UNIVERSIDAD DE  VALLADOLID

E.T.S.I. TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Entorno virtual para la simulación de conducción basado en Unity

Autor:
Dña. Rebeca Paniagua Herrero
Tutor:
D. David González Ortega
Valladolid, 16 de Julio de 2015

TÍTULO:	**Entorno virtual para la simulación de conducción basado en Unity**
AUTOR:	**Dña. Rebeca Paniagua Herrero**
TUTOR:	**D. David González Ortega**
DEPARTAMENTO:	**Departamento de teoría de la señal y comunicaciones e ingeniería telemática**

TRIBUNAL

PRESIDENTE:	**Dña. Miriam Antón Rodríguez**
VOCAL:	**D. Mario Martínez Zarzuela**
SECRETARIO:	**D. David González Ortega**
SUPLENTE:	**D. Francisco Javier Díaz Pernas**
SUPLENTE:	**D. José Fernando Díez Higuera**

FECHA:	**16 de Julio de 2015**
CALIFICACIÓN:	

Resumen de TFG

El objetivo de este Trabajo Fin de Grado es crear un simulador de conducción para utilizarlo en tareas de investigación con el fin de recrear situaciones que podrían darse en las carreteras. Este sería utilizado para la educación, y así los usuarios podrían aprender sin comprometer su seguridad. Almacena información como la velocidad, revoluciones, consumo, infracciones cometidas, etc. en la simulación realizada. De este modo podemos analizar los resultados obtenidos, y así contribuir con la seguridad vial ayudando a las personas a corregir sus fallos a la hora de conducir. Además se podrían probar los efectos de la conducción en un mal estado psicofísico para ayudar a concienciar a las personas.

Este simulador se ha desarrollado esencialmente con el motor de juegos Unity y el programa de modelado 3ds Max. Para la utilización del simulador el usuario dispone del periférico Logitech G27, que incluye volante, pedales y cambio de marchas.

Palabras clave

Simulador, conducción, Unity3D, 3ds Max y seguridad vial.

Abstract

The purpose of this Final Proyect is to create a driving simulator for use it in research in order to recreate situations that could occur on the roads. This would be used for education, so users could learn without compromising safety. It stores information such as speed, revolutions, consumption, infringements, etc. in the carried out simulation. Thus we can analyze the results, and contribute to road safety by helping people to correct their faults when driving. Besides, we could test the effects of driving in a bad physical and psychological state to help educate people.

This simulator has been developed primarily with the Unity game engine and 3ds Max modeling program. To use the simulator the user has the peripheral Logitech G27, which includes steering wheel, pedals and gear shift.

Keywords

Simulator, driving, Unity3D, 3dsMax and road safety.

Agradecimientos

"A mi familia, especialmente mis padres y mi hermano por estar durante estos años en la carrera.

A mis amigos, tanto los de siempre como las nuevas amistades hechas durante mi estancia en Valladolid.

A Adrián, por su compañía y apoyo en los últimos años de carrera.

Por último, a mi tutor David, por ofrecerme la oportunidad de realizar este TFG."

Índice de contenidos

Índice de figuras

Índice de tablas

1. Introducción

En este apartado se realizará una introducción a este documento, de manera que se tratarán las causas que motivan la realización de este trabajo, de igual forma que los objetivos que se esperan alcanzar mediante su realización. Se describirán de manera general las fases y métodos que se seguirán para la realización de este trabajo y los medios utilizados para ello. Además se comentará la estructura que seguirá el resto del documento, describiendo de forma resumida sobre qué tratará cada uno de los siguientes apartados.

1.1. Motivación

El uso de simuladores de conducción para formar a conductores se está incrementando actualmente. Estos simuladores pueden resultar muy útiles como herramientas educativas para nuevos conductores, e incluso para mejorar la conducción en el caso de conductores veteranos. A través de ellos podemos crear situaciones significativas de una forma sencilla y simular situaciones difíciles (pero posibles) de encontrar en una carretera real.

Los simuladores de conducción han sido utilizados desde la década de 1960, aunque este concepto data de muchos años antes (Fisher D.L. et al., 2011).

Estos simuladores pueden acelerar el proceso de adquisición de habilidades básicas. Algunas de las ventajas del aprendizaje mediante simuladores son que se tiene un mayor control acerca de la situación sobre la que trabajar (se puede elegir qué escenario o entorno es el que se quiere simular), las condiciones de entrenamientos son seguras (no hay ningún posible peligro para el conductor) y mejoran la retroalimentación y las posibilidades de instrucción (los datos acerca de la conducción pueden ser registrados y medirse con precisión) (Vlaked, 2015).

De este modo se podría contribuir a reducir las cifras de la siniestralidad vial. Estas cifras han ido mejorando a lo largo de los últimos años, llegando a ser en 2013 España el quinto país de la Unión Europea con menor número de fallecidos por población (DGT, 2013), y utilizando estos simuladores podríamos continuar reduciendo esta cifra de una manera aún más significativa.

1.2. Objetivos

El objetivo de este Trabajo Fin de Grado es la creación de un entorno virtual para la simulación de conducción mediante el motor de juegos *Unity*, y con él conseguir que los usuarios puedan aprender a manejar un automóvil y a saber cómo actuar en diferentes situaciones que se nos podrían presentar en la carretera. De este modo podríamos evitar accidentes en la vida real, y que hubiese un mejor comportamiento por parte de los usuarios en la carretera.

Se realizará un ejecutable en el que contaremos con diferentes escenarios en los que podremos observar diferentes situaciones. El usuario controlará un automóvil

mediante el periférico Logitech G27 (volante, pedales y marchas), y deberá conducirlo por los escenarios intentando realizar esta conducción de la manera más correcta posible y respetando las normas de conducción. La información acerca de las simulaciones realizadas será guardada en ficheros externos XML de modo que estos serán accesibles posteriormente para cualquier tipo de consulta. Ahí podremos observar su posición, velocidad, revoluciones, marcha, consumo, etc. en los diferentes instantes de tiempo, así como las infracciones cometidas y los datos de conducción del momento en el que se producen. Además se proporcionará información acerca de la velocidad media que se ha llevado durante el recorrido, así como el consumo de combustible medio y total.

1.3. Fases y métodos

Las fases que seguiremos en este proyecto son:

- Estudio sobre la seguridad vial, en el que se proporcionará información sobre el modo correcto de circulación en ciertas situaciones, y el estado actual del mundo en cuanto a este tema.
- Estudio acerca de los diferentes simuladores de conducción hasta ahora disponibles en el mercado y cómo podrían contribuir a la seguridad vial.
- Análisis acerca de las tecnologías y herramientas disponibles tanto para la realización de videojuegos (motores de juegos) como para el modelado 3D. Entre otras cosas, se analizarán sus ventajas y desventajas.
- Elección y estudio del funcionamiento de las herramientas a utilizar finalmente.
- Desarrollo de la aplicación, donde crearemos diferentes escenarios para las simulaciones (con su correspondiente física necesaria) además de los scripts necesarios, principalmente en C# y puntualmente en JavaScript, para su correcto funcionamiento.
- Se realizarán algunas pruebas para observar los resultados obtenidos de la conducción de diferentes usuarios.
- Estudio sobre posibles líneas futuras para poder continuar con un desarrollo más amplio del proyecto.

