UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

ENTORNO VIRTUAL PARA EVALUAR PRÁCTICAS DE MANEJO

TESIS

Que para obtener el título de

Ingeniero en Computación

P R E S E N T A

Arturo Pérez De La Cruz

DIRECTOR DE TESIS

Ing. Luis Sergio Valencia Castro

Ciudad Universitaria, Cd. Mx., 2018

Agradecimientos

Este proyecto y mi formación académica es consecuencia en primer lugar de mis dos grandes mentores de vida, por todo su apoyo incondicional a pesar de las adversidades y el tiempo, siempre están a mi lado.

Es consecuencia del apoyo, ánimo y soporte de mi gran compañera de vida, ya que ha estado conmigo en todos los momentos de esta gran etapa de mi vida.

Es consecuencia de un gran profesor que no solo considero profesor, sino un gran mentor, compañero, socio y amigo, por toda la experiencia y apoyo que ha aportado a lo largo de este camino.

Y de todas y cada una de las personas que participaron en el proyecto.

Investigación realizada gracias al Programa de Apoyo a Proyectos de Investigación e Innovación Tecnológica (PAPIIT) de la UNAM en el Proyecto “Simulador Para Evaluar Prácticas de Manejo” clave IT 102015.

Agradezco a la DGAPA-UNAM la beca recibida.

Índice general

1.-Introducción

Esta tesis es el resultado de haber participado en el desarrollo de un proyecto acerca de la construcción de un simulador, para realizar una prueba de manejo mediante el uso de Realidad virtual. Para la implementación fue necesario construir un ambiente virtual tridimensional y se propuso construir una plataforma capaz de moverse conforme a los movimientos del auto virtual.

Está conformada por 5 capítulos, los cuáles se describen a continuación:

El primer capítulo trata acerca de justificación que se realizó para llevar a cabo el proyecto. Analizando una problemática en el distrito federal.

El segundo capítulo habla acerca de proponer varias herramientas tecnológicas actuales para poder llevar a cabo el desarrollo del proyecto.

El tercer capítulo abarca el desarrollo del proyecto dividido en varias etapas de desarrollo. Desde la planeación del proyecto, hasta la etapa de pruebas.

El cuarto capítulo consiste en el análisis de los resultados obtenidos a partir de las pruebas aplicadas en la simulación del sistema construido.

El quinto capítulo muestra las conclusiones obtenidas al haber participado en este desarrollo así como del trabajo a futuro que se pudiera continuar con el simulador construido.

Objetivos:

El objetivo principal de este proyecto consistió en la construcción de un simulador de manejo implementando tecnología de realidad virtual. Para poder llevar a cabo este objetivo se plantearon varios objetivos secundarios que fueron importantes para poder cumplir el objetivo principal.

Uno de los objetivos secundarios fue crear un ambiente virtual tridimensional, modelado desde cero, con la finalidad de poderlo implementar en el simulador.

Otro objetivo secundario fue construir una plataforma capaz de moverse en tiempo real, sincronizado al simulador para que pueda emular los movimientos en el interior de un auto real.

Por último se estableció que el simulador de manejo pueda ser capaz de realizar pruebas de manejo y evaluar algunas habilidades y conocimientos de las personas que utilizan el simulador.

Hipótesis:

Para llevar a cabo el cumplimiento de los objetivos se establecieron algunas hipótesis descritas a continuación.

Actualmente los simuladores de manejo se basan en la visualización del ambiente virtual por medio de pantallas, esto limita el campo de visión a los usuarios. Por esta razón, se establece que con la implementación de dispositivos para realidad virtual mejorará su campo de visión de las personas cuando y de esta manera se logrará una mejor inmersión durante la simulación...

La mayoría de los simuladores implementan un asiento fijo para el usuario y esto ocasiona que estos no tengan noción de los movimientos en una simulación, como por ejemplo ir a gran velocidad, o frenar repentinamente. Se establece que con la implementación de una plataforma que se mueva conforme se mueve el interior del auto virtual durante la simulación mejorará la inmersión de las personas al momento de realizar la simulación.

Alcances:

La construcción de simulador se realizó gracias a la colaboración de 15 personas aproximadamente de diversas carreras como ingeniería en computación, arquitectura, ingeniería mecatrónica y diseño gráfico. Divididos en varias áreas de desarrollo como son: planeación, elaboración de modelos tridimensionales, programación, construcción de hardware, integración, pruebas.

El desarrollo del proyecto se dividió en 2 fases, la primera consistió en la elaboración de todos los elementos necesarios para poder construir un simulador de manejo, como son: la elaboración de un entorno virtual, la implementación de tecnología de realidad virtual, la construcción de una plataforma con movimiento en tiempo real y la integración de todos estos elementos en un motor de render capaz de poder realizar la interacción con el usuario en tiempo real.

La segunda fase se trató acerca de elaborar un sistema capaz de realizar una prueba de manejo y a su vez evaluar algunas de las habilidades y conocimientos de los usuarios durante la prueba.

1.1Análisis de la problemática

Los accidentes viales son una de las causas de muertes con mayor concurrencia en nuestro país (figura 1.1): En la mayoría de los casos se debe a que las personas sólo realizan un trámite administrativo para obtener una licencia de conducir y no se sabe si realmente la persona está capacitada para poder conducir adecuadamente.

Actualmente en la Ciudad de México no existe algún tipo de prueba o examen que permita saber si las personas que solicitan una licencia de manejo, cuentan las capacidades teóricas y prácticas para obtenerla.

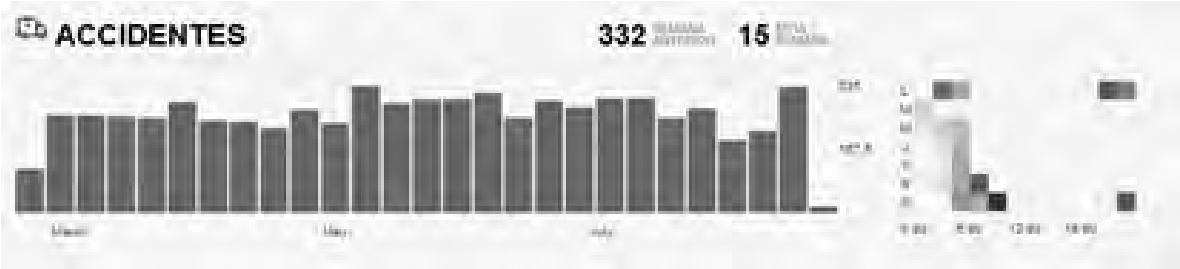


Figura 1.1.1 Gráfica que representa la cantidad de muertes por accidentes viales durante el periodo de marzo a julio del 2013 y los días de la semana en los que hay más accidentes.

Según datos de la Secretaría de Transporte y Vialidad (Setravi) se esperaba que a partir del 2014, los capitalinos que realizarán el trámite para expedición de licencia de conducir, tendría que hacer exámenes teóricos y prácticos, sin embargo hasta el día de hoy esto sigue sin concluir.

En México, las lesiones causadas por el tránsito siguen encontrándose entre las diez principales causas de muerte. En el 2014, se registraron 15,886 defunciones, cifra un 0.9% menor que lo registrado en el año previo. Con ello, se calcula una tasa de 13.3 muertos por cada 100 mil habitantes. De acuerdo con el Informe sobre la Situación de la Seguridad Vial en la Región de las Américas (OMS/OPS, 2015), México ocupa la posición número 20, de 32 países que conforman esta región [1].

1.2 Estudio Histórico acerca de las pruebas de manejo en el Distrito Federal

En la ciudad de México actualmente no se realiza ningún tipo de prueba teórica o práctica para que un ciudadano pueda obtener el permiso de conducir si eres menor de edad o la licencia para conducir si eres mayor de edad.

Hace varios años, para solicitar la licencia o permiso de conducir era necesario pasar por un examen teórico e incluso se realizaban pruebas médicas, pero a partir de 1996 quitaron las pruebas médicas.

Posteriormente con el paso del tiempo, los pasos para solicitar la licencia de conducir se convirtieron en solamente trámites administrativos, debido a la corrupción que se generó. Los ciudadanos al querer aprobar los exámenes sin siquiera conocer los lineamientos básicos del reglamento de tránsito, preferían pagar sobornos, como comúnmente se le llama en México.

Debido a la gran corrupción que se generó en torno a los trámites para conseguir la licencia para conducir, en el año 2003 el gobierno tomó la decisión de eliminar los exámenes para obtener la licencia de conducir. Lo único que se debe hacer es cumplir con el pago solicitado y llevar algunos papeles como comprobante de domicilio y una identificación oficial.

Con esa decisión se eliminó la corrupción en esa parte del trámite, pero no se puede saber si el ciudadano tiene las nociones básicas para conducir, si conoce el reglamento de tránsito, si ha tomado clases o ha conducido antes o incluso si es físicamente capaz de conducir un vehículo, como por ejemplo sufrir alguna discapacidad visual, auditiva, etc.

Por estas razones, los accidentes viales son una de las principales causas de muerte, cualquier persona que quiera solicitar el permiso de conducir solo debe cubrir con el trámite y listo. Sin importar si es responsable al conducir, si sabe o si puede.

Como una medida para evitar que esto siga ocurriendo, el gobierno implementó un sistema de puntos. Consiste básicamente en que si cometes una infracción o algún tipo de falta al reglamento de tránsito, le restaban puntos a tu licencia, al perder todos los puntos te quitan la licencia y ya no es posible tramitarla durante un periodo de tiempo establecido.

Esta medida se cree que no ha funcionado, debido a que muy pocos elementos de tránsito implementaron el sistema de puntos, otros caían en la corrupción aceptando sobornos con tal de no perder puntos. Otro factor que hay que recalcar es que el perder puntos no justifica el hecho de poder quitarle la vida a alguien en un accidente por no ser responsable al conducir.

Es por esta razón que desde el 2013 comenzó una propuesta por volver a implementar una prueba teórica y práctica para poder adquirir la licencia de conducir. La prueba consistiría en un cuestionario para la parte teórica, donde habría que tener conocimientos principalmente acerca del reglamento de tránsito y posteriormente realizar una prueba práctica en la que se evaluarían las capacidades de la persona para conducir.

Lamentablemente desde ese año, se anunciaba año tras año que ahora si se tiene planeado realizar dicha prueba para poder obtener la licencia de conducir, pero no se ha generado ninguna acción al respecto.

Posteriormente en el 2015 se lanzó una propuesta para realizar una prueba virtual, con el propósito de convertirla en requisito aprobar en un simulador si se quiere obtener la licencia para conducir. Las autoridades mencionaron que esta prueba virtual comenzaría en el 2016 y sería requisito obligatorio para poder obtener la licencia de conducir.

Con esta prueba se buscaba evaluar si la persona realmente tiene las capacidades físicas y las habilidades para conducir responsablemente. Otro factor es que al ser un sistema virtual, no se requiere de mucho personal o se necesita disponer de muchos vehículos físicamente para que se lleve a cabo. Y sobre todo se buscaba eliminar la corrupción ya que al ser un sistema computarizado, no es tan fácil poder corromperlo.

Desafortunadamente hasta la fecha no se ha realizado ningún tipo de prueba, por lo tanto el obtener la licencia continúa siendo un trámite administrativo solamente.

Con respecto a otros países, para solucionar el problema de accidentes viales, recurren a completar un examen teórico y práctico para la obtención de la licencia de conducir. Por ejemplo. En la Unión Europea se emplean exámenes que duran entre 40 a 60 minutos aproximadamente.

Cada país realiza sus propias pruebas en específico, aunque generalmente usan personas altamente calificadas por una escuela de manejo y certificadas a nivel estatal, como es el caso en Alemania, Austria y Suiza, además de que se pide como requisito haber tomado un curso de manejo en una escuela oficialmente reconocida por su país en específico.

Para la realización del examen teórico, es un test de opción múltiple y se puede contestar en línea, varios meses antes de realizar la prueba práctica.

El examen práctico se puede realizar hasta un mes antes de alcanzar la edad mínima requerida para tener la licencia de conducir, es requisito haber aprobado el test teórico previamente. Generalmente consiste en conducir en una calle transitada, en tener un control sobre la seguridad técnica, estacionarse en reversa, entre otros, dependiendo de cada país que aplique el examen.

Otro factor importante que influye para obtener la licencia de conducir es contar con la edad mínima necesaria dependiendo de cada país. En la Unión Europea las edades mínimas para poder realizar el examen son de 25 años para obtener la licencia de tipo A, 21 para la clase D, 18 para la clase B y C y 16 para la clase A (Tabla 1.2.1).

Tipo de licencia a obtener	Edad mínima
Tipo A	25
Clase D	21
Clase B y C	18
Clase A	16

Tabla 1.2.1 Tabla que muestra los diferentes tipos de licencia existentes en la unión europea y la edad mínima requerida para obtenerla.

Una particularidad adicional que se solicita en Londres es tener un pasaporte británico y cumplir con los requisitos del sentido de la visión, consiste en ver con ayuda o sin ayuda a una distancia mínima de 20 metros una matrícula. El examen teórico consta de 2 partes, una incluye 50 preguntas de opción múltiple para responder en 50 minutos y otra acerca de detección de riesgos en 14 videos de situaciones reales en carretera. Estas pruebas se realizan en el mismo día.

La prueba práctica consta de 40 minutos con el supervisor y otros 10 minutos de conducción independiente.

1.3 Realidad Virtual

La realidad virtual consiste en crear un entorno inexistente, hecho por medio de gráficos por computadora: Pudiendo ser cualquier cosa que se pueda imaginar, como por ejemplo un mundo fantástico, el espacio exterior, una ciudad. Este entorno puede ser creado lo más apegado o desapegado a la realidad como se quiera.

Para poder apreciar el entorno creado, es necesario contar con algún tipo de dispositivo, para poder mandar el contenido creado por computadora.

El concepto de realidad virtual comienza a surgir en 1844, con la creación del estereoscopio, fabricado por Charles Wheatstone, considerado como la base de los primeros visores de realidad virtual, se basa en colocar 2 imágenes casi iguales, al colocarlas una al lado de la otra en el campo de visión de cada ojo por separado, el cerebro mezcla las 2 en una sola imagen generando el efecto de tridimensional.

Posteriormente en 1891, Louis Ducos du Hauron patenta el Anaglifo, este sistema se basa en la imagen estereoscópica, pero que adicionalmente añade un filtro fotográfico distinto en cada ojo, uno de color rojo, para el ojo derecho y uno de color azul, para el izquierdo.

A partir de esa fecha se comenzaron a construir diversos aparatos para proyectar video en 3D, como por ejemplo el sensorama en 1961 construido por Morton Heilig. Posteriormente se construyó un casco de realidad virtual usando tubos de rayos catódicos construido por Ivan Sutherland.

Pasaron varios años después hasta que se comenzaron a emplear los gráficos por computadoras, en específico los gráficos en 3D, para que la realidad virtual pudiera avanzar significativamente.

En un principio la realidad virtual fue empleada con fines militares, para la construcción de simuladores de vuelo, con el propósito de entrenar a los pilotos. Posteriormente se empleó con fines recreativos. La compañía Nintendo lanzó la consola de realidad virtual, llamada “Virtual boy” (figura 1.3), la cual fue un fracaso debido a que provocaba fuertes dolores de cabeza. [2]

Figura 1.3.1 Consola de realidad virtual Virtual Boy.

Pasaron muchos años tratando de implementar mejores dispositivos de realidad virtual, pero sin mucho éxito, debido a que no se contaba con la tecnología para mostrar gráficos tan realistas sin que sea necesario de equipos muy grandes y costosos.

Posteriormente en el año 2000 Google contribuyó a que resurgiera el tema de realidad virtual con el lanzamiento de Street View, que consiste en poder visualizar un cualquier lugar por medio de imágenes acomodadas de tal manera que simula que el usuario se encuentra en ese lugar.

En el año 2012 se publicó un proyecto en la plataforma Kickstarter llamado Oculus Rift. Consiste en el primer dispositivo de realidad virtual capaz de lograr un ángulo de visión de 90 grados, esto no había sido logrado antes. Este dispositivo se diseñó para conectarse a una computadora de alto rendimiento mediante una conexión HDMI y USB, permitiendo que el usuario pueda apreciar el entorno virtual generado en la computadora.

A partir de allí en el 2016, muchas empresas grandes percibieron el potencial de esta tecnología y comenzaron a trabajar en la creación de sus propios dispositivos de realidad virtual, como es el caso de PlayStation VR, HTC Vive (Figura 1.3.2). Además de incorporar más sensores, algunos incorporan controles u otros dispositivos para lograr una mejor interacción entre el usuario y el entorno virtual.

Figura 1.3.2 Dispositivos de realidad virtual de diferentes compañías.

Otro factor importante que impulsó el desarrollo de la realidad virtual fue el gran crecimiento en la incorporación de sensores dentro de los dispositivos móviles, como giroscopios y acelerómetros. Este tipo de sensores permite una mayor integración de la realidad virtual con el usuario debido a que genera una interactividad más fácil e intuitiva. Así como también una notable disminución de costos en estos dispositivos.

El gran crecimiento que tuvo el mercado de los dispositivos permitió construir dispositivos para realidad virtual donde se usa el Smartphone como pantalla para visualizar e interactuar con el entorno virtual, tal es el caso de Google Cardboard. Este destaca por su bajo costo debido que inicialmente se publicó una plantilla de cartón, el cual involucra 2 cristales cóncavos y un par de imanes para tener interacción. Por otro lado el dispositivo móvil se coloca dentro del Cardboard (figura 1.3.3).

Figura 1.3.3 Google Cardboard, diseño que permite visualizar aplicación móvil en realidad virtual.

A partir de esta plantilla surgieron muchos otros dispositivos construidos principalmente de plástico, logrando la misma función que la del Cardboard (figura 1.3.4).

Figura 1.3.4 Diseños de otras marcas basados en cardboard para visualizar aplicaciones de realidad virtual.

Cabe resaltar que la calidad de video y rendimiento del entorno virtual dependen de las capacidades del dispositivo que se usa, por eso al realizarlo en un dispositivo móvil, varía mucho de un dispositivo a otro.

Por esa razón Samsung lanzó su propio dispositivo de realidad virtual, llamado Samsung Gear VR (figura 1.3.5), este dispositivo solo funciona con algunos teléfonos de gama alta de Samsung, como el Galaxy S6, S7 y S8 o Galaxy note 4. Además este dispositivo

incorpora sensores extra para una mayor precisión e interacción con el usuario. También cuenta con un acuerdo con Oculus para garantizar la calidad en cuanto a las aplicaciones ofrecidas para estos dispositivos. [3]

Figura 1.3.5 Dispositivos para visualizar aplicaciones de realidad virtual únicamente con móviles de la marca Samsung.

1.4 Estado del arte en simuladores

Simulador

Un simulador es un conjunto de elementos de software que pueden acompañarse de hardware, sirve para generar comportamientos de un sistema real en un ambiente virtual.

Las utilidades más comunes de un simulador son, entender el funcionamiento del sistema, estudiar el sistema sin correr riesgos mayores, reducir costos estudiando el sistema en un simulador virtual comparado con la construcción en la realidad.

Simuladores de manejo

Los simuladores de manejo han sido implementados desde hace ya varios años, principalmente con el propósito de entrenamiento y capacitación en unidades específicas, como por ejemplo en autos de carreras, o en entrenamientos de operativos para el ejército. Posteriormente se implementó para el uso en entretenimiento como videojuegos.

Los simuladores de manejo normalmente constan de un software específico para la simulación, se hace uso de elementos de hardware para su implementación, como son un equipo de cómputo, una o varias pantallas para visualizar la simulación y diversos dispositivos para la interacción, que van desde un teclado y mouse, hasta el uso de un asiento especial, volante, pedales y palanca de cambios de velocidad.

El software se enfoca en la construcción de un mundo virtual buscando recrear el mundo real lo más posible, se construye en 3D un escenario con medidas proporcionales a las reales y se crea un entorno con cualidades similares.

Se crean entornos virtuales de casos reales, que van desde casos simples como estacionarse, manejar de frente y de reversa, hasta casos muy extremos como puede ser manejar con lluvia intensa, manejar en la nieve, etc.(figura 1.4.1).

Figura 1.4.1 Imágenes de una simulación de manejo en diferentes situaciones.

Actualmente el software de los simuladores es muy realista y con el avance de la tecnología se logra apreciar visualmente que los gráficos por computadora se acercan cada vez más a la realidad.

El hardware se enfoca de realizar la interacción del mundo real con el mundo creado en la computadora, lo primero que se debe mencionar son los medios de visualización, para lograr eso actualmente lo más recurrente es el uso de pantalla y/o monitor y generalmente se usan 3 dispositivos (figura 1.4.2). Otros simuladores emplean proyectores sobre superficies curvas de gran tamaño.

Figura 1.4.2 Ejemplo del hardware implementado normalmente para simuladores.

Implementar 3 pantallas se hace con la finalidad de tener un campo de visión mayor para la persona que usa el simulador, de esta manera se logra un mejor acercamiento a cómo lo hacemos en el mundo real.

Otro factor importante para hacer más realista la simulación es el uso de dispositivos como volantes y pedales, con esto se busca manejar un carro como lo hacemos en la vida real. Aunque algunos no son tan precisos o sensibles a los reales.

Ciertos simuladores emplean el uso de asientos personalizados o cabinas (figura 1.4.3), esto con la finalidad de colocar sistemas de audio envolvente que ayuda a dar más realismo durante la simulación. Algunos incluyen sistemas mecánicos para darle movimiento al asiento y así lograr que se tenga una interacción del sistema virtual a la persona que usa el simulador.

Figura 1.4.3 Ejemplos de asientos personalizados para la simulación de manejo.

