoVRPlayer es una aplicación que permite ver videos en 360°, no solo hay videos recomendados dentro de la aplicación sino que también se pueden ver copiando una URL o videos locales del Smartphone.

En otras aplicaciones, el personaje se mueve automáticamente en la dirección en que se rote el casco. Es similar a las transmisiones en 360° pero la diferencia está en que es un juego. Por ejemplo: Lamper VR es un endless runner que consiste en recorrer la mayor cantidad de metros sin chocar contra obstáculos y en agarrar la mayor cantidad de caramelos. No se usan controles externos, todo se realiza con el movimiento del casco.

Por otro lado, están las aplicaciones que implican la utilización de joysticks o del pulsador del Google Cardboard. Como ejemplo de Google Cardboard tenemos el juego House of Terror. Por otro lado, cualquier aplicación de Oculus Rift y de HTC Vive está diseñada para la utilización de controles.

Finalmente, la solución más avanzada e inmersiva, y más costosa también, implica la utilización de plataformas especiales diseñadas exclusivamente para este fin. Es decir que el usuario camina y corre sobre la plataforma. Algunos ejemplos de estas plataformas son Virtuix Omni y Cyberith Virtualizer con juegos como el Karnage Chronicle y el Fallout 4 que permiten el uso de estas plataformas.

Descripción

Esta fase es el cuerpo principal del proyecto por lo que a continuación se procederá a explicar el desarrollo completo del prototipo desde el análisis hasta las pruebas realizadas. Cabe destacar que todas las imágenes, gráficos y diagramas utilizados en este documento son de producción propia de los autores del trabajo, expresando claramente las excepciones a esta afirmación.

Definición de Requerimientos Funcionales

La mayoría de los requerimientos funcionales fueron especificados al momento de definir el alcance del proyecto. Sin embargo, muchos de ellos fueron modificados durante el

desarrollo del proyecto o bien agregados para darle forma al prototipo. Estos fueron agrupados en un backlog del producto y ordenados de forma descendente según prioridad:

1. El usuario, al andar en una bicicleta fija, podrá moverse en un mundo de realidad virtual.
2. El usuario podrá elegir en qué dirección desea moverse al rotar la cabeza.
3. El usuario podrá elegir cuándo iniciar el entrenamiento.
4. El usuario podrá ver todas las opciones disponibles en un "Menú" (Inicio, Configuración -Música o Silenciar, Dirección Fija o Variable por Rotación, Dificultad Baja, Media o Alta, Duración por Tiempo o por Distancia, Sensibilidad, Escenarios-, Entrenamientos Anteriores).
5. El usuario podrá seleccionar la opción deseada dentro del Menú fijando la vista durante 1.5 segundos sobre el ítem correspondiente.
6. El usuario podrá pausar el entrenamiento luego de estar 5 segundos en reposo.
7. El usuario podrá Reanudar o Salir del entrenamiento cuando esté en Pausa.
8. El usuario podrá elegir fijar la dirección antes de comenzar el entrenamiento para poder rotar la cabeza y ver los alrededores.
9. El usuario podrá visualizar la cantidad de metros recorridos y el tiempo transcurrido
10. El usuario podrá seleccionar la dificultad del entrenamiento entre baja, media y alta. Al seleccionar entre los diferentes niveles, el recorrido durará más tiempo o más metros dependiendo si se encuentra en duración por tiempo o duración por distancia respectivamente.
11. El usuario podrá seleccionar la duración del entrenamiento en tiempo o en distancia.
12. El usuario podrá cambiar la sensibilidad del dispositivo para requerir mayor o menor esfuerzo para detectar el movimiento.
13. El usuario podrá elegir entre dos escenarios: playa o bosque.
14. El usuario podrá visualizar los entrenamientos realizados con anterioridad.

15. El usuario podrá seleccionar escuchar música ambiental o silenciar (en cuyo caso deberá poder utilizar la aplicación que desee para escuchar otra cosa).

Gamification

Uno de los conceptos más fuertes utilizados como base de este proyecto es el concepto de Gamificación. La gamificación o ludificación es la utilización de técnicas, elementos y mecánicas utilizadas principalmente en juegos en actividades no lúdicas, para así mejorar la motivación, creatividad y abordar problemas de diferentes perspectivas. También ayuda al aprendizaje y a la participación y cohesión de un grupo de personas. Es un concepto muy utilizado en empresas con el fin de alcanzar objetivos y mejorar la motivación del personal.

Un ejemplo de utilización de este concepto es un edificio donde el ascensor es muy utilizado y se requiere que los usuarios que van a los primeros pisos utilicen la escalera con el fin de disminuir la carga en los ascensores. Otro objetivo también es que las personas realicen actividad física. Para ello las escaleras simulan un piano en donde cada escalón emite un sonido diferente, generando así música al subirlas.

Bajo este concepto, se utilizan diferentes elementos y mecánicas de juegos llevados a estas actividades no lúdicas. Algunas de estas son:

- Puntos: se busca que el usuario intente lograr una buena performance en la actividad para sumar estos puntos. Al conseguirlos, se pueden entregar premios en base a esto. Se utilizan también para la competencia entre usuarios, intentando fortalecer las relaciones sociales y fomentar la participación en grupo. Por ejemplo, se otorgan más puntos a aquellos usuarios que trabajan en conjunto.
- Tabla de puestos o “leaderboard”: sumado al concepto anterior, estas tablas muestran qué usuario posee más puntos con respecto a los demás. Se utiliza para despertar el espíritu competitivo entre usuarios.
- Niveles: este concepto se utiliza para medir progreso y participación. Mientras más involucrado esté el usuario, su nivel será mayor. Esto incentiva a los nuevos usuarios que al involucrarse más obtienen mayor nivel.

- Coleccionables y logros: son premios otorgados luego de que un usuario cumpla una determinada tarea. Estas tareas implican que el usuario realizó un esfuerzo grande y por eso fue premiado. Al mostrar los coleccionables o logros de un usuario a otros, se fomenta la competitividad.
- Feedback: cuando el usuario realiza una actividad dentro del proceso, el sistema lo gratifica, ya sea desde un mensaje de felicitaciones hasta un premio virtual o real. Esto les da un refuerzo positivo a los usuarios al indicarles que se dirigen por buen camino y que deben seguir por esa dirección.

Gamification en el proyecto

Si bien estos conceptos no son todos, abarcan la gran mayoría de los utilizados en el proceso de gamificación. Entrando en el detalle de este proyecto, se utilizó el concepto de puntos asociados al tiempo y a la distancia del entrenamiento, así como también la tabla de entrenamientos anteriores para ver el progreso individual de cada usuario. Pero se podrían sumar todos los otros conceptos para etapas posteriores sin grandes dificultades.

A modo de ejemplo, se pueden desarrollar los niveles tomando la cantidad de metros acumulados para definir en qué nivel se encuentra el usuario. Por ejemplo, en juegos es muy utilizada la sucesión de Fibonacci para realizar una curva de progresión para definir niveles, en donde 1000 metros o menos corresponde a nivel uno, entre 1000 y 2000 metros corresponde a nivel dos, de 2000 a 3000 nivel tres, de 3000 a 5000 nivel cuatro y así siguiendo la curva. Es un método muy utilizado ya que el nivel puede ser calculado en tiempo de ejecución solo guardando los metros acumulados hasta ese momento y sin necesidad de persistir los umbrales de cada nivel.

En el caso de los coleccionables o logros, se puede incentivar a realizar una cierta cantidad de metros en un único entrenamiento para obtener un trofeo, o bien llevarlo a realizar un entrenamiento diario para aumentar la constancia en el entrenamiento, logrando de esta manera la retención del usuario.

Como feedback se puede sumar un mensaje de felicitaciones luego de cada entrenamiento, así como también calcular aproximadamente las calorías quemadas y mostrarlas. Esto genera el refuerzo positivo buscado incentivando al usuario a volver.

En cuanto a la tabla de puestos, se podría agregar la competencia entre usuarios de forma online, o bien en un gimnasio mostrarla por pantalla.

Atributos de Calidad

- Usabilidad: al tratarse de una aplicación orientada a un público general y con realidad virtual, es importante que sea atractiva y fácil de usar. Hay que tener en cuenta que la selección de opciones se realiza con el movimiento de la cabeza, es por esto que no se deben hacer menús extensos.
- Mantenibilidad: al tratarse de un prototipo funcional, es importante que el producto sirva de base para una aplicación productiva. Por ello, debe ser simple y fácil de entender, documentado lo suficiente como para que alguien que no participó del proyecto inicial pueda continuar y mejorar la aplicación.
- Performance: como es una aplicación de realidad virtual que correrá en Smartphone, es importante optimizar la performance para que la aplicación funcione lo más fluido posible y para que no haya fallas visuales que puedan producir náuseas o confusiones al usuario.
- Seguridad: por el lado del usuario, nuestra aplicación no contendrá información importante a proteger. Por el lado del código, se podría ofuscar para protegerlo ya que, al no haber servidores, toda la propiedad intelectual del código corre sobre el cliente, y con una ingeniería reversa se puede obtener fácilmente si no se hace nada al respecto.
- Escalabilidad: para los fines de este prototipo, la escalabilidad no es un atributo a tener en especial consideración. Pero en caso de expandir el prototipo a una aplicación productiva e incorporar competencia online o leaderboard, hay que tenerlo en cuenta.
- Portabilidad: si bien el prototipo será desarrollado para iOS y Android (Smartphone), será portable a otras plataformas, como Oculus Rift y HTC Vive, en caso de querer hacerlo.

Arquitectura

La siguiente figura (Fig. 1) corresponde al Diagrama de Arquitectura:

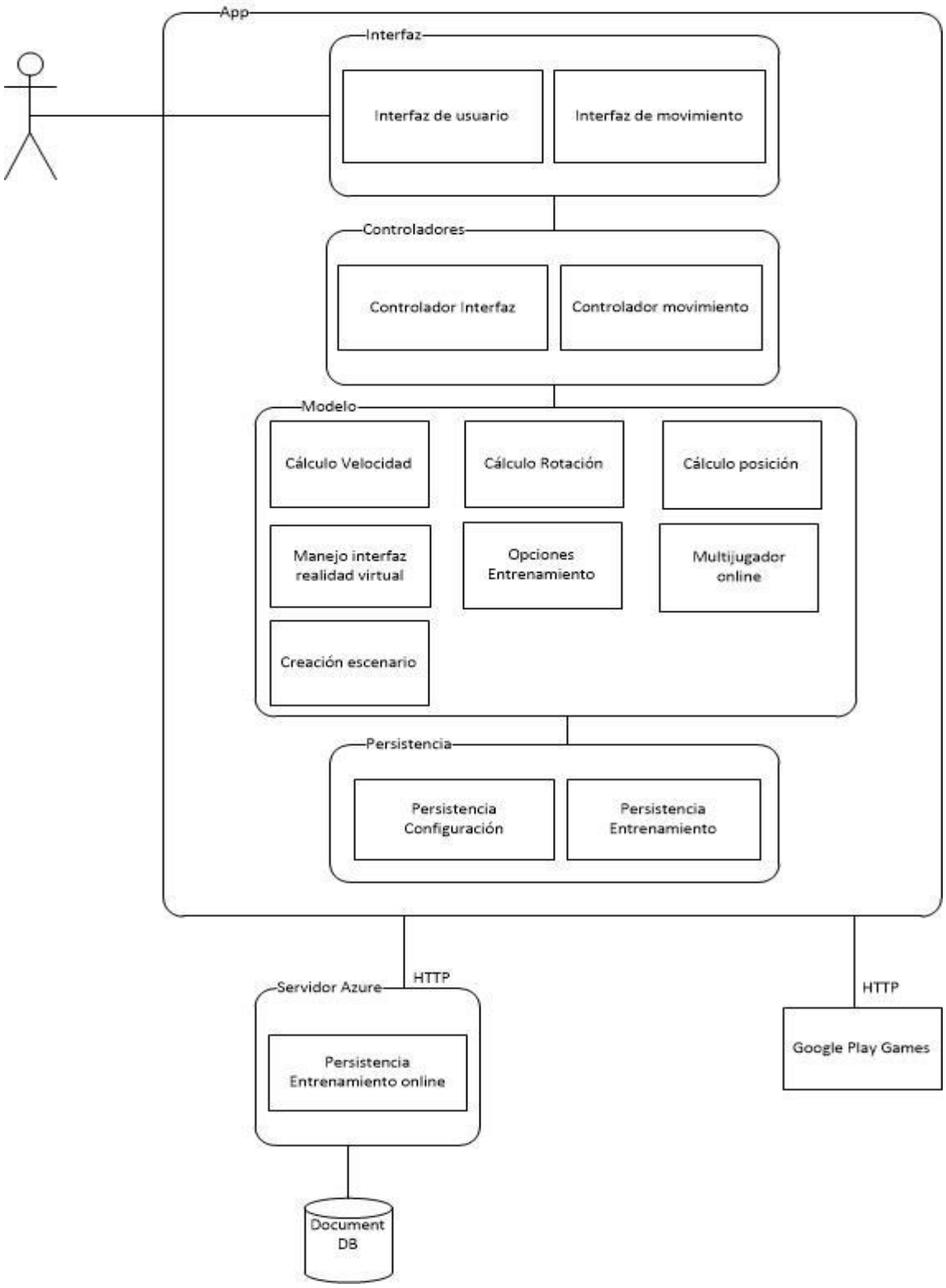

Figura 1: Diagrama de Arquitectura

Descripción

Controlador

- Interfaz: Se encarga de controlar la interacción con todos los botones de la aplicación. Al fijar la vista durante 1.5 segundos en cada botón, se carga una barra que al finalizar ejecuta la acción del mismo.
- Movimiento: Es el encargado de interpretar la rotación y aceleración del dispositivo móvil. Cada uno de ellos es guardado en un vector de tres dimensiones y utilizado por los módulos del modelo para diversos cálculos.

Modelo

- Cálculo de velocidad: Este módulo se encarga del cálculo de la velocidad recibiendo como entrada a los sensores del dispositivo móvil capturados por el controlador de movimiento. Utilizando un contador de tiempo, el giroscopio y el acelerómetro este módulo genera un valor de velocidad que luego se utiliza en el módulo de posición.
- Cálculo de rotación: Este módulo es el encargado de calcular la rotación dentro del mundo virtual al recibir la rotación percibida por el giroscopio del dispositivo móvil. Genera un vector de tres dimensiones que será utilizado por el cálculo de posición posteriormente.
- Cálculo de posición: Es el encargado de ubicar al personaje en el mundo de realidad virtual en la posición adecuada. Para ello utiliza los valores del cálculo de velocidad y el cálculo de rotación sumado a los objetos del entorno, como la altura del piso en el cual se está recorriendo, así como objetos con los que se pueda colisionar, como piedras o árboles.
- Opciones entrenamiento: Se encarga de mostrar y administrar todas las opciones de configuración del entrenamiento, es decir, el sonido de ambiente, la dificultad, el escenario seleccionado, la sensibilidad y la dirección por rotación o fija.
- Multijugador: Es el encargado de enviar los datos a sincronizar entre usuarios en una misma partida. Este envía los datos vía HTTP al servidor de google, el cual genera la partida, y luego los datos se envían de punto a punto.

- Creación de escenario: Este módulo se responsabilidad de cargar el escenario según la opción elegida, levantando todos sus componentes para que pueda ser navegable. Utiliza la posición del personaje para insertar partes del terreno frente a él produciendo así la sensación de terreno infinito. También, en caso de desarrollarse la generación del terreno aleatorio, debe ser el encargado de realizarlo según las especificaciones del código, instanciando todos los elementos pertinentes a ese escenario como piedras, árboles y agua.
- Manejo de interfaz: Es el encargado de almacenar y mostrar en la pantalla el tiempo transcurrido y la distancia recorrida por el usuario.

Servidor

- Persistencia entrenamiento: Se encarga de guardar los datos enviados por el cliente con relación a los entrenamientos.

Google Play Games

- Es el servidor de Google que se encarga de encontrar dos usuarios que estén intentando competir en simultáneo. El servidor se encarga de crear la partida pero no de administrar ya que uno de los dos clientes funciona como servidor controlando la partida.

Ofuscador

- Como la lógica está presente del lado del cliente, toda la propiedad intelectual queda vulnerable para ser robada. Es por eso que se decidió utilizar un ofuscador de código para agregar un nivel de dificultad mayor para compilar el APK generado por Unity. La ofuscación consiste en iterar por todo el código generando un hash por cada nombre de variable y clase, de modo que el código sea más difícil de interpretar. También se agregan métodos que no se utilizan y se encriptan todas las preferencias guardadas por Unity.

Herramientas

- Android SDK: Es el software utilizado para realizar la compilación en dispositivos Android. Unity lo utiliza realizar la compilación directamente desde su framework.
- GoogleVR SDK: Software incluido en Unity que genera la visión de un entorno desde dos cámaras separadas en la distancia adecuada para realidad virtual. Permite realizar pruebas desde el editor de Unity sin necesidad de compilar a un dispositivo móvil.
- Environment Low Poly Assets: Desde el Asset Store de Unity, se compró un producto con componentes en tres dimensiones de terrenos, agua, árboles y rocas para mejorar la visual del producto.

Servicios Externos

- Google play games: es un plugin encargado de realizar el protocolo de "handshake" para dos usuarios que están buscando partida online en simultáneo. Este plugin se encarga de unir los puntos, designar al usuario que funcionará de host y enviar los datos a sincronizar entre los usuarios sin pasar por el servidor de google.

Análisis General

Cascos Realidad Virtual

Los cascos de realidad virtual, también conocidos como Head-Mounted Displays (HMDs), cumplen una función vital en la experiencia del usuario.

Hay 3 grandes tipos de HMD en cuanto a Hardware:

Smartphone Dependiente

Son cascos en donde se coloca el Smartphone. Este es quien contiene el software, la batería, la pantalla y los parlantes (o auriculares). El casco tiene una división en el medio para separar la visión de cada ojo y contiene 2 lentes cuya posición es ajustable para mejorar la experiencia del usuario. Las ventajas generales que tienen es que son fácilmente trasladables, no se conectan a nada y son de menor costo. Hay una gran variedad de estos cascos, cada uno con sus pros y sus contras. Ejemplos de estos son el Google Cardboard, que es apto para hacer

uno mismo o bien comprarlo por 15 USD, VR Box por 35 USD, y el Samsung Gear VR por 130 USD (incluido el control). La siguiente es una tabla de comparación de estos 3 cascos:

Tabla 1: Tabla de Comparación Cascos Smartphone Dependiente

Nombre	Costo Aprox. (USD)	Tipo Display	Resolución	Audio	Tasa de refrezco	Optica	Input	Tipo
Samsung Gear VR Samsung	$95.74	Super AMOLED	2960x1440 (Galaxy S8 and S8+)	Smartphone	60 Hz	Sistema de doble lente, Foco ajustable, distancia entre ojos no ajustable	Touchpad, Control Inalámbrico	Smartphone Dependiente
VR Box	$35	Smartphone	Smartphone	Smartphone	Smartphone	Sistema de doble lente, Foco ajustable, distancia entre ojos ajustable	Smartphone y Control Remoto por Bluetooth	Smartphone Dependiente
Google Cardboard Google	$15	Smartphone	Smartphone	Smartphone	Smartphone	No Ajustable	Smartphone	Smartphone Dependiente

Computadora o Consola Dependiente

Son aquellos que necesitan interactuar con una computadora externa. Tienen una entrada HDMI y usan un conector USB para transmitir la información desde y hacia la computadora externa. En términos de fidelidad visual y frame rate los cascos cuyo procesamiento pasa por computadoras externas son la mejor opción. La gran desventaja de estos pasa por el alto costo no solo de los cascos sino también de una computadora que tenga la capacidad de procesamiento que requieren, además de que no son fácilmente trasladables ya que se necesita una computadora que cumpla con los requisitos mínimos. Ejemplos de estos cascos son el Oculus Rift, el Sony Playstation VR y el HTC Vive.

Para tener una idea de los costos, el precio actual del set de Oculus Rift que incluye el Rift y los controladores Touch Pad es de 500 USD, y los requisitos de hardware recomendados por Oculus en su página web son los siguientes:

- Tarjeta gráfica: NVIDIA GTX 1060/AMD Radeon RX 480 o superior
- Tarjeta gráfica alternativa: NVIDIA GTX 970/AMD Radeon R9 290 o superior
- CPU: Intel i5-4590 / AMD Ryzen 5 1500X o superior
- Memoria: 8 GB DE RAM O MÁS
- Salida de vídeo: Salida de vídeo HDMI 1.3 compatible
- Puertos USB: 3 puertos USB 3.0 más 1 puerto USB 2.0
- SO: Windows 7 SP1 64 bits o posterior

Las computadoras optimizadas para Oculus y recomendadas en su página web van desde 719 USD la IdeaCentre Y720-Cube, le sigue la IdeaCentre Y900 por 1500 USD ambas de Lenovo, hasta la X7 V6 de Aorus por 2999 USD.

La siguiente es una tabla de comparación entre el Oculus Rift y el Sony Playstation VR:

Tabla 2: Tabla de Comparación entre Cascos Consola Dependiente

Nombre	Costo Aprox. (USD)	Tipo Displa y	Resolución	FOV	Audio	Tasa de refrezco	Latencia	Optica	Input	Tipo
Sony Playstation VR Sony	$349.00	OLED	960x1080x2	10º	Auriculares externos incluidos en el Set	120Hz or 90Hz	<18 ms (Estimada)	Lentes fijas, Compatibles con los Lentes de los Usuarios	Control de PS4, Mando Pistola, Camara, Controles PSMove	Consola Dependiente
Oculus Rift Oculus	$500.00	OLED	1080x1200 x2	110°	Auriculares Incorporados al Casco	90Hz	<20 ms	Lentes ajustables	Sensores de movimiento, Controles Oculus Touch	Computadora Dependiente

Totalmente Autónomo

Son cascos que tienen su propio hardware interno (RAM, placa de video, procesador, pantalla) con características similares a un Smartphone. Su gran ventaja es que no requieren de nada externo. Aún no hay muchos en el mercado pero sí los hay en desarrollo. Un ejemplo de estos es el Auravisor, un proyecto kickstarter que ya está productivo y disponible para comprar por 400 USD. Las desventajas reportadas del Auravisor son el peso, problemas de compatibilidad con otros sistemas y problemas para enfocar las lentes.

