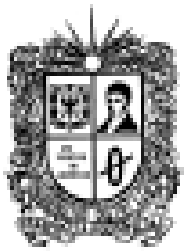
Diseño e implementación de sistema de navegación con uso de realidad virtual

Eder Ferney Duran Suarez

Código: 20101005101

Director Interno: Gustavo Adolfo Puerto Leguizamón

Director Externo: Christian Bodensiek



Universidad distrital francisco José de caldas

Facultad de ingeniería

Proyecto curricular de ingeniería electrónica

Bogotá D.C.

2017

Tabla de contenido

1. Objetivos

1.1. Objetivo General

Realizar implementación y acoplamiento de hardware y software, garantizando así el buen funcionamiento del dispositivo y la optimización en implementación para futuros proyectos con esta tecnología.

1.2. Objetivos Específicos

- Realizar una óptima integración de dispositivo de realidad virtual HTC vive con el Motor de desarrollo Unreal Engine 4, lo cual incluye la correcta instalación de drivers y demás software para dar soporte al dispositivo.

- Programar interacciones de usuario (mecánicas de movimiento).
- Realizar pruebas de optimización para garantizar una experiencia de alta calidad.
- Realizar pruebas con porcentajes de pantalla y manejar escalas de distorsión con el fin de asegurar una adecuada experiencia de usuario.
- Programar diferentes interacciones del usuario con el escenario como lo pueden ser apertura de puertas, manipulación de objetos dentro del escenario y demás.
- Implementar diferentes tipos de efectos sonoros con el objetivo de simular un entorno real.
- Programar e implementar diferentes tipos de inteligencia artificial que hagan del entorno un lugar más dinámico.

2. Resultados Alcanzados

A Continuación, se describen los resultados alcanzados en cada uno de los aspectos de la Práctica.

2.1 HTC VIVE y Unreal Engine

Antes de comenzar con el desarrollo y la integración fue de vital importancia poder entender de que estaba compuesto el hardware, su funcionamiento y por supuesto la utilidad que este tiene.

En resumen, el dispositivo de realidad virtual consta de un "headset", unos controles y dos estaciones base.

El "headset" es la ventana a la realidad virtual e interactúa directamente con nuestra visión. Estos tienen múltiples sensores que son seguidos por dos estaciones base. Además de ello cuenta con un sistema de calibración de los lentes dependiendo del IPD (Interpupillary distance) que es simplemente la distancia que hay entre el centro de la pupila de nuestros ojos.

El dispositivo cuenta con dos controles que se usan para interactuar con objetos en el mundo virtual. Al igual que el Headset, los controles también cuentan con sensores que son monitoreados por las estaciones base.

Las estaciones base son dos y mantienen un monitoreo constante de los sensores en el headset y los controles. La ubicación de las estaciones base juega un papel importante en la experiencia ya que de esto depende el área en el que se podrá interactuar. Fue importante entender y diferenciar las dos formas de realizar una experiencia en realidad

virtual ya que, por un lado, están las experiencias “room-scale” en las cuales se debe contar con un espacio considerable y libre de obstáculos. Este modo se caracteriza principalmente porque el espacio real es igual al espacio virtual y hay monitoreo no solo del movimiento de la cabeza y los controles sino también de la traslación del cuerpo. Por otro lado, la experiencia también puede ser” seated/standing”, la cual se caracteriza porque no necesita de un espacio grande ya que no se monitorea la translación del usuario y la experiencia se puede realizar con el usuario sentado. La distancia máxima que se debe tener entre estaciones base es de 5 metros y lo ideal es que estas superpongan el espacio de visión de la otra ya que de otra forma hay que conectar un cable de sincronización entre las estaciones base. Para la Practica no se usó este cable ya que las estaciones base se ubicaron una en frente de la otra.

Para crear experiencias inmersivas capaces de engañar al cerebro humano la realidad virtual requiere escenas con renderizado complejo y un alto rango de frames por segundo, es por ello que Unreal Engine resulto ser una excelente elección ya que ha sido diseñado para aplicaciones que demandan una alta calidad gráfica como videojuegos AAA, pre visualización y demás. Por otro lado, este motor cuenta con una interfaz sencilla para la integración además de brindar funciones pre programadas para facilitar el desarrollo de aplicaciones con realidad virtual.

2.2 Optimización

Existen ciertas consideraciones que se tuvieron en cuenta para asegurarnos que el proyecto corriera con una velocidad suficiente de cuadros por segundo. Con tal objetivo se enumeran ciertos ajustes que involucraron la configuración del motor de desarrollo.

- Cuando creamos el proyecto lo más recomendable, según la documentación, fue usar la siguiente configuración, “Target Hardware: Mobile/Tablet”, “Graphics: Scalable 3D or 2D”. Esta configuración deshabilita ciertas opciones de renderización que pueden llegar a ser demasiado pesadas como por ejemplo el “motion blur” y el “Antialiasing”.
- La opción “Instanced Stereo” es una optimización de CPU. Su propósito es reducir los hilos de renderizado y el costo de los manejadores de gráficos en la CPU cuando se renderiza en “estéreo”. Sin el “Instanced Stereo” habilitado se ejecutaría el renderizado de cada cuadro dos veces, uno para cada ojo. Es importante señalar que para que esta opción haga la diferencia tenemos que tener una escena gráficamente compleja, es decir, con muchos modelos en escena a la vez de otro modo esta opción no sería muy eficaz. En un principio no se notó diferencia alguna, sin embargo, en pruebas posteriores con más modelos en escena la diferencia comenzó a ser visible. Se puede habilitar la opción “Instanced Stereo” yendo a Edit ->Project Settings -> Rendering ->VR. Después de habilitar esta opción se debe reiniciar el motor de desarrollo y recompilar el sombreado.
- Por defecto Unreal Engine Utiliza “Deferred Renderer” ya que provee una mayor versatilidad y garantiza el acceso a más opciones de renderizado. Sin embargo, hay algunas ventajas que podrían no ser las más apropiadas para las experiencias con realidad virtual. Fue por ello que se optó por usar “Forward Rendering” ya que proporciona una línea de base más rápida con pases de renderización más rápidos lo cual permite un mejor rendimiento en realidad virtual.
- Muchas de las características avanzadas de post procesamiento están habilitadas por defecto sin embargo para el uso de realidad virtual estas debieron ser deshabilitadas para evitar que el proyecto tuviera problemas de rendimiento. Para

lograr esto se debió agregar un volumen de post procesamiento(PP.), una vez agregado lo seleccionamos y deshabilitamos la opción “Unbound”. Luego de esto expandir las demás configuraciones deshabilita todas las opciones de pp.

- Los mapas de normales no tienen el mismo impacto en realidad virtual por ello, para nuestro modelador 3D resulto mejor utilizar “Parallax Mapping”. Este método lleva el mapa de normales a un nivel superior mediante el monitoreo de las señales de profundidad que el mapeo de normales no hace. El “Parallax Mapping” puede mostrar mejor la información de profundidad, haciendo que los objetos parezcan tener más detalle de lo que en realidad tienen. Esto debido a que no importa desde que Angulo de visión, el “Parallax Mapping” siempre se auto corrige para mostrarte la información de profundidad correcta.
- Para la iluminación se usaron luces estáticas y “LightMaps” ya que son las opciones menos exigentes para renderizar. No fue necesario usar luces dinámicas, sin embargo, cuando tengan que utilizarse lo mejor es hacerlo en la menor cantidad posible y asegurarse que no se sobrepongan la una sobre la otra ya que hace aumenta el gasto en GPU.
- Utilizar LODs (Level of details) tanto como sea posible siempre es lo más recomendable. Cuando estamos cerca de un objeto se quiere que el usuario observe tanto detalle de este como sea posible, sin embargo, cuando el usuario se encuentra a una distancia considerable se puede cambiar el modelo por uno menos pesado y con menos detalle con el fin de optimizar la aplicación. Esta optimización no se usó de forma masiva ya que la escena de trabajo fue un espacio cerrado y reducido por ello se utilizó más que todo para dar detalle a objetos pequeños de decoración.

