UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Simucar: Simulador de Manejo

TESINA

Que para obtener el título de

Ingeniero en Computación

P R E S E N T A

Harold Antonio Cruz Gómez

DIRECTORA DE TESINA

M.I. Norma Elva Chávez Rodríguez

Ciudad Universitaria, Cd. Mx., 2019

AGRADECIMIENTOS

A mi familia, a mi madre Socorro, mi padre Marco y mi hermano Henry los cuales fueron el impulso y motivación durante este largo camino académico. Ustedes son la razón de más importante de todos mis esfuerzos y de cada éxito obtenido.

A la UNAM, Facultad de Ingeniería por todos los conocimientos que me proporcionaron para el desarrollo de este proyecto de investigación y además por inculcarme el espíritu profesional que se necesita para el desarrollo profesional.

A todos mis amigos y compañeros que conocí a lo largo de esta etapa y tuve el honor de compartir aulas. A ellos les agradezco no solo su amistad, sino que también todas las experiencias que compartimos y que fueron de gran ayuda para mi aprendizaje.

A la Maestra en Ingeniería Norma Elva Chávez Rodríguez por ser quien aceptó ser la directora de este trabajo y se encargó de la supervisión y revisión de todo el proyecto y llevarlo a un buen camino.

A cada uno de los profesores con los que tuve el placer de tomar clases y poder absorber todo el conocimiento, consejos y valores que a partir de hoy son el día a día de mi vida profesional.

A cada uno de ustedes les agradezco por ser parte de esta etapa final y esperando corresponderles dentro de esta nueva etapa profesional en la cual pondré en práctica dada una de las enseñanzas, valores éticos y morales y conocimientos que me otorgaron

DEDICATORIAS

Para mi madre Socorro por todo su amor, cariño y comprensión y tiempo que me ha dedicado sin esperar nada a cambio

Para mi padre Marco por sus consejos y su perseverancia durante todo este tiempo

Para mi hermano Henry por su apoyo

Para mis amigos por todos sus consejos y buenas experiencias

Para mi familia, especialmente a mi tío Miguel por siempre brindarme su apoyo en todo momento

Para Karen por sus consejos, por su tiempo y paciencia y por cada momento feliz que hemos pasado.

ÍNDICE

CAPÍTULO I
INTRODUCCIÓN

SIMUCAR
SIMULADOR DE MANEJO

1.1. Introducción

La computación gráfica es una de las ramas de las ciencias de la computación que se encarga del estudio, diseño y trabajo del despliegue de imágenes en la pantalla de un computador a través de las herramientas proporcionadas por la física, la óptica, la térmica, la geometría, etc.

Actualmente existen muchas aplicaciones de la computación gráfica en diversos campos de la ingeniería e investigación científica, que demandan una gran cantidad de recursos computacionales, por lo cual es de vital importancia analizar e implementar en futuros proyectos que promuevan la innovación dentro de dichas áreas de conocimiento.

Una de las tecnologías que más trascendencia ha tenido dentro de este contexto gráfico y que además ha sido el pionero para el desarrollo de diversas aplicaciones es la realidad virtual, la cuales es una ciencia basada en el empleo de ordenadores y de otros dispositivos de visualización, cuyo fin es producir una apariencia de la realidad que permita al usuario tener la sensación de estar presente en ella.

Actualmente la realidad virtual es una de las herramientas con mayor número de utilidades no solo dentro el área de los videojuegos, también se ha ido extendiendo dentro de nuevas áreas tales como la medicina, la geología, la geofísica, el área deportiva y otras áreas.

En el presente proyecto de investigación se habla sobre la implementación de dicha tecnología para un caso de estudio el cual se basa en el desarrollo de un simulador de manejo el cual tiene como objetivo implementarse como medida de evaluación para la obtención de licencias de manejo y además brindarle a los usuarios una nueva forma de enseñanza de conducción de manera interactiva.

Capítulo I
Introducción

Cabe mencionar que este proyecto solo es un prototipo adaptado como una forma de evaluación, esto quiere decir que durante el desarrollo del mismo se buscó que fuera fácil de utilizar, accesible en cuestiones de costos y adquisición de elementos de hardware para el mismo y que fuera interactivo, ya que actualmente existen diversos simuladores de manejo los cuales solo son utilizados para ser utilizados por pilotos experimentados dentro del mundo de la F1 o NASCAR.

Por ello, dentro de este trabajo, se muestran las etapas de construcción de dicho prototipo, desde el planteamiento del problema y cuál es el caso de estudio en el cual se está enfocando dicho simulador, el marco teórico que se investigó tales como las herramientas de software y hardware que se utilizó, así como él porque se utilizaron dichas herramientas.

Además, se muestra el proceso de diseño e implementación del prototipo del simulador, cómo se fue construyendo paso por paso los escenarios, la configuración de la realidad virtual, etc. De igual forma se explica la fase de pruebas con los usuarios y la explicación de los resultados tomando en cuenta las opiniones de los usuarios para posibles futuras mejoras.

CAPÍTULO II
PLANTEAMIENTO DEL PROBLEMA

SIMUCAR
SIMULADOR DE MANEJO

2.1. Objetivo de la Propuesta

Desarrollar un simulador de manejo en tiempo real mediante la utilización de una IDE (Integrated Development Environment) de programación (Unity) y hardware especializado como un volante, pedales, controles y gafas de realidad virtual. La imagen 2.1 muestra un ejemplo de un simulador de manejo.

Imagen 2.1: Ejemplo de un simulador de manejo con sus principales componentes

El simulador evaluará al usuario mediante un cierto número de pruebas las cuales servirán de apoyo para verificar si el usuario es acreedor a la licencia de conducir o no, esto con el fin de concientizar a la población de que no solo se trata de obtener la licencia de conducir sin conocimientos previos, esto para evitar posibles accidentes automovilísticos.

2.1. Definición del Problema

Uno de los principales problemas que se quiere resolver con la implementación del simulador de manejo es reducir el número de accidentes automovilísticos en la CMDX, ya que actualmente con las estadísticas realizadas por la OMS (Organización Mundial de la Salud) realizadas en el 2017, las muertes por accidentes automovilísticos ocupan el lugar número 10 con más de 1.3 millones de personas.

Otro de los problemas más importantes y que está ligado con la situación ya mencionada anteriormente, es que muchos de los nuevos solicitantes que obtienen la licencia para conducir no tienen bastante claro la forma correcta de respetar los señalamientos de tránsito, indicaciones de velocidad y eso provoca muchos accidentes día con día los cuales desencadenan más eventos desafortunados los cuales afectan a muchas personas.

2.2. Método

La aplicación estará disponible para PC el cual tendrá implementado un kit de manejo (Logitech G27) o un control de Play Station 4 con los cuales podrán manejar el automóvil a través de los recorridos virtuales en cada una de las pruebas.

Los participantes contarán con unas gafas de realidad virtual (Oculus Rift) los cuales les permitirán estar inmersos dentro de los diferentes recorridos de la aplicación tal como se muestra en la imagen 2.2.

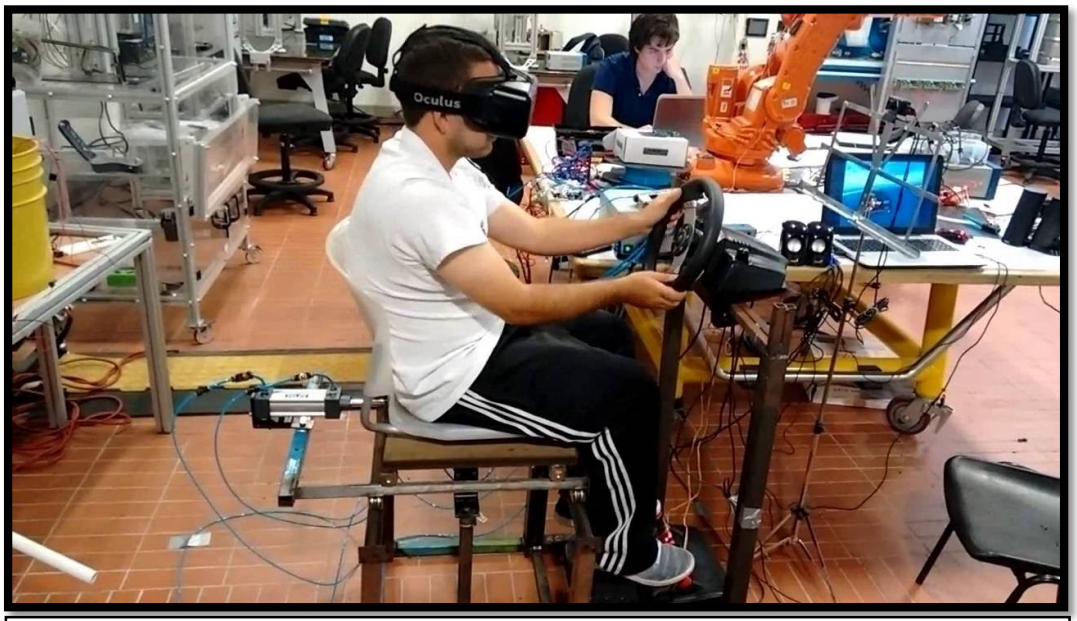


Imagen 2.2: Implementación de la Realidad Virtual utilizando los visores Oculus Rift

2.3. Resultados Esperados

Lograr la implementación del sistema de evaluación propuesto utilizando el simulador de manejo el cual está diseñado con las pertinentes instrucciones las cuales permitan que el usuario pueda realizar la prueba sin ninguna dificultad.

Con esto se busca que el gobierno se interese por esta nueva medida de evaluación ya que permitirá una mejor organización al momento de otorgar una licencia de manejo y además se puede resolver uno de los principales problemas el cuál es el desconocimiento de las reglas de transito vigentes.

CAPÍTULO III
MARCO TEÓRICO

SIMUCAR
SIMULADOR DE MANEJO

3.1. Visual Studio

Visual Studio es un entorno de desarrollo integrado (IDE) el cual contiene un conjunto de herramientas y algunas otras tecnologías de desarrollo de software basado en componentes para crear aplicaciones de alto rendimiento, permitiendo a los desarrolladores crear sitios y aplicaciones web, además de otros servicios en cualquier entorno que soporte la plataforma como se muestra en la imagen 3.1.

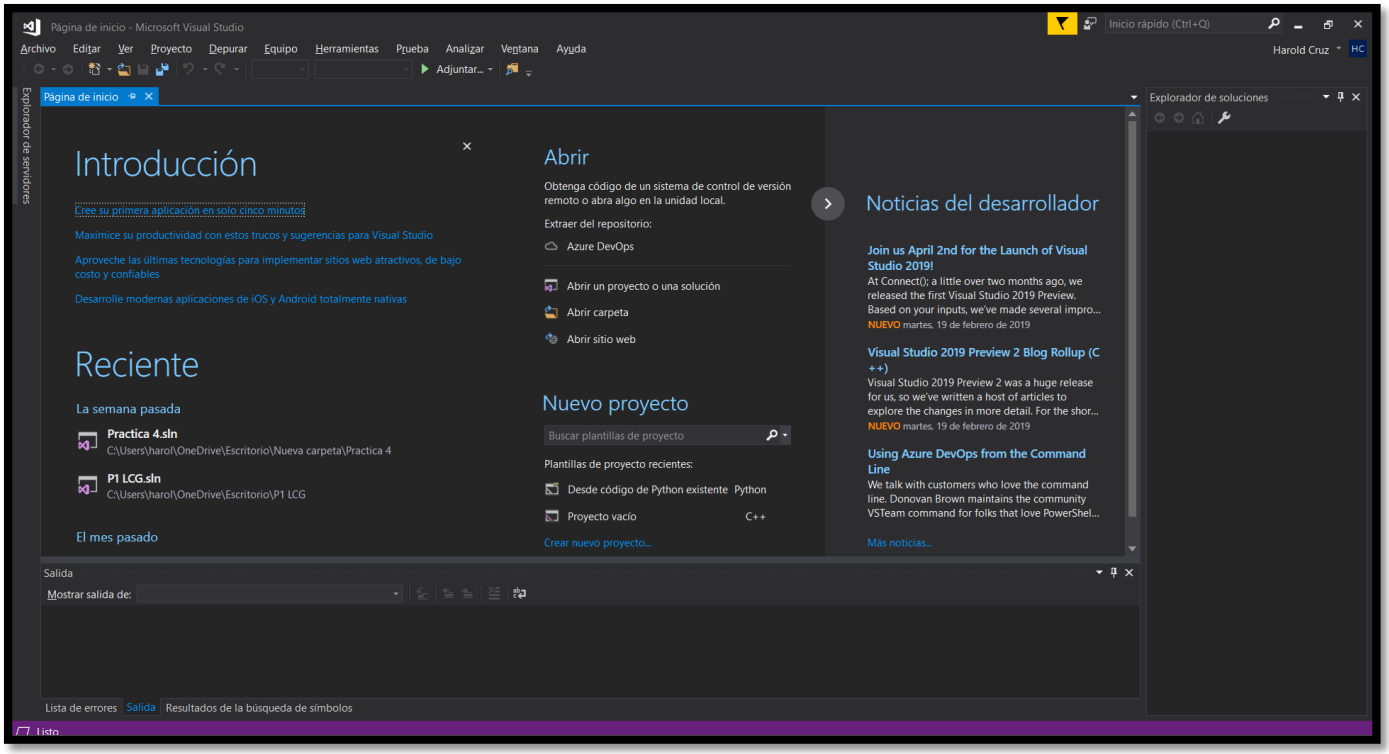


Imagen 3.1. Interfaz Gráfica de inicio de Visual Studio

En términos más específicos, Visual Studio contiene un conjunto completo de herramientas de desarrollo tales como ASP.NET, Servicios Web XML, aplicaciones de escritorio y aplicaciones móviles, además de contar con soporte para algunos lenguajes de programación como Visual Basic, Visual C# y Visual C++, además de tener a la disposición herramientas de trabajo para bases de datos (SQL) y un área de trabajo para el desarrollo de

videojuegos utilizando el game engine de programación Unity. En la imagen 3.2 se observa el menú de aplicaciones con las que Visual Studio puede trabajar.

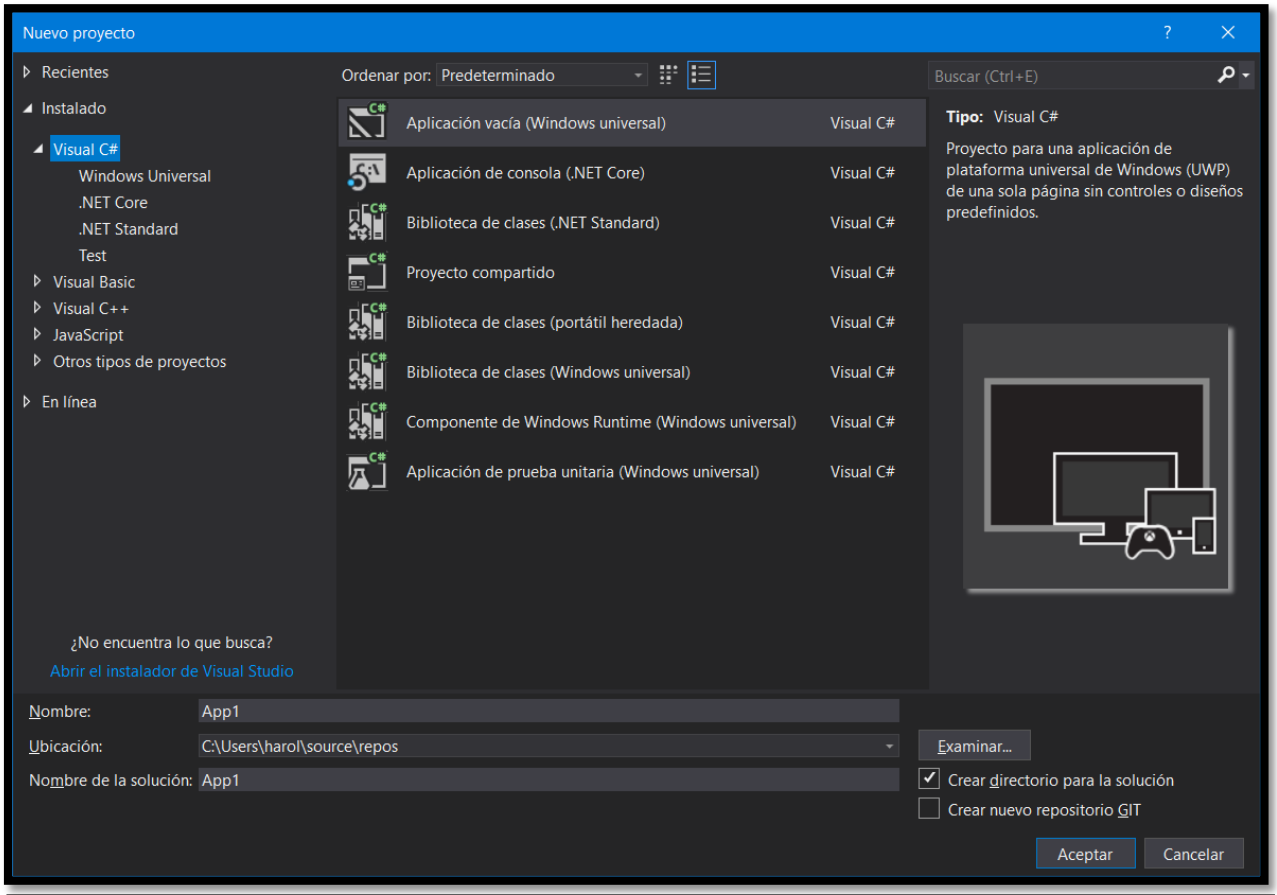


Imagen 3.2. Dentro del menú de Visual Studio se pueden crear diferentes aplicaciones.

3.1. Unity

Unity es una de las plataformas de desarrollo de videojuegos más completos que existen en la actualidad. Permite crear videojuegos para múltiples plataformas a partir de un único desarrollo, incluyendo herramientas tanto para las consolas (PlayStation, Xbox y Nintendo) como para escritorio (Windows, Mac, y Android), navegador, móviles y tabletas (iOS, Windows

Phone y Android). La imagen 3.3 muestra un ejemplo de lo que se puede desarrollar con la plataforma.

Imagen 3.3. Ejemplo de un desarrollo de un videojuego utilizando Unity

3.1.1. Historia

La primera versión de Unity se lanzó en la Conferencia Mundial de Desarrolladores de Apple en 2005. Fue construido exclusivamente para funcionar y generar proyectos en los equipos de plataforma Mac. En 2010 Unity 3 fue lanzado y se implementaron más herramientas de alta gama con el fin de captar el interés de los desarrolladores. En la imagen 3.4 se muestra la primera interfaz de Unity 3.

En 2015 se lanza Unity 5 con soporte para DirectX 11 y soporte para juegos en Linux. En la actualidad es una de las versiones más utilizadas para el desarrollo de videojuegos por su amabilidad en tanto

a la interfaz gráfica, así como la implementación de funciones externas. En la imagen 3.5 se muestra la presentación de Unity 5

Imagen 3.4. Presentación de Unity 3.0

Imagen 3.5. Presentación de Unity 5.0

3.1.2. Editor de Unity

El editor de Unity nos presenta características particulares las cuales permiten al desarrollador tener una edición e iteración rápidas durante su ciclo de trabajo, además de tener adaptabilidad con softwares externos tales como Blender, Vuforia, Xbox, PlayStation, entre otros. En las imágenes 3.6 y 3.7 se muestran las aplicaciones externas con las que Unity puede trabajar.

Imagen 3.6. Unity puede desarrollar aplicaciones para dispositivos Android.

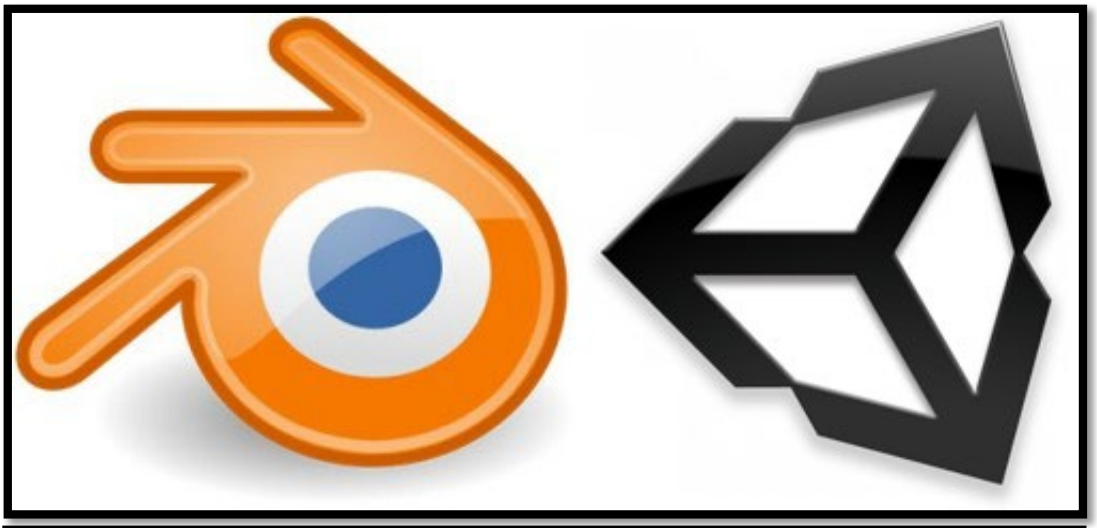


Imagen 3.7. Blender es una de las aplicaciones con más compatibilidad dentro de Unity

3.1.2.1. Editor Todo en Uno

Incluye una serie de herramientas sencillas para el artista para el diseño de experiencias y mundos de juego inmersivos, así como un juego completo de herramientas para implementar la lógica del juego (tal y como se muestra en la Imagen 3.8) y una experiencia de juego con una performance de alta gama.

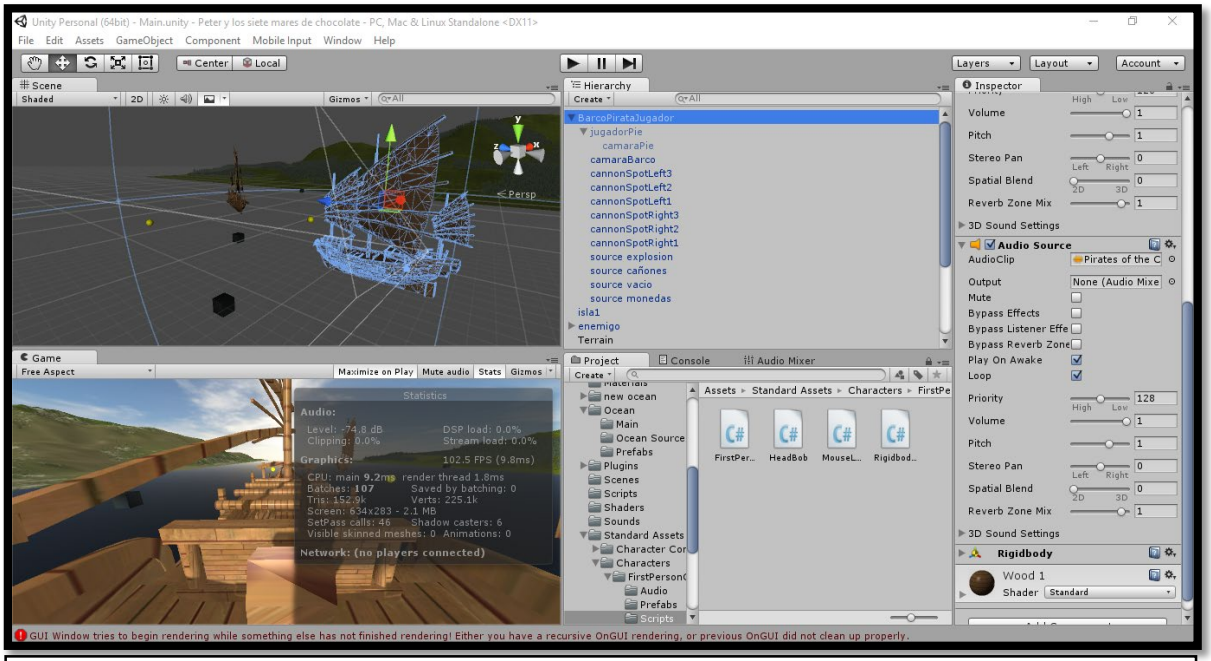


Imagen 3.8. Ejemplo de la interfaz de trabajo

3.1.2.2. Tecnología 2D y 3D

Unity apoya tanto el desarrollo de la tecnología 2D como el de la 3D con prestaciones y funcionalidades para tus necesidades específicas en los diversos géneros.

3.1.2.3. Herramientas AI Pathfinding

Unity incluye un sistema de navegación que permite crear NPC que pueden moverse con inteligencia por el mundo del juego. El sistema

usa mallas de navegación que se crean automáticamente a partir de la geometría de la escena, o incluso obstáculos dinámicos para alterar la navegación de los personajes en el tiempo de ejecución.

3.1.2.4. Flujos de Trabajo Eficientes

Los prefabs de Unity, que son Game Objects precargados, proporcionan flujos de trabajo eficientes y flexibles que te permiten trabajar con confianza sin la preocupación de cometer errores que consumen mucho tiempo.

3.1.3. Características

- Se puede utilizar con software externo tales como Blender, 3ds Max, Maya, Cinema 3D y 4D, Adobe Photoshop, etc.
- Utiliza el motor gráfico OpenGL (Windows, Mac y Linux), Direct3D (Windows) y OpenGL ES (Android y iOS)
- Contiene soporte para Nvidia, el motor de física PhysX, además de soporte en tiempo real para mallas arbitrarias y sin piel, además de contener herramientas de colisiones.

3.2. Blender

Blender es un software de modelado 3D, apoyado por varias herramientas, es multiplataforma (corre en Windows, Linux y MacOS).

Está orientado a artistas y profesionales del diseño y multimedia, puede ser usado para crear, visualizaciones 3D estáticas o videos de alta calidad. También incorpora un motor de 3D en tiempo real el cual permite la creación de contenido tridimensional interactivo que puede ser reproducido de forma independiente. En la Imagen 3.9 se muestra un ejemplo de modelado en Blender 3D

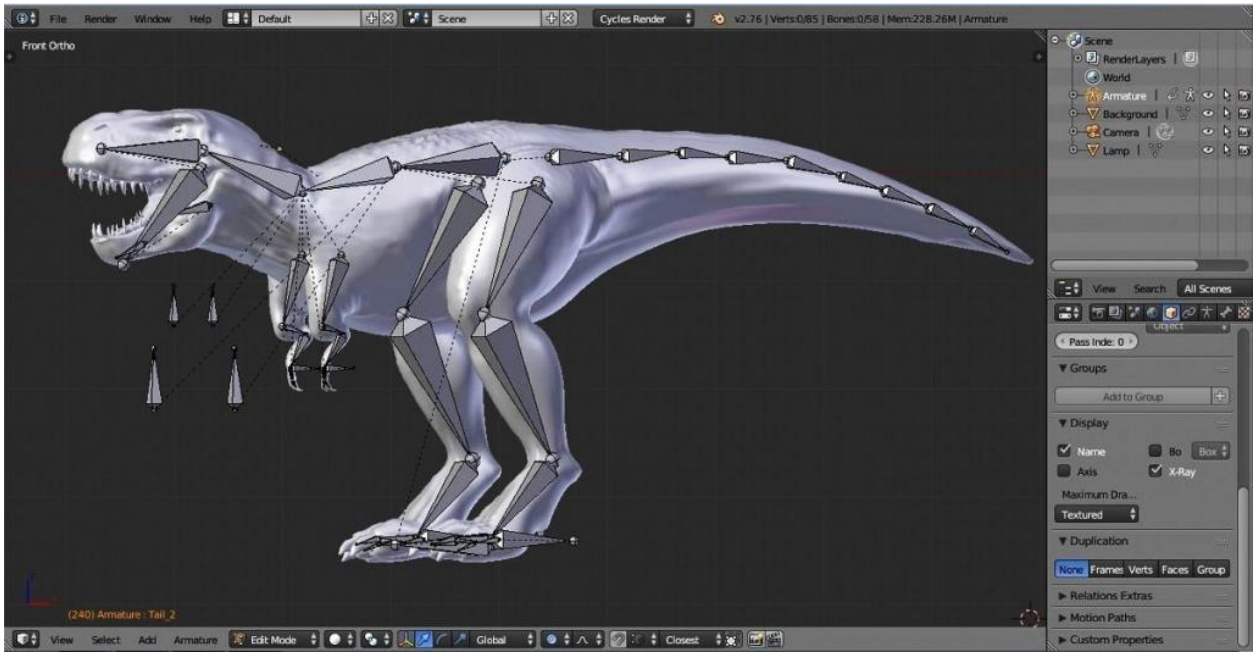


Imagen 3.9. Ejemplo de modelado

3.2.1. Historia

En 1988 Ton Roosendall co-fundó el estudio de animación holandés NeoGeo, la cual se convirtió rápidamente en una de las primeras cosas de animación en Europa. Ton era responsable de dirección de arte y desarrollo del software. Después de una cuidadosa deliberación Ton decide que el software 3D necesitaba ser reescrito totalmente. En 1995 esta reescritura comenzó y se destinó a ser la suite de creación 3D que ahora se conoce como Blender.

3.2.2. Características

- Software libre, gratuito y multiplataforma
- Potente y versátil
- Importa y exporta de múltiples formatos 3D
- Soporte gratuito vía blender3d.org
- Manual multilenguaje en línea

- Una comunidad mundial creciente
- Un archivo ejecutable pequeño que permite una fácil distribución
- Múltiples plugin también gratuitos que expanden las posibilidades del programa

3.2.3. Herramientas

Este software de código libre nos permite realizar diseños 3D, animaciones muy complejas e incluso juegos; además de ser gratuito es extremadamente liviano para lo que nos ofrece.

3.2.3.1. Modelado

Es el proceso mediante el cual se crean objetos del mundo real en un espacio 3D de manera digital; los modelados están formados de líneas, puntos cuadrados, triángulos, curvas, superficies, etc. En desarrollo libre se tienen algunos ejemplos de modelado 3D en Blender de objetos y textos.

Imagen 3.10. Representación de una escena en Blender

En la Imagen 3.10 se muestra la representación de un cuarto utilizado modelos creados en Blender

3.2.3.2. Iluminación

En Blender se tiene varios tipos de luz a la cual la escena se puede adaptar, además de que se puede aplicar materiales para indicar el comportamiento de los objetos con la luz recibida.

3.2.3.3. Tracking

Se puede especificar que cualquier objeto dentro del escenario 3D pueden tener sus propias características y especificar el comportamiento del objeto.

3.2.3.4. Animaciones

Todo objeto dentro de Blender se puede animar o es animable, es decir, las lámparas, cámaras y los objetos que se encuentren dentro de nuestro espacio de trabajo, esto se realiza con un timeline al estilo de flash el cual guarda cada frame en un cuadro de tiempo determinado como se muestra en la Imagen 3.11

Imagen 3.11. Representación de una animación en Blender

3.2.3.5. Modificadores

Los modificadores en Blender permiten realizar un conjunto de operaciones sobre el mallado como, por ejemplo, multiplicar el número de caras, espejar nuestro mallado, darle groso, entre otras acciones.

3.2.3.6. Composición de Nodos

La composición de nodos permite aplicar ajustes a la escena final que constituiría la capa del renderizado de la escena. Utilizando una analogía, la composición de nodos es la comparación de utilizar un software de edición de imágenes como Photoshop o Gimp.

3.2.3.7. Texturizado

Las texturas son un medio para agregar detalles a la superficie, proyectando imágenes sobre la superficie del mallado y se pueden aplicar varios patrones para la personalización del objeto. En la Imagen 3.12 se observa un claro ejemplo de un objeto que pasa por el proceso de texturización. En la Imagen 3.13 se muestra una Imagen ya texturizada.