La siguiente es una tabla de comparación entre 2 cascos totalmente autónomos:

Tabla 3: Tabla de Comparación entre Cascos Totalmente Autónomos

Nombre	Costo Aprox. (USD)	Tipo Display	Resolución	FOV	Audio	Tasa de refrezco	Optica	Input	Tipo	Estado
Auravisor Auravisor	$399.99	LCD	1920x1080	100º	Si	60Hz	Ajustable, no requiere lentes	Bluetooth, WiFi	All-In-One HMD	Terminado
Atheer Air Smart Glasses Atheer	TBA	LCD	1280x720	50º	Si	60Hz	Pantalla proyectada	USB-C, WiFi, Bluetooth, Gestos 3D	All-In-One HMD	En desarrollo

Elección

Se es consciente de que si bien los Smartphone actuales son capaces de proveer una experiencia virtual razonable, aún están por debajo de lo que pueden hacer las computadoras potentes y las consolas de videojuegos.

Por otro lado, hoy en día cualquier persona tiene un Smartphone y comprar un casco es realmente accesible. Es por esta razón que se eligió desarrollar el prototipo para Smartphone.

Como el Google Cardboard parece incómodo para hacer ejercicio ya que está hecho de cartón, y el Samsung Gear VR solo sirve para unos pocos celulares además de que el precio es alto, se decidió utilizar el VR Box que permite celulares de 4.7'' a 5.7'' y es fácilmente accesible en Argentina. De todas formas, el prototipo podrá utilizarse en cualquier Smartphone con Android o iOS y en cualquier Casco Smartphone Dependiente que sea adecuado para el Smartphone correspondiente y que sea de conveniencia para el usuario.

Por último, cabe aclarar que al realizar el desarrollo en Unity, se puede adaptar el software y portar el prototipo a otros tipos de cascos.

Opciones para detectar movimiento

Se analizaron varias formas de detectar el movimiento del usuario y de convertirlo a la velocidad en que se mueve el personaje, cada una con sus ventajas y desventajas:

Tomar datos de la bicicleta:

Es el más preciso de todos los métodos, consiste en conectar la bicicleta fija directamente al conector de audio o mediante Bluetooth y obtener directamente la velocidad. En caso que la bicicleta no otorgue la velocidad pero sí la distancia recorrida se puede calcular internamente con la siguiente ecuación (1):

$$v = \Delta d \; / \; \Delta t \qquad (1)$$

Donde v es la velocidad a calcular, Δd es la distancia recorrida y Δt es el tiempo transcurrido.

La gran desventaja es que la mayoría de las bicicletas no cuentan con integración directa con celulares, por lo que habría que analizar cada caso aislado siendo así inviable.

Sensor magnético externo:

Se requiere un microcontrolador, como por ejemplo una placa con Arduino, al que se le conecta un sensor magnético en la bicicleta para que por cada vuelta reciba una señal y la

transmita al celular. Midiendo el tiempo entre pulsos se puede calcular la velocidad, aunque no es tan preciso ya que desconocemos el radio en el que se colocó el receptor magnético. La gran ventaja es que es utilizable en cualquier bicicleta. El problema es que se requiere de hardware adicional, el cual hay que transportar e instalar en la bicicleta a utilizar, y esto ya excede el esfuerzo que esperamos del usuario.

Utilizar un control externo sin medir la velocidad:

Utilizar un control analógico para moverlo mientras se utiliza la bicicleta parece la peor de las opciones ya que se pierde la esencia de sentir que uno está andando en bicicleta y, por otro lado, tener que sostener un control no es cómodo para el usuario.

Sensores del Smartphone:

Los Smartphone tienen diferentes sensores que miden movimiento y orientación. Estos sensores son capaces de proveer información precisa que sirve para inferir movimientos o gestos tales como inclinar, sacudir o rotar el dispositivo.

A grandes rasgos hay dos tipos de sensores de movimiento: aquellos basados en hardware y los basados en software.

Los basados en hardware son aquellos construidos con componentes físicos dentro del dispositivo. Obtienen la información midiendo directamente propiedades ambientales específicas como la aceleración o cambios angulares. Algunos ejemplos son el giroscopio y el acelerómetro.

Los basados en software no son dispositivos físicos, pero utilizan uno o más de los sensores basados en hardware para obtener la información. Algunos ejemplos son la aceleración lineal o el sensor de gravedad.

Por otro lado, hay dos tipos de sensores según el movimiento que se desea monitorear: los sensores de movimientos y los sensores de posición.

Cuando el movimiento es un input directo del usuario (como un usuario manejando un auto o controlando una pelota en un juego moviendo el dispositivo móvil) se monitorea el movimiento relativo al marco de referencia del dispositivo. A los sensores que miden estos movimientos se los llama “Sensores de Movimiento”.

Cuando el movimiento es un reflejo del ambiente físico en el que reposa el dispositivo (por ejemplo, un dispositivo que viaja en un colectivo en el bolsillo de una persona) se monitorea el movimiento relativo al marco de referencia del mundo. A los sensores que miden estos movimientos se los llama “Sensores de Posición”.

Como se trata de una aplicación que correrá en un dispositivo móvil que utilizará una persona en una bicicleta fija dentro de un casco, los que interesan son los Sensores de Movimiento.

Por otro lado, se utilizaron los sensores basados en hardware y se adaptaron a las necesidades del proyecto ya que la existencia de los basados en software depende de cada dispositivo móvil y el sistema operativo que utiliza.

Elección

Si bien tomar los datos que arroja la bicicleta fija es lo más preciso para obtener la velocidad a la que va el usuario, seleccionar esta opción parece inviable de no limitar el mercado a las nuevas bicicletas fijas que vienen con integración a celular mediante Bluetooth, porque habría que interpretar qué bicicleta se está conectando y qué información arroja cada una.

En cuanto a precisión, el sensor magnético también es ideal, el gran problema es, una vez más, limitar el mercado. Los usuarios deberían comprarse el sensor magnético para poder utilizar la aplicación.

Por otro lado, el control externo viene con la mayoría de los cascos virtuales y es la opción más fácil de desarrollar, pero se considera que, por un lado, pueden incomodar al usuario en el entrenamiento, y por el otro, limitan la inmersión en el juego ya que el movimiento no es el que hace la persona durante el entrenamiento.

En conclusión, como la idea es ampliar el mercado lo más posible y que la experiencia sea real para el usuario, en el sentido que sienta que está andando en un bicicleta móvil en un mundo virtual, se cree que la mejor opción, por más que no sea la más precisa, es utilizar los sensores del celular para obtener el movimiento de la persona.

Cabe aclarar que en caso de querer utilizar otro tipo de cascos virtuales que no impliquen Smartphone es posible ya que todos los cascos virtuales tienen estos sensores.

Análisis

En función de los requerimientos definidos en el backlog, se realizaron las tarjetas de usuario del sistema, con el fin de satisfacer cada una de las funcionalidades requeridas. Estas tarjetas fueron clasificadas en 3 grupos con el objetivo de agruparlas y priorizarlas:

Tarjetas de usuario

Desplazamiento

Tabla 4: Historia de Usuario - Movimiento

Historia de Usuario	
Número: U.S.001	Nombre: Movimiento
Modificación de Historia Número: -	Iteración Asignada: 1
Prioridad en Negocio: Alta	
Riesgo en Desarrollo: Alto	Dependencia: -
Descripción: Como usuario quiero, al andar en una bicicleta fija, comenzar a moverme en un mundo virtual.	
Criterios de aceptación: Al pedalear en la bicicleta, el personaje dentro del mundo virtual, se mueve.	
Condiciones de completitud (son siempre los mismos, no cambia por historia): ● Pasó todos los test de regresión. ● Pasó todos los testeos por cada criterio de aceptación. ● Disponible para mostrar en demo.	

Tabla 5: Historia de Usuario - Rotación

Historia de Usuario	
Número: U.S.002	Nombre: Rotación
Modificación de Historia Número:-	Iteración Asignada: 1
Prioridad en Negocio: Alta	
Riesgo en Desarrollo: Alto	Dependencia:
Descripción: Como usuario quiero poder moverme en la dirección en que roto la cabeza.	
Criterios de aceptación: El personaje se debe mover en dirección hacia donde está apuntando la cabeza del usuario.	
Condiciones de completitud (son siempre los mismos, no cambia por historia): ● Pasó todos los test de regresión. ● Pasó todos los testeos por cada criterio de aceptación. ● Disponible para mostrar en demo.	

Tabla 6: Historia de Usuario – Inicio Entrenamiento

Historia de Usuario	
Número: U.S.003	Nombre: Inicio entrenamiento
Modificación de Historia Número:-	Iteración Asignada: 2
Prioridad en Negocio: Alta	
Riesgo en Desarrollo: Baja	Dependencia: 1, 2, 4

Descripción: Como usuario quiero, al seleccionar “Inicio”, comenzar a moverme libremente en un mundo virtual.
Criterios de aceptación: Integración de movimiento, rotación y presentación de escenario.
Condiciones de completitud (son siempre los mismos, no cambia por historia): ● Pasó todos los test de regresión. ● Pasó todos los testeos por cada criterio de aceptación. ● Disponible para mostrar en demo.

Navegabilidad

Tabla 7: Historia de Usuario – Menú Principal

Historia de Usuario	
Número: U.S.004	Nombre: Menú Principal
Modificación de Historia Número:-	Iteración Asignada: 2
Prioridad en Negocio: Media	
Riesgo en Desarrollo: Bajo	Dependencia:
Descripción: Como usuario quiero poder ver todas las opciones disponibles en un Menú: Inicio Configuración ● Música o Silenciar ● Dirección Fija o Variable por Rotación ● Dificultad (Fácil, Moderado, Difícil)	

• Duración (por Tiempo o por Distancia) • Sensibilidad (0, 1, 2, 3, 4) • Escenarios (bosque o playa) Entrenamientos Anteriores
Criterios de aceptación: Todas las opciones mencionadas deben verse en el Menú Principal.
Condiciones de completitud (son siempre los mismos, no cambia por historia): • Pasó todos los test de regresión. • Pasó todos los testeos por cada criterio de aceptación. • Disponible para mostrar en demo.

Tabla 8: Historia de Usuario – Selección opciones

Historia de Usuario	
Número: U.S.005	Nombre: Selección opciones
Modificación de Historia Número:-	Iteración Asignada: 2
Prioridad en Negocio: Media	
Riesgo en Desarrollo: Bajo	Dependencia: 4
Descripción: Como usuario quiero poder seleccionar la opción deseada dentro del Menú fijando la vista durante 1.5 segundos sobre el ítem correspondiente.	
Criterios de aceptación: Dentro de cada menú, al fijar la vista en una de las opciones, se debe visualizar una barra de progreso y al completar los 1.5 segundos, debe completarse la acción correspondiente a	

la opción seleccionada.
Condiciones de completitud (son siempre los mismos, no cambia por historia): ● Pasó todos los test de regresión. ● Pasó todos los testeos por cada criterio de aceptación. ● Disponible para mostrar en demo.

Tabla 9: Historia de Usuario - Pausa

Historia de Usuario	
Número: U.S.006	Nombre: Pausa
Modificación de Historia Número:-	Iteración Asignada: 2
Prioridad en Negocio: Alta	
Riesgo en Desarrollo: Alto	Dependencia: 1, 2, 3
Descripción: Como usuario quiero poder pausar el entrenamiento luego de quedarme 5 segundos quieto.	
Criterios de aceptación: Durante el entrenamiento, luego de 5 segundos de inactividad, este se debe pausar.	
Condiciones de completitud (son siempre los mismos, no cambia por historia): ● Pasó todos los test de regresión. ● Pasó todos los testeos por cada criterio de aceptación. ● Disponible para mostrar en demo.	

Tabla 10: Historia de Usuario – Menú Pausa

Historia de Usuario	
Número: U.S.007	Nombre: Menú Pausa
Modificación de Historia Número:-	Iteración Asignada: 2
Prioridad en Negocio: Baja	
Riesgo en Desarrollo: Bajo	Dependencia: 6
Descripción: Como usuario quiero poder Salir o Reanudar el Entrenamiento cuando está en Pausa.	
Criterios de aceptación: Una vez pausado el entrenamiento, se debe mostrar un Menú con las opciones Salir o Reanudar.	
Condiciones de completitud (son siempre los mismos, no cambia por historia): ● Pasó todos los test de regresión. ● Pasó todos los testeos por cada criterio de aceptación. ● Disponible para mostrar en demo.	

Tabla 11: Historia de Usuario – Selección Salir

Historia de Usuario	
Número: U.S.008	Nombre: Selección Salir
Modificación de Historia Número:-	Iteración Asignada: 2
Prioridad en Negocio: Media	

Riesgo en Desarrollo: Bajo	Dependencia: 6 y 7
Descripción: Como usuario quiero que, al seleccionar Salir en el menú de Pausa, finalice el entrenamiento y me muestre todas las opciones disponibles en el Menú Inicial.	
Criterios de aceptación: Una vez pausado el entrenamiento, habiendo seleccionado Salir, debe finalizar el entrenamiento mostrando todas las opciones iniciales de cuando se abre la aplicación.	
Condiciones de completitud (son siempre los mismos, no cambia por historia): ● Pasó todos los test de regresión. ● Pasó todos los testeos por cada criterio de aceptación. ● Disponible para mostrar en demo.	

Tabla 12: Historia de Usuario – Selección Reanudar

Historia de Usuario	
Número: U.S.009	Nombre: Selección Reanudar
Modificación de Historia Número:-	Iteración Asignada: 2
Prioridad en Negocio: Media	
Riesgo en Desarrollo: Bajo	Dependencia: 6 y 7
Descripción: Como usuario quiero que, al seleccionar Reanudar en el menú de Pausa, continúe el entrenamiento como había quedado antes de pausar.	
Criterios de aceptación: Una vez pausado el entrenamiento, habiendo seleccionado Reanudar, debe continuar el	

entrenamiento, mostrando todos los contadores tal y como estaban antes de la pausa.
Condiciones de completitud (son siempre los mismos, no cambia por historia): ● Pasó todos los test de regresión. ● Pasó todos los testeos por cada criterio de aceptación. ● Disponible para mostrar en demo.

Extras

Tabla 13: Historia de Usuario – Fijar Dirección

Historia de Usuario	
Número: U.S.010	Nombre: Fijar dirección
Modificación de Historia Número:-	Iteración Asignada: 3
Prioridad en Negocio: Media	
Riesgo en Desarrollo: Medio	Dependencia:1, 3, 4
Descripción: Como usuario quiero, al seleccionar “Dirección Fija”, moverme en forma recta dentro del mundo virtual y poder rotar la cabeza para ver los alrededores sin cambiar de dirección.	
Criterios de aceptación: El personaje se debe mover en línea recta. Al mover la cabeza, se deben visualizar los al redores. Al mover la cabeza, no se debe cambiar la dirección.	
Condiciones de completitud (son siempre los mismos, no cambia por historia): ● Pasó todos los test de regresión. ● Pasó todos los testeos por cada criterio de aceptación. ● Disponible para mostrar en demo.	

Tabla 14: Historia de Usuario – Visualización distancia y tiempo

Historia de Usuario	
Número: U.S.011	Nombre: Visualización distancia y tiempo
Modificación de Historia Número:-	Iteración Asignada: 3
Prioridad en Negocio: Baja	
Riesgo en Desarrollo: Medio	Dependencia: 1, 3
Descripción: Como usuario quiero poder visualizar constantemente cuánta distancia he recorrido y cuánto tiempo ha transcurrido desde el Inicio del entrenamiento.	
Criterios de aceptación: Durante un entrenamiento, se deberán visualizar dos indicadores con la información de distancia y tiempo.	
Condiciones de completitud (son siempre los mismos, no cambia por historia): ● Pasó todos los test de regresión. ● Pasó todos los testeos por cada criterio de aceptación. ● Disponible para mostrar en demo.	

Tabla 15: Historia de Usuario – Selección Dificultad

Historia de Usuario	
Número: U.S.012	Nombre: Selección dificultad
Modificación de Historia Número:-	Iteración Asignada: 3

Prioridad en Negocio: Baja	
Riesgo en Desarrollo: Alto	Dependencia: 4
Descripción: Como usuario quiero poder seleccionar la dificultad del entrenamiento entre baja, media y alta, lo que conlleva a una mayor duración del entrenamiento ya sea por tiempo o por distancia.	
Criterios de aceptación: Estando en duración por tiempo y dificultad baja, el entrenamiento debe durar 10 minutos. Estando en duración por tiempo y dificultad media, el entrenamiento debe durar 20 minutos. Estando en duración por tiempo y dificultad alta, el entrenamiento debe durar 30 minutos. Estando en duración por distancia y dificultad baja, el entrenamiento debe durar 1000 metros. Estando en duración por distancia y dificultad media, el entrenamiento debe durar 2000 metros. Estando en duración por distancia y dificultad alta, el entrenamiento debe durar 4000 metros.	
Condiciones de completitud (son siempre los mismos, no cambia por historia): ● Pasó todos los test de regresión. ● Pasó todos los testeos por cada criterio de aceptación. ● Disponible para mostrar en demo.	

Tabla 16: Historia de Usuario – Selección Duración

Historia de Usuario	
Número: U.S.013	Nombre: Selección duración

Modificación de Historia Número:-	Iteración Asignada: 3
Prioridad en Negocio: Media	
Riesgo en Desarrollo: Medio	Dependencia: 4
Descripción: Como usuario quiero poder seleccionar la duración del entrenamiento en tiempo o en distancia.	
Criterios de aceptación: Al seleccionar por tiempo, el contador de tiempo debe mostrarse en cuenta regresiva mientras que el contador por metros debe mostrarse en cuenta progresiva. Y el entrenamiento debe finalizar cuando el contador de tiempo llegue a 0. Al seleccionar por distancia, el contador de tiempo debe mostrarse en cuenta progresiva mientras que el contador por metros debe mostrarse en cuenta regresiva. Y el entrenamiento debe finalizar cuando el contador de distancia llegue a 0.	
Condiciones de completitud (son siempre los mismos, no cambia por historia): ● Pasó todos los test de regresión. ● Pasó todos los testeos por cada criterio de aceptación. ● Disponible para mostrar en demo.	

Tabla 17: Historia de Usuario - Sensibilidad

Historia de Usuario	
Número: U.S.014	Nombre: Selección Sensibilidad
Modificación de Historia Número:-	Iteración Asignada: 3
Prioridad en Negocio: Baja	

Riesgo en Desarrollo: Alto	Dependencia:
Descripción: Como usuario quiero poder ajustar la sensibilidad del dispositivo para que requiera mayor o menor esfuerzo del entrenamiento para captar el movimiento.	
Criterios de aceptación:	
Condiciones de completitud (son siempre los mismos, no cambia por historia): ● Pasó todos los test de regresión. ● Pasó todos los testeos por cada criterio de aceptación. ● Disponible para mostrar en demo.	

Tabla 18: Historia de Usuario – Selección Escenarios

Historia de Usuario	
Número: U.S.015	Nombre: Selección Escenarios
Modificación de Historia Número:-	Iteración Asignada: 4
Prioridad en Negocio: Media	
Riesgo en Desarrollo: Medio	Dependencia:
Descripción: Como usuario quiero poder elegir entre diferentes escenarios.	
Criterios de aceptación: En la selección de escenarios debe haber como mínimo 2 escenarios distintos. Al seleccionar uno u otro, se debe notar la diferencia del entorno.	

Condiciones de completitud (son siempre los mismos, no cambia por historia): ● Pasó todos los test de regresión. ● Pasó todos los testeos por cada criterio de aceptación. ● Disponible para mostrar en demo.

Tabla 19: Historia de Usuario – Música ambiental

Historia de Usuario	
Número: U.S.016	Nombre: Música ambiental
Modificación de Historia Número:-	Iteración Asignada: 4
Prioridad en Negocio: Media	
Riesgo en Desarrollo: Medio	Dependencia:
Descripción: Como usuario quiero escuchar música ambiental dentro de la aplicación.	
Criterios de aceptación: Con la opción “Sonido” en modo ON, se debe escuchar música de fondo.	
Condiciones de completitud (son siempre los mismos, no cambia por historia): ● Pasó todos los test de regresión. ● Pasó todos los testeos por cada criterio de aceptación. ● Disponible para mostrar en demo.	

Tabla 20: Historia de Usuario - Silenciar

Historia de Usuario

Número: U.S.017	Nombre: Silenciar
Modificación de Historia Número:-	Iteración Asignada: 4
Prioridad en Negocio: Media	
Riesgo en Desarrollo: Medio	Dependencia:
Descripción: Como usuario quiero poder silenciar la aplicación, y utilizar otras aplicaciones que poseo para escuchar diferentes cosas mientras entreno.	
Criterios de aceptación: Con la opción “Sonido” en modo OFF, no se debe escuchar ningún sonido ni música emitido por la aplicación.	
Condiciones de completitud (son siempre los mismos, no cambia por historia): ● Pasó todos los test de regresión. ● Pasó todos los testeos por cada criterio de aceptación. ● Disponible para mostrar en demo.	

Tabla 21: Historia de Usuario – Visualización Entrenamientos

Historia de Usuario	
Número: U.S.018	Nombre: Visualización Entrenamientos
Modificación de Historia Número:-	Iteración Asignada: 4
Prioridad en Negocio: Media	
Riesgo en Desarrollo: Medio	Dependencia:
Descripción:	

Como usuario quiero poder visualizar los entrenamientos realizados con anterioridad. Esto es, cuántos metros he recorrido, en cuánto tiempo y con qué dificultad.
Criterios de aceptación: Cuando no se está realizando un entrenamiento, debe estar disponible una tabla con la información de al menos los últimos 5 entrenamientos.
Condiciones de completitud (son siempre los mismos, no cambia por historia): ● Pasó todos los test de regresión. ● Pasó todos los testeos por cada criterio de aceptación. ● Disponible para mostrar en demo.

Tabla 22: Historia de Usuario – Persistencia Opciones

Historia de Usuario	
Número: U.S.019	Nombre: Persistencia Opciones
Modificación de Historia Número:-	Iteración Asignada: 4
Prioridad en Negocio: Media	
Riesgo en Desarrollo: Medio	Dependencia:
Descripción: Como usuario quiero que se mantengan las opciones seleccionadas cuando cierro y abro la aplicación.	
Criterios de aceptación: Se seleccionará un set de opciones, se cerrará la aplicación y al abrirse nuevamente, se debe mostrar el set seleccionado.	
Condiciones de completitud (son siempre los mismos, no cambia por historia):	

- Pasó todos los test de regresión.
- Pasó todos los testeos por cada criterio de aceptación.
- Disponible para mostrar en demo.