Para poder integrar el software con el hardware es necesario contar con computadoras de gran capacidad, algunas de las características que tienen este tipo de simuladores son:
Procesador Intel core i7, 8 GB de memoria RAM, tarjeta de video dedicada de 2GB, conexiones suficientes para diferentes periféricos como volante, pedales, pantallas, etc.[4][5]

1.5Marco teórico:

Se presentan algunos temas considerados necesarios para una mejor comprensión del desarrollo en el proyecto.

Metodología SCRUM:

Es considerada una metodología ágil para el desarrollo de proyectos de software, para varios equipos de trabajo, ya que permite dividir un proyecto en varios desarrollos pequeños, de esta manera se puede trabajar éstos en paralelo para acelerar la producción.

Esta metodología consiste en realizar avances llamados Sprints dentro de cada desarrollo, con entregas evaluadas en semanas, generalmente no más de cuatro semanas. Las entregas realizadas sirven para obtener retroalimentación del avance del proyecto e ir midiendo el avance total del desarrollo.

Está conformado por roles, eventos, plantillas y reglas que se asocian entre sí, y cada elemento tiene un propósito en específico, que por medio del conocimiento empírico se busca obtener el mejor resultado en el desarrollo.

Dentro de la metodología se consideran 3 bases fundamentales:

Transparencia: Se establece que para el buen avance y desarrollo del proyecto se debe aclarar la forma de comunicación, las herramientas y lenguajes de programación de manera que sea clara y entendible por todos los miembros del equipo.

Inspección: Habla acerca de que se deben realizar inspecciones frecuentes pero sin impedir el tiempo de trabajo en el desarrollo, se recomienda a una persona experta en el campo pero que no se encuentre participando en el desarrollo.

Adaptación: Se refiere a que una vez realizada la inspección de algún elemento y no cumple con las características requeridas, este elemento se deba ajustar lo antes posible para retomar el cumplimiento de las necesidades.

Los Sprint deben contemplar varios elementos como por ejemplo, la planificación, los objetivos, la retrospectiva, etc.

Algoritmo de búsqueda A estrella:

Es un algoritmo de búsqueda también conocido como A asterisco (a*), presentado en 1968 por Peter E. Hart, Nils J. Nilsson y Bertram Raphael , se basa en encontrar soluciones en grafos.

Debido a que se basa en grafos se debe establecer un nodo origen y un nodo objetivo.

Este algoritmo sirve para encontrar una ruta, cumpliendo con condiciones preestablecidas en un inicio, siempre y cuando que exista una.

El algoritmo A* es usado para encontrar la ruta más cercana para ir de un lugar a otro (figura 1.5.1) (llamados nodos), es el más usado debido a que es sencillo y rápido.

Figura 1.5.1 diagrama para ejemplificar el funcionamiento del algoritmo a*.

La representación es la siguiente:
f(n) = g(n) + h(n)
g(n) es la distancia total que se toma de ir de la posición inicial a la posición actual.
h(n) es la distancia estimada desde la posición inicial a la posición de destino del final, en este caso se usa una función heurística para calcular el valor estimado.
f(n) es la suma de g(n) y h(n), y es el valor calculado más corto.

En otras palabras lo que se necesita es:
– Un nodo o punto inicial.
– Un nodo final que representa el final del algoritmo.
– Un método para identificar qué nodos son traspasables y cuales son sólidos.
– Un método para determinar el costo directo (g) de moverse entre los nodos.
– Un método para determinar el costo indirecto (h).
– Una lista de nodos abiertos, en esta lista se guardarán todos los nodos que se han identificado como posibles movimientos, pero aún no han sido evaluados.
– Una lista de nodos cerrados, donde se guardaran todos los nodos evaluados y descartados, aunque no es necesario una lista, basta con un estado que indique que el nodo se encuentra cerrado.
– Una forma de identificar qué nodo procede a otro, para poder retornar la cadena de los nodos.

De manera gráfica se ejemplifica de la siguiente manera, para poder satisfacer la forma de identificar los nodos sólidos y los que son traspasables y cada nodo será representado por un cuadrado.

Si vamos a permitir que los actores del juego puedan moverse diagonalmente, debemos darle un costo directo más bajo al movimiento en diagonal que al movimiento en línea recta de dos nodos, por ejemplo:

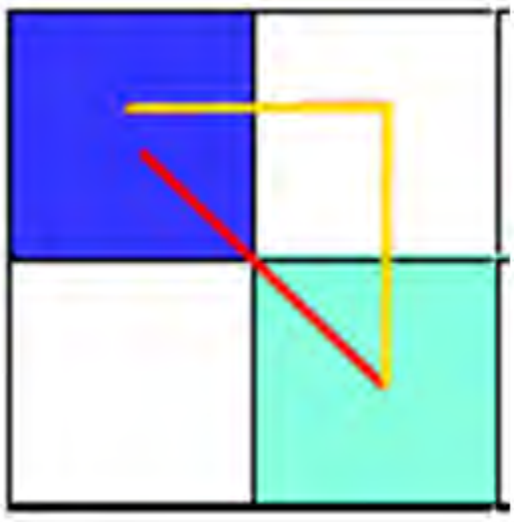


Figura 1.5.2 Representación de movimiento en el algoritmo a*.

En la imagen anterior (figura 1.5.2), la línea amarilla representa el movimiento de ir del nodo azul claro al nodo azul oscuro, pero sin hacer ningún movimiento diagonal, si el costo de moverse de un nodo a otro es de 10, el resultado de la línea amarilla es de 20,

en cambio el movimiento de la línea roja, que es diagonal debe ser de 15 o 14, que sería el resultado del teorema de Pitágoras.

Ahora, para el valor indirecto se usa la heurística que es un cálculo aproximado de ir de un nodo actual al nodo final o destino, en la mayoría de casos usan como heurística la distancia de Manhattan o la distancia Euclidiana.

El algoritmo es el siguiente:

1. Si el nodo inicial es igual al nodo final, se retorna el nodo inicial como solución
2. Si no, se adiciona el nodo inicial a la lista abierta
3. Mientras la lista abierta no esté vacía, se recorre cada nodo que haya en la lista abierta y se toma el que tenga el costo total más bajo
4. Si el nodo obtenido es igual al nodo final, se retornan todos los nodos sucesores al nodo encontrado
5. Si no , se toma el nodo y se elimina de la lista abierta para guardarse en la lista cerrada y se buscan todos los nodos adyacentes al nodo obtenido y se adicionan a la lista abierta a menos que el nodo se encuentre en la lista cerrada o que el nodo sea sólido
6. Si el nodo adyacente ya se encuentra en la lista abierta se verifica que el costo sea menor, si es menor se cambian los valores de costo, sino se ignora
7. Se vuelve al paso 3 y se repite hasta que el punto 4 sea verdadero o que la lista abierta quede vacía

Un ejemplo es el siguiente (figura 1.5.3):

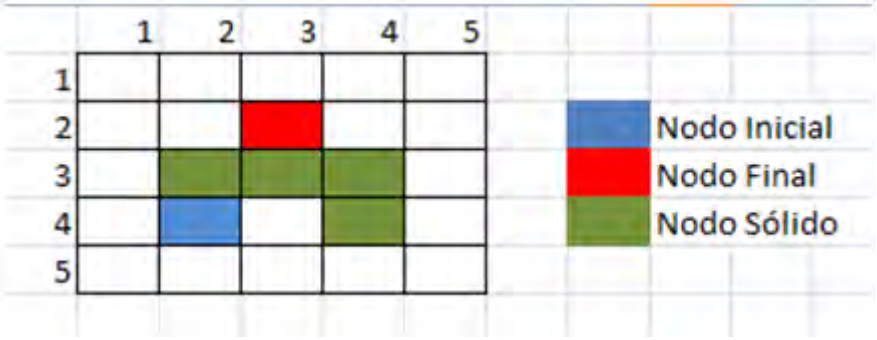


Figura 1.5.3 Ejemplo del funcionamiento del algoritmo a*.

El nodo inicial es el nodo (2,4) o el que está de color azul, el nodo final es el nodo (3,2) o nodo rojo, los nodos de color verde son nodos sólidos y no pueden ser traspasados.
Al empezar el algoritmo tendremos el nodo azul como opción y lo agregamos a la lista abierta, luego como no es el nodo final, obtenemos sus nodos adyacentes y luego lo dejaremos en la lista cerrada, calculamos los valores de los nodos de la lista abierta (a menos que ya los hayamos calculado) y tomamos el de menor valor:

Figura 1.5.4 Cálculo de los valores en los nodos del ejemplo.

La heurística que se ha usado es la distancia de Manhattan (figura 1.5.4):
H = Math.Abs(nodoActual.X – nodoFinal.X) + Math.Abs(nodoActual.Y – nodoFinal.Y)
Como ejemplo tomamos el cuadrado B (1,4) hasta el nodo final (3,2)
H = (1 – 3) + (4 – 2)
H = 2 + 2
H = 4
Lo que se hizo fue contar los cuadrados que hay para llegar del nodo actual al nodo final. Ahora, el nodo que tiene el menor valor es C, por lo tanto buscamos sus nodos adyacentes, dando como resultado el nodo E y F porque los demás son sólidos y el nodo (4,5) aunque es alcanzable diagonalmente, es inalcanzable porque uno de los nodos cerca es sólido.
Se añade el nodo C a la lista cerrada y los demás nodos a la lista abierta, pero como ya se encuentran, se verifica que el valor calculado no sea menor, los costos nuevos de los nodos son (figura 1.5.5):

	Costo del Padre	G	H	T
E	12	15	4	31
F	12	10	3	25

Figura 1.5.5 Nuevos valores para los nodos E y F

Como el costo no es menor que el que tienen en la lista abierta, se descartan y se vuelve a tomar el menor valor de la lista abierta, lo que da un empate entre B y E, pero si tomamos el nodo E, volveríamos a descartar los demás nodos, volviendo a tomar el valor B porque el E es enviado a la lista cerrada [16].
Si se sigue con el algoritmo va a dar lo siguiente (figura 1.5.6):

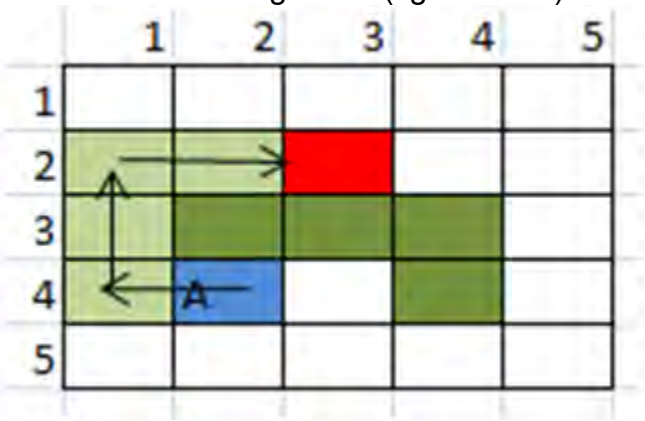


Figura 1.5.6 Solución obtenida por el algoritmo

2.- Propuesta de tecnologías a utilizar

2.1 Análisis de las tecnologías de software

Para llevar a cabo el desarrollo del simulador de manejo, se realizó un análisis de las herramientas que se pueden utilizar, revisando sus características técnicas, las ventajas y desventajas frente a otras herramientas similares. Esto con la finalidad de saber cuáles son las que más se adecuan a las necesidades del proyecto.

Para la selección de las herramientas de software que se utilizaron durante el desarrollo, se tomó como criterio la experiencia de cada uno de los participantes en el desarrollo para la realización del mismo.

A continuación se muestran las herramientas propuestas.

Herramientas de Software

Las herramientas de software son muy diversas por lo tanto se realizó una división por área de desarrollo de acuerdo a las necesidades del proyecto: modelado y animación 3D, arte 2D, manipulación de hardware e integración. A continuación se detallarán las características de las posibles herramientas que pudieron ser utilizadas, así como ventajas y desventajas que contribuyeron en la elección final.

A continuación se mostrarán las herramientas de software propuestas para el desarrollo, con una breve descripción la herramienta y los requerimientos de hardware necesarios para su correcto funcionamiento.

La propuesta de software se dividió por áreas de desarrollo descritas a continuación:

- Desarrollo en 3D: Se refiere a la creación y animación de modelos virtuales tridimensionales. Como por ejemplo modelos de casas, autos, semáforos, etc.

  Herramientas propuestas Blender, 3ds Max, Maya.

- Desarrollo en 2D: Se refiere a la creación de contenido virtual bidimensional Como por ejemplo texturas, imágenes de fondo, imágenes, etc.

  Herramientas propuestas Gimp, substance.

- Motor de render en tiempo real: Se refiere a la integración de todos los elementos desarrollados en un motor de render para que se puedan mostrar en tiempo de ejecución y de esta manera obtener una interacción con el usuario.

Herramientas propuestas Unity, Unreal.

Desarrollo en 3D:

Blender

Es una herramienta gratuita de código abierto (figura 2.1.1), que contiene una suite de varios elementos para la creación, manipulación y producción de elementos en 3D. Blender cuenta con herramientas de modelado y esculpido en 3D, rigging, animación, simulación, captura de movimiento, editor de video y herramienta de creación de videojuegos.

Además cuenta con una parte donde se pueden añadir códigos para personalizar las herramientas o crear nuevas. Este código usa el lenguaje Python.

Al ser de código abierto cuenta con miles de desarrolladores que contribuyen al proyecto y cuenta con bastante documentación. [6]

Requerimientos de hardware mínimos:

Procesador de 32 bits de doble núcleo con soporte para SSE2
2 GB de memoria RAM
Monitor de 24 bits y resolución de 1280x768
Mouse
Tarjeta de video con 512 MB de memoria compatible con OpenGL 2.1.

Requerimientos de hardware recomendados:

Procesador de 64 bits de cuatro núcleos
8GB de memoria RAM
Monitor de 24 bits con resolución HD
Mouse con al menos 3 botones
Tarjeta de video con 2 GB de memoria compatible con OpenGL 3.2

Figura 2.1.1 Logotipo del software para realizar modelado y animación “blender”.

3DS MAX

Software de modelado, animación y renderizado en 3D (figura 2.1.2).

Contiene herramientas para la creación, edición y producción de elementos en 2D y 3D, de animación como trayectorias de movimiento, herramientas de animación y manipulación de personajes, efectos de sistemas de partículas, modelado de mallas y superficies, asignación y edición de texturas, renderizador Raytracer.

Capacidad de integrar códigos de herramientas usando Max script.

Para su uso comercial es necesario adquirir una licencia que va desde los 2,466.00M.N. Mensuales. Cabe resaltar que se puede usar mediante un convenio con la universidad, obteniendo licencia gratuita de estudiante por 3 años. [7]

Requerimientos de hardware:

Procesador Intel o AMD de 64 bits de núcleos múltiples
4 GB de memoria RAM
4.5 GB de espacio libre en disco duro
Mouse con al menos 3 botones
Tarjeta de video con al menos 1 GB de memoria.

Figura 2.1.2 Logotipo del software para realizar modelado y animación “3DS Max”.

Maya

Software de renderización, simulación, modelado y animación 3D (figura 2.1.3).

Ofrece un conjunto de herramientas integrado y potente, que puede usar para crear animaciones, entornos, gráficos de movimiento, realidad virtual y personajes. Herramientas de edición de mapas de texturas UV, vínculo activo con el software After effects para realizar cambios en tiempo real, múltiples herramientas para el modelado y la animación en 3D.

Para su uso comercial es necesario adquirir una licencia que va desde los 2,468.25M.N. Mensuales. Cabe resaltar que se puede usar mediante un convenio con la universidad, obteniendo licencia gratuita de estudiante por 3 años. [8]

Requerimientos de hardware:

Procesador Intel o AMD de 64 bits de núcleos múltiples
4 GB de memoria RAM
4.5 GB de espacio libre en disco duro
Mouse con al menos 3 botones
Tarjeta de video con al menos 1 GB de memoria.

Figura 2.1.3 Logotipo del software para realizar modelado y animación “Maya”.

A continuación se muestra una tabla (tabla 2.1.1) comparando las herramientas con las que cuenta cada uno de los softwares presentados, así como los formatos a los que permite exportar los modelos y el tipo de licencia con la que se debe contar para el uso del mismo

Software	Herramientas de modelado 3D	Herramientas de animación	Herramientas de edición de UV	Formatos para exportar datos	Tipo de licencia
Blender	Si	Si	Si	obj,3ds,blend, fbx, dae	Gratuita
3Ds Max	Si	Si	Si	obj,3ds,max, fbx, dae	2466 M.N. Mensuales o Licencia de estudiante por 3 años.
Maya	Si	Si	Si	obj,3ds,ma,mb, fbx, dae	2468.2 M.N. Mensuales o Licencia de estudiante por 3 años.

Tabla 2.1.1 Tabla comparativa de las herramientas con las que cuenta cada software para Modelado y animación

Debido a que todos los software para modelado cuentan con las características necesarias y los formatos en los que exportan los modelos 3D son compatibles con la mayoría de los motores gráficos existentes (se utilizará el formato FBX), por motivos técnicos se puede utilizar cualquiera de estas herramientas de modelado.

La única gran diferencia es la interfaz gráfica y los atajos de teclado que tiene cada uno de los programas, por lo tanto el software que se use dependerá de la experiencia que cada participante tiene en el uso del software para modelado y animación.

Debido a que se cuenta con experiencia previa en el uso de la herramienta 3Ds Max, y se apoyó impartiendo cursos en la Facultad de Ingeniería para aprender a utilizar este software, se decidió trabajar con 3Ds Max.

Desarrollo 2D:

GIMP

GIMP es un acrónimo de GNU Image Manipulation Program (figura 2.1.4). Es un programa de libre distribución para tareas como el retoque fotográfico, la composición de imágenes y la creación de imágenes. Los términos de uso y las reglas sobre la copia se enumeran claramente en la Licencia Pública General de GNU. [9]

Requerimientos de hardware:

Procesador Intel o AMD de 64 bits.
4 GB de memoria RAM
1 GB de espacio libre en disco duro
Mouse con al menos 3 botones

Figura 2.1.4 Logotipo del programa para creación y edición de imágenes.

Substance

Conjunto de herramientas diseñadas para la creación de texturas y materiales procedurales, compatible con la mayoría de los motores gráficos y programas de modelado.

Contiene creador de materiales por medio de nodos (figura 2.1.5), aplicador de materiales en 3D, generadores de ruido, visualización en tiempo real de los materiales y texturas creados. [10]

Requerimientos mínimos de hardware:

Procesador de 64 bits de ocho núcleos
8GB de memoria RAM
1 monitor de 24 bits con resolución HD
Mouse con al menos 3 botones
Tarjeta de video con 2 GB de memoria compatible con OpenGL 3.2

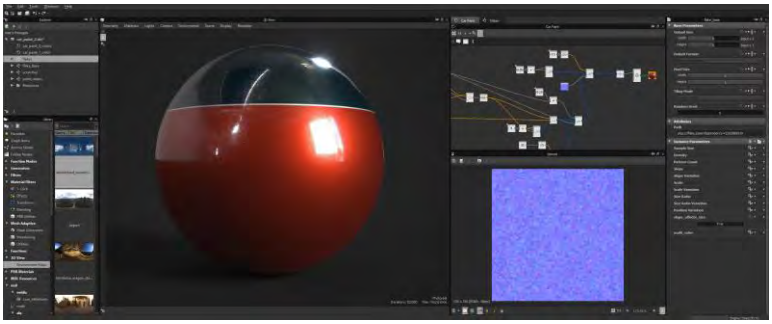


Figura 2.1.5 Vista de la herramienta para creación de materiales por nodos.

Motor de render en tiempo real:

Unreal Engine

Motor gráfico (figura 2.1.6) que contiene herramientas de desarrollo para trabajar con ambientes 3D en tiempo real, esta permite compilar aplicaciones para múltiples plataformas como PC, web, consolas, dispositivos móviles, dispositivos para realidad virtual y realidad aumentada.

Es un motor de render en tiempo real, contiene: motor de física, manipulación mediante lenguaje C++, creación de proyectos mediante plantillas (no requiere programación), sistemas de partículas, editor de materiales, herramientas de animación, editor de realidad virtual en modo realidad virtual, editor de terreno, inteligencia artificial pre programada , etc.[11]

Requerimientos de hardware:

Procesador de 64 bits de cuatro núcleos
8GB de memoria RAM
1 monitor de 24 bits con resolución HD
Mouse con al menos 3 botones
Tarjeta de video con 2 GB de memoria compatible con OpenGL 3.2

Licencia gratuita para desarrollo y 5% de las ganancias si se comercializa un proyecto realizado con este software.

Figura 2.1.6 Logotipo del programa para renderizar en tiempo real “Unreal”.

Unity

Motor gráfico (figura 2.1.7) que contiene herramientas de desarrollo para trabajar en ambientes 3D en tiempo real, con la capacidad de compilar aplicaciones para múltiples plataformas como PC, web, consolas, dispositivos móviles, dispositivos para realidad virtual y realidad aumentada.

Motor de render en tiempo real, que contiene características como: motor de física, manipulación mediante lenguaje C# y java script, sistemas de partículas, editor de materiales, herramientas de animación, editor de terreno e inteligencia artificial pre programada. [12]

Requerimientos de hardware:

Procesador de 64 bits de cuatro núcleos
8GB de memoria RAM
1 monitor de 24 bits con resolución HD
Mouse con al menos 3 botones
Tarjeta de video con 2 GB de memoria compatible con OpenGL 3.2

Figura 2.1.7 Logotipo del programa para renderizar en tiempo real “Unity”.

Debido a que se cuenta con experiencia previa en el uso de la herramienta y se apoyó impartiendo cursos en la Facultad de Ingeniería para aprender a utilizar este software, se decidió trabajar con Unity.