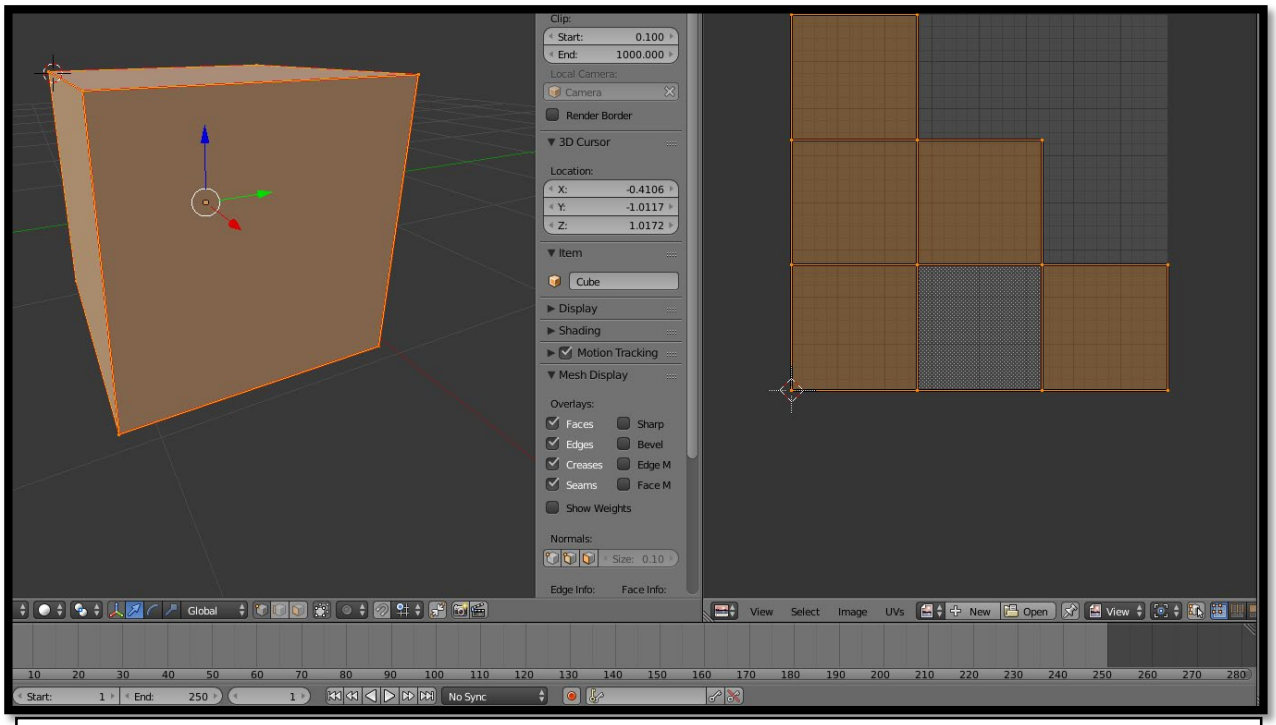


Imagen 3.12. Proceso de texturizado

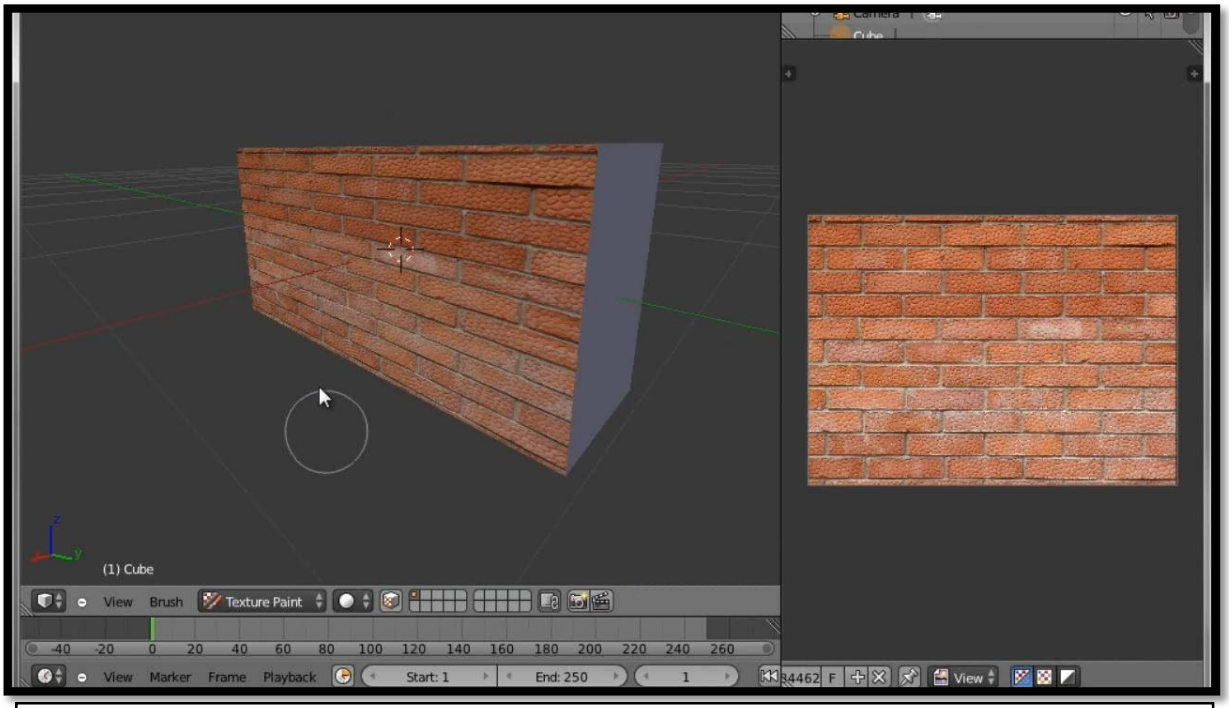


Imagen 3.13. Texturización de un objeto

3.2.3.8. Materiales

Los materiales son un medio para mostrar la sustancia o composición de un objeto que se modele, se pueden definir ciertos aspectos sin la necesidad de aplicar texturas al objeto como el color.

3.3. Oculus Rift

Oculus Rift son unas gafas de realidad virtual capaces de recrear imágenes en 3D y mostrar diferentes perspectivas en función de los movimientos de la cabeza, por lo que ofrece un seguimiento personalizado a 360 grados y una experiencia muy intuitiva. En la Imagen 3.14 se muestra un kit de Oculus Rift los cuales incluye las gafas y controles de movimiento.

Imagen 3.14. Kit completo de Oculus Rift

3.3.1. Historia

Oculus Rift fue diseñado por la empresa Oculus VR en el año 2012 por Palmer Luckey, Brendan Iribe y Jack McCauley. Su principal producto es Rift, un proyecto de gafas de realidad virtual, que ofrece una experiencia invmersiva en el mundo virtual.

En 2014, la empresa fue adquirida por la empresa Facebook el cual se enfocó en el desarrollo en la división de realidad virtual lanzando al mercado en 2017 el Oculus Rift y los Samsung Gear VR.

3.3.2. Aplicaciones

Los Oculus Rift cuentan con un gran campo de aplicaciones, aunque en el que se prevee mayor comercialización es en el mundo de los videojuegos.

3.3.2.1. Videojuegos

Lo que se desea es que los diseñadores adapten sus videojuegos a las gafas Oculus Rift tales como Minecraft, Left 4 Dead, Half – Life 2, Portal 2 entre otros. En la Imagen 3.15 se muestra un videojuego utilizando Oculus Rift

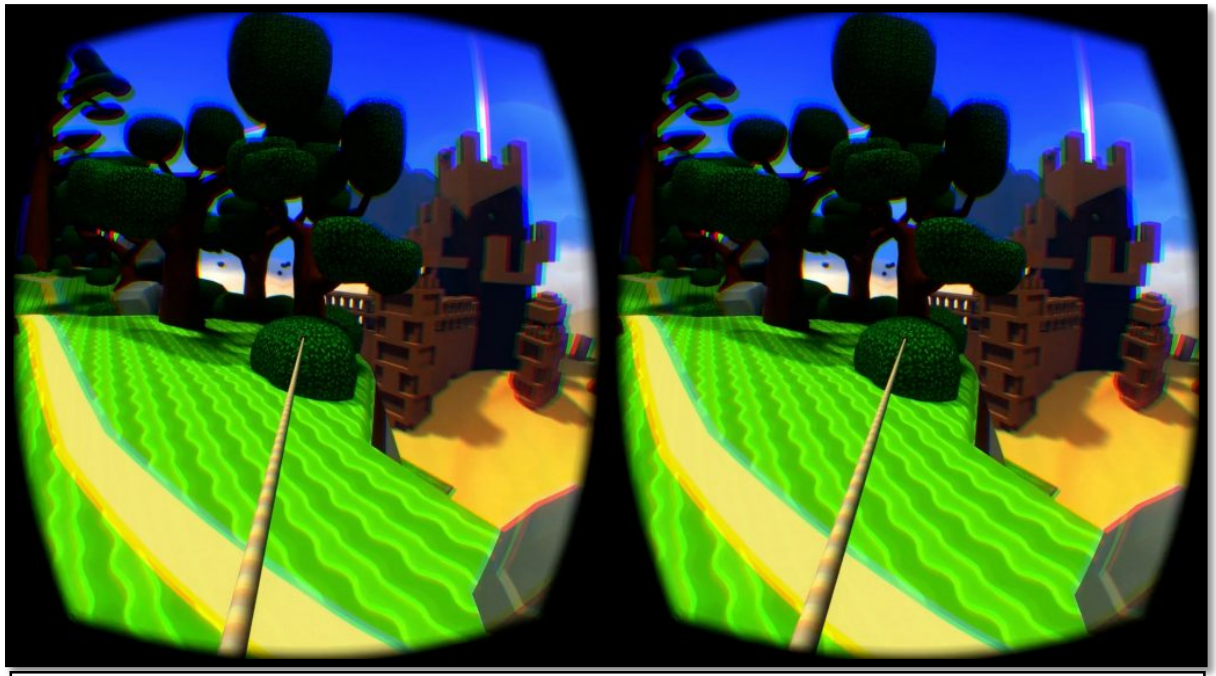


Imagen 3.15. Aplicación de los Oculus Rift en el área de videojuegos

3.3.2.2. Medicina

Situada en la cabeza del cirujano y grabando todos los pasos de la operación con Oculus Rift, se reproduce en streaming tal como se muestra en la Imagen 3.16. El resultado es que el estudiante obtiene una experiencia muy cercana a la visión del cirujano sin ni siquiera estar presente en la misma sala de operación.

Imagen 3.16. Operación utilizando los Oculus Rift

3.3.2.3. Diseño

Los diseñadores de producción pueden evaluar sus diseños de forma virtual, y los arquitectos visualizar obras incluso antes de iniciar la construcción. En la Imagen 3.17 se visualiza un ejemplo del uso del Kit Oculus Rift en el arte.

Imagen 3.17. Visualización de obras

3.3.2.4. Educación

En la parte educativa los Oculus Rift se están utilizando sobre todo para niños con autismo y otras discapacidades, observando cómo se comportan ante determinadas circunstancias de una forma mucho más real.

3.3.3. Elementos

3.3.3.1. Cable

Envía el video a las pantallas de las gafas, se utiliza un cable HDMI que también cuenta con un adaptador DVI para las nuevas tarjetas gráficas del mercado y las laptops. Incluye USB para enviar energía y datos adicionales.

3.3.3.2. Pantalla

Cuenta con una pantalla cuya resolución es de 2160 x 1200

3.3.3.3. Gafas

Gracias a los dos lentes, se pueden ajustar las pantallas para que no cause dolores de cabeza al usuario y una especie de disco integrado que permite ajustar aún más la visibilidad de los lentes.

3.3.3.4. Accesorio de rastreo externo

Permite rastrear a un grupo de LED infrarrojos que se encuentran dentro de las gafas, permite la localización del espacio en 3D.

CAPÍTULO IV
CONGRUENCIA METODOLÓGICA

SIMUCAR
SIMULADOR DE MANEJO

Capítulo IV
Congruencia Metodológica

TEMA: Desarrollo y modificación de un simulador de manejo para brindar soluciones de manera eficaz a las deficiencias de manejo que presentan los nuevos automovilistas, disminuir el número de accidentes automovilísticos e implementar una forma de evaluación para la obtención de la licencia para conducir mediante el uso de una interfaz gráfica generada por el simulador el cual permita verificar si el automovilista cuenta con las habilidades necesarias para ser acreedor a la licencia.

PROBLEMA: Cada año el número de accidentes automovilísticos crece de forma paulatina a tal grado de posicionarse en la posición 10 de causas de muertes en México. Esto sucede debido a ciertas situaciones las cuales pueden ir desde pequeñas distracciones como estar con el celular al momento de estar detrás del volante hasta conducir en estado de ebriedad. Pueda que estas situaciones sean sucesos cotidianos, pero el problema surge desde que se otorga los permisos para conducir, ya que, al no tener una forma de evaluación, no se sabe cómo está preparado el nuevo automovilista.

A continuación, se presenta la matriz metodológica que aborda los objetivos generales del proyecto que se desarrolla para poder brindar una solución al problema planteado, de igual forma se ven plasmados los objetivos específicos que se enfocan más al aspecto técnico y de desarrollo de este caso de estudio.

Para cada objetivo se consideran las hipótesis que son las que dan origen al caso de estudio y cada segmentación para deducir los objetivos tanto generales como específicos, del mismo modo se exponen las variables que intervienen en la problemática en cuestión, así como los indicadores que se tendrán como referencia para poder realizar el análisis correcto de cada objetivo planteado.

4.1. Matriz Metodológica.

En tabla 4.1 representa la matriz metodológica asociada con el desarrollo del simulador, especificando tanto los objetivos generales, así como los específicos tomando en cuenta las hipótesis correspondientes para cada caso, se mencionan también las variables que se encuentran involucradas y los indicadores que sirven como análisis.

OBJETIVOS	HIPÓTESIS	VARIABLES	INDICADORES
GENERAL: Determinar los factores causales de los accidentes automovilísticos.	La mayoría de los conductores no cuentan con los conocimientos necesarios o básicos sobre la conducción de un automóvil, además de no tener en cuenta la importancia del reglamento vehicular (con esto se considera manejar en estado inconveniente, no respetar los señalamientos de tránsito, etc.).	• Falta de experiencia • Poco conocimiento del reglamento vehicular	• Número de accidentes que se reportan al día por las aseguradoras • Usuarios menores de 18 años
GENERAL: Determinar las razones por la cual el gobierno no cuenta con una forma de evaluación	De acuerdo con la publicación de la Ley de Movilidad propuesta en el 2014 se dictamino en el artículo 65 lo siguiente “Para la obtención de	• Tiempo debido a el número de solicitantes por día.	• Reportes sobre el número de automovilistas que obtienen una licencia.

	licencias o permisos de conducir, será necesario acreditar evaluaciones las cuales establezca la Secretaria de Movilidad". Dicha ley no se ha establecido debido a que no se propuesto un sistema de evaluación correcto.	• Dinero en cuestiones de poco presupuesto • La ley de Movilidad no se ha puesto en vigor.	
ESPECÍFICO Desarrollo y modificación de un simulador de manejo apto solo para pruebas de manejo	El usuario necesita un simulador de fácil acceso y manipulación que ponga a prueba sus habilidades, además de orientarlo al uso correcto del mismo	• IDE de programación para el desarrollo del simulador (Visual Studio)	• Lenguajes de programación • Configuración de Hardware para realidad virtual
ESPECÍFICO Desarrollo de una interfaz sencilla y óptima para el usuario	El usuario debe interactuar con la interfaz gráfica por el simulador en el cual se le dará un puntaje dependiendo las diferentes pruebas que se le presente.	• Motor de Juego para recreación de escenarios (Unity) • Diferentes IDE's de programación para el desarrollo de la aplicación	• Programación orientada a objetos para el control del automóvil • Utilización de hardware económico, fácil de implementar y de utilizar.

CAPÍTULO V
DISEÑO E IMPLEMENTACIÓN

SIMUCAR
SIMULADOR DE MANEJO

5.1. Modelado de los Elementos del Entorno Virtual

Utilizando el programa de modelado Blender se inicia con el diseño de las estructuras básicas, como se muestra en la figura 5.1 la cual contiene dicho simulador, incluyendo los siguientes elementos:

- Elementos de decoración
  - Basureros
  - Hidrantes
  - Bancos (asientos)
  - Porta periódicos
  - Cercas
  - Presas

- Pavimentación
  - Calles
  - Señalizaciones de pavimento (cebras, líneas continuas y descontinuas)

- Señalizaciones
  - Semáforos
  - Postes de luz
  - Señales
  - Fantasmas peatonales

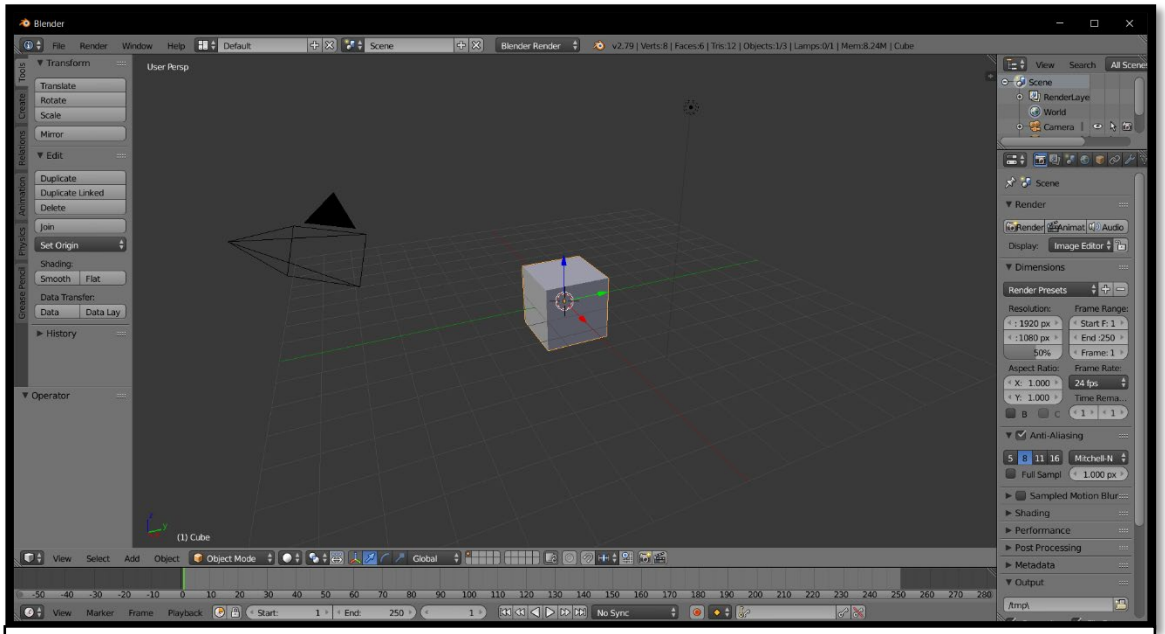


Imagen 5.1 Inicio de modelado utilizando Blender

5.1.1. Modelado de Elementos de Decoración

Los elementos de decoración nos permiten generar un ambiente más real al de una ciudad. Como se muestran en las imágenes 5.2 a la 5.7, se pueden apreciar los elementos decorativos del simulador.

- **Basureros**

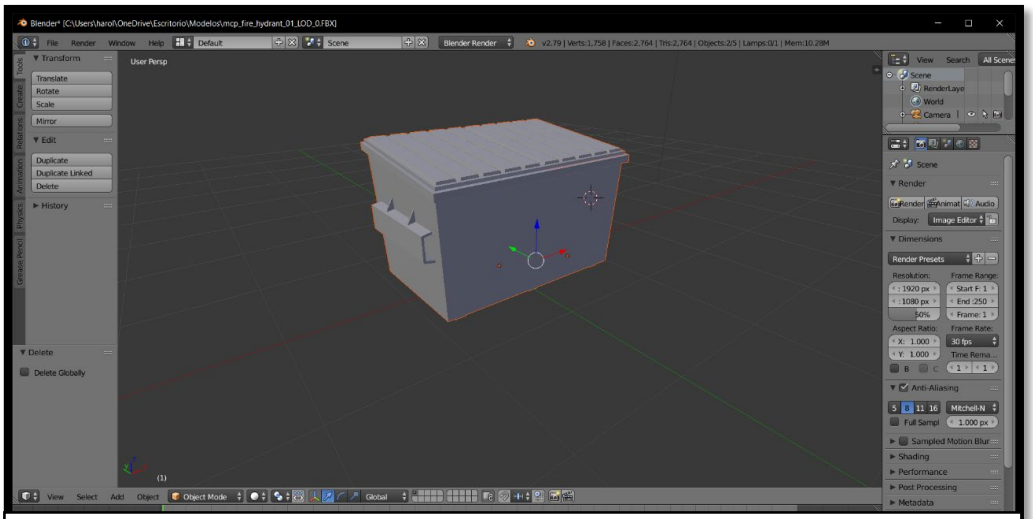


Imagen 5.2 Modelado de un basurero

- **Hidrantes**

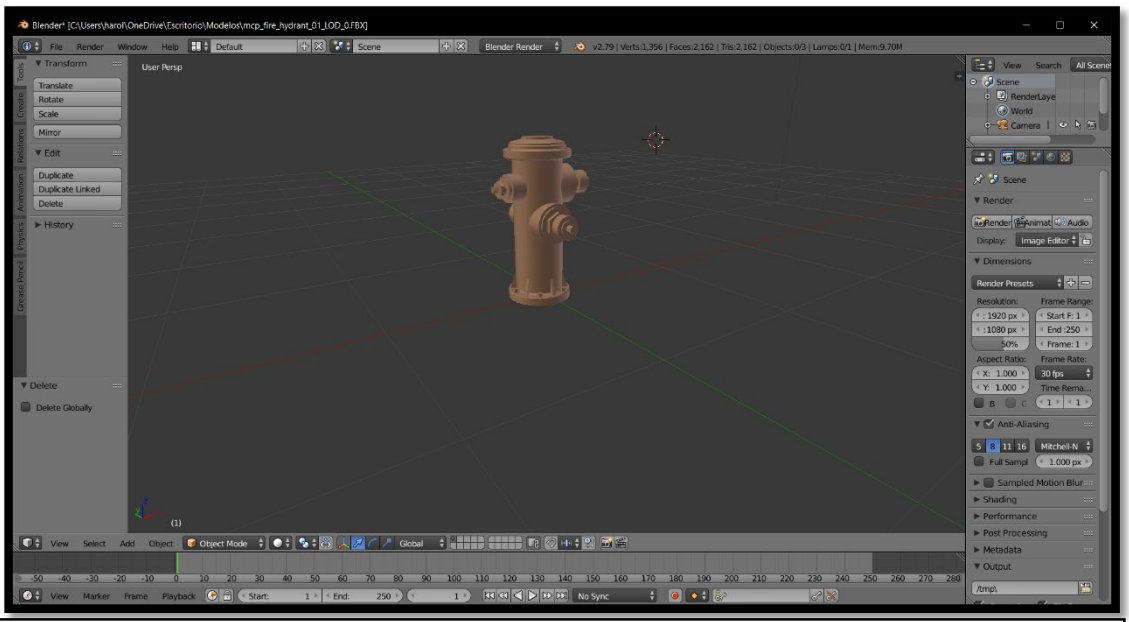


Imagen 5.3 Modelado de un hidrante

- **Bancos**

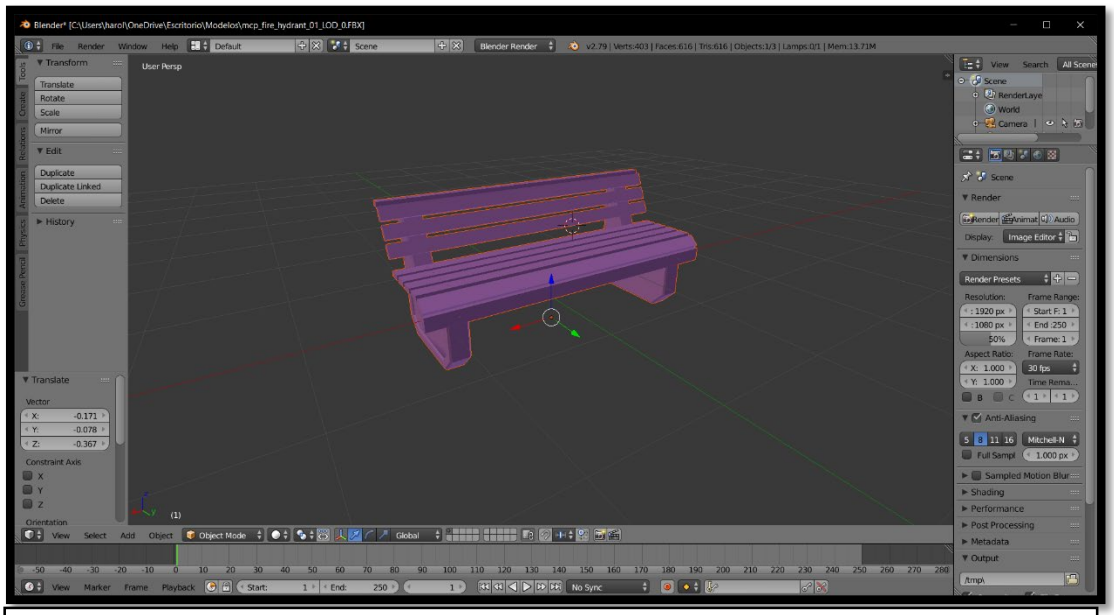


Imagen 5.4 Modelado de un banco o asiento

- **Porta periódicos**

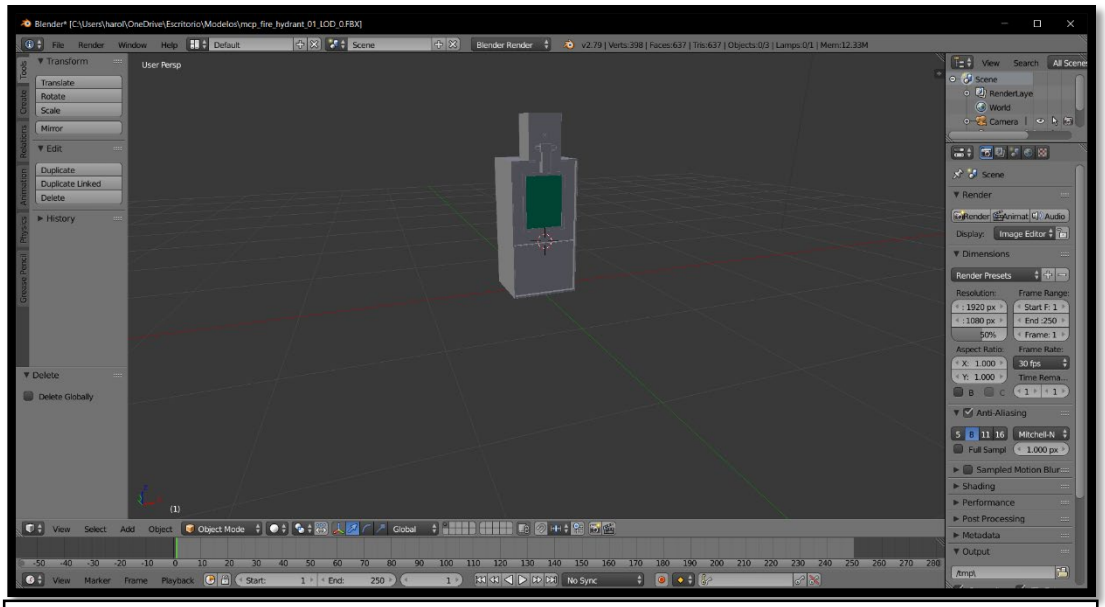


Imagen 5.5 Modelado de una porta periódicos

- **Cercas**

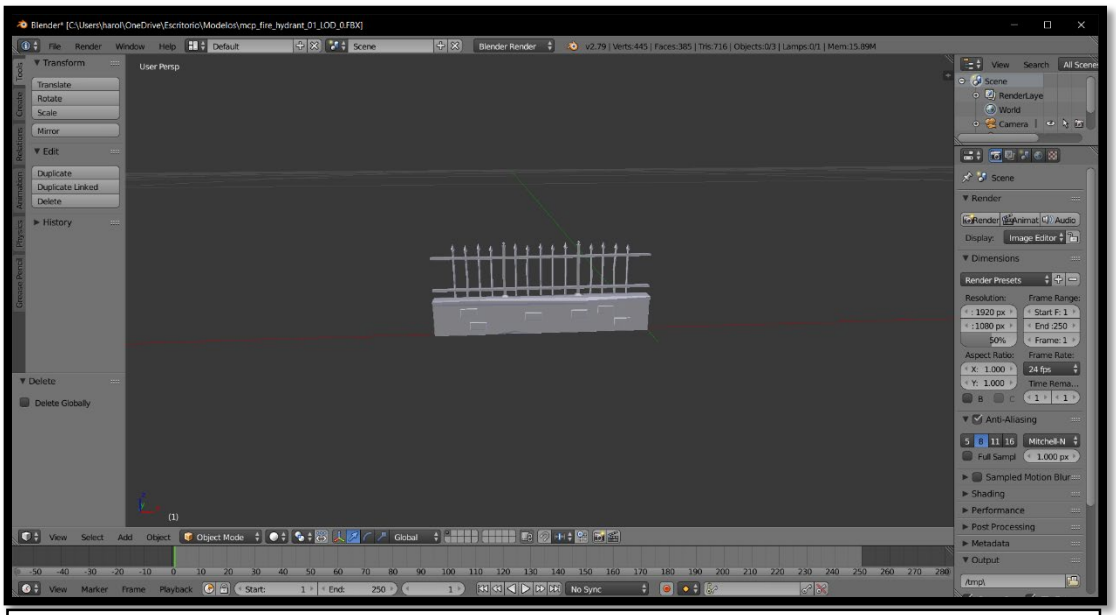


Imagen 5.6 Modelado de una cerca

- **Presas**

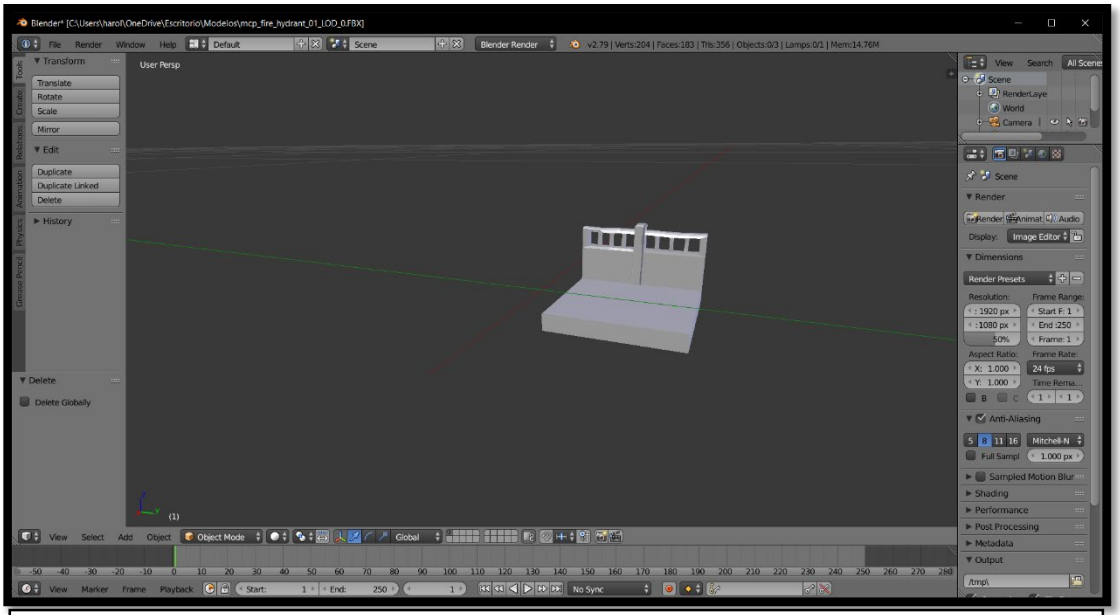


Imagen 5.7 Modelado de una presa

5.1.2. Modelado de pavimentación

La creación del pavimento permite que el automóvil pueda moverse dentro del simulador, además de ser un elemento importante dentro del diseño de los niveles.