1.4. Medios

Durante el desarrollo de este trabajo, se utilizará un equipo con las siguientes características:

- Procesador: Intel Core I7-4510U CPU @ 2.0GHz 2.60GHz
- Memoria RAM: 8 GB
- Sistema operativo: Windows 8.1 (64 bits)
- Tarjeta gráfica: Nvidia GeForce 820M

Además se dispondrá del dispositivo periférico Logitech G27, el cual está formado por volante (con levas incluidas), pedales y palanca de cambios.

1.5. Estructura de la memoria

Esta memoria se compone de diferentes apartados, los cuales explicaremos a continuación.

En este apartado estamos viendo una introducción acerca de lo que constará este trabajo y las partes que se irán tratando a lo largo del resto del documento.

En el segundo apartado se hablará acerca de la seguridad vial, donde se tratarán definiciones y principios relacionados con esta, desde el concepto de tráfico hasta el de seguridad vial; también se tratará acerca de los accidentes y los factores que influyen en ellos, además de las diferentes fases en que se produce un accidente; se mencionarán algunas medidas a llevar a cabo para obtener una mejora de la seguridad vial; y finalmente se tratará el tema de la conducción eficiente, viendo algunas pautas sobre cómo debería realizarse la conducción para minimizar el consumo. Además se hablará de algunos de los simuladores de conducción existentes en la actualidad, viendo sus funcionalidades y utilidades. Por último se hará un estudio acerca de cómo pueden contribuir los simuladores de conducción a una mejora en la seguridad vial.

El tercer apartado tratará acerca de las diferentes tecnologías disponibles que podrían ser utilizadas para realizar un simulador de conducción. Para ello necesitaríamos un motor de juegos para hacer la física y programar la lógica del simulador, además de un programa de modelado 3D para realizar los modelos a utilizar en el simulador. Se mostrarán diferentes motores de juegos y sus características, y se hará lo mismo para los diferentes programas de modelado 3D. A continuación se hará una comparativa entre los diferentes programas que se han visto, eligiendo los utilizados finalmente para la realización de este trabajo.

En el cuarto, veremos los distintos componentes de nuestro simulador, comenzando con el menú inicial. A continuación se comentarán los diferentes escenarios con los que se cuenta y cómo han sido desarrollados. Posteriormente se verá cómo funcionan y cómo se han creado diferentes elementos como son el coche a conducir, los diferentes coches con inteligencia artificial que circularán por las escenas, peatones, y otros componentes y elementos que forman parte del simulador como son, por ejemplo, los controladores de juego o los que se encargan de guardar los datos. Respecto a la inteligencia artificial mencionada sobre los coches, esta representa la capacidad de un sistema para tomar sus decisiones como lo haría un ser humano (en este caso como si alguien lo estuviese conduciendo realmente). (APA, 2015).

En el quinto apartado veremos un manual sobre este simulador, en el que se mostrarán cuáles son los diferentes controles para utilizarlo, tanto con el teclado del ordenador como con el periférico Logitech G27. Además se contará cómo funciona el simulador y cómo se deberá utilizar desde que se ejecuta la aplicación y se accede al menú de inicio.

El sexto apartado tratará acerca de diferentes pruebas que se han hecho con este simulador, siendo manejado por diferentes usuarios. Se analizarán parámetros como son la velocidad y el consumo del automóvil. De este modo, posteriormente se contrastará información acerca de los resultados obtenidos por los diferentes usuarios y entre las diferentes escenas y modos de conducción. Para ello se hará un test previo a los usuarios acerca de su probabilidad de sueño en diferentes situaciones y el estado en el que se encuentran antes de comenzar la simulación. Finalmente se mostrarán las opiniones de los usuarios respecto a la funcionalidad del simulador.

En el séptimo apartado se comentan las conclusiones obtenidas, y se trata sobre posibles líneas futuras, para poder continuar el desarrollo de un proyecto mayor y más útil.

El apartado 8 trata sobre un estudio económico del coste que habría tenido este proyecto.

En el apartado 9 tenemos la bibliografía, en la cual podemos encontrar las referencias mencionadas a lo largo del documento, y otros lugares de los que se ha obtenido información útil para la realización del proyecto.

Finalmente tenemos un anexo en el que se trata sobre la corrección de errores que se ha realizado en el modo de sobrecarga de métodos utilizado en varios de los scripts proporcionados en un *plugin* que ha sido utilizado en el desarrollo de 2 escenas (*plugin Road & Traffic System*).

2. Seguridad vial y simuladores de conducción

En este apartado se tratarán ciertas cuestiones sobre seguridad vial. Además se comentarán cuáles son algunos de los simuladores de conducción existentes en la actualidad, y se estudiará cómo estos pueden ser útiles para mejorar la seguridad vial.

2.1. Seguridad vial

A continuación se hablará sobre el concepto de seguridad vial y los accidentes y cuáles son las causas de que se produzcan. Además se tratará el tema de la conducción eficiente.

2.1.1. Definiciones y principios

El tráfico puede definirse como "el desplazamiento de personas, animales y vehículos por las carreteras, calles y caminos" (DGT, 2014). Los factores que intervienen en el tráfico son: el factor humano, el vehículo y la vía y su entorno, de forma interrelacionada. Concretamente el factor humano interviene en el 90% de los accidentes, y en ello influyen la psicología, la pedagogía, la medicina, las normas, la técnica de conducción, etc. En la figura 2.1 se muestra el logo de la DGT, es decir, la Dirección General de Tráfico.

Figura 2.1. Logo de la DGT (Dirección General de Tráfico)

Los principios fundamentales que deben regular el tráfico son: la responsabilidad, la confianza en la normalidad del tráfico, la conducción defensiva, la seguridad en la conducción, la señalización o conducción dirigida y la integridad personal. Estos sirven para obtener los fines de la seguridad vial que son: la seguridad, la fluidez, la comodidad, la economía y la no contaminación. Estos fines son indispensables para una conducción segura. De hecho la seguridad vial podría ser definida como la "no producción de accidentes" (DGT, 2014).

2.1.2. Accidentes

El accidente puede ser definido como "cualquier evento como resultado del cual el vehículo queda de manera anormal, dentro o fuera de la carretera, o produzca lesiones en las personas y daño a terceros". Más de un millón de personas mueren al año a causa de accidentes de tráfico en todo el mundo (DGT, 2014). En España en los últimos años ha disminuido mucho el número de víctimas mortales en accidentes de tráfico, llegando

a alcanzar unas 1.128 víctimas mortales en vías interurbanas respecto a las 4.200 en 1996.

Los factores que intervienen, como ya mencionamos anteriormente, son la vía y su entorno, el vehículo y el factor humano. En la figura 2.2 podemos observar estos 3 componentes en el triángulo de la seguridad vial. Respecto al factor humano, que es el más cercano al estudio que realizaremos con nuestro simulador, influye el estado físico y psíquico del conductor, el nivel de vigilancia, el conocimiento de la normativa y la competencia técnica (que depende de la formación y práctica).

Figura 2.2. Triángulo de seguridad vial: factor humano, vehículo y vía

Los accidentes pueden ser debidos a muy diversas razones por cada uno de los factores mencionados anteriormente. En cuanto al factor humano, puede ser debido al comportamiento incorrecto de peatones, como puede ser cruzar la vía fuera de la zona marcada o infringiendo una señal; al comportamiento incorrecto del conductor que podría sobrepasar la velocidad máxima permitida, no respetar alguna señal de prioridad, etc.; el estado psicofísico del conductor, ya que este puede estar distraído, cansado, haber tomado alcohol o drogas, etc. Respecto a las causas relacionadas con el vehículo, a este le pueden fallar los frenos, las luces, las ruedas, entre otros (por lo que es importante mantenerlo en un buen estado de conservación). En cuando a las causas relacionadas con la vía, estas podrían ser ciertas curvas, pasos a nivel y estado del pavimento.