Casos de Prueba

Desplazamiento

Tabla 23: Caso de Prueba - Desplazamiento

Caso de Prueba	
Número Caso de Prueba: C.P.001	Número Historia de Usuario: U.S.001
Nombre Caso de Prueba: Desplazamiento	
Descripción: Se probará que al comenzar a pedalear en la bicicleta fija, el personaje se comienza a mover en línea recta.	
Condiciones de ejecución: Utilizando el casco de realidad virtual sobre la bicicleta fija	
Entradas: Pedaleo constante	
Resultado esperado: El personaje se desplaza a velocidad constante por el entorno	
Evaluación:	

Tabla 24: Caso de Prueba - Desplazamiento

Caso de Prueba	
Número Caso de Prueba: C.P.002	Número Historia de Usuario: U.S.001
Nombre Caso de Prueba: Desplazamiento	

Descripción: Se probará que al pedalear en la bicicleta fija de forma acelerada, el personaje en el mundo virtual tendrá un movimiento acelerado.
Condiciones de ejecución: Utilizando el casco de realidad virtual sobre la bicicleta fija
Entradas: Pedalear cada vez con más frecuencia.
Resultado esperado: El personaje acelera su velocidad hasta un límite.
Evaluación:

Tabla 25: Caso de Prueba - Desplazamiento

Caso de Prueba	
Número Caso de Prueba: C.P.003	Número Historia de Usuario: U.S.001
Nombre Caso de Prueba: Desplazamiento	
Descripción: Se probará que al estar pedaleando en la bicicleta fija, si se deja de pedalear el personaje comienza a desacelerar hasta frenar.	
Condiciones de ejecución: Utilizando el casco de realidad virtual sobre la bicicleta fija	
Entradas: Dejar de pedalear estando en movimiento dentro de la realidad virtual	
Resultado esperado: El personaje desacelera su velocidad hasta frenar completamente	
Evaluación:	

Tabla 26: Caso de Prueba - Rotación

Caso de Prueba	
Número Caso de Prueba: C.P.004	Número Historia de Usuario: U.S.002
Nombre Caso de Prueba: Rotación	
Descripción: Se probará que al rotar la cabeza de forma horizontal 90º el personaje en el mundo virtual rote de igual manera	
Condiciones de ejecución: Utilizando el casco de realidad virtual sobre la bicicleta fija	
Entradas: Rotar la cabeza de forma horizontal de izquierda a derecha con un ángulo de 90º	
Resultado esperado: La visual en el mundo virtual realiza el mismo paneo que la cabeza del usuario	
Evaluación:	

Tabla 27: Caso de Prueba - Rotación

Caso de Prueba	
Número Caso de Prueba: C.P.005	Número Historia de Usuario: U.S.002
Nombre Caso de Prueba: Rotación	
Descripción: Se probará que al rotar la cabeza de forma vertical 90º el personaje en el mundo virtual rote de igual manera	
Condiciones de ejecución: Utilizando el casco de realidad virtual sobre la bicicleta fija	

Entradas: Rotar la cabeza de forma vertical de abajo hacia arriba con un ángulo de 90º
Resultado esperado: La visual en el mundo virtual realiza el mismo paneo que la cabeza del usuario
Evaluación:

Tabla 28: Caso de Prueba – Inicio Entrenamiento

Caso de Prueba	
Número Caso de Prueba: C.P.006	Número Historia de Usuario: U.S.003
Nombre Caso de Prueba: Inicio entrenamiento	
Descripción: Se probará que al fijar la vista por 1.5 segundos en el botón de inicio, comience un entrenamiento en un escenario virtual.	
Condiciones de ejecución: Utilizando el casco de realidad virtual	
Entradas: Fijar la vista en el botón de inicio.	
Resultado esperado: Se muestra una barra de progreso que al completarse comienza la carga del mundo virtual.	
Evaluación:	

Navegabilidad

Tabla 29: Caso de Prueba – Menú Principal

Caso de Prueba	
Número Caso de Prueba: C.P.007	Número Historia de Usuario: U.S.004

Nombre Caso de Prueba: Menú Principal
Descripción: Se probará que al ingresar a la aplicación se puedan visualizar todas las opciones de configuración.
Condiciones de ejecución: Utilizando el casco de realidad virtual
Entradas: Ingresar a la aplicación.
Resultado esperado: Se muestran todas las opciones de configuración en el inicio ● Música o Silenciar ● Dirección Fija o Variable por Rotación ● Dificultar ● Duración ● Calibrar ● Escenarios ● Entrenamientos Anteriores
Evaluación:

Tabla 30: Caso de Prueba – Selección opciones

Caso de Prueba	
Número Caso de Prueba: C.P.008	Número Historia de Usuario: U.S.005
Nombre Caso de Prueba: Selección opciones	
Descripción: Se probará que al fijar la vista por 1.5 segundos en cualquier botón de menú principal me permite interactuar con él.	

Condiciones de ejecución: Utilizando el casco de realidad virtual.
Entradas: Fijar la vista en cualquier botón del menú principal.
Resultado esperado: Se muestra una barra de progreso que al completarse ejecuta la función del botón.
Evaluación:

Tabla 31: Caso de Prueba – Visualización Entrenamientos

Caso de Prueba	
Número Caso de Prueba: C.P.009	Número Historia de Usuario: U.S.018
Nombre Caso de Prueba: Visualización entrenamientos	
Descripción: Se probará que al ingresar a la aplicación se puedan visualizar los últimos 5 entrenamientos.	
Condiciones de ejecución: Utilizando el casco de realidad virtual.	
Entradas: Ingresar a la aplicación y buscar la tabla de “últimos entrenamientos”.	
Resultado esperado: Al ingresar a la aplicación se debe poder visualizar los últimos entrenamientos viendo su duración, metros recorridos, dificultad y hora.	
Evaluación:	

Tabla 32: Caso de Prueba – Pausa

Caso de Prueba

Número Caso de Prueba: C.P.010	Número Historia de Usuario: U.S.006
Nombre Caso de Prueba: Pausa	
Descripción: Se probará que al quedarse quieto durante cinco segundos durante el entrenamiento, se pausara el entrenamiento.	
Condiciones de ejecución: Utilizando el casco de realidad virtual dentro de un entrenamiento.	
Entradas: No moverse durante tres segundos durante un entrenamiento.	
Resultado esperado: Al quedarse quieto durante un entrenamiento por 5 segundos este se pausa.	
Evaluación:	

Tabla 33: Caso de Prueba – Menú Pausa

Caso de Prueba	
Número Caso de Prueba: C.P.011	Número Historia de Usuario: U.S.007
Nombre Caso de Prueba: Menú Pausa	
Descripción: Se probará que al pausar el entrenamiento, se visualiza el Menú de Pausa.	
Condiciones de ejecución: Utilizando el casco de realidad virtual dentro de un entrenamiento.	
Entradas: Entrenamiento pausado.	
Resultado esperado: Se visualiza el Menú de Pausa con 2 opciones: salir o reanudar.	

Evaluación:

Tabla 34: Caso de Prueba – Salir

Caso de Prueba	
Número Caso de Prueba: C.P.012	Número Historia de Usuario: U.S.008
Nombre Caso de Prueba: Salir	
Descripción: Se probará que al fijar la vista por 1.5 segundos en el botón de salir en el menú de pausa se vuelve al menú principal.	
Condiciones de ejecución: Utilizando el casco de realidad virtual dentro de un entrenamiento pausado.	
Entradas: Fijar la vista en el botón de reanudar.	
Resultado esperado: Al fijar la vista en el botón de salir en el menú de pausa por 1.5 segundos este vuelva al menú principal cancelando el entrenamiento.	
Evaluación:	

Tabla 35: Caso de Prueba – Pausa

Caso de Prueba	
Número Caso de Prueba: C.P.013	Número Historia de Usuario: U.S.009
Nombre Caso de Prueba: Pausa	
Descripción: Se probará que al fijar la vista por 1.5 segundos en el botón de reanudar en el	

menú de pausa se continúa con el entrenamiento.
Condiciones de ejecución: Utilizando el casco de realidad virtual dentro de un entrenamiento pausado.
Entradas: Fijar la vista en el botón de reanudar.
Resultado esperado: Al fijar la vista en el botón de reanudar en el menú de pausa por 1.5 segundos este reanuda el entrenamiento.
Evaluación:

Tabla 36: Caso de Prueba – Selección Duración

Caso de Prueba	
Número Caso de Prueba: C.P.014	Número Historia de Usuario: U.S.013
Nombre Caso de Prueba: Selección Duración	
Descripción: Se probará que al fijar la vista por 1.5 segundos en el botón de duración por distancia, se establezca como seleccionada.	
Condiciones de ejecución: Utilizando el casco de realidad virtual y duración por distancia.	
Entradas: Fijar la vista en el botón de duración por distancia.	
Resultado esperado: Al cambiar de modo a duración por distancia se espera que el entrenamiento finalice al llegar a la cantidad de metros preseleccionada.	
Evaluación:	

Tabla 37: Caso de Prueba – Selección Duración

Caso de Prueba	
Número Caso de Prueba: C.P.015	Número Historia de Usuario: U.S.013
Nombre Caso de Prueba: Selección duración	
Descripción: Se probará que al fijar la vista por 1.5 segundos en el botón de duración por tiempo, se establezca como seleccionada.	
Condiciones de ejecución: Utilizando el casco de realidad virtual y duración por distancia.	
Entradas: Fijar la vista en el botón de duración por tiempo.	
Resultado esperado: Al cambiar de modo a duración por tiempo se espera que el entrenamiento finalice al llegar a la cantidad de minutos preseleccionada.	
Evaluación:	

Tabla 38: Caso de Prueba – Selección Sensibilidad

Caso de Prueba	
Número Caso de Prueba: C.P.016	Número Historia de Usuario: U.S.014
Nombre Caso de Prueba: Selección Sensibilidad	
Descripción: Se probará que al fijar la vista por 1.5 segundos en el botón de Sensibilidad este cambie de valor al siguiente.	
Condiciones de ejecución: Utilizando el casco de realidad virtual.	

Entradas: Fijar la vista en el botón de Sensibilidad.
Resultado esperado: Al cambiar el valor de Sensibilidad a un mayor número se espera que con menos movimiento de la bicicleta fija se genera más movimiento en el mundo de realidad virtual. Estos valores van desde 0 (menos sensible) a 4 (más sensible).
Evaluación:

Tabla 39: Caso de Prueba – Rotación

Caso de Prueba	
Número Caso de Prueba: C.P.021	Número Historia de Usuario: U.S.002
Nombre Caso de Prueba: Rotación	
Descripción: Se probará que al fijar la vista por 1.5 segundos en el botón de dirección este cambie su estado de dirección fija a dirección por rotación.	
Condiciones de ejecución: Utilizando el casco de realidad virtual y la dirección fija.	
Entradas: Fijar la vista en el botón de dirección.	
Resultado esperado: Al cambiar de modo se espera que al rotar la cabeza el personaje rote siguiendo su ángulo de rotación.	
Evaluación:	

Tabla 40: Caso de Prueba – Fijar Dirección

Caso de Prueba

Número Caso de Prueba: C.P.022	Número Historia de Usuario: U.S.010
Nombre Caso de Prueba: Fijar dirección	
Descripción: Se probará que al fijar la vista por 1.5 segundos en el botón de dirección este cambie su estado de dirección por rotación, a dirección fija.	
Condiciones de ejecución: Utilizando el casco de realidad virtual y la dirección variable por rotación.	
Entradas: Fijar la vista en el botón de dirección.	
Resultado esperado: Al cambiar de modo se espera que al rotar la cabeza vea el entorno sin modificar la dirección.	
Evaluación:	

Tabla 41: Caso de Prueba – Selección Dificultad

Caso de Prueba	
Número Caso de Prueba: C.P.023	Número Historia de Usuario: U.S.012
Nombre Caso de Prueba: Selección dificultad	
Descripción: Se probará que al fijar la vista por 1.5 segundos en el botón de dificultad media, se establezca como seleccionada.	
Condiciones de ejecución: Utilizando el casco de realidad virtual y en una dificultad baja.	
Entradas: Fijar la vista en el botón de dificultad media.	
Resultado esperado: Al cambiar de modo a dificultad media se espera que el entrenamiento	

dure 20 minutos o 2000 metros (dependiendo si esta seleccionada la Duración por Tiempo o por Distancia).
Evaluación:

Tabla 42: Caso de Prueba – Selección dificultad

Caso de Prueba	
Número Caso de Prueba: C.P.023	Número Historia de Usuario: U.S.012
Nombre Caso de Prueba: Selección dificultad	
Descripción: Se probará que al fijar la vista por 1.5 segundos en el botón de dificultad alta, se establezca como seleccionada.	
Condiciones de ejecución: Utilizando el casco de realidad virtual y en una dificultad media.	
Entradas: Fijar la vista en el botón de dificultad alta.	
Resultado esperado: Al cambiar de modo a dificultad media se espera que el entrenamiento dure 30 minutos o 4000 metros (dependiendo si esta seleccionada la Duración por Tiempo o por Distancia).	
Evaluación:	

Tabla 43: Caso de Prueba – Selección dificultad

Caso de Prueba	
Número Caso de Prueba: C.P.024	Número Historia de Usuario: U.S.012

Nombre Caso de Prueba: Selección dificultad
Descripción: Se probará que al fijar la vista por 1.5 segundos en el botón de dificultad baja, se establezca como seleccionada.
Condiciones de ejecución: Utilizando el casco de realidad virtual y en una dificultad alta.
Entradas: Fijar la vista en el botón de dificultad baja.
Resultado esperado: Al cambiar de modo a dificultad baja se espera que el entrenamiento dure 10 minutos o 1000 metros (dependiendo si esta seleccionada la Duración por Tiempo o por Distancia).
Evaluación:

Tabla 44: Caso de Prueba – Selección Escenarios

Caso de Prueba	
Número Caso de Prueba: C.P.026	Número Historia de Usuario: U.S.015
Nombre Caso de Prueba: Selección Escenarios	
Descripción: Se probará que al fijar la vista por 1.5 segundos en el botón de escenario, se establezca el escenario seleccionado.	
Condiciones de ejecución: Utilizando el casco de realidad virtual.	
Entradas: Fijar la vista en el botón de escenario.	
Resultado esperado: Al cambiar de escenario se espera que el entrenamiento se realice en otro espacio virtual que el anterior. El cambio se realizará entre playa y bosque.	

Evaluación:

Tabla 45: Caso de Prueba – Música Ambiental

Caso de Prueba	
Número Caso de Prueba: C.P.028	Número Historia de Usuario: U.S.0016
Nombre Caso de Prueba: Música ambiental	
Descripción: Se probará que al fijar la vista por 1.5 segundos en el botón de música silenciada este cambie su estado actual al de activa y se escuche música de fondo.	
Condiciones de ejecución: Utilizando el casco de realidad virtual con música silenciada	
Entradas: Fijar la vista en el botón de música silenciada.	
Resultado esperado: Se muestra una barra de progreso que al completarse se activa la música de la aplicación.	
Evaluación:	

Tabla 46: Caso de Prueba – Silenciar

Caso de Prueba	
Número Caso de Prueba: C.P.029	Número Historia de Usuario: U.S.0017
Nombre Caso de Prueba: Silenciar	
Descripción: Se probará que al fijar la vista por 1.5 segundos en el botón de música este cambie su estado actual el de silenciado.	

Condiciones de ejecución: Utilizando el casco de realidad virtual con música activa
Entradas: Fijar la vista en el botón de música.
Resultado esperado: Se muestra una barra de progreso que al completarse se silencia la aplicación.
Evaluación:

Tabla 47: Caso de Prueba – Persistencia Opciones

Caso de Prueba	
Número Caso de Prueba: C.P.030	Número Historia de Usuario: U.S.019
Nombre Caso de Prueba: Persistencia Opciones	
Descripción: Se probará que al cerrar la aplicación todas las opciones fueron persistidas.	
Condiciones de ejecución: Utilizando el casco de realidad virtual.	
Entradas: Ingresar a la aplicación.	
Resultado esperado: Al ingresar a la aplicación se deben ver todos los cambios realizados en la configuración como fueron seleccionados previamente.	
Evaluación:	

Tabla 48: Caso de Prueba – Visualización distancia y tiempo

Caso de Prueba

Número Caso de Prueba: C.P.031	Número Historia de Usuario: U.S.011
Nombre Caso de Prueba: Visualización distancia y tiempo.	
Descripción: Se probará que al avanzar en el escenario, en caso de estar seleccionada la Duración por Distancia, el contador de metros disminuye hasta 0 y el contador de tiempo aumente sin límite.	
Condiciones de ejecución: Utilizando el casco de realidad virtual sobre la bicicleta fija dentro de un entrenamiento.	
Entradas: Duración por Distancia, pedalear en la bicicleta fija durante un entrenamiento.	
Resultado esperado: Al avanzar, el acumulador de metros en pantalla disminuye y el de tiempo aumenta, en relación a lo pedaleado.	
Evaluación:	

Tabla 49: Caso de Prueba – Visualización distancia y tiempo

Caso de Prueba	
Número Caso de Prueba: C.P.032	Número Historia de Usuario: U.S.011
Nombre Caso de Prueba: Visualización distancia y tiempo.	
Descripción: Se probará que al avanzar en el escenario, en caso de estar seleccionada la Duración por Tiempo, el contador de metros aumente sin límite y el contador de tiempo disminuye hasta 0.	
Condiciones de ejecución: Utilizando el casco de realidad virtual sobre la bicicleta fija dentro de un entrenamiento.	

Entradas: Pedalear en la bicicleta fija durante un entrenamiento.
Resultado esperado: Al avanzar, el acumulador de metros en pantalla aumenta y el de tiempo disminuye, en relación a lo pedaleado.
Evaluación:

Diseño Funcional

Interfaz del usuario

Para seleccionar las opciones, el usuario debe fijar la vista durante 1.5 segundos en la deseada y podrá ver cómo se completa una barra.