5.1.2.1. Calles

Dentro de Blender se inicia con un plano muy sencillo el cual después va a ser rodeado de una textura para dar la forma del pavimento. En la imagen 5.8 se puede apreciar los diferentes diseños del pavimento.

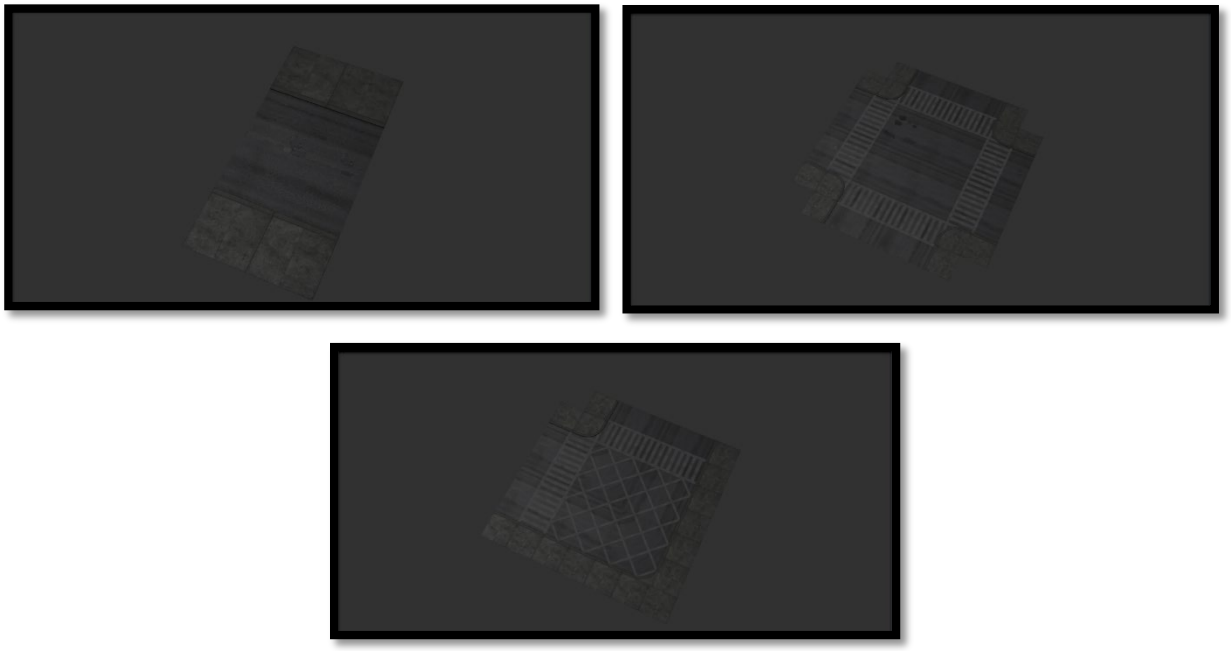


Imagen 5.8 Diseños de las calles

5.1.3. Señalizaciones

Las señalizaciones permiten al usuario estar atento de diferentes indicaciones tales como cuando una calle está cerrada, que la calle ésta iluminada o que hay algún indicador importante. En las imágenes 5.9 a la 5.12 se muestran dichos elementos.

- **Semáforos**

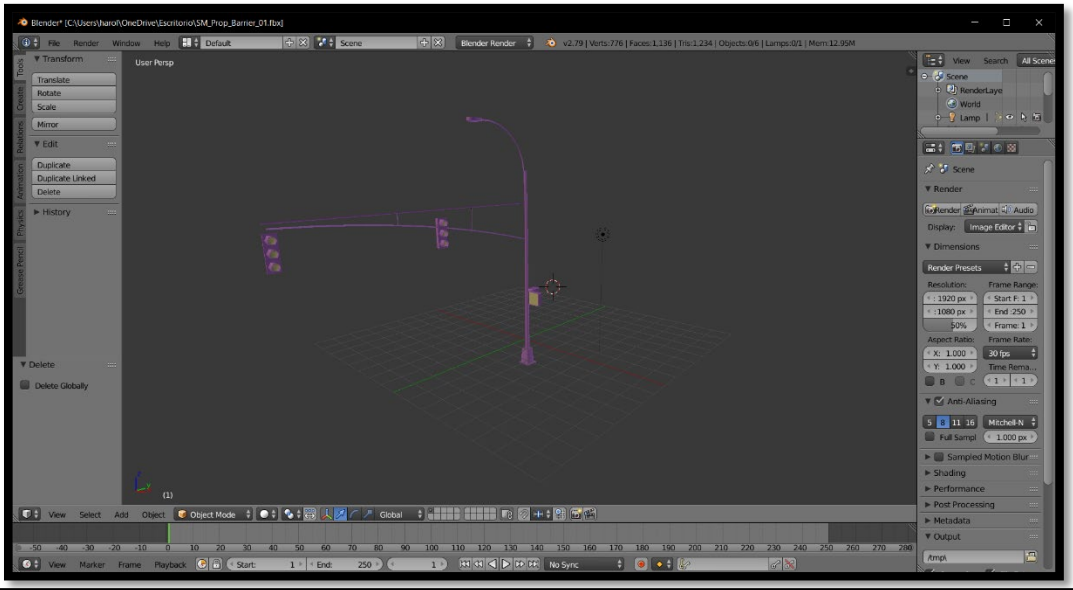


Imagen 5.9 Modelado de semáforos

- **Postes de luz**

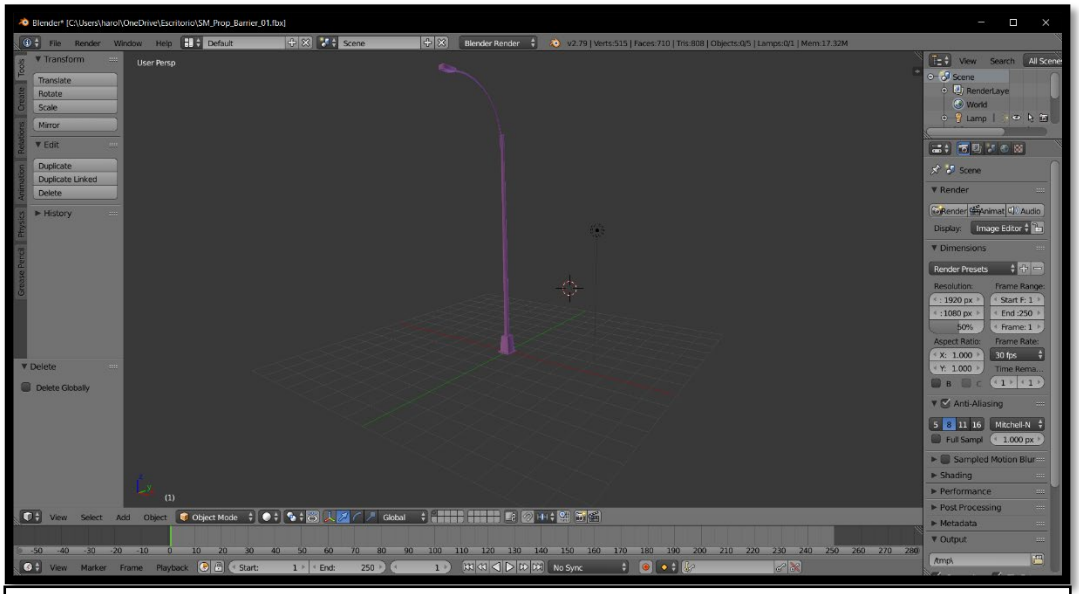


Imagen 5.10 Modelado de postes de luz

- **Señales**

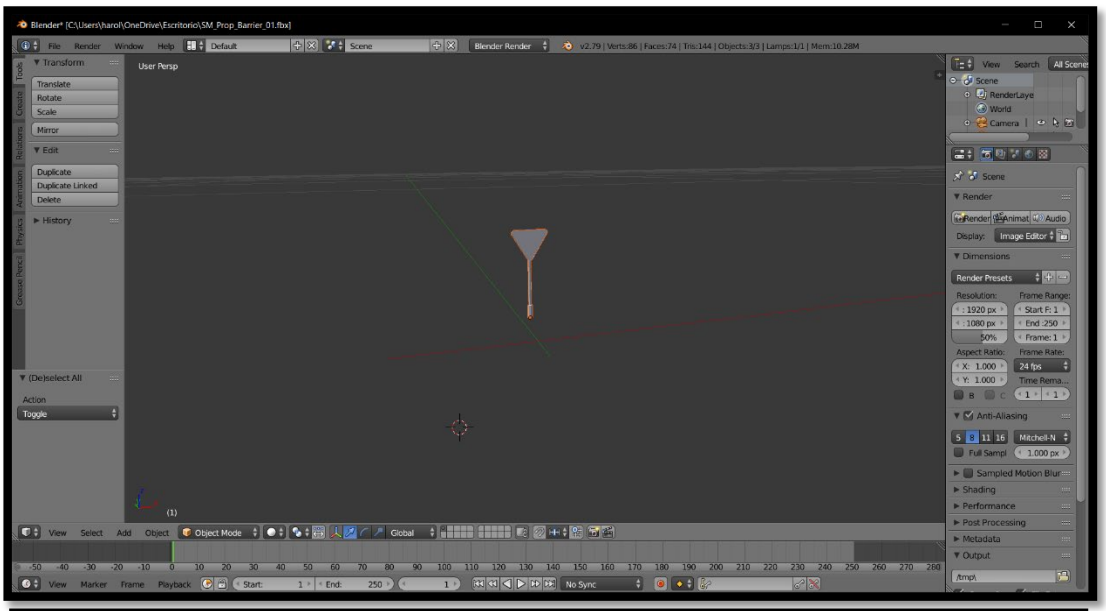


Imagen 5.11 Modelado de señalizaciones

- **Fantasmas Peatonales**

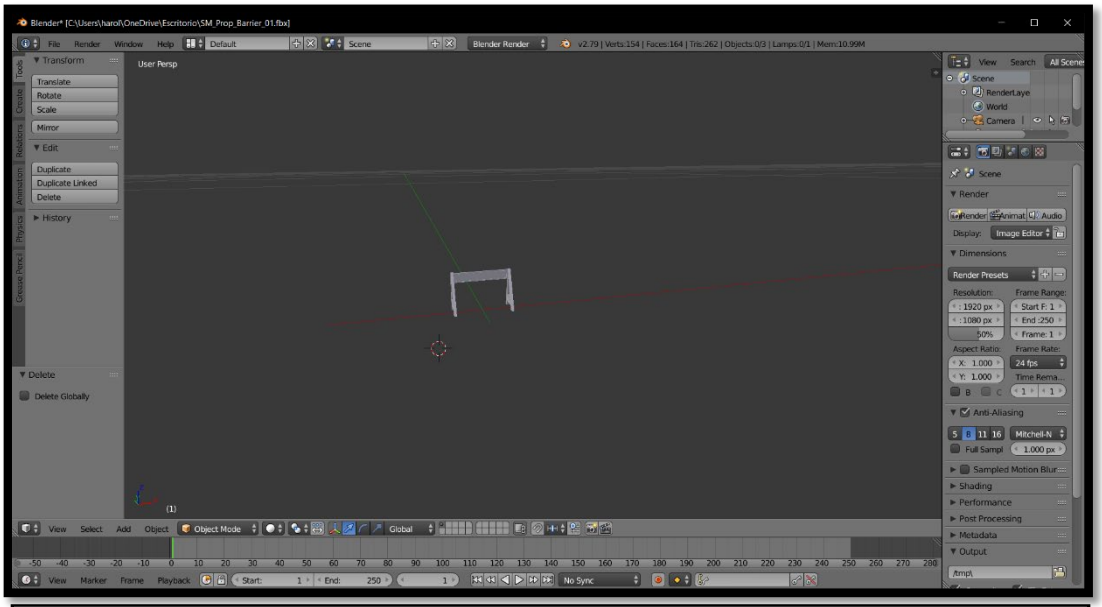


Imagen 5.12 Modelado de fantasmas peatonales

5.2. Creación de Un Nuevo Proyecto en Unity

En esta parte se explicará la forma de cómo se desarrolla el proyecto desde la creación de un nuevo proyecto hasta la implementación de todos los modelos creados en los puntos anteriores.

5.2.1. Iniciando Un Nuevo Proyecto

Para poder iniciar cualquier tipo de proyecto en Unity es importante identificar las herramientas de trabajo las cuales ya se han explicado en el capítulo 3.

Estas herramientas son muy importantes para todo desarrollador de aplicaciones y de videojuegos.

Para comenzar Unity despliega una ventana en donde se pide al usuario un registro ya sea mediante Google o Facebook como se muestra en la imagen 5.13. Una vez que el usuario se registre, se tendrá que colocar el nombre del proyecto y la dirección donde se guardará el

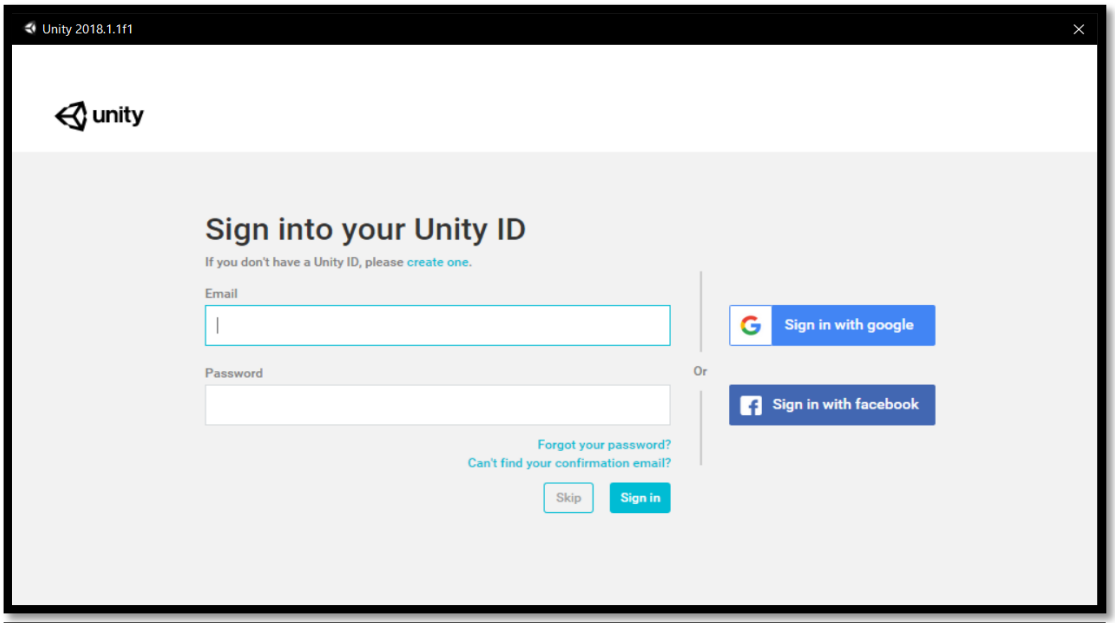


Imagen 5.13 Ventana de registro de Unity

proyecto como se puede apreciar en la imagen 5.14 (es recomendable dejar la dirección por defecto)

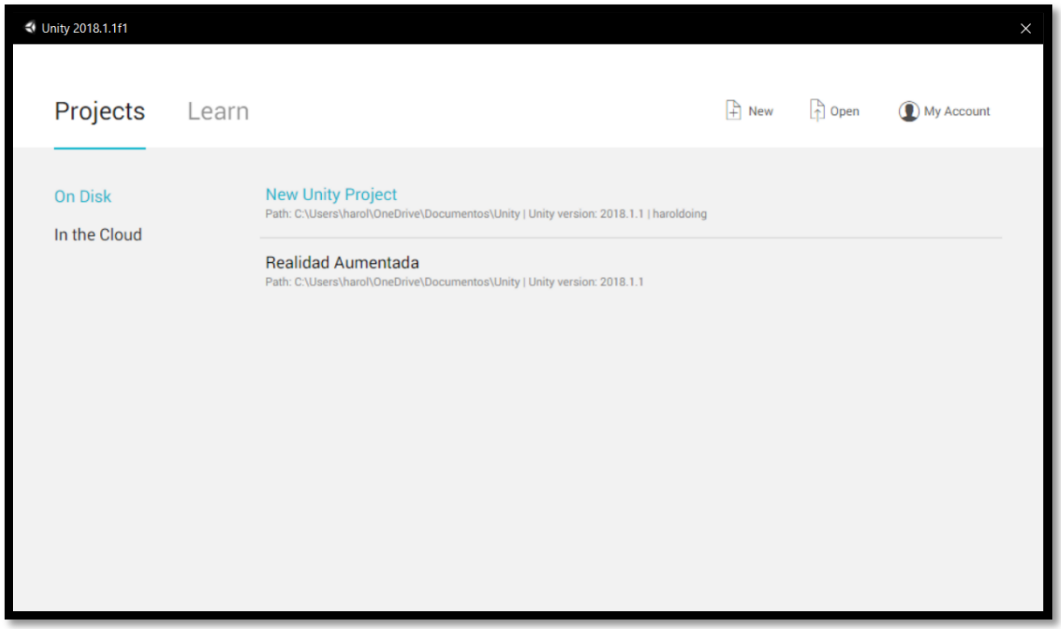


Imagen 5.14 Ventana de nombre y dirección del proyecto

5.2.2. Construcción del Entorno Virtual

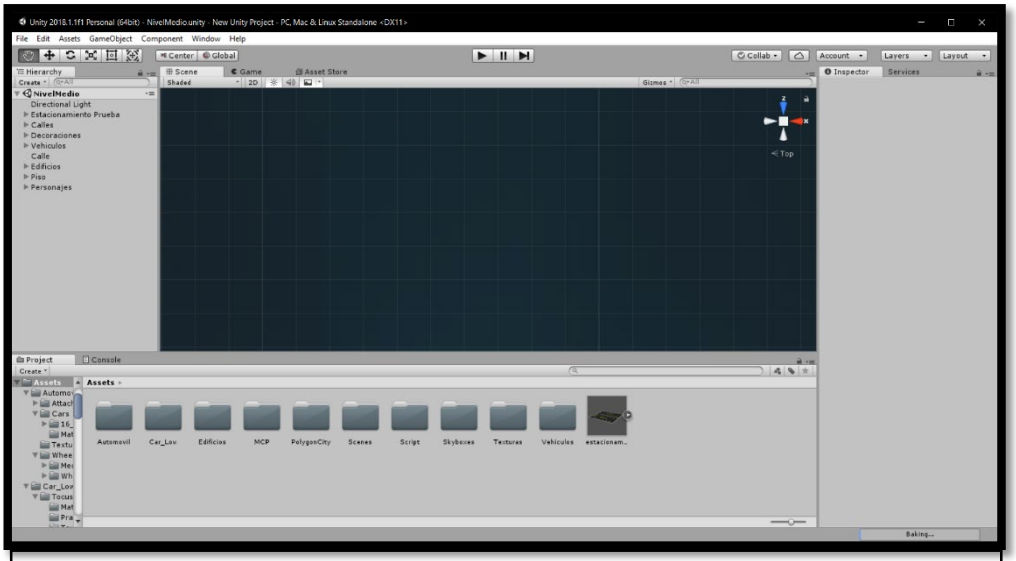


Imagen 5.15 Interfaz Gráfica de Unity

Una vez que se creó el proyecto, se desplegará la interfaz gráfica de Unity con la cual se comenzará la construcción de los niveles del simulador tal como se muestra en la figura 5.15.

Dentro de la interfaz se cuenta con diferentes apartados los cuales se explican con mayor detalle a continuación:

- **Hierarchy:** En esta sección se muestra todos los elementos que se han incluido dentro de nuestro proyecto. Es importante mencionar que, durante la construcción del simulador, cada elemento está asociado a otras subelementos de forma jerárquica tal como se muestra en la figura 5.16.

Imagen 5.16 Ventana de Hierarchy donde se muestran los elementos utilizados en el simulador

- **Proyect:** En esta sección se muestran todos los elementos que han sido importados desde la Asset Store o de forma externa.

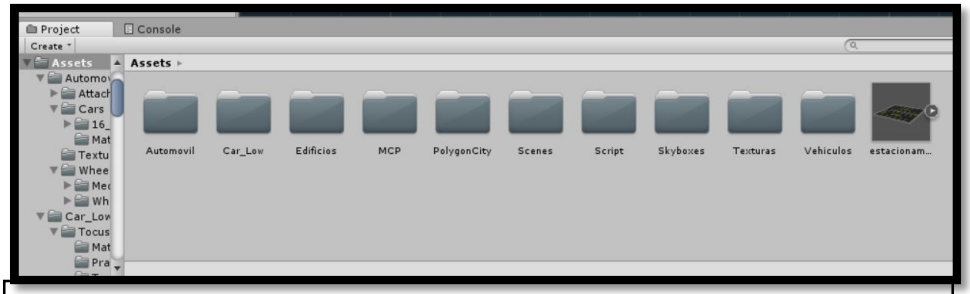


Imagen 5.17 Ventana de Project organizado por carpetas de cada elemento del simulador

Como se muestra en la figura 5.17, en la cual se puede observar todas las carpetas de los elementos utilizados para el simulador.

- **Scene:** Es una de las ventanas más importantes de la interfaz de inicio ya que en esta sección se realiza la edición, modificación e implementación de todos los elementos. En la figura 5.18 se muestra un ejemplo de la construcción y modificación de uno de los niveles del simulador.

Imagen 5.18 Ventana de Scene donde se muestra un ejemplo de la edición de un nivel

- **Game:** Otra de las ventanas más importantes de la interfaz de inicio ya que en esta sección se muestra la ejecución del proyecto en tiempo real. Para habilitar esta opción se presiona el botón play tal como se muestra en la figura 5.19 y automáticamente comenzará con la ejecución de la escena.

Imagen 5.19 Ejecución de la escena en la ventana de Game

- **Asset Store:** En esta ventana se pueden descargar todos los elementos necesarios para complementar cualquier tipo de proyecto. Se pueden descargar desde modelos de cualquier tipo, animaciones, scripts, etc.

  Para el simulador se descargan ciertos elementos como edificios, y el automóvil. En la figura 5.20 se muestra la página principal de Asset Store mientras que las figuras 5.21 y 5.22 se observa los elementos que se descargaron para los niveles del simulador.

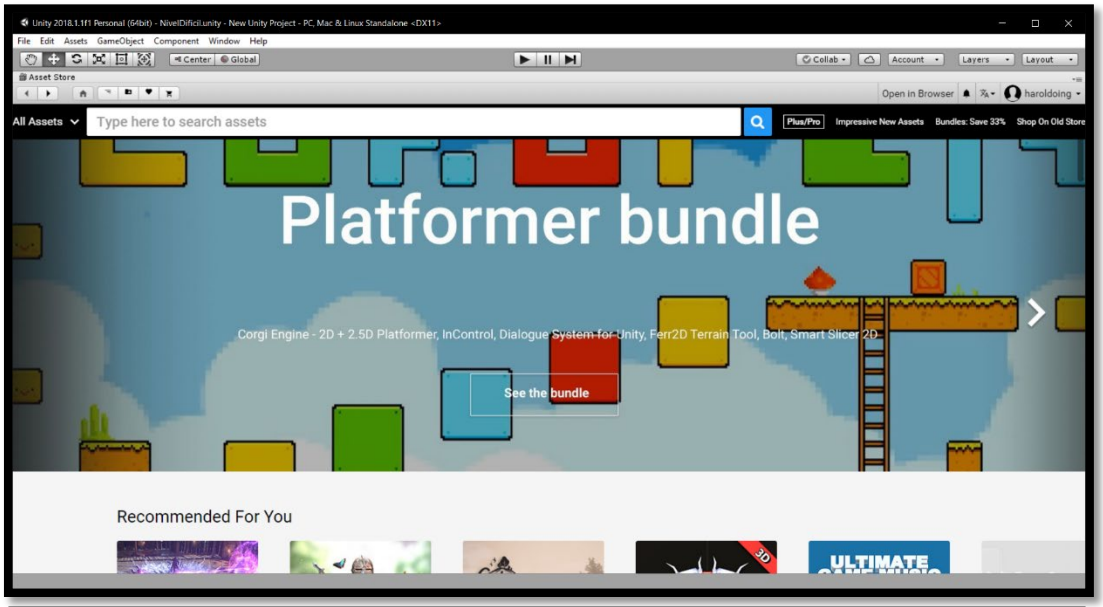


Imagen 5.20 Ventana de Asset Store

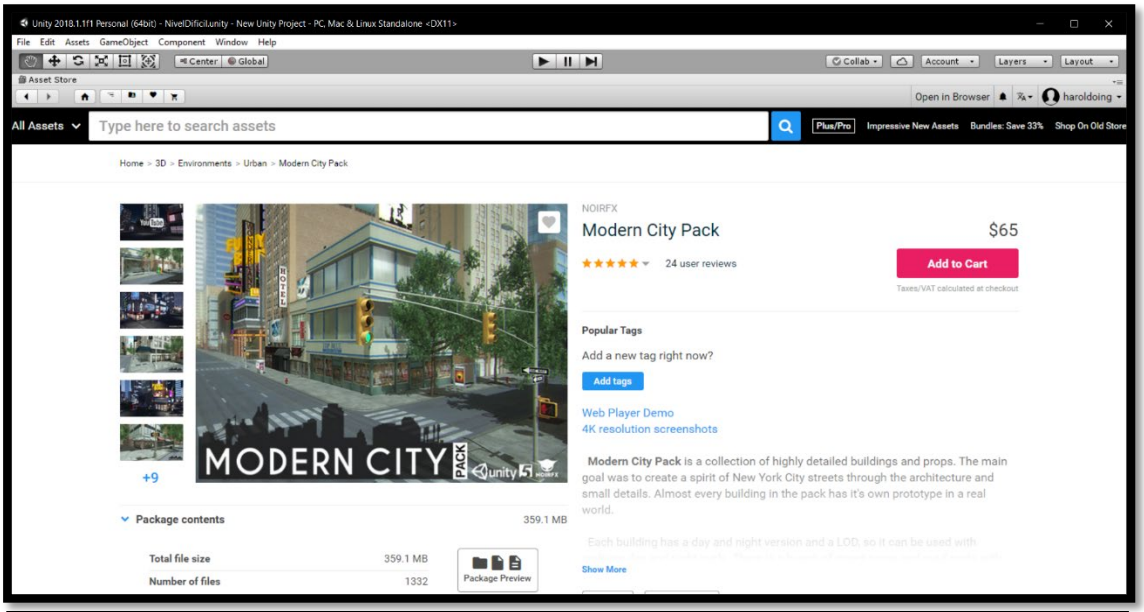


Imagen 5.21 Edificios descargados de Asset Store

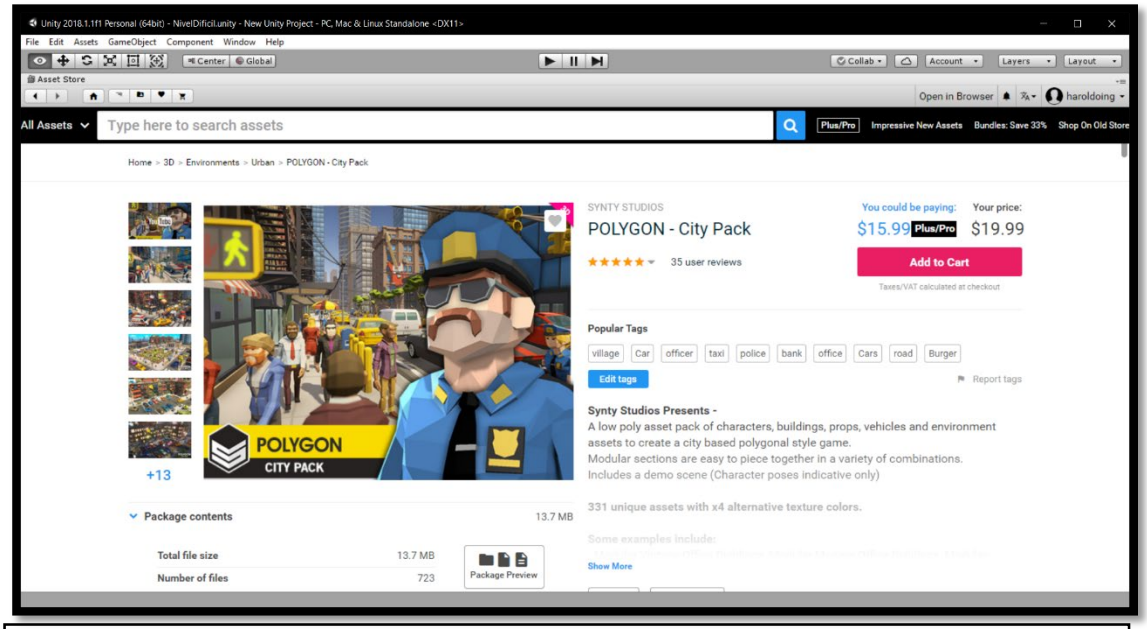


Imagen 5.22 Auto y decoraciones

- **Console:** Ventana que muestra los errores y advertencias dentro del proyecto.
- **Inspector:** Ventana que muestra los detalles de cada uno de los elementos dentro de la escena.

5.2.3. Descripción de Niveles y Tutoriales

El simulador contará con tres niveles los cuales tendrán distintas dificultades (fácil, intermedio y difícil) en los cuales el usuario demostrará sus habilidades de conducción.

Además, el simulador contará con pequeños tutoriales, los cuales irán explicando al usuario conceptos básicos como la aceleración y la reversa del automóvil, el giro del volante entre otras cuestiones importantes antes de avanzar a los niveles.

5.2.3.1. Tutorial 1

El objetivo de este tutorial es que el usuario conozca lo básico que necesitará durante los niveles del simulador. Se explicará la forma de utilizar el control del Play Station 4 o el control Logitech G27 para acelerar y retroceder.

5.2.3.2. Tutorial 2

En este tutorial el usuario continuará aprendiendo los elementos básicos para los niveles avanzados. En este segundo tutorial aprenderá la importancia de los giros utilizando los joysticks del control de Play Station 4 o el volante del Logitech G27.

5.2.3.3. Nivel Fácil

El diseño de este nivel consta de un pequeño circuito en donde el usuario comenzará de un punto inicial tal como se muestra en la imagen 5.23 y tendrá que rodear un gran edificio hasta un punto final del circuito tal como se muestra en las imágenes 5.24 y 5.25.

Imagen 5.23 Punto Inicial de Partida del nivel

La finalidad de este nivel es que el usuario comience a experimentar las primeras dificultades al momento de manejar con la moderación de la velocidad y el cálculo de cuánto debe girar para no chocar con los edificios.

Imagen 5.24 Edificio de referencia por el cual el usuario tendrá que rodear.

Imagen 5.25 Punto final del nivel

En la imagen 5.26 se puede observar el panorama general del nivel mientras que la imagen 5.27 se puede apreciar con más detalle cuales son los elementos del nivel.