2.3 Experiencia De Usuario

La gran diferencia de la realidad virtual frente a otras tecnologías es el feedback que tiene el usuario del entorno virtual. Es evidente que hay una gran diferencia entre ver y estar en un entorno virtual por la sensación de inmersión. No es lo mismo vivir una historia a ser escuchada de otra persona. En el desarrollo nos dimos cuenta que es esencial que la calidad de los gráficos, las texturas, la iluminación sean lo suficientemente buenos como para que la experiencia no cause rechazo por parte del usuario. Buscamos la inmersión, y del realismo depende este grado de presencia. Lograr esto no fue para nada sencillo ya que la realidad virtual no solo debe ser en tiempo real e interactiva, sino que además su característica estereoscópica e interacción por movimiento ocular y movimiento de la cabeza hacen que las tasas de actualización sean de no menos de 90 frames(cuadros) por segundo por cada ojo.

La tasa de actualización en lo relativo al rendimiento computacional jugó un papel bastante importante para evitar sensación de mareo, parpadeos visuales y consecuentemente fatiga en la experiencia. Como ya se mencionó, para asegurar una experiencia de calidad es necesario lograr una calidad visual geométrica y lumínica muy sobresaliente sin embargo esto hace que la tasa de actualización sea más costosa computacionalmente, es por ello que se debió trabajar para lograr un equilibrio conjunto entre calidad visual y velocidad de refresco.

En la experiencia de realidad virtual y con la intención de lograr la inmersión deseada se hizo uso de los sentidos de la vista, la audición y moderadamente del tacto. Por otro lado, también existen los sentidos del olfato y el gusto que si bien influyen de manera notable

en la percepción del entorno por ahora no se tienen en cuenta en la experiencia, quizá en un futuro cercano se pueda llegar a involucrar estos también.

Con las primeras pruebas se llegó rápidamente a la conclusión que la primera persona es la fórmula que mejor funciona en cuanto inmersión además que influye para que la narrativa de la aplicación se adapte al usuario como protagonista de la experiencia.

Uno de los componentes más importantes del desarrollo de la experiencia en realidad virtual fue el sonido. Cuando se habla de sonido no solo se habla de la experiencia en narrativa sino a cómo interactúa con el usuario y complementa la experiencia de inmersión. El sonido fue importante ya que nos ayudó a obtener información del entorno y dar contexto en la aplicación. Es por ello que se le dio gran importancia durante el desarrollo. Si nos encontramos en un entorno 3D es importante que el sonido también parezca venir de un entorno tridimensional. Es por ello que se decidió usar sonido binural. El sonido binural se diferencia del stereo porque este permite diferenciar las fuentes de sonido de forma individual. En términos generales lo que hace es simular como escuchamos los humanos, replicando la distancia entre las orejas para que el espectador tenga una percepción real del foco de sonido dentro del espacio. Este jugo un papel importante durante el desarrollo ya que nos permitió no solo complementar la inmersión sino también interactuar más con el usuario.

2.4 Interacciones de Entorno

Cuando trabajamos en las interacciones de entorno quizá uno de los elementos más importantes a definir fue que información íbamos a recibir del usuario, es decir, cuantas entradas iban a tener los controles. Para ello lo que debemos hacer es definir estas entradas en el motor, darles un nombre y asignarles un botón especifico del control. Es importante

entender que existen dos tipos de entradas a configurar. Por un lado, están los “Action Mappings” los cuales al pulsar un botón activan el evento una sola vez sin importar si el usuario mantiene oprimido el botón, cuando el usuario suelta el botón en cuestión se activa una vez más el evento. Por otro lado, están los “Axis Mappings” los cuales son eventos que se ejecutan en intervalos muy cortos de tiempo dependiendo de si el usuario mantiene oprimido el botón.

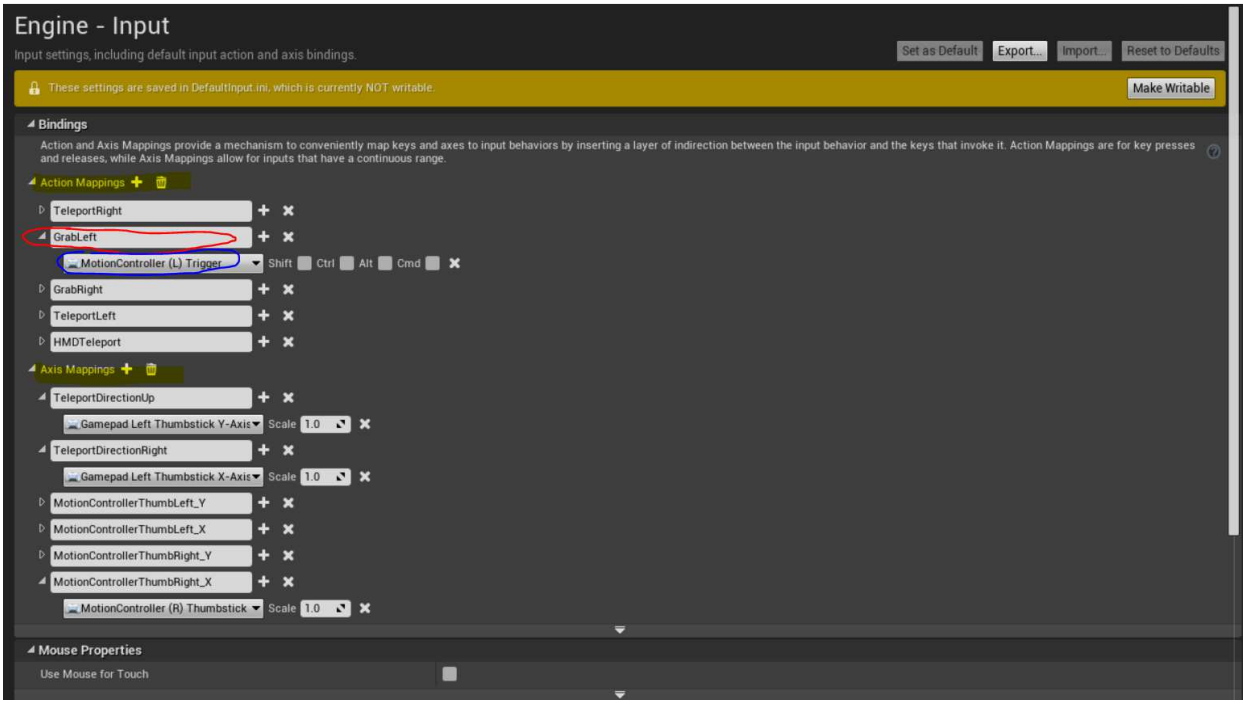


Figura 1. Declaración y configuración de entradas para la aplicación.

En la figura 1 se muestran resaltados los dos tipos de entradas mencionados anteriormente. Además de ello también se muestra encerrado con rojo el nombre que se le da a la entrada. Cabe señalar que este nombre es arbitrario, nosotros podemos elegirlo a nuestro antojo. Más abajo y encerrado en azul está el botón con el que se mapea la entrada. Si desplegamos las opciones encontraremos que podemos mapear cada uno de los botones que vienen integrados en nuestros controles.