2.2 Análisis de las tecnologías de hardware

El hardware se dividió en varias categorías:
*Equipo de procesamiento
*Interfaces de comunicación.
*Dispositivos de despliegue.

El equipo de procesamiento comprende a la máquina donde se controlará todo el proyecto, desde el software, interfaces y dispositivos de despliegue necesario para que el simulador funcione.

Para el simulador, siguiendo los requerimientos de los programas que se usarán, se solicitó un equipo de cómputo PC (figura 2.2.1) con las siguientes características:

Procesador Intel Core I7 de 4 núcleos.
Tarjeta madre Gigabyte modelo Z97-HD3P.
16 GB de memoria RAM tipo DDR3.
Disco duro mecánico de 1TB de capacidad.
Tarjeta de video NVIDIA GeForce GTX 970 con 4GB de memoria tipo GDDR5.
Gabinete Aero Cool.
Mouse y teclado.

Figura 2.2.1 Imagen del equipo de cómputo.

Interfaces de comunicación

Las interfaces comprende todos los dispositivos de hardware que nos ayudan a la interacción entre el usuario con el simulador.

Volante Logitech G27

Volante, con pedales y palanca de cambios manual (figura 2.2.2).

Este volante servirá como medio de interacción entre el usuario y la unidad de procesamiento, logrando un mayor realismo durante la simulación, el volante y pedales tienen una sensibilidad comparable a la de los autos reales.

Los pedales y palanca de cambios se conectan mediante una conexión única al volante, y el volante se conecta alámbrica mente a la computadora mediante conexión tipo USB, además se debe conectar un cable de alimentación de corriente eléctrica adicional, este cable se conecta a una toma de corriente de 110v a 127v.

El volante cuenta con 6 botones en la parte de enfrente, tira de led indicadores y 2 palancas en la parte de atrás, la palanca cuenta con 8 botones y un pad de dirección de 2 ejes. Se cuenta con 3 pedales, generalmente usados como clutch del lado izquierdo, freno en medio y acelerador del lado derecho. [13]

Figura 2.2.2 Imágenes del volante palanca y pedales respectivamente.

Tarjetas de control

Se analiza trabajar con las siguientes herramientas para poder manipular un asiento que pueda moverse en tiempo real durante la simulación.

Se realizó una búsqueda de distintas opciones de tarjetas de desarrollo de hardware y microcontroladores para realizar una comparación de los mismos, pudiendo destacar ventajas y desventajas de los componentes seleccionados.

De esta manera se destacaron los componentes más óptimos para establecer una comunicación entre la plataforma física y una computadora.

Se seleccionaron los siguientes controladores:

- Microprocesador PIC16F882
- Microcontrolador Arduino Uno
- Tarjeta de Control Galileo Gen2

Se analizaron las ventajas y desventajas de cada uno de los dispositivos, pudiendo destacar las siguientes:

PIC16F882

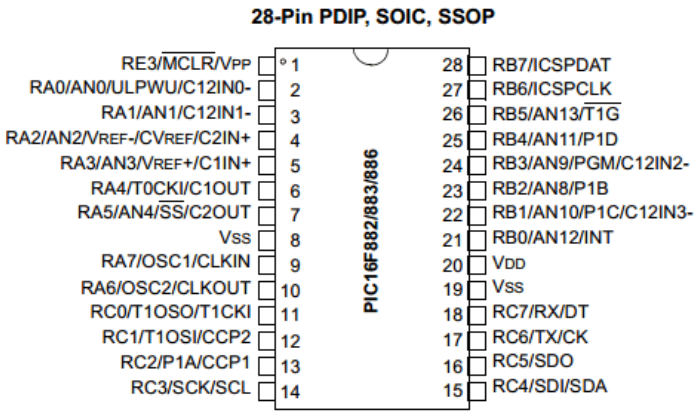


Figura 2.2.3 Diagrama del circuito PIC16F882.

- **Ventajas**
  - Bajo costo.
  - Uso de lenguaje C para programar.
  - Cubre necesidades básicas de control.
- **Desventajas**
  - Se requiere hardware adicional para conexión serial con PC.
  - Se requiere de un programador especial como Master prog para cargar el código.
  - No hay tarjetas de control armadas con este pic, por lo tanto se tendría que crear una.
  - Tecnología muy vieja.

- **Arduino UNO**

Figura 2.2.4 Imagen de la tarjeta de control Arduino UNO

- **Ventajas**
  - Costo medio.
  - Uso de IDE de arduino para programar.
  - Conexión serial con PC mediante puerto USB.
- **Desventajas**
  - Poca capacidad de expansión con otros módulos.
  - No tiene conexiones de red en caso de ser necesario.

- **Galileo Segunda Generación**

Figura 2.2.5 Imagen de la tarjeta de control Galileo GEN2.

- **Ventajas**
  - Conexión serial con PC mediante USB.
  - Programación con IDE de arduino.
  - Conexión Ethernet para implementaciones de red en caso de ser necesario.
  - Puerto de memoria micro SD
- **Desventajas**
  - Costo alto.

Después de analizar las opciones se trabajaron los diseños funcionales con el hardware Arduino UNO e Intel Galileo GEN 2.

Dispositivo de despliegue

Los dispositivos de despliegue nos permiten mostrar al usuario el entorno virtual creado. Para la construcción del simulador se ha elegido el dispositivo Oculus Rift DK2 (figura 2.2.6).

Es un dispositivo de realidad virtual el cual consta de un casco con una pantalla OLED de alta resolución. Un par de lentes, que se ajustan para ajustar la imagen, el casco incluye sensores, para garantizar un mayor realismo, de esta manera al girar la cabeza, varía el entorno que estamos observando en el dispositivo.

Figura 2.2.6 Imagen del dispositivo Oculus Rift DK2.

Cuenta con una cámara (figura 2.2.7) de posicionamiento para que pueda ubicar si te desplazas, aunque su rango de alcance es muy poco.

Figura 2.2.7 Imagen de la cámara de posicionamiento.

Para su conexión es necesario una computadora con al menos 2 puertos USB disponibles, una tarjeta de video que permita trabajar a resoluciones de 1080 a 75 fps mínimo con una conexión HDMI o DVI.[14]

3.- Desarrollo (Etapas a desarrollar)

3.1 Etapa de planeación

Esta etapa comprende la organización del equipo de trabajo del proyecto a desarrollar. El proyecto se realizó con un equipo de 12 personas principalmente, repartidos en varias áreas para trabajar paralelamente.

Se propone trabajar el desarrollo del proyecto con una metodología SCRUM, la cual es una de las metodologías ágiles más populares para el desarrollo de software.

Con esta metodología, se establecen los requerimientos generales del proyecto y posteriormente dividirlo en requerimientos particulares para cada módulo.

Establece una forma de trabajo incremental, es decir, cada cierto periodo de tiempo se tendrán avances de cada módulo de trabajo por medio de iteraciones.

Por otro lado se establece la metodología XP (Extreme Programing) para cada módulo de trabajo, con esto se pretende realizar todo el proceso de construcción en paralelo para terminar el proyecto en el menor tiempo posible, para posteriormente integrar cada uno conforme se tengan avances significativos.

Se planeó trabajar varias etapas las cuales comprenden:

- Modelado 3D
- Programación
- Hardware
- Integración
- Pruebas

3.2 Etapa de Modelado 3D

Esta etapa comprende al equipo de trabajo encargado de crear todo el entorno virtual en 3D que consiste en un escenario que incluirá diferentes modelos de tipo habitacional, comercial, edificios y supermercados.

Se propuso una lista de modelos genéricos, basado en artículos relacionados con el mobiliario urbano de la CDMX, los cuales se repartió el trabajo entre las personas asignadas a este módulo.

Se buscaron imágenes como referencias visuales (figura 3.2.1) de los modelos que se realizaron.

Figura 3.2.1 Imágenes de referencia para la creación de los modelos en 3D.

Debido a la carga de trabajo se deberá realizar la unidad de procesamiento, se han considerado unas restricciones a cada modelo en 3D, esto con la finalidad de optimizar lo mejor posible la carga de trabajo.

Estas restricciones son:

-El archivo deberá estar en formato FBX.
-Se estableció como unidad de medida un metro, por lo que se deberá respetar la escala de los modelos con base en la unidad establecida.
-Las texturas de los modelos deberán estar dentro del archivo FBX.
-La geometría de los modelos debe ser tipo primitiva o convertida a polígonos.
-Realizar los modelos con la menor cantidad de polígonos posible, cada modelo no deberá exceder los 1,500 polígonos.

Se dividió la lista de modelos en categorías como se muestra a continuación:

Autos:

Se realizaron 3 tipos de autos distintos. Los autos se moverán con inteligencia artificial, el que maneja el usuario, y los que estarán estacionados sobre las calles.

En el caso de los autos que estarán estacionados y los que se mueven mediante inteligencia artificial (figura 3.2.2), sólo se realizó el modelo por la parte el exterior, con la finalidad de no emplear muchos polígonos en esos modelos.

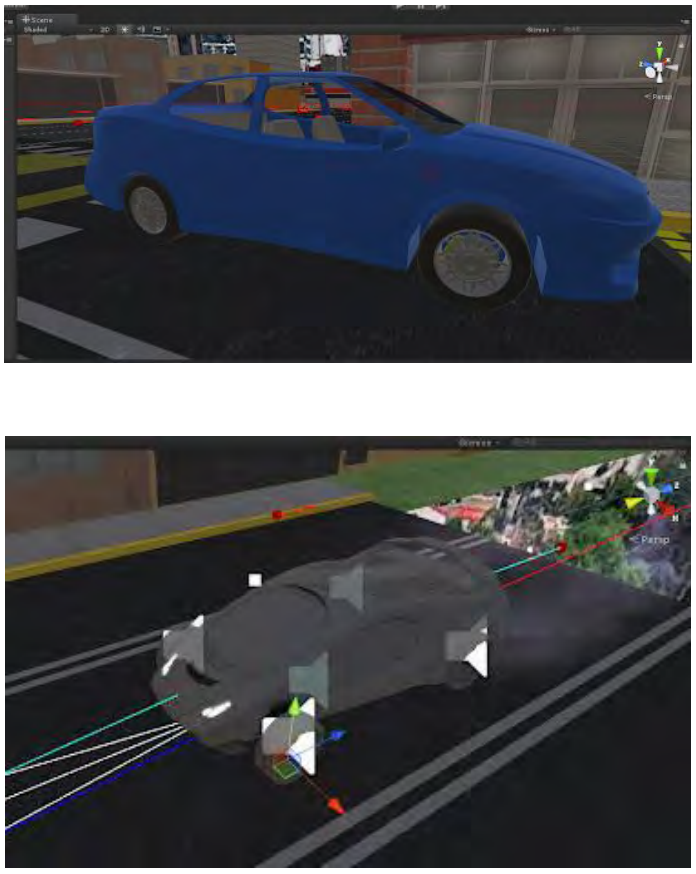


Figura 3.2.2 Vista de los modelos de los autos que están estacionados y los que se mueven con IA respectivamente.

En el caso del auto usado para la prueba de manejo, fue necesario crear el modelo tridimensional de la parte exterior e interior (figura 3.2.3), buscando el mayor detalle posible, para dar mayor realismo al usuario. En este caso por la razón antes mencionada este auto sí sobrepasa la cantidad de polígonos establecida.

Se propone que la malla tenga las proporciones promedio de un vehículo actual, debido a que son muy variadas sobre todo en longitud, se considera que las dimensiones más apropiadas para un sedán estén en el rango siguiente:

Ancho 1.6 - 1.8 [m]
Largo 4.3 - 4.6 [m]
Alto 1.4 - 1.5 [m]

El modelo contiene las dimensiones:

Ancho 1.7 [m]
Largo 4.5 [m]
Alto 1.45 [m]

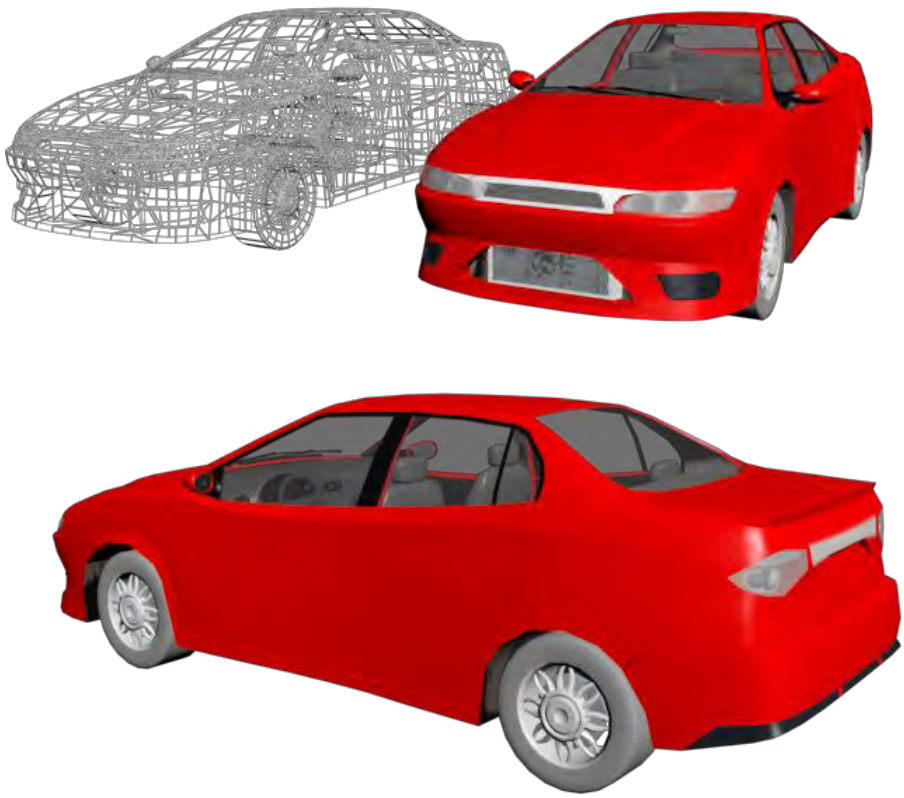


Figura 3.2.3 Diferentes vistas del modelo que manipula el usuario durante la simulación.
Zona Habitacional:

Comprende modelos de casa habitación (figura 3.2.4), únicamente realizando las fachadas de éstas, serán diferentes modelos genéricos, y tienen un color en la textura diferente por zona.

Figura 3.2.4 Imágenes de modelos tridimensionales realizados de tipo casa habitación.

Los modelos deben cubrir una dimensión de 16 unidades cuadradas, pudiendo ser de uno o dos pisos máximo, sin contar el espacio de la banqueta.

Zona de comercios:

Comprende a modelos de establecimientos (figura 3.2.5) y mercados, estos deben tener una dimensión de 8 unidades cuadradas, teniendo únicamente un piso.

Figura 3.2.5 Imágenes de modelos tridimensionales tipo local comercial.

El modelo del supermercado (figura 3.2.6) él puede tener 16 unidades cuadradas.

Figura 3.2.6 Imagen de modelo tridimensional tipo supermercado.

Zona de edificios:

Comprende a modelos realizados para edificios (figura 3.2.7) habitacionales, estos deben respetar un tamaño de 32 unidades cuadradas, sin límite de pisos, siempre y cuando no exceda el número de polígonos antes establecido.

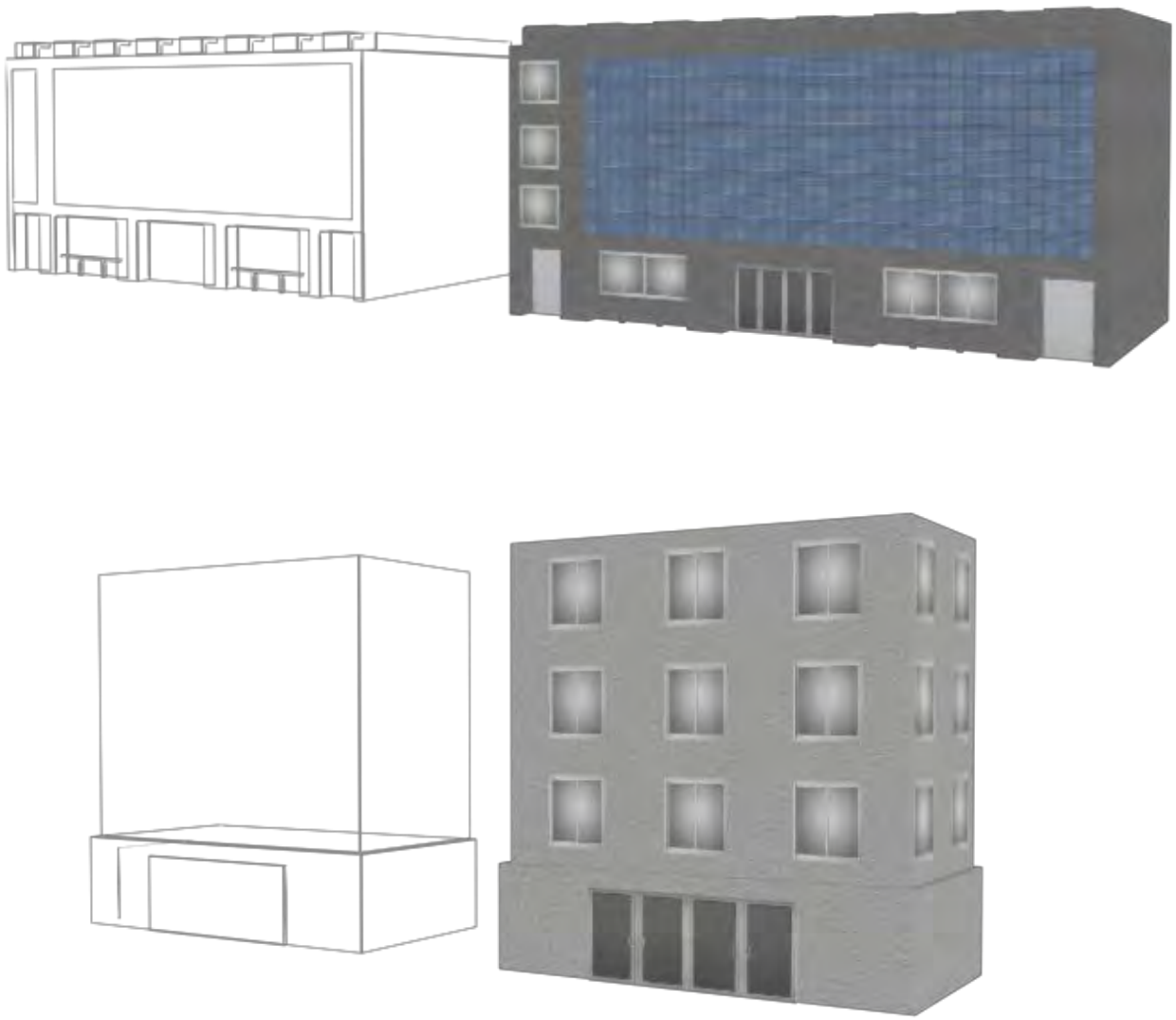


Figura 3.2.7 Imágenes de modelos tridimensionales de tipo edificio.

Zona de calles y ambientación.

Comprende a los modelos realizados para la creación de las calles (figura 3.2.8) por donde se realizará la prueba, en este caso se realizaron modelos genéricos para replicarlos y formar las cuadras.

Banqueta
Esquina de banqueta
Pavimento
Semáforos

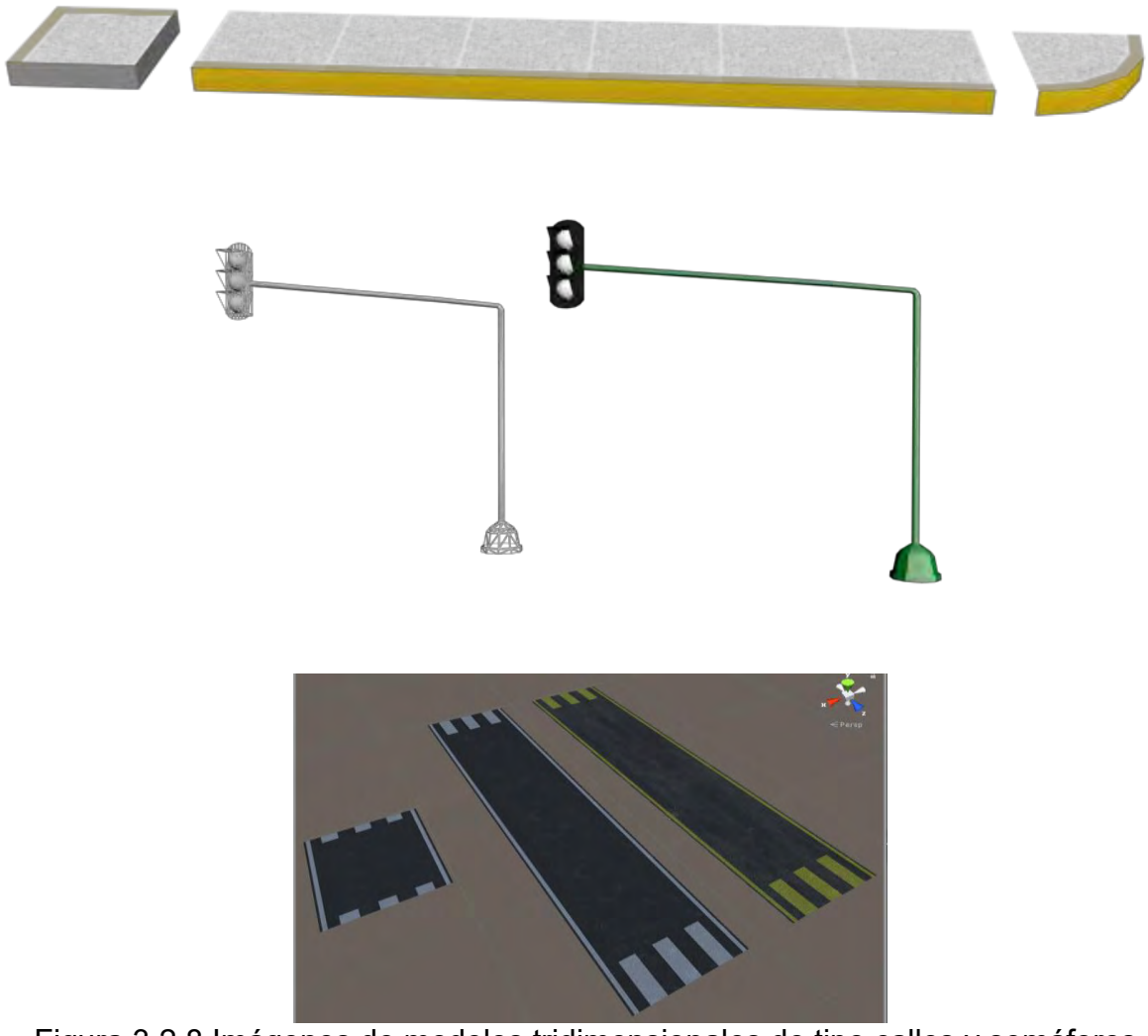


Figura 3.2.8 Imágenes de modelos tridimensionales de tipo calles y semáforos.