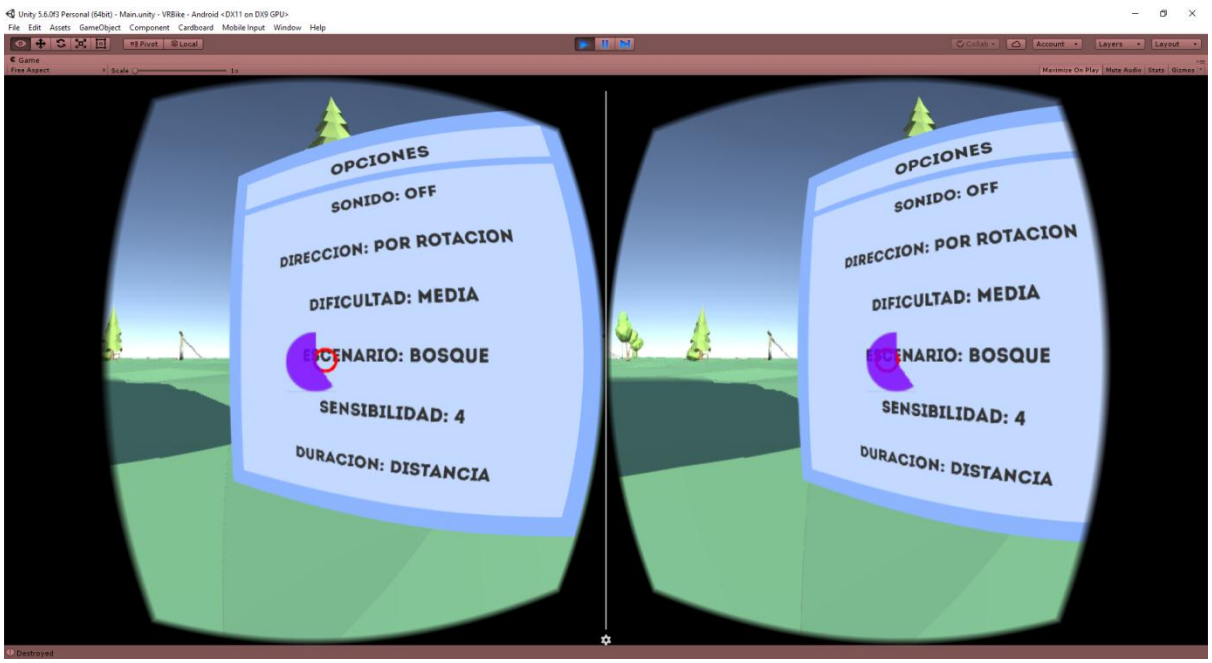

Figura 2: Pantalla Selección Opción

Pantalla Menú

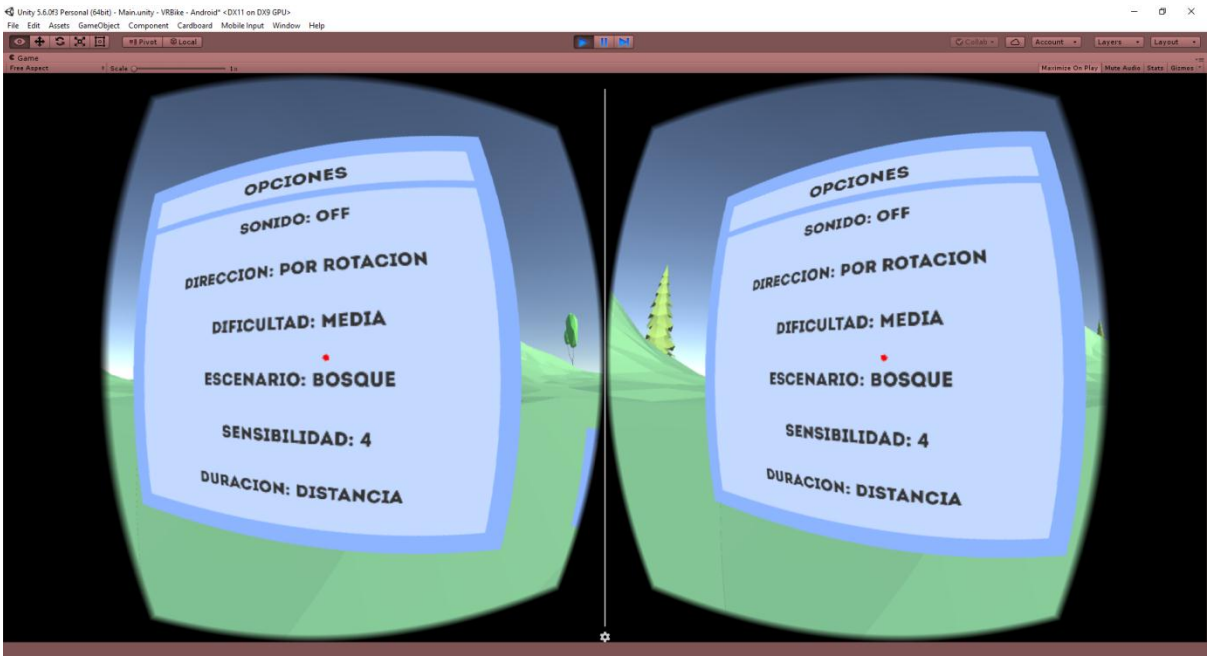

Figura 3: Pantalla Menú

Pantalla Botón Inicio

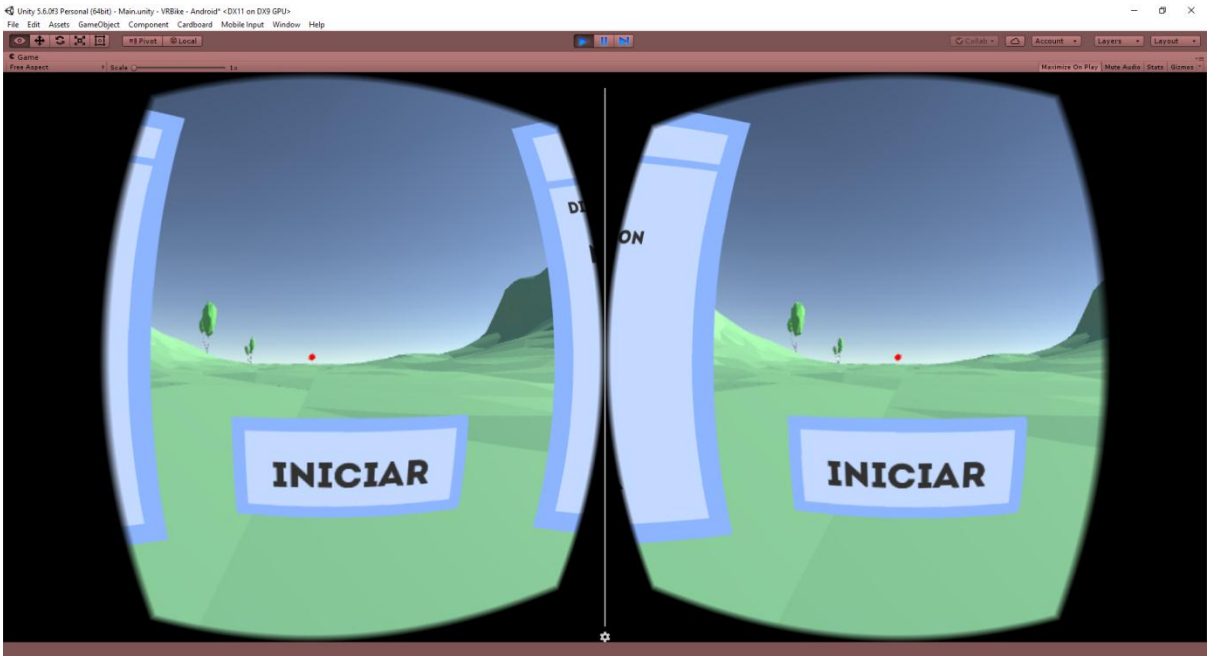

Figura 4: Pantalla Botón Inicio

Pantalla Entrenamientos Anteriores

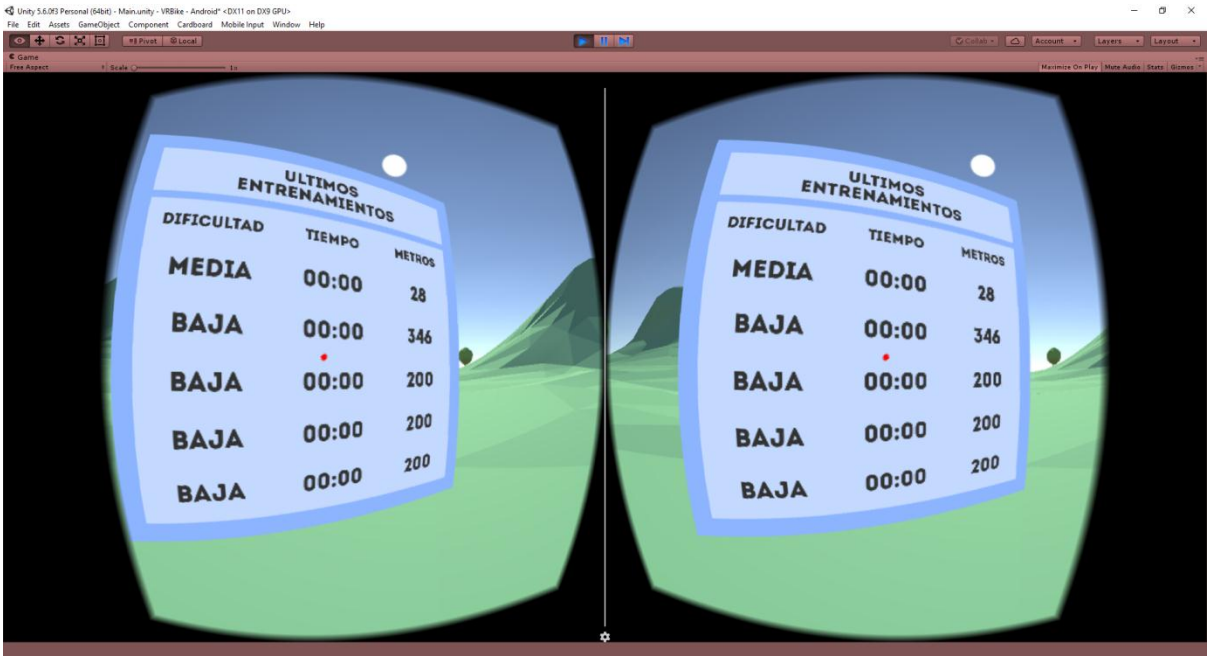

Figura 5: Pantalla Entrenamientos Anteriores

Pantallas escenarios

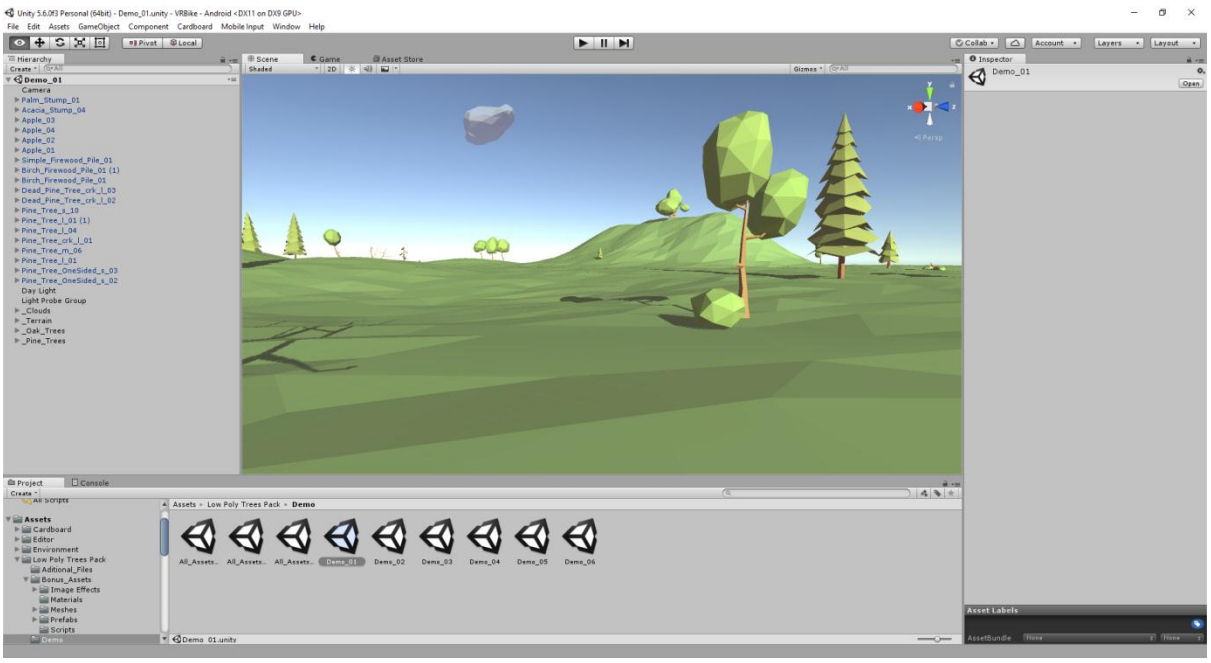

Figura 6: Pantalla Escenarios Bosque

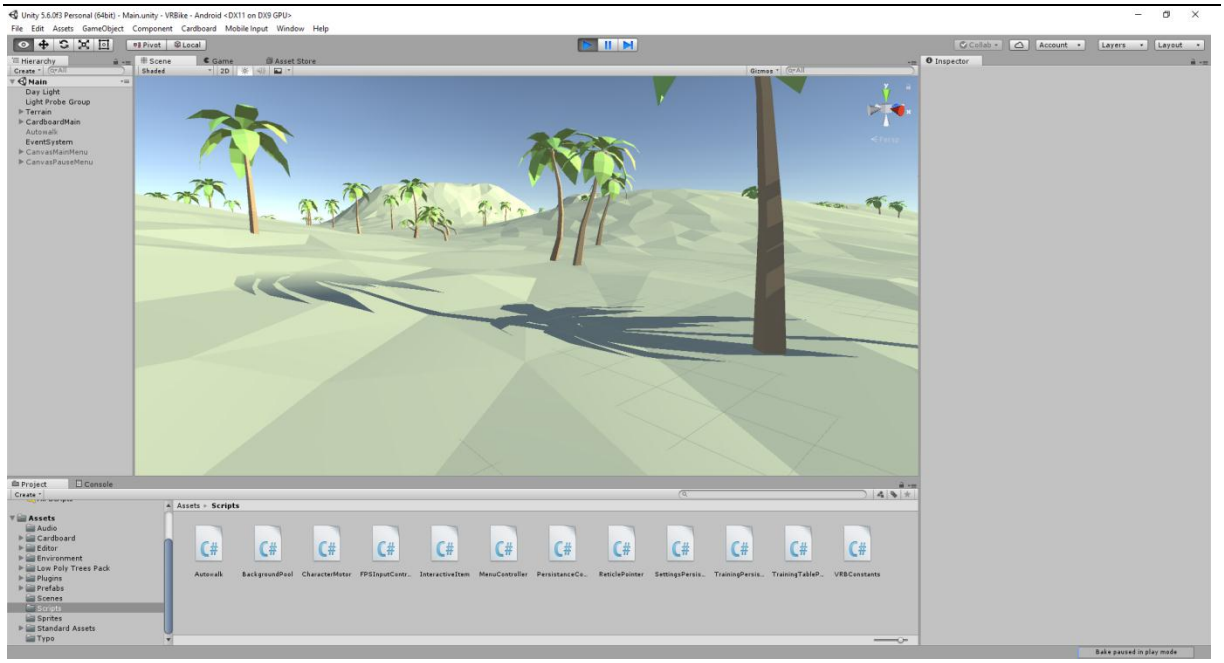

Figura 7: Pantalla Escenario Playa

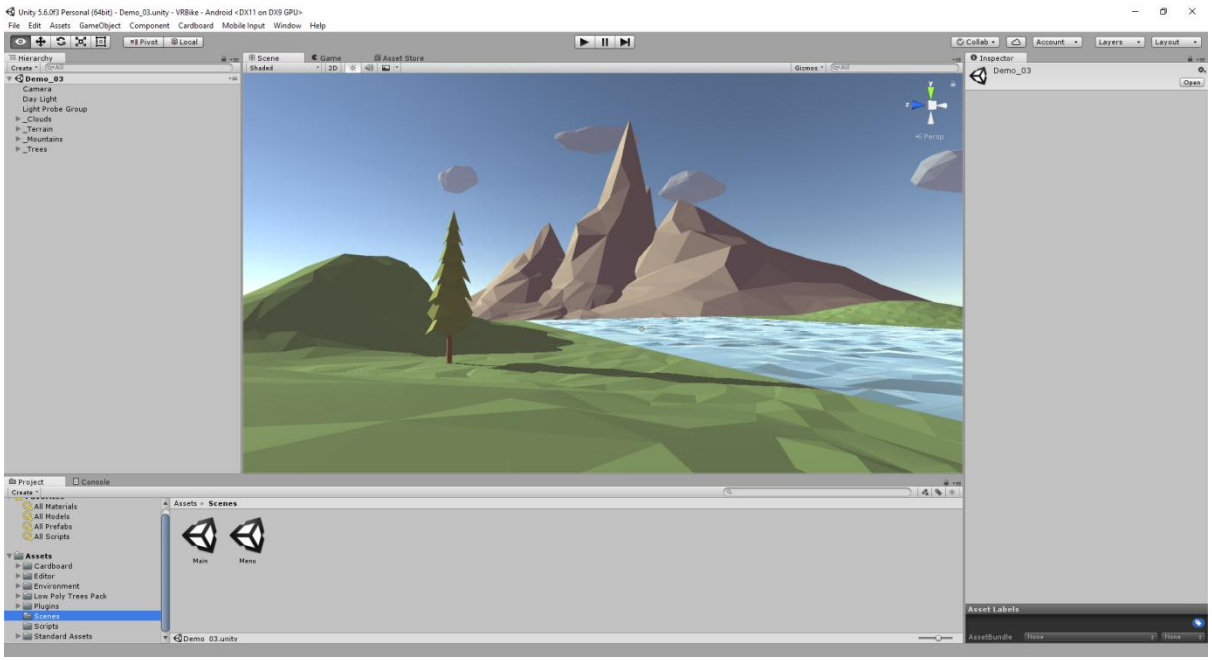

Figura 8: Pantalla Escenario

Pantallas interfaz entrenamiento dentro de la aplicación

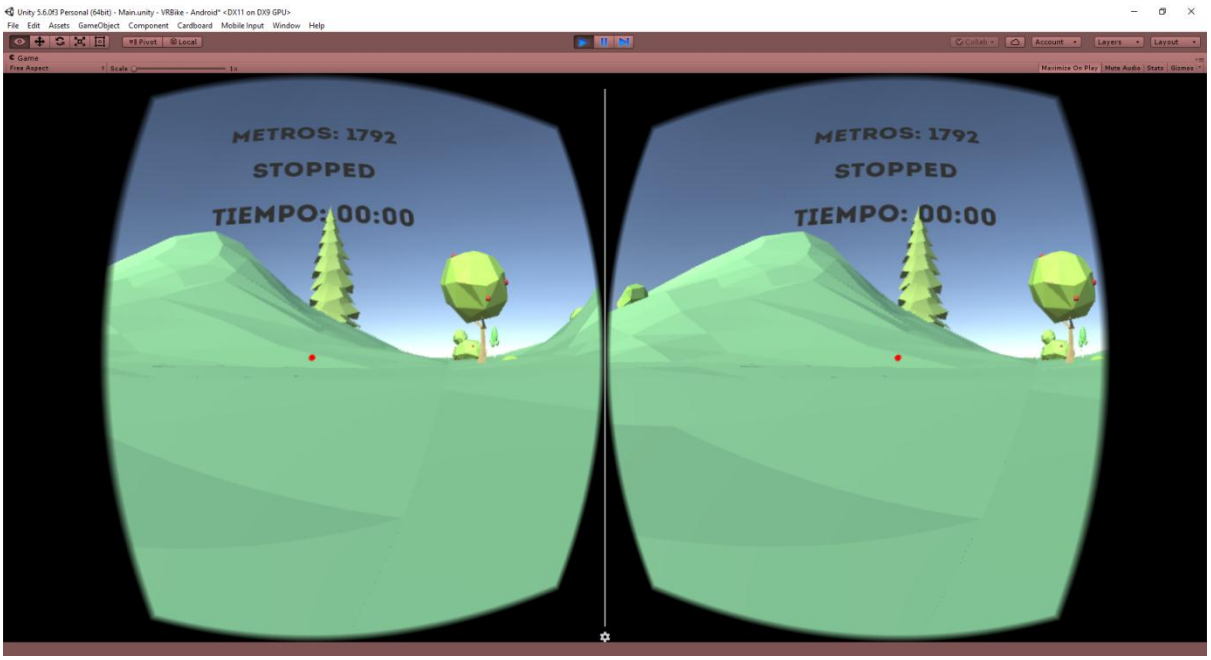

Figura 9: Pantalla Interfaz Entrenamiento

Patrones utilizados

- Singleton: Este patrón limita a uno el número de instancias posibles de una clase deseada dentro de la aplicación así como también da un acceso global a la misma. Se utilizó este patrón, por ejemplo, para el Head (objeto que contiene las dos cámaras) dentro de Unity ya que debe ser único y debe ser accedido por los demás componentes.
- Iterator: Se utiliza para poder iterar por los elementos de un conjunto de forma secuencial sin necesidad de exponer su implementación específica. Este patrón es muy utilizado en Unity ya que lo soporta nativamente, se puede iterar por una lista de elementos asociados a la escena sin la necesidad de conocer su implementación particular.
- Object Pool: Mediante este patrón se obtienen objetos nuevos a través de la clonación, dejándolos en un contenedor para ser tomados cuando sean necesarios. Se utiliza

cuando el costo de crear una clase es mayor que el costo de clonarla. Se utilizó este patrón para la generación de terrenos "infinitos" que es explicada más adelante.

Diseño Lógico y Físico

Optimización para realidad virtual

Como ya se mencionó con anterioridad, uno de los aspectos más importantes para una aplicación de realidad virtual es la performance. Con menos de 60 cuadros por segundo, la aplicación solo podrá utilizarse por algunos minutos o peor aún puede llegar a ser inutilizable y causar mareos o náuseas, más si se trata del procesamiento limitado que posee un dispositivo móvil.

Existen varias formas de optimizar las aplicaciones de realidad virtual, pero todas ellas apuntan a lo mismo: reducir las llamadas a la función "Draw" que es la que utiliza el GPU para mostrar objetos en la pantalla del celular. Mientras menos llamadas se hagan al "Draw", mayor será la performance es la aplicación.

Hay que tener en cuenta que para poder utilizar el casco de realidad virtual, el Smartphone está dividiendo su pantalla en dos y mostrando dos veces lo mismo con una leve desviación para generar el efecto de realidad virtual. Es decir que la llamada a la función de dibujo se realiza dos veces ya que está mostrando lo mismo en dos lugares de la pantalla en simultáneo.

Una ventaja de Unity es que nos permite ver por cada objeto en la escena el número de Draw Calls que éste está realizando. De esta manera sabemos qué objeto es más propenso a traer problemas de performance y cuál no.

A continuación, se nombran varias técnicas utilizadas en el desarrollo de este proyecto para reducir las llamadas al "Draw".

Occlusion culling y Frustum culling

Dos técnicas ampliamente utilizadas para mejorar la performance de las aplicaciones y juegos son "Occlusion culling" y "Frustum culling".

Frustum culling es el método en el cual los objetos se hacen invisibles si salen del espectro que toma la cámara para mostrar las imágenes. Es decir, la cámara tiene un punto

mínimo de cercanía desde donde comienza a mostrar las imágenes y un punto máximo de lejanía, al igual que para la izquierda como para la derecha; todo lo que esté fuera de estos cuatro puntos no se dibuja. Visto como un objeto tridimensional, es como una pirámide donde en su vértice está la cámara y luego se le hace un corte desde donde se comienzan a mostrar los objetos y hasta donde se dejan de mostrar los objetos. Este proceso itera por todos los objetos de la escena preguntando si se encuentran dentro o fuera de este espectro, decidiendo así si los dibuja o los ignora respectivamente. Esto permite ahorrar ciclos de dibujo al GPU (procesador de gráficos).

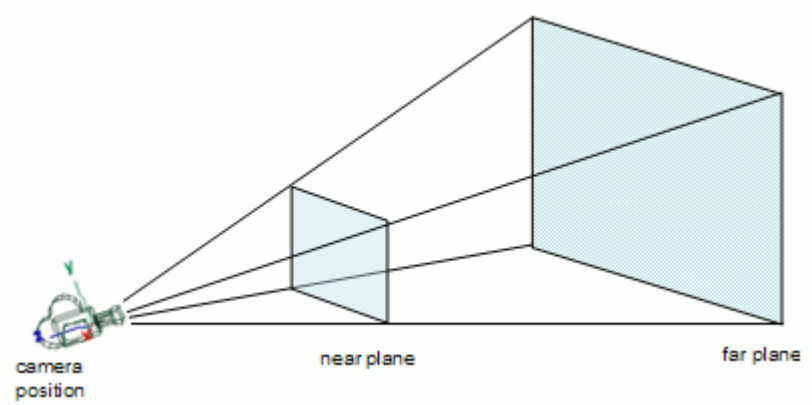

Figura 10: Visualización de Frustum Culling. Imagen propiedad de LighHouse3d.

Luego del Frustum culling se ejecuta el Occlusion culling. Este último consiste en ignorar los objetos que se encuentren tapados por otros objetos. Por ejemplo, si dentro de un cuarto hay una mesa, pero la puerta del cuarto está cerrado y el personaje está fuera del cuarto, no tiene sentido dibujar la mesa porque no se verá; pero si se abre la puerta, habrá que dibujar la mesa. El Occlusion culling itera por los objetos activos que dejó el Frustum culling analizando si se deberían ver o no porque hay un objeto delante de él.