Es importante señalar la importancia de los “blueprints” ya que con este sistema se programó gran parte de la aplicación. Los blueprints en Unreal Engine, es todo un módulo de programación grafica que agiliza y facilita el proceso de programar muchas funcionalidades en aplicaciones y juegos. Los blueprints nos permiten lograr casi lo mismo que logramos con la programación en código nativo con el diferencial que el código nativo se ejecuta más rápido, sin embargo, esta ventaja se ve reflejada en aplicaciones muy complejas que requieren mucha ejecución de código. Los blueprints nos permiten crear variables, funciones, Clases además de usar operaciones de control como los IF Else, For loops, switch y demás. A continuación, se, mostrarán algunos apartes con la programación en blueprints para el desarrollo de las interacciones.

Para el movimiento se usaron indicadores de tele transportación, como se ve en la figura 2. De esta forma el usuario con el movimiento del control y oprimiendo sus botones de dirección puede observar y elegir a qué lugar quiere tele transportarse.

Figura 2. Proyección de señal para desplazamiento de usuario.

A continuación, Veremos un fragmento del blueprint responsable del movimiento. Debido a que El informe no pretende enseñar programación en blueprints no se ahonda mucho en la explicación sino simplemente se da un panorama general.

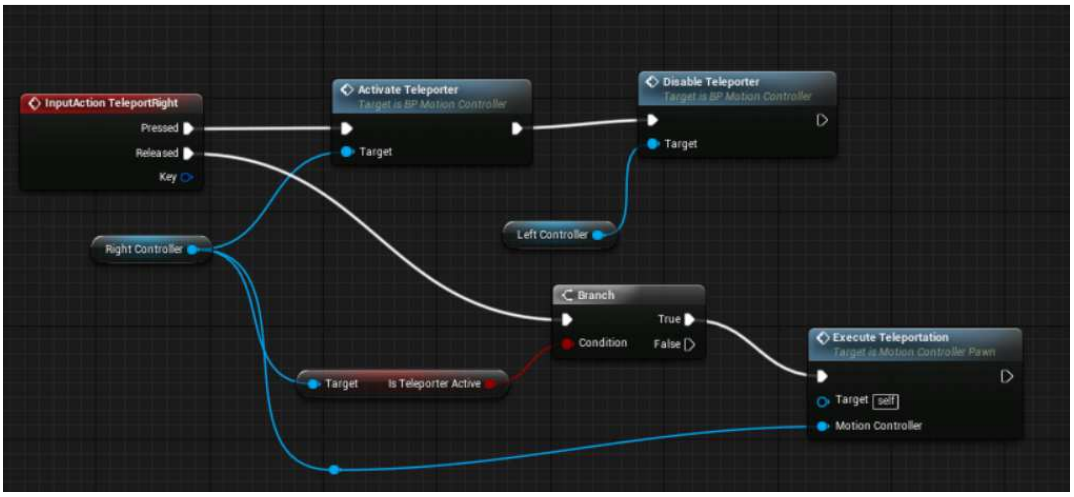


Figura 3. Blueprint que se ejecuta con la acción "TeleportRight" definida en la configuración de las entradas

Como vemos en la figura 3, la caja de color Rojo representa un evento definido anteriormente en el mapeo de las entradas. Este evento en particular se llama "Teleport Rigth". Como vimos, este evento es de tipo "Action Mapping" por lo cual se activa dos veces. Una vez cuando es oprimido y otra vez cuando se suelta. De la caja se desprenden dos hilos, uno llamado "pressed" y otro llamado "released", los cuales representan lo anteriormente dicho. Como vemos estos hilos conectan con unas cajas azules las cuales representan métodos propios de las clases "Right Controller" y "left Controller". Por último, encontramos una caja gris o "Branch", la cual funciona como un If- else en programación.

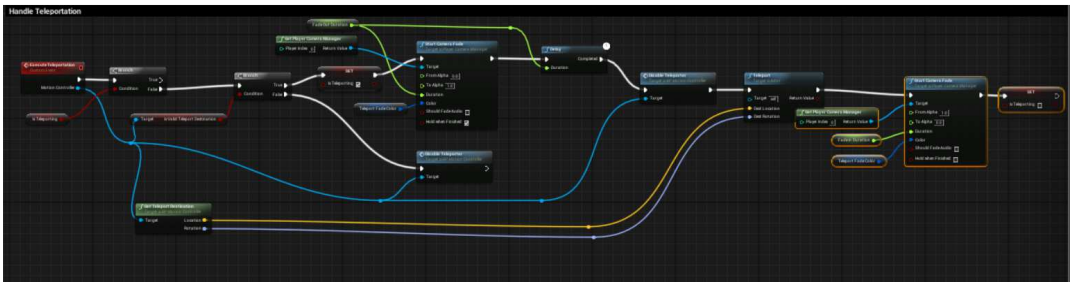


Figura 4. Definición de la función "Teleportation"

En la figura 4 esta la definición del método “Execute Teleportation”, que es donde se evalúa si la posición a tele transportar es válida y donde se hace efectiva la translación de la cámara.

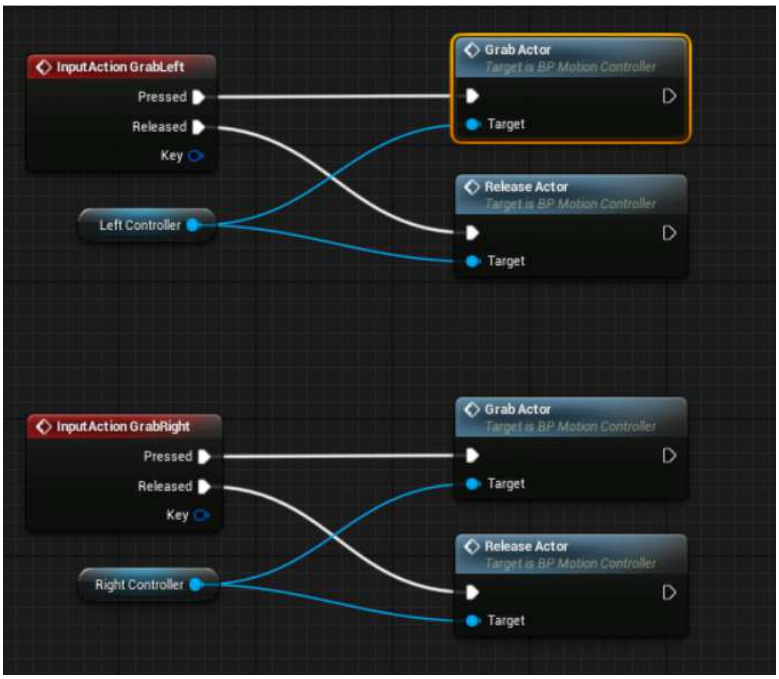


Figura 5. Blueprint que se ejecuta con las acciones “GrabLeft” y “GrabRight” definidas en la configuración de las entradas

En el blueprint mostrado en la figura 5 es donde comienza la programación para la manipulación de objetos. Como podemos ver están presentes otros dos eventos definidos en el mapeo de las entradas definido anteriormente seguidos por unas funciones definidas para la manipulación de objetos con cada mano. En la imagen de abajo se ve un apartado de como el usuario puede interactuar con los objetos, de forma tal que puede manipularlos y explorar en detalle cada parte de su modelado.