Además los accidentes producen tanto un coste social por la pérdida de vidas o de salud, como un coste económico por los costes sanitarios, indemnizaciones, reparaciones de vehículos, etc.

El rango de edad de personas que son más frecuentemente afectadas por accidentes va desde los 25 hasta los 44 años. Además los accidentes son la mayor causa de muerte en jóvenes de edades entre los 15 y 30 años.

Los accidentes pueden ser clasificados utilizando los siguientes criterios: por su situación, por sus resultados, por el número de vehículos implicados, por el modo en que se producen, por ser accidentes con características especiales u otros criterios. En cuanto a su situación estos pueden ser urbanos o interurbanos. Respecto a los resultados podemos tener accidentes mortales, con heridos y/o con daños materiales. En cuanto al número de vehículos implicados, estos pueden ser simples (si solo interviene una unidad de tráfico) o complejos (si intervienen dos o más). El modo en el que se producen pueden ser: colisiones (frontales, embestidas, reflejas, por alcance, por raspado o múltiples), choques, salidas de la vía (despeñamientos o vuelcos) o atropellos. Los accidentes con características especiales son los que no pueden ser clasificados en ninguna de las anteriores opciones. Además se pueden hacer otro tipo de clasificaciones como podría ser según la hora del día, según el día o según la naturaleza del transporte.

En un accidente pueden distinguirse tres fases: percepción, decisión y conflicto.

Dentro de la fase de percepción encontramos el área de percepción que se encuentra entre el punto de percepción posible y el real. El punto de percepción posible (PPP) es el momento en el que el conductor debió darse cuenta de la situación que podía terminar en accidente (momento en el que se daría cuenta un conductor cuidadoso). El punto de percepción real (PPR) es el momento en el que el conductor percibe el posible peligro.

En la fase de decisión es cuando la persona implicada reacciona ante el estímulo de percepción. Aquí el área de decisión se encuentra entre el punto de percepción real y el punto de conflicto. La persona implicada tiene un tiempo de reacción desde que se da cuenta del peligro hasta que hace algo para intentar evitarlo. Este tiempo puede oscilar entre 0.4 y 2 segundos, dependiendo del estado y características del conductor. El punto de decisión (PD) es aquel en el que se inicia la maniobra de evasión, la cual puede ser simple (tocar el claxon, disminuir la velocidad,…) o compleja (disminución de la velocidad y giro, giro y uso del claxon,…). El área de maniobra comienza en el punto en que el conductor inicia la maniobra y termina en el punto de conflicto (PC).

Respecto a la fase de conflicto, aquí podemos distinguir el área de conflicto que se encuentra entre el punto clave (punto en el que si no se actúa el conflicto resulta inevitable) y el punto de conflicto (en el que se produce el accidente). Además se distingue la posición final que es la que adoptan los elementos que han intervenido en el accidente cuando llegan a quedar inmóviles. En la figura 2.3 se muestran estas fases del accidente, con los puntos que las definen.

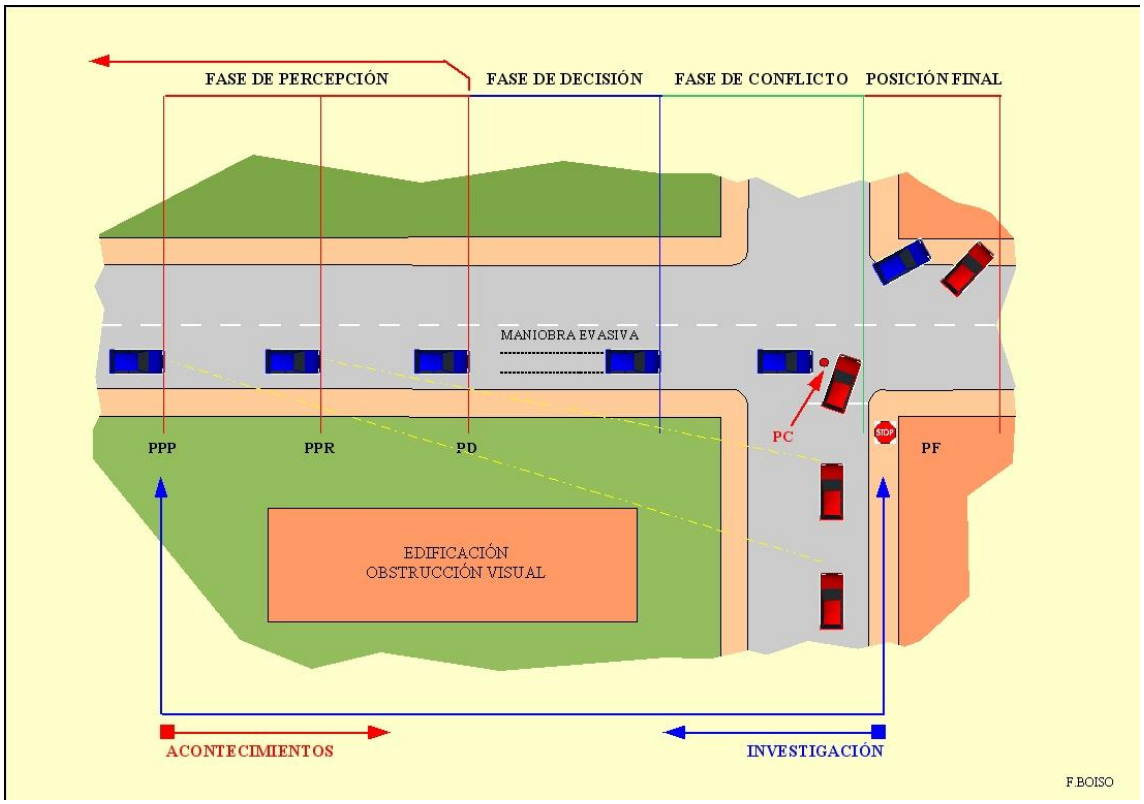

Figura 2.3. Fases del accidente

Actualmente se están llevando a cabo investigaciones acerca de las causas que originan los accidentes, el proceso de desarrollo de estos y sus consecuencias. Esto se puede realizar mediante métodos estadísticos o de modelización (diseños matemáticos), de simulación de accidentes, de reconstrucción de accidentes, etc.

2.1.3. Medidas para mejorar la seguridad vial

Estas medidas pueden ser en relación con la vía, con el conductor-vehículo, con el factor humano y con la asistencia al usuario.

En cuanto a la vía, se deben disminuir las exigencias del entorno. Esto puede ser en cuanto a la infraestructura, por lo que se deben corregir riesgos objetivos obteniendo carreteras bien trazadas, señalizadas y conservadas; en cuanto a la normativa, por la que se debe perfeccionar la regulación conciliando las exigencias de fluidez con las de seguridad; y en cuanto a los sistemas de aviso, por los que se debe mejorar la información proporcionada por ejemplo en condiciones meteorológicas adversas.

Respecto a la relación conductor-vehículo, hay que mejorar las condiciones mecánicas de los vehículos y las aptitudes de los conductores y usuarios de las vías. En el vehículo hay que tener en cuenta los factores que surgen en la fabricación del vehículo y posteriormente los relacionados con el uso de este. Se deben realizar comprobaciones de los mandos del automóvil, la puesta en marcha del motor, cambio de marchas y uso del freno, entre otras.