:

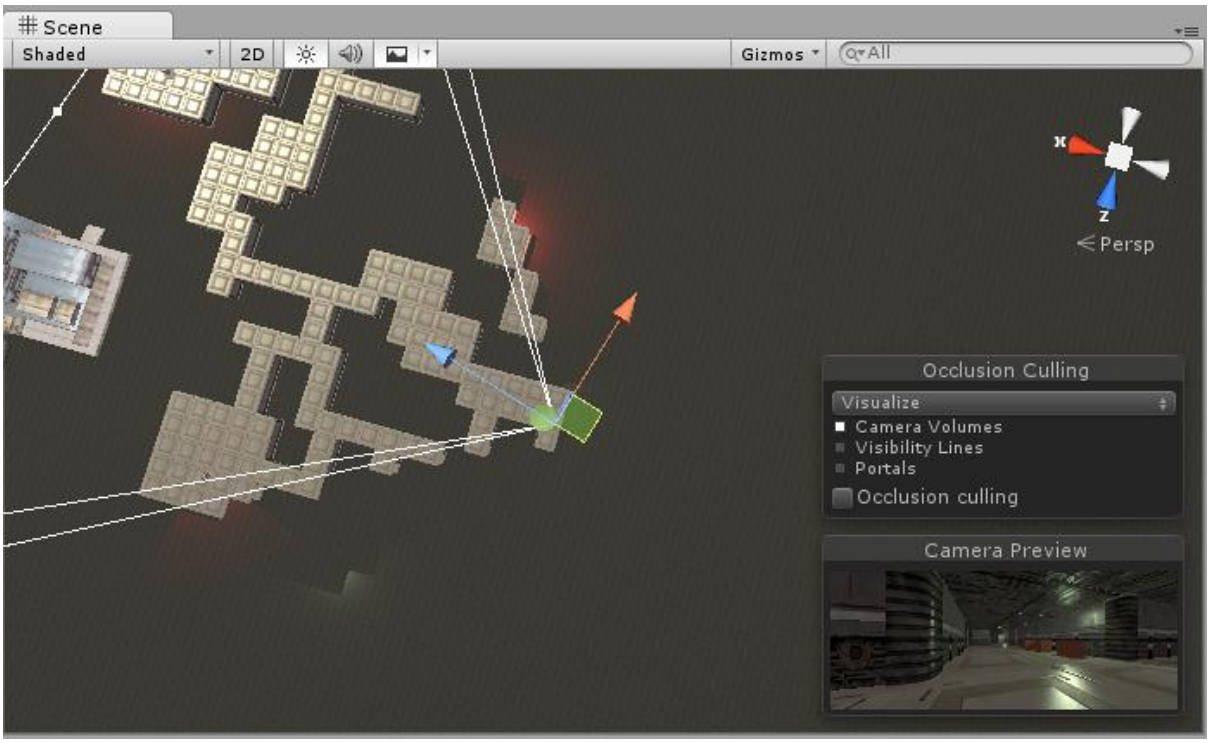

Figura 11: Visualización Frustum Culling sin Occlusion Culling. Imagen propiedad de Unity.

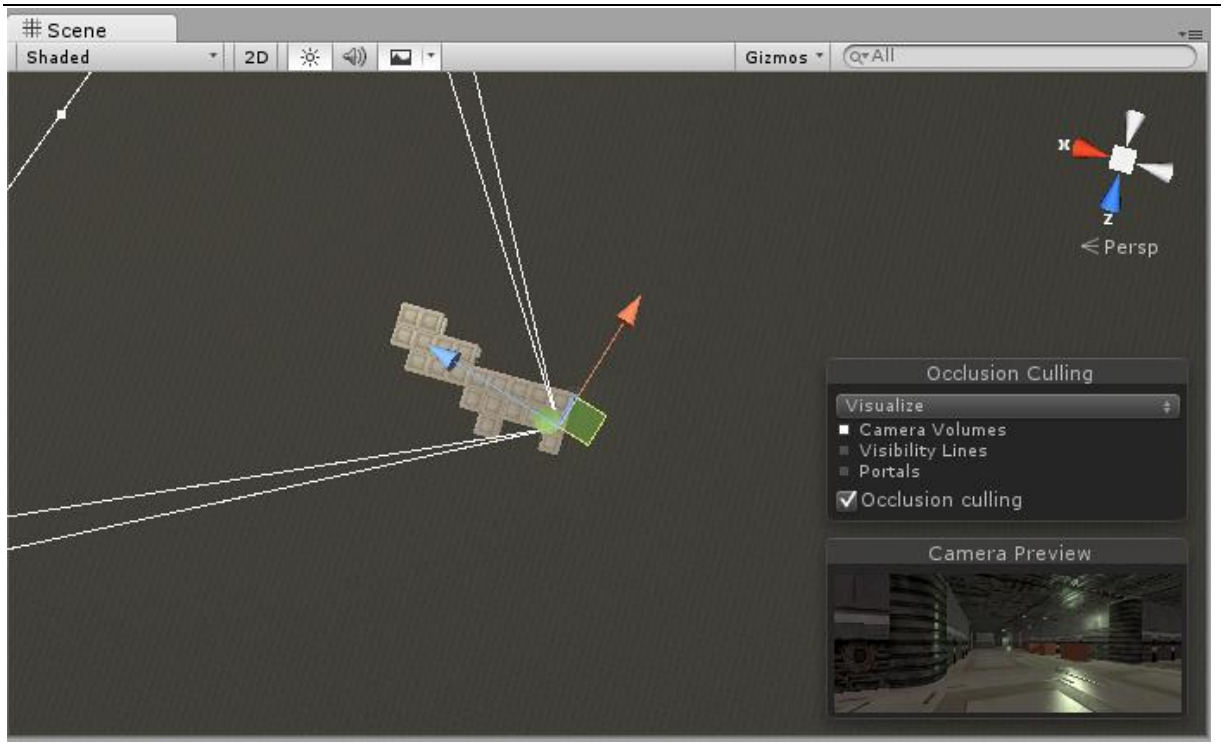

Figura 12: Visualización mismo mapa con Frustum Culling y Occlusion Culling. Imagen propiedad de Unity.

Cantidad de polígonos

Para una computadora de hoy en día la cantidad de triángulos puede ser de varios millones sin que se vea afectado el rendimiento, pero para un proyecto de realidad virtual en un dispositivo móvil puede afectar de gran manera la performance. Es decir que reducir la cantidad de polígonos ayudará a mejorarla. Por ejemplo, la luz dibujada en tiempo real debe iterar por menos cantidad de elementos al igual que los "shaders" que serán explicados más tarde.

Por este motivo, se decidió utilizar la técnica de "Low Poly" (baja cantidad de polígonos) para la aplicación, ya que es una estética ampliamente utilizada en juegos en donde los objetos tratan de seguir formas simples con pocos triángulos para modelar, sin realismo.

A este concepto también se lo conoce como LOD por sus siglas en inglés *level of detail*. Este se utiliza para bajar el nivel de detalle de los elementos que no estén en primera plana.

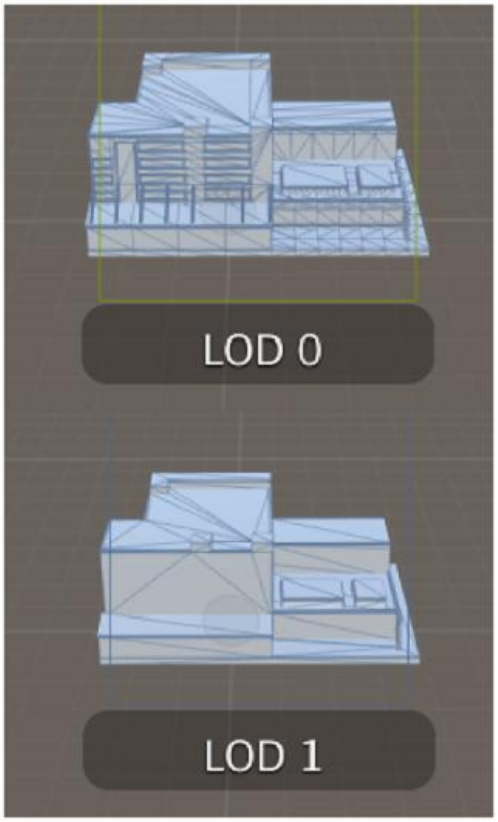

Figura 13: Utilizando el concepto de LOD para disminuir los polígonos. Imagen propiedad de Unity.

Cantidad de objetos

Reducir la cantidad de objetos en simultáneo en la escena siempre es un buen camino a seguir para reducir los ciclos de dibujo del GPU, así como también intentar unir o utilizar las mismas texturas para distintos objetos.

No solo reducir la cantidad de objetos ayuda a la performance sino que unificar varios objetos en uno también. Por ejemplo, para los árboles se suele combinar la textura del tronco y de la copa en una sola textura.

Por otro lado, si hay múltiples objetos que se utilizan como un conjunto pero son instanciados por separados lo más óptimo para la performance es unificarlos e instanciarlos como uno. Por ejemplo, como en este proyecto hay varios conjuntos de piedras por separado se decidió unificarlas en 3 grupos para que dé una idea de aleatoriedad y así reducir el número de llamadas.

Calcular previamente la luz y sombras

Si bien la iluminación en tiempo real es un efecto que agrega mucho a la experiencia de realismo, también sobrecarga al procesador de cálculos y al GPU. Es por esto que es conveniente que a los objetos que están inmóviles, o a los que no sea necesario calcular sus sombras y luces en tiempo real, se les deben calcular previamente y no en tiempo de ejecución. Para esto se necesita un programa de modelado en 3D, por ejemplo Maya, donde se crea el "lightmap" que es la textura de la cual se parte para calcular previamente la luz que recibe un objeto. Luego de procesarlo se exporta el modelo 3D iluminado directamente a Unity.

Shaders

Los Shaders son programas encargados del post proceso de imágenes. Estos programas modifican visualmente lo que está siendo renderizado por el GPU, un ejemplo muy común es el Motion Blur, que difumina los objetos que están en movimiento en la pantalla.

Los Shaders son muy demandantes de recursos del GPU, es por esto que la mejor opción para un proyecto de realidad virtual en móvil es omitirlos. Para este proyecto, se decidió no utilizar ninguno ya que, si bien aportan valor estético a la aplicación, se considera preferente priorizar el uso de iluminación en tiempo real para el escenario.

Texturas y mapas

Una textura es la imagen que se le aplica al objeto 3D para que sea visible.

Además de las texturas existen distintos tipos de mapas, que ayudan a percibir la textura de forma más realista sumando relieve, reflexión, sombreado y luz.

- Bump map: Son aquellos mapas que le dan relieve a las texturas como el caso de los "height map" y "normal map". Estos mapas se ven afectados en caso de usar luz en tiempo real. Se usan para dar una definición extra a aquellas superficies que tienen muchos hendiduras y protuberancias, como por ejemplo una pared de ladrillos. En la siguiente imagen se ve en primer lugar un material sin mapa asignado, en segundo lugar un material con "normal map" asignado, y en tercer lugar un material con "normal map" y "height map" asignado.

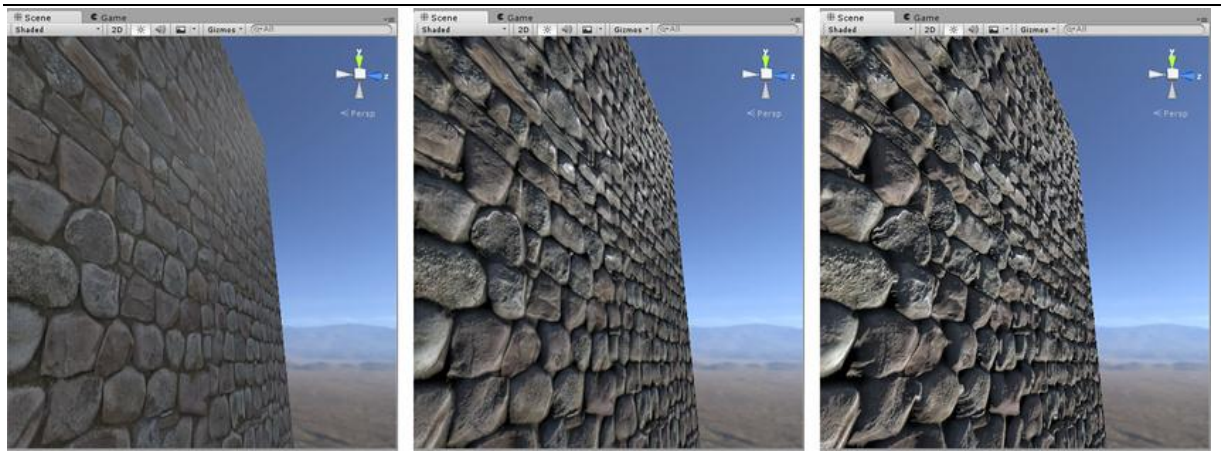

Figura 14: Bump Map. Imagen propiedad de Unity.

- Transparency map: Se utilizan para simular objetos con transparencias, como por ejemplo agua. Para simularlo usan el canal de transparencia (alfa) del color de la textura. En la siguiente imagen se observan los diferentes niveles de transparencia:

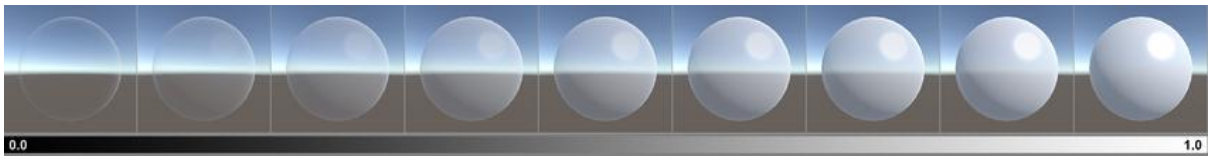

Figura 15: Transparency Map. Imagen propiedad de Unity.

- Specular map: Se utilizan para cambiar la reflectividad de un objeto. En la siguiente imagen se observan los diferentes niveles por los que puede pasar.

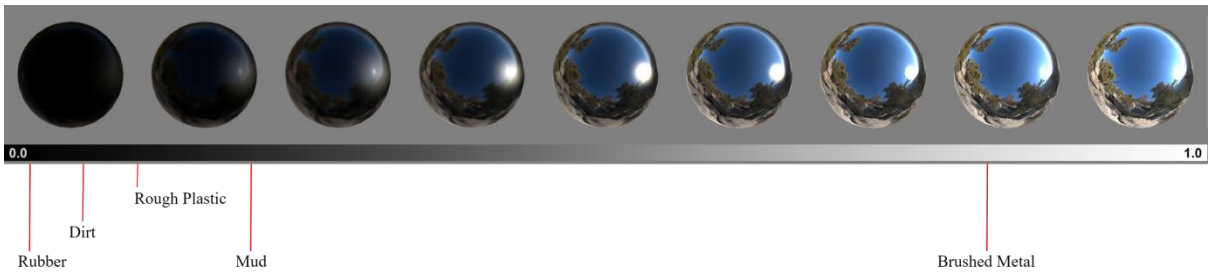

Figura 16: Specular Map. Imagen propiedad de Unity.

Object pool

Para que el usuario pueda moverse libremente en la dirección que desee, sin límites, se utilizaron varios terrenos en simultáneo. En total se utilizaron 9 terrenos que son escogidos al

azar entre 6 posibilidades distintas dentro de cada escenario. Cuando el usuario se está desplazando, se controla si se está acercando a alguno de los límites del terreno, y de ser así las secciones de terreno más lejanas son reposicionadas en la dirección en la cual se dirige. Es el mismo principio utilizado en las listas infinitas pero llevado a dos dimensiones espaciales, el eje X y el eje Z.

A esta técnica se la denomina object pool, y surge debido a la premisa de que instanciar objetos es costoso para el procesador. Lo que se hace es instanciar un "pool" (una cantidad N de esos objetos que serán utilizados) y los objetos son reposicionados para dar el efecto de terreno infinito.

Si bien esta técnica ayuda a mejorar la performance, porque reduce la cantidad de objetos que se instancia, hay un proceso que consulta en todos los ciclos si el usuario está llegando a uno de los límites para saber si debe asignar la posición de un terreno frente a él. Para mitigar este proceso, se utiliza un método llamado "frameskip" que consiste en un semáforo que permite evitar que ese proceso corra por un determinado tiempo. De no utilizarse este semáforo, el proceso correría 60 veces por segundo. Utilizando esta técnica, y teniendo en cuenta que el usuario no puede desplazarse a más de cierta velocidad, se puede reasignar terrenos, por ejemplo, una vez cada segundo.

Persistencia de datos

Debido a que en esta etapa del desarrollo no se realizó nada que requiera una importante estructura de datos, la persistencia es bastante sencilla y se hizo utilizando un Json para guardar las Opciones seleccionadas por el usuario y otro para guardar los últimos 5 Entrenamientos realizados por este.

Json opciones:

```
{
	"bMusic":false,
	"bDirectionByRotation":false,
	"iDifficulty":1,
	"iSensibility":2,
	"iScene":0,
	"bDistanceDuration":true
}
```

Json entrenamiento:


```
{
	"TablePersistance":[
		{
			"iDifficulty":1,
			"fMeters":400.0,
			"fTime":90.0
		},
		{
			"iDifficulty":1,
			"fMeters":200.0,
			"fTime":50.0
		},
		{
			"iDifficulty":2,
			"fMeters":200.0,
			"fTime":80.0
		},
		{
			"iDifficulty":2,
			"fMeters":200.0,
			"fTime":85.0
		},
		{
			"iDifficulty":0,
			"fMeters":200.0,
			"fTime":52.0
		},
		{
			"iDifficulty":0,
			"fMeters":200.0,
			"fTime":50.0
		},
		{
			"iDifficulty":0,
			"fMeters":200.0,
			"fTime":53.0
		},
		{
			"iDifficulty":0,
			"fMeters":346.0,
```

```
                "fTime":60.0
            },
            {
                "iDifficulty":1,
                "fMeters":365.0,
                "fTime":60.0
            }
        ]
}
```

Diagrama de clases

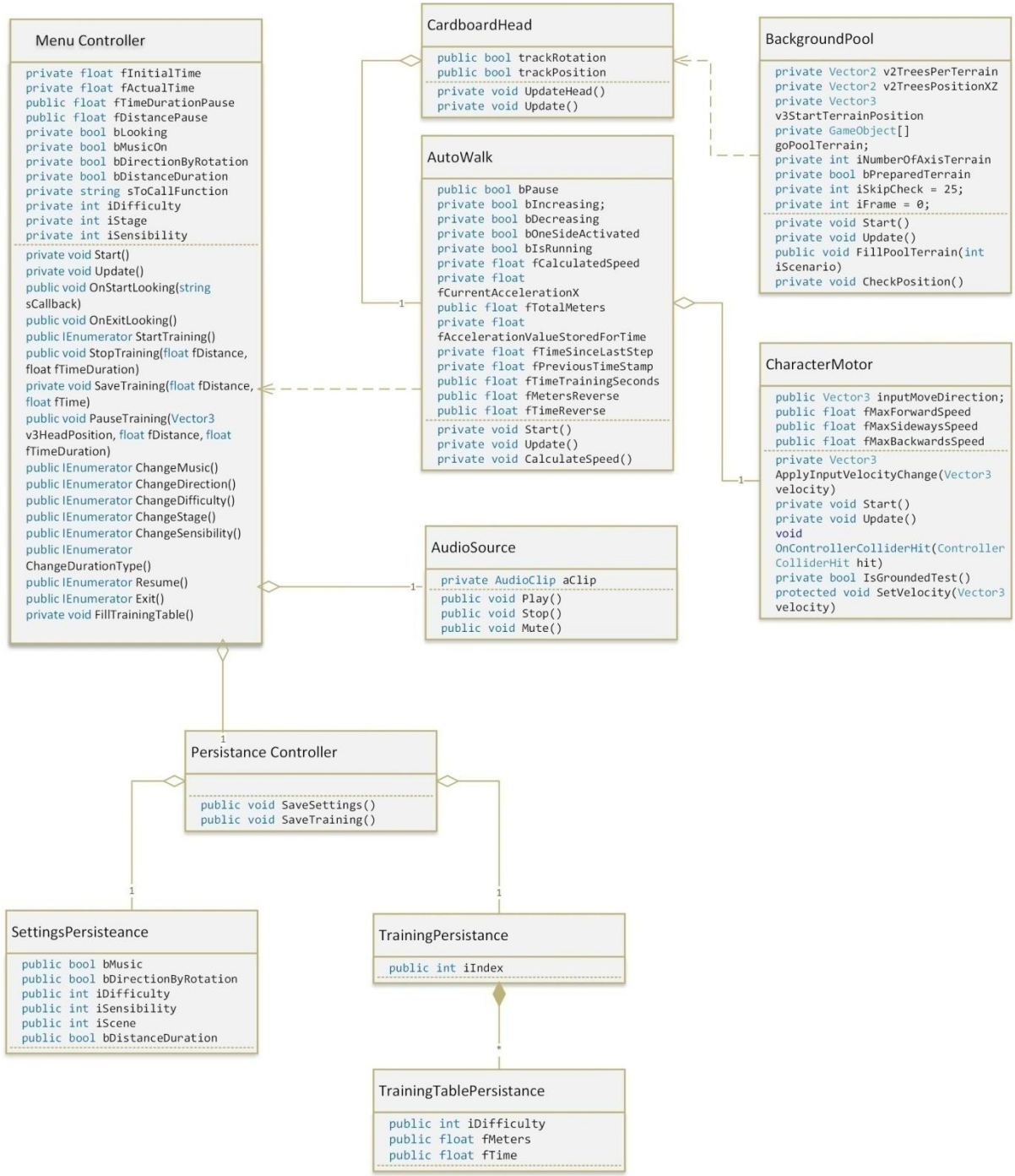

Figura 17: Diagrama de Clases

Desarrollo y Codificación del Prototipo

El desarrollo y codificación del prototipo fueron realizados siguiendo un enfoque ágil, realizando pruebas durante el desarrollo e investigando distintas formas de cumplir con los atributos de calidad descritos al comienzo del trabajo. Teniendo en cuenta esto, fue una de las etapas que más tiempo llevó.

Estructura del Proyecto

El proyecto cuenta con una única escena, “Main”, la cual se utiliza para el menú principal y para el entrenamiento.

Figura 18: Escena Main

La jerarquía de objetos es la siguiente:

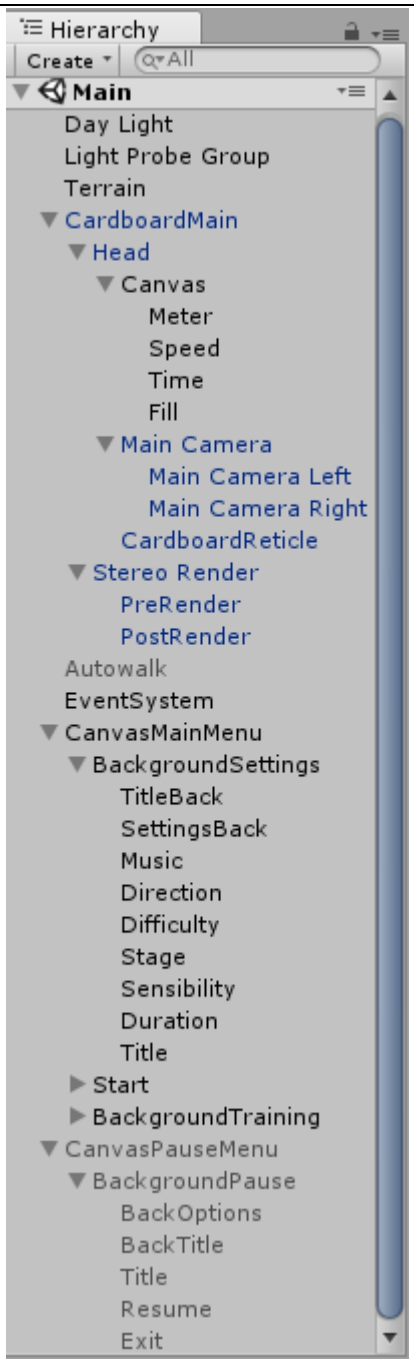

Figura 19: Jerarquía de Objetos

En donde podemos encontrar los siguientes Game Object:

- Terrain es el manejador de terrenos.