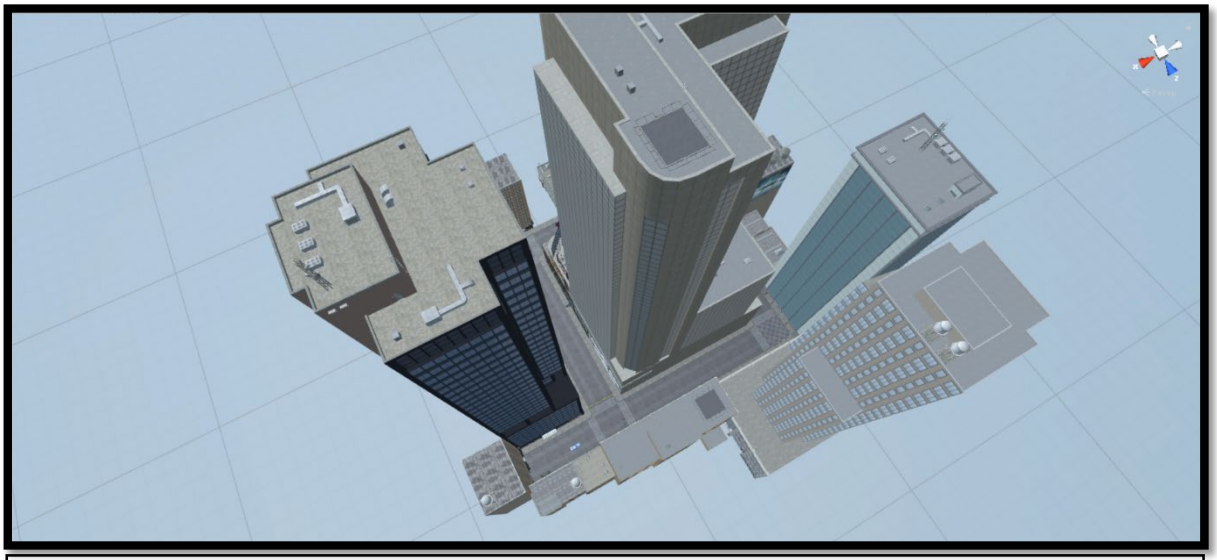


Imagen 5.26 Panorama general del nivel

Imagen 5.27 Fragmentos del nivel desde diferentes perspectivas

5.2.3.4. Nivel Intermedio

El diseño de este nivel consta de un circuito un poco más grande a comparación del nivel fácil, en el cual el usuario comenzará desde un punto inicial y su objetivo es llegar al estacionamiento del hospital que se encuentra al final del circuito. Durante el recorrido el usuario tendrá que esquivar algunos obstáculos que se colocaron en todo el circuito tal como se muestra en la imagen 5.28.

Imagen 5.28 Obstáculos del circuito

La finalidad de este nivel es que el usuario esté concentrado en todo momento con los obstáculos que se presenten durante el recorrido y pueda llegar.

En la imagen 5.29 se observa un mapa general de todo el nivel, mientras que en la imagen 5.30 se muestran partes del nivel desde diferentes perspectivas.

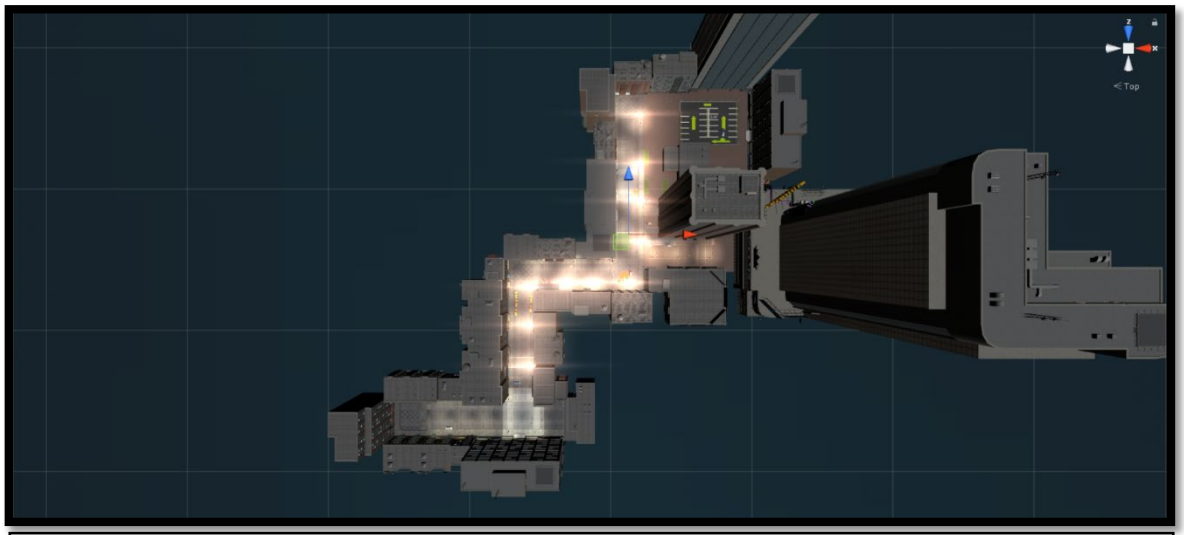


Imagen 5.29 Mapa General del Nivel

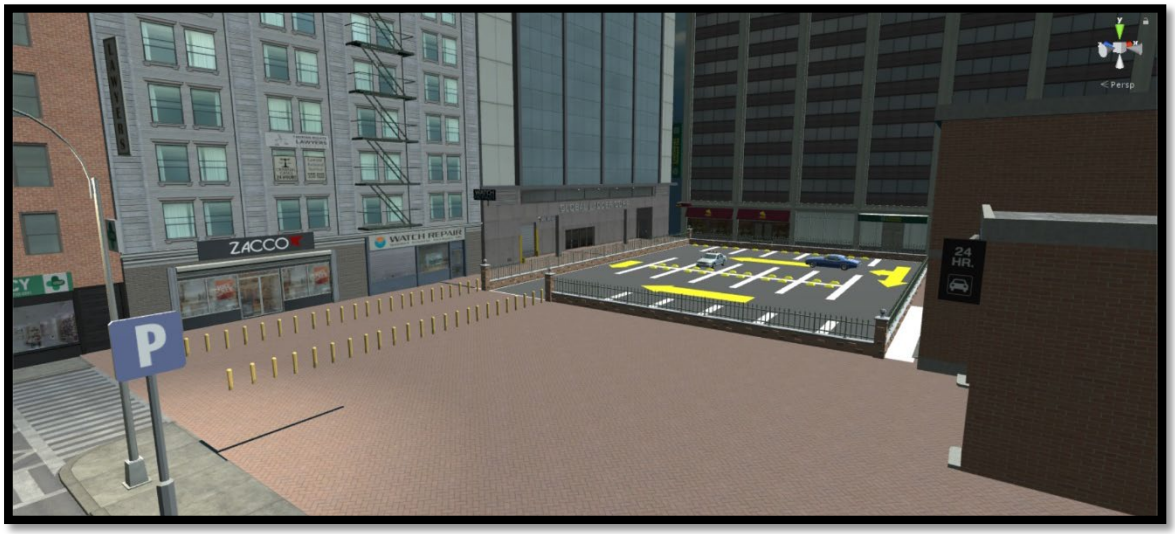


Imagen 5.30 Partes del nivel desde diferentes perspectivas.

5.2.3.5. Nivel Difícil

El diseño de este nivel consiste en un amplio escenario en donde el usuario podrá recorrer libremente el circuito y pueda poner en práctica todos los conocimientos que adquirió durante los tutoriales y los niveles anteriores.

En la imagen 5.31 se muestra un mapa general del nivel mientras que en la imagen 5.32 se muestran partes del nivel vistas desde diferentes perspectivas.

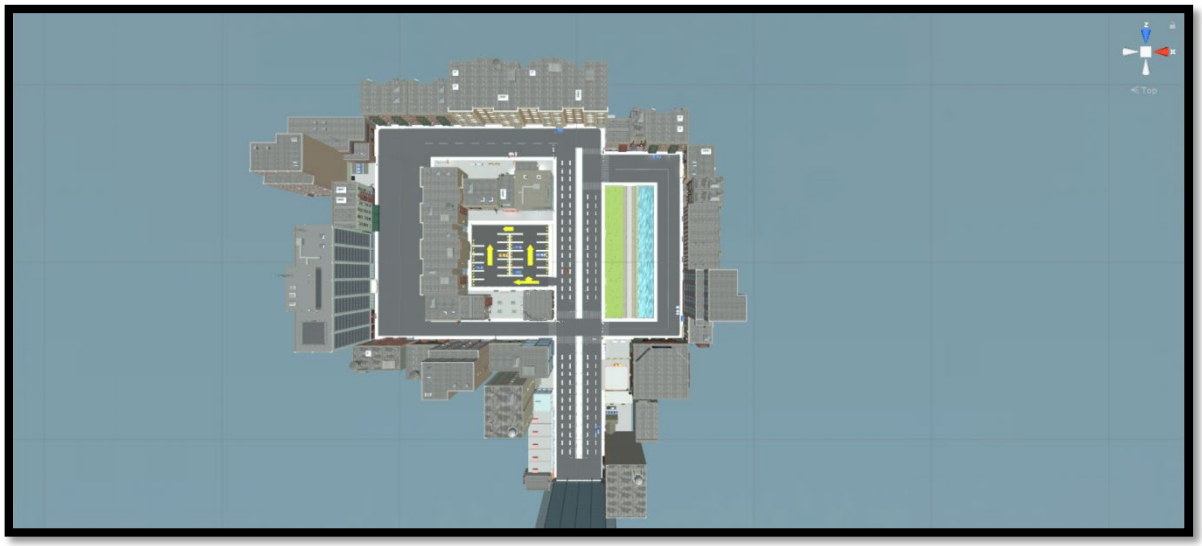


Imagen 5.31 Mapa general del nivel.

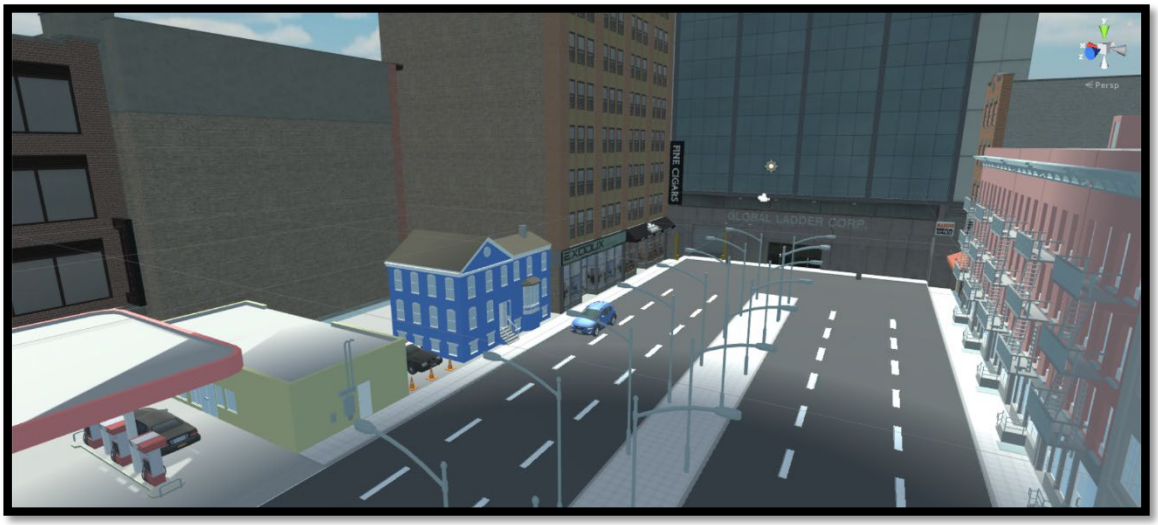


Imagen 5.32 Partes del nivel desde diferentes perspectivas.

5.2.4. Implementación del Pavimento

Para estos niveles se utilizaron dos tipos de pavimento, los creados en Blender tal como se muestra en la imagen 5.33 y los que se encontraban por defecto en el Asset Modern City Pack (MCP) los cuales se observan en la imagen 5.34.

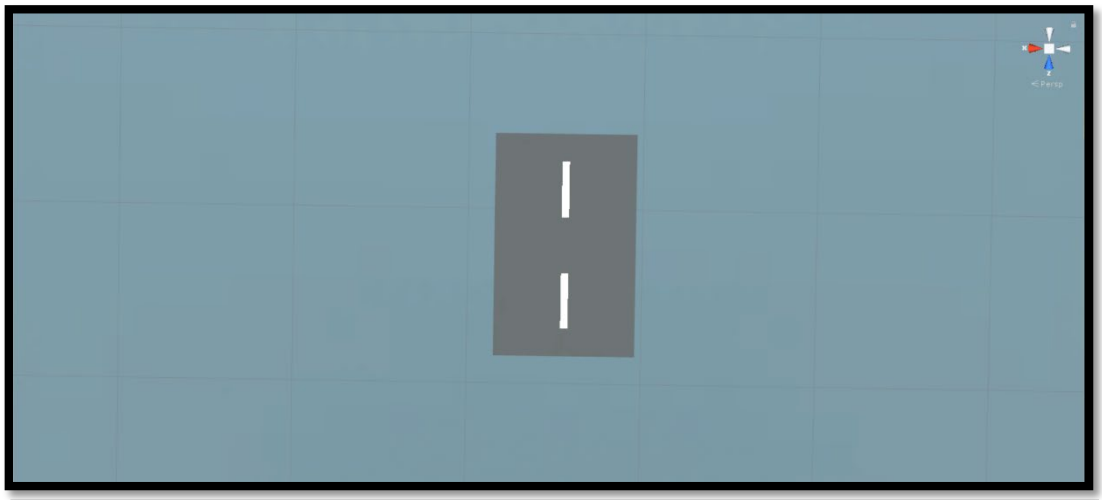


Imagen 5.33 Pavimento previamente modelo.

Imagen 5.34 Pavimento del Asset MCP

Para el nivel difícil se utilizó el pavimento previamente creado en Blender el cual se puede observar con diferentes texturas dependiendo del área en donde fue colocado como se observa en la imagen 5.35

Imagen 5.35 Texturas del pavimento dependiendo la zona

Para los niveles medio y fácil además de los dos tutoriales se utilizó el pavimento del Asset MCP el cual también tenía cambios dependiendo su colocación, ya sea una curva, o estaba cerca de un cruce tal como se muestra en la imagen 5.36.

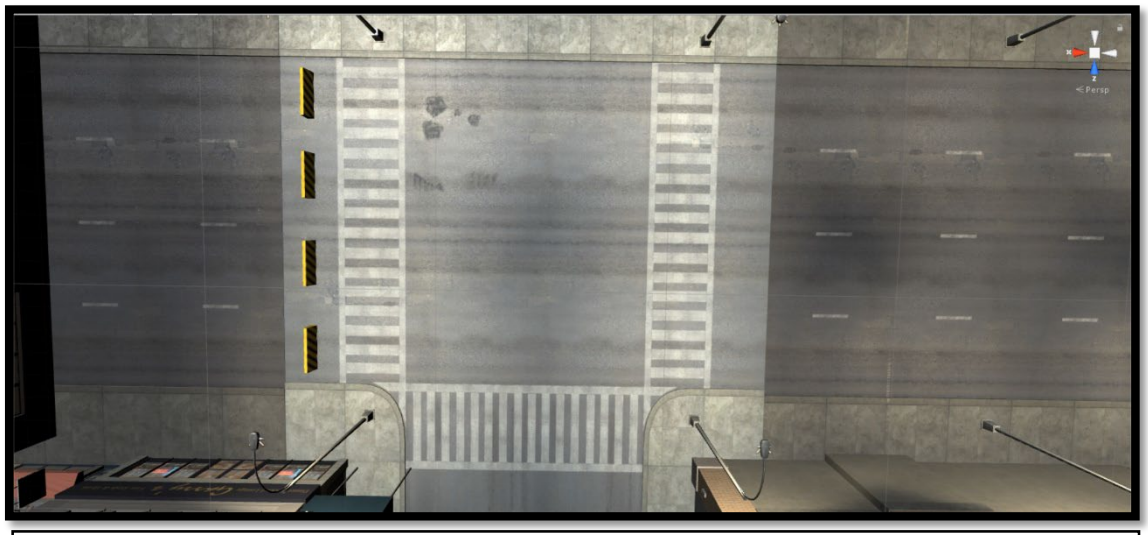
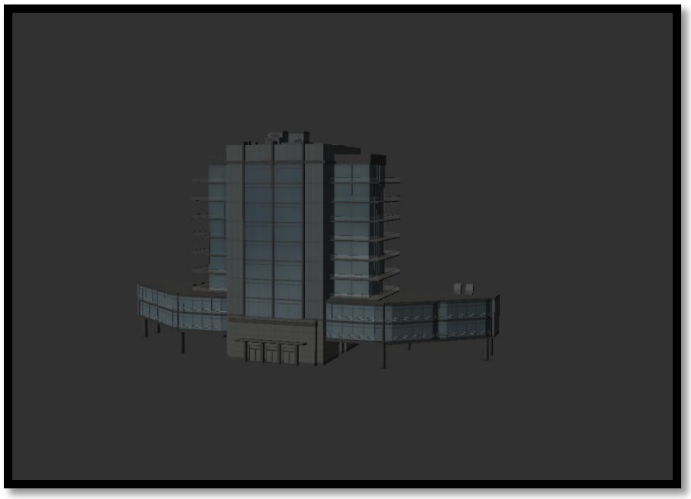


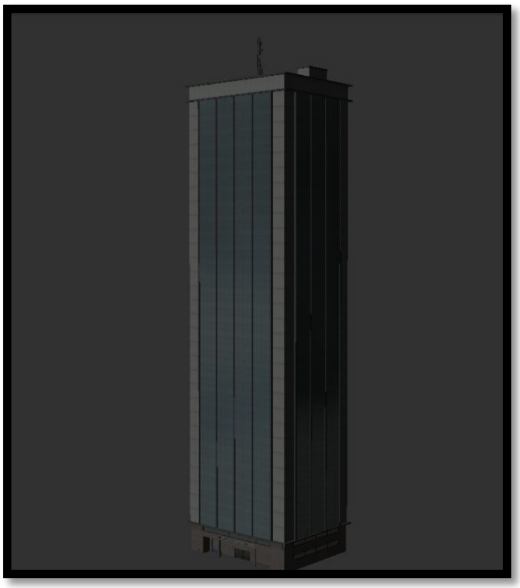
Imagen 5.36 Texturas del pavimento de MCP

5.2.5. Implementación de Edificios

Durante el recorrido, se muestran diferentes tipos de edificios los cuales se encuentran ubicados dentro el recorrido de los tres niveles. Los edificios que se utilizaron fueron descargados del Asset MCP.

Como se muestra en la imagen 5.37, se observan algunos de los diferentes edificios utilizados para la construcción de los tres niveles.





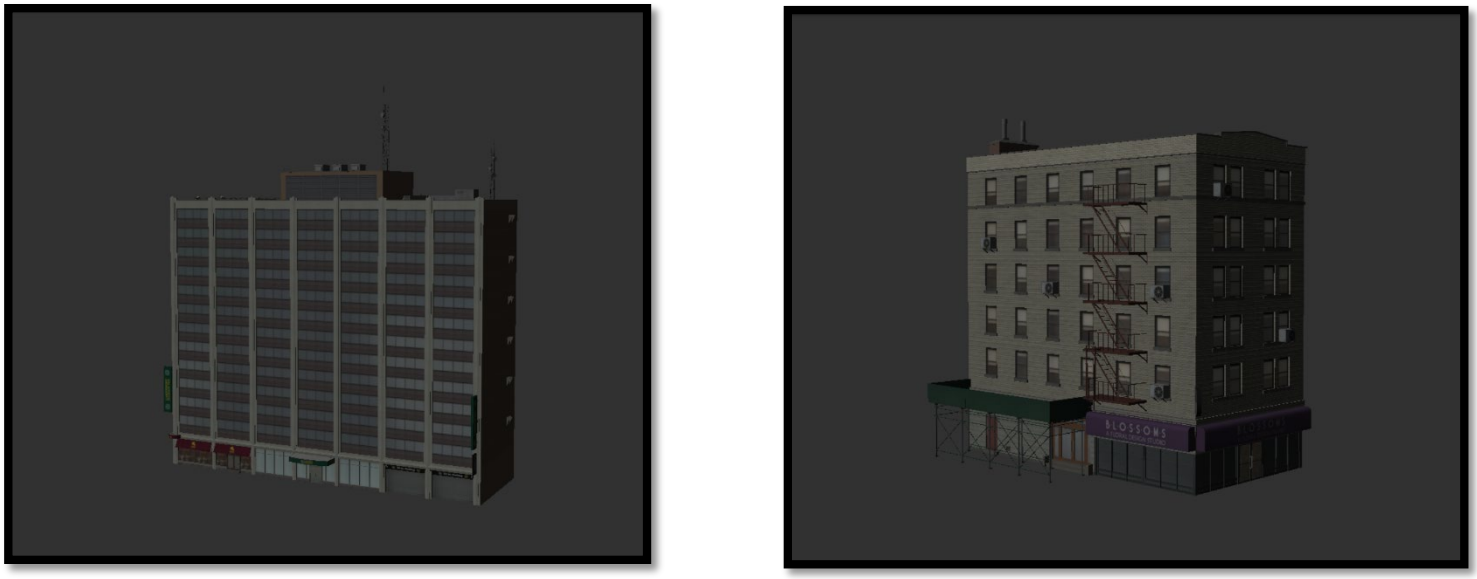


Imagen 5.37 Algunos edificios utilizados en el simulador

5.2.6. Implementación de la Decoración.

Dentro de los niveles podemos encontrar pequeños elementos los cuales se utilizaron para darle un toque más realista al nivel y así mostrar al usuario que se encuentra manejando en una verdadera ciudad.

Estos elementos decorativos fueron descargados del Asset MCP los cuales fueron ubicados dependiendo el escenario. Los elementos decorativos se describen a continuación:

- **Caseta Telefónica:** Una pequeña caseta que contiene un teléfono tal como se muestra en la Imagen 5.38.

Imagen 5.38 Caseta Telefónica

- **Parada de Autobús:** Caseta con logotipos que solo funciona como decoración tal como se muestra en la Imagen 5.39.

Imagen 5.39 Parada de Autobús

- **Carrito de Hot Dogs:** Elemento de decoración que se encuentra en el nivel difícil tal como se muestra en la Imagen 5.40

Imagen 5.40 Carrito de Hot Dogs

- **Cercas:** Permiten delimitar ciertas áreas dentro del simulador. En la imagen 5.41 se muestra como la cerca funciona para delimitar el área del estacionamiento, mientras que en la imagen 5.42 se puede observar que la cerca nos permite delimitar el área del parque.

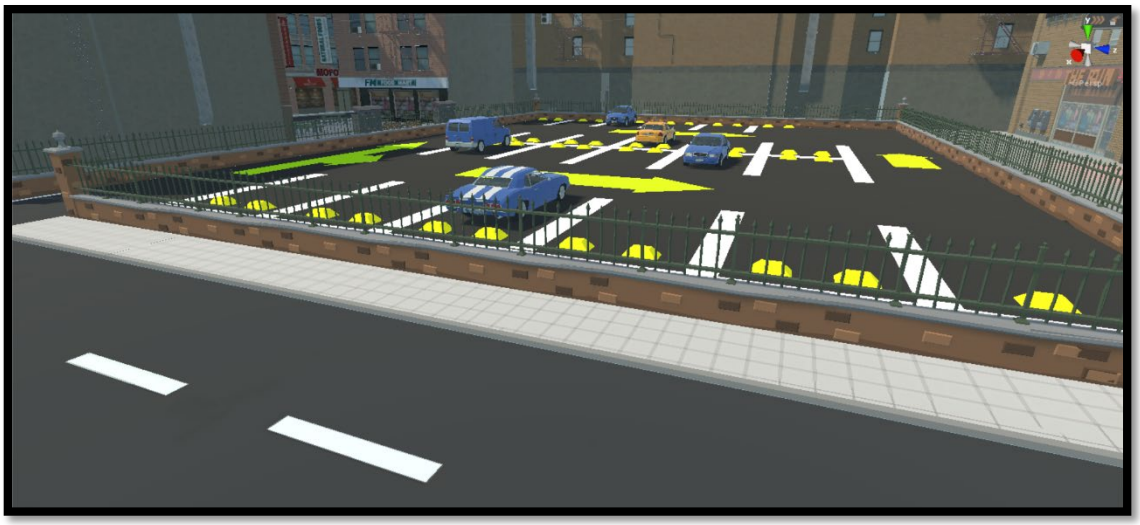


Imagen 5.41 Cerca delimitando el área de estacionamiento

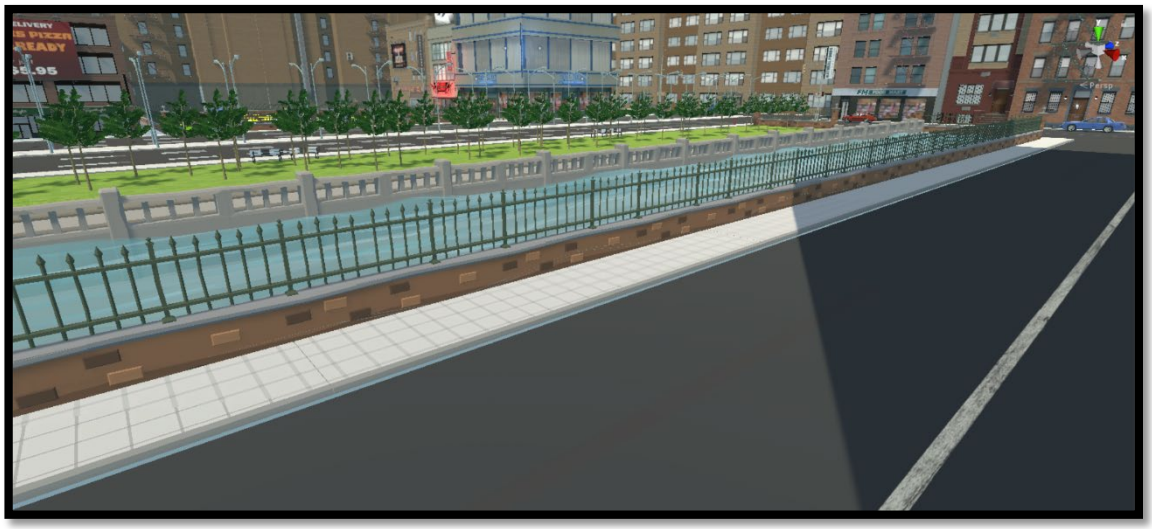


Imagen 5.42 Cerca delimitando el área del parque

- **Presas:** Este elemento decorativo nos permite delimitar el área del lago que se encuentra dentro del parque tal como se observa en la imagen 5.43.

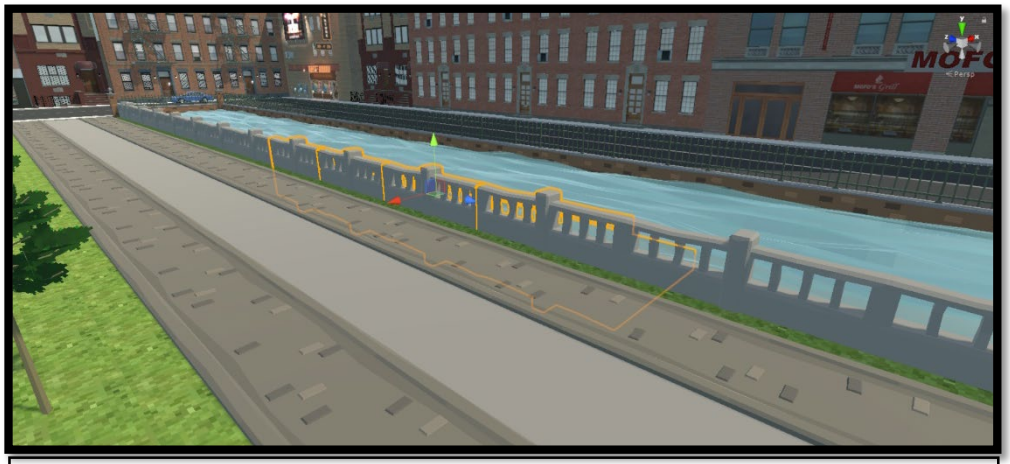


Imagen 5.43 Presas dentro del parque

- **Árboles:** Este elemento se encuentra dentro del parque como parte decorativa tal como se muestra en la imagen 5.44.

Imagen 5.44 Arboles decorativos

- **Fantasmas Peatonales:** Estos elementos permiten delimitar áreas en la cual el usuario no puede acceder. Se pueden encontrar en el nivel medio como obstáculos como se muestra en la imagen 5.45.

Imagen 5.45 Fantasmas Peatonales

- **Automóviles:** Estos automóviles los podemos encontrar en cualquiera de los tres elementos de forma estática como parte decorativa dentro de un estacionamiento o calle tal como se muestra en la imagen 5.46 o como parte de los obstáculos como se observa en la imagen 5.47.

Imagen 5.46 Automóviles decorativos

Imagen 5.47 Automóviles utilizados como obstáculos

- **Conos:** Estos elementos tienen la misma utilidad que los fantasmas peatonales, se encuentran dentro del nivel medio como parte del circuito de obstáculos como se muestran en la imagen 5.48

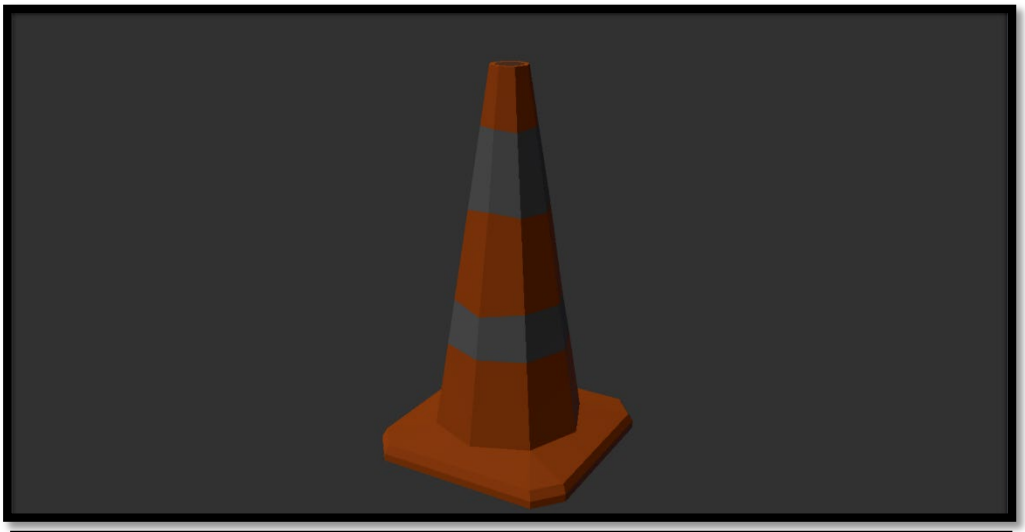


Imagen 5.48 Conos Utilizados como obstáculos

- **Basureros y bolsas de basura:** Elementos utilizados para el circuito de obstáculos tal como se muestra en la imagen 5.49.

Imagen 5.49 Basurero y bolsas utilizados como obstáculos.

- **Personas:** Este elemento se encuentra dentro del nivel medio como obstáculo para el usuario tal como se muestra en la imagen 5.50.

Imagen 5.50 Conos Utilizados como obstáculos

5.2.7. Implementación del Semáforo

Durante el recorrido de los niveles, se incorporó una de las señalizaciones la cual es muy importante ya que es el que define uno de los propósitos fundamentales del simulador el cual es el semáforo que consta de dos partes, su implementación dentro del escenario y su script (código) de funcionamiento.