En el factor humano, el cual es el factor más importante a la hora de prevenir los accidentes, tenemos varias medidas a tener en cuenta:

- Condiciones psicofísicas: en la carretera, el conductor debe recibir información, procesarla y evaluarla, tomar decisiones, ejecutarlas y controlar los resultados obtenidos. Por ello algunos factores que pueden afectar negativamente a la capacidad para conducir en el caso de no estar en buenas condiciones serían la capacidad visual o auditiva, el sistema locomotor, sistema cardiovascular, trastornos mentales y de conducta, trastornos relacionados con sustancias como el alcohol y las drogas, etc.
  Los procesos psicológicos implicados en la conducción son la capacidad perceptiva y atencional, la capacidad intelectual, la toma de decisiones, la capacidad de respuesta y la personalidad.

- Educación vial: persigue la formación del comportamiento del ciudadano al ser usuario de las vías públicas, ya sea como conductor, peatón o viajero. Esto es algo que influye a todas las personas, incluso a los niños ya que todos utilizamos las vías públicas. Se deben tener unas condiciones de convivencia ordenada, solidaria, responsable y de respeto mutuo.

- Formación de conductores: consiste en la adquisición de las aptitudes y habilidades necesarias para la conducción del vehículo, los conocimientos sobre las señales y normas que regulan la circulación y las actitudes respecto a esas normas. Esta formación consta de dos fases: aprendizaje (asistencia a la autoescuela hasta adquirir los conocimientos necesarios hasta pasar las pruebas de aptitud) y puesta en práctica y perfeccionamiento. La inexperiencia es uno de los factores causales de accidentes.

- Actividad informativa y motivadora: aquí entran en juego las campañas divulgativas a través de los medios de información.

- Vigilancia y control: la presencia de agentes encargados de la vigilancia del tráfico. Se imponen sanciones de tráfico con el fin de que el conductor sancionado (incluso el resto) no cometa nuevas infracciones.

Para una conducción segura es fundamental una observación y atención adecuadas (donde es importante la anticipación), una comunicación con el resto de usuarios y una actuación correcta.

Por último, en relación con la asistencia al usuario se encuentran el auxilio sanitario, el auxilio mecánico y una red de postes de petición de auxilio.

2.1.4. Conducción eficiente

En España existe una demanda de petróleo muy grande. Esta ha crecido en cantidades enormes desde el año 1965. Por ello se están elaborando modelos energéticos que reduzcan el consumo de energía y aprovechen energías renovables. En

la figura 2.4 vemos un gráfico acerca del consumo de energía primara en España en el año 2008, donde podemos ver que las energías renovables son poco utilizadas.

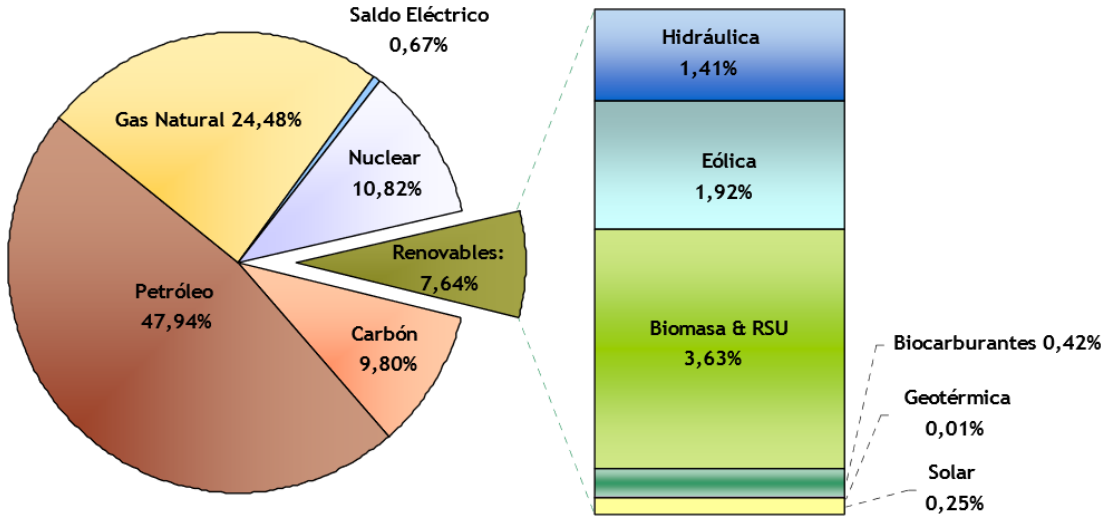

Figura 2.4. Consumo de energía primaria en España (2008)

Las medidas del plan de ahorro y eficiencia energética 2008-2011 incluyen: impulsar una conducción eficiente, fomentar el uso del transporte público, impulsar el uso de vehículos eléctricos, plan *VIVE* (Vehículo Innovador-Vehículo Ecológico), etiqueta energética comparativa de consumo en los vehículos, incorporación de técnicas de conducción eficiente en el sistema de enseñanza, promoción del transporte urbano en bicicleta (con bicicletas de uso público y carriles bici urbanos) y carriles reservados al transporte colectivo de viajeros (*BUS-VAO*).

Un usuario debe trasladarse con el mínimo esfuerzo y además con el mínimo consumo posible, tanto para su bien económico como para el bien social de su país. Del combustible que gastamos, sólo un 25% de su poder calorífico es aprovechado por el motor, ya que tenemos muchas pérdidas al calentar el agua de refrigeración y sus elementos (~30%), a través del tubo de escape (~30%), por rozamientos internos (~6%) y por rozamientos de transmisión y rodadura (~9%). El gasóleo tiene un 14% más de energía por litro que la gasolina. La relación potencia-cilindrada que ofrece la gasolina es muy buena. Para obtener la misma en motores de gasóleo, es necesario aumentar la cilindrada, la sobrealimentación y la inyección controlada electrónicamente, aumentando las rpm (revoluciones por minuto). En tráfico urbano al llevar bajas velocidades el diésel consume un 25% menos, ya que los motores necesitan proporcionar poca potencia. Por carretera al llevar velocidades más altas el rendimiento de los motores de gasolina se aproxima a los de diésel.

En el caso de un vehículo que viaja a 120km/h, de la energía que se aprovecha para moverlo un 72% se utiliza para vencer la resistencia del aire, un 20% para vencer la resistencia de rodadura y un 8% para vencer la de transmisión, caja de cambios y diferencia.

En vehículos automáticos se debe utilizar el acelerador con suavidad y progresividad, de modo que se suministre una inyección "tranquila"; si se pisa y suelta con brusquedad el sistema selecciona una inyección "enérgica". Si tras pisar el acelerador al 100% (el recorrido normal) se pisa aún más oponiéndose a la resistencia que presenta, se acciona el "Kick Down" (mostrado en la figura 2.5) y se suministra la mayor potencia posible reduciendo la marcha a la más corta que se pueda, sin cambiarla hasta llegar al máximo de revoluciones (potencia máxima), inyectando la mayor cantidad de combustible posible.