- Cardboard Main es el módulo principal de la aplicación que contiene todo el manejo de la realidad virtual y del movimiento del usuario.
- En Canvas Main Menu, está el manejo del menú principal que contiene todas las opciones a seleccionar por el usuario.
- En Canvas Pause Menu, está el manejo del menú de pausa que permite reanudar o finalizar el entrenamiento.
- Day Light y Light Probe Group manejan la luz del ambiente.

Y los Assets que se utilizaron son los siguientes:

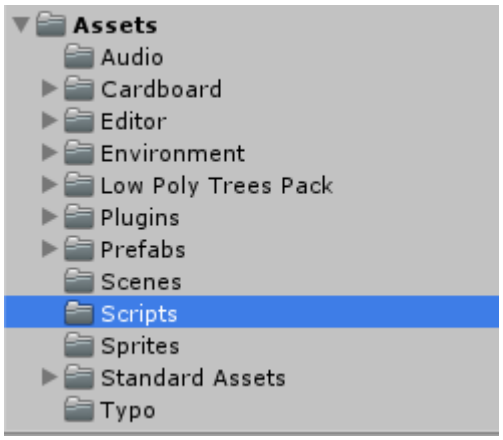

Figura 20: Assets

Persistencia

Un aspecto importante de entrenar es ver el progreso de cómo fue avanzando en tiempos y distancias, así como también en periodicidad.

Al finalizar cada entrenamiento se guardan todos los detalles del mismo para luego ser visualizado para hacer un análisis de datos. Los datos a guardar son los siguientes:

- Duración del entrenamiento: cuanto tiempo a partir de la fecha y hora se estuvo entrenando.
- Dificultad del entrenamiento: qué nivel de dificultad fue el entrenamiento (fácil, moderado, difícil)
- Distancia recorrida: distancia recorrida en el entrenamiento en metros.

Estos datos fueron persistidos como un objeto Json en PlayerPrefs de Unity, y en caso de realizarse una segunda etapa, se podría enviar mediante un Web Service a un servidor que

almacene los entrenamientos para luego visualizarlos en una web y así analizarlos de una forma más cómoda.

Dentro de la aplicación también podrán ser visualizados los últimos 5 entrenamientos realizados para poder tomar como referencia para los próximos entrenamientos.

Por otro lado, se persisten los datos de configuración del usuario, estos datos son:

- Sonido: un entero que por default es 1 (activo) y para silenciado es 0.
- Dirección: un entero que por default es 1 (Variable por Rotación) y 0 para dirección fija.
- Dificultad: un entero donde 1 es el default (media), 0 es baja y 2 es alta. Se utiliza para saber la cantidad de metros o minutos que durará el entrenamiento.
- Duración: un entero donde 1 es el default y establece la duración del entrenamiento por tiempo y 0 es para que sea por una cantidad de metros predefinida dentro de la aplicación.
- Sensibilidad: un entero que guarda por default 2 y mientras más bajo será menos sensible y mientras más alto será más sensible. Establece un mínimo de 0 y un máximo de 4.
- Escenarios: un entero 0 es para el primer escenario y 1 es para el segundo escenario.

La clase PlayerPrefs de Unity nos permite persistir datos entre sesiones de juego, en Android todo lo que se persiste con PlayerPrefs queda guardado en la ruta "/data/data/pkg-name/shared_prefs/pkg-name.xml".

Cálculo de Velocidad

Uno de los aspectos más difíciles de este proyecto es el cálculo de la velocidad con la que se debería mover una persona en el mundo virtual. Para lograr esto se pusieron en discusión varias opciones de cómo lograr una velocidad precisa pero que no implique sensores externos ni bicicletas especiales de donde se tome la velocidad. Se busco la forma más sencilla para que el usuario se abstraiga lo máximo posible de conectores y dispositivos extras los cuales cargar para ir a entrenar por lo cual se decidió utilizar los sensores del Smartphone.

Un acelerómetro es un mecanismo que permite conocer tanto la orientación de un dispositivo como su aceleración. Los celulares tienen la combinación de 3 acelerómetros MEMS (micro-electromechanical system), uno para cada eje, que están hechos de partes microscópicas de silicona. Cuando una fuerza perturba alguno de estos acelerómetros, generan una corriente eléctrica que se traduce en datos de aceleración.

Por esta razón, cuando el dispositivo está apoyado en una mesa (sin moverse), el acelerómetro lee una magnitud de g=9.81m/s2. Con la misma lógica, cuando el celular está en caída libre con una aceleración de 9.81m/s2, el acelerómetro lee una magnitud de g=0m/s2.

Debido a que la información que devuelve el acelerómetro puede contener algo de ruido, se le puede aplicar un "low-pass filter" para obtener un resultado más suave para interpretar.

Los acelerómetros usan Sistema de Coordenadas estándar para sensores, esto es:

Si una persona sostiene el dispositivo móvil derecho con el botón de inicio debajo y con la pantalla frente a sí mismo, el eje X es positivo hacia la derecha, el eje Y es positivo hacia arriba y el eje Z es positivo hacia la persona que sostiene el celular.

Por lo que en la práctica aplica la siguiente lógica cuando un dispositivo está en reposo en su posición natural:

Si se empuja el dispositivo desde el lado izquierdo (para que se mueva hacia la derecha), el valor de la aceleración del eje x es positivo.

Si se empuja el dispositivo desde abajo (para que se mueva hacia arriba), el valor de la aceleración en y es positivo.

Si se empuja el dispositivo hacia el cielo con una aceleración A (en m/s2), la aceleración en z es igual a A + 9.81, que corresponde a la aceleración del dispositivo menos la fuerza de gravedad (-9.81 m/s2).

El acelerómetro en Unity puede ser accedido de la siguiente forma:

Input.acceleration

Devuelve un Vector3 que incluye la aceleración de la gravedad. Para obtener un valor sin la aceleración de la gravedad, se utiliza Input.gyro.userAcceleration del giroscopio que se verá más adelante.

Un giroscopio es un dispositivo que permite medir la rotación. Los dispositivos móviles, usan giroscopios MEMS basados en la vibración. Como los objetos que vibran, tienden a continuar vibrando en el mismo plano aunque hayan sido rotados, esta masa vibrante genera una fuerza que puede ser detectada electrónicamente y que se traduce en medidas de rotación

El giroscopio mide el ratio de rotación en rad/s en los ejes X, Y y Z del dispositivo.

La rotación es positiva en el sentido contrario a las agujas del reloj, esto es: un observador mirando desde un lugar positivo en el sentido de los 3 ejes a un dispositivo ubicado en el origen, reportaría una rotación positiva si el dispositivo rota en el sentido contrario de las agujas del reloj.

El giroscopio en Unity se puede acceder de la siguiente forma:

Input.gyro

Los atributos del Giroscopio son:

Tabla 50: Atributos del Giroscopio

attitude	Devuelve la altitud (ej.: orientación en el espacio) del dispositivo.
enabled	Setea o devuelve si está habilitado el giroscopio.
gravity	Devuelve la aceleración de la gravedad expresado en un vector3 en el marco de referencia del dispositivo.
rotationRate	Devuelve el radio de rotación. Es un Vector3 que representa la velocidad de rotación en cada uno de los 3 ejes en radianes por segundo. Este valor está expresado directamente como lo reporta el hardware del dispositivo.
rotationRateUnbiased	Devuelve el radio de rotación corregido. Es un Vector3 que representa la velocidad de rotación en cada uno de los 3 ejes en radianes por segundo. Este valor ha sido procesado para obtener una medida más precisa.

updateInterval	Setea o devuelve el intervalo por segundos en que obtiene la información el giroscopio.
userAcceleration	Devuelve la aceleración que el usuario le está dando al dispositivo. La aceleración de la gravedad ha sido removida para obtener solamente la aceleración del usuario.

Cálculo

Se tomó la aceleración del dispositivo en el eje X con la función Input.gyro.userAcceleration.x.

Se iba a utilizar la aceleración de los 3 ejes (aplicando un filtro a los ejes de menor relevancia) pero se descubrió un bug en el cálculo del absoluto de la aceleración de los ejes Y y Z con la función Mathf.Abs.

Por esta razón, sólo se utilizó la aceleración en el eje X, multiplicada por un factor de velocidad. Este factor es el que se ve alterado al cambiar la sensibilidad del dispositivo.

A todo esto se le aplicó un filtro de ruido, ya que la medición del acelerómetro en muy variable, interpolando entre el valor anterior y el actual que mostraba el acelerómetro para hacerlo más fluido.

Por otro lado, se utilizó un semáforo por tiempo, para poder mirar hacia los costados sin que el personaje comience a desplazarse. Es decir que si en "t" cantidad de tiempo no realizó dos aceleraciones opuestas en el eje X, el personaje no avanzará, permitiendo así apreciar el paisaje.

Pasos para descargar:

Para finalizar con el contenido de esta sección, a continuación se describen los pasos para descargar la aplicación para desarrollo:

1. Descargar SourceTree desde la página oficial de Atlassian (https://www.sourcetreeapp.com/).
2. Clonar el repositorio alojado en bitbucket.org con la URL "**https://FrancoMoG@bitbucket.org/tesisVRBike/vrbike.git**".

3. Descargar Unity 5.6.0 f3 Personal o posterior.
4. Abrir el proyecto con Unity 5.6.0 f3 Personal. Si bien es posible migrar a otra versión, se debe tener en cuenta que el código fue optimizado y compilado para esta.
5. Desde Unity:
    a. Para elegir la plataforma a compilar, ingresar en “File/Build Settings”, y seleccionar utilizando el botón “Switch Platform”.
    b. Para correr la aplicación, seleccionar el botón “Build And Run”. Luego de algunos minutos la aplicación debe correr en el dispositivo seleccionado.

Pruebas

El prototipo desarrollado fue probado tanto dentro del entorno en el que fue construido (Unity) como en dos dispositivos móviles iPhone (iPhone 6 y iPhone 5S) y un dispositivo móvil Android (Motorola Moto G2).

Si bien durante la construcción fueron realizadas múltiples pruebas para validar el correcto funcionamiento del producto, muchas de ellas no fueron documentadas de forma concreta debido a la extensión del presente trabajo, por lo que se decidió documentar no más de 5 casos de prueba por cada historia de usuario.

Se adjuntan a continuación los casos de prueba documentados.

Pruebas funcionales

Tabla 51: Pruebas Funcionales

ÍTEM	USER STORY	ACCIÓN	ENTRADA	RESULTADO ESPERADO	RESULTADO OBTENIDO
CP001	U.S.1: Desplazamiento	Pedalear en la bicicleta fija	Pedaleo constante	El personaje se desplaza a velocidad constante por el entorno	El personaje se desplaza a velocidad constante por el entorno
CP002	U.S.1: Desplazamiento	Pedalear en la bicicleta fija acelerando	Pedaleo acelerado	El personaje acelera su velocidad	El personaje acelera su velocidad
CP003	U.S.1: Desplazamiento	Dejar de pedalear	Dejar de pedalear estando en movimiento dentro de la realidad virtual	El personaje desacelera su velocidad hasta frenar completamente	El personaje desacelera su velocidad hasta frenar completamente
CP004	U.S.2: Rotación	Rotar la cabeza de forma horizontal 90º	Rotar la cabeza de forma horizontal de izquierda a derecha con un ángulo de 90º	La visual en el mundo virtual realiza el mismo paneo que la cabeza del usuario	La visual en el mundo virtual realiza el mismo paneo que la cabeza del usuario
CP005	U.S.2: Rotación	Rotar la cabeza de forma vertical 90º	Rotar la cabeza de forma vertical de abajo hacia arriba con un ángulo de 90º	La visual en el mundo virtual realiza el mismo paneo que la cabeza del usuario	La visual en el mundo virtual realiza el mismo paneo que la cabeza del usuario
CP006	U.S.3: Inicio entrenamiento	Fijar vista 1.5 segundos en botón Iniciar	Fijación de vista durante 1.5 segundos en botón Iniciar	Comienza el entrenamiento	Se oculta el botón Iniciar, el menú de opciones y la tabla de entrenamientos. Comienza el entrenamiento.
CP007	U.S.4: Menú Principal	Ingresar a la aplicación	Ingreso a la aplicación	Se observa el Menú Principal, el botón Iniciar y la tabla de entrenamientos.	Visualización de botón Iniciar, menú de opciones y tabla de entrenamientos.
CP008	U.S.5: Selección opciones	Fijar vista 1.5 segundos en	Fijación de vista durante 1.5 segundos en cualquier	Cambia la opción seleccionada.	Se observa barra de progreso completándose y cambio de

		cualquier botón	botón		opción.
CP009	U.S.18: Visualización Entrenamientos	Realizar 5 entrenamientos	5 entrenamientos realizados	Se observa la tabla con los últimos 5 entrenamientos realizados.	Visualización de la tabla de entrenamientos con la información de los últimos 5 entrenamientos realizados.
CP010	U.S.6: Pausa	Quedarse quieto durante 5 segundos	No se capta movimiento durante 5 segundos.	Se pausa el entrenamiento.	Se pausa el entrenamiento, no se desplaza el personaje, se pausan los contadores de tiempo y distancia.
CP011	U.S.7: Menú Pausa	Pausar entrenamiento	Entrenamiento pausado.	Se muestra el Menú de Pausa.	Se visualiza el menú de pausa con dos opciones: reanudar o salir.
CP012	U.S.8: Selección Salir	Fijar vista 1.5 segundos en Salir	Entrenamiento pausado, fijación de vista durante 1.5 segundos en botón Salir.	Se finaliza el entrenamiento.	Se finaliza el entrenamiento, los contadores vuelven a cero, se visualiza el menú de opciones, el botón Iniciar y la tabla de entrenamientos. El entrenamiento realizado aparece en el primer lugar de la tabla de entrenamientos.
CP013	U.S.9: Selección Reanudar	Fijar vista 1.5 segundos en Reanudar	Entrenamiento pausado, fijación de vista durante 1.5 segundos en botón Reanudar.	Se reanuda el entrenamiento.	Se reanuda el entrenamiento, los contadores continúan como habían quedado antes de la pausa, se oculta el menú de opciones, el botón Iniciar y la tabla de entrenamientos. El personaje comienza a desplazarse.
CP014	U.S.13: Selección Duración	Fijar vista 1.5 segundos en Duración	Duración en estado "Tiempo", fijación de vista durante 1.5 segundos en botón Duración	Cambia la opción Duración a "por Distancia". Cuando comienza el entrenamiento se observa en el display de Distancia la cantidad de metros	Duración pasa a estado "Distancia". El entrenamiento finaliza luego de recorrer la cantidad de metros

				seleccionada por Dificultad y el display de Tiempo en 0. Termina el entrenamiento cuando se llega a 0 metros.	preseleccionada sin importar el tiempo que haya transcurrido.
CP015	U.S.13: Selección Duración	Fijar vista 1.5 segundos en Duración	Duración en estado "Distancia", fijación de vista durante 1.5 segundos en botón Duración	Cambia la opción Duración a "por Tiempo". Cuando comienza el entrenamiento se observa en el display de Tiempo la cantidad de minutos seleccionada por Dificultad y el display de Distancia en 0. Termina el entrenamiento cuando se llega a 0 minutos, 0 segundos.	Duración pasa a estado "Tiempo". En el entrenamiento finaliza luego que haya transcurrido el tiempo preseleccionado sin importar la cantidad de metros recorridos.
CP016	U.S.14: Selección Sensibilidad	Fijar vista 1.5 segundos en Sensibilidad	Sensibilidad en estado "Sensibilidad 0", fijación de vista durante 1.5 segundos en botón Sensibilidad	Cambia la opción Sensibilidad a 1. Se requiere un esfuerzo levemente inferior a la Sensibilidad 0.	Sensibilidad pasa a estado "Sensibilidad 1". Esfuerzo levemente inferior a "Sensibilidad 0".
CP017	U.S.14: Selección Sensibilidad	Fijar vista 1.5 segundos en Sensibilidad	Sensibilidad en estado "Sensibilidad 1", fijación de vista durante 1.5 segundos en botón Sensibilidad	Cambia la opción Sensibilidad a 2. Se requiere un esfuerzo intermedio.	Sensibilidad pasa a estado "Sensibilidad 2". Esfuerzo medio.
CP018	U.S.14: Selección Sensibilidad	Fijar vista 1.5 segundos en Sensibilidad	Sensibilidad en estado "Sensibilidad 2", fijación de vista durante 1.5 segundos en botón Sensibilidad	Cambia la opción Sensibilidad a 3. Se requiere un esfuerzo levemente superior a la Sensibilidad 4.	Sensibilidad pasa a estado "Sensibilidad 3". Esfuerzo levemente superior a "Sensibilidad 4".
CP019	U.S.14: Selección Sensibilidad	Fijar vista 1.5 segundos en Sensibilidad	Sensibilidad en estado "Sensibilidad 3", fijación de vista durante 1.5 segundos en botón	Cambia la opción Sensibilidad a 4. Se requiere un esfuerzo mínimo para desplazarse.	Sensibilidad pasa a estado "Sensibilidad 4". Mínimo esfuerzo para desplazarse.

			Sensibilidad		
CP020	U.S.14: Selección Sensibilidad	Fijar vista 1.5 segundos en Sensibilidad	Sensibilidad en estado "Sensibilidad 4", fijación de vista durante 1.5 segundos en botón Sensibilidad	Cambia la opción Sensibilidad a 5. Se requiere un esfuerzo máximo para desplazarse.	Sensibilidad pasa a estado "Sensibilidad 0". Máximo esfuerzo para desplazarse.
CP021	U.S.2: Rotación	Fijar vista 1.5 segundos en Dirección	Dirección en estado "Fija", fijación de vista durante 1.5 segundos en botón Dirección	Cambia la opción Dirección a "por Rotación". El personaje debe desplazarse en la dirección en que el usuario está mirando.	Dirección pasa a estado "por rotación". Al rotar la cabeza, el personaje se dirige en la dirección que está mirando.
CP022	U.S.10: Fijar Dirección	Fijar vista 1.5 segundos en Dirección	Dirección en estado "por Rotación", fijación de vista durante 1.5 segundos en botón Dirección	Cambia la opción Dirección a "Fija". El personaje se desplaza en línea recta mientras el usuario observa los alrededores.	Dirección pasa a estado "Fija". El usuario puede rotar la cabeza para ver el entorno, mientras el personaje continua moviéndose en línea recta.
CP023	U.S.12: Selección Dificultad	Fijar vista 1.5 segundos en Dificultad	Dificultad en estado "Baja", fijación de vista durante 1.5 segundos	Cambia la opción Dificultad a "Media". El entrenamiento debe durar 20 minutos o 2000 metros según Duración por Tiempo o Duración por Distancia respectivamente.	Dificultad pasa a estado "Media". En caso de estar Duración por Tiempo, el entrenamiento durara 20 minutos. En caso de estar Duración por Distancia, el entrenamiento durara 2000 metros.
CP024	U.S.12: Selección Dificultad	Fijar vista 1.5 segundos en Dificultad	Dificultad en estado "Media", fijación de vista durante 1.5 segundos	Cambia la opción Dificultad a "Alta". El entrenamiento debe durar 30 minutos o 4000 metros según Duración por Tiempo o Duración por Distancia respectivamente.	Dificultad pasa a estado "Alta". En caso de estar Duración por Tiempo, el entrenamiento durara 30 minutos. En caso de estar Duración por Distancia, el entrenamiento durara 4000 metros.
CP025	U.S.12: Selección	Fijar vista 1.5	Dificultad en estado	Cambia la opción Dificultad a "Baja".	Dificultad pasa a estado "Baja".

	Dificultad	segundos en Dificultad	"Alta", fijación de vista durante 1.5 segundos	El entrenamiento debe durar 10 minutos o 1000 metros según Duración por Tiempo o Duración por Distancia respectivamente.	En caso de estar Duración por Tiempo, el entrenamiento durara 10 minutos. En caso de estar Duración por Distancia, el entrenamiento durara 1000 metros.
CP026	U.S.15: Selección Escenarios	Fijar vista 1.5 segundos en Escenario	Escenario en estado "Playa", fijación de vista durante 1.5 segundos en botón Escenario	Cambia la opción Escenario a "Bosque". Se visualiza el terreno bosque.	Escenario pasa a estado "Bosque". Se visualiza el cambio del terreno de Playa a Bosque.
CP027	U.S.15: Selección Escenarios	Fijar vista 1.5 segundos en Escenario	Escenario en estado "Bosque", fijación de vista durante 1.5 segundos en botón Escenario	Cambia la opción Escenario a "Playa". Se visualiza el terreno playa.	Escenario pasa a estado "Playa". Se visualiza el cambio del terreno de Bosque a Playa.
CP028	U.S.16: Música Ambiental	Fijar vista 1.5 segundos en Sonido	Sonido en estado "on", se escucha música, fijación de vista durante 1.5 segundos en botón Sonido	Cambia la opción Sonido de ON a OFF. Se silencia la música.	Sonido pasa a estado "off", queda silenciado.
CP029	U.S.17: Silenciar	Fijar vista 1.5 segundos en Sonido	Sonido en estado "off", no se escucha música, fijación de vista durante 1.5 segundos	Cambia la opción Sonido de OFF a ON. Se escucha la música.	Sonido pasa a estado "on", se escucha música.
CP030	U.S.19: Persistencia opciones	Cerrar aplicación y volver a abrirla	Antes de cerrar la aplicación opción seteadas: Sonido off, Dificultad Alta, Sensibilidad 2, Duración por Tiempo, Dirección Fija, Escenario Playa.	Se observan las opciones: Sonido off, Dificultad Alta, Sensibilidad 2, Duración por Tiempo, Dirección Fija, Escenario Playa.	Al abrir la aplicación opciones seteadas: Sonido off, Dificultad Alta, Sensibilidad 2, Duración por Tiempo, Dirección Fija, Escenario Playa.
CP031	U.S.11:	Pedalear en la	Duración por Distancia,	Disminuyen los metros hasta 0.	El contador de metros disminuye

	Visualización distancia y tiempo	bicicleta fija	pedaleo constante.	Aumenta el tiempo.	hasta 0, el contador de tiempo aumenta sin límite.
CP032	U.S.11: Visualización distancia y tiempo	Pedalear en la bicicleta fija	Duración por Tiempo, pedaleo constante.	Disminuyen los minutos hasta 0. Aumentan los metros.	El contador de tiempo disminuye hasta 0, el contador de metros aumenta sin límite.