5.2.7.1. Ubicación y Partes del Semáforo

Este semáforo solo se puede encontrar en zonas muy específicas del nivel difícil los cuales son los cruces del recorrido tal como se observa en la imagen 5.51. Este semáforo está constituido por los tres colores característicos y además una pequeña pantalla peatonal (el cual no es

indispensable que funcione ya que este nivel no contiene personas)

Imagen 5.51 Colocación de los semáforos dentro del nivel.

5.2.8. Checkpoints

Los Checkpoints son elementos que permiten realizar el guardado de un nivel, regresar a un punto de control cuando un personaje muere. En el caso del simulador, los Checkpoints permiten realizar un cambio de nivel cuando el usuario completo el objetivo.

5.2.8.1. Creación del Checkpoint.

Para la creación del Chekpoint se crea un cubo dentro del escenario al cual se le quita la propiedad del Mesh Renderer, además se le tiene que asignar el tag “player”.

Es importante agregar una Box Collider con la casilla de “Is Trigger” ya que de esa manera el automóvil podrá atravesarlo permitiendo realizar el cambio de nivel. En la imagen imagen 5.52 se muestra las especificaciones anteriormente explicadas.

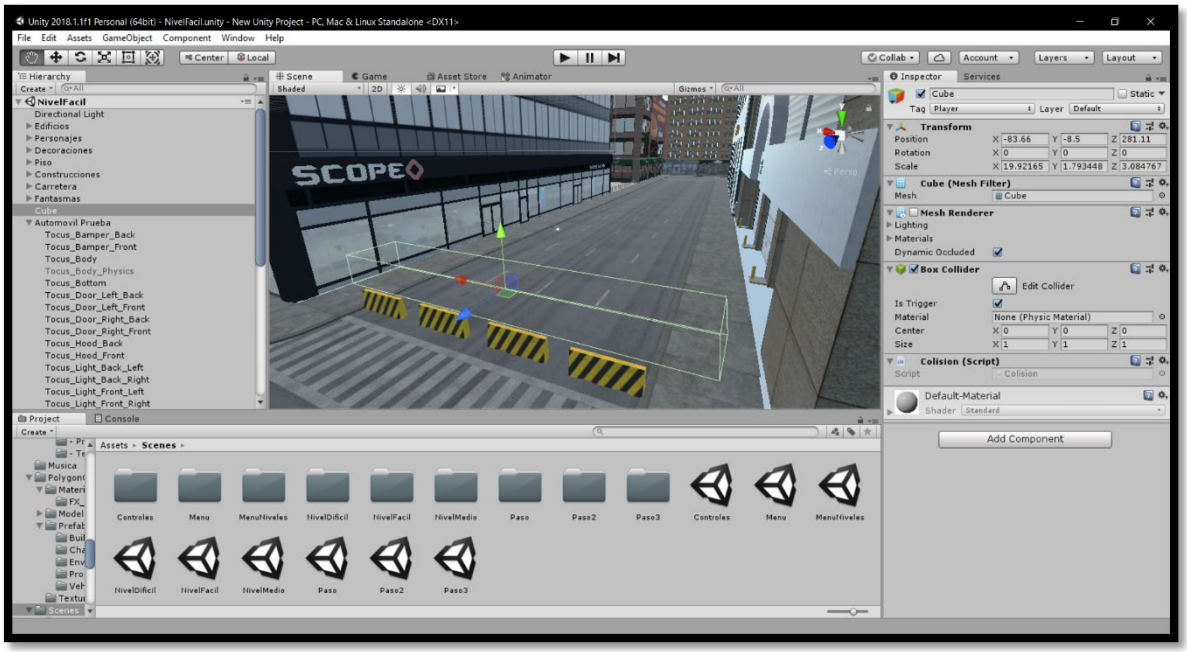


Imagen 5.52 Creación del Chekpoint con un cubo son la propiedad de Mesh Renderer.

5.2.8.2. Script del Checkpoint

Como todo Script creado en Unity, cargamos las librerías por defecto, además de una adicional llamada UnityEngine.SceneManagement la cual nos permite trabajar con variables de escena creadas en el editor. En la imagen 5.53 se muestran dichas librerías.

Se crea la clase con el nombre colisión y se comienza con la implementación de las funciones.

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

Imagen 5.53 Librerías de trabajo de Unity.

En la imagen 5.54 se muestra una función llamada OnTriggerEnter() la cual nos permite interactuar con aquellos elementos que posean cualquier tipo de colisión (Box Collider, Capsule Collider, etc).

Dentro de esta función definimos un target el cual se colocó con anterioridad en el cubo creado en el editor. Se coloca una condicional if que nos dice, si el elemento interacciona con el elemento que contiene la colisión, entonces haz el cambio de escena.

```
public class Colision : MonoBehaviour {


    void OnTriggerEnter(Collider other)
    {
        if(other.tag == "Player")
        {
            SceneManager.LoadScene("Paso");
        }
    }
}
```

Imagen 5.54 Función OnTriggerEnter y su funcionamiento.

5.2.8.3. Implementación del Checkpoint

En el cubo que previamente se creó en el editor de Unity, se le agrega el script creado. Para verificar el funcionamiento del Chekpoint, se tiene que atravesar el cubo con el automóvil para que automáticamente se realice el cambio de escena.

En este caso la escena que aparece al realizar dicha acción se muestra en la imagen 5.55 la cual nos indica que el usuario a completado el nivel.

Imagen 5.55 Pantalla que muestra cuando se ha alcanzado el Chekpoint.

5.2.9. Sistema de Diálogos

Durante el inicio de cada uno de los niveles, se presentará un cuadro de diálogos el cual indicará las instrucciones que el usuario tiene que seguir para terminar el nivel correctamente.

En la imagen 5.56 se muestra un ejemplo de como se observa este sistema de diálogos.

Imagen 5.57 Pantalla que muestra cuando se ha alcanzado el Chekpoint.

5.2.9.1. Implementación del Asset de Diálogo.

El Asset tal como se muestra en la imagen 5.58 viene empaquetado en una carpeta que contiene 3 subcarpetas correspondientes a las animaciones del cuadro de diálogo, los scripts que se requieren para la generación del diálogo, los sprites y los elementos de interfaz de usuario, así como una escena de ejemplo en la que se encuentran todos los elementos interactuando de manera funcional.

Imagen 5.58 Elementos del Asset de diálogos.

5.2.9.2. Animaciones

- **DialogueBox.controller:** Este archivo contiene la secuencia de animación para la entrada y salida de la ventana de diálogo, el cual se encuentra adjunto al objeto DialogueBox para generar la animación correspondiente. En la imagen 5.59 se muestra la construcción de la secuencia de animación.

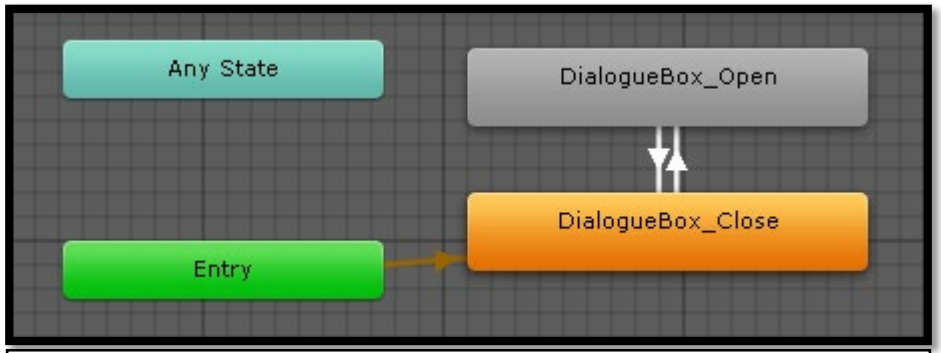


Imagen 5.59 Secuencia de animación de los diálogos.

5.2.9.3. Scripts

- **Dialogue.cs:** Define la clase dialogue que se utiliza para la construcción de cada diálogo y la cual define un nombre de NPC o personaje, y un array de frases que representan las ventanas en las que se divide el diálogo del personaje. En la imagen 5.60 se muestra el código de dialogue.cs.

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

[System.Serializable]
public class Dialogue {

    public string name;

    [TextArea(3, 10)]
    public string[] sentences;

}
```

Imagen 5.60 Script de dialogue.cs

- **DialogueManager.cs:** Este script es el encargado de recibir los datos del diálogo a generar, así como de asignar a los campos de texto Name y Dialogue, presentes en la escena, el nombre del personaje y el diálogo, respectivamente.

  Dentro de las funciones se encuentran StarDialogue, la cual se encarga de la animación de aparición de la ventana de diálogo. Recibe los parámetros del nombre del personaje y las frases que componen el diálogo, asigna el nombre al campo de texto correspondiente y crea una cola con las oraciones que van a visualizarse en la ventana de diálogo. En la imagen 5.61 se muestra la implementación de dicha función.

```
public class DialogueManager : MonoBehaviour {
    public Text nameText;
    public Text dialogueText;
    public Animator animator;
    private Queue<string> sentences;

    void Start () {
        sentences = new Queue<string>();
    }

    public void StartDialogue (Dialogue dialogue){
        animator.SetBool("IsOpen", true);
        nameText.text = dialogue.name;
        sentences.Clear();
        foreach (string sentence in dialogue.sentences)
            sentences.Enqueue(sentence);
        DisplayNextSentence();
    }
```

Imagen 5.61 Función StartDialogue

La segunda función DisplayNextSentence, se dispara por el botón de continuar cada vez que se realiza un click en dicho botón. Esta función es la encargada de mostrar la escritura de manera progresiva en la pantalla, haciendo uso de una corrutina que escribe carácter por carácter a una velocidad determinada. En la imagen 5.62 se muestra la implementación de dicha función.

```
    public void DisplayNextSentence (){
        if (sentences.Count == 0){
            EndDialogue();
            return;
        }
        string sentence = sentences.Dequeue();
        StopAllCoroutines();
        StartCoroutine(TypeSentence(sentence));
    }

    IEnumerator TypeSentence (string sentence){
        dialogueText.text = "";
        foreach (char letter in sentence.ToCharArray()){
            dialogueText.text += letter;
            yield return null;
        }
    }

    void EndDialogue(){
        animator.SetBool("IsOpen", false);
    }
}
```

Imagen 5.62 Función DisplayNextSentence

- **DialogueTrigger.cs:** Este script se encarga de recibir el nombre del personaje que interactúa en el diálogo, así como la cantidad de frases y su contenido, posteriormente crear la clase Dialogue el cual le será enviada al script DialogueManager.cs para que se inicie la secuencia. En la imagen 5.63 se muestra la implementación del script.

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

public class DialogueTrigger : MonoBehaviour{
    public Dialogue dialogue;

    private void Start(){
        Invoke("TriggerDialogue", 0.5f);
    }

    public void TriggerDialogue(){
        FindObjectOfType<DialogueManager>().StartDialogue(dialogue);
    }
}
```

Imagen 5.63 Script DialogueTrigger

5.2.9.4. Elementos de Interfaz de Usuario

Dentro de esta carpeta se encuentran las fuentes y los sprites para las ventanas y botones que se utilizaron dentro del simulador. En la imagen 5.64 se muestran dichos elementos.

Imagen 5.64 Elementos de interfaz de usuario

5.2.9.5. Dentro del Editor de Unity

Una vez configurados los scripts necesarios, dentro del editor de Unity se implementa un elemento de tipo canvas el cual nos permite trabajar con botones, texto entre otros. Para el caso del simulador la jerarquía de elementos quedaría tal como se muestra en la imagen 5.65.

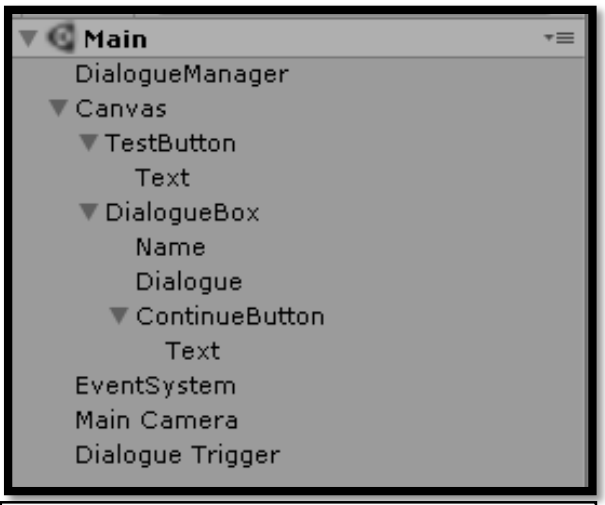


Imagen 5.65 Jerarquia de elementos dentro de la ventana de Main

A continuación, se explicará con más profundidad cada uno de los elementos:

- **DialogueManager:** Es un objeto vacío que contiene el script DialogueManager.cs y define los campos de texto que van a ser utilizados (Name y DialogueBox), así como el animator usado para la animación de la ventana. Estos elementos se los pasa como parámetros al script. En la imagen 5.66 se muestra la configuración del objeto.

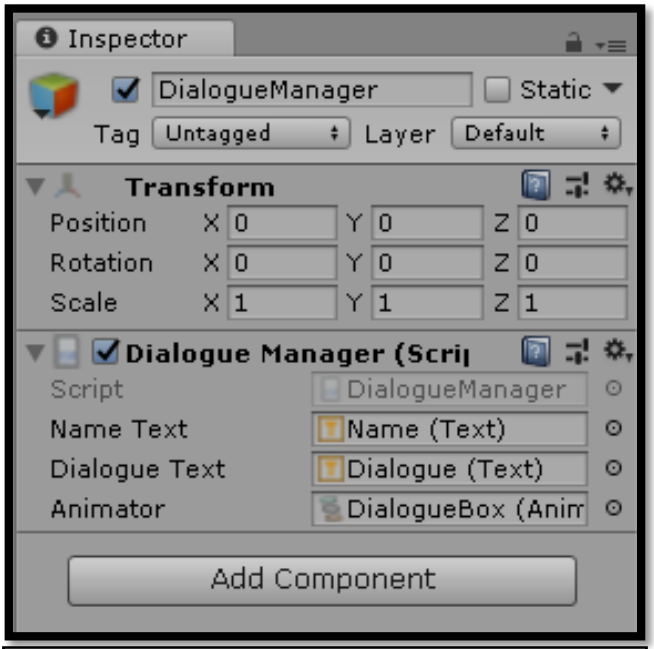


Imagen 5.66 Objeto DialogueManager ya configurado.

- **TestButton:** Es el botón que se dispara para la aparición de la ventana de dialogo y se inicializan las variables de nombre y del objeto Dialogue. Dentro de la función OnClick() hace la llamada al método TriguerDialogue() del script DialogueTrigger.cs. En la

imagen 5.67 se observa la implementación de dicho botón.

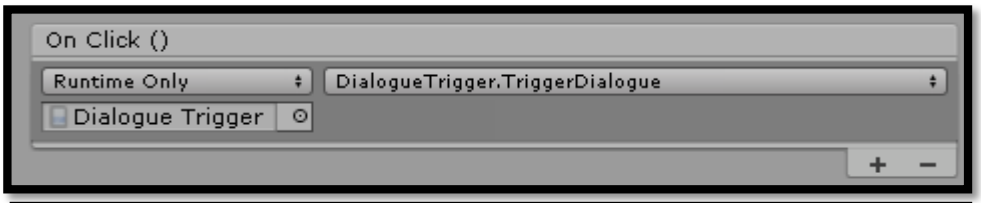


Imagen 5.67 Configuración del TestButton.

- **Dialogue Trigger:** Es un objeto vacío que se encarga de definir el nombre del personaje que está hablando y determina el número de frases, así como el contenido de cada una de ellas. Estos datos los envía al script DialogueTrigger.cs que es el encargado de crear el objeto Dialogue correspondiente con el que trabajará el script DialogueManager.cs. En la imagen 5.68 se observa la configuración de este objeto.

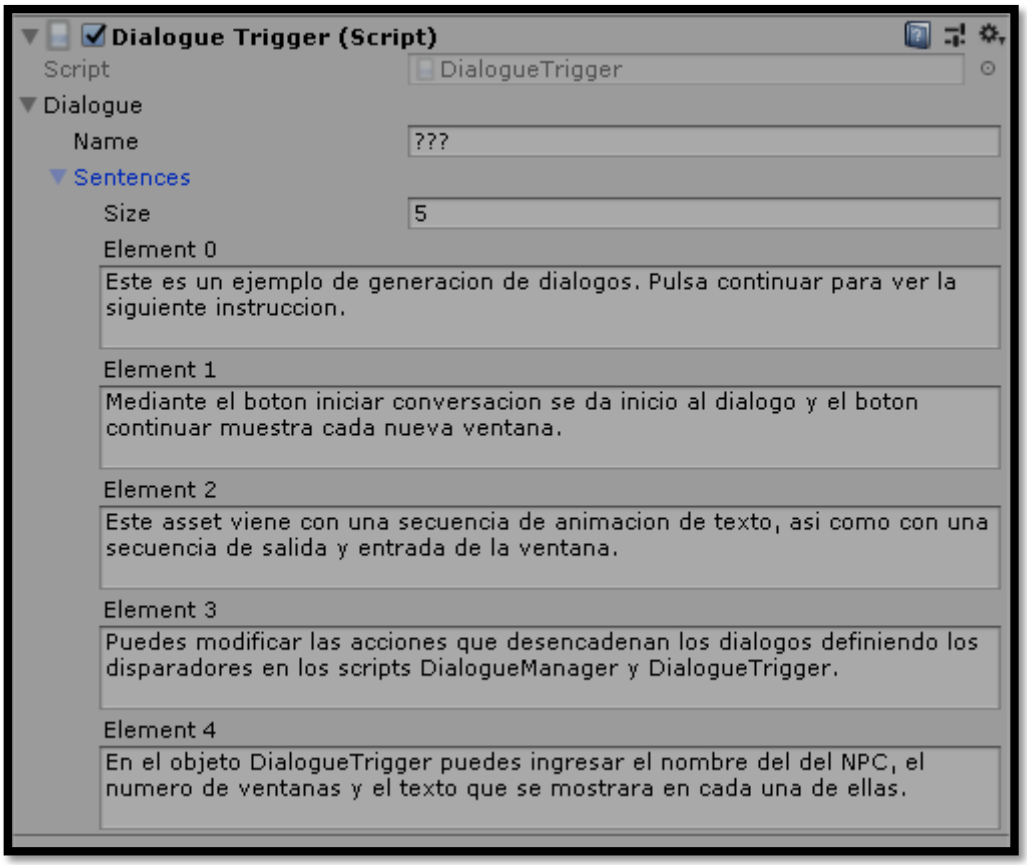


Imagen 5.68 Configuración del Dialogue Trigger.

5.2.10. Menú de Pausa

En cada uno de los niveles (fácil, medio o difícil) se implementó un pequeño menú de pausa en el cual se le brinda tres opciones al usuario: seleccionar nivel, menú principal o salir tal como se muestra en la imagen 5.69.

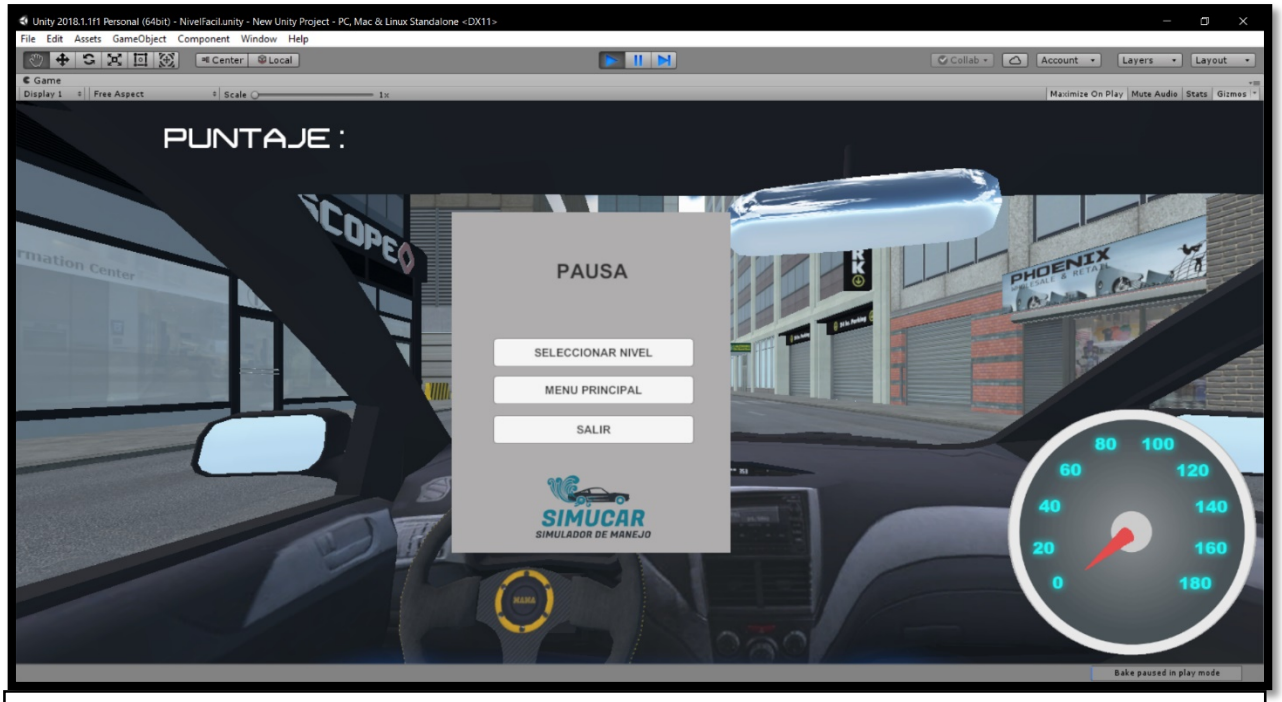


Imagen 5.69 Menú de Pausa

5.2.10.1. Script Menú de Pausa.

Para el funcionamiento de este script se crea un objeto de nombre BotonPausa, en el cual se trabajaran con las dos funciones Unity brinda por defecto: Start() y Update().

Dentro de la función Start() se definirá un Gameobject el cual será importante dentro del editor ya que es quien realizará el trabajo de pausar las acciones dentro del simulador. En la imagen 5.70 se muestra la implementación de la función Start().

```
void Start () {

    PantallaPausa = GameObject.Find ("menu");
    PantallaPausa.SetActive (false);
}
```

Imagen 5.70 Definición de la función Start.

En la función Update() se declara el funcionamiento de la pantalla del menú el cual será activada por la tecla de options del PS4 controller. Cuando ese evento inicia se activa una serie de banderas las cuales indican que el juego no tendrá movimiento, pero en el menú el usuario podrá seleccionar cualquiera de las opciones.

Cuando la tecla vuelva a ser activada el menú desaparecerá y el juego volverá a su estado normal. Esta descripción se muestra en la imagen 5.71.

```
// Update is called once per frame
void Update () {
//Poner en pausa
    if (Input.GetKeyDown(KeyCode.JoystickButton9)){
        if (pausado == false){
            pausado = true;
            PantallaPausa.SetActive (true);
            Time.timeScale = 0;
        }else {
            pausado = false;
            PantallaPausa.SetActive (false);
            Time.timeScale = 1;
        }
    }
}
```

Imagen 5.71 Definición de la función Update.

5.2.10.2. Diseño del Menú de Pausa Dentro de la Interfaz de Usuario.

En la imagen 5.72 se muestra la jerarquía de elementos que conforman el menú de pausa los cuales se explicaran a continuación:

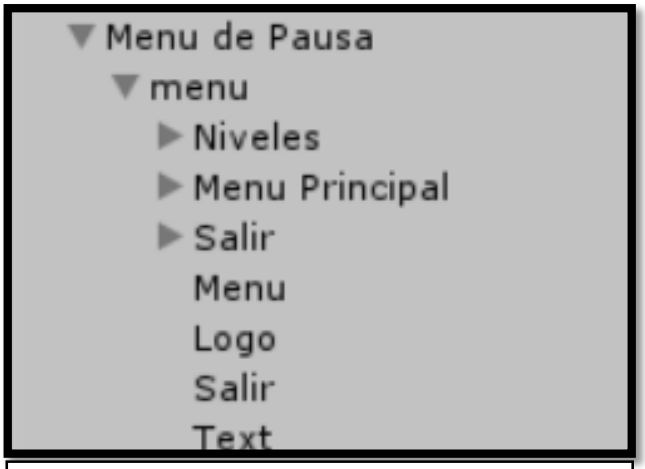


Imagen 5.72 Jerarquía del menú de pausa

- **Menú:** es un gameobject en el cual es la base para la implementación de los botones y texto. En este game object se configura una música de fondo cuando se activa desde el ps4 controller tal como se muestra en la imagen 5.73

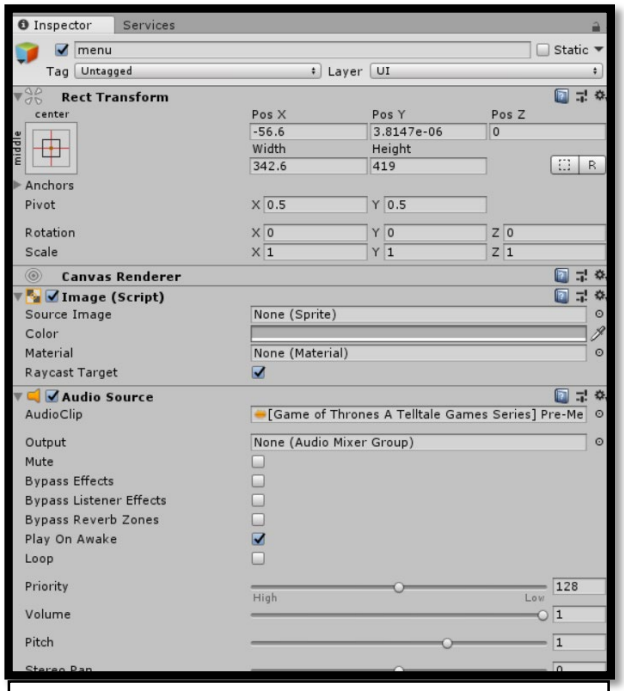


Imagen 5.73 Configuración de menú

- **Botones:** Dentro de la jerarquía se encuentran 3 botones (Niveles, Menú Principal, Salir) los cuales permiten cambiar de escena dependiendo a lo que el usuario requiera en ese momento. En la imagen 5.73 se muestra como está configurado uno de los botones para su funcionamiento.

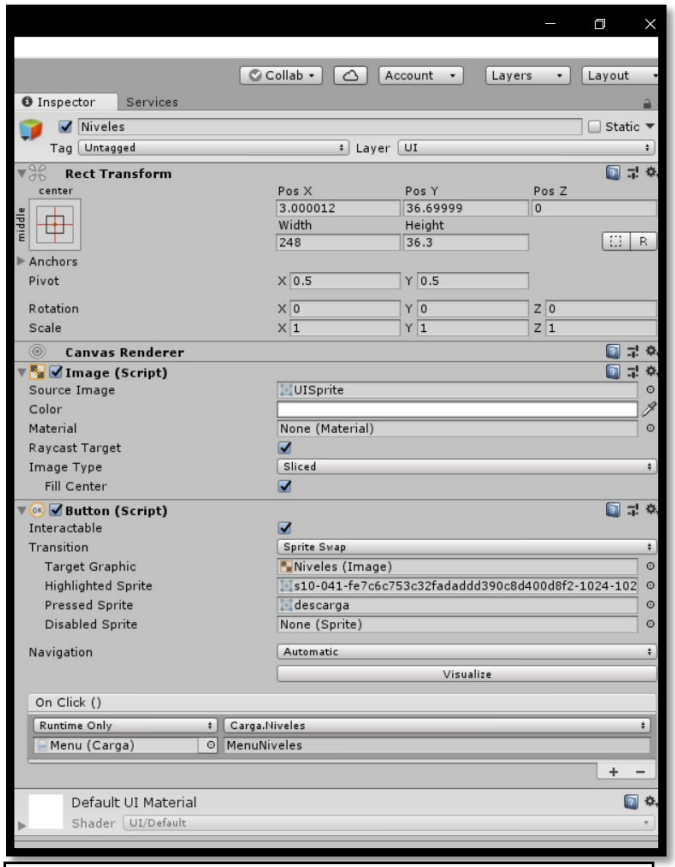


Imagen 5.73 Configuración de los botones

5.3. Implementación del Automóvil

La pieza fundamental del simulador es el automóvil cuya función es adaptarse a las órdenes del usuario mediante los controles, ya sea avanzar, retroceder o frenar. Para fines educativos se utilizó un automóvil el cual permitiera ser de fácil implementación además de fácil modificación para su movimiento. Este automóvil fue descargado de la Asset Store tal como se muestra en la imagen 5.74.

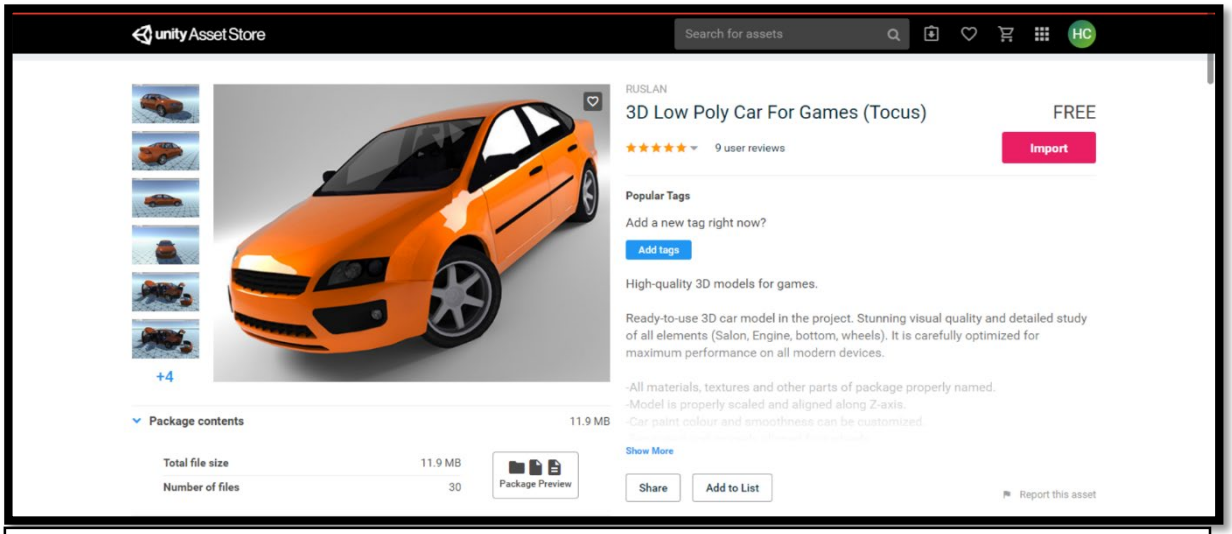


Imagen 5.74 Modelo del automóvil utilizado en el simulador descargado desde el Asset Store.

Al descargar e importar el modelo, se creará una carpeta en donde se encuentran unas carpetas y archivos tal como se muestra en la imagen 5.75, los cuales se describen a continuación:

- **Materials:** Carpeta destinada a los materiales que le dan color al automóvil.
- **Prefab:** Carpeta en donde se encuentra el modelo del automóvil que se utilizó dentro del simulador
- **Textures:** Carpeta destinada a las texturas que conforman al automóvil
- **Scene:** Escena agregada por defecto donde nos da una muestra del funcionamiento del modelo.

Imagen 5.75 Elementos que integran al automóvil.

Para poder visualizar el automóvil dentro de la escena del editor, dentro de la carpeta prefab se encuentra un modelo de un automóvil el cual se tiene que arrastrar dentro de la escena para que aparezca tal como se muestra en la imagen 5.76.