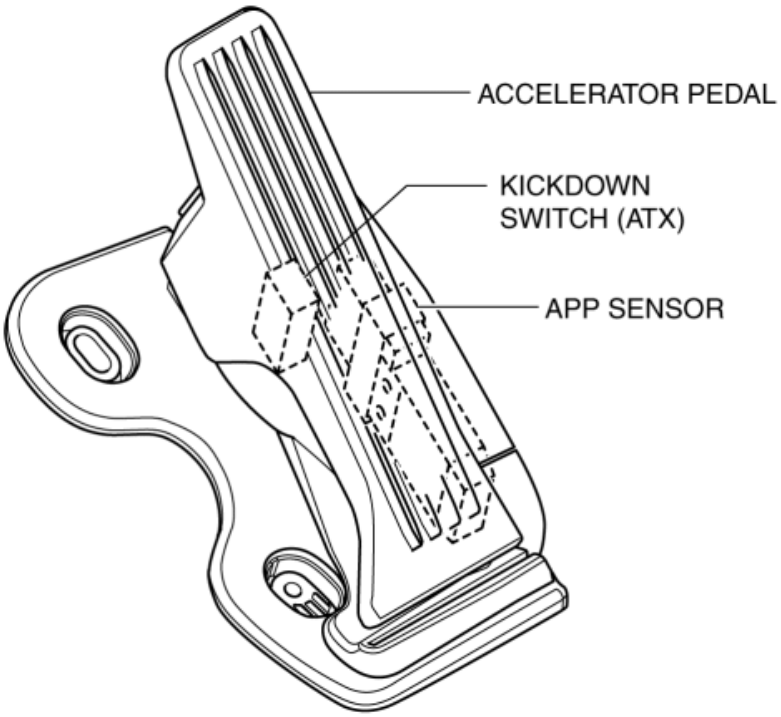

Figura 2.5. Acelerador con Kick Down

Mediante los cambios manuales es preferible mantener el acelerador en una posición fija (mejor que mantener una velocidad fija).

Si un vehículo está en perfectas condiciones mecánicas se puede obtener un ahorro del 20%, aunque la forma con la que se obtiene el mayor ahorro es mediante una conducción eficiente.

Algunas formas de ahorrar combustible mediante conducción eficiente son:

- Calentar el motor en movimiento circulando a una velocidad moderada y a un régimen de revoluciones constante (lo más uniformemente posible). Se debe iniciar la marcha inmediatamente después de haber arrancado el motor (salvo que sea diésel o sea aconsejado por el fabricante por otro motivo).
- Repostar combustible sin derramarlo y sin llenarlo hasta el borde (para que no se pueda derramar durante la marcha).
- Arrancar sin acelerar o acelerando con suavidad para no desperdiciar combustible ni hacer patinar el embrague.

- Utilizar la relación de marchas más adecuada, generalmente utilizando la más larga posible para mantener las revoluciones bajas. Se debe intentar cambiar el menor número de veces posible de marcha (se puede saltar una o dos marchas al cambiar cuando sea factible). Es diferente en el caso de gasolina que en el de diésel, pudiéndose encontrar el par motor en motores de gasolina en las 3.000 – 6.000 rpm, mientras que en los de diésel estaría en las 1.600 – 1.800 rpm. En caso de pendiente descendente, se debe soltar el freno del vehículo para adquirir inercia y poder meter la marcha segunda. En tramos con pendiente ascendente debe utilizarse la marcha más larga posible (al igual que en tramos llanos).

  Este aspecto también es importante en el caso de adelantamientos, para los cuales se podría utilizar una marcha alta de forma que aumentaríamos la velocidad más despacio y tardaríamos más en adelantar, o utilizar una marcha baja de manera que la velocidad aumentará más rápidamente al acelerar, y aunque en ese momento consumiremos más combustible, el tiempo de adelantamiento es más breve y por tanto antes volvemos a la normalidad (además es importante si es una vía de doble sentido de circulación).
- Circular con un régimen de revoluciones adecuado (normalmente el más bajo posible).
- Parar el motor en detenciones largas.
- No dar acelerones en vacío, especialmente al arrancar el motor (sin engrase suficiente ni temperatura ideal de trabajo). Utilizar el acelerador de una forma progresiva y constante.
- Mantener una velocidad constante.
- Utilizar buen aceite para reducir los rozamientos internos.
- Anticiparse a posibles acciones como detenciones, ya que se puede dejar de acelerar y llegar hasta ese punto por inercia (se deben aprovechar las fuerzas inerciales). Así además el tiempo de espera será menor. También puede darse el caso en que realizando esto el conductor pueda evitar detenerse ya que en ese transcurso de tiempo ha dejado de ser necesario (se ha puesto ya el semáforo en verde, etc.), y ahorraría pastillas de freno, embrague y combustible, además de dar cierto descanso al motor. La anticipación además proporciona una mayor seguridad a la hora de conducir.
- Llevar las ventanillas cerradas.
- Distribuir la carga evitando levantar la parte delantera.
- Circular por vías con circulación fluida para evitar posibles detenciones.
- Evitar utilizar baca.
- No sobrepasar las velocidades establecidas.
- No sobrecargar el vehículo.
- Llevar los neumáticos a la presión adecuada (no a menor).
- No utilizar gasolina de distinto octanaje al recomendado.
- No realizar doble embrague.
- No circular en punto muerto.
- Manejar el volante con suavidad para no forzar la dirección, la suspensión ni los neumáticos, y conseguir poca resistencia.

(DGT, 2014)

Respecto a los cambios de marcha recomendados para obtener estos resultados, si nuestro coche es de gasolina, el cambio de marcha se debe producir entre las 2.000 y las 2.500 rpm; en cambio, si nuestro coche es diésel, este cambio debe ser realizado antes, concretamente entre las 1.500 y las 2.000 rpm.

Si consideramos el cambio de marchas en función de la velocidad en lugar de en función de las revoluciones, los cambios deberían realizarse (como se muestra en la figura 2.6):

- A 2ª marcha a los 2 segundos o 6 metros recorridos.
- A 3ª marcha a partir de 30km/h.
- A 4ª marcha a partir de 40km/h.
- A 5ª marcha a partir de 50km/h.

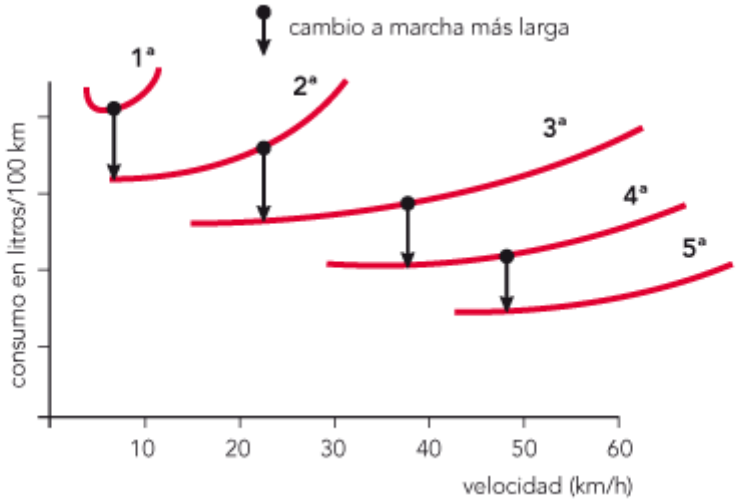

Figura 2.6. Gráfico sobre el cambio de marchas en función de la velocidad

(Cecu, 2015)

2.2. Simuladores de conducción

En la actualidad existen una serie de simuladores de conducción, de los cuales algunos incluso podrían ser utilizados para aprender y/o mejorar la conducción y en el caso de aprender, poder tener más facilidades a la hora de ir a la autoescuela.