Pruebas no funcionales

Debido a que para el proyecto no fueron definidos de forma cuantitativa los requerimientos no funcionales para el prototipo funcional, la responsabilidad de establecer los objetivos cuantificables que se esperan lograr con cada prueba recaerá en el equipo de testers y/o implementadores de la aplicación productiva, que deberán analizar los valores obtenidos como resultado de la ejecución de las pruebas que se definen a continuación, y de muchas otras, para poder determinar si se cumples sus objetivos.

Pruebas de usabilidad

1. Solicitar a los usuarios que se suban a la bicicleta fija.
2. Una vez acomodados, solicitarles que abran la aplicación, que coloquen el dispositivo móvil dentro del casco y lo ajusten según su comodidad.
3. Una vez listos para comenzar, solicitarles que realicen una vez cada una de las funcionalidades principales que provee la aplicación, es decir:
    a. Comenzar un entrenamiento.
    b. Pausar un entrenamiento.
    c. Reanudar el entrenamiento.
    d. Finalizar el entrenamiento.
    e. Cambiar cada una de las opciones.
    f. Visualizar el entrenamiento realizado y verificar que los datos sean correctos.
    g. Cambiar la sensibilidad para ajustarla al nivel deseado.
    h. Elegir la dificultad del entrenamiento.
    i. Elegir qué tipo de duración desea (tiempo o distancia).
    j. Elegir un escenario.
4. Medir el tiempo promedio que les lleva a los usuarios realizar cada una de las funcionalidades de forma completa.
5. Indicar a los usuarios que deben ejecutar las mismas tareas dos veces más, y medir nuevamente el tiempo promedio en cada iteración, pudiendo medir así la curva de aprendizaje de la aplicación.
6. Una vez finalizado el paso anterior, entrevistar a cada usuario individualmente y obtener sus opiniones y recomendaciones acerca de la interfaz de la aplicación, adquiriendo el feedback necesario para poder mejorarla y hacerla más “user-friendly”:

Pruebas de Performance

1. Abrir la aplicación y realizar un entrenamiento, observando el display con la cantidad de FPS y verificando que no sea inferior a 60 fps.
2. Medir la batería del celular antes de abrir la aplicación, abrirla y realizar un entrenamiento. Al finalizar, medir la batería del celular nuevamente.

Pruebas de Ofuscación

1. Compilar la aplicación con el código ofuscado generando el APK.
2. Abrir el APK y revisar el código en búsqueda de código legible tratando de reconstruir el código original sin utilizar la clave de desofuscación.

Pruebas de Portabilidad

1. Probar la aplicación en Android.
2. Probar la aplicación en iOS.
3. Probar la aplicación en Oculus Rift.
4. Probar la aplicación en HTC Vive.
5. Probar la aplicación en la plataforma que se desee implementar.

Metodología de desarrollo

Para realizar este proyecto se utilizó el ciclo de vida iterativo incremental ya que se adosa a la creación y refinamiento de prototipos. Este ciclo de vida se basa en una secuencia de actividades ordenadas lógicamente donde el producto y las tareas se refinan por ciclos sucesivos. Gracias a esto se pueden realizar productos intermedios con los cuales obtener feedback y así retroalimentar al sistema para mejorar el producto.

SCRUM

Para llevar a cabo el proyecto se utilizaron técnicas del paradigma ágil y se basó en el manifiesto de este paradigma. Para el trabajo día a día se utilizó la técnica de SCRUM, que es un ambiente de trabajo iterativo e incremental para la gestión ágil de proyectos para el desarrollo de productos y aplicaciones.

SCRUM consiste en organizar el trabajo en ciclos cortos llamados “sprints”. Antes de comenzar cada sprint se decide qué tareas del backlog (lista del total de tareas) se van a realizar. Estos ciclos pueden durar de 1 a 4 semanas pero el tiempo siempre es fijo, y si el

trabajo planeado para ese sprint no se finaliza, la finalización de esa tarea se posterga a otro sprint. Para este proyecto se acordó que utilizar sprints semanales era una cantidad de tiempo suficiente para realizar tareas para un prototipo.

Los elementos de SCRUM son los siguientes:

- Backlog: Es el conjunto total de las tareas identificadas hasta el momento, a realizarse durante el desarrollo. En caso de aparecer nuevas tareas se deben sumar aquí, al igual que las tareas que no fueron finalizadas durante el sprint. Todas las tareas están priorizadas y esta prioridad es modificable.
- Sprint Backlog: Es un subconjunto de tareas del backlog que conforman el alcance definido para completar en un sprint seleccionadas por el equipo de trabajo previamente.
- Uncommitted Backlog: Son las tareas del Backlog que aún no se ha acordado con el cliente incorporar en algún sprint.
- Sprint: Ciclo de 1 a 4 semanas en el cual se desarrollan las tareas definidas y priorizadas en el sprint backlog. En el sprint se espera los siguiente:
    - El equipo selecciona del backlog priorizado las tareas a realizar durante el sprint, definiendo el sprint backlog.
    - El equipo se compromete a cumplir con las tareas al final del sprint.
    - La duración del sprint es fija. Finaliza aun cuando hayan quedado tareas pendientes del sprint backlog.
    - Durante el sprint no se pueden agregar/eliminar tareas o cambiarles la prioridad.
- Daily Meetings: Son reuniones diarias de corta duración para evaluar el progreso del trabajo realizado donde cada integrante informa:
    - Las tareas realizadas en el día anterior.
    - Qué tareas tiene previstas para ese día.
    - Qué impedimentos tuvo para completar las tareas.
- Roles: si bien en SCRUM hay 3 roles, Product Owner, Scrum Master y Team Members, para la realización del proyecto cada uno tuvo que ocupar su rol de Team Member y ambos intentaron realizar las tareas del Product Owner y Scrum Master.

El proceso de SCRUM que se siguió es el siguiente:

1. Se definieron en conjunto las tareas del backlog estableciendo riesgos y prioridades para cada una.
2. Previo al comienzo de cada Sprint, se acordaron las tareas a realizar del backlog.
3. Se desarrolló el Sprint, sin cambiar las tareas seleccionadas y actualizando el backlog.
    a. No se hicieron reuniones diarias ya que al ser dos personas es más sencillo comentar cualquier dificultad en el momento y solucionarla para poder proseguir. Pero sí se hizo una distribución de las tareas que realizaba cada uno en el día.
4. Luego de cada sprint, se revisó el entregable generado. Se analizó si todas las tareas del sprint backlog fueron completadas y se compiló la versión para probar directamente en el Smartphone. Todas las tareas que no fueron finalizadas volvieron al backlog y fueron re priorizadas antes del siguiente sprint. Por ejemplo, una de las tareas que hubo que re priorizar por no haber finalizado a tiempo fue el cálculo de la velocidad.
5. Luego de revisar el entregable se puso en común qué se podría mejorar del proceso para obtener un producto de mejor calidad y así aplicar este cambio en la próxima iteración. Una de las mejoras que se hablaron luego de la primera iteración fue que, si bien el merge es automático con la herramienta “SourceTree”, cuando se trabaja sobre las mismas clases, lleva más tiempo hacer el merge y revisar que se haya realizado correctamente, que trabajar en clases diferentes de ser posible y que el merge sea sin conflictos. Es por esto que, luego de la primera iteración, al iniciar el día se hacía una evaluación a conciencia de cómo repartir las tareas.

Para llevar a cabo esta técnica se utilizó Trello, que es una herramienta online donde se pueden generar tarjetas encolumnadas bajo un título en común. Se utilizaron los siguientes títulos:

- Uncommited Backlog
- Backlog
- Sprint Backlog
- Sprint Doing
- Done

Bajo estos títulos se agruparon las tareas, cada una representada con una tarjeta, y al finalizar se pasaron al grupo “Done”.

Las siguientes imágenes fueron sacadas durante el tercer Sprint:

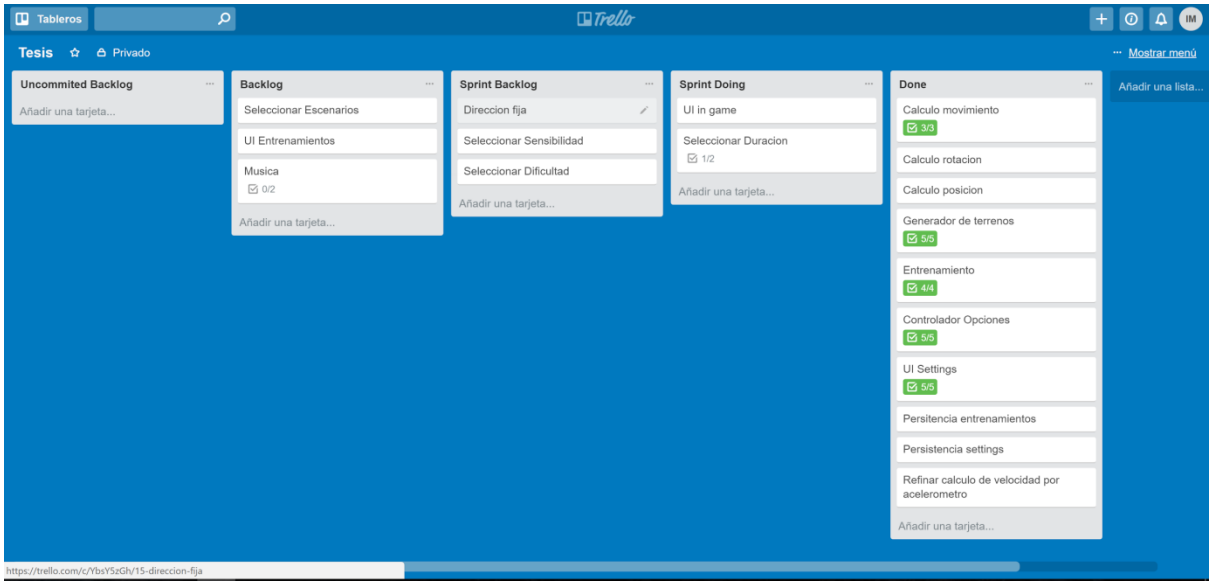

Figura 21: Tablero realizado en Trello durante el tercer Sprint

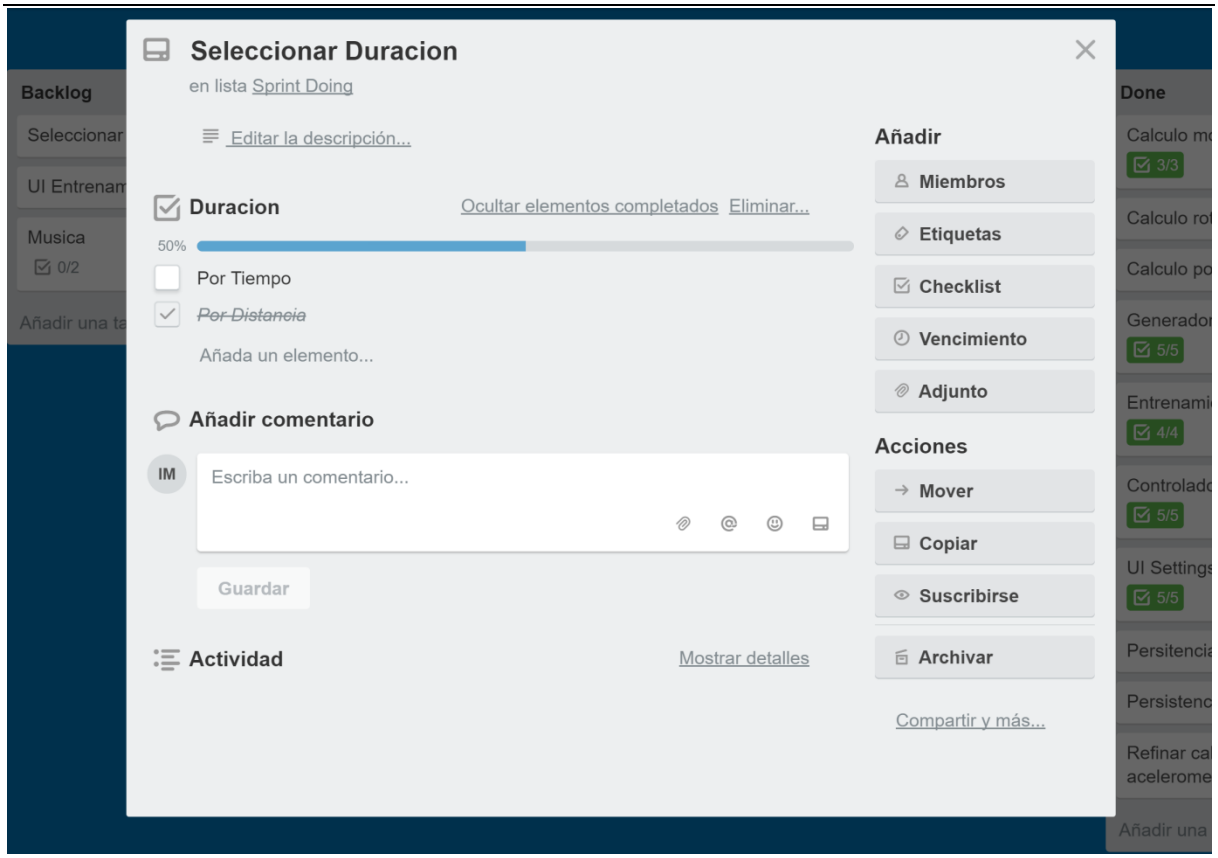

Figura 22: Tarjeta “Seleccionar Duración” durante tercer Sprint

Para este proyecto, se dividieron las tareas del backlog en los siguientes Sprints:

Sprint 1:

- El usuario, al andar en una bicicleta fija, podrá moverse en un mundo de realidad virtual.
- El usuario podrá elegir en qué dirección desea moverse al rotar la cabeza.

Sprint 2:

- El usuario podrá elegir cuándo iniciar el entrenamiento.
- El usuario podrá ver todas las opciones disponibles en un “Menú” (Inicio, Configuración -Música o Silenciar, Dirección Fija o Variable por Rotación, Dificultad Baja, Media o Alta, Duración por Tiempo o por Distancia, Sensibilidad de 0 a 4, Escenarios-, Entrenamientos Anteriores).
- El usuario podrá seleccionar la opción deseada dentro del Menú fijando la vista durante 1.5 segundos sobre el ítem correspondiente.
- El usuario podrá pausar el entrenamiento luego de estar 5 segundos en reposo.
- El usuario podrá Reanudar o Salir del entrenamiento cuando esté en Pausa.

Sprint 3:

- El usuario podrá elegir fijar la dirección antes de comenzar el entrenamiento para poder rotar la cabeza y ver los alrededores.
- El usuario podrá visualizar la cantidad de metros recorridos y el tiempo transcurrido.
- El usuario podrá seleccionar la dificultad del entrenamiento entre baja, media y alta. A medida que el nivel sea más alto, el entrenamiento durará más tiempo o más metros (de acuerdo al tipo de duración que haya seleccionado).
- El usuario podrá seleccionar la duración del entrenamiento en tiempo o en distancia.
- El usuario podrá seleccionar la sensibilidad que interpreta el software para ajustarlo a su capacidad. Al seleccionar entre los distintos niveles, deberá pedalear con menor o mayor intensidad de acuerdo al nivel seleccionado para que la aplicación detecte el movimiento.

Sprint 4:

- El usuario podrá elegir entre diferentes escenarios.
- El usuario podrá visualizar los entrenamientos realizados con anterioridad.
- El usuario podrá seleccionar escuchar música ambiental o silenciar (en cuyo caso deberá poder utilizar la aplicación que desee para escuchar otra cosa).

XP programming

Otra de las técnicas ágiles que se decidió utilizar es XP programming ya que se adecuaba a los requisitos del proyecto y a la forma en la cual estamos acostumbrados a realizar el trabajo.

User stories XP

Un artefacto XP que utilizamos para la realización del proyecto son las user stories. Esta técnica es utilizada para ver los requerimientos del proyecto y estimar su esfuerzo. Cada una aporta valor al producto final y se enfocan en el usuario, no en la forma en que serán implementadas. Cada tarjeta de usuario tiene asociada uno o varios casos de pruebas y tarjeta de tarea que indica quién realiza la user story, quien la prueba, quien la corrige y fechas de inicio y fin.

Prácticas XP

De las 12 prácticas de XP las que más se adecuaban a nuestra filosofía de trabajo son:

- Pequeñas entregas: Cada versión debe ser tan pequeña como sea posible, conteniendo los requisitos de negocios más importantes, y deben poder ser probados. Esta práctica se complementa con SCRUM ya que luego de cada sprint generamos un pequeño prototipo para probar y obtener feedback rápidamente.
- Diseño simple: Este proyecto posee una complejidad elevada en cuanto a la codificación, por eso es esencial mantener el diseño lo más simple posible para que funcione en todas las pruebas, no poseer lógica duplicada y mantener al mínimo el número de clases y relaciones ya que de lo contrario el producto será muy difícil de mantener. Esta práctica está muy bien aplicada en el cálculo de la velocidad de la bicicleta ya que es la lógica más compleja del proyecto.
- Refactorización: Al estar programando horas continuas el código pierde calidad, es por esto que al ver código mal ejecutado se debe rehacer para que sea más legible y performante. No se debe refactorizar por gusto si no sólo cuando el sistema, la complejidad y las nuevas características a implementar lo requieran.
- Programación por parejas: Para las partes del código donde se requería mucha precisión para que la performance sea elevada y no haya código inútil, o bien cuando la dificultad lo requería se programó con dos personas mirando una misma máquina, con un solo teclado y un solo mouse. Cada uno cumplía un rol, uno programaba mientras que el otro pensaba cómo mejorar y analizaba si había algún caso en el que falle el código. Siempre ambos utilizando el principio de diseño simple, sin duplicar código.
- Propiedad colectiva: Cualquiera que crea que puede aportar valor al código en cualquier momento puede hacerlo, ningún miembro del equipo es propietario del código.
- Integración continua: El código se debe integrar como mínimo una vez al día, y realizar las pruebas sobre la totalidad del sistema. Esto se realizó con Bitbucket utilizando SourceTree de Atlassian. El código de ambos se subió diariamente y se ejecutó el comando merge. Luego de esto se probó y se ejecutó el comando push

al "Master" donde se encuentra el código fuente final, siempre revisando el código generado por la otra persona en búsqueda de errores o formas de optimizarlo.

- Estándares de codificación: Se establecieron estándares de codificación para que el código sea legible por los dos integrantes. Por ejemplo se propuso realizar todo el código en Inglés y al nombre de cada variable anteponer la primer letra de su tipo, tal como "int iCounter".

Recomendaciones para próximas etapas

Métodos de monetización (A revisar mucho)

Existen varias formas de monetizar aplicaciones, modelos de pago hasta modelos totalmente gratuitos. A continuación se expondrán los modelos más utilizados analizando sus ventajas y desventajas, y se recomendará cuál utilizar.

Premium

Este modelo implica que la aplicación solo se pueda descargar posterior al pago de la misma. Los precios varían de 0.99 USD hasta 999 USD haciendo que cada descarga nueva genere ingresos. Hay que tener en cuenta que las dos tiendas más grandes de aplicaciones cobran un 30% de comisión por cada venta. Por eso si se vende la aplicación en 1 USD solo se recibirá 0.70 USD por cada venta. La clave para poder vender la aplicación con este modelo es diferenciarse de las aplicaciones parecidas que sean gratis ya que sino los usuarios elegirán estas últimas.

Los beneficios de este modelo son varios. El primero es que se gana directo por descarga, siendo este el modelo donde el dinero se obtiene más rápido. Por otro lado al ser paga, la aplicación no tiene publicidad, lo que lleva a una interfaz de usuario más limpia y sin publicidad que generalmente baja la percepción de la calidad del producto. Por último, los usuarios que pagan aplicaciones son más propensos a ser retenidos por la aplicación y, a mayor retención, se puede mejorar aún más la monetización.

La desventaja más grande del modelo premium es que pagar para utilizar una aplicación es una barrera muy alta para la mayoría de los usuarios ya que las tiendas poseen muchas aplicaciones gratis con funcionalidad parecida.

Gratis con publicidad

Siendo el modelo con más difusión entre aplicaciones, la clave está en acumular una gran base de usuarios y obtener información de los mismos. Luego, esta información se utiliza para mostrar publicidad enfocada específicamente en el usuario, lo cual da dinero. Lo mejor es utilizar alguna plataforma que haga esto ya que cuentan con muchas empresas las cuales quieren publicitar, como por ejemplo Unity Ads o AdColony o AdMob de Google.

Hay tres tipos de publicidad para incorporar:

- Los banners son publicidad estática fijada en una parte de la pantalla. Estos pagan por visualización y por click, pero tienen la gran desventaja de que al estar todo el tiempo visibles empeoran la interfaz de usuario.
- Los "Instertitials" son publicidades estáticas a pantalla completa que generalmente se muestran luego de finalizar algo en la aplicación, como por ejemplo al finalizar un entrenamiento.
- Por último están los videos publicitarios que son muy similares a los "Instertitials". La diferencia es que el video es dinámico y el usuario puede elegir verlo con un beneficio a cambio. Este tipo de publicidad es la que mejor paga y la que mejor se adapta al flujo natural de la aplicación ya que no se lo está obligando al usuario a consumir publicidad.

Sea cual sea el tipo de publicidad elegido, se debe medir cómo rinde ésta dentro de la aplicación. Para ello se utiliza la medida conocida como eCPM (effective cost per millie), es decir, la ganancia generada por la publicidad cada 1000 vistas. Este valor varía dependiendo de varios factores. El primero es el país en donde se ve la publicidad, por ejemplo Canadá o Estados Unidos tendrán un eCPM mucho mayor al de Argentina o Brasil. Otro factor es la cantidad de publicidad que se ve por persona; por ejemplo, si una persona ve varias publicidades en forma de video, el primer video pagará mejor que los siguientes haciendo que a más videos por persona el eCPM baje.

Las ventajas de este modelo es que no posee barrera de entrada para usuarios nuevos, y en caso de usar la publicidad de forma inteligente puede atraer más usuarios gracias a los incentivos por ver los mismos.

Las desventajas son que se necesita una base muy grande de usuarios que miren publicidad diariamente para que el modelo de negocio funcione y que la aplicación puede perder calidad debido a la excesiva cantidad de publicidad.