3.3 Etapa de programación

Esta etapa comprende la realización de varios códigos para el correcto funcionamiento del simulador. Se subdividió en submódulos, con el fin de trabajar varias tareas en paralelo y terminar en menor tiempo la realización del proyecto.

Para lograr la correcta integración de los submódulos, se establecen normas en la creación de los programas requeridos:

Se usará Unity como motor gráfico por lo tanto, cada submódulo debe trabajar en dicha plataforma, usando la misma versión del software, la cual será la 5.1.2, pudiendo actualizar a una versión más reciente, siempre y cuando se justifique adecuadamente la razón por la cual se actualiza de versión y no afecte dicha actualización a lo realizado hasta ese momento el proyecto.

Se usa C# como lenguaje de programación en todos los códigos generados para el proyecto.

Los objetivos se dividieron en subsistemas pequeños con la finalidad de trabajar en varios grupos de trabajo, para poder realizarlos en paralelo, de esta manera se busca tardar menos en la realización del proyecto.

Subsistema para controlar el movimiento del auto virtual por parte del usuario

Este subsistema busca crear la lógica de programación para que el usuario pueda controlar el movimiento del auto interactuando por medio del volante, palanca de velocidades y pedales Logitech modelo G27. Se programó un comportamiento lo más similar a como lo hacen los autos reales.

Se busca que el auto pueda moverse hacia adelante y en reversa haciendo uso de la selección con la palanca de velocidades. La palanca manda la información para saber la velocidad que fue seleccionada.

Debido al uso del casco de Realidad Virtual, Oculus Rift, no le es posible al usuario ver la posición de la palanca física, por esta razón únicamente con moverla hacia adelante o atrás cambiará la selección de velocidad, simulando el comportamiento de un auto con transmisión automática.

Para el control de la dirección de movimiento del vehículo el subsistema recibe valores del volante físico. Se reciben valores de tipo flotante entre -1 y 1, indicando la posición del volante, siendo -1 el giro a la izquierda y 1 giro a la derecha y 0 cuando el volante está en su posición inicial, es decir sin giro alguno.

Subsistema de vehículos autónomos

Este subsistema comprende el desarrollo de algoritmos de inteligencia artificial para el control de vehículos autónomos, capaz de simular un ambiente de tráfico real.

El subsistema se encarga del movimiento de vehículos para que tengan la capacidad de avanzar hacia adelante sobre la calle, girar en una esquina determinada, detenerse en caso de que haya otro auto adelante y detenerse en cruces cuando el semáforo esté en color rojo. Así como avanzar cuando el semáforo cambia a color verde.

Este subsistema se creó empleando el algoritmo de A estrella, este es un algoritmo de búsqueda, que se puede representar mediante el uso de grafos, sirve para encontrar la ruta más corta entre un punto A hasta un punto B. Su modo de funcionamiento es que evalúa el costo que tendrá el recorrer cada una de las rutas posibles entre cada nodo para llegar al punto B y recorre la ruta con el menor costo.

Tomando en cuenta la siguiente ecuación:

$$f(n) = g(n) + h'(n)$$

Básicamente consiste en llegar al punto destino desde el punto de origen respetando las restricciones que se asignan. El algoritmo automáticamente busca la ruta más corta para llegar al punto destino.

Las restricciones establecidas para el algoritmo son:

1. Que el vehículo sólo puede transitar sobre calles.
2. No puede avanzar si hay un semáforo en color rojo.
3. No puede avanzar si hay otro vehículo enfrente a corta distancia.
4. Debe respetar una velocidad de movimiento preestablecido.

Para lograr las restricciones es necesario implementar varios recursos al vehículo autónomo. Como el uso de ray tracing para saber cuándo hay un vehículo adelante.

Este funciona emitiendo un rayo que sale de la parte delantera del vehículo y va en dirección frontal del hasta que llega a una distancia establecida. Si el rayo impacta con otro objeto durante su trayectoria, el subsistema manda información del objeto con el que impactó. De esta manera podemos saber que el vehículo debe detenerse ya que hay otro enfrente.

Para lograr que el vehículo transite únicamente por las calles, es necesario emplear restricciones para el pathfinding. Esto se realiza mediante el uso de etiquetas, que se le dan a los modelos 3D, para identificar de qué tipo son y para que se usarán los modelos. Por ejemplo un modelo de una casa tendrá una etiqueta llamada “habitacional”, un local como “comercial”, de esta manera el algoritmo identificará las calles marcadas con la etiqueta “calle”. Posteriormente se le asigna el vehículo la restricción de que solamente podrá avanzar sobre modelos marcados con esa etiqueta.

Otra herramienta empleada es el uso de detectores de choques por medio de cajas de colisión. La manera en que funcionan es colocando una caja que cubre al vehículo lo más preciso posible. Esta caja visualmente no se aprecia para el usuario; Sin embargo, nos ayudará a detectar cuando el vehículo choca con algún objeto externo.

Por último se utilizó un motor de física para añadir el realismo de movimiento al vehículo. Esto quiere decir que se implementará una simulación de cómo funcionan las leyes de la física en la vida real, como por ejemplo la implementación de la fuerza de gravedad, la fricción y la aceleración.

Subsistema de control de semáforos.

Este subsistema comprende la realización de un algoritmo que sirva para simular el comportamiento de los semáforos.

Para lograr esto se plantearon varios objetivos que el sistema debe realizar para su correcto funcionamiento:

1. Controlar el cambio de luces emitidos por el semáforo.
2. Detectar si el auto controlado por el usuario pasa por un cruce con la luz roja en el semáforo.
3. Contribuir a que los vehículos autónomos se detengan con la luz roja del semáforo.

Para que el sistema realice el cambio de luz automáticamente, se hace uso de un contador, donde se establecen ciertos parámetros de tiempo iniciales antes de la simulación uno de esos parámetros controla el tiempo de duración que el semáforo tendrá para los estados verde, amarillo y rojo. Posteriormente el sistema se encarga de realizar el cambio de color cuando el tiempo del contador sea igual al tiempo preestablecido para cada color.

Para detectar si el usuario se pasa un cruce con la luz roja del semáforo se emplearon disparadores. Son cajas de colisión similares a que se emplean en el vehículo para detectar un choque. Sólo que en este caso es posible atravesar la caja de colisión. De esta manera este disparador se activará cada vez que el semáforo se encuentre en color rojo y se desactiva cada vez que el semáforo esté en verde o amarillo.

Para que los vehículos autónomos se detengan cuando la luz está en rojo. Se hace uso de una caja de colisión. Esta se coloca un poco antes de llegar al cruce para que el vehículo lo pueda detectar por medio del ray trace. De esta manera al ser detectado el vehículo lo interpreta como que hay un objeto al frente y se detendrá. Al cambiar de color el semáforo a verde se desactiva la caja de colisión.

Subsistema de monitoreo

Este subsistema sirve para tener registrado las infracciones que comete el usuario durante la simulación.

Se plantearon varios objetivos para su correcto funcionamiento:

1.- Obtener información de cada subsistema que tenga relación con las infracciones que se pueden cometer durante la simulación, se realizaron subsistemas para indicar si el usuario se pasó el alto, no usó direccionales para girar en la esquina o impactó con otro auto.

2.- Registrar el número de veces que se comente cada infracción.

3.- Mostrar un informe al usuario mostrando los resultados obtenidos de su simulación.

4.-Guardar en un archivo de texto el informe de los resultados obtenidos por cada usuario.

El control de infracciones obtiene información de los demás sistemas por medio de variables de tipo booleano, esta variable cambia a verdadero cuando se comete alguna infracción. El sistema guardará cada vez que cada bandera de los subsistemas está en verdadero para obtener el total de infracciones que se cometen en la simulación.

Al terminar la simulación se muestra sobre el parabrisas los resultados obtenidos por el usuario, mostrando el número de veces que cometió cada infracción y además el tiempo que tardó en completar la simulación.

El resultado obtenido en cada simulación es guardado en formato .txt. Este archivo guarda los resultados del usuario, la fecha y hora en la que se realizó.

3.4 Etapa de hardware

Se desarrollaron dos propuestas de plataformas

Plataformas

Con la intención de crear mayor sensación de realismo al usuario, se propuso construir una plataforma capaz de recrear los movimientos de un vehículo en movimiento, capaz de moverse en tiempo real en función de la manera de conducir del usuario. La función de la plataforma es que en ella se colocará un asiento donde se sienta el usuario.

Se buscó asesoría de alumnos de la carrera de Ingeniería en Mecatrónica, donde se logró llegar a dos propuestas de plataformas.

Para la primera propuesta:

Se realizaron tres actuadores de movimiento lineal (figura 3.4.1), los cuales son unos pistones mecánicos que internamente son accionados por un motorreductor que mueve un sedal de nylon que, con ayuda de poleas, arrastra a la barra que funge como pistón a distintas longitudes.

Estos actuadores se accionan de manera independiente y efectúan los distintos movimientos que el robot puede tener. Se añadieron dos rotulas de 3 grados de libertad a cada pistón, se le colocaron en ambos extremos las cuales permiten que los distintos elementos tengan libertad de movimiento sin colisionar o generar esfuerzos no deseados sobre la estructura. Estas rótulas se tuvieron que fabricar.

Figura 3.4.1 Actuadores de movimiento lineal.

Se diseñó un sistema de amortiguación (figura 3.4.2) para que los actuadores no se sobre-esfuerzan, evitando que carguen la totalidad del peso que se encuentre sobre la plataforma, así también, permite bloquear algunos de los grados de libertad de la plataforma que no se desean tener.

Este sistema hace uso de sistemas de ligas como elementos base para generar la amortiguación que, junto con una estructura a manera de compás, logra soportar posibles cargas sobre la plataforma.

Sobre la articulación de dicho compás se fabricó una junta tipo cruz, de 2 grados de libertad, para unirlo después a una plataforma (figura 3.4.3). Con esto es posible agregar peso sobre la estructura del robot y permite un buen movimiento de la misma sin tener repercusiones sobre los elementos que lo efectúan.

Figura 3.4.2 sistema de amortiguación.

Figura 3.4.3 Plataforma del sistema.

Se construyó una base (figura 3.4.4) para anclar los elementos antes mencionados y una plataforma que representa el efector final del simulador de movimiento vehicular y donde se ven reflejados los 3 grados de libertad mencionados al principio.

Figura 3.4.4 Base del sistema.

Para la segunda propuesta

Se construyó una plataforma Stewart.

En términos generales consiste en construir una plataforma que se puede manipular para moverse en 6 grados de libertad, los cuales consisten en trasladarse en los ejes x, y, z y permite rotar sobre cada uno de estos.

La conforma básicamente una plataforma superior hexagonal (figura 3.4.5), unida a una plataforma base, mediante 6 actuadores lineales.

La plataforma base sirve para soportar los actuadores y a la plataforma superior, además en el prototipo construido se colocó la tarjeta de control.

Cada actuador lineal está formado por un servomotor, conectado a una articulación y esta se conecta a la plataforma superior. Los actuadores son los encargados de generar los movimientos necesarios para llegar aúna posición y orientación de la plataforma deseados.

La plataforma superior está unida a las articulaciones. Es la encargada de simular los 6 grados de libertad. [15]

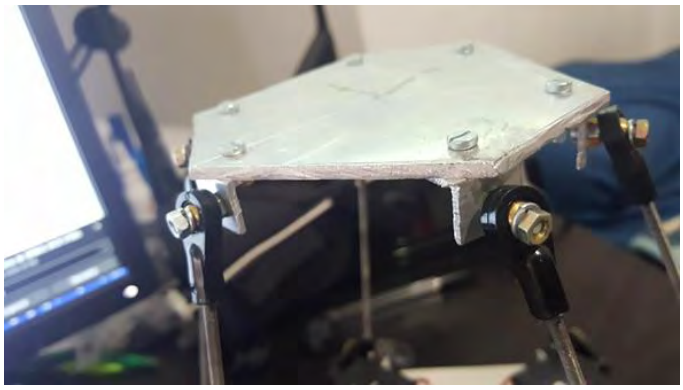


Figura 3.4.5 Fotografía de la parte superior de la plataforma.

Para controlar los movimientos de la plataforma, se usa una tarjeta de control arduino UNO (figura 3.4.6). Esta tarjeta nos permite mandar señales de tipo PWM, que son las necesarias para establecer la posición de los servomotores.

Figura 3.4.6 Fotografía de la tarjeta de control arduino conectado a los servomotores.

El sistema desarrollado se compone de una etapa de “potencia” (figura 3.4.7) donde alimentamos a los servomotores de manera externa con un voltaje y amperaje ideal de 5Volts y 4 Amperes.

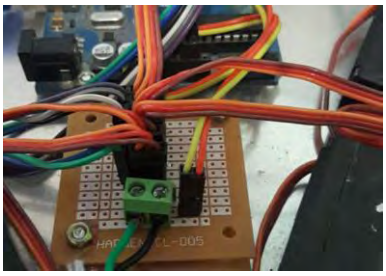


Figura 3.4.7 Fotografía de la tarjeta de potencia para alimentar de corriente eléctrica a los servomotores.

Para realizar el movimiento de la plataforma (figura 3.4.8), se hace mediante una cinemática inversa, esta establece que debemos elegir la posición y orientación que deseamos de la plataforma y se programó en la tarjeta de control una serie de movimientos que los actuadores deben realizar para llegar a esa posición y orientación.

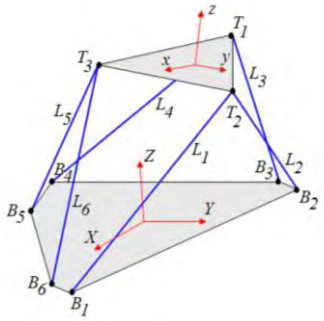


Figura 3.4.8
Geometría de la plataforma Stewart

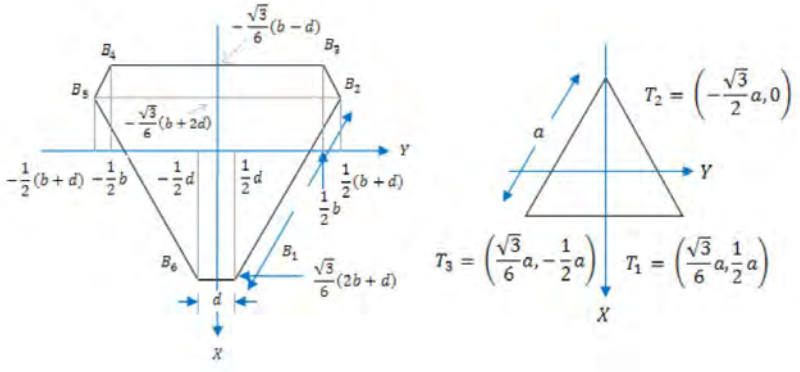


Figura 3.4.9
Dimensiones físicas de la plataforma

$$L_1 = \sqrt{\left(X_{T1} - \frac{d}{2\sqrt{3}} - \frac{b}{\sqrt{3}}\right)^2 + \left(Y_{T1} - \frac{d}{2}\right)^2 + Z_{T1}^2}$$

$$L_2 = \sqrt{\left(X_{T1} - \frac{d}{2\sqrt{3}} + \frac{b}{2\sqrt{3}}\right)^2 + \left(Y_{T1} - \frac{d}{2} - \frac{b}{2}\right)^2 + Z_{T1}^2}$$

$$L_3 = \sqrt{\left(X_{T2} + \frac{d}{\sqrt{3}} + \frac{b}{2\sqrt{3}}\right)^2 + \left(Y_{T2} - \frac{b}{2}\right)^2 + Z_{T2}^2}$$

$$L_4 = \sqrt{\left(X_{T2} + \frac{d}{\sqrt{3}} + \frac{b}{2\sqrt{3}}\right)^2 + \left(Y_{T2} + \frac{b}{2}\right)^2 + Z_{T2}^2}$$

$$L_5 = \sqrt{\left(X_{T3} - \frac{d}{2\sqrt{3}} + \frac{b}{2\sqrt{3}}\right)^2 + \left(Y_{T3} + \frac{b}{2} + \frac{d}{2}\right)^2 + Z_{T3}^2}$$

$$L_6 = \sqrt{\left(X_{T3} - \frac{d}{2\sqrt{3}} - \frac{b}{\sqrt{3}}\right)^2 + \left(Y_{T3} + \frac{d}{2}\right)^2 + Z_{T3}^2}$$

Con estas 6 ecuaciones se obtiene la solución de la cinemática inversa de la Plataforma Stewart

Los valores posición y orientación son recibidos a la tarjeta de control mediante una conexión tipo USB y los datos son recibidos de manera serial.

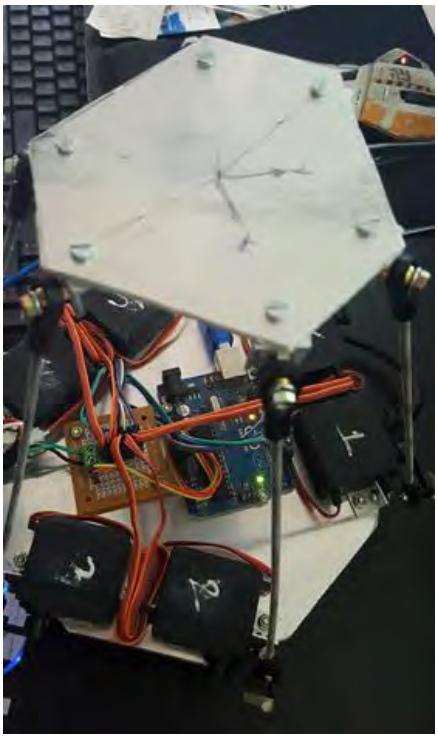


Figura 3.4.10 Fotografía de la plataforma construida.

3.5 Etapa de Integración

Esta etapa se enfoca de juntar el trabajo de las demás etapas, supervisando que cada uno funcione correctamente, primero cada uno por separado y posteriormente cada uno junto con los otros.

Integración del volante al simulador

Primeramente se comenzó integrando el volante, palanca y pedales, al motor gráfico, para que el auto virtual pueda ser controlado por el usuario.

La integración se realizó con un plugin externo, llamado “Rewire”. Esto se hizo con la finalidad de ahorrar tiempo y así enfocarnos en la realización del sistema completo.

El funcionamiento del plugin (figura 3.5.1) es muy sencillo, solo se debe seleccionar el modelo del volante que se ocupará, y los valores que se quieren obtener del hardware para posteriormente manipular esos valores para controlar al auto.

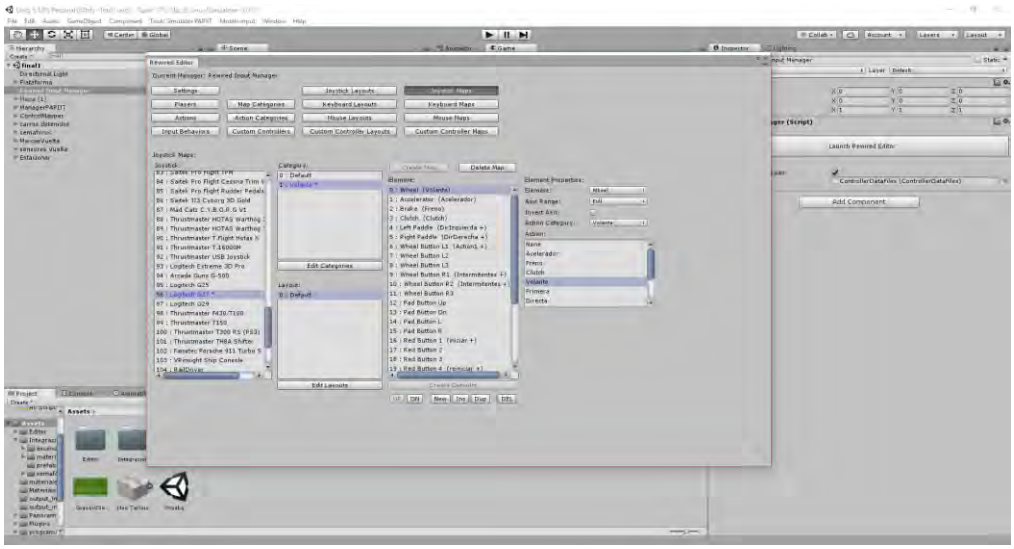


Figura 3.5.1 Imagen que muestra la configuración para el uso del volante, palanca y pedales dentro del software Unity.

Integración de los modelos 3D al motor de render Unity

Antes de integrar los modelos 3D fue necesario revisar cada uno de ellos para corroborar que cumpliera con las normas establecidas.