2.2.1. City Car Driving

City Car Driving es un simulador de conducción que proporciona a sus usuarios una experiencia de conducción tanto por ciudad como por campo, y posibilita esta en diferentes condiciones. Su objetivo es una educación en el ámbito de las normas de tráfico y la seguridad vial. Este contiene una gran variedad de situaciones de tráfico diferentes, además de ejercicios avanzados que serán necesarios para pasar el examen de conducir.

Su sistema de control de las normas de circulación avisa al conductor en el caso de cometer alguna infracción además de proporcionar consejos sobre cómo actuar ante ciertas situaciones. Estas infracciones cometidas pueden ser revisadas posteriormente en "Estadísticas". En la figura 2.7 se observa una imagen de ejemplo sobre cómo es la conducción en este simulador.

Utiliza la física de automóviles avanzados para lograr una sensación realista y un motor de renderizado de alta calidad para obtener también un realismo gráfico.

Los objetos 3D, la física y otros archivos específicos están abiertos para modificación y permiten a los usuarios añadir sus propios modelos de coches en el simulador, modificar la física de los coches, etc. Por tanto se trata de un simulador muy personalizable y expansible.

(Forward Development, 2015)

Figura 2.7. Simulador City Car Driving

2.2.2. DriveSim

Drivesim es un simulador especialmente diseñado para impartir formación vial práctica en autoescuelas o centros de formación. Permite una conducción realista en un entorno seguro. Como otros simuladores, incluye tráfico real y peatones, además de permitir elegir diferentes configuraciones horarias o climatológicas. Los coches también se pueden configurar cambiando la tracción, el tipo de marchas, la potencia, etc. Podemos observar un ejemplo de cómo es este simulador en la figura 2.8.

Se compone de diferentes módulos:

- Formación inicial en pista: primera toma de contacto del alumno con el vehículo para manejar los diferentes controles del coche y poner en práctica lo aprendido teóricamente.
- Formación vial básica: se compone de ejercicios cortos mediante los cuales el alumno aprende maniobras específicas, como pueden ser adelantamientos o rotondas.
- Formación vial avanzada: pone en práctica los conocimientos adquiridos en los módulos anteriores. Facilita la circulación por diferentes tipos de vías (urbanas, autovías, etc.).
- Conducción eficiente: enseña a conducir de manera que se produzca un menor consumo del automóvil y una menor emisión de CO_2, y muestra las diferencias presentes en estos aspectos en función de algunos parámetros del coche como podría ser la presión de las ruedas.

(DriveSim, 2015)

Figura 2.8. Simulador DriveSim

2.2.3. Driver Test Pro

Driver Test Pro es un simulador 3D que permite a sus usuarios desarrollar habilidades de conducción y seguridad vial. Proporciona una experiencia real en un ambiente seguro. Incluye diferentes ejercicios interactivos donde se practica el conocimiento de los controles, incorporación al tráfico, intersecciones, prioridades, rotondas, interactividad con otros tipos de vehículos, entre otros. Una imagen de ejemplo de este simulador se muestra en la figura 2.9.

(Airsoft, 2015)

Figura 2.9. Simulador Driver Test Pro

2.2.4. STISIM Drive

STISIM Drive es un simulador de conducción programable y completamente interactivo. Este simulador ha sido utilizado por universidades, centros médicos y centros de formación. Un ejemplo de escena en este simulador se muestra en la figura 2.10.

Presenta diversas situaciones de conducción y entornos, además de recoger los datos del usuario durante la conducción (estado del vehículo, tiempos, distancias). Permite además modificar o incluso crear nuevos escenarios. Los coches tienen opción de cambio automático o manual. Presenta peatones y coches con inteligencia artificial, edificios, intersecciones, etc.

Es posible añadir un módulo adicional, el cual contiene funcionalidades como añadir retardos realistas a las entradas del conductor para simular las respuestas de este cuando se encuentra en malas condiciones. Simula situaciones en las que el conductor hubiese tomado alcohol, en función de la cantidad tomada respecto al cuerpo del usuario, produciendo efecto túnel cuando estos niveles de alcoholemia son bastante altos.

Este simulador permite al usuario interactuar con él mediante el lenguaje de programación C++ para crear aplicaciones personalizadas o añadir funcionalidades. También se podría utilizar algún otro lenguaje como C o Visual Basic.

(STISIM Drive, 2015)

Figura 2.10. Simulador STIMSIM Drive

2.3. Contribución de los simuladores de conducción a la seguridad vial

Muchas personas que se están iniciando en la conducción tienen miedo a conducir en una gran ciudad con mucho tráfico y cruces difíciles. Los simuladores de conducción pueden ayudar a los usuarios a superar este problema y a obtener una destreza a la vez que ciertas nociones de conducción, de manera que cuando se enfrenten a ello en una carretera real, esto pueda ser más fácil. Los usuarios pueden acostumbrarse a los controles del coche mediante este tipo de simuladores, manejándolos con pedales, volante y cambio de marchas (utilizando algún periférico similar al Logitech G27), al igual que lo harían en un coche real.

Estos simuladores además de permitir una práctica en la conducción libre de riesgos, aportan la posibilidad de repetir situaciones que puedan presentar dificultades tantas veces como sean necesarias. Además permiten hacerlo en un entorno bastante personalizable. También permiten replicar situaciones de la vida real que sería prácticamente imposible obtener a propósito en la carretera, de forma que el abanico de situaciones que se puede plantear es mucho más amplio, y los usuarios estarán mejor preparados ante situaciones que se le pueden presentar en un futuro.

Debido a la gran variedad de posibilidades sobre las situaciones a simular, podemos probar muchas cosas diferentes en un ámbito controlado, seguro y repetible. Con ellos podemos probar sin ningún peligro los efectos producidos por la privación del sueño o la fatiga, la ingesta de agentes farmacológicos o alcohol, la manifestación de alguna enfermedad, la geriatría, efectos de un derrame cerebral, entre otros factores.

Esto contribuye a la seguridad vial, ya que facilita el conocimiento de en qué condiciones se debe o no poder conducir.

Otra ventaja de los simuladores es la posibilidad de recopilación de datos. Tanto para el aprendizaje en la conducción como para estudios acerca de los efectos que pueda provocar el estado de cierto conductor, la obtención de datos es muy importante, ya que con esta podemos revisar posteriormente con detalle todo lo ocurrido durante la simulación, y analizar los posibles fallos y problemas. En el caso de la utilización de estos simuladores para la enseñanza, mediante estos datos se puede proporcionar retroalimentación al conductor de forma que pueda corregir todo lo que no haya hecho correctamente. Para el caso de la investigación clínica estos datos sirven para observar en qué casos un conductor estaría en condiciones de poder conducir y en cuáles sería peligroso.

3. Tecnologías

En este apartado trataremos acerca de las tecnologías disponibles para utilizar al crear un simulador de conducción. Para ello necesitamos un motor de juegos y un programa de modelado 3D. A continuación veremos cuáles son las opciones posibles que existen y luego elegiremos las más adecuadas para este trabajo.

3.1. Alternativas tecnológicas

Aquí se comentarán y estudiarán las características diferentes opciones tecnológicas, tanto de motores de juegos como de programas de modelado 3D, para posteriormente compararlas y elegir las más adecuadas para nuestro fin.