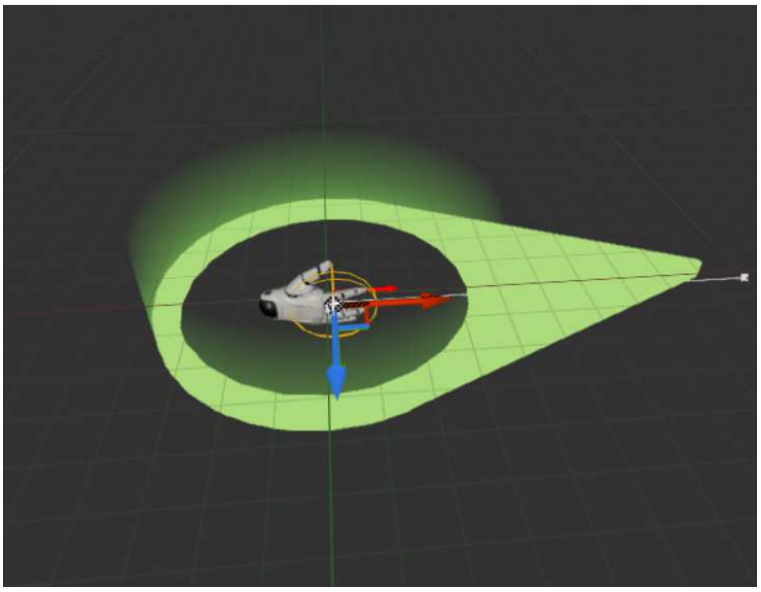


Figura 6. Volumen de Colisión para la detección de objetos.

La detección de los objetos se realizó colocando un volumen de colisión, como se puede observar en la figura 6. Unreal Engine cuenta con volúmenes de colisión los cuales sirven para detectar colisiones con otros objetos en el escenario, en la imagen podemos ver un volumen de colisión en forma de esfera, Cuando este volumen toca otro objeto inmediatamente dispara un evento el cual le dice al programa que puede agarrar el objeto y además obtiene la referencia del objeto que se quiere manipular. El hecho de poder manipular objetos en el entorno hace que sea una experiencia mucho interesante y abre incluso la posibilidad de desarrollar muchos tipos de aplicaciones no solo de visualización sino también de capacitación y demás aplicaciones que requieran un nivel alto de interacción.

Figura 7. Prueba de usuario manipulando una silla.

2.5 Experiencia Sonora

Para La implementación del sonido el motor Unreal Engine cuenta con clases pre-programadas que nos facilitaron de cierta forma la implementación de estos. Una de estas clases es el "Ambient Sound Actor". Esta clase fue usada con varios propósitos como sonidos de ambientes cíclicos. De esta clase se pueden sacar objetos, los cuales fueron ubicados en el entorno 3D, de tal forma que su sonido se atenúo dependiendo de la distancia del foco. Esta clase cuenta con algunas propiedades de atenuación que nos ayudaron a adaptar los sonidos a la experiencia que queríamos brindar. La atenuación es la habilidad de disminuir el volumen a medida que el usuario se aleja del emisor.

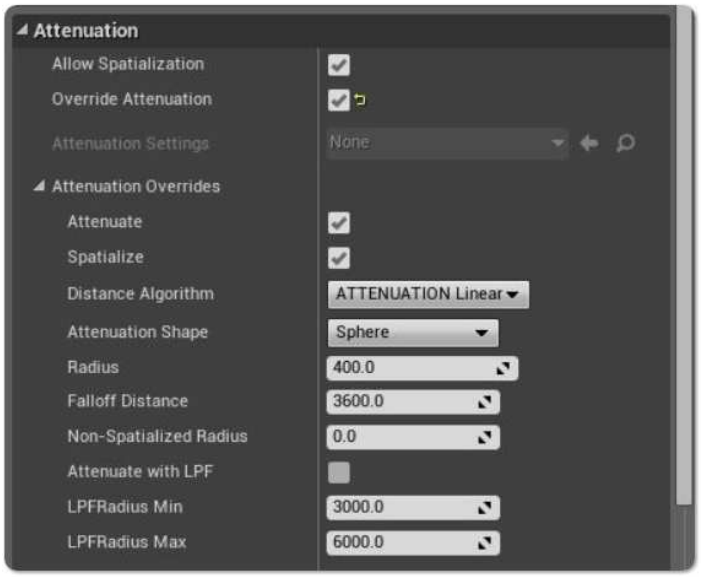


Figura 8. Opciones de atenuación de la clase "Ambient Sound Actor".

En las opciones de atenuación se tuvo la posibilidad de elegir la forma en que el sonido se atenúa. La que más se acomodó a nuestras necesidades fue una atenuación esférica la cual además cuenta con dos propiedades muy interesantes, un radio mínimo y un radio máximo. En cuanto más cerca estamos del radio mínimo más alto es el sonido y en cuanto más nos acercamos al radio máximo más bajo es el sonido, todos estos ajustes se realizaron gracias a un algoritmo de distancia hecho por los desarrolladores del motor. Además de las opciones de atenuación también contamos con la posibilidad de modificar algunas propiedades de modulación las cuales permiten tener control sobre los niveles de volumen y tono.

Figura 9. Opciones de Modulación de la clase "Ambient Sound Actor".

Por otro lado, Unreal cuenta también con una clase llamada “Sound Cue”. Esta clase nos permitió ejecutar sonidos que son llamados desde código en momentos clave de la aplicación. De esta clase podemos obtener objetos que pueden ser colocados en escena. Este objeto nos permite aplicar modificaciones a nuestro archivo de sonido en tiempo de ejecución. Esto significa que podemos ejecutar modificación del sonido tales como su modulación, oscilación, atenuación, efecto doppler y otras opciones disponibles en su configuración.

2.6 IA (Inteligencia Artificial)

La inteligencia artificial se podría definir como un agente racional flexible, que percibe su entorno y lleva a cabo acciones que maximizan su posibilidad de éxito en alguna tarea. En un principio se planteó el desarrollo de un agente artificial que guiara la experiencia del usuario sin embargo se encontraron algunos obstáculos para este desarrollo, Se indagó bastante en el desarrollo de este y lo que se logró se muestra a continuación.

Unreal Engine como se mencionó en un principio, es un motor de desarrollo que está pensado principalmente para el desarrollo de videojuegos de consola y PC. Este tipo de juegos hacen un uso extenso de la inteligencia artificial, es por ello que el motor tiene preparadas algunas herramientas que facilitan el desarrollo de una IA.

Uno de los principales objetivos de la IA es el movimiento. El sistema que se utilizó para lograr tal objetivo se denomina “path finding”. La forma en que este funciona es obteniendo la posición dentro del espacio que es alcanzable. Estos puntos de inicio y puntos de llegada alimentan una función que calcula la ruta más cercana entre los puntos.

El algoritmo usa información de la posición relativa, de forma que determina si está bloqueada por un objeto o un actor en el mapa y marca la posición como alcanzable. Esto es bastante útil cuando se encuentran objetos dinámicos en el mapa. El núcleo del sistema "Path Finding" es el algoritmo que calcula la distancia más corta entre dos puntos alcanzables. Este se denomina algoritmo de Dijkstra debido al científico de computación llamado Edsger DijstKra, quien lo creo originalmente. Existen variedad de algoritmos para encontrar la ruta más corta, pero el más usado es una variación del algoritmo de DijstKra denominado el algoritmo *A.

Otro componente Importante para la implementación del movimiento de la IA fue el "Navigation Mesh". Este sirve para indicarle al motor cuales son los puntos alcanzables en el mapa.

Figura 10. Escena con implementación de "Navigation Mesh".

Como se puede observar en la figura 10, cuando Hacemos uso del el "Navigation Mesh" este nos podrá indicar con color verde la superficie que podrá ser recorrida por la IA.