Los modelos para la ciudad se agruparon en cuadras, cada cuadra tiene el mismo tamaño.

Se realizaron varios tipos de cuadras:

Cuadras de tipo habitacional (figura 3.5.2).

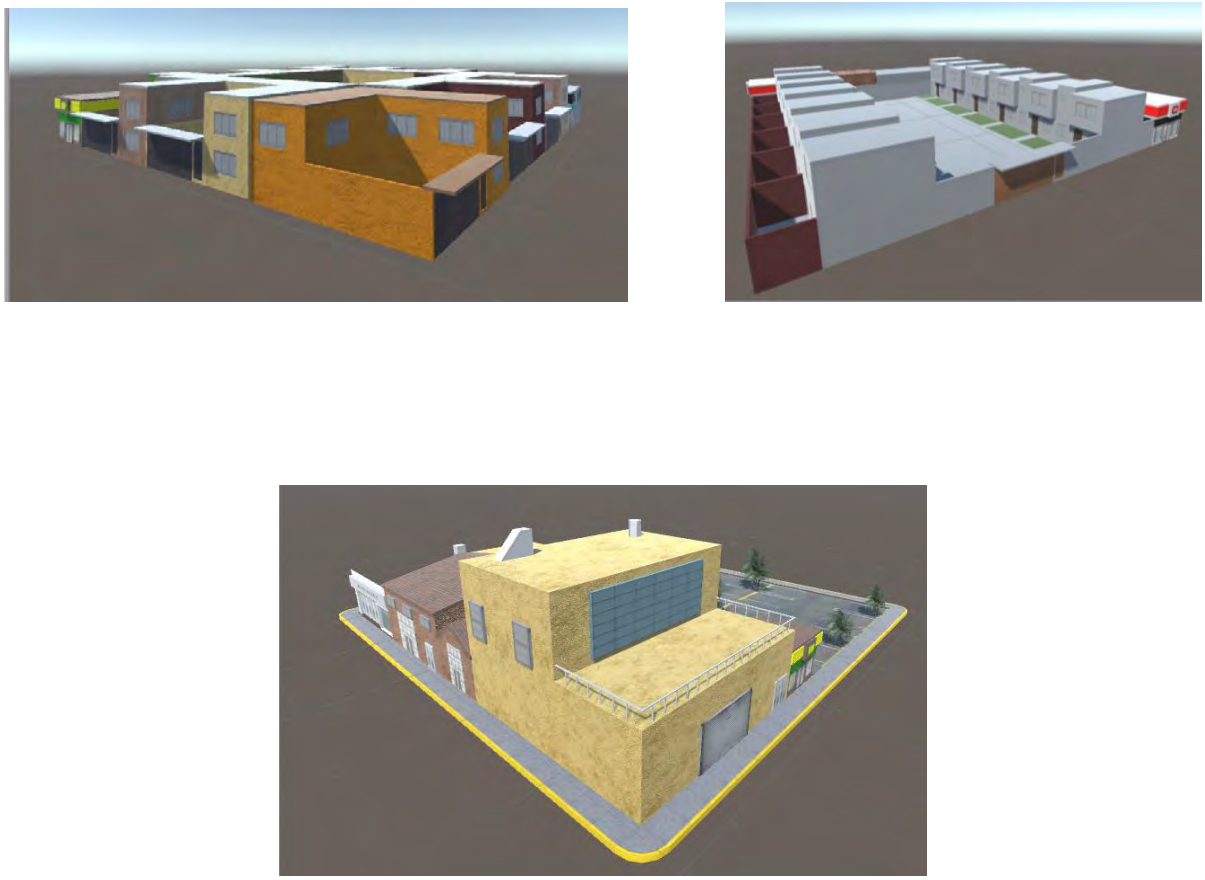


Figura 3.5.2 Imágenes de algunas cuadras generadas de tipo habitacional.

Cuadras con edificios (figura 3.5.3)

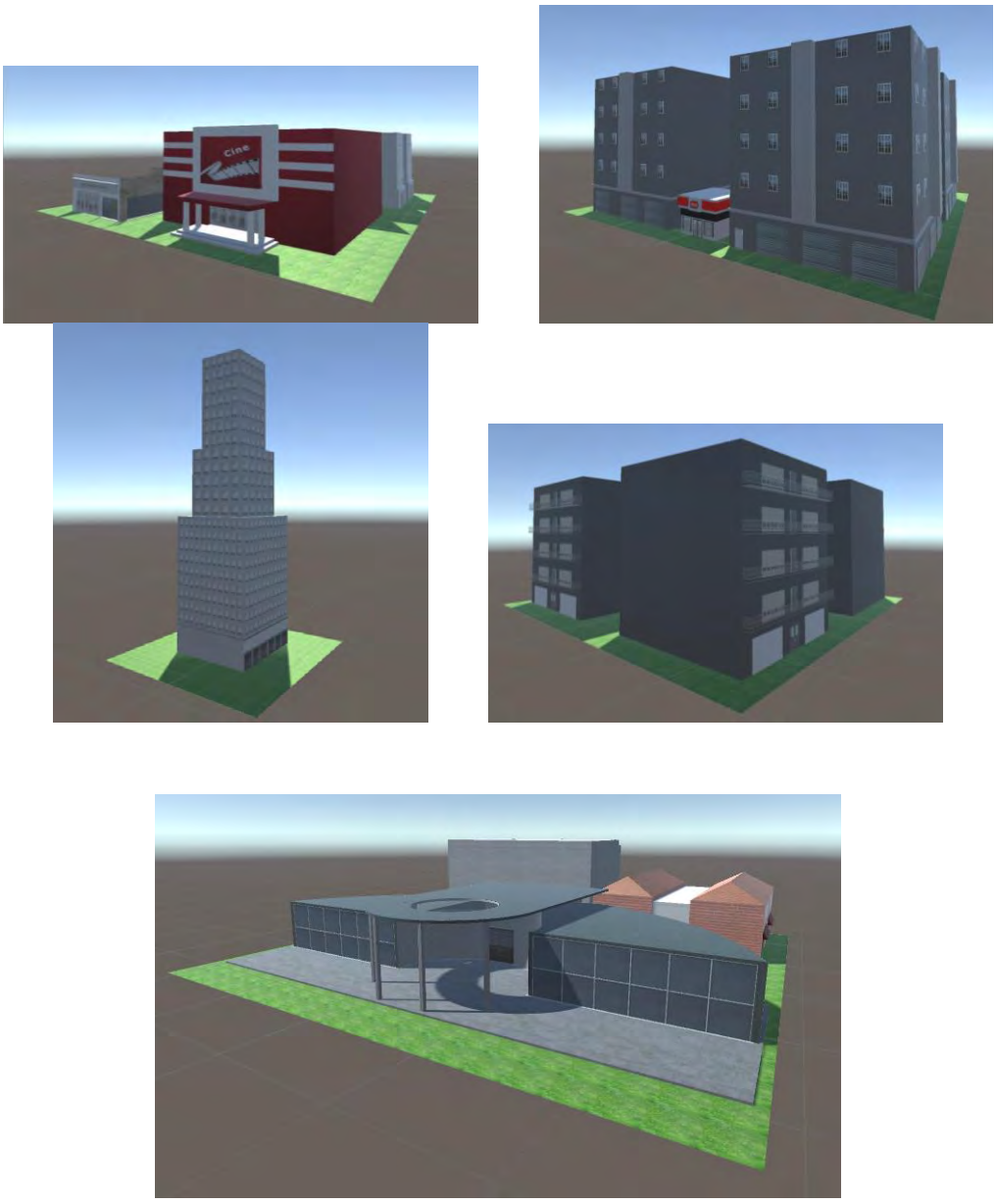


Figura 3.5.3 Imágenes de algunas cuadras generadas de tipo edificio.

Cuadras con locales comerciales (figura 3.5.4).

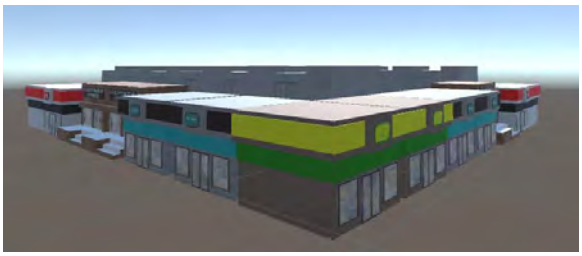


Figura 3.5.4 Imágenes de cuadras generadas de tipo comercial.

Figura 3.5.5 Imagen que muestra la configuración del componente Semi Track Script.

Cada calle tiene un componente llamado Semi Track Script (figura 3.5.5), el cual nos sirve para configurar la velocidad a los que los autos autónomos pueden avanzar, y saber cuáles son las calles que se encuentran al cerca para que el auto se pueda mover y además se les añadió el layer “SemiTrack”.

Posteriormente se integraron las calles (figura 3.5.6) y banquetas de manera que se dejó el espacio para acomodar las cuadras.

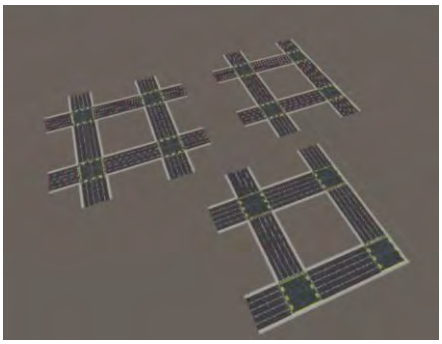


Figura 3.5.6 Imagen de los diferentes tipos de calles generado.

Se realizaron 3 tipos diferentes de calles (figura 3.5.6), para cerrar por completo la ciudad.

Se realizaron bloques de nueve cuadras cada uno (figura 3.5.7), realizando variantes con la cantidad de cuadras realizadas.

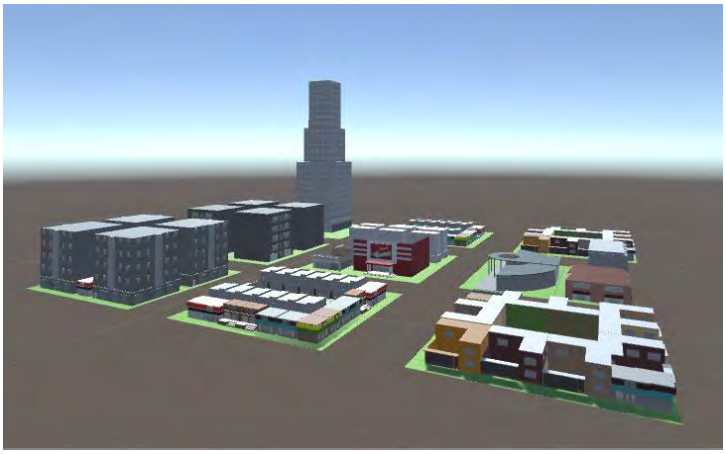


Figura 3.5.7 Imagen de uno de los bloques generado con nueve cuadras distintas.

Finalmente se colocaron los semáforos en los cruces de cada calle donde se realizará la simulación y se juntaron los bloques de cuadras para formar el mapa completo (figura 3.5.8).

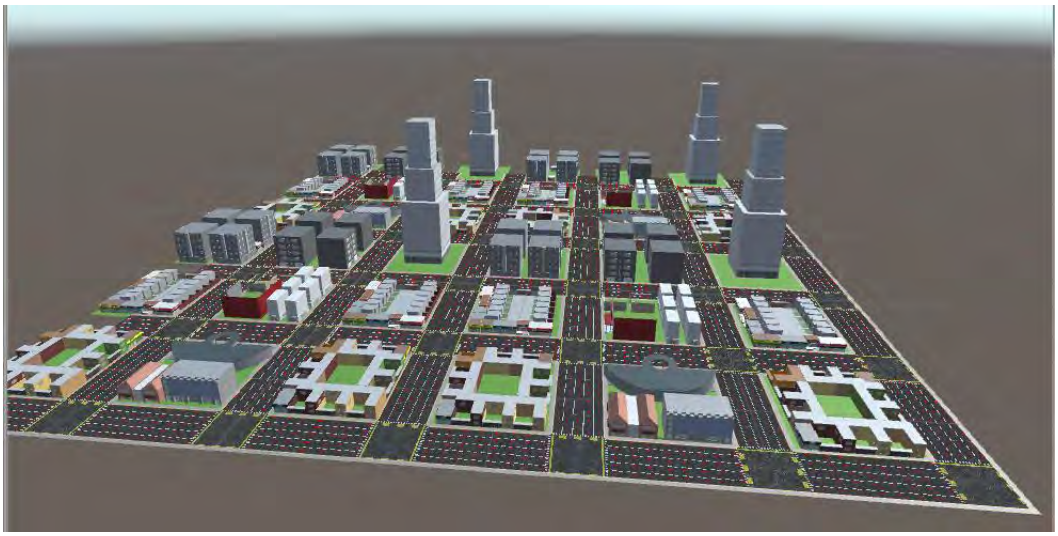


Figura 3.5.8 Imagen del mapa realizado para la simulación.

Una vez formada la ciudad donde se realizará la prueba de manejo, se agregaron componentes necesarios para que los algoritmos realizados en el módulo de programación funcionen adecuadamente.

Primero se agregó un objeto llamado “ManagerPAPIIT” con los siguientes componentes:

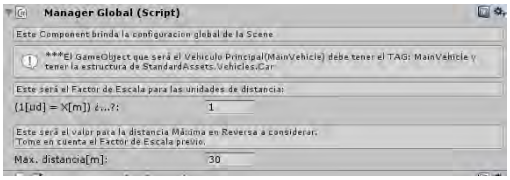


Figura 3.5.8 Imagen de configuración del componente Manager Global.

El componente “Manager Global” (figura 3.5.8), sirve para configurar el factor de escala en la que se colocan los modelos 3D, con este factor podemos saber la distancia a la que se encuentra un modelo y otro, siendo cada unidad equivale a un metro.

Posteriormente se configura el valor de la distancia máxima que estará permitido avanzar en reversa, esto para saber si se comete una infracción al superar dicho parámetro.

Figura 3.5.9 Imagen de configuración del componente Manager Penalty.

El componente “Manager Penalty” (figura 3.5.9) sirve para mostrar las infracciones cometidas durante la simulación al usuario y posteriormente guardar sus resultados en un archivo de texto.

Para su correcto funcionamiento se debe configurar un canvas donde se mostrarán las infracciones cometidas y se configura el nombre de la carpeta donde se guardan los resultados obtenidos en archivos separados.

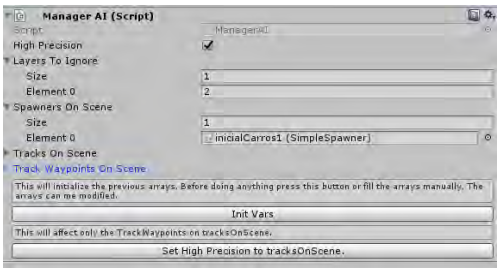


Figura 3.5.10 Imagen de configuración del componente Manager AI.

El componente “Manager AI” (figura 3.5.10) sirve para que los autos que se moverán automáticamente puedan detectar cuáles modelos de la ciudad son calles por donde pueden avanzar y no otro tipo de modelo como una casa, comercio, banqueta o auto estacionado.

Para configurarlo se debe seleccionar el layer donde se encuentran las calles llamado “SemiTrack”, posteriormente se añade el objeto donde se encuentra el inicio de los carros autónomos, para establecer el punto de salida de estos.

Una vez colocados los componentes necesarios el componente reconocerá las calles y le mandará esta información a los autos.

Se integraron los modelos de autos al algoritmo para que se muevan automáticamente.

Para lograr esto se deben colocar varios parámetros necesarios para que el algoritmo funcione, descritos a continuación:

Figura 3.5.11 Imagen de configuración del componente Manager AI Car Controller.

El componente “Car Controller” (figura 3.5.11) sirve para controlar el movimiento del auto lo más realista posible simulando el manejo de un auto. En el componente se deben agregar los objetos que contienen al modelo del auto, las llantas, la carrocería y sus modelos de colisión correspondientes.

Los parámetros que se configuran en el componente son la velocidad máxima a la que se moverá el auto, el torque que tendrán las ruedas, la posición donde se encuentra el centro de masa del modelo y las unidades que se usarán, seleccionando kilómetros por hora.

Figura 3.5.12 Imagen de configuración del componente Car Audio.

El componente “Car Audio” (figura 3.5.12) sirve para agregar los efectos de sonido al auto, con esto se le da mayor realismo, simulando los sonidos de un auto cuando acelera o frena. Lo único que se debe configurar es agregar los sonidos al componente y configurar los parámetros acerca los límites máximos y mínimos a los que queremos que se reproduzca el sonido.

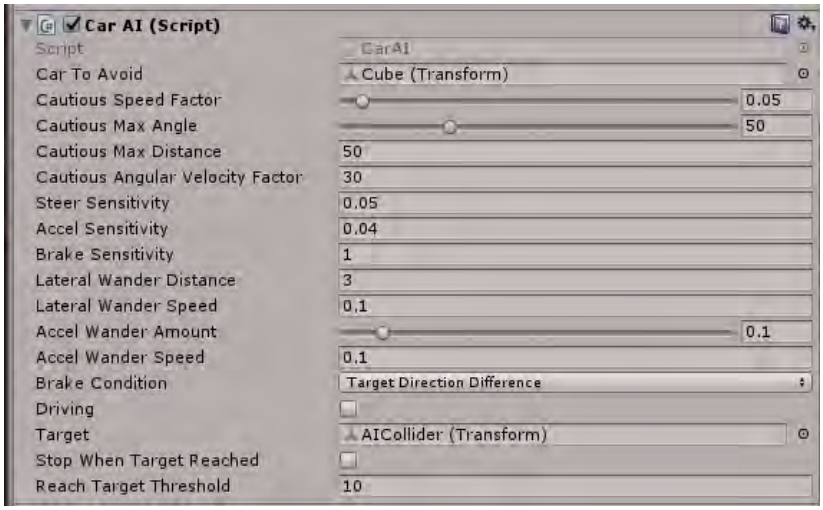


Figura 3.5.13 Imagen de configuración del componente Car AI.

El componente “Car AI” (figura 3.5.13) sirve para controlar la trayectoria de cada auto, respetando las restricciones previamente establecidas. Lo que hay que configurar son los parámetros de sensibilidad que tiene el auto para la detección de objetos.

Una vez que se configuró la ciudad y los autos adecuadamente se procedió a marcar la ruta que deben seguir los autos, colocando componentes adicionales a las calles por las cuales se desea que el auto pase.

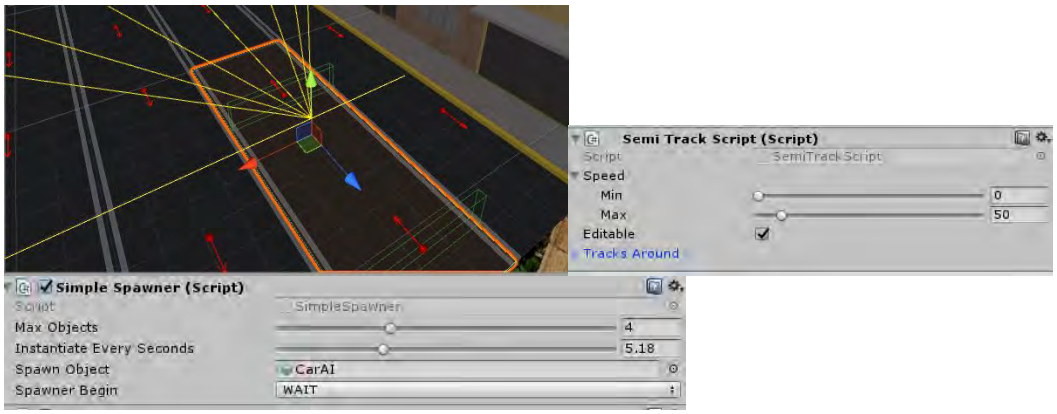


Figura 3.5.14 Imagen de configuración del componente Semi Track Script y Simple Spawner.

Adicionalmente al componente “Semi Track Script” (figura 3.5.14) se debe colocar el componente “Simple Spawner” (figura 3.5.14), este componente sirve para colocar los autos automáticamente en el escenario. Para que funcione se debe configurar el objeto que va a salir, en este caso se coloca el auto, posteriormente se configura el número máximo de autos que saldrán en total y cuánto tiempo aparecerán los autos.

Figura 3.5.15 Imagen de configuración del componente Track Waypoint.

Otro componente que se debe añadir es el componente “Track Waypoint” (figura 3.5.15), el cual nos sirve para indicar el siguiente punto ubicado en la calle que debe seguir el auto. Solo se añade el siguiente objeto de calle para que funcione.

Una vez agregados y configurados los componentes mencionados el algoritmo establece una ruta la cual los autos deberán seguir (figura 3.5.16).

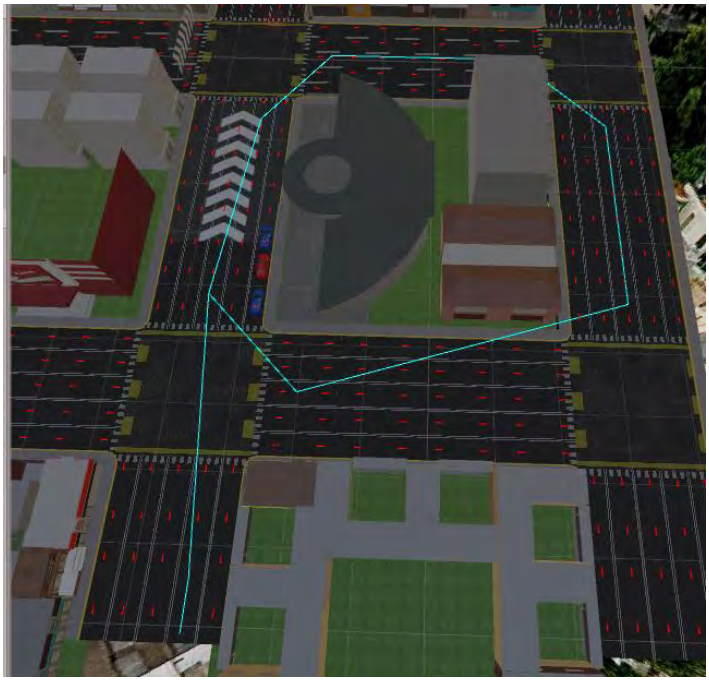


Figura 3.5.16 Imagen que muestra la ruta que algunos autos deberán seguir durante la simulación.