Freemium

El modelo freemium ofrece una aplicación gratuita con funcionalidad básica donde el contenido extra se desbloquea pagando con dinero. Generalmente el contenido extra bloqueado es lo más útil de la aplicación, y esto se hace para lograr que los usuarios quieran pagar.

Lo positivo del modelo es que inicialmente los usuarios pueden usar la aplicación sin necesidad de pagar, y en caso de retenerlos y querer más, compran el contenido adicional.

Lo difícil es decidir cuánto contenido gratis ofrecer, ya que si es mucho los usuarios no pagarán por el contenido extra, y si es poco no sabrán de qué trata la aplicación y no podrán ser convertidos a pagos. El balance entre esos aspectos es lo más importante en este modelo.

Compras dentro de la aplicación

Este modelo se basa en que la aplicación es gratuita pero ofrece compras dentro de ella de poco valor. La clave es saber integrar estas compras en el flujo de la aplicación.

Existen tres tipos de compras dentro de una aplicación:

- Consumibles: son aquellas compras que se pueden repetir, es decir una vez que se utiliza lo que dio la compra ya no se puede volver a utilizar a menos que se compre nuevamente. Un ejemplo muy común son mejoras temporales dentro de la aplicación.
- No Consumibles: son compras que se realizan por única vez y el contenido comprado queda desbloqueado para siempre. Un ejemplo de no consumible seria comprar una herramienta bloqueada en la aplicación, o un tema para cambiar la forma de visualizarla.
- Subscripciones: son compras que se renuevan periódicamente, generalmente cada vez que se paga cambia el contenido o se agrega más al existente. Este tipo de compras es el que se utiliza en aplicaciones de revistas y diarios aunque otros rubros de aplicaciones también lo están comenzando a adaptar. Es muy útil para usuarios que valoran la aplicación ya que es una compra fija periódica.

El mínimo precio de las compras dentro de la aplicación en las tiendas más grandes es de 0.99 USD, por lo que en caso de querer vender algo de menor precio lo que se suele hacer es generar una moneda propia donde los productos se vendan por esta moneda virtual generando así un nivel de abstracción entre lo que cuesta realmente el producto y la moneda ficticia. La desventaja de esto es que se debe controlar la cantidad de monedas por usuario y persistirlo en un servidor, ya que de estar en el cliente será de fácil acceso por los usuarios y se corre el riesgo de que lo alteren.

En este último tiempo, las compras dentro de la aplicación son la fuente de ingreso más grande en aplicaciones y juegos del Play Store y App Store. Este método de monetización trae bajo riesgo ya que los objetos que se venden son virtuales y no requieren logística ni nada que esté asociado con un producto físico.

Vender a un tercero

Es el último de los métodos de monetización y consiste básicamente en vender la aplicación a un tercero. En el caso de nuestra aplicación podría llegar a venderse a una cadena de gimnasios en donde cada bicicleta tendría su propio casco de realidad virtual con un celular con la aplicación instalada. Esto genera un diferencial para el gimnasio ya que es algo innovador que sus clientes no poseen. La desventaja es que es difícil llegar a negociar con un gimnasio sobre la compra de una aplicación, ya que además deberá invertir en dispositivos en los cuales ejecutarla.

Recomendación

En primer lugar se analizó utilizar videos luego de cada entrenamiento, pero esta opción se descartó ya que los videos no están preparados para ser vistos con el casco de realidad virtual y no son visualizados correctamente.

Por otro lado, se consideró que, para utilizar el modelo premium, la barrera que hay que superar es muy grande ya que los usuarios nunca han experimentado algo así y es posible que no se arriesguen a comprar algo que no conocen.

Finalmente, se cree que el modelo de negocio freemium con compras dentro de la aplicación es la mejor opción dado que es el modelo que más conviene para esta aplicación. Las compras serían distintos escenarios por los cuales navegar a un precio de 0.99 USD cada uno, dejando gratis un escenario de base para que todos puedan utilizar. Luego en una etapa

dos, se pueden utilizar las compras para vender objetos de customización de personaje o de la misma bicicleta.

De todas formas, no se descarta la opción de vender la aplicación a un tercero. De lograr el contacto, puede ser un buen negocio.

Registros de entrenamiento

Con los datos persistidos en la Web se puede armar un graficador que muestre el progreso de tiempo/dificultad/metros temporalmente y proponer una meta a alcanzar y así poder seguir progresando.

Por otro lado, la forma en la que fue desarrollado el entrenamiento permite guardar metro a metro la posición del usuario. Utilizando estos puntos, se podría trazar todo su recorrido y mostrarlo en un mapa de dos dimensiones.

Generación de terreno aleatorio

Se puede generar terrenos aleatorios con características realistas de varias maneras. Una de ellas es la función creada por Ken Perlin en 1983 para crear texturas por computadoras llamada “Ruido Perlin”. Este algoritmo si se utiliza en tres dimensiones puede generar terrenos pseudoaleatorios, así es como Minecraft y otros juegos generan sus escenarios.

Figura 23: Textura en dos dimensiones generadas con Ruido Perlin.

Además de generar texturas o terrenos esta función permite crear continuidad en objetos con movimiento como fuego o agua.

Un breve ejemplo de cómo funciona el código para tener terrenos navegables y lo más realista posible:

La función PerlinNoise posee dos parámetros float y da como resultado un tercer float:

Float PerlinNoise (float x, float z)

Desde código se crea un plano con N triángulos y se itera por los 3 puntos de cada triángulo con dos bucles anidados (mientras más triángulos, más detallado será el terreno). Por cada punto de cada triángulo se llama a la función “PerlinNoise” con su posición en ‘X’ y su posición en ‘Z’ respectivamente, pero antes se divide a estos dos por una constante de factor de relieve lo que hará que el terreno sea más homogéneo y suave para facilitar su navegabilidad. La función retornará la altura de este punto que es donde se debe posicionar en el eje Y al punto del triángulo por el que se está iterando.

Queda visualizado como la siguiente imagen:

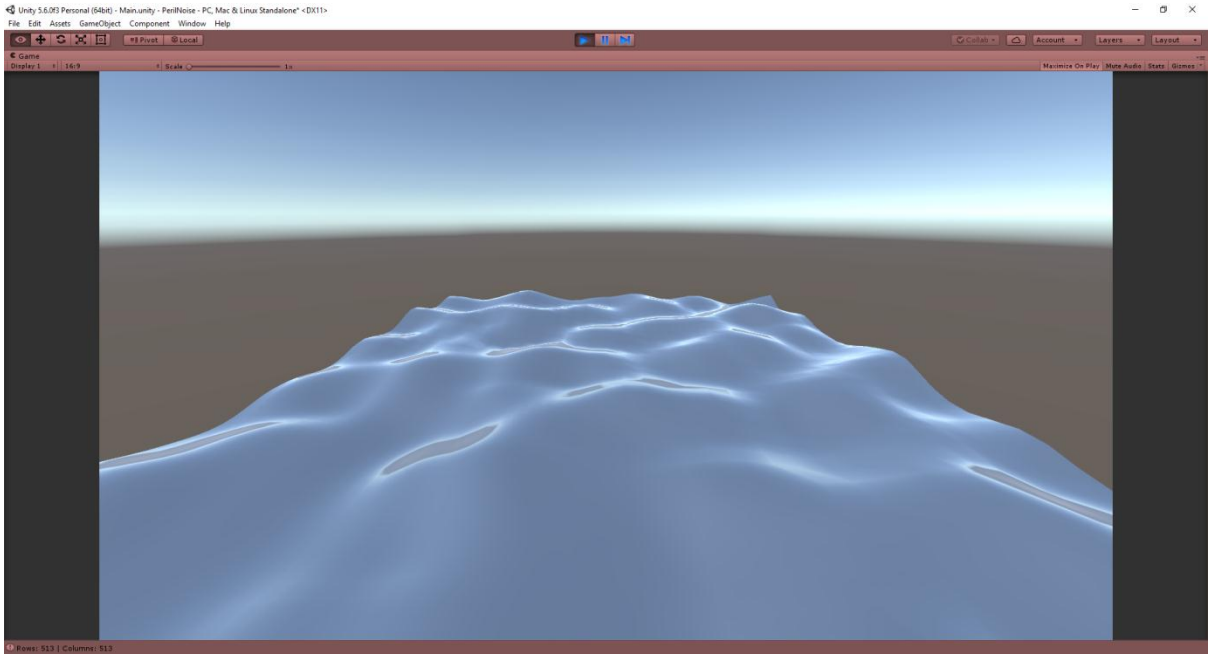

Figura 24: Generación de terrenos aleatorios creado por Franco Ares e Ivanna Mohovich.

La función “PerlinNoise”, al ser pseudoaleatoria, cada vez que se ejecute el código dará siempre el mismo resultado de no cambiarse la semilla. Esto se puede utilizar para persistir el terreno y no tener que gastar más capacidad que solo un entero que guarde esta semilla de inicialización, y en caso de querer generar un nuevo terreno se cambia ese entero por otro.

Utilizando como base este código se puede generar varios tipos de terreno y mezclarlos para tratar de simular diferentes biomas por los cuales recorrer, ya sea llanuras, montañas o playas.

Cómo influir para mejorar la performance física

Para hacer un análisis más preciso y detallado, se podría hablar con un deportólogo o un entrenador personal para evaluar distintas formas de mejorar la performance física de las personas.

De todas formas, se considera como general que la música, la variedad de terrenos, la visualización de progreso y la competencia entre jugadores puede ayudar en la performance de una persona.

Si bien varias de estas funciones fueron agregadas al prototipo, se podrían mejorar o llevar a un nivel más alto. Por ejemplo, en cuanto a la música se podría incorporar un reproductor de música que acceda a la música del dispositivo y que a través de ciertos gestos se pueda cambiar a la siguiente canción o bien pausar la reproducción. En cuanto a la variedad de terrenos, es posible que incorporar escenarios con distintas ciudades del mundo sea una manera de atraer a las personas a acceder a la aplicación. Por el lado de la visualización de progreso ya se ha nombrado mantener una aplicación web que muestre una barra de progreso de la persona, y tal vez hacer un análisis más detallado del entrenamiento realizado por ésta que permita, por ejemplo, visualizar la cantidad de calorías gastadas. Finalmente, una parte muy interesante sería agregar la competencia entre jugadores que será detallada a continuación.

Competencia entre jugadores

La competencia entre jugadores es un aspecto muy importante en la experiencia en el entrenamiento ya que lleva a los jugadores a dar su máximo rendimiento para ganarle a su competidor. Esto se puede llevar a cabo de varias maneras:

La primera es que ambos jugadores estén conectados en simultáneo viéndose entre sí en tiempo real. Para realizar esto se necesita un servidor, ya sea uno de los celulares de los competidores o uno externo. Esta experiencia es la más realista ya que se ve en tiempo real la posición del competidor y su velocidad, pero así también es la más costosa de llevar a cabo ya que se deben sincronizar en tiempo real las velocidades, posiciones y rotaciones de ambos

jugadores. Posee problemas de latencia ya que depende enteramente del ancho de banda del canal y en caso de desconexión del celular que está funcionando como servidor se pierde la partida y hay que realizar la recolección nuevamente.

Para este método se puede utilizar el Plugin de Unity de Google Play Games el cual permite realizar partidas de tiempo real sincronizando datos, donde uno de los dos usuarios actúa como “host” y el otro como cliente donde se envían datos bidireccionalmente. Enviando la cantidad de información mínima para funcionar se reduce notablemente la latencia para mejorar la sincronización. Los datos a enviar son la dirección, la posición y la velocidad de cada uno de los usuarios, se cicla por estos datos a intervalos de tiempo constantes y se interpola entre el valor anterior recibido y el último para mostrar con fluidez el movimiento de ambos jugadores.

La segunda es realizar partidas entre pares de manera asincrónica. Esto es el competidor A desafía al competidor B, pero primero A corre su carrera y luego B cuando lo disponga corre la suya. Cuando B finaliza su carrera se comparan los resultados y se determina quién fue el ganador. Esta experiencia disminuye la experiencia inmersiva pero es más sencillo de implementar y manejar ya que no hay problemas de latencia. Solo se compara el resultado final, ya sea metros o tiempo y el mejor es el ganador.

Por último se puede implementar una tabla de posiciones donde se muestre el mejor tiempo o la mayor cantidad de metros. Este es el más simple debido a que solo se guarda el mejor puntaje, que se reemplaza en caso de superarlo y luego se muestra en comparación al resto. Lo negativo es que la experiencia menos competitiva ya que se compite contra todos los que utilizan la aplicación y no existe la competencia personal de jugador contra jugador.

En caso de vender a un gimnasio, se podría o bien streamear la partida en vivo en una pantalla, o bien mostrar una tabla de posiciones entre los usuarios del gimnasio.

Análisis Final

Resultados de las pruebas

El resultado de la ejecución de las pruebas funcionales fue satisfactorio ya que ayudaron a verificar la correcta funcionalidad básica del prototipo desarrollado. Todas las historias de usuario definidas para esta etapa están contempladas en los casos de pruebas realizados y han superado correctamente la ejecución. Dicho esto, cabe aclarar que la

cantidad de casos de prueba definidos es sumamente inferior a la cantidad necesaria para asegurar el correcto funcionamiento de la aplicación productiva, si bien las pruebas informales ejecutadas durante los distintos sprints de desarrollo ayudaron a asegurar un mínimo funcionamiento para la utilización del prototipo.

Por otro lado, si bien se hizo un repaso de las pruebas no funcionales que habría que realizar, queda fuera del alcance de este proyecto la definición formal y cuantitativa de los requerimientos no funcionales, así como también la definición de las pruebas que se deben realizar y la posterior evaluación de las pruebas realizadas para verificar que igualen o superen el criterio de aceptación establecido. La única definición que se realizó y se alcanzó lo definido fue la de no bajar de los 60 FPS, debido a que una aplicación de realidad virtual es completamente inutilizable si se baja de esta barrera. Pero en caso de realizar modificaciones a este prototipo, será imperativo volver a probar este requerimiento.

Limitaciones

Gimbal Lock

Gimbal Lock se refiere a la pérdida de rotación sobre un eje en un sistema de coordenadas de tres dimensiones. Esto ocurre cuando dos de los ejes se superponen en la misma rotación, como por ejemplo al rotar 90 grados el eje Y mientras los otros quedan con rotación 0. Al suceder esto el sistema queda con una rotación bidimensional.

Solución

Existen dos soluciones para este problema. Por un lado, se puede utilizar un cuarto parámetro para almacenar la rotación, generando redundancia para así solucionar el bloqueo. Por otro lado, cuando se genere el Gimbal Lock se puede reiniciar la rotación a una posición arbitraria para reiniciarlo.

Gimbal lock en Unity

En el caso de Unity, este problema surge al utilizar ángulos eulerianos para manejar las rotaciones. Es por eso que se debe usar siempre la clase Quaternion que utiliza 4 parámetros para la rotación.

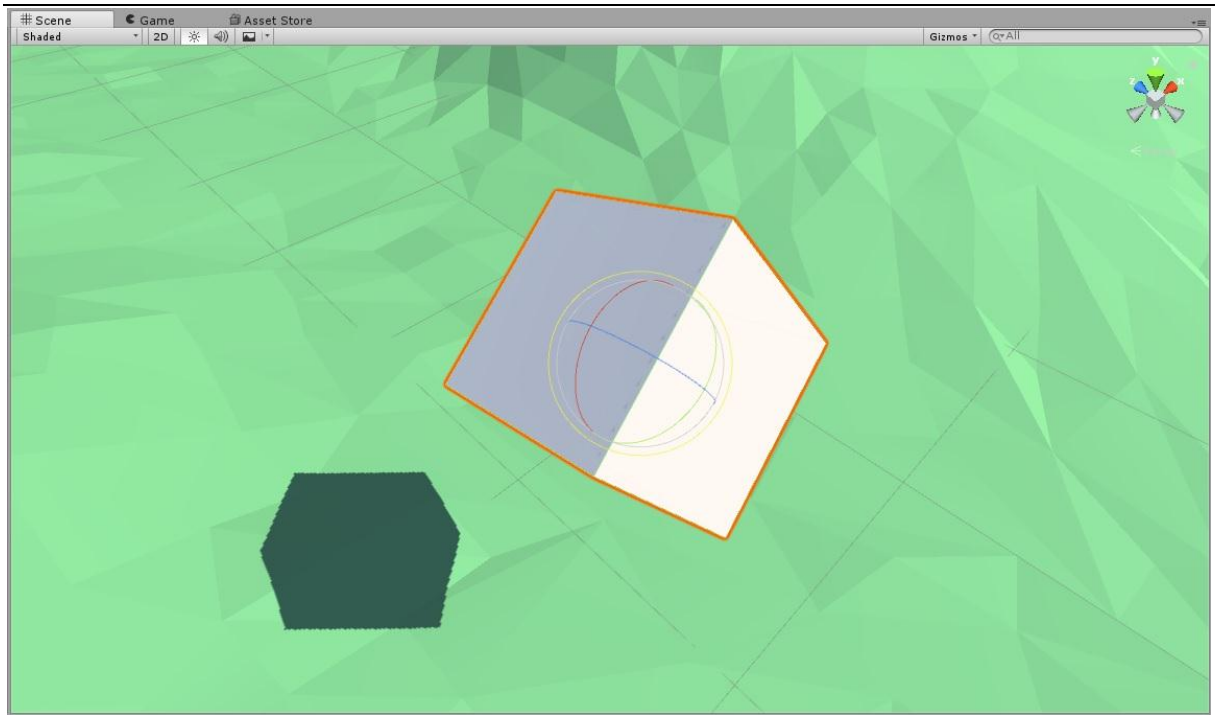

Figura 25: Editor de Unity utilizando 4 Gimbals para evitar Gimbal Lock.

Al inicio del proyecto, para realizar las rotaciones tridimensionales se utilizaron ángulos eulerianos. Al tomar conocimiento de este problema, se pasó todo a la clase Quaternion.

Motion Sickness

Como se mencionó con anterioridad uno de los problemas más relacionados con la realidad virtual es el concepto de motion sickness. Este problema surge cuando hay diferencias en el movimiento dentro del mundo virtual con el movimiento realizado en el mundo real, pudiendo causar náuseas en el usuario.

Este problema apareció en las primeras etapas de desarrollo del prototipo ya que las pruebas se realizaron con un Smartphone Motorola Moto G2, un dispositivo lanzado en 2014, por lo que su poder de procesamiento de video era notablemente menor que el de uno lanzado este año, y aún no se habían aplicado los conceptos de optimización nombrados con anterioridad. Ésto, sumado a que la persona estaba realizando actividad física, hacía que se vea con retraso la pantalla del Smartphone con respecto a la rotación realizada por la cabeza, lo que generaba un consiguiente malestar para la persona que lo estaba utilizando, incrementando aún más la sensación de mareo. Se pudo encontrar que la principal causa era la gran cantidad de objetos en pantalla que hacían caer los cuadros por segundo a 30. Al

implementar las optimizaciones se pudo llegar a 60 haciendo que el movimiento se vea fluido.

Conclusiones

Para concluir, se puede afirmar que el proyecto realizado cumplió con los objetivos principales planteados por los autores de analizar el movimiento de una persona andando en bicicleta fija, y de desarrollar un prototipo completamente funcional que sirva como base para desarrollos futuros que apunten a la construcciones de aplicaciones de realidad virtual que capten este movimiento para evitar el uso de accesorios externos.

El trabajo realizado logró generar una gran cantidad de documentación asociada a las diversas etapas del proceso de desarrollo de software que, en conjunto con el contenido de este documento, servirán para orientar a los desarrollos futuros sobre la temática, y facilitarán los trabajos de extensión e implementación de las funcionalidades descritas. Se invita a los futuros desarrolladores a trabajar sobre las distintas recomendaciones de extensión detalladas en la sección correspondiente para ofrecer una mejor experiencia de usuario y lograr así una aplicación lista para salir al mercado. Se hace hincapié también en la necesidad de probar exhaustivamente los diversos aspectos de calidad del producto antes de realizar su implementación en un entorno real, así como también realizar todas aquellas pruebas que se consideren necesarias para validar el correcto funcionamiento del producto final.

A su vez, el proyecto cumplió con el objetivo secundario de proveer a los autores de una experiencia práctica e integradora que permitió la unificación de los diversos conceptos, metodologías y prácticas propias de la ingeniería de software, estudiadas e incorporadas durante el desarrollo de la carrera de grado, en un trabajo académico único y completo, y enfocado en el logro de un conjunto determinado de objetivos de bien común, ajenos a cualquier tipo de beneficio económico.

Bibliografía

- Welcome to Virtual Reality Society [https://www.vrs.org.uk]
- From the Operating Room to VR and back through advanced medical applications [http://ovidvr.com/]
- Storyboard VRA Tool to Prototype VR Ideas [https://www.artefactgroup.com/work/storyboard-vr/]
- Prototype VR applications [https://www.kalloctech.com/vr.jsp]
- Bringing the arts to life through innovative VR technology. [http://blvrd.com/]
- Unimersiv is the largest platform for VR educational experiences. [https://unimersiv.com/]
- La Aplicación de Fitness Vescape para Bicicleta Estática [http://www.vescape.com/es/index.html]
- Journey cycling across Britain in virtual reality [http://www.cyclevr.com/]
- Cutting-Edge VR for the Commercial Market [http://www.virtuix.com/]
- Leading provider of VR locomotion solutions [http://cyberith.com/research/]
- Head mounted displays (HMD) [https://www.vrs.org.uk/virtual-reality-gear/head-mounted-displays/]
- Rotation and Orientation in Unity [https://docs.unity3d.com/Manual/QuaternionAndEulerRotationsInUnity.html]
- Normal map (Bump mapping) [https://docs.unity3d.com/es/current/Manual/StandardShaderMaterialParameterNormalMap.html]
- Heightmap [https://docs.unity3d.com/Manual/StandardShaderMaterialParameterHeightMap.html]
- Transparency

[https://docs.unity3d.com/Manual/StandardShaderMaterialParameterAlbedoColor.html]

- VR Optimization Tips from Underminer Studios [https://software.intel.com/en-us/articles/vr-optimization-tips-from-underminer-studios]
- Occlusion Culling y Frustum Culling [https://docs.unity3d.com/es/current/Manual/OcclusionCulling.html]
- View frustum culling [http://www.lighthouse3d.com/tutorials/view-frustum-culling/]
- Specular Map [https://docs.unity3d.com/Manual/StandardShaderMaterialParameterSpecular.html]
- Gimbal Angles, Gimbal Lock, and a Fourth Gimbal for Christmas [https://www.hq.nasa.gov/alsj/gimbals.html]
- Interaction in VR [https://unity3d.com/es/learn/tutorials/topics/virtual-reality/interaction-vr]