Imagen 5.76 Modelo del automóvil dentro de la escena.

5.3.1. Funcionamiento

El objetivo del simulador es que el usuario sienta la inmersión de que está dentro de un automóvil conduciendo. Es por ello que se consideran ciertas funcionalidades las cuales son muy importantes de implementar para cumplir dicho objetivo.

5.3.1.1. Implementación de la Cámara

El objetivo de la cámara es centralizar un punto específico del automóvil (área interna del automóvil) para que el usuario pueda experimentar que se encuentra dentro del automóvil.

Dentro de Unity en el área de Hierarchy, se crea un objeto de tipo cámara tal como se muestra en la imagen 5.77.

Una vez que el objeto de tipo cámara se haya creado, el elemento se tiene que agregar al modelo del automóvil de nombre “Automóvil prueba” el cual heredará todos los elementos que lo constituyan. Una vez que la cámara se encuentra dentro del modelo, se acomoda de tal forma que se tenga una perspectiva interna del automóvil tal como se muestra en la imagen 5.78.

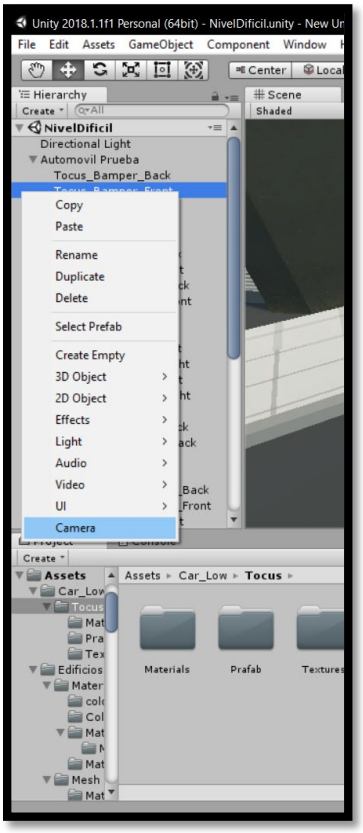


Imagen 5.76 Creación del objeto cámara.

Imagen 5.77 La cámara se encuentra dentro del automóvil.

5.3.1.2. Colisiones

Las colisiones nos permiten que el modelo del automóvil no atraviese los objetos que se encuentran dentro del simulador. Para esta implementación se utilizaron dos tipos de colisiones, el box Collider y las Wheel Colliders

5.3.1.2.1. Box Collider.

Box Collider es una caja invisible la cual permite envolver al objeto y evitar que esté atraviese los objetos. Box Collider al ser implementada, muestra ciertos parámetros los cuales pueden ser modificados dependiendo su aplicación. En la imagen 5.78 se muestra dichos parámetros a modificar.

- **Edit. Collider:** Opción que permite modificar la caja de colisiones de forma libre.

- **Is Trigger:** Si esta opción está habilitada, el Collider será utilizado para eventos de triggering e ignora los posibles contactos con los elementos en su entorno.
- **Material:** Opción que determina en qué forma el Collider interactúa con otros.
- **Center:** Opción que posiciona el Collider en el espacio local del objeto.
- **Size:** Determina el tamaño del Collider en las direcciones X, Y, Z.

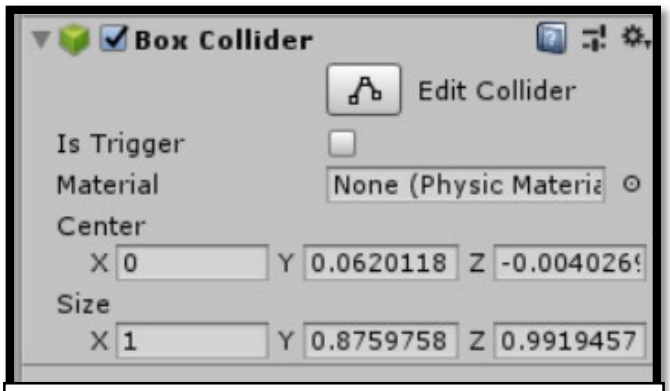


Imagen 5.78 Parámetros que modifican el Box Collider.

Dentro del escenario, el Box Collider rodea la parte superior del modelo con un cubo de líneas verdes tal como se muestra en la imagen 5.79.

Imagen 5.79 Aplicación del Box Collider sobre el Automóvil.

5.3.1.2.2. Wheel Collider

El Wheel Collider es un collider especial que solo se utiliza para vehículos del suelo. Cuenta con una detección de colisión integrada, física de ruedas, y un modelo de deslizamiento basado en la fricción de la llanta. Puede ser utilizado para objetos que no sean ruedas, pero es específicamente diseñado para vehículos con ruedas.

En la imagen 5.80 se muestran los principales componentes que describen a este Collider.

- **Mass:** Masa de la rueda
- **Radius:** Radio de la rueda
- **Wheel Damping Rate:** Valor de amortiguación aplicado a la rueda.

- **Suspension Distance:** Distancia máxima de extensión de la suspensión de una rueda, medida en el espacio local.
- **Force App Point Distance:** Parámetro que define el punto donde las fuerzas de la rueda serán aplicadas.
- **Center:** Centro de la rueda en el espacio local del objeto.
- **Suspension Spring:** Tipo de suspensión que intenta alcanzar una Target Position al agregar fuerzas de spring (resorte) y damping (amortiguación)
- **Spring:** La fuerza del resorte intenta alcanzar una Target Position.
- **Damper:** Amortigua la velocidad de suspensión.
- **Target Position:** Distancia de descanso de la suspensión a lo largo de la Suspension Distance.
- **Forward/Sideways Friccion:** Propiedades de la fricción de la llanta cuando la rueda está rodando hacia adelante y hacia los lados.

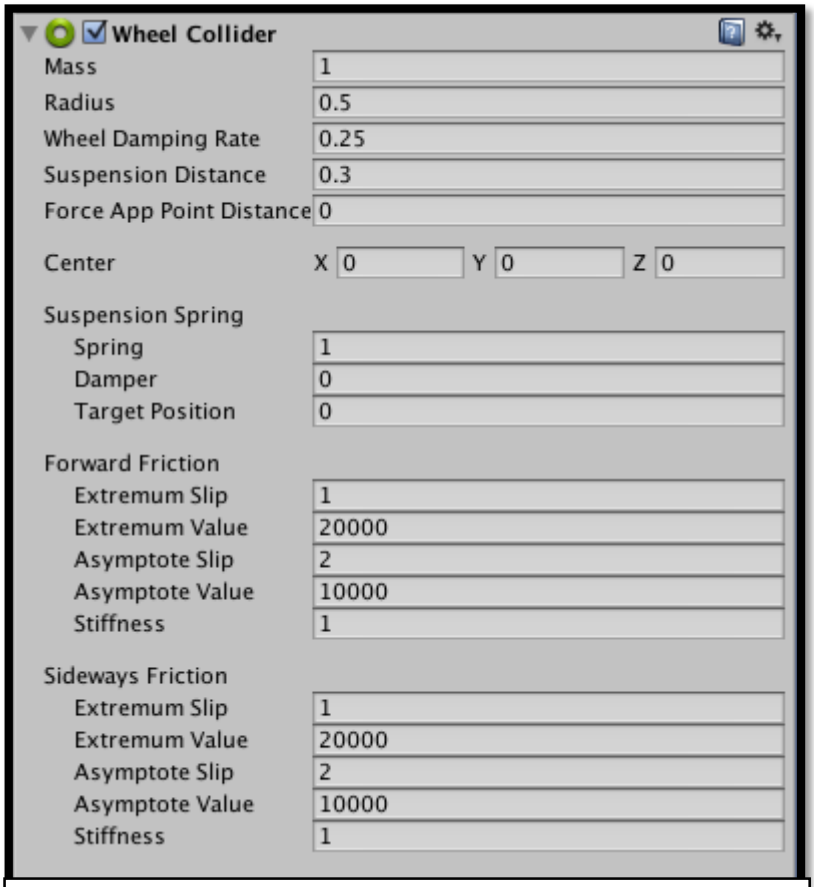


Imagen 5.80 Parámetros del Wheel Collider

Dentro de la escena, las Wheel Collider actúan sobre las cuatro llantas del modelo del automóvil tal como se muestran en la imagen 5.81. Estas Collider aparecen con un círculo de líneas verdes.

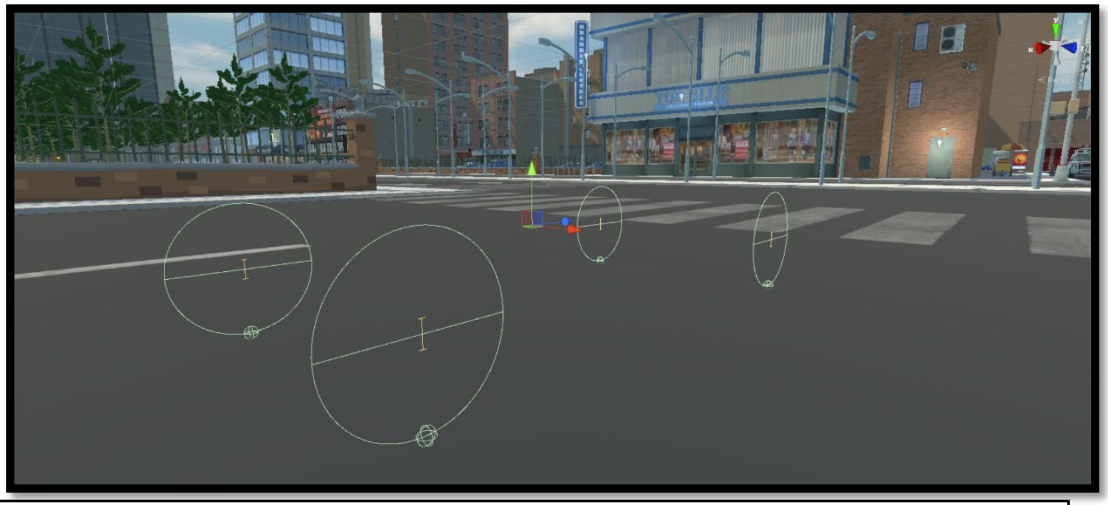


Imagen 5.81 Uso de las Wheel Collider

5.3.1.3. Script de Manejo

5.3.1.3.1. Creación del Script

El script de manejo es el que permite que el automóvil dentro de la escena obtenga movimiento. El script funciona de la siguiente manera:

- Al crear el script, Unity brinda tres bibliotecas las cuales son importantes para el desarrollo de las aplicaciones tal como se muestra en la imagen 5.82.

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

Imagen 5.82 Librerías por defecto de Unity

- Después se tiene una clase de tipo pública llamada Movimiento (es importante mencionar que el script debe de tener el mismo

nombre que la clase) en la cual se van declarando variables que solo afectarán al objeto seleccionado.

Dentro de la clase Movimiento, se comenzará declarando variables de tipo público relacionadas a un elemento WheelCollider tal como se muestra en la imagen 5.83, las cuales nos permiten utilizar las Wheel Collider previamente colocadas en las llantas del automóvil.

```
public WheelCollider Llanta1;
public WheelCollider Llanta2;
public WheelCollider Llanta3;
public WheelCollider Llanta4;
```

Imagen 5.83 Declaración de variables de tipo WheelCollider.

- Posteriormente dentro de la zona de declaración de variables, se declaran variables de tipo público que corresponderán al movimiento, frenado y giro del automóvil tal como se muestra en la imagen 5.84.

```
//Fuerza de rotacion que se le aplica a las ruedas para que giren y avanzen
public float Motor;
//Fuerza maxima del motor
public float fuerzamaxima;
//Valor de giro de las ruedas delanteras
public float giroRuedas;
//Fuerza maxima de giro
public float maximogiro;
//Freno
public float freno;
//Indica que el auto esta frenando
public bool Frenado;
//Posicion del centro de masa del vehiculo
public Vector3 centrodemasa;

//Posicionar el vehiculo con el centro de masa
public Vector3 posicionrueda;
public Quaternion rotacionrueda;
```

Imagen 5.84 Declaración de variables para el movimiento, frenado y giro del automóvil.

- Los objetos de tipo Transform nos permiten darles animación a los objetos dentro de la escena. En este caso la animación será destinada a las llantas del automóvil, para ello a cada una de ellas se les asigna el objeto de tipo Transform como se muestra en la imagen 5.85.

```
//Para hacer animacion de las ruedas
public Transform Neumatico1;
public Transform Neumatico2;
public Transform Neumatico3;
public Transform Neumatico4;
```

Imagen 5.85 Variables de tipo Transform para la animación de las ruedas.

- Dentro de la función Start() se definen los valores iniciales de nuestras variables de movimiento previamente declarados recordando que estas variables solo serán aplicada cuando la simulación comience. En la imagen 5.86 se muestra la asignación de valores a las variables.

```
// Use this for initialization
void Start()
{
    Motor = 0.0f;
    fuerzamaxima =3000.0f;
    giroRuedas = 0.0f;
    maximogiro = 25.0f;
    centrodemasa = Physics.gravity = new Vector3(0.0f, -9.8f, 0.0f);
    Frenado = false;

    this.gameObject.GetComponent<Rigidbody>().centerOfMass = centrodemasa;
    //Physics.gravity = new Vector3(0, -9.8F, 0);
}
```

Imagen 5.86 Asignación de valores a las variables de movimiento dentro de la función Start().

- Con anterioridad se declararon variables de tipo Transform las cuales permitirán crear una pequeña animación de movimiento a las ruedas del automóvil. Dentro de la función Update() se declaran tanto el giro en forma vertical y horizontal de dichas ruedas tal como se muestra en la imagen 5.87.

```
void Update()
{
    //Rotacion de los Neumaticos
    Neumatico1.localEulerAngles = new Vector3(cuentaAngulo * Llanta1.rpm / 60 * 360 * Time.deltaTime, Llanta1.steerAngle, 0);
    Neumatico2.localEulerAngles = new Vector3(cuentaAngulo * Llanta2.rpm / 60 * 360 * Time.deltaTime, Llanta2.steerAngle, 0);
    cuentaAngulo++;
    if (cuentaAngulo > 360) cuentaAngulo = 0;

    Neumatico3.Rotate(new Vector3(Llanta3.rpm / 60 * 360 * Time.deltaTime, 0.0f, 0.0f));
    Neumatico4.Rotate(new Vector3(Llanta4.rpm / 60 * 360 * Time.deltaTime, 0.0f, 0.0f));
}
```

Imagen 5.87 Declaración del movimiento de las ruedas

- Parte importante del movimiento del automóvil es la manera en que el usuario utilizará los controles de movimiento (Control Play Station 4, Volante Logitech o teclado). Para ello dentro de la función FixedUpdate() se declaran los Inputs (teclas) los cuales nos permitirán tener control del automóvil y así permitir su movimiento.

  En la imagen 5.88 se muestran la configuración de los Inputs para la aceleración, frenado y giro mientras que en la imagen 5.89 se observa la configuración del Input para la desaceleración del automóvil.

```
// Update is called once per frame
void FixedUpdate()
{
    //Declaramos las teclas para el movimiento, giro y frenado
    Motor = fuerzamaxima * Input.GetAxis("Vertical");
    giroRuedas = maximogiro * Input.GetAxis("Horizontal");
    if (Input.GetButton("Jump"))
    {
        Frenado = true;
    }
    else
    {
        Frenado = false;
    }
```

Imagen 5.88 Configuración de los Inputs para el movimiento, giro y frenado del automóvil

```
//Condicion para que le coche desacelere al momento de dejar de presionar el boton
if (Input.GetAxis("Vertical") == 0)
{
    Llanta1.brakeTorque = desacelreacion;
    Llanta2.brakeTorque = desacelreacion;
}
else
{
    Llanta1.brakeTorque = 0;
    Llanta2.brakeTorque = 0;
}
```

Imagen 5.89 Configuración del Input para la desaceleración del automóvil.

- Por último, se asignan las fuerzas de aceleración y de frenado a las ruedas del automóvil tal como se muestra en la imagen 5.90.

```
    //Asignacion de las ruedas para el giro
    Llanta1.steerAngle = giroRuedas;
    Llanta2.steerAngle = giroRuedas;

    //Asignacion de las ruedas para la aceleracion
    Llanta1.motorTorque = Motor;
    Llanta2.motorTorque = Motor;
    Llanta3.motorTorque = Motor;
    Llanta4.motorTorque = Motor;

    //Fuerza de frenado
    if (Frenado)
    {
        Llanta1.brakeTorque = 1000.0f;
        Llanta2.brakeTorque = 1000.0f;
        Llanta3.brakeTorque = 1000.0f;
        Llanta4.brakeTorque = 1000.0f;
    }
    else
    {
        Llanta1.brakeTorque = 0.0f;
        Llanta2.brakeTorque = 0.0f;
        Llanta3.brakeTorque = 0.0f;
        Llanta4.brakeTorque = 0.0f;
    }

}
```

Imagen 5.90 Asignación de fuerzas a las ruedas.

5.3.1.3.2. Implementación del Script de Manejo Dentro de la Escena.

Una vez que es script de manejo es compilado y sin errores, dicho script tiene que ser implementado dentro del modelo del automóvil.

Para ello, el script se debe colocar dentro del automóvil de nombre “Automóvil Prueba” el cual utilizará todos los elementos declarados en el script tal como se muestra en la imagen 5.91.

Imagen 5.91 Script “Movimiento” asignado al automóvil “automóvil Prueba.

Una vez que el script es asignado al modelo, los elementos llanta (Llanta 1, Llanta 2, Llanta 3, Llanta 4) se le asignarán los correspondientes Wheel Collider que previamente se colocaron dentro de las llantas del automóvil.

Los elementos Neumáticos (Neumático 1, Neumático 2, Neumático 3, Neumático 4) tendrán los elementos correspondientes a los modelos de las llantas los cuales se encuentran dentro del automóvil. Con esto tendríamos la animación de las ruedas al momento de acelerar o frenar.

Los demás elementos corresponden a las fuerzas de aceleración, frenado, giro cuyos valores irán cambiando dependiendo la utilidad que se necesite del automóvil.

5.3.1.4. Velocímetro.

El velocímetro es uno de los elementos más importantes dentro del simulador, ya que indica ciertas acciones como la velocidad del automóvil y el frenado del mismo.

5.3.1.4.1. Script del Velocímetro

Para su construcción se necesitó de un script el cual describe el comportamiento de las acciones y como debe de estar configurado para su buen funcionamiento.

En el script se declararon ciertas variables las cuales nos definen que tenemos que trabajar con variables de tipo imagen, además de establecer ciertos parámetros como el ángulo de giro, y la velocidad en la cual avanzará el velocímetro.

En la imagen 5.92 se muestra la declaración de dichas variables.

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

public class Speedometer : MonoBehaviour {

    public Texture2D speedometer_main_tex;
    public Texture2D speedometer_needle_tex;

    Rect speedometer_main_rect;
    Rect speedometer_needle_rect;

    Vector2 needle_pivot;

    float needle_angle = 0;
    public float zero_angle = -130;
    public float speed_add_value = 1;
    float speed;

    public bool isKph = true;
```

Imagen 5.92 Implementación de variables del velocímetro.

Una vez que se asignaron las variables correspondientes, se crea una función llamada OnGUI() la cual permite crear funciones donde se definirán el comportamiento del velocímetro y como es que tiene que estar acoplado al automóvil. En la figura 5.93 se muestra la implementación de la función.

```
void OnGUI() {

    // Calculate the desired rects and the GUI size
    float speedometer_main_size = 0.512f * ((Screen.height / 3.5f) + (Screen.width / 3.5f));
    float speedometer_needle_sizeX = (Screen.height / 110f) + (Screen.width / 110f);
    float speedometer_needle_sizeY = (Screen.height / 30f) + (Screen.width / 30f);

    speedometer_main_rect = new Rect(Screen.width - speedometer_main_size, Screen.height - speedometer_main_size, speedometer_main_size, speedometer_main_size);
    speedometer_needle_rect = new Rect(Screen.width - (speedometer_main_size / 2) - (speedometer_needle_sizeX / 2),
                                       Screen.height - (speedometer_main_size / 2) - (speedometer_needle_sizeY),
                                       speedometer_needle_sizeX, speedometer_needle_sizeY);

    // Pivot rotation Vector2
    needle_pivot = new Vector2(speedometer_needle_rect.x + (speedometer_needle_sizeX / 2), speedometer_needle_rect.y + (speedometer_needle_sizeY));

    // Draw the speedometer
    GUI.DrawTexture(speedometer_main_rect, speedometer_main_tex);

    //Get the speed of the vehicle
    if(isKph == true) {

        speed = transform.GetComponent<Rigidbody>().velocity.magnitude * 4.6f;

    } else if(isKph == false) {

        speed = transform.GetComponent<Rigidbody>().velocity.magnitude * 2.237f;