3.1.1. Motores de juegos

Hay una gran variedad de motores de juego disponibles en la actualidad, y aunque en esta sección solo hablaremos sobre algunos de los más importantes actualmente, existen muchísimos más. Entre otros podemos encontrar los siguientes: Unreal Engine 4, UDK (Unreal Engine 3), CryEngine 3 SDK, Source Engine, Unity3D, Leadwerks, Torque3D, Blender, Neoaxis, C4 Engine, Shiva 3D, Panda 3D, Eshencel Engine, iDTech 4, Ogre3D, Rage e Irlicht Engine.

Muchos juegos exitosos han sido realizados mediante algunos de estos motores, incluso con los que no trataremos en detalle a continuación. Por ejemplo con Source Engine se han realizado juegos como Counter Strike y Left 4 Dead 2; utilizando CryEngine 3 se hicieron otros como son Crysis 2 y Crysis 3, etc.

3.1.1.1. Unity 3D

Unity 3D es una plataforma de desarrollo flexible y potente para la creación de juegos multiplataforma 3D o 2D y experiencias interactivas. Realmente tiene dos motores de física por separado, uno para trabajar en 2D y otro para 3D.

Figura 3.1. Plataformas en las que puede trabajar *Unity*

Tiene la posibilidad de moverse libremente entre 21 plataformas (cuyos logos se muestran en la imagen 3.1), incluyendo las más recientes como son WebGL y Oculus Rift. Este número de plataformas que soporta se encuentra en aumento continuamente. En cuanto a las plataformas para móvil podemos encontrar iOS, Android, Windows Phone 8, BlackBerry 10 y Tizen. Para escritorio tenemos Windows, Windows Store Apps, Mac, Linux/Steam OS. Para web estaría Web Player y WebGL. Para consolas tendríamos PlayStation 3, PlayStation 4 y Morpheus, PlayStation Vita, Xbox One, Xbox 360 y Wii U. Por último para VR tendríamos Oculus Rift y Gear VR. A parte de estos

también tenemos para Android TV y Samsung Smart TV. *Unity 3D* es una herramienta de uso intuitivo y muy personalizable. Se pueden añadir tus propias herramientas por comodidad y productividad y los tiempos de compilación son muy rápidos. Es un motor de juegos con una gran calidad en sus optimizaciones, y con mucha velocidad y eficacia en los flujos de trabajo. Su calidad visual es muy buena. En cuanto a las herramientas extensibles, hay una gran cantidad de ellas incluso gratuitas.

La forma de importar archivos es muy sencilla, basta con arrastrarlos a *Unity* y se importarán de forma automática. Además soporta una gran variedad de formatos, que se encuentran indicados en la figura 3.2.

Figura 3.2. Formatos soportados por Unity

La herramienta *Profiler* ayuda a optimizar el juego. En ella se informa de cuánto tiempo se dedica en las distintas áreas del juego, como pueden ser el renderizado, la animación o la lógica del juego. Puede registrar los datos de rendimiento, y allí ver las áreas que toman más o menos tiempo.

Unity tiene componentes que facilitan la simulación física, ya que para tener un comportamiento físico convincente, un objeto en un juego debe acelerar correctamente y ser afectado por las colisiones, la gravedad y otras fuerzas. Esto hace que el comportamiento de estos objetos sea más realista.

En cuanto al Scripting, podemos programar en C#, JavaScript o Boo. Esta parte es muy importante ya que para cualquier juego necesitamos aunque sea mínimamente algún script. Estos sirven para controlar el comportamiento físico de los objetos, crear efectos gráficos, etc.

Unity además está diseñado para ayudar a las necesidades de colaboración de equipos profesionales que trabajan de forma conjunta de manera remota, con la licencia de *Unity Professional Edition*, de manera que aumente su productividad.

Contiene un *Asset Store* donde se pueden encontrar muchas cosas tanto gratuitas como de pago. Esto pueden ser animaciones, modelos 3D, aplicaciones, audio, proyectos completos, sistemas de partículas, scripts, texturas, materiales, etc. Además cuenta con una gran cantidad de recursos que pueden ayudar para el aprendizaje del manejo de este motor de juegos. Cuenta con tutoriales en los que podemos encontrar videos y artículos con los que aprender sobre muchos temas, documentación en la que se incluye un manual completo y la referencia de todos los componentes de *Unity*, y una comunidad en la que hay una gran cantidad de preguntas en foros y en la que se puede pedir ayuda a otras personas de esa comunidad con problemas que puedan surgir.

Podemos ver un ejemplo de proyecto en *Unity* en la figura 3.3.

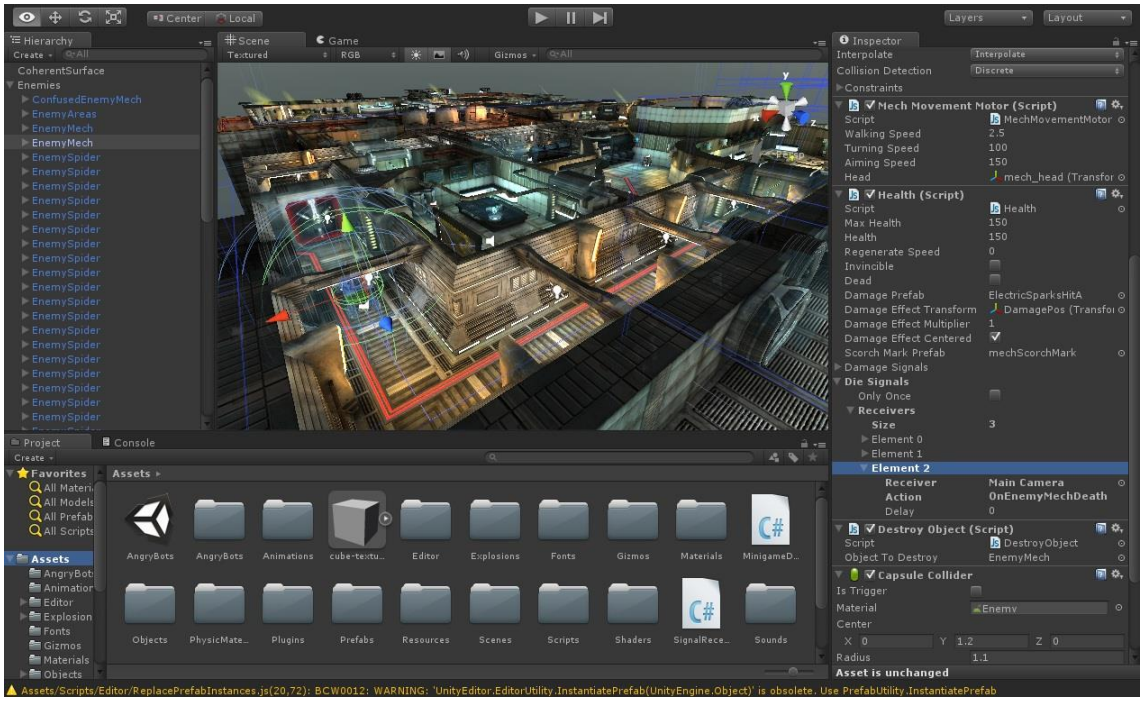

Figura 3.3. Proyecto en Unity 3D

En cuanto a los requisitos del sistema, para desarrollo necesitamos como sistema operativo Windows XP SP2+, 7SP1+, 8 o Mac OS X 10.8+. En cuanto a la GPU, capacidades de tarjeta de video con DX9 (modelo shader 2.0). Dependiendo la complejidad de los proyectos existen requisitos adicionales. Para ejecutar juegos de Unity, necesitamos como sistema operativo Windows XP+, Mac OS X 10.7+, Ubuntu 12.04+ o StreamOS. La tarjeta de video debe tener unas capacidades DX9. La CPU debe ser compatible con el conjunto de instrucciones SSE2. El reproductor web es compatible con Internet Explorer, Google Chrome, Firefox, Safari y otros. En cuanto a los sistemas operativos móviles, tenemos que iOS requiere la versión 6.0 o posteriores, Android requiere OS 2.3.1. o posterior, ARMv7 (Cortex) CPU o Atom CPU, OpenGL ES 2.0 o posterior, Blackberry requiere OS 10 o posterior y WebGL la versión de escritorio de Firefox, Chrome o Safari.