Otro componente importante fue el “AI Controller”. Esta es una de las clases que ya vienen definidas por el motor y sirve para controlar a la máquina y sus movimientos. Una de las principales ventajas de esta clase es que viene con algunas funciones ya programadas, como la función “MOVE TO LOCATION”, la cual tiene entradas muy interesantes que nos permiten decidir el radio de aceptación sobre el punto objetivo y la opción de habilitar el uso del “path Finding” entre otras más opciones. Además, esta función se puede llamar desde el sistema de blueprints como se muestra en la siguiente imagen.

Figura 11. Representación en Blueprints de la función “Move to Location”.

Como vemos en la figura 11, todos lo pines al lado izquierdo de la caja representa las entradas de la función y al lado derecho están las salidas o lo que retorna la función.

Otros dos componentes que fueron muy importantes para el trabajo con inteligencia artificial y que además trabajan muy de la mano son el “Behavior Three y el BlackBoard”. Se podría decir a grandes rasgos que el “Behavior Three” es el mecanismo para la toma

de decisiones y que el “BlackBoard” funciona como la memoria que ayuda a tomar estas decisiones.

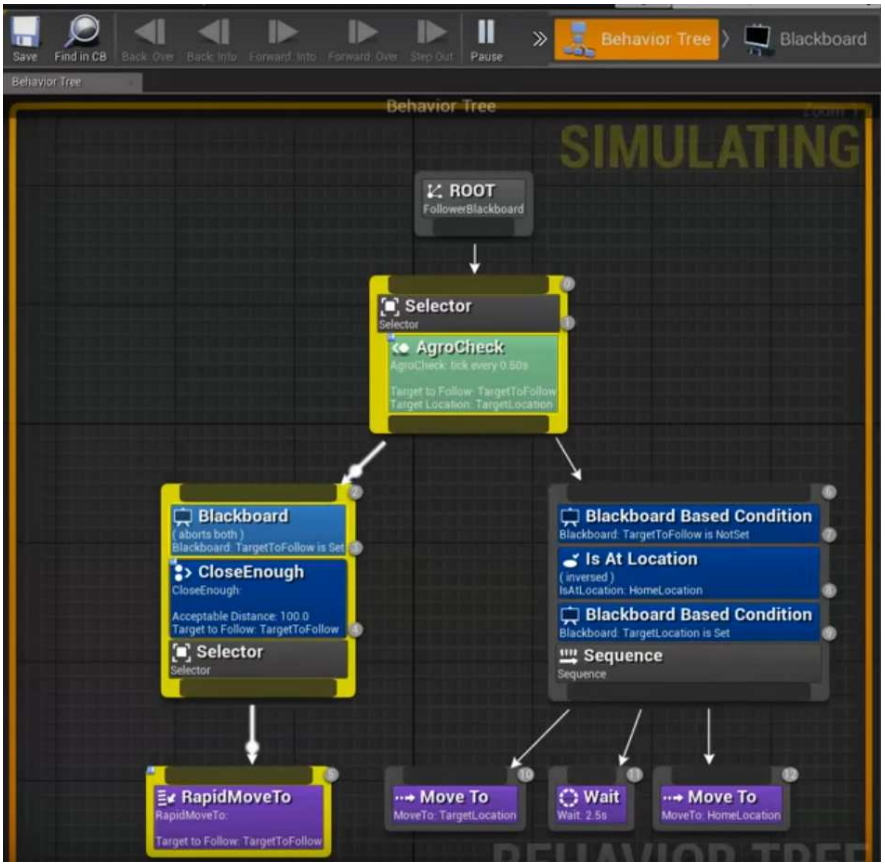


Figura 12. Árbol de comportamientos “Behavior Tree” implementado en las pruebas con la IA.

La Figura 12 muestra un árbol de comportamientos bastante sencillo donde por un lado puede realizar la detección de un objetivo y por otro lado puede ir a una posición aleatoria. Además de ello una característica muy interesante es que cuando se ejecuta la aplicación se puede ver que parte del árbol se está ejecutando y de esta forma analizar mejor el comportamiento de la IA y corregir errores de implementación.

Existen otro gran número de herramientas que ayudan al perfeccionamiento de la IA sin embargo estas herramientas y las ya mencionadas están diseñadas para el desarrollo de un IA que se acomode al comportamiento de un juego. Es decir, El motor facilita el diseño

de IAs que interactúen con un avatar y otros entes virtuales de forma espacial mas no de forma lingüística que era lo que se pretendía desarrollar. Finalmente se optó por prescindir de esta característica ya que no aportaba en gran manera a la experiencia y contaba con algunas imprecisiones.

3. Análisis De Resultados

3.1 Integración del Hardware

El dispositivo de realidad virtual (HTC VIVE) requirió de algunos controladores que se encuentran directamente en la página de vive. Además de ello fue necesaria la instalación del software steamVR para que el Computador pudiese tener un reconocimiento del dispositivo. Este software posee unos módulos de configuración y calibración ya sea para el uso en el modo "room-scale" o para su uso en el modo" seated/standing". Para la iniciación del dispositivo fue necesaria la previa instalación de las estaciones bases en las posiciones correctas. Es importante siempre disponer estas estaciones por encima de la cabeza y sobreponiendo sus espacios de cobertura a una distancia no mayor a 5 metros. Para el Uso del "HeadSet" es importante realizar una correcta calibración de los lentes ya que la experiencia puede resultar pobre y de baja calidad con una calibración incorrecta.

La práctica se realizó en un computador con las siguientes especificaciones:

- Windows 10 Home
- Intel Core i7-5700HQ @2.7GHZ
- 16GB RAM

- TG GeForce GTX 970m

Lo cual entra en las especificaciones mínimas para el uso del dispositivo de realidad virtual sin embargo la maquina se vio forzada y el framerate no fue el Óptimo en todo momento. Las HTC VIVE posee dos pantallas de 1080x1200 pixeles y una frecuencia de refresco de 90 Hz, lo cual hace que la cantidad de datos que se transportan hacia el PC sean un tema considerable y que la frecuencia de refresco se convierta en el mayor inconveniente. A continuación, en la figura 13, se muestra los resultados de una prueba de rendimiento al pc donde pudimos ver una gráfica de la fiabilidad del sistema, así como la cantidad de frames evaluados y si hubieron frames por debajo de la frecuencia recomendada.

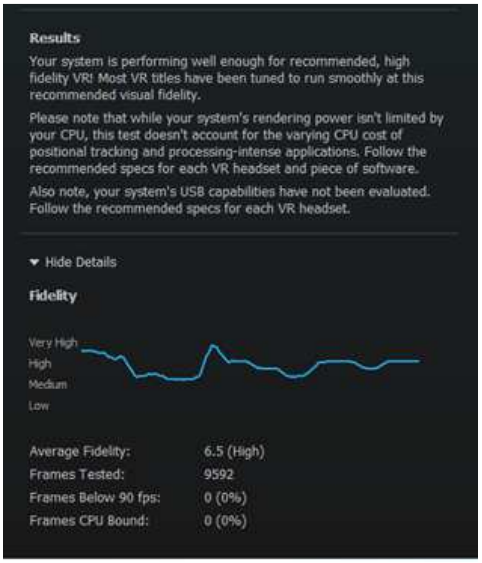


Figura 13. Resultados prueba de rendimiento usando el software SteamVR Test.

La anterior prueba de rendimiento fue posible gracias al software steamVR Test que a grandes rasgos mide si un PC es óptimo para el uso de realidad virtual. El resultado de la prueba muestra una alta fidelidad sin embargo advierte que no tiene en cuenta el gasto en CPU para el seguimiento de los sensores y demás tares de procesamiento, finalmente

recomienda realizar los ajustes de optimización necesarios para asegurar la mejor experiencia.