Para la integración del automóvil que será manipulado por el usuario, se implementó el código realizado en el módulo de programación con el auto creado (figura 3.5.17) en la etapa de modelado.

Figura 3.5.17 Imagen del auto tridimensional en el software Unity.

Se añadieron sonidos (figura 3.5.18) conforme a los que realiza un auto al avanzar y al frenar. Además se utilizó el motor de física que tiene el programa Unity, para añadir el realismo de fuerzas que intervienen como la gravedad, la aceleración al avanzar y la fuerza de fricción que hay en el piso con las ruedas del auto.

Figura 3.5.18 Imagen del componente utilizado para implementar efectos de sonido a la simulación.

Se añadió el modelo de colisión de tipo caja (figura 3.5.19), con la finalidad de detectar si choca con otros autos, ya sean estacionados o los que se mueven automáticamente. Se eligió el modelo de tipo caja con la finalidad de optimizar recursos de la unidad de procesamiento.

Figura 3.5.19 Imagen que muestra el modelo de colisión tipo caja sobre el auto.

Se incorporó la interacción mediante el volante, palanca y pedales (figura 3.5.20). Para los pedales únicamente funciona en esta versión del simulador el acelerador para avanzar y el freno para detenerse, la palanca sirve para seleccionar si el vehículo avanzará o irá en reversa, siendo adelante para avanzar y atrás para ir en reversa, la posición del centro será para tener el auto estacionado.

Figura 3.5.20 Imagen que muestra el interior del auto modelado.

La interacción que se agregó con el volante Logitech, además de elegir la dirección de las llanta delanteras, es usar las luces direccionales con las palancas que se encuentran detrás del volante, las luces direccionales se desactivan cuando el volante gira en sentido donde se colocaron como izquierda o derecha y regresa a su posición original. El volante cuenta con botones al frente con los que se puede activar o desactivar las luces intermitentes.

Figura 3.5.21 Imagen de configuración del componente Car User Control

El componente “Car User Control” (figura 3.5.21) sirve para encender las luces intermitentes y direccionales del auto, para que funcione solo basta con colocar el componente al auto.

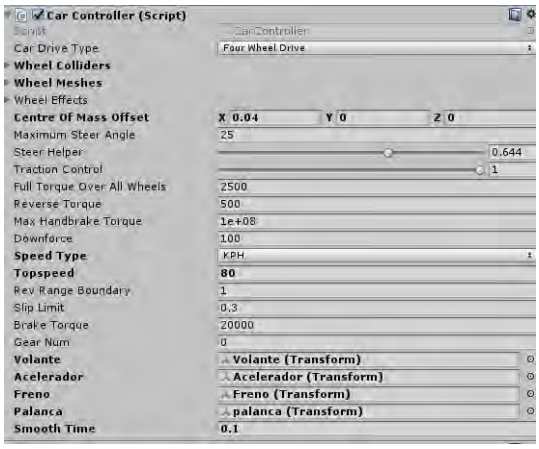


Figura 3.5.22 Imagen de configuración del componente Car Controller.

El componente “Car Controller” (figura 3.5.22) sirve para poder controlar el auto por el usuario, por medio del volante y pedales físicos, así como para establecer los límites que se considerarán como la velocidad máxima que tendrá el auto al avanzar y la fuerza de torque que tendrán las llantas.

Figura 3.5.23 Imagen de configuración del componente Velocímetro.

El componente “Velocimetro” (figura 3.5.23) sirve para mostrar información de la velocidad a la que va el usuario y las revoluciones por minuto.

Para lograr su correcto funcionamiento se debe agregar al componente los objetos del tacómetro, el objeto del carro y el modelo de colisión de una llanta del auto.

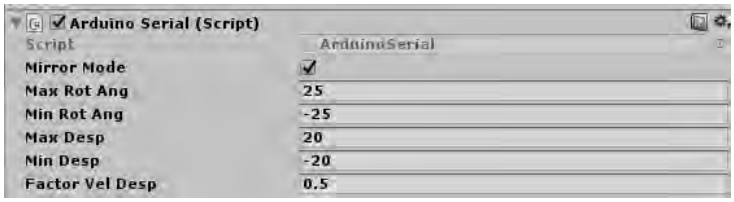


Figura 3.5.24 Imagen de configuración del componente Arduino Serial.

El componente “Arduino Serial” (Figura 3.5.24) sirve para establecer una conexión con la tarjeta de control Arduino y mandar las señales necesarias para mover la plataforma. Se deben configurar el parámetro máximos y mínimos que se ocupan durante la simulación, para su correcto funcionamiento.

Se colocaron modelos de autos detenidos (figura 3.5.25) en el inicio y fin de la prueba:

Figura 3.5.25 Imagen del modelo de tipo auto estacionado en el software Unity.

Para su correcto funcionamiento, únicamente se le añadió un componente de tipo Box collider, para poder detectar si choca con el auto controlado por el usuario.

Se añadieron los modelos 3D de semáforos (figura 3.5.26), y los componentes realizados en el módulo de programación descritos a continuación:

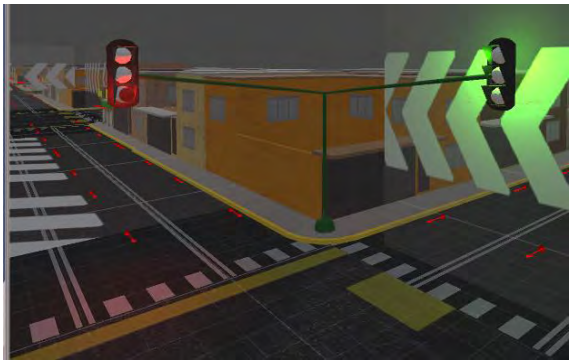


Figura 3.5.26 Imagen de los modelos de tipo semáforos implementados en el software Unity.

Cada cruce del recorrido por el que tiene que pasar el usuario durante el recorrido está compuesto de 2 semáforos.

Cada semáforo está compuesto por un objeto llamado “luz” (figura 3.5.27), este objeto es una esfera con un material blanco.

Figura 3.5.27 Imagen que muestra los estados por los que pasa el componente luz para cambiar el color del semáforo.

Este objeto cambiará de posición y de color para establecer el color del semáforo que tiene actualmente.

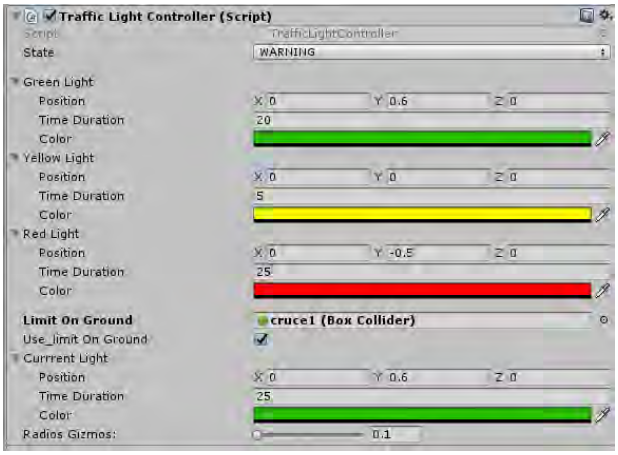


Figura 3.5.28 Imagen de configuración del componente Traffic light controller.

El componente “Traffic light controller” (figura 3.5.28) sirve para seleccionar el color y la posición que tendrá el objeto “luz” y se establece cuánto tiempo durará cada color en el semáforo, así como también cuál será el primer color que aparecerá durante la simulación.

Posteriormente se colocaron planos en algunos cruces y sobre el pavimento con una textura de flechas blancas (figura 3.5.29) para indicar el recorrido por el que debe pasar el usuario.

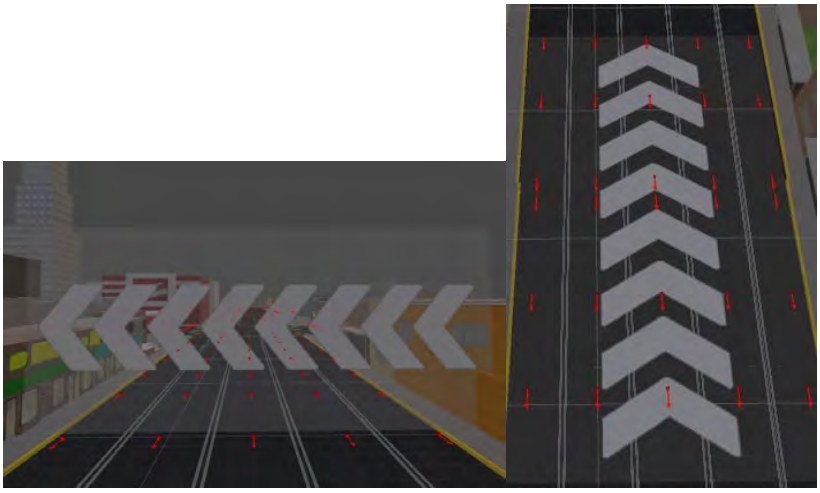


Figura 3.5.29 Imágenes de las flechas texturizadas sobre planos para indicar el camino a seguir durante la simulación.

Además se les agregó una animación para mover la textura en dirección hacia donde apunta la flecha.

Para concluir el módulo de integración se colocó un área establecida para que el usuario termine su prueba estacionando el auto en dicha área (figura 3.5.30).

Figura 3.5.30 Imagen que muestra la zona donde el usuario debe estacionarse durante la simulación.

El área está marcada con una señal representada por un modelo 3D de estacionarse, y con un recuadro punteado en color amarillo, además se colocaron 2 modelos de autos 3D estacionados al frente y atrás del área (figura 3.5.31).

Figura 3.5.31 Imagen que muestra los elementos alrededor de la zona para estacionarse durante la simulación.

Para que el sistema detecte que el auto del usuario está el área marcada se colocó una caja de colisión como activador (figura 3.5.32).

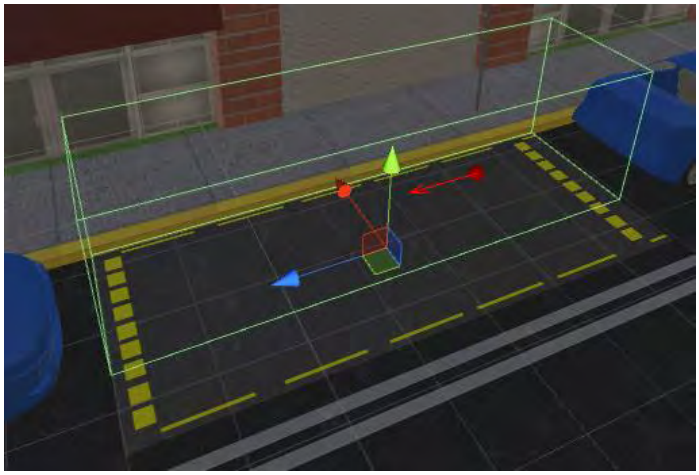


Figura 3.5.32 Imagen que muestra la caja de colisión implementada para detectar cuando el usuario se estacionó.

3.6 Etapa de pruebas

Este módulo comprende al proceso de pruebas que se llevaron a cabo durante la integración de cada uno de los sub-sistemas generados con el motor gráfico, así como a los errores que se generaron al realizar dicha integración.

Pruebas al integrar el volante Logitech G27 con Unity:

Al integrar el volante G27 con el motor Unity, funcionó bien inicialmente, configurando los parámetros necesarios. Se realizaron pruebas de movimiento de todos los componentes con los que el usuario puede interactuar.

Se comenzó probando el uso de los pedales, se revisó que cuando se presione alguno, mande correctamente la información y, el motor Unity, reciba adecuadamente los datos. Siendo en el caso de cada uno de los pedales un rango de tipo flotante para cada uno de ellos, que va desde cero hasta uno, cero indica que no está presionando y uno indica que se presionó el pedal completamente.

Posteriormente se realizaron pruebas para el correcto funcionamiento del volante, una vez configurado los parámetros necesarios, se revisó que se mande correctamente la información y que el motor Unity la reciba adecuadamente. Para el volante se revisó que se recibiera un rango de datos de tipo flotante que va desde menos uno a uno [-1, 1], siendo menos uno cuando el volante gira por completo a la izquierda, cero cuando el volante está en posición central y uno cuando el volante gira por completo a la derecha.

Para revisar el correcto funcionamiento de la palanca de velocidades, se corroboró que se mande correctamente la información al seleccionar una velocidad en la palanca y que posteriormente se reciba la posición de la palanca, recibiendo este parámetro como una posición y el motor Unity la interpreta como una variable booleana, marcando como true el valor de la posición donde está la palanca.

Para corroborar el buen funcionamiento de los botones que tiene el volante (figura 3.6.1), se revisó que se mandará de manera correcta la información y que el motor Unity la reciba adecuadamente, marcando como true la variable de tipo booleana en el botón presionado.

Figura 3.6.1 Imagen que muestra el funcionamiento de cada elemento del volante, palanca y pedales dentro de la simulación,

Las pruebas realizadas con la integración del módulo de modelado 3D fueron en su mayoría visuales, revisando que cada modelo se encontrará correctamente texturizado, que la escala de los modelos fuera la correcta y que cada modelo cuente con la cantidad mínima de polígonos posible.

Algunas de las pruebas que se realizaron con el módulo de programación fueron previas a la integración, las cuales consistieron en pruebas de escritorio para cada subsistema, después se verificó su correcta compilación y ejecución. Posteriormente al integrar únicamente se probó su correcta compilación y ejecución de cada subsistema dentro del sistema principal.

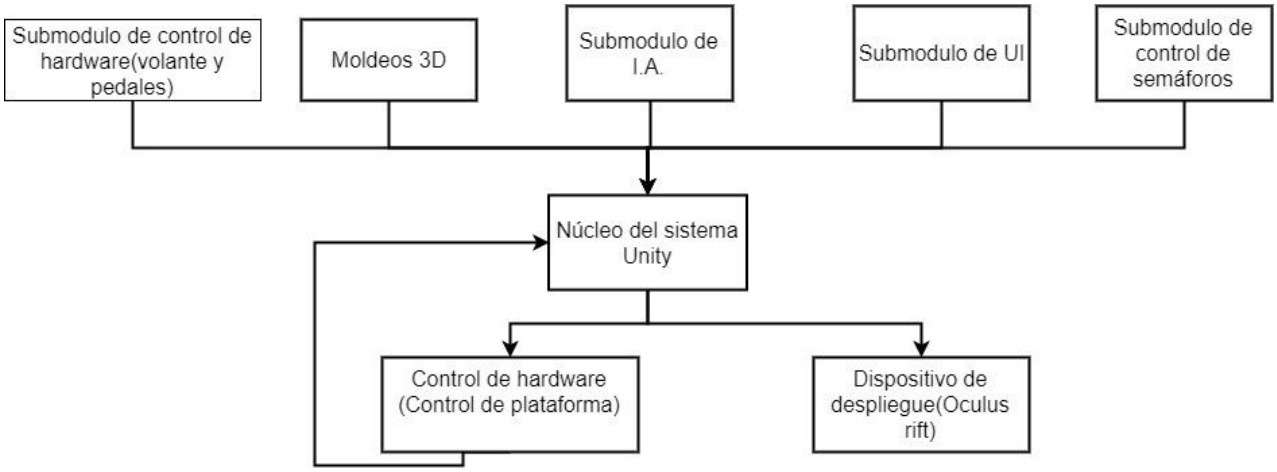


Figura 3.6.2 Diagrama de la conexión general del sistema para realizar la simulación.

4.- Análisis de resultados

4.1 Descripción de pruebas realizadas

La prueba comienza estando estacionado en el lado derecho de la calle teniendo al frente y atrás vehículos estacionados, se espera que el usuario se familiarice con el entorno, el usuario debe incorporarse a la circulación.

Posteriormente debe seguir una ruta establecida (figura 4.1.1) dividida en varios puntos de control definidos en cada zona donde el usuario debe pasar, marcada por flechas de color blanco, durante este trayecto deberá respetar los semáforos, el uso de direccionales para incorporarse a otras calles.

La última parte de la prueba consiste en estacionar el vehículo en el lado extremo de la banqueta, en un espacio ubicado entre 2 vehículos.

Durante el trayecto el usuario debe pasar por 4 cruces con semáforo,

Puntos de control de la prueba:

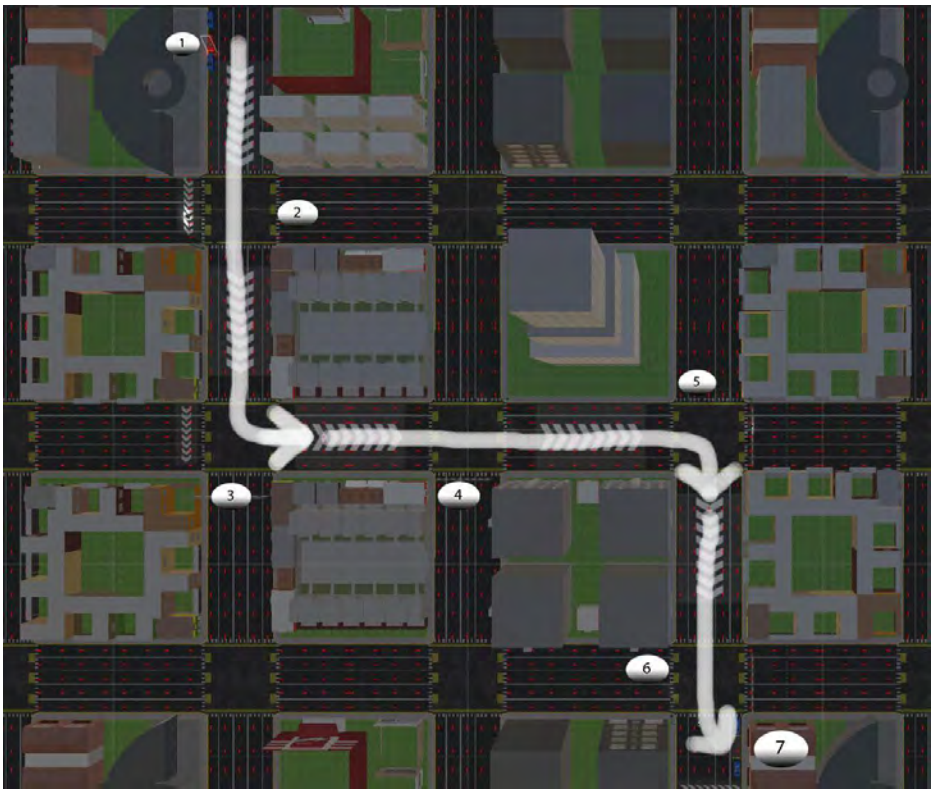


Figura 4.1.1 Imagen de la ruta establecida que el usuario debe seguir durante la simulación.

Descripción de los puntos de control:

1.- Es el inicio de la prueba donde el usuario está estacionado entre dos carros (figura 4.1.2), el objetivo a evaluar es que se incorpore a la circulación.

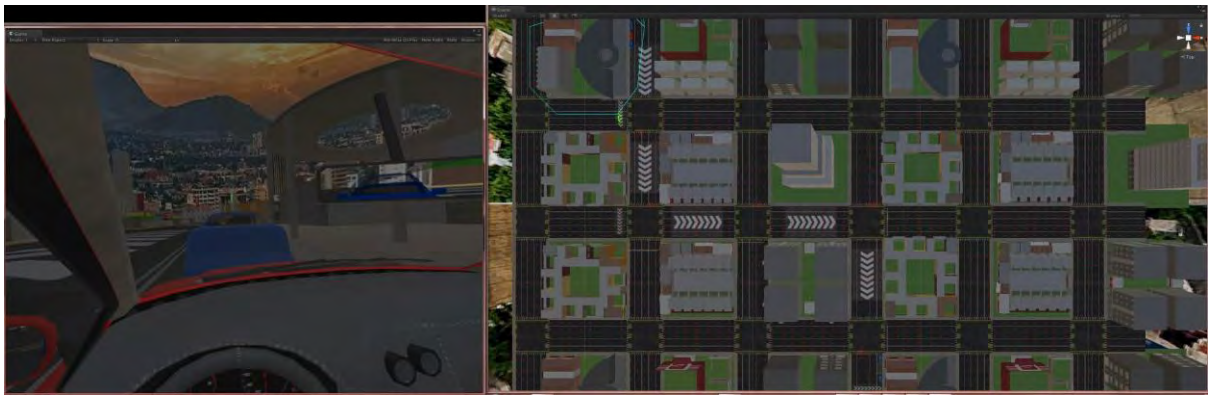


Figura 4.1.2 Imagen de una vista del usuario y una vista aérea en el primer punto de control.

2.- En este punto se evalúa que el usuario respete la señal del semáforo para pasar el cruce (figura 4.1.1).