    }
```

Imagen 5.93 Implementación de la función OnGUI.

5.3.1.4.2. Asignación del Velocímetro al Automóvil

Para la asignación del velocímetro, el script se tiene que agregar dentro del “Automóvil Prueba”. Para poder efectuar correctamente el script, se anexarón imágenes las cuales servirían de apoyo para el funcionamiento del mismo. En la imagen 5.94 se muestra la base del Velocímetro y su respectivo puntero.

Una vez que estos elementos se anexaron al automóvil, el script realiza su función el cual solo se activa cuando se ingresa a cualquiera de los tres niveles. En la figura 5.95 se muestra el funcionamiento del velocímetro dentro de uno de los niveles del simulador.

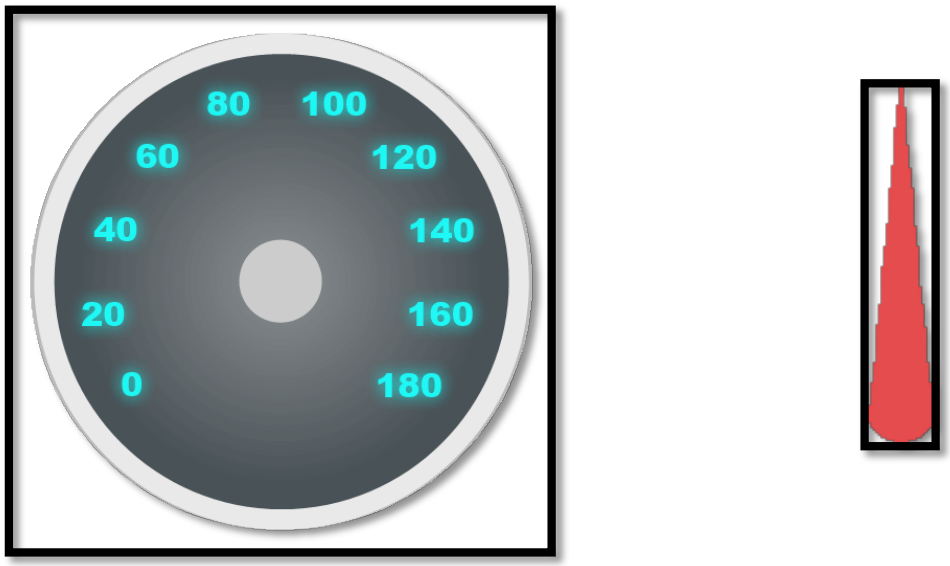


Imagen 5.94 Base y puntero del Velocímetro

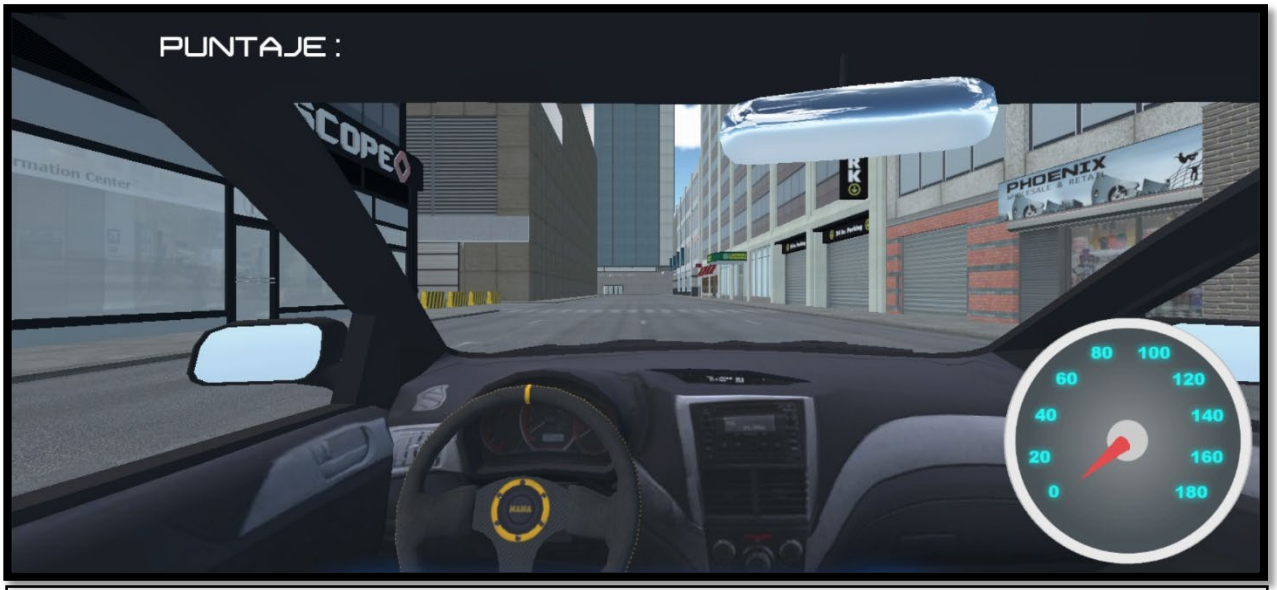


Imagen 5.95 Funcionamiento del Velocímetro dentro del simulador.

5.4. Configuración de los Inputs

Anteriormente en el script de “Movimiento” se había configurado los Inputs que se necesitan para poder tener control del automóvil. Dentro de Unity hay un bloque que nos permite realizar la configuración de las teclas para cualquier control.

Dentro de la ventana de trabajo de Unity, se selecciona la opción de Edit. El cual desplegará un pequeño menú tal como se muestra en la imagen 5.96. Dentro del menú se selecciona la opción de Input.

Imagen 5.96 Selección de la opción Input dentro del menú de Edit.

Dentro de Input Manager se observa los Axes los cuales son llamados desde el script de “Manejo” y se tienen que configurar de acuerdo a las teclas que el usuario desee utilizar.

5.4.1. Configuración Inputs (Teclado)

El primer elemento a configurar es el teclado el cual utiliza los inputs “Vertical”, “Horizontal”, y “Jump”. En la imagen 5.97 se muestra la configuración de las teclas correspondientes al teclado.

En el Axes Horizontal le corresponden las teclas right y left además de sus equivalentes a y d los cuales permiten el movimiento adelante y atrás del automóvil.

El Axes Vertical le corresponden las teclas up y down los cuales permiten el giro derecha o izquierda de las ruedas del automóvil.

Por último, el Axes Jump le corresponde la tecla space el cual permite el frenado del automóvil.

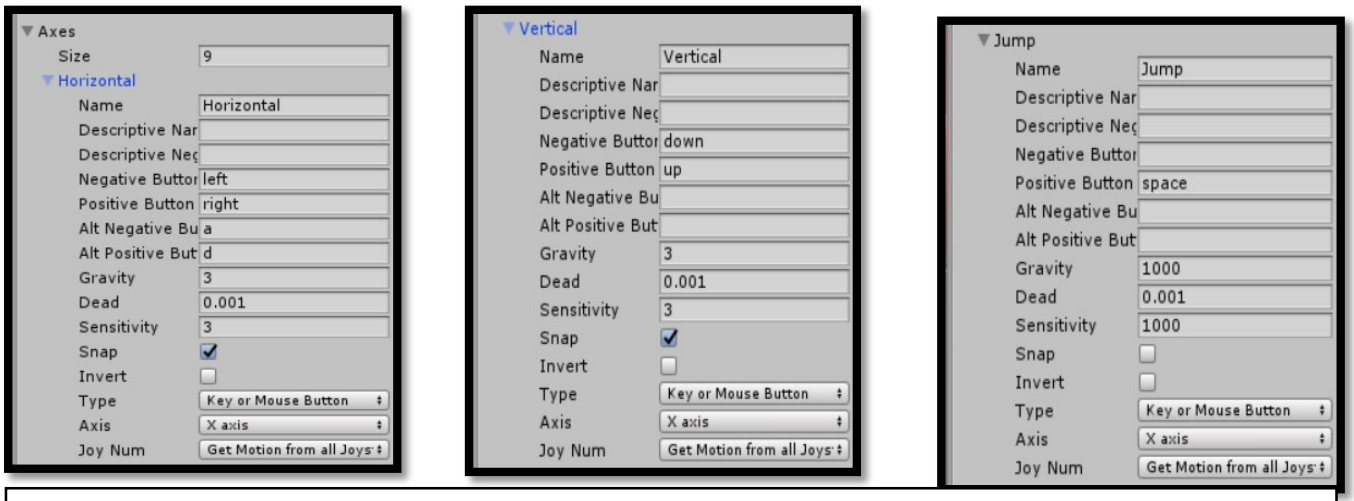


Imagen 5.97 Configuración de los Axes para él teclado.

5.4.2. Configuración de Inputs (Control PlayStation 4)

De la misma forma que se configuro el teclado, el control de PlayStation 4 sigue la misma dinámica, solo que tiene pequeñas modificaciones en cuanto la asignación de los botones.

Para esas modificaciones en la figura 5.98 se muestra el control de PlayStation 4 con su respectivo nombre de los botones.

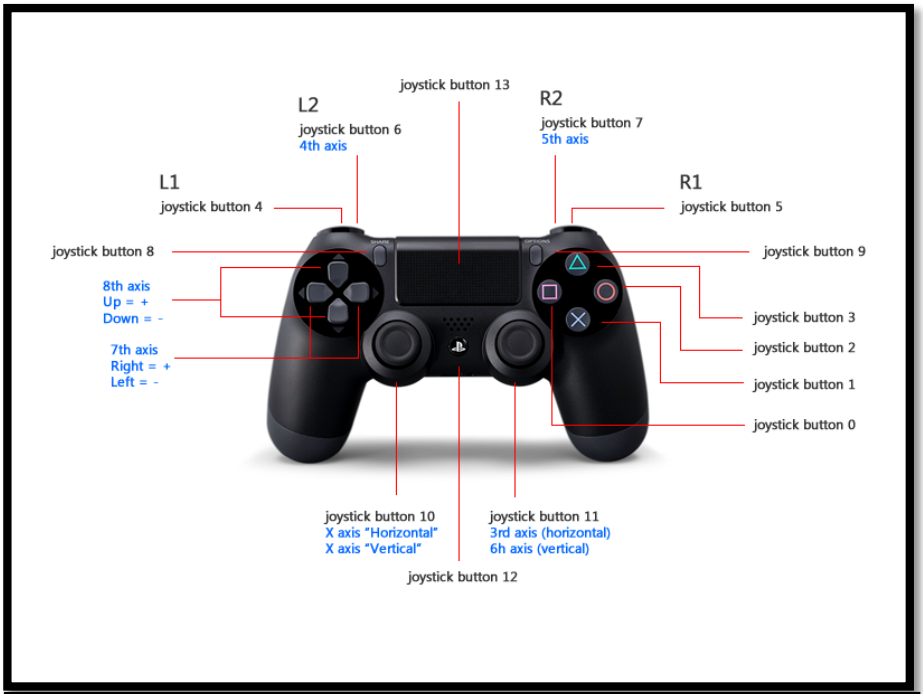


Imagen 5.98 Control de PlayStation 4 con la configuración de los botones.

Para el Axes Vertical se le asignan los joysticks button 4 y 5 ya que estos tendrán la función de acelerar o dar reversa al automóvil. Este joystick es equivalente a las teclas up y down dentro del teclado

Para el Axes Horizontal se le asigna el joystick button 10 ya que tendrá la función de realizar el giro de las ruedas del automóvil. Este joystick es equivalente a las teclas left y right del teclado.

Para el Axes Jump se le asigna el joystick button 7 ya que tendrá la función de frenado del automóvil. Este joystick es equivalente a la tecla espace del teclado.

En la imagen 5.99 se muestra con mayor detalle la configuración de los Axes del control de PlayStation 4.

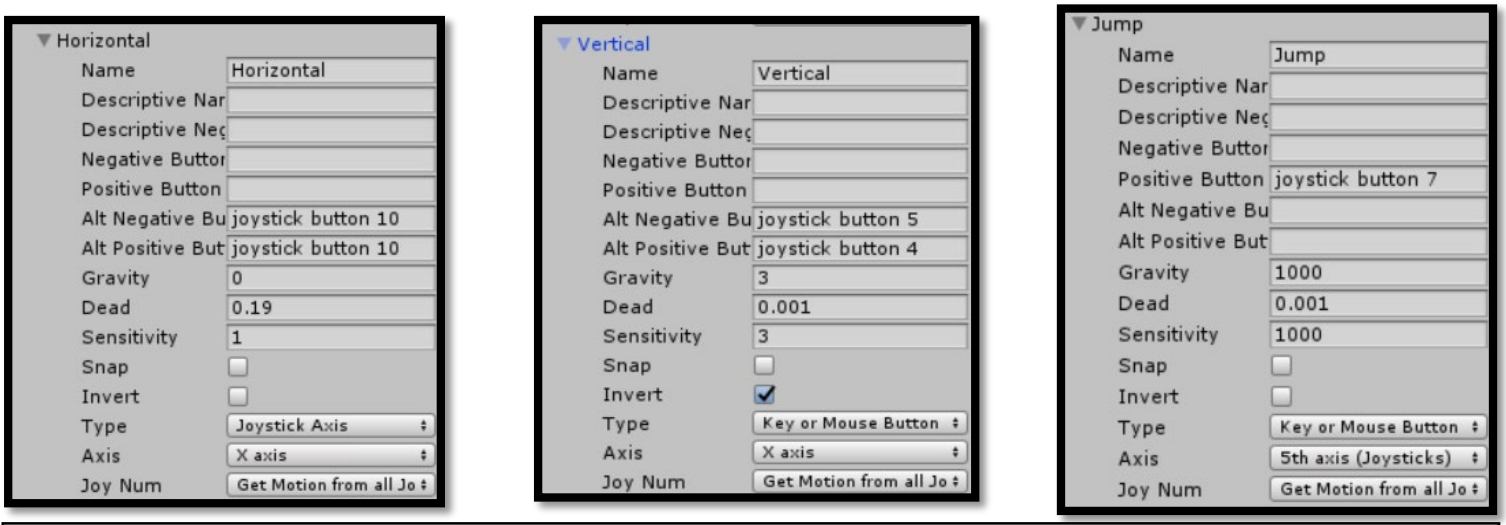


Imagen 5.99 Configuración de los Axes para el control PlayStation4.

5.5. Implementación de Realidad Virtual (Oculus Rift)

Uno de los objetivos principales del simulador es que el usuario tenga la sensación de que está dentro del automóvil, por lo cual el uso de la realidad virtual es indispensable para ello implementar el software de Oculus Rift.

5.5.1. Instalación del Software

Dentro de la página oficial de Oculus Rift se puede descargar el driver necesario el cual permitirá configurar los aditamentos que se necesitan para la implementación de la Realidad Virtual. En la imagen 5.100 se muestra el software que se necesita descargar.

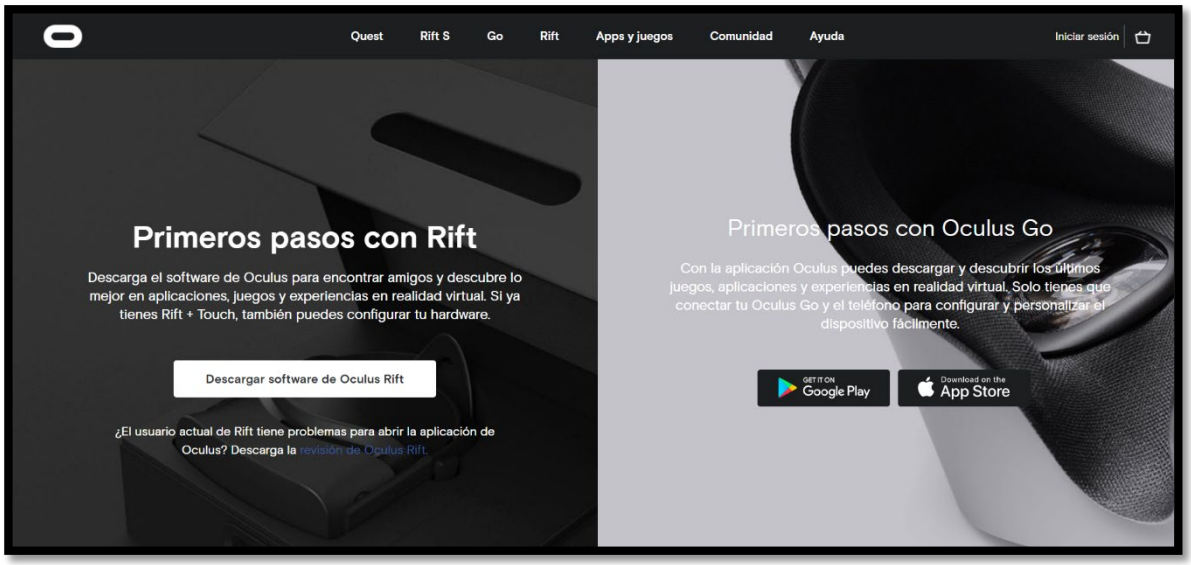


Imagen 5.100 Pagina de descarga del software de Oculus.

5.5.2. Configuración de Oculus Rift

Una vez que el software se instala en el equipo, se comienza el proceso de configuración del hardware. Se muestra de forma automática el asistente en Oculus Home el cual permite configurar los mandos y el área de juego, así como emparejar los controladores Touch con el Rift. En la imagen 5.101 se muestra la pantalla principal de Oculus Home.

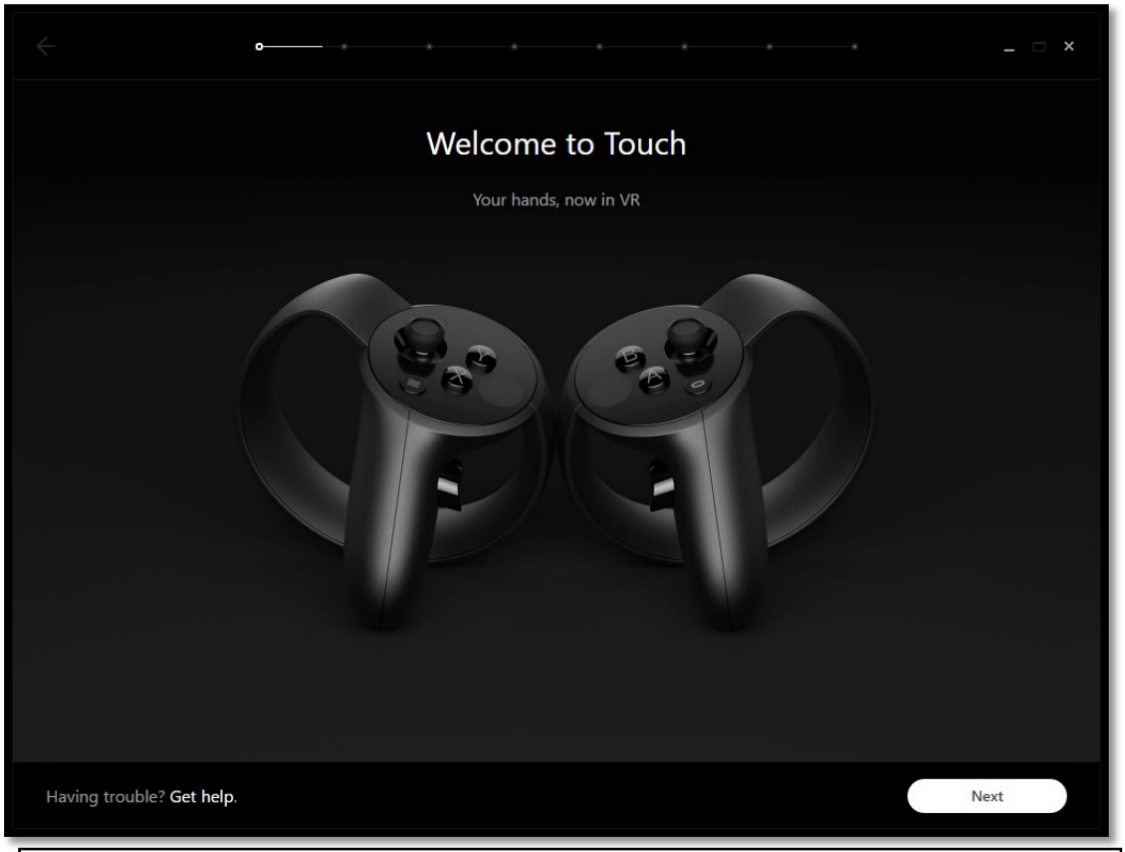


Imagen 5.101 Pantalla de inicio de Oculus Home.

El emparejamiento se realiza individualmente por cada uno de los mandos. El asistente indicará que se presione el botón de Home o System junto con los demás botones de acción. Esto se realiza hasta que comience a parpadear una luz. En la imagen 5.102 se muestra la pantalla de configuración de los controles de Rift

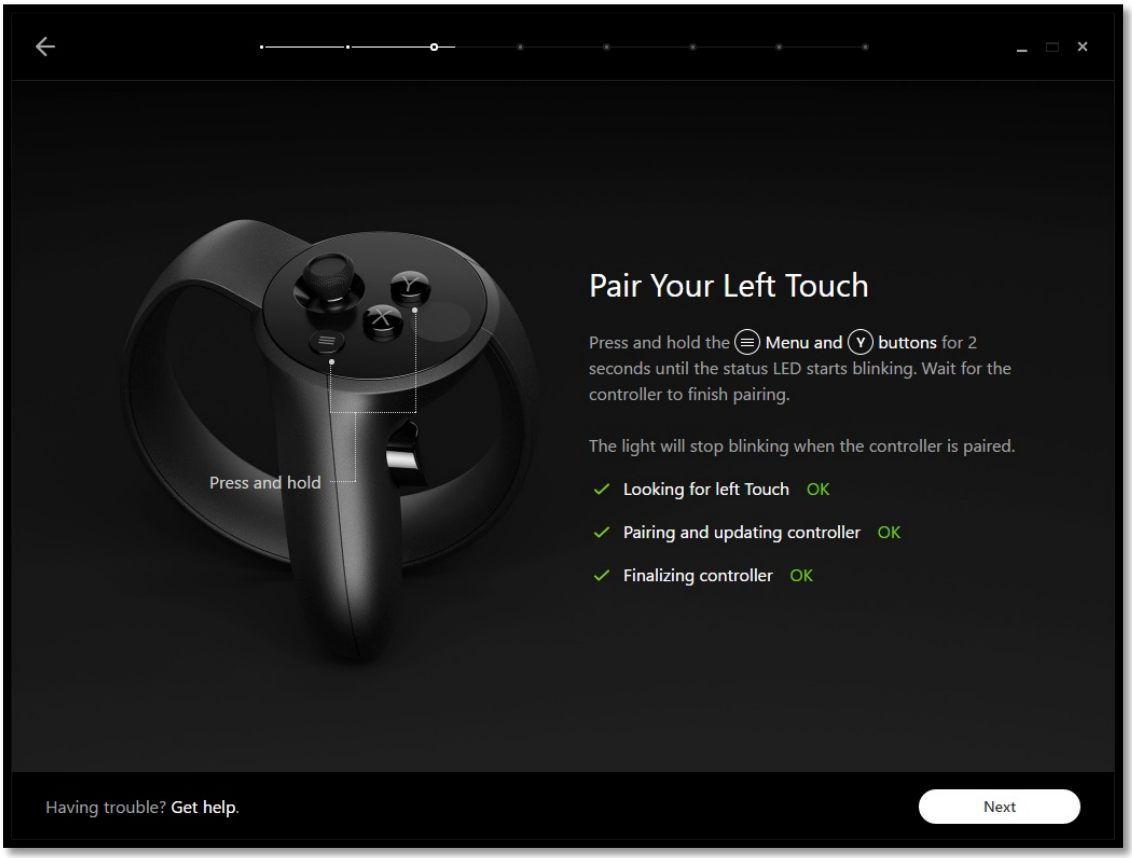


Imagen 5.102 Configuración de controles de Rift.

Después continúa el proceso de configuración de las cámaras las cuales se recomienda para el seguimiento frontal en 180°. Oculus recomienda situar ambas cámaras a la altura de los ojos y a la misma distancia del centro de la zona de juego, esta separación deberá ser en torno a 2 metros aproximadamente. En la imagen 5.103 se muestra como el asistente da las indicaciones para dicha configuración de la cámara.

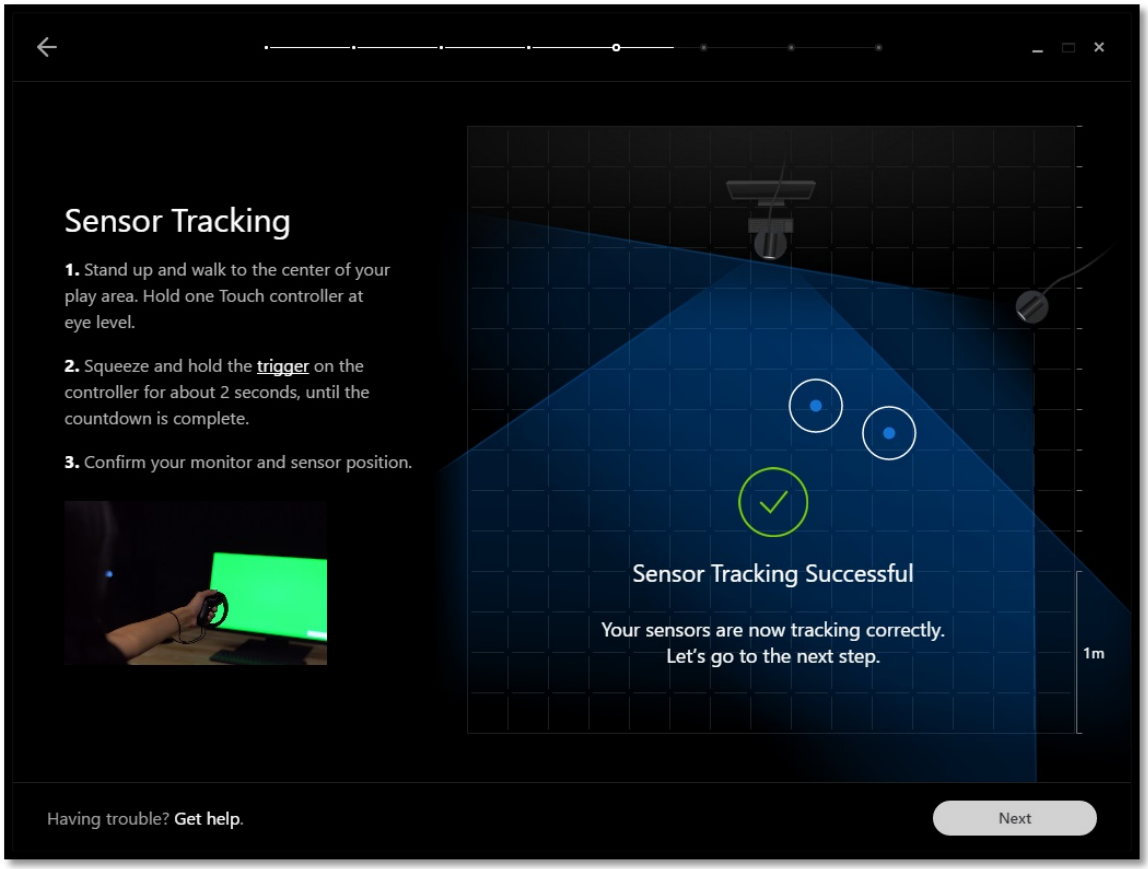


Imagen 5.103 Configuración de la cámara de Rift.

Se le indica al sistema la posición del sujeto con respecto a la cámara, para este proceso se sostiene un controlador Touch mientras se realiza un movimiento por el área de juego. En la imagen 5.104 se muestra el proceso de configuración del sistema de posición.

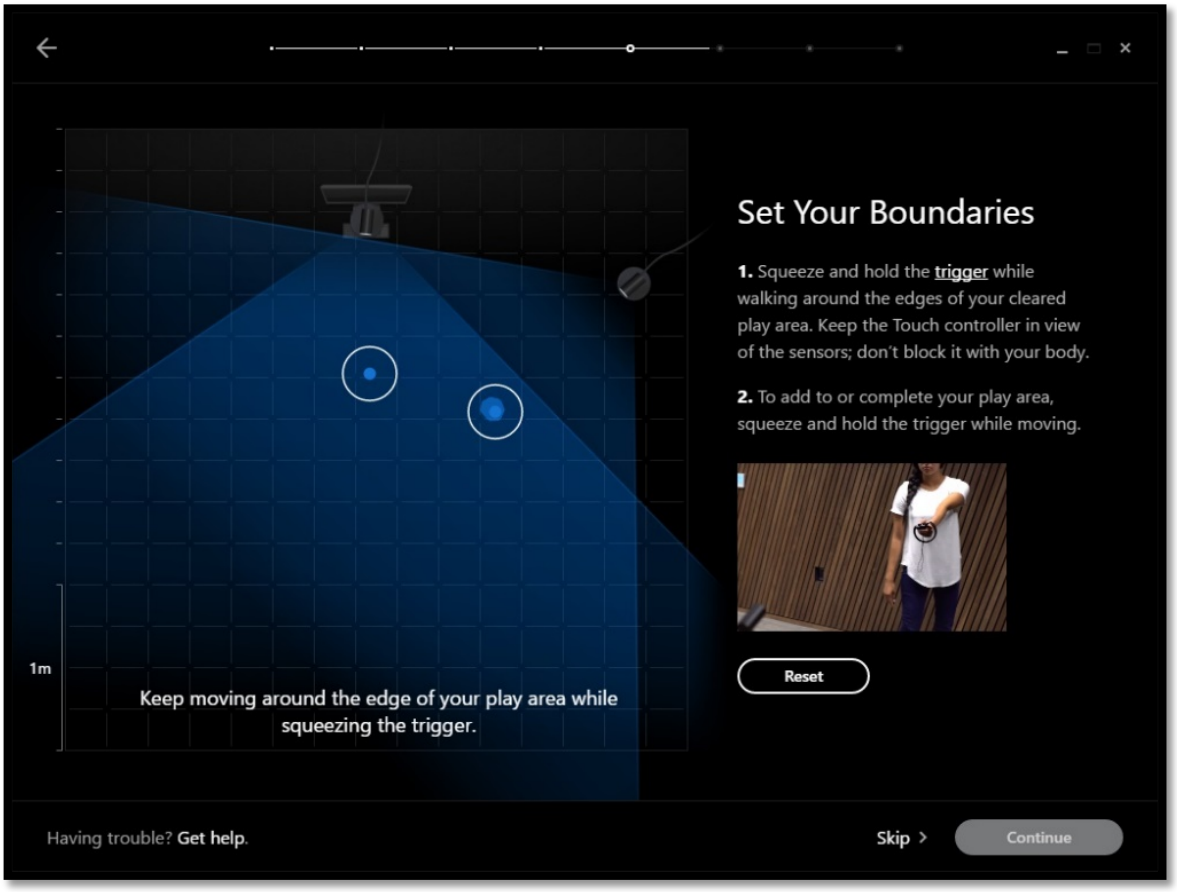


Imagen 5.104 Configuración del sistema de posición.

Por último, comienza el proceso de configuración del casco en el cual se tendrán que seguir un cierto número de instrucciones para poder tener imagen en la pantalla. Para comprobar que la configuración está completa, se recomienda utilizar el programa First Contact, el cual es un programa donde un robot da un pequeño tutorial sobre el uso de los controles. En la imagen 5.105 se muestra un ejemplo de como se ve el programa First Contac cuando se termina la configuración de Rift.

Imagen 5.105 Ejecución de First Contact.

5.5.3. Configuración de Oculus Rift en Unity

Dentro del editor de Unity existe un apartado para la configuración de realidad virtual. En la pestaña del editor en la parte de file, se encuentra una opción llamada Build Settings tal como se observa en la imagen 5.106.

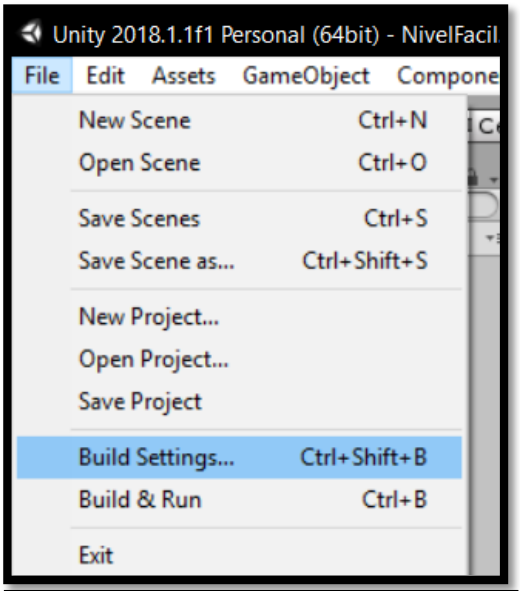


Imagen 5.106 Build Settings.

Cuando se selecciona esta opción, se desplegará una venta tal como se muestra en la imagen 5.107 donde se muestran las escenas que se han utilizado en el proyecto, además de la plataforma en que se desea desarrollar el proyecto. Se selecciona la opción de Player Settings la cual mostrará una ventana adicional en el inspector tal como se muestra en la imagen 5.108.

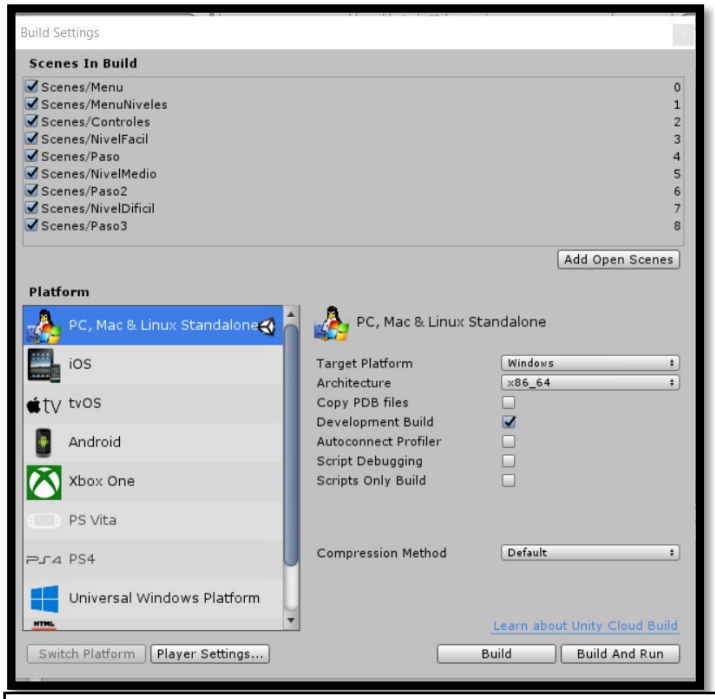


Imagen 5.107 Ventana de configuración de Build Settings.

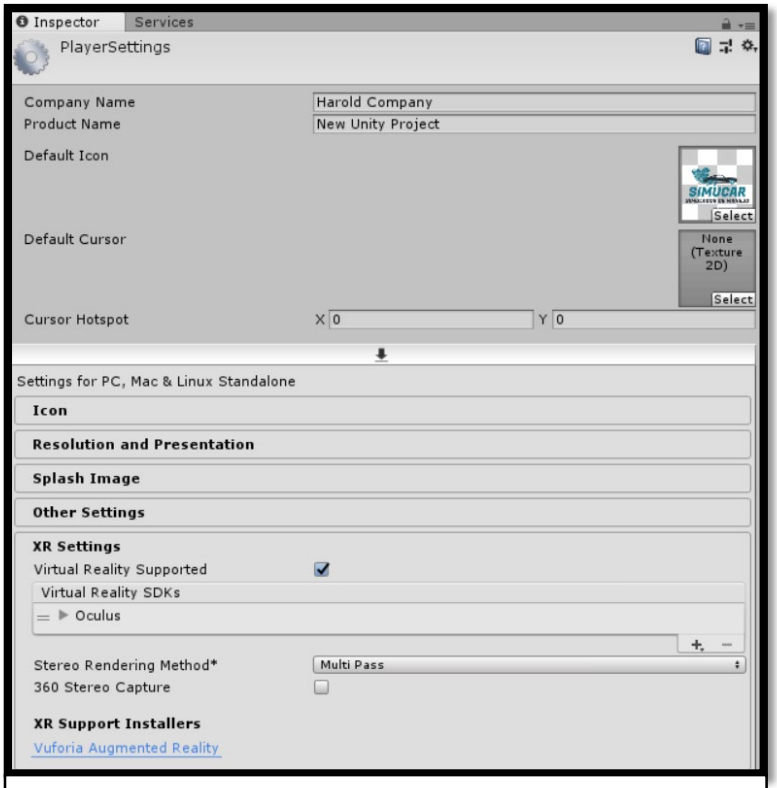


Imagen 5.108 Configuración de Player Settings.

Dentro de Player Settings se encuentra una opción llamada XR Settings la cual contendrá la opción de habilitar Oculus Rift. Basta con habilitar esta opción y tener el equipo conectado a la computadora y automáticamente la Realidad Virtual se activará al iniciar la aplicación.

5.6. Implementación del Control de Manejo (Logitech G27)

El simulador no solo podrá ser controlado por el control de PS4, sino que también se tendrá la opción de controlar el automóvil mediante un volante y pedales los cuales permitan una mayor experiencia al usuario al momento de usar el simulador.

5.6.1. Asset Logitech Gaming SDK

Es importante considerar que para que el volante Logitech G27 funcione correctamente, se necesitará de un Asset especial el cual toma en cuenta las funcionalidades del mismo control y cualquier otro dispositivo de la marca Logitech. En la imagen 5.109 se muestra el Asset a descargar.

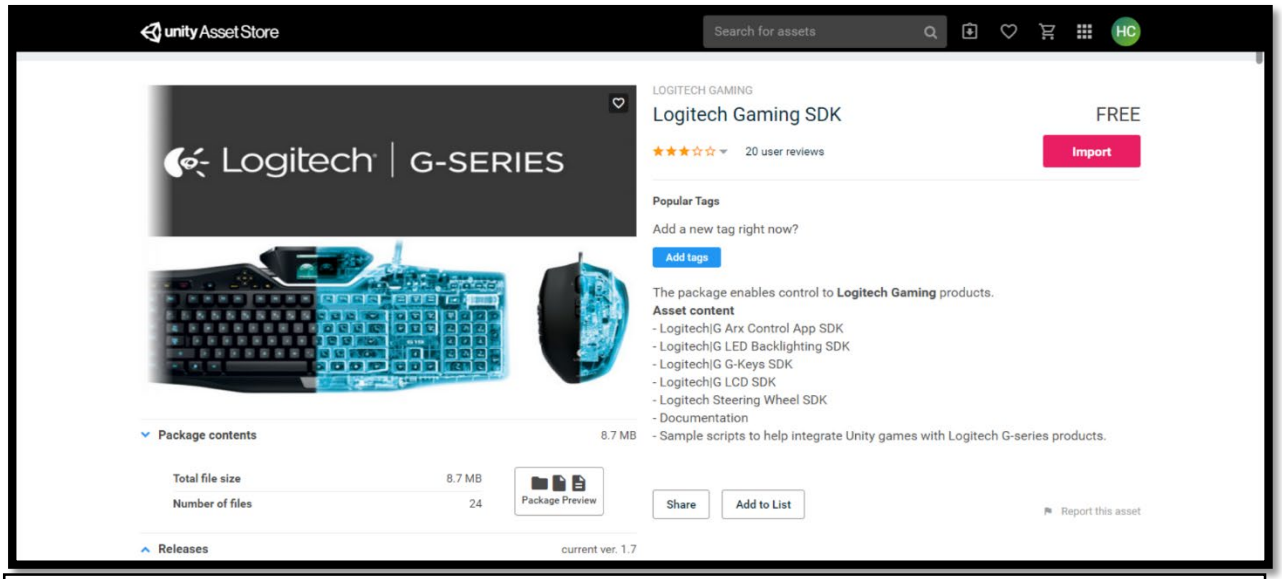


Imagen 5.109 Descarga del Asset de Logitech SDK.

Una vez que el Asset es descargado, se presentará la siguiente carpeta tal como se muestra en la imagen 5.110 la cual contiene otras subcarpetas y elementos adicionales de las cuales se destacan las siguientes:

Imagen 5.110 Elementos del Asset de Logitech SDK.

- **AppleData:** Esta carpeta contiene elementos que se utilizan para aplicaciones que se encuentren en páginas de html, además de incluir dos imágenes decorativas tal como se muestran en la imagen 5.111.

Imagen 5.111 Elementos de AppletData.

- **Script Sample:** Carpeta con los elementos más importantes del Asset ya que en ella se encuentran todos los scripts necesarios para implementar el control Logitech G27 y otros elementos importantes. En la imagen 5.112 se muestra los scripts que contiene dicha carpeta.

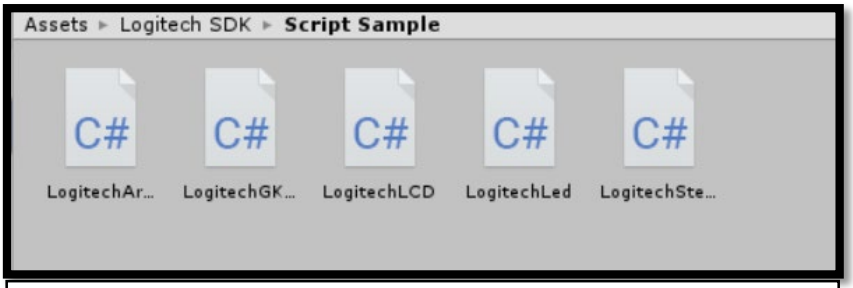


Imagen 5.112 Scripts de Logitech SDK.

- **SDK Documentation:** En esta carpeta se encuentra toda la documentación acerca de cómo utilizar los scripts y otros elementos que contenga el Asset de Logitech G27. En la imagen 5.113 se logra apreciar toda la documentación referente a Logitech SDK.

Imagen 5.113 Scripts de Logitech SDK.

5.7. Ejecutable del Simulador.

Una vez finalizado el proceso de construcción de los niveles, escenas y demás elementos que son parte del simulador, se genera el ejecutable el cual permitirá entrar directamente en la aplicación sin la necesidad de abrir Unity, recordando que este es solo un editor y no la versión final.

Para crear dicho ejecutable en la sección de Build Settings aparecerá un botón con el nombre Build. Cómo es una aplicación para computadora, no

importando el sistema operativo (Windows, Mac o Linux) se selecciona dicha plataforma para crear el ejecutable tal como se muestra en la imagen 117.

Es importante mencionar que antes de comenzar con el proceso de construcción se deben de colocar las escenas creadas en el editor tal como se muestra en la imagen 5.114.

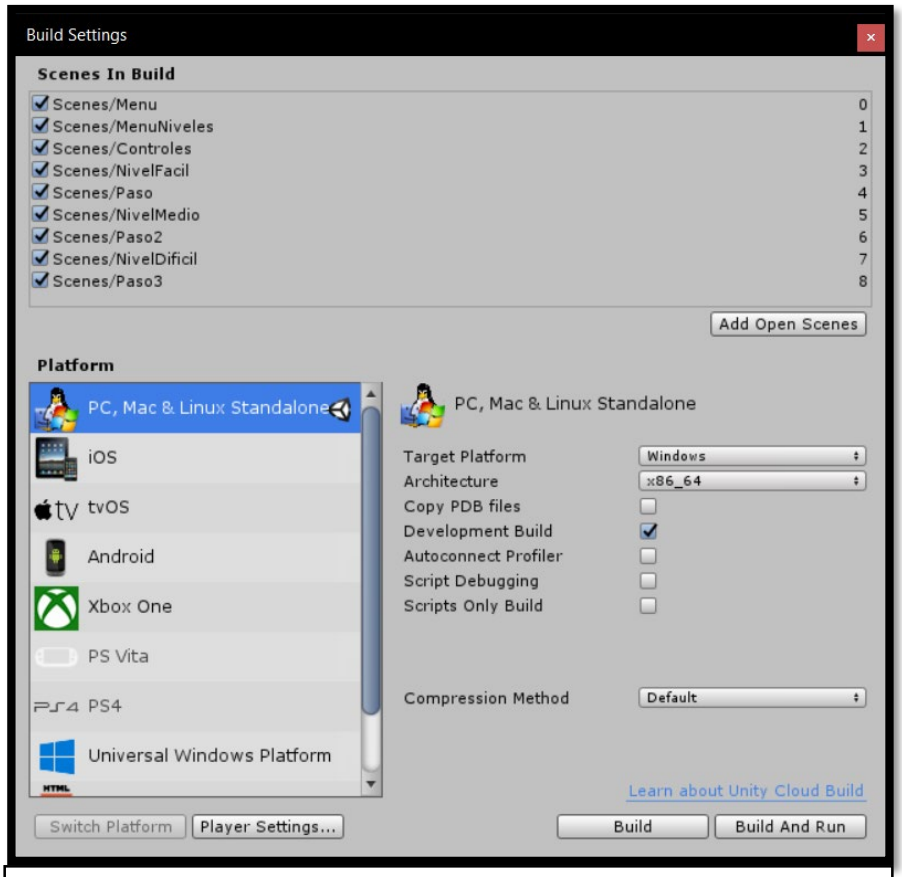


Imagen 5.114 Creación del ejecutable con las escenas creadas en el editor.

Una vez que se tienen los elementos listos para la construcción del ejecutable, Unity requerirá una carpeta en donde se guarde dicho ejecutable tal como se muestra en la imagen 5.115.

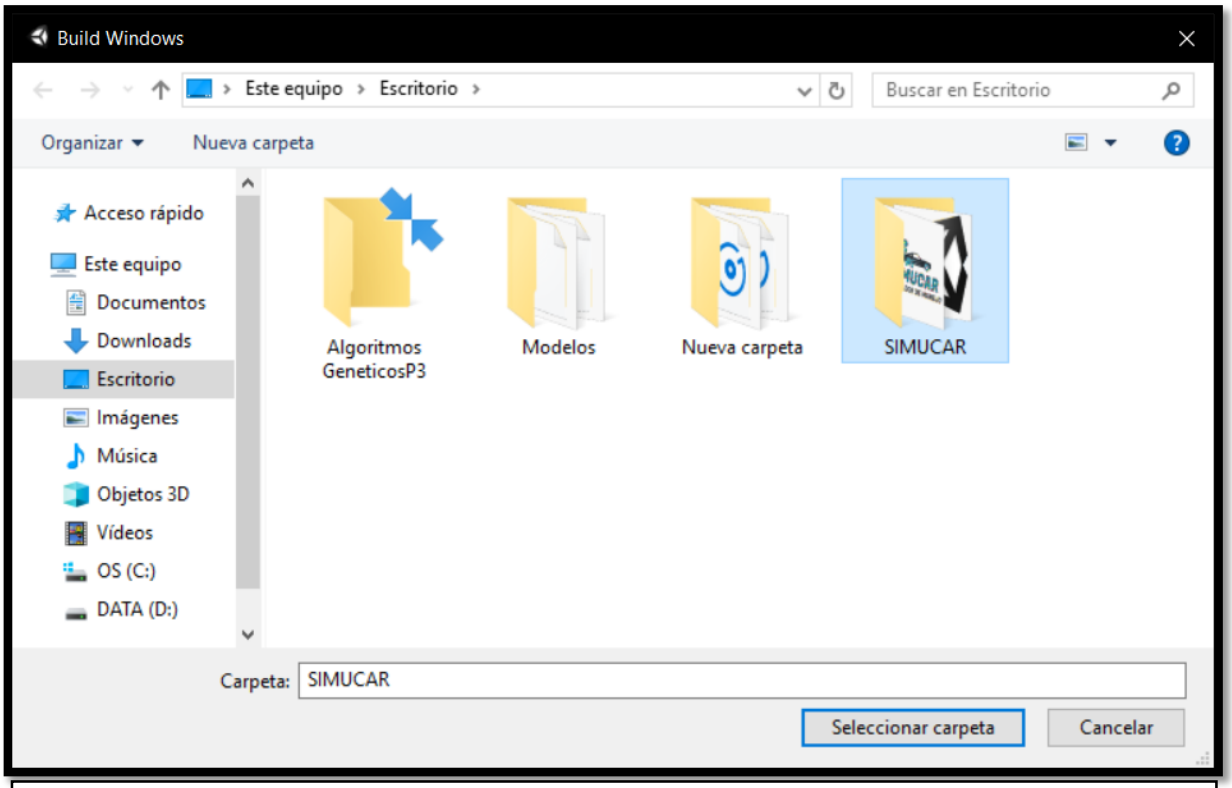


Imagen 5.115. Selección de la ruta donde se guardará el ejecutable.

Cuando el proceso de creación de ejecutable termine, se mostrará los elementos necesarios para que dicho ejecutable funcione además del archivo .exe que es con el cual se iniciará la aplicación. Estos elementos se pueden apreciar en la imagen 5.116.

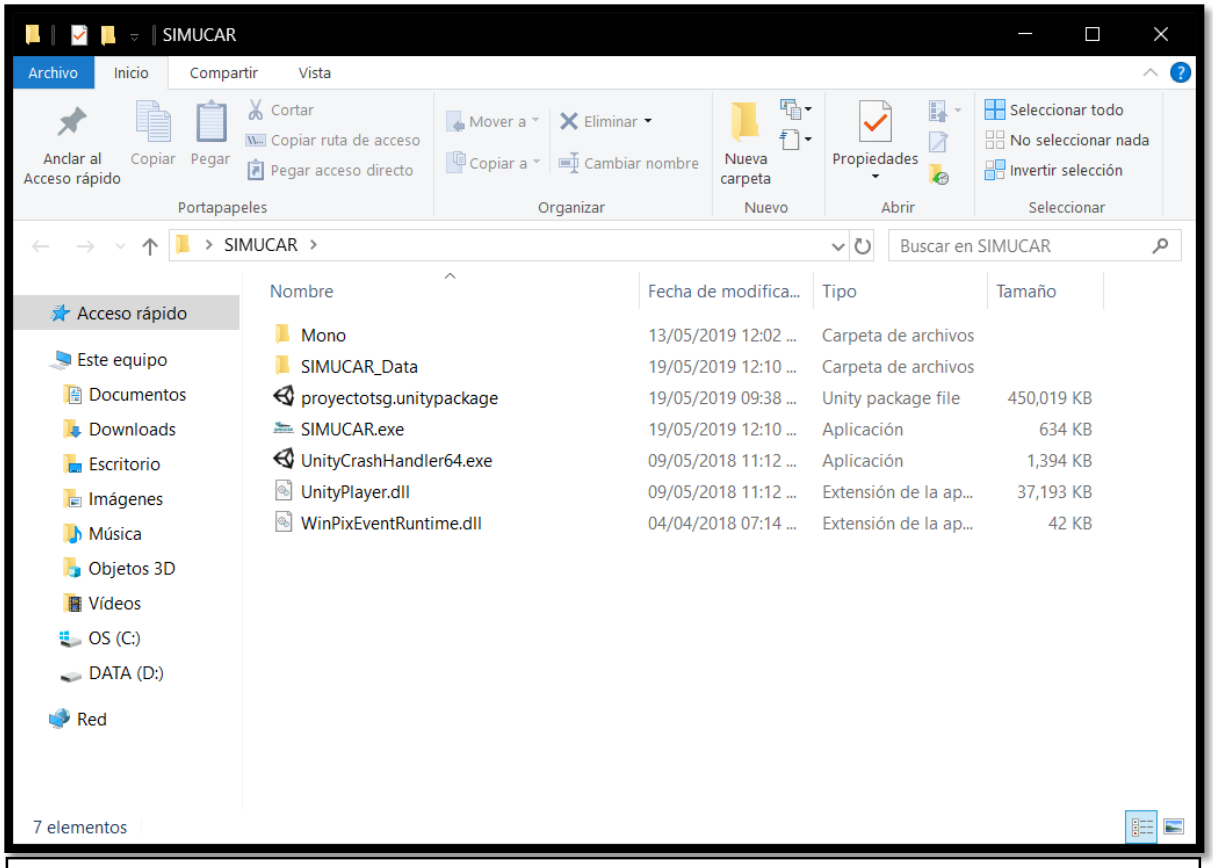


Imagen 5.116. Elementos del ejecutable junto con él .exe

CAPÍTULO VI
PRUEBAS

SIMUCAR
SIMULADOR DE MANEJO

6.1. Muestra de la Interfaz

Durante las pruebas del simulador, los diferentes usuarios conocieron el prototipo en el cual se muestra la interfaz de inicio del simulador como se aprecia en la imagen 6.1.

Imagen 6.1 Interfaz inicial del simulador

En esta interfaz los usuarios podían seleccionar la opción de niveles o salir del simulador, ya que la opción de tutoriales aún no estaba habilitada.

6.1.1. Pruebas de los Niveles.

Al seleccionar la opción de niveles, el simulador arroja una ventana en donde los usuarios pueden seleccionar entre las 3 dificultades nivel, medio, difícil, tal como se muestra en la imagen 6.2.

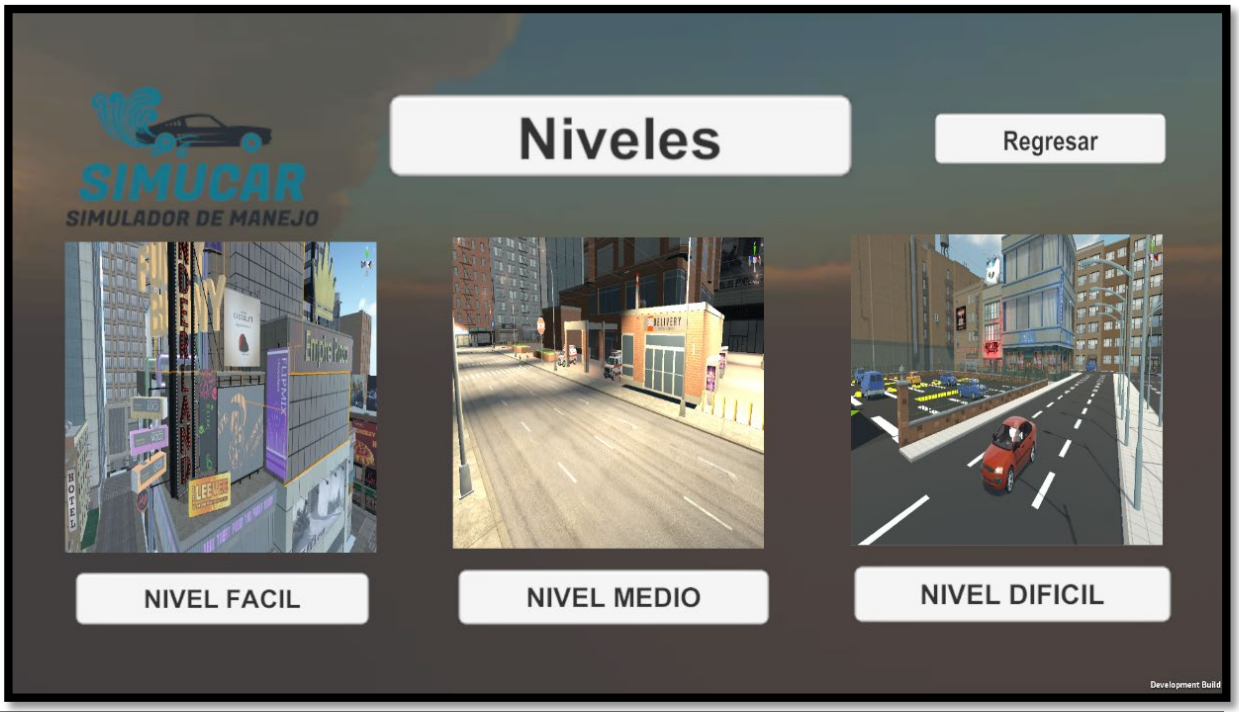


Imagen 6.2 Menú de los niveles

El nivel seleccionado por los usuarios fue el nivel medio. Durante este nivel se logró observar cómo los usuarios demostraban su habilidad de conducción durante el transcurso del recorrido tratando de esquivar los obstáculos que se les presentaban.

Antes de comenzar la prueba o el inicio del nivel, el simulador arroja un pequeño cuadro en el cual indica las instrucciones que deben de seguir los usuarios para completar el nivel. En la imagen 6.3 se muestran dichas instrucciones.

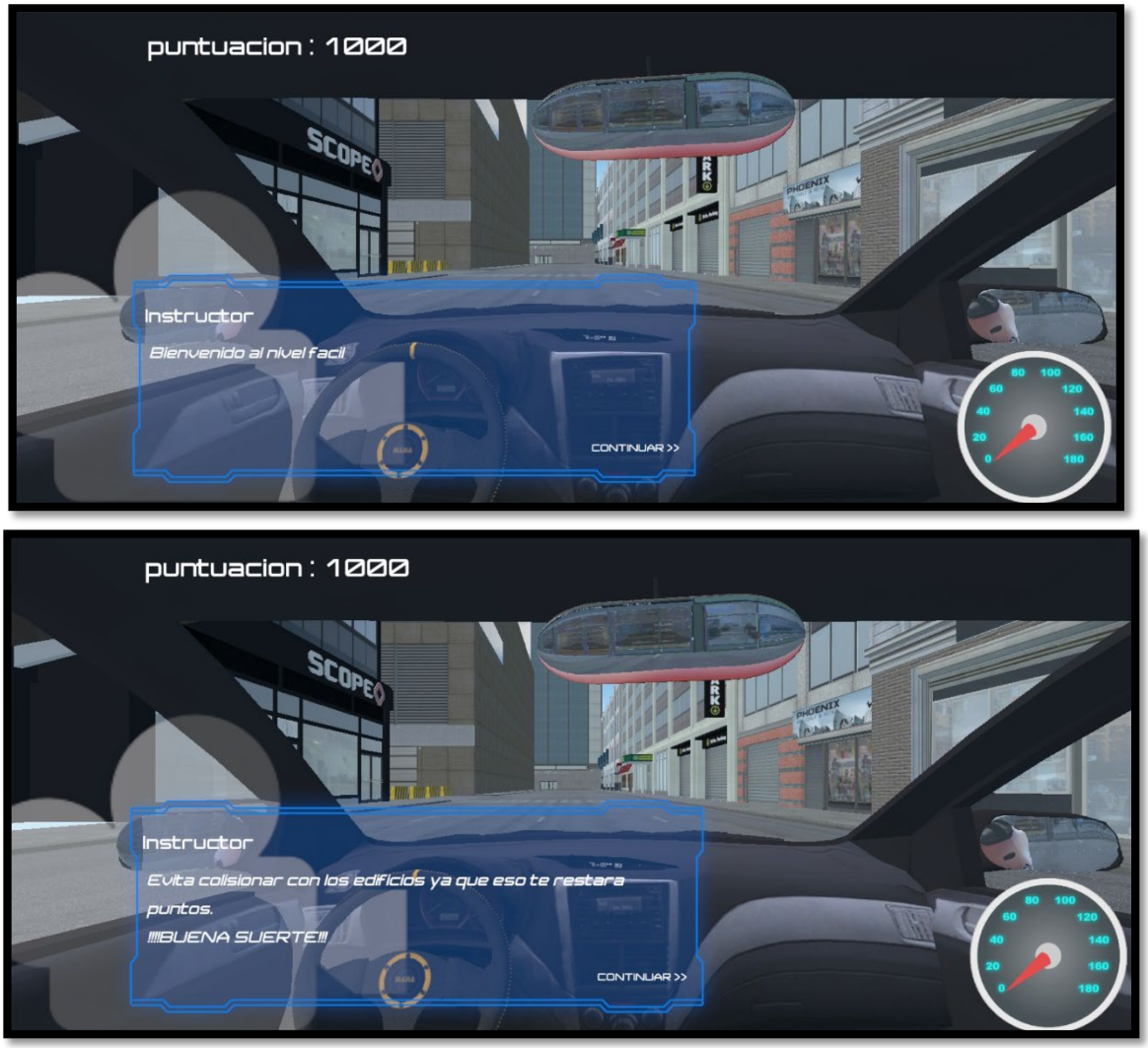


Imagen 6.3 Instrucciones previas a las pruebas.

Al llegar a su objetivo del nivel, el simulador mostraba una pequeña pantalla donde mencionaba que el nivel estaba completado, además de dar opciones al usuario de continuar al nivel siguiente o salir al menú principal. En la imagen 6.4 se muestra la ventana de nivel completado.

Imagen 6.4 Pantalla de nivel superado.

.

Para la parte de pruebas se solicitó la ayuda de 10 alumnos voluntarios los cuales realizaron las pruebas del simulador. En la imagen 6.5 se observan a dichos alumnos probando el simulador.

Imagen 6.5 Usuarios probando el simulador.

6.2. Pruebas Utilizando Realidad Virtual.