3.2 Resultados de optimización

Como se vio en la anterior sección se hicieron algunos ajustes con el fin de optimizar el rendimiento en el proyecto. A continuación, se muestra el análisis y lo que se observó a raíz de algunos de estos ajustes.

Como ya se ha mencionado, al trabajar con realidad virtual la consistencia en la frecuencia de los “frames” (cuadros) es quizá lo más importante a la hora de hablar de optimización y rendimiento. Con esto en mente es necesario dejar un buffer para asegurar que el refresco de cuadros siempre este sobre el promedio. El hilo de la aplicación está a solo un “frame” (cuadro) del hilo de renderizacion en todo momento. La GPU y el hilo de renderizacion están sincronizados en el mismo “frame”. Se tiene ligeramente menos de 11ms por la sobrecarga de proyección. Teniendo en cuenta esto, puede haber demora en los “frames” debido a una sobrecarga en el hilo de la aplicación por código o demora en el hilo de renderizacion debido a la sobrecarga de “draw calls” (Objetos que son dibujados o renderizados en escena) o en la GPU debido a el procesamiento de luz y sombras.

En la figura 14 se muestra una prueba que muestra los tiempos generales de los diferentes hilos de procesamiento en la aplicación. Estos datos se visualizan en la esquina derecha de la imagen y se muestran introduciendo el comando “Stat Unit”

Figura 14. Prueba en hilos de procesamiento usando el comando Stat Unit.

Además de las pruebas anteriores también se pudo examinar más a fondo el comportamiento de la GPU ingresando el comando “stat gpu”. Estas características son acumulativas de forma que se pueden observar las características generales sin ir a indagar objeto por objeto. Por ejemplo, la característica “shadow-Projection” es la suma de todas las sombras proyectadas por todas las luces.

Figura 15. Pruebas de comportamiento en la GPU haciendo uso del comando stat gpu.

Todas estas estadísticas que se muestran en tiempo real son muy útiles para identificar en que momento de la experiencia la aplicación se puede tornar más lenta y como contrarrestar y optimizar estos fallos. Es común que el mayor costo en el tiempo de los frames se deba al gasto en GPU, es por ello que es importante saber analizar estos datos.

Además de estas herramientas propias del motor que nos permiten evaluar rendimiento en la aplicación también hay una herramienta muy importante desarrollada por HTC que es el "SteamVR Frame Timing Tool". Esta herramienta nos permite observar como el proyecto de realidad virtual se comporta tanto al ejecutarlo con el motor y también cuando ya se exporta como aplicación. Esta aplicación nos ayudó a verificar los tiempos de CPU y GPU, excluyendo cualquier "Throttling" causado por la aplicación (Cabe señalar que un "Throttling" es un proceso usado para disminuir la velocidad del sistema temporalmente cuando este está estático por un tiempo ayudando a la CPU a mantenerse estable y ahorrar batería en el pc).

A continuación, en la figura 16 se muestra un gráfico mostrado por el "SteamVR Frame Timing Tool" y su análisis.

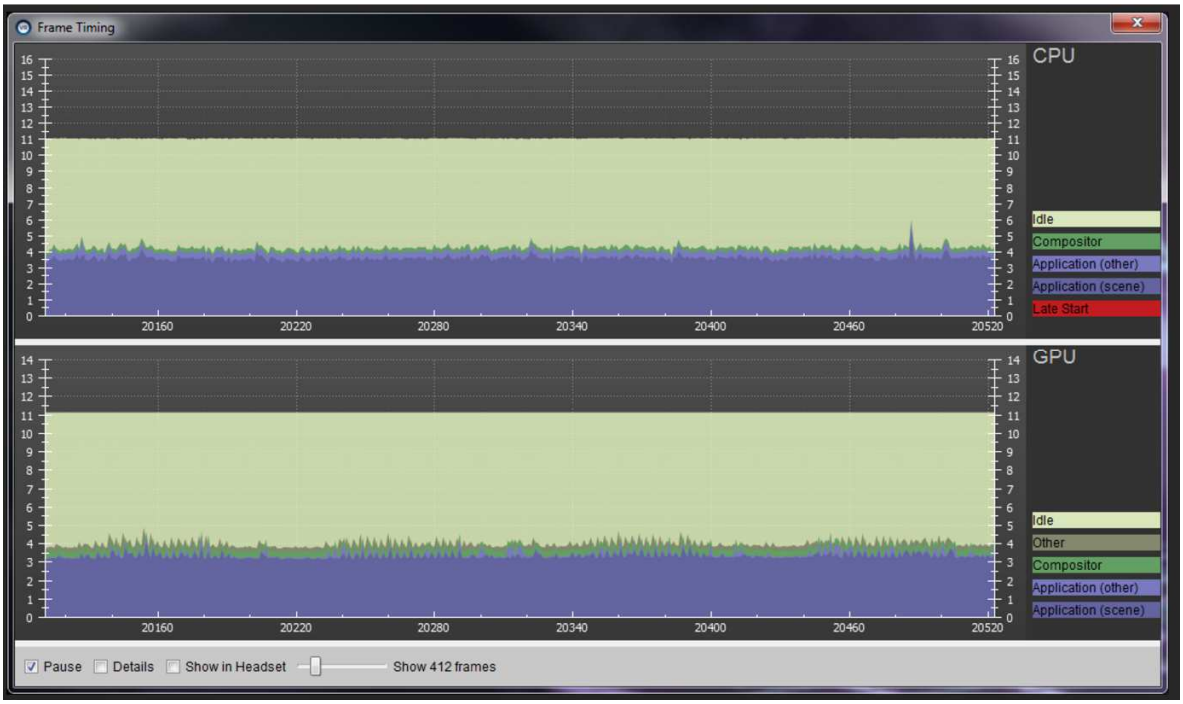


Figura 16. Resultados prueba de rendimiento usando el software SteamVR Frame Timing Tool.

La sección azul muestra los tiempos gastados por la aplicación y esta a su vez se divide entre "Application Scene" y "Application (other)". La sección que se refiere a "Scene" es la cantidad de trabajo realizado por el proceso "WaitGetPoses". La aplicación llama al "WaitGetPoses" para obtener las posiciones usadas para renderizar la cámara y otros objetos, renderiza la imagen del ojo derecho y del ojo izquierdo y finalmente envía las texturas no distorsionadas para que el compositor se muestre en la ventana. Por otro lado, la sección "Application (other)" es el tiempo gastado en el renderizado de los componentes visuales que acompañan la aplicación. El tiempo en CPU no captura el trabajo de tareas paralelas que se estén realizando.

3.3 Análisis De La Experiencia De Usuario

En términos generales se puede decir que la experiencia de usuario que se logró fue satisfactoria. Se logró implementar la manipulación de algunos objetos lo cual resulto bastante interesante para la mayoría de usuario de prueba. Además de ello se evidencio el interés del usuario por recorrer cada espacio del entorno virtual, lo cual es uno de los objetivos principales al realizar estos tipos de experiencias. Por otro lado, se logró una buena implementación sonora a pesar de los pocos recursos encontrados ya que la producción de los recursos sonoros puede llegar a incrementar de gran manera los costos del proyecto. En cuanto al balance entre la calidad visual y el rendimiento en refresco de pantalla se puede decir que se pudo trabajar más en los modelos sin embargo se consideró que fueron lo suficientemente buenos para lograr una experiencia de inmersión adicionando además que hubo limitaciones en Hardware. Uno de los inconvenientes fue la dificultad de los usuarios por entender los controles y adaptarse a la forma de desplazamiento e interacción con objetos. Es por ello que para el futuro se planea realizar tutoriales explicativos con el objetivo de facilitar el uso de estos para el usuario final y de esta forma disminuir su curva de aprendizaje y adaptación.