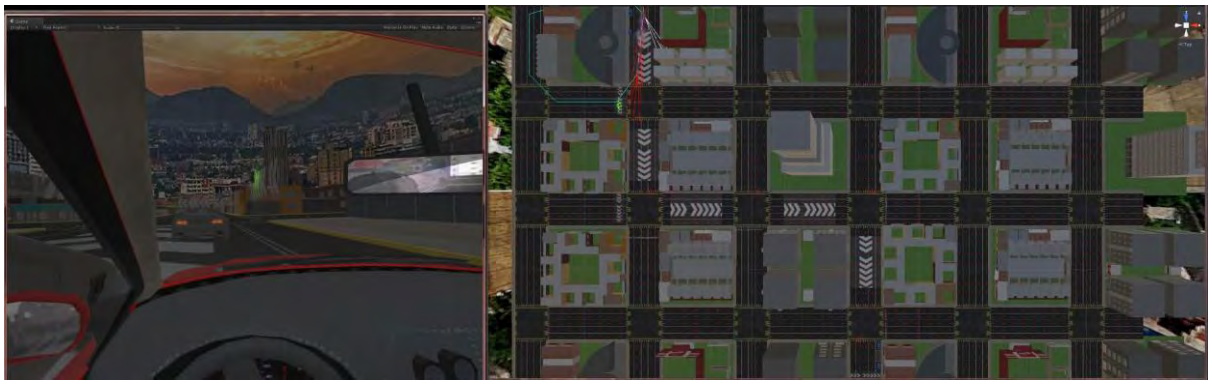


Figura 4.1.1 Imagen de una vista del usuario y una vista aérea en el segundo punto de control.

3.-En este punto se evalúa que el usuario respete la señal del semáforo para pasar el cruce y que haga uso de las luces direccionales para girar a la izquierda y continuar la ruta (figura 4.1.2).

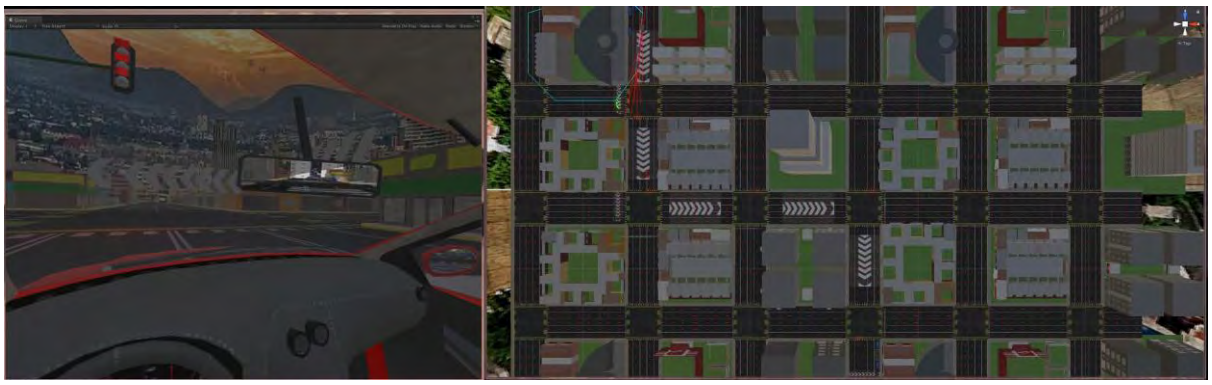


Figura 4.1.2 Imagen de una vista del usuario y una vista aérea en el tercer punto de control.

4.-En este punto se evalúa que el usuario respete la señal del semáforo para pasar el cruce (figura 4.1.3).

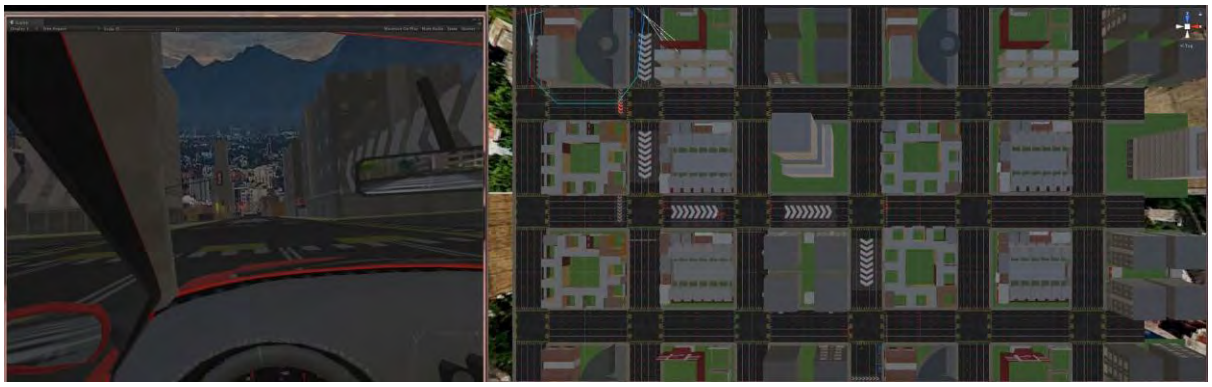


Figura 4.1.3 Imagen de una vista del usuario y una vista aérea en el cuarto punto de control.

5.-En este punto se evalúa que el usuario respete la señal del semáforo para pasar el cruce y que haga uso de las luces direccionales para girar a la derecha y continuar la ruta (figura 4.1.4).

Figura 4.1.4 Imagen de una vista del usuario y una vista aérea en el quinto punto de control.

6.-En este punto se evalúa que el usuario respete la señal del semáforo para pasar el cruce (figura 4.1.5).

Figura 4.1.5 Imagen de una vista del usuario y una vista aérea en el sexto punto de control.

7.-En este punto el usuario debe estacionarse entre 2 carros a un costado de la banqueta, una vez que se estaciona se muestran los resultados obtenidos durante la prueba (figura 4.1.6).

Figura 4.1.6 Imagen de una vista del usuario y una vista aérea en el séptimo punto de control.

Evaluación.
Una vez que el usuario se estaciona aparece una pantalla con los resultados obtenidos (figura 4.1.7)

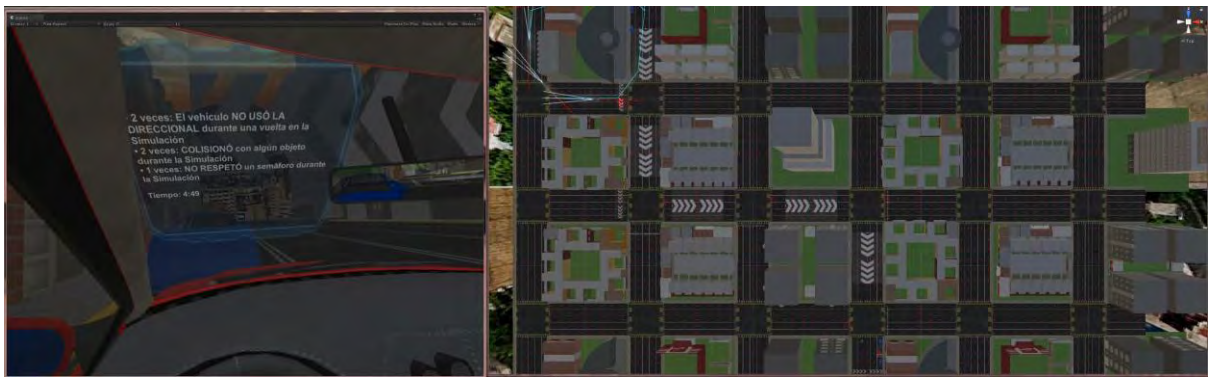


Figura 4.1.7 Imagen de una vista del usuario y una vista aérea al final de la prueba.

4.2 Resultados a las pruebas realizadas

Se hizo la propuesta de tener 2 grupos de prueba a los cuales se les permitirá el uso del simulador, cada grupo cumplirá con ciertas características:

Grupo 1:

- Gente habituada al uso de tecnología y con nociones de realidad virtual.

Se propone alumnos de la Facultad de Ingeniería de la carrera de Ingeniería en Computación, de últimos semestres.

Grupo 2:

- Gente que no sepa en lo absoluto acerca de realidad virtual.

Ambos grupos que cuenten con licencia para conducir.

En la fase de pruebas nos percatamos que se le debe permitir al usuario varios intentos para acostumbrarse tanto al uso del casco de realidad virtual como al uso del volante, palanca de velocidades y pedales que simulan al vehículo, ya que es una nueva sensación y, sobre todo para la gente alejada del uso de tecnologías de cómputo es difícil ubicar cada uno de los elementos que intervienen en la toma de decisiones para el correcto manejo del simulador. Fue de especial mención el uso de la palanca para el uso de las intermitentes que marcan el cambio de carril y/o dirección.

- Estadísticas obtenidas de las prácticas de manejo de la gente que utilizó el sistema:

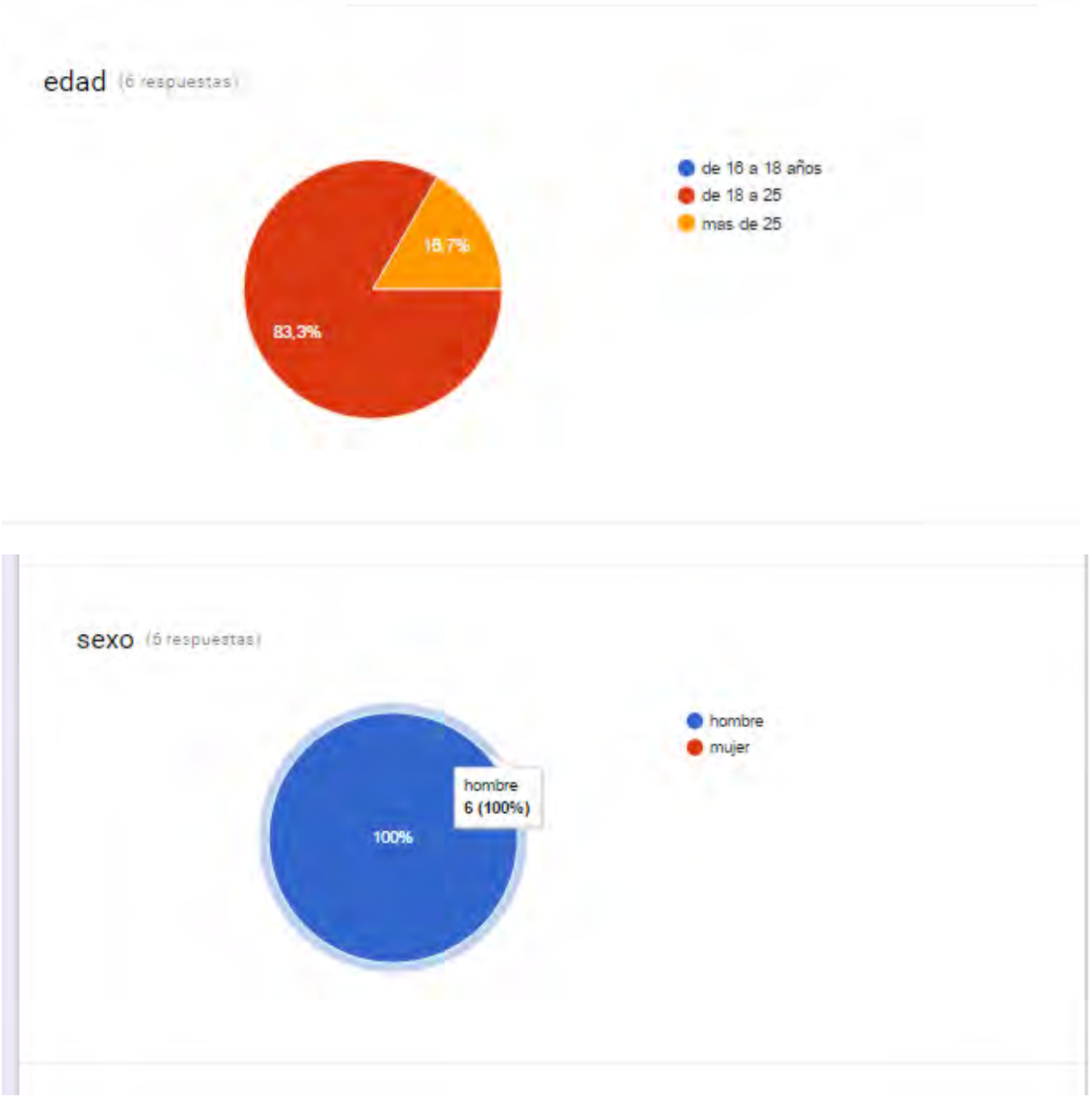


Figura 4.2.1 Gráficas que muestran la edad y el sexo de las personas que realizaron la prueba de manejo en el simulador.

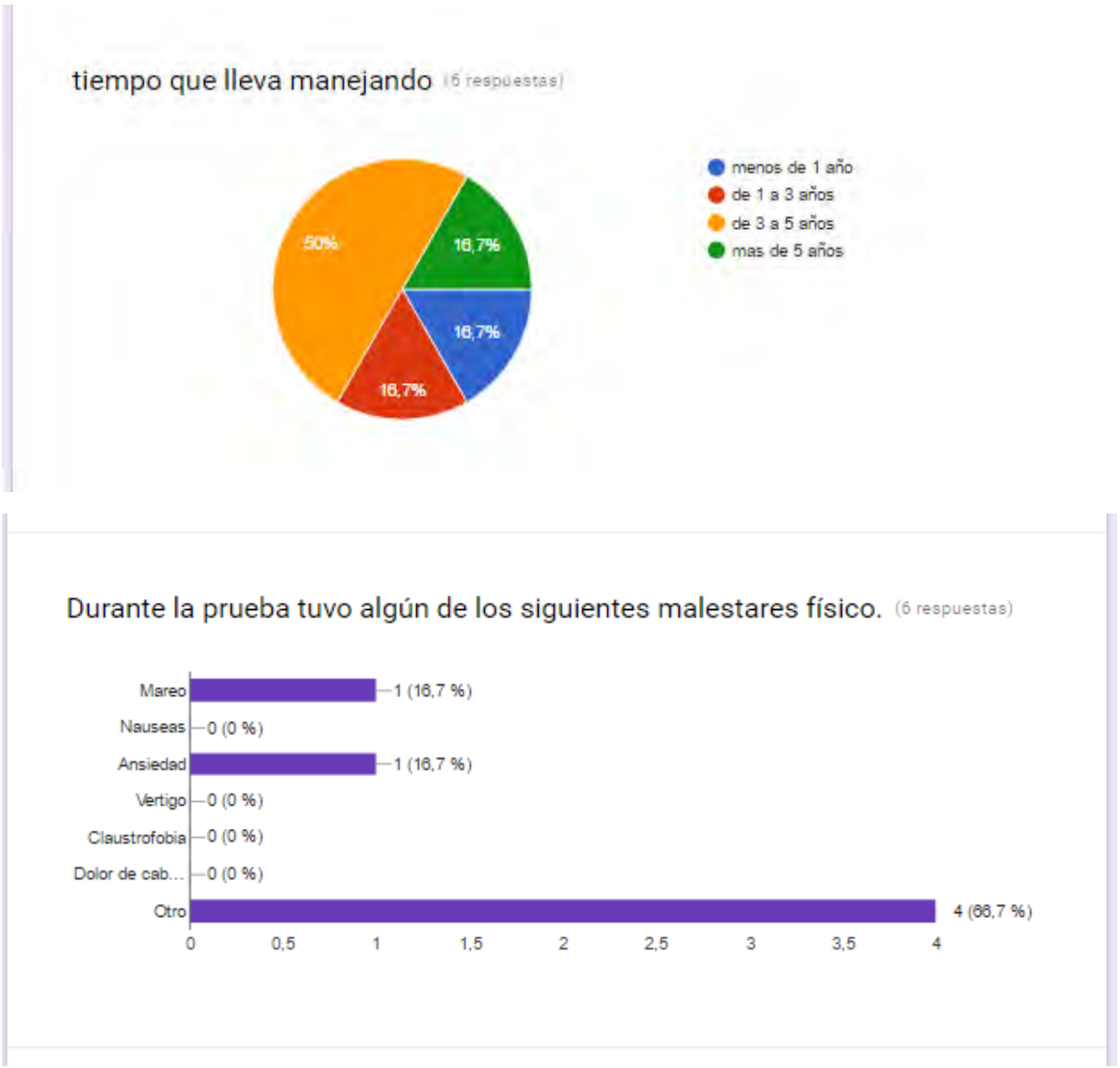
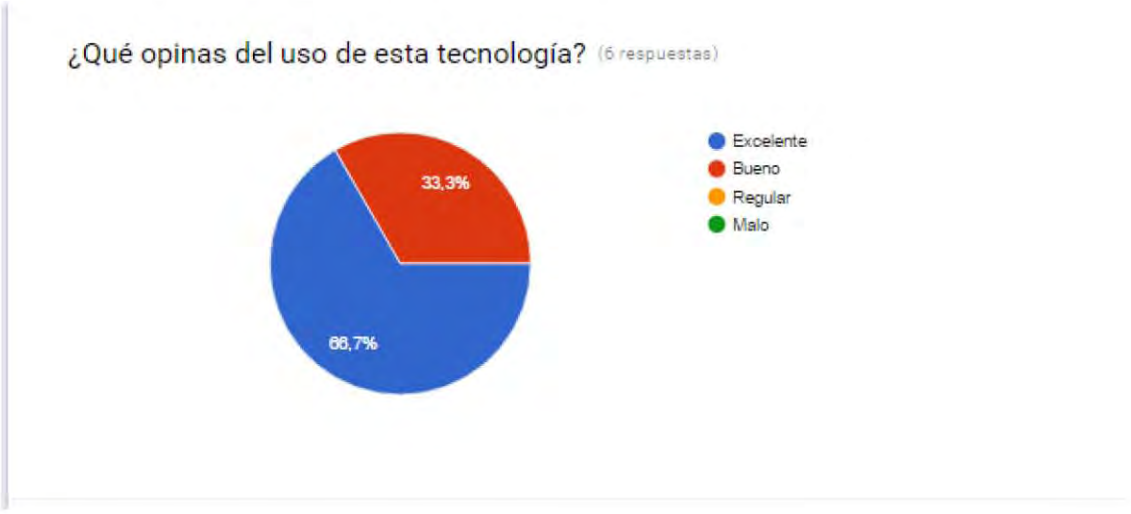
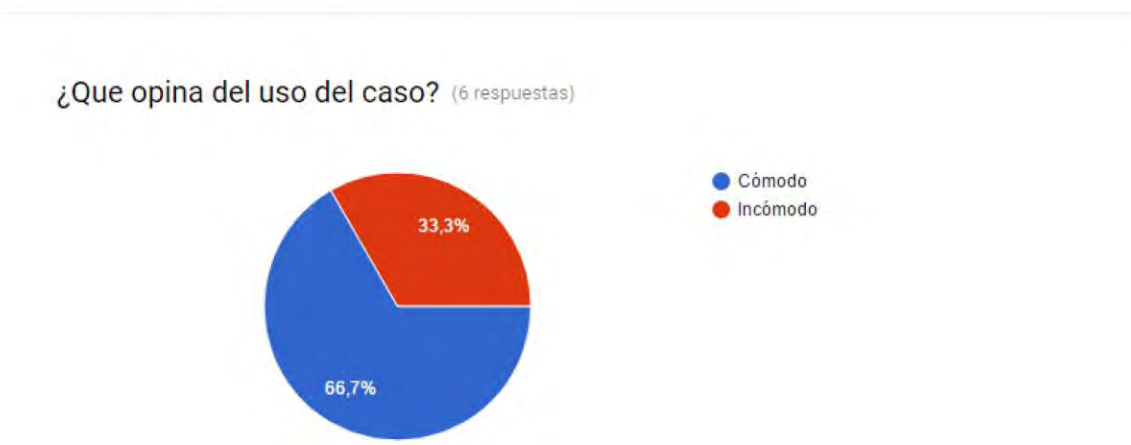


Figura 4.2.2 Gráficas que muestran el tiempo aproximado que llevan manejando y respuesta a si presentaron malestar físico durante la prueba de manejo en el simulador.





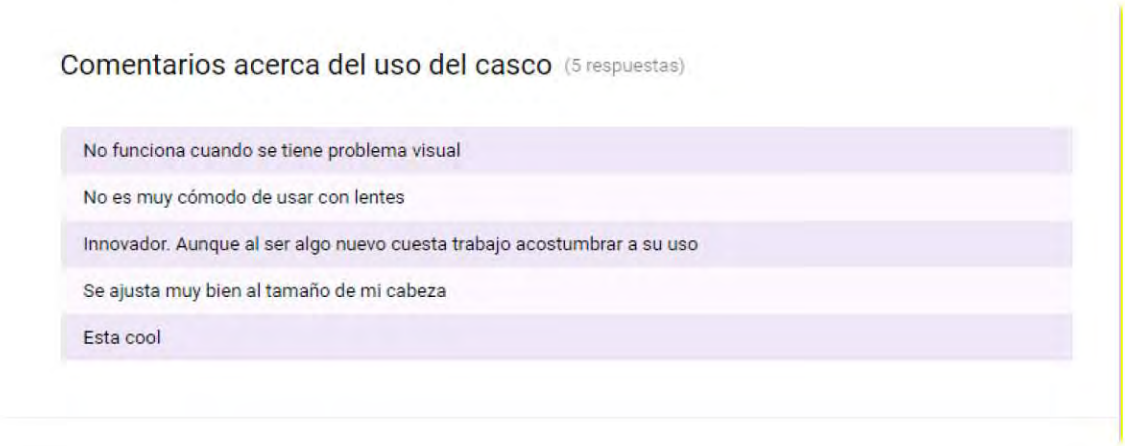


Figura 4.2.3 Gráficas que muestran opiniones referentes al uso de la tecnología y al dispositivo oculus rift.