En este tipo de prueba se utilizó la Realidad Virtual para verificar el correcto funcionamiento del simulador. Este tipo de prueba consiste en que el usuario se coloque las gafas de Oculus Rift para que visualice el simulador desde las mismas gafas. En la imagen 6.6 se muestra a los usuarios configurando los dispositivos de Oculus Rift mientras que en la imagen 6.4 se observa ya la implementación de la Realidad Virtual dentro del simulador.

Imagen 6.6. Calibración de Oculus Rift

Imagen 6.7. Realidad Virtual dentro del simulador

6.3. Detalles del Simulador.

Durante las pruebas, los usuarios mencionaron que existen ciertos detalles dentro del simulador los cuales son los siguientes:

- **Sensibilidad del control:** Algunos de los usuarios mencionan que, durante el manejo del automóvil, sentían la sensibilidad muy baja al momento de girar.

- **Derrapes:** Durante la prueba del nivel fácil, los usuarios se percataron que, en algunas secciones del nivel, el automóvil se derrapaba de forma innecesaria y perdían el control del mismo automóvil.

- **Velocímetro:** El velocímetro tenía ciertos problemas ya que al frenar el indicador no se detenía y marcaba velocidades por encima de los 120 KM.

- **Colisiones de los edificios:** Al momento de que el automóvil intentaba chocar con algunos edificios del nivel medio, este los atravesaba.

- **Mejor control del freno:** Durante las pruebas, dos de los usuarios mencionaron que el frenado el automóvil era muy seco, por lo que a veces dificultaba mantener el control durante los trayectos.

- **Visibilidad:** Algunos de los usuarios tuvieron problemas en cuanto a la visibilidad al momento de conducir el automóvil ya que mencionaban que no podían ver muy bien las líneas de apoyo de la calle o si ya habían chocado con algún elemento del escenario.

- **Sonidos:** Al momento de tener el control del automóvil, los usuarios mencionaron que sería óptimo agregar el sonido de ambientación del automóvil al momento de acelerar o frenar.

CAPÍTULO VII
RESULTADOS

7.1. Datos Específicos.

Los resultados esperados durante las pruebas del simulador fueron exitosos ya que uno de los objetivos principales del proyecto es crear un sistema de evaluación, con el cual se tuviera un criterio adicional para que los usuarios tengan acceso a la licencia de conducir.

7.1.1. Evaluación de los Usuarios.

Para corroborar el objetivo del punto anterior, se requirió la ayuda de 10 usuarios los cuales fueron los responsables de probar el simulador y ser evaluados. En la tabla 7.1 se muestran los datos de los usuarios, así como la puntuación que obtuvieron por cada nivel.

#	Nombre	Edad	Puntuación Nivel Fácil	Puntuación Nivel Medio	Puntuación Nivel Difícil
1	Usuario 1	24	600	900	500
2	Usuario 2	24	1000	500	400
3	Usuario 3	25	200	200	600
4	Usuario 4	24	800	900	700
5	Usuario 5	21	600	500	400
6	Usuario 6	24	200	0	900
7	Usuario 7	23	600	500	400
8	Usuario 8	23	200	900	200
9	Usuario 9	24	600	800	800
10	Usuario 10	25	1000	900	900

Tabla 7.1 Datos generales de los usuarios y puntaje obtenido por nivel.

7.1.2. Métricas de Evaluación.

Las métricas que se utilizaron para verificar si el usuario debe ser acreedor o no de la licencia de conducir se muestran a continuación en la tabla 7.2.

Aprobado	Reprobado
1000	500
900	400
800	300
700	200
600	100
	0

Tabla 7.2 Métricas de evaluación.

7.1.3. Perdida de la Puntuación

De acuerdo a las métricas de evaluación se realizaron consideraciones en cuanto al manejo de puntos. Se estableció la pérdida de puntos de acuerdo a los siguientes criterios:

- Colisión con otros automóviles: -300 puntos
- Colisión con objetos dentro del escenario (botes de basura, señalizaciones, edificios): -200 puntos
- Colisión contra personas: -1000 puntos (Reprobación automática)
- Ir en sentido contrario: -200 puntos
- No respetar los semáforos: -400 puntos

7.1.4. Resultados Generales de los Usuarios.

Tomando en cuenta la información de las tablas anteriores, se realizó el cálculo correspondiente para verificar si los usuarios aprobaron o reprobaron dicha prueba. El cálculo de los resultados se llevó a cabo de la siguiente manera:

$$\boldsymbol{resultado} = \frac{\boldsymbol{NF} + \boldsymbol{NM} + \boldsymbol{ND}}{\boldsymbol{3}}$$

Donde:

- NF = puntaje obtenido en el nivel fácil
- NM = puntaje obtenido en el nivel medio
- ND = puntaje obtenido en el nivel difícil

Por lo tanto, aplicando la fórmula anterior a los resultados que obtuvo cada uno de los usuarios, se obtiene la información si aprobaron o no la prueba del simulador tal como se muestra en la tabla 7.3.

#	Nombre	Puntaje Promedio	Estado
1	Usuario 1	666.66	Aprobado
2	Usuario 2	633.33	Aprobado
3	Usuario 3	333.33	Reprobado
4	Usuario 4	800	Aprobado
5	Usuario 5	500	Reprobado
6	Usuario 6	366.66	Reprobado
7	Usuario 7	500	Reprobado
8	Usuario 8	433.33	Reprobado
9	Usuario 9	733.33	Aprobado

10	Usuario 10	933.33	Aprobado

Tabla 7.3 Resultados Finales

Como se observa en la tabla anterior, de los 10 participantes que realizaron la prueba del simulador, la mitad reprobó mientras que la otra mitad lograron acreditar la evaluación del simulador.

Esto quiere decir que el simulador cumple con el objetivo propuesto de poner a prueba a los usuarios mediante los niveles que se les presentan y así tratar de buscar un puntaje adecuado para aprobar y obtener la licencia.

7.2. Observaciones Generales.

Durante los procesos de pruebas del simulador se logró observar algunos comportamientos por parte de los usuarios los cuales permitieron la constante mejora del prototipo del simulador.

Estos comportamientos que mostraban los usuarios iban desde comentarios positivos acerca de que el simulador es una buena medida para evaluar este tipo de situaciones ya que no es una forma arcaica de realizar una evaluación, sino más bien, se presenta como una forma más dinámica de realizar los procesos de evaluación.

CAPÍTULO VIII
CONCLUSIONES Y RECOMENDACIONES

8.1. Conclusiones.

A través de la elaboración del presente se ha logrado verificar la importancia de implementar un sistema de evaluación para la obtención de las licencias de conducción ya que como se observó en el capítulo VII, la prueba realizada a los diferentes usuarios permitió verificar si tenían las aptitudes y conocimientos básicos para obtener la licencia a través de la prueba del simulador.

Con esto se logró cumplir uno de los objetivos propuestos para este trabajo, ya que se desarrolló un prototipo de simulador de manejo cuya finalidad es evaluar al usuario mediante distintas pruebas y así obtener la información necesaria para que dicho usuario acreditara o no la prueba y se lograra obtener la licencia.

El uso del software y hardware para el desarrollo del prototipo fue de mucha utilidad ya que facilitó en gran parte a que los usuarios lograran estar inmersos en un ambiente gráfico especializado a la conducción y que además observaran una manera diferente de ser evaluados.

Con este prototipo se sientan las bases para futuras modificaciones que se pueden llegaran a realizar dependiendo de las necesidades del usuario como del cliente que requiera de este tipo de software especializado.
Además, cabe destacar que dicho prototipo no solo funciona para una evaluación sencilla, sino que además puede ser utilizado para diferentes escuelas de manejo que se encuentren dentro del caso de estudio el cual es la CDMX.

Es evidente que, al ser un prototipo, los errores estuvieron presentes durante la fase pruebas, por tal razón se realizó un constante mantenimiento cada que un usuario realizaba la prueba. Los comentarios y observaciones fueron

de vital importancia ya que, con ello, permitió mejorar ciertos aspectos del simulador y así lograr cumplir el objetivo principal del proyecto.

8.1. Recomendaciones

Como se mencionó al final del punto anterior del capítulo, las fallas que se presentaron durante el desarrollo y fase de pruebas fueron importantes ya que con ellas permitió un constante mantenimiento del simulador logrando obtener un prototipo en condiciones capaz de cumplir con el objetivo del proyecto y las expectativas de los usuarios.

Las recomendaciones más solicitadas por parte de los usuarios fueron detalles tanto en la parte de mejorar la interfaz gráfica, más interacción, niveles más complicados entre otros.

Sin embargo, es importante mencionar que este proyecto no solo puede enfocarse específicamente al objetivo planteado; este puede ser modificado de tal manera que pueda ser utilizado para fines lucrativos, educativos o de negocio.

CAPÍTULO IX
LIMITACIONES DE LA INVESTIGACIÓN

SIMUCAR
SIMULADOR DE MANEJO

9.1. Limitaciones.

Debido al poco tiempo que se tenía para realizar la planeación, implementación y ejecución del prototipo, este se vio limitado a distintas situaciones las cuales se tomaron en cuenta para un trabajo a futuro. Los elementos limitados se describen a continuación:

9.1.1. Construcción de Niveles

Al solo tener 6 meses de trabajo, solo se logró construir 3 niveles (fácil, intermedio y difícil). En la planeación original se planeó construir entre 7 u 8 niveles en donde se pusiera a prueba al usuario en distintos escenarios (el siguiente más complicado que el anterior) y de esa forma mostrar diferentes perspectivas de cómo evaluar al usuario.

9.1.2. Pruebas Utilizando un Volante y Pedales.

El proyecto se logró configurar tal como se propuso, con un control de ps4 y además con un volante y pedales. Al no contar con un volante propio se tuvo que recurrir a un volante ajeno el cual no es de uso público y eso limitó realizar pruebas con dicho control dejando abierta la posibilidad de corregir errores o escuchar las recomendaciones de los usuarios.

9.1.3. Pruebas con Usuarios que no Fueran Alumnos

En este caso solo se realizaron pruebas con alumnos los cuales deja diferentes puntos de vista ya que, al ser un ambiente escolar, no genera muchos resultados que se necesitan para comprobar el funcionamiento correcto del simulador. De igual manera el problema que se presentó fue el préstamo del equipo (Oculus Rift) el cual era la parte más importante del proyecto y sin él las pruebas no se podían realizar y esto

genera que sea muy complicado transportar el proyecto fuera de la zona escolar para que otros usuarios lograran probar el simulador.

CAPITULO X

FUENTES Y REFERENCIAS BIBLIOGRÁFICAS

Capítulo X
Fuentes y Referencias Bibliográficas

1. Aggarwal, Anshul, (s/a), “Introduction to Visual Studio” información recuperada la siguiente url: https://www.geeksforgeeks.org/introduction-to-visual-studio/

2. s/n, (s/a), “Documentation for Visual Studio Products”, información recuperada de la página oficial de Microsoft en la siguiente url: https://docs.microsoft.com/en-us/visualstudio/products/?view=vs-2019#pivot=products&panel=products1

3. Landa Cosio, Nicolas Arrioja, (2013), “Unity, Diseño y Programación de Videojuegos”, Buenos Aires, Argentina, Users, pp:320.

4. Okita, Alex, (2014), “Learning C# Programming with Unity 3D”, United States, CRC Press, pp:661

5. s/n, (s/a), “Codificación en C# en Unity para principiantes” , información recuperada de la página oficial de Unity 3D en la siguiente url: https://unity3d.com/es/learning-c-sharp-in-unity-for-beginners

6. Cushman, Dominic, (2013), “Developing AR Games for iOs and Android”, United Kingdom, Packt Publishing.

7. Gonzalez Morcillo, Carlos, David Vallejo Fernandez, (2011), “Realidad Aumentada. Un enfoque Practico con ARToolKit y Blender”, España, Bubok Publishing S.L.

8. Alvares, J, (2008), “Objetivos de la realidad virtual”, información recuperada de la siguiente url: http://www.ugr.es/~sevimeco/revistaeticanet/Numero2/Articulos/Realidadvirtual.pdf

9. Ziqiang, W & Yuanzhou, L & Manqing, C & Zhe, Z & Xiang, Y, (2011), "Future Applications of Virtual Reality Technology Outlook", información recuperada de la siguiente url: http://ieeexplore.ieee.org.ezproxy.sibdi.ucr.ac.cr:2048/xpl/articleDetails.jsp?arnumber=6113659&newsearch=true&queryText=virtual%20reality%20in%20computer%20science

10. s/n, (s/a), "Documentación Unity", manuales recuperados de la página oficial de Unity en la siguiente url: https://docs.unity3d.com/es/530/Manual/UnityManual.html

11. s/n, (s/a), "Blender 2.80. Manual de referencia", manuales recuperados de la página oficial de Blender en la siguiente url: https://docs.blender.org/manual/es/latest/

12. s/n, (s/a), "Guia de introducción de Rift" pdf recuperado del siguiente url: https://images-na.ssl-images-amazon.com/images/I/A1ceYsnNvOS.pdf

13. Trenzano Álvarez, Mariola, (2017), "Desarrollo de un videojuego de plataformas 2D en Unity", pdf recuperado del siguiente url: https://riunet.upv.es/bitstream/handle/10251/91746/TRENZANO%20-%20Desarrollo%20de%20un%20videojuego%20de%20plataformas%202D%20con%20Unity.pdf?sequence=1

14. Blacjman, Sue, (2014), "Unity for Absolute Beginners", New York, United Stated, Apress.

15. Thorn, Alan, (2014), "Pro Unity Game development with c#", New York, United Stated, Apress.

16. Barambones, Juan, (2015), "Realidad Virtual", Madrid, España, Juan Barambones, pp:

17. s/n, Brackeys,(2017), “How to make a Dialogue System in Unity”, tutorial recuperado en el siguiente url: https://www.youtube.com/watch?v=_nRzoTzeyxU

18. Domínguez Diaz, Adrian, (2017), “Unity 2017.X Curso Practico”, España, rama, pp:313.