3.4 Resultados En Interacciones De Entorno

El trabajo acá presentado no requirió de interacciones muy complejas con el entorno, pero aun así estas generaron una muy buena retroalimentación de los usuarios. El ser humano tiene desarrollado el sentido del tacto y cuando levantamos objetos no solo podemos sentir su textura sino también su peso y de esta forma el cerebro obtiene información muy importante sobre el objeto. Capturar estas sensaciones aún resulta bastante complejo ya

que no se cuenta con la tecnología para lograr este objetivo sin embargo por las experiencias realizadas el poder levantar y examinar objetos dejando su peso y textura de lado resulta bastante cómodo para el usuario.

Por otro lado, la forma de desplazamiento en el espacio resulto no ser la más apropiada, ya que, por instinto, si queremos desplazarnos en un espacio lo que hacemos es caminar. Cabe aclarar que la captura de movimiento y desplazamiento del cuerpo ya es posible, en el momento resulta un poco engorroso ya que por un lado debemos contar con un espacio de trabajo amplio y por otro lado el cableado resulta ser un obstáculo, ya que el "headset" debe ir siempre conectado al computador para la transmisión de la información y estos cables al ser tan largos hacen que el usuario por momentos se enrede y la experiencia no resulte satisfactoria. Quizá en algún momento estos datos se puedan transmitir inalámbricamente y será en ese momento entonces donde la mejor forma de desplazarse en el espacio virtual sea caminando, también está la posibilidad incluso de realizar la instalación de unos rieles que permitan un desplazamiento cómodo para el usuario.

3.5 Implementación De Experiencia Sonora

Ramani Duraiswami, profesor de ciencias de la computación de la universidad de Maryland dice: "Hay un pequeño mapa en cada cerebro, incluso cuando no se está observando ningún objeto. Si el sonido es consistente con la geometría, tú sabrás automáticamente donde están las cosas incluso si estas no están en tu campo visual." La gran promesa de la realidad virtual es crear una alternativa a la realidad, pero sin el audio correcto que encaje con lo visual el cerebro no podrá crear esta ilusión.

Una de las conclusiones a las que se llegó es que una experiencia puede ser gráficamente muy inmersiva, pero el uso de sonidos marca una línea diferencial grande para el usuario, se notó ya que se hicieron pruebas antes y después de realizar la implementación sonora. Es importante además el uso de buenos auriculares, capaces de reproducir los diferentes tonos de audio y aislar al usuario del sonido externo.

Un aporte importante por parte de los usuarios fue que el hecho de no escuchar apropiadamente el sonido de los objetos con los que interactuaban, más específicamente de las puertas, quitaban para de la experiencia. Es por lo tanto importante entender en que interacciones el sonido toma gran importancia para el usuario y de esta forma trabajar en ello para hacerlo lo más exacto posible de forma que también este sincronizado al momento exacto del comienzo de la acción y el fin de esta.

3.6 Logros y Fallos En La Implementación De IA (Inteligencia Artificial)

Por motivos de tiempo y costo no se logró implementar la inteligencia artificial que en un principio se pretendía. Si bien es cierto que el motor de desarrollo cuenta con múltiples herramientas que ayudan al desarrollo, estas herramientas están más orientadas a que la maquina tenga conocimiento del entorno y de esta forma interactuar con este, sin embargo, el objetivo para este proyecto era elaborar una inteligencia artificial que más allá de reconocer e interactuar con el entorno, fuera capaz de interactuar con el usuario de una manera lingüística.

El reconocimiento de voz es un tema muy interesante pero no es para nada sencillo. Hoy en día existe software que entrega un rendimiento en promedio regular, lo cual significa

que necesitas hablar fuerte y asegurándose de que vocalices bien cada palabra de forma que el software pueda reconocerlo.

Se investigó sobre el tema y una de las opciones que se encontró para facilitar y agilizar un software para el reconocimiento de voz fue utilizar el SDK (Software development Kit) SAPI. Este es una API (application programming interface) de Microsoft que ayuda a reducir de gran forma el código requerido para realizar un reconocedor de voz, haciendo esta tecnología más accesible y robusta para un amplio número de aplicaciones. Esta API provee una interfaz de alto nivel entre la aplicación y el "speech Engine (SAPI)". SAPI implementa todos los detalles de bajo nivel necesarios para controlar y manejar las operaciones en tiempo real.

Existen dos sistemas(motores) SAPI, por un lado, está el TTS(Text-To-Speech) y por otro lado el "Speech Recognizer" el cual es el que más nos interesa. El sistema TTS sintetiza texto y archivos y los lleva a un audio usando voces sintéticas. El "Speech Recognizer" convierte la voz humana en textos y archivos reconocibles.

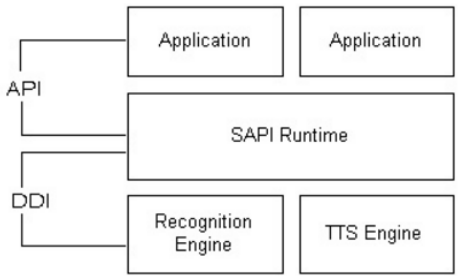


Figura 17. Diagrama estructural de SAPI.

La poca documentación y la falta de ejemplo hicieron de esta opción la menos favorable respecto a la segunda que se hablara a continuación.

La otra opción con la cual se lograron realizar pruebas fue el "Speech Recognition Plugin", el cual es una extensión del motor el cual se puede instalar y manipular de forma gratuita. Este plugin hace uso de la librería "Pocket-sphinx". En este momento el plugin puede ser utilizado para detectar frases. La detección de palabra por palabra es bastante deficiente, aunque tiene buena variedad de idiomas ya que tiene soporte para inglés, mandarín, francés, español y ruso.

Finalmente se optó por la implementación del plugin, sin embargo, se observó que tenía bastantes fallos y solo reconocía ciertas oraciones y estas tenían que ser dichas con bastante lentitud y pronunciación. Dado lo anterior se decidió prescindir de la implementación de la IA ya que no cumplía con un papel importante dentro la aplicación y además presentaba bastantes errores. Sin embargo, se deja abierta la posibilidad de seguir trabajando en ella para aplicaciones más interesantes en el futuro.

3. Evaluación y Cumplimiento De Los Objetivos

Evaluación de Objetivos		
Objetivo	Evaluación	Observaciones
Integración del Hardware HTC vive junto con el	La integración del Hardware fue satisfactoria	En algún momento se presentaron algunos

Motor de desarrollo Unreal Engine.	y hay que agregar que la documentación es muy completa, lo cual ayudo a que no hubiese mayor problema. La calibración del dispositivo no llevo mucho tiempo ya que steamVR cuenta con una guía en forma de tutorial que facilita esta tarea.	inconvenientes, pero solo fue necesario actualizar algunos drivers para solucionar el problema. También se debe instalar preferiblemente la última versión de Unreal Engine ya que solo las últimas versiones tienen soporte para realidad virtual.
Realizar Optimización de aplicación.	El dispositivo de realidad virtual consume bastante recurso tanto de CPU como de GPU. El haber aplicado estas optimizaciones ayudaron a que fuese más fluida la aplicación y no se recalentara mucho la máquina por lo que se considera que se logró el objetivo satisfactoriamente.	A pesar de los cambios Positivos que se lograron con los ajustes de optimización se recomienda usar una maquina con una mejor tarjeta gráfica (GTX1080), ya que se tuvieron que hacer pequeños sacrificios en calidad debido a la sobrecarga en GPU. Además de ello se observó un decrecimiento en el rendimiento de la aplicación cuando se