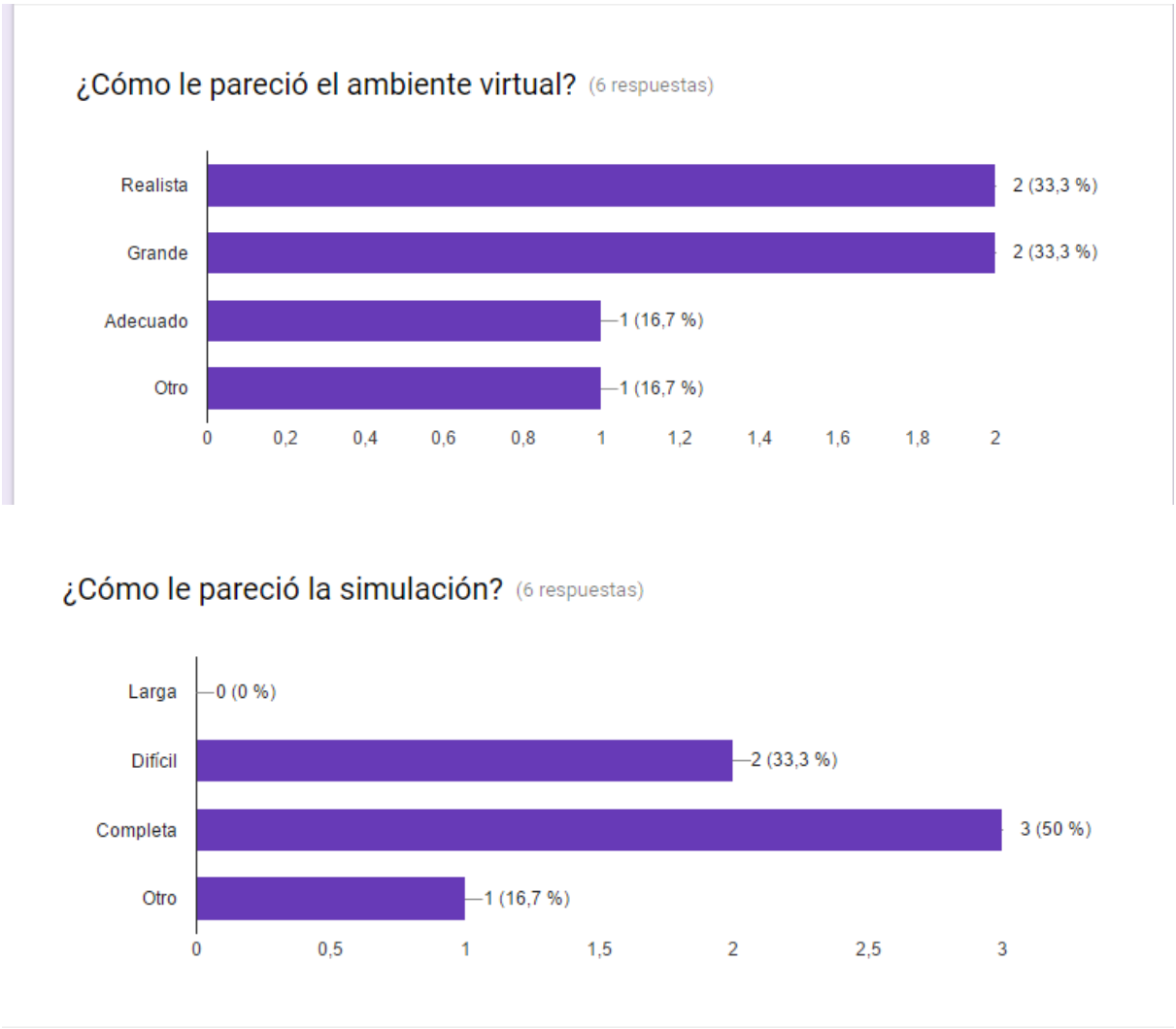


Figura 4.2.4 Gráficas que muestran opiniones referentes al entorno virtual realizado y a la simulación.

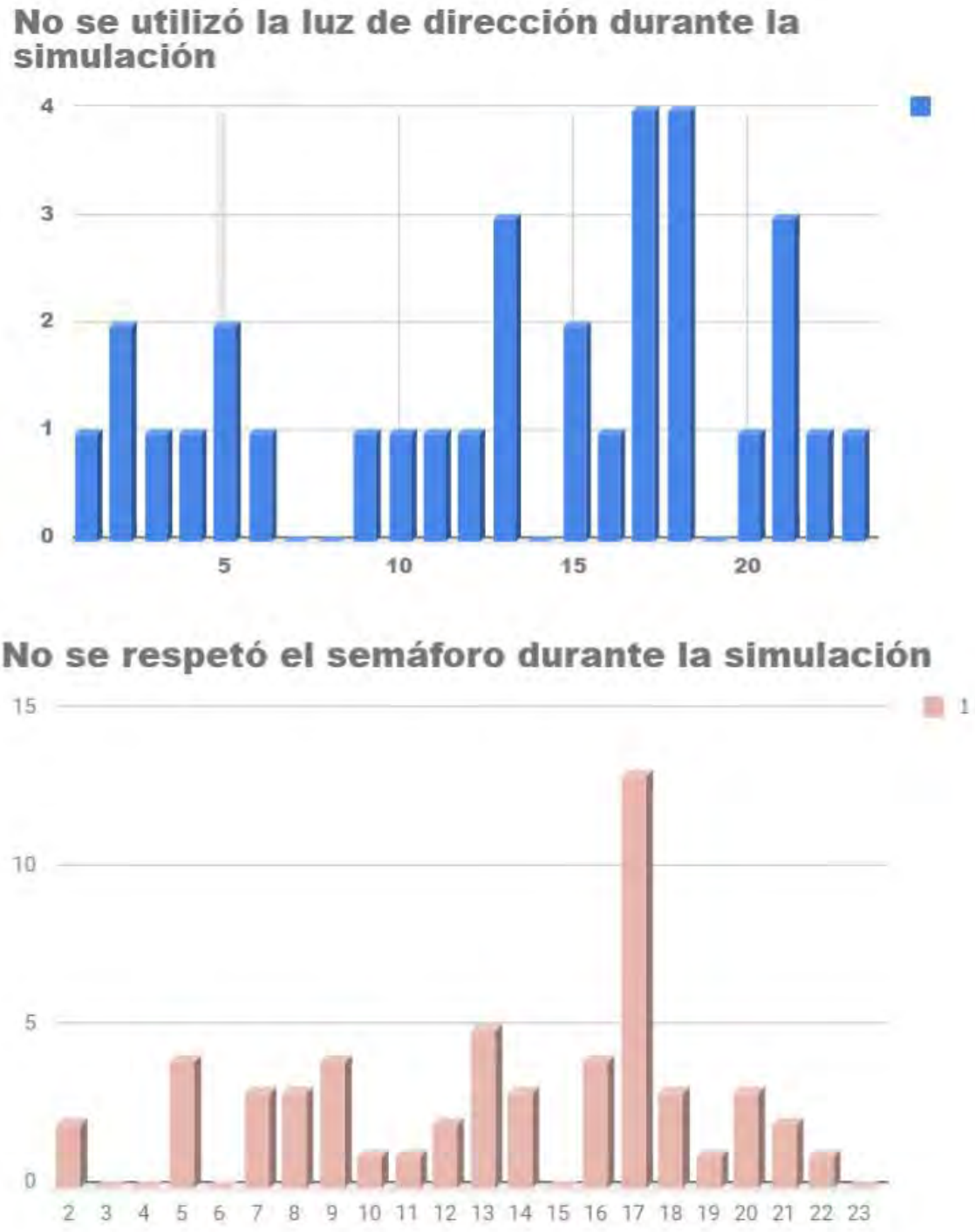


Figura 4.2.5 Gráficas que muestran el número de veces que el usuario no utilizó la luz de dirección al dar vuelta y las veces que no respetó la señal de alto durante la simulación.

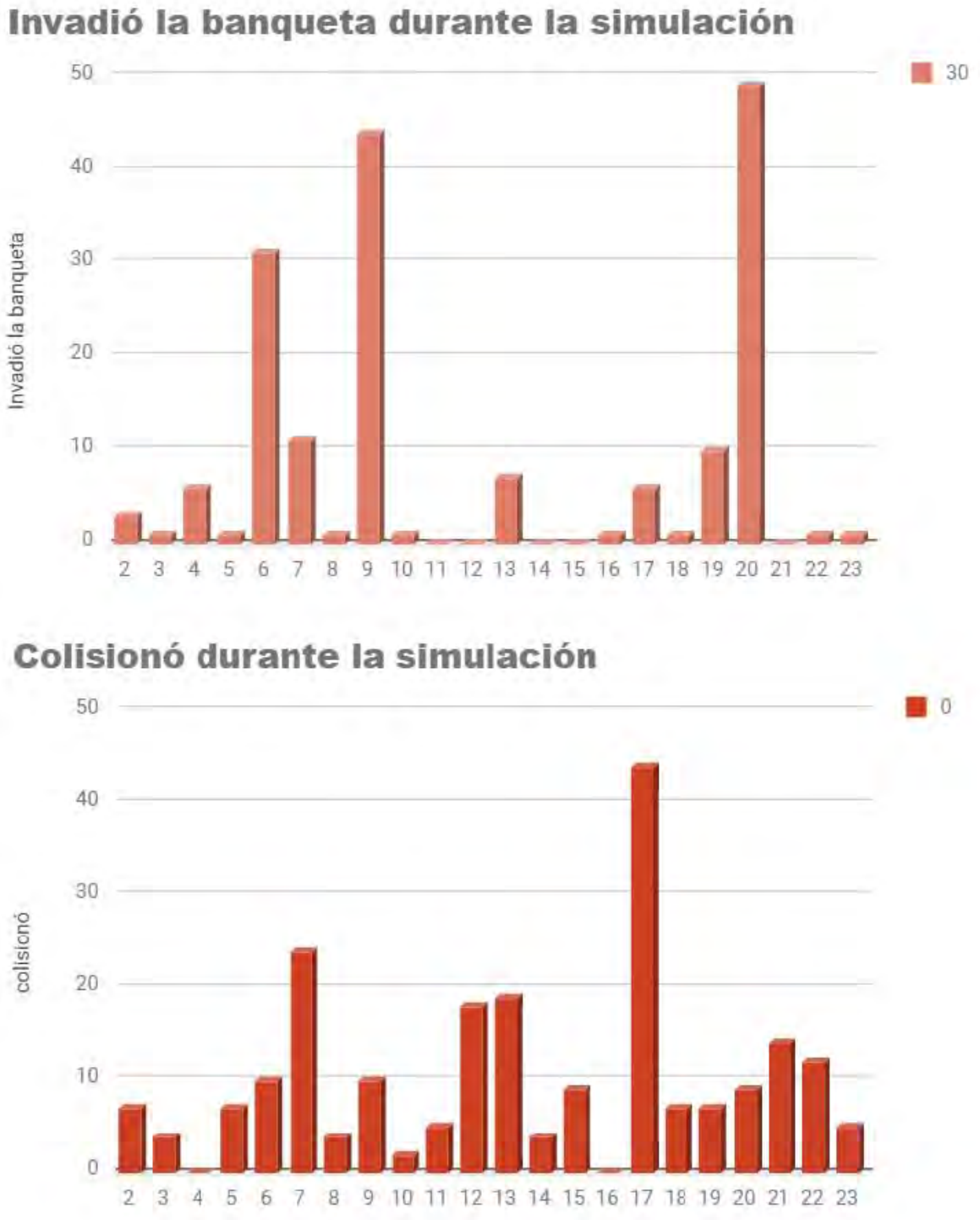


Figura 4.2.6 Gráficas que muestran el número de veces que el usuario invadió la banqueta y las veces que colisionó durante la simulación.

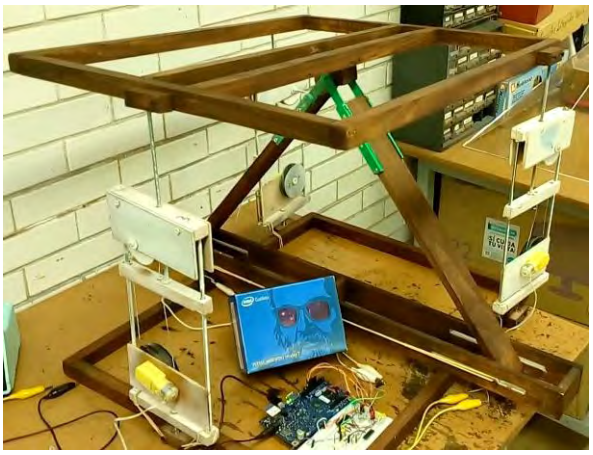
Figura 4.2.7 Gráfica que muestra el tiempo que el usuario empleó en finalizar la simulación.

4.3 Comparativas de las plataformas realizadas

Se construyeron dos propuestas distintas para simular los movimientos del auto virtual en el mundo real, con la finalidad de que la sensación de movimiento se transmita al usuario por medio de la plataforma donde está el asiento del conductor.

La primera se basó en 3 actuadores lineales y la segunda en un arreglo de 6 actuadores.

Comparación	Primer plataforma	Segunda plataforma
Ventajas	Más barata de construir. Fácil de manipular. No necesita retroalimentación.	Movimientos más precisos. Tiempo de respuesta más rápido. Más grados de libertad de movimiento.
Desventajas	Menor precisión de movimiento. Tiempo de respuesta más lento. Movimientos imprevistos (vibraciones en la plataforma). Menos grados de libertad de movimiento.	Más costosa de construir. Mayor complejidad para la implementación. Difícil de manipular.



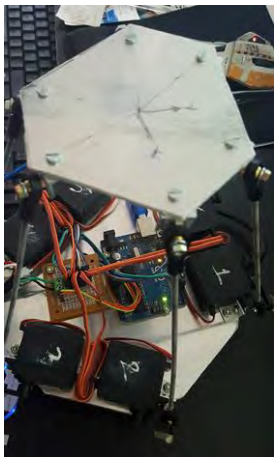


Figura 4.3.1 Fotografías de la primera y segunda plataforma construidas respectivamente.

4.4 Ventajas y desventajas del sistema creado

A diferencia de otros simuladores que utilizan pantallas para visualizar el entorno en que te encuentras, en éste proyecto se está implementando el uso de una nueva tecnología de realidad virtual llamada Oculus Rift.

Este dispositivo, a diferencia de las pantallas tradicionales, logra una mayor inmersión para el usuario, con esto logramos una mayor experiencia en cuanto a simulación ya que el sistema responde dinámicamente, es capaz de mostrar todo el ambiente virtual en función a la dirección donde el usuario está dirigiendo la mirada.

Con el uso de esta tecnología logramos una mejor integración del usuario con el sistema, ya que se consigue estar en el entorno virtual de una manera más natural que con el uso de pantallas ya que estas limitan lo que se puede observar.

Sin embargo hemos detectado algunos problemas con esta tecnología:

- El uso de cables, puede llegar a ser estorboso.
- El uso excesivo de la tecnología genera algunos malestares al usuario como puede ser mareo, vértigo, irritación en los ojos, y dolor de cuello, sobre todo a la gente que no está acostumbrada a interactuar con equipos de cómputo.
- El usuario pierde noción del entorno real donde se encuentra, por lo tanto, perdemos algunas funciones como es encontrar la palanca física de velocidades, o algunos botones. Por ese motivo se acordó que la palanca de velocidades únicamente tuviera 2 opciones, adelante y atrás. Para implementar esta solución fue necesario cambiar la mecánica del funcionamiento de las velocidades de un auto estándar, a un auto automático. Y demoró una semana más de lo planeado implementar dichos cambios.

5.- Conclusiones

El presente trabajo de tesis, surgió de la oportunidad de participar en el proyecto institucional PAPIIT IT102015, “Simulador para evaluar prácticas de manejo”, siendo el responsable el Ing. Juan José Carreón Granados.

Con base en los objetivos establecidos, se puede concluir que el desarrollo del proyecto se realizó satisfactoriamente al cumplir con el objetivo principal de construir un simulador de manejo implementando tecnología de realidad virtual.

Se realizó el modelado tridimensional implementado para la simulación. No se pudo realizar el asiento como se estableció en un inicio, pero se realizó un modelo a escala capaz de realizar los movimientos de un asiento. Se implementó un sistema para evaluar algunas de las habilidades consideradas necesarias para conducir.

Durante las pruebas de manejo aplicadas con el sistema, pudimos resaltar que algunas personas que tienen el permiso para conducir, no respetan algunas señalizaciones establecidas durante la prueba, no acostumbran hacer uso de las luces para el cambio de carril o girar en alguna esquina.

La mayoría respetó la señalización del semáforo al no pasar un cruce con la luz en rojo, ni tuvo ningún percance con otro vehículo en movimiento ni estacionado.

Otro factor detectado durante la prueba en donde los usuarios se demoran más incluso algunos no pudieron completar es al momento de estacionarse en paralelo. Preguntando a los usuarios que no terminaron la prueba, respondieron algunos que no saben realizar la maniobra necesaria para estacionarse, otros porque mencionan que no tienen la misma percepción del entorno como en el mundo real.

La oportunidad de participar en la realización de este proyecto fue para mí muy enriquecedor ya que apliqué muchos conceptos que aprendí a lo largo de la carrera, pero que muchas veces se quedaban en la teoría.

Estar en todo el proceso del proyecto desde su concepción, solicitud de aprobación, planeación, organización, desarrollo, pruebas, documentación y entrega, me aportaron mucha experiencia, aprendizaje acerca de cómo se debe llevar un proyecto por completo.

Un gran aprendizaje en particular que tuve con la realización del proyecto fue el trabajo en equipo, no solo con gente de la misma área sino con personas de diferentes áreas que a mi parecer es algo que se debe dar a lo largo del estudio de la carrera pero que desafortunadamente no se ve en varias materias que se imparten en la Facultad.

En particular creo que hay muchas materias que necesitan actualizarse en la Facultad ya que durante el desarrollo del proyecto me di cuenta que algunas herramientas que se usan actualmente no las aprendí en clases, como por ejemplo el uso de manejador de versiones como github, el cual es una gran herramienta que ayudaría bastante durante la carrera y no se ve en ninguna materia.

El desarrollo de este proyecto es una muestra del potencial que tiene actualmente el uso de la realidad virtual y de motores de juegos, demostrando que no solo se usa para fines de entretenimiento, sino que también sirve para fines de investigación, capacitación y evaluación.

5.1 Trabajo a futuro

Esperando que en un futuro el sistema no solo pueda evaluar, sino que también sirva como simulador de entrenamiento para la gente que no tenga las cualidades necesarias para manejar pueda adquirirlas con el uso del sistema.

Se espera que pueda servir también como simulador de manejo para ciertos casos en particular, como puede ser con lluvia abundante, nieve, neblina, etc.

Otra aplicación que pudiera tener el sistema es utilizarlo como sistema de capacitación para transporte público, privado o para unidades especiales, como puede ser el caso de una ambulancia, patrulla, camión de bomberos, camión de rescate, etc.

El sistema se construyó de manera modular, esto permite agregar cualquier elemento tridimensional adicional como pueden ser modelos de edificios diferentes, autos, árboles, o cualquier otro elemento deseado sin la necesidad de realizar grandes cambios. Por lo tanto el entorno virtual se encuentra diseñado para poder crecer rápidamente.

Referencias

[1] DF. (2014). Principales causas de muerte en México. 2015, de Rei IO Sitio web: http://ret.io/mx/DF/stats/

[2] DF. (2013). Antecedentes históricos de realidad virtual. 2017, de S.A.B.I.A.web: http://sabia.tic.udc.es/gc/Contenidos%20adicionales/trabajos/3D/Realidad%20Virtual/web/historia.html

[3] DF. (2016). Qué es la VR: historia y tipos de gafas de realidad virtual. 2017, de Mediatrends, web: https://www.mediatrends.es/a/65544/que-es-vr-historia-tipos-gafas-realidad-virtual/

[4] DF. (2017). Simulator specifications and pricing. 2017, de Simulator Driver Training web: http://www.sdt.com.au/docs/SDT%20Simulator%20Specs%20and%20Pricing.pdf

[5] DF. (2016). Carnetsoft driving simulator software. 2017, de Carnetsoft web: http://cs-driving-simulator.com/

[6] DF. (2015). Software Blender. 2017, de Blender web: https://www.blender.org/

[7] DF. (2015). Software 3ds Max. 2017, de Autodesk web:https://www.autodesk.mx/products/3ds-max/overview

[8] DF. (2015). Software Maya. 2017, de Autodesk web: https://www.autodesk.mx/products/maya/overview

[9] DF. (2015). Software GIMP.2017, de GNU IMAGE MANIPULATION PROGRAM web: https://www.gimp.org/

[10] DF. (2015). Software Substance. 2017, de Allegorithmic web: https://www.allegorithmic.com/

[11] DF. (2015). Software Unreal Engine. 2017, de Epic Games web: https://www.unrealengine.com/en-US/what-is-unreal-engine-4

[12] DF. (2015). Software Unity. 2017, de Unity web: https://unity3d.com/es

[13] DF. (2016). Análisis del Volante Logitech G27. 2017, de Playseat web: http://www.playseat.com/es/volante-logitech-g27-analisis

[14] DF. (2016). ¿Qué es el kit para desarrolladores Oculus?. 2017, de Gafas oculus web: http://www.gafasoculus.com/kit-desarrolladores-dk2/

[15] John Andrés Gòmez Portilla, José Vicente Guacaneme Gonzàlez. (2012). Diseño e implementación de una plataforma de Stewart. BOGOTÁ: Trabajo de grado para optar el título de Ingeniero en Mecatrònica, Universidad Militar Nueva Granada Facultad de Ingeniería Programa de Mecatrónica

[16] DF. (2018). Algoritmo de búsqueda A* (PathFinding A*) – XNA. 2011, de Escarbando Código web: https://escarbandocodigo.wordpress.com/2011/07/11/1051/