		corría la aplicación por varios minutos seguidos ya que el computador se recalentaba y comenzaba a bajar el rendimiento.
Asegurar una experiencia de Usuario satisfactoria.	Se obtuvo una excelente retroalimentación por parte de los usuarios ya que fueron muy pocos los problemas de mareo y desubicación. Se logró satisfactoriamente el objetivo de introducir al usuario en la experiencia a pesar de algunos inconvenientes que tuvieron inicialmente para entender el manejo de los controles.	A pesar de la buena retroalimentación se considera que se puede mejorar en algunos aspectos, como lo puede ser una guía inicial para el correcto uso de los controles.
Implementar Interacciones de entorno .	Esta característica fue muy bien implementada y llamo bastante la atención de los usuarios, por lo cual jugó un papel muy	Las interacciones que se implementaron fueron la manipulación de objetos y la apertura de puertas. Al haber llamado tanto la atención de los usuarios se

	importante en la experiencia.	planea implementar más interacciones en el futuro.
Realizar Correcta implementación de audio.	La implementación del audio no tuvo mayor problema ya que las clases pre programadas del motor dan mucha flexibilidad para la manipulación de estos.	Se pueden mejorar varios aspectos en calidad de audio , sin embargo , con el objetivo de no incrementar costos se usaron recursos gratuitos de internet, por lo cual no se implementaron muchos efectos de audio y los que se implementaron no contaban con la mejor calidad.
Implementación de Inteligencia artificial que complementara la experiencia.	Este es el único de los aspectos que quizá no se lograron los objetivos deseados, Ya que finalmente se prescindió de usarlo debido a que la IA implementada no aportaba mucho a la experiencia.	Queda abierta la posibilidad de seguir investigando en el tema y de pronto llegar a implementarlo en proyectos futuros que sean más ambicioso, ya que por temas de tiempo y costo no valía la pena.

Tabla 1. evaluación de cada uno de los objetivos de la práctica.

4. Conclusiones y Recomendaciones

El mareo por simulación es común en realidad virtual y puede afectar de gran manera la experiencia. Para ayudar a reducir la probabilidad de una mala experiencia se tuvieron en cuenta las siguientes recomendaciones:

1. Asegurar el "FrameRate" y también idealmente un pequeño buffer para asegurarnos que siempre estemos trabajando sobre le "frameRate" nativo de las HTC vive. Los "FrameRates" bajos son grandes potenciadores del mareo por simulación por lo cual nos aseguramos de optimizar al máximo la experiencia. En la Figura 18 se muestran los dispositivos de realidad virtual soportados por Unreal con su "FrameRate Ideal".

HMD Device	Target Frame Rate
DK1	60 FPS
DK2	75 FPS
Rift Retail	90 FPS
Vive	90 FPS
Gear VR	60 FPS
PSVR	Variable up to 120 FPS

Figura 18. Dispositivos de realidad virtual soportados por el motor de desarrollo Unreal Engine junto con el frame rate de los dispositivos.

2. Es importante que la experiencia de realidad virtual sea testeada por varias personas, para asegurarnos que es confortable para cualquiera ya que puede ocurrir que nosotros los desarrolladores nos adaptemos a la experiencia por el uso continuo.
3. Evitar cualquier tipo de cinemática con la cámara o cualquier acción que tome control del movimiento de la cámara lejos del usuario ya que esto demostró generar problemas en la experiencia.
4. Evitar sobrescribir el campo de visión (FOV) manualmente y tampoco dejarlo como una variable a modificar para el usuario final. Este valor debe encajar con la geometría física de los dispositivos de RV y sus lentes y debe ser ajustado automáticamente por el SDK y configuración interna del dispositivo.
5. En videojuegos con cámara en primera persona es usual usar sacudidas de cámara para simular situaciones como temblores o movimientos bruscos. Sin embargo, esto en experiencias con realidad virtual demostró ser un gran potenciador de mareos y empobrecimiento de la experiencia.
6. En el diseño de niveles para realidad virtual es recomendable usar luces regulables con colores estándar. El uso de luces fuertes y vibrantes tiene efectos negativos en la experiencia y hace que el mareo se pronuncie más rápido en el usuario.
7. El uso de escaleras en ocasiones puede ser una mala elección debido a que puede causar desorientación en el usuario, sobre todo cuando este las recorre de manera veloz.

8. Cuando se usan controles para simular el caminado en realidad virtual es recomendable dejar la velocidad y la aceleración como valores constantes ya que el incremento o disminución de esta puede causar mareo.
9. Obtener una correcta escala del mundo real es una de las cosas más importantes para ayudar a entregar la mejor experiencia posible. Tener una escala incorrecta podría conllevar a todo tipo de problemas sensoriales e incluso llegar a ser una experiencia con mareo. Los objetos son fácilmente observables en el rango de 0.75 a 3.5 metros desde la cámara del usuario. La equivalencia de las unidades de Unreal (U.U) es de 1 uu= 1cm. Esto significa que los objetos son fácilmente observables entre 75 uu a 350 uu. La relación entre unidades puede ser modificada ingresando a la opción "world settings" modificando el valor "World to meters" que por defecto es de 100.
10. La configuración de la cámara debe ir acorde al tipo de experiencia que se vaya a realizar, bien sea una experiencia "Seated" o una "Standing". Para una experiencia "Seated" es necesario subir el origen de la cámara. Para una experiencia "Standing" es necesario mover el origen de la cámara a 0.
11. La escala de los objetos que se encuentran en el mundo virtual deben acercarse lo mayor posible a como son en la vida real de otro modo la inmersión en el mundo virtual puede verse afectada y puede generar confusión.
12. Los mapas de normales simulan la impresión de una superficie 3D, es decir, de relieve, Pero este relieve no va a proyectar ninguna sombra y no obstruirá ningún objeto. Cuando miras un mapa de normales en un objeto en RV se darán cuenta que estas no tienen el impacto al que estamos

acostumbrados. Esto es porque el mapeo de normales no tiene en cuenta una vista binocular. Por ello es común que los objetos se vean algo planos con un dispositivo de realidad virtual. Sin embargo, esto no significa que no debas usarlos, esto significa que es necesario evaluar más detenidamente si los datos que intentas transmitir en un mapa de normales quizá sea mejor representarlos con geometrías.

13. En gráficos 3D renderizar transparencia es extremadamente costoso ya que la transparencia generalmente necesitara ser reevaluada cada frame para asegurarse que nada ha cambiado. Debido a esta reevaluación, el renderizado de transparencia en RV puede ser tan costosa vale la pena buscar otras opciones para suplir esta necesidad. Una opción para suplir esta necesidad es usar la función "DitherTemporalAA" en el material del objeto.
14. De las características que más consumen recursos son el uso de sombras dinámicas e iluminación. Por tanto, el poder fingir estos efectos u omitirlos es una gran ventaja cuando de optimización se trata.

La aplicación o herramienta de realidad virtual debe ser pensada y adaptada para el usuario por lo cual surge un problema ya que las características perceptuales de cada persona varían de forma notable dependiendo de su genética o de posibles patologías. La sensibilidad Ocular, resistencia al mareo, la facilidad en la manipulación de los controles o la fuerza ejercida son diferentes en cada usuario. Es por ello que las calibraciones no deben ser restrictivas en función a las pruebas realizadas durante el desarrollo, lo más correcto en mi experiencia seria tomar calibraciones para cada individuo en una muestra de sujetos lo suficientemente grande para poder tomar valores medios de la población a la que va dirigida la aplicación. Sin embargo, lo ideal es establecer mecanismos que

calibren la velocidad de rotación de la cámara, la interpolación de las fuerzas, la distancia interocular, etc. en tiempo de ejecución, adaptándose de manera real a cada usuario y en cada momento.