3.1.1.2. Source engine

Source es un motor de videojuegos desarrollado por *Valve*. Cuenta con una tecnología rápida, fiable y flexible, que ayuda a reproducir incluso las escenas más complejas de una manera rápida y eficiente. Tiene la posibilidad de trabajar tanto con *Direct3D* como con *OpenGL*. Está escrito íntegramente en C++. Este motor es multiplataforma siendo sus objetivos el PC, tanto *Windows* como *Mac OS* y *GNU/Linux*, y la *Xbox*, tanto la original como *Xbox 360*, además de *Playstation 3*. Utiliza procesadores multi-núcleo para ofrecer experiencias de juego de alto rendimiento.

En cuanto al sistema de renderizado, *Source* tiene soporte para *shaders HLSL*. Además cuenta con una librería de *shaders* avanzada, puede usar librerías de *shaders* de Valve o nuevos algoritmos definidos por el usuario. Tiene gestión automática de nivel de detalle (LOD), que permite alcanzar el máximo rendimiento, y enmascarado de texturas independientemente de su resolución.

Figura 3.4. *Motion Blur* en *Counter Strike*

Respecto a la iluminación, contiene mapas de iluminación que codifican información direccional de la iluminación, de modo que esta se combina con mapas de relieve dando una iluminación precisa y con auto-sombreado. También permite iluminación indirecta, sombras dinámicas con una elevada calidad, mapeado de profundidad de sombras, iluminación de bordes y renderizado avanzado de materiales.

Es posible crear efectos con partículas, humo, chispas, sangre, etc. La optimización multinúcleo mejora el rendimiento de renderizado de efectos de partículas. Su editor permite previsualizar los efectos de un modo interactivo. Incluye *Motion Blur* en tiempo real (rastro dejado por los objetos en movimiento, como se puede observar en la figura 3.4). En el agua se pueden obtener efectos muy realistas mediante reflejos y refracción.

Mediante su sistema de materiales, en cada material se especifica el material y la textura de cada objeto, además de cómo debe reaccionar ese objeto cuando se rompe, cómo debe sonar, su masa, etc. Incluye mapas de relieve en los que se hace auto-sombreado y mapas de arrugas aplicables tanto a personas como a ropa. Además presenta grandes facilidades a la hora de utilizar múltiples texturas combinadas.

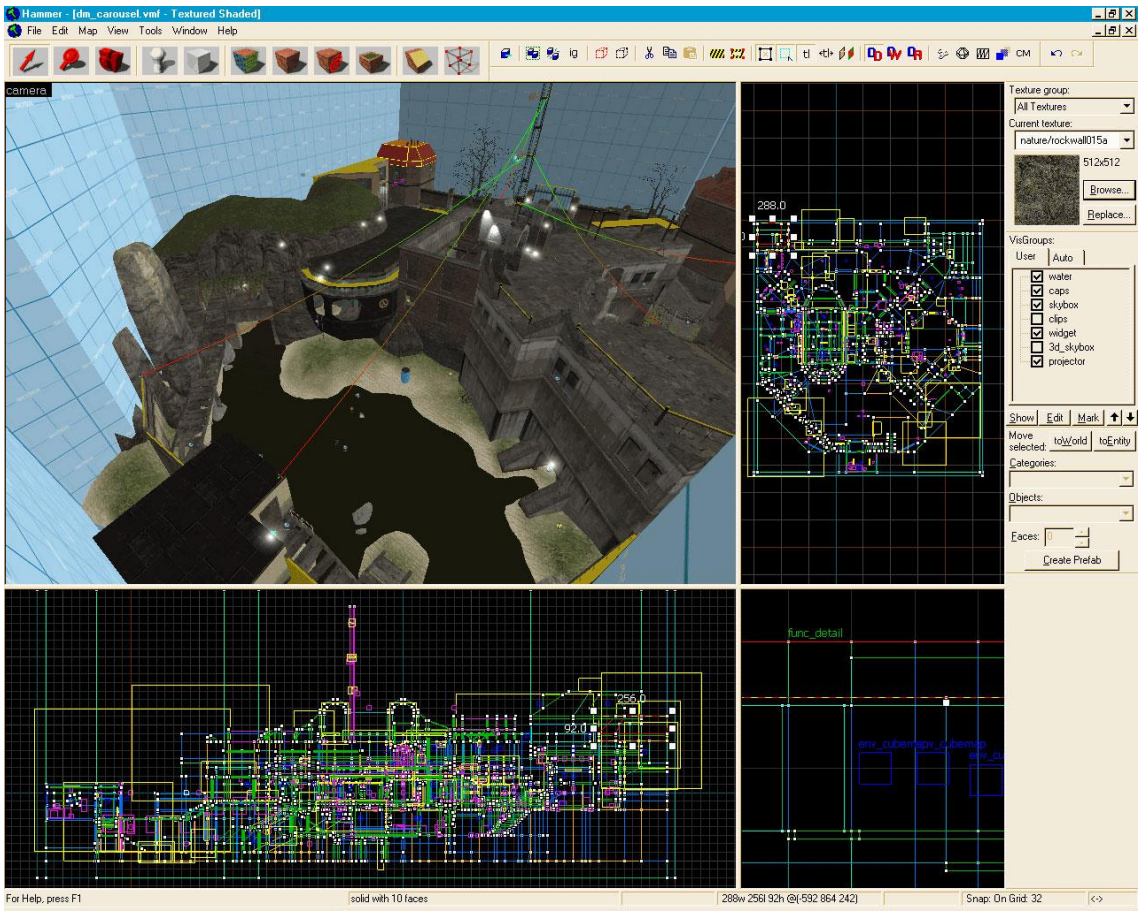

Figura 3.5. Valve Hammer Editor

Sus características de modelado y animación permiten realizar personajes muy realistas, simulando la musculatura para representar los movimientos y las emociones del personaje. Permite la creación de ojos y labios con gran precisión. Los modelos pueden ser articulados de una manera fluida, logrando tener incluso movimientos faciales muy avanzados (utilizando la herramienta *Faceposer*). Las animaciones se pueden mezclar y sintetizar a partir de varias partes individuales.

Para la creación del entorno, es posible esculpir valles o colinas entre otros entornos naturales. Además es posible añadir "cajas" que representan el cielo (*skyboxes*). La herramienta *Valve Hammer Editor*, mostrada en la figura 3.5, se utiliza para crear mapas en un entorno de desarrollo intuitivo. Ahí se pueden colocar y editar *scripts* de modelos además de compilar y ejecutar niveles. También incluye herramientas para construir maquinaria detallada, vehículos con parámetros ajustables (caballos, velocidad máxima, material de las llantas, fricción del neumático,…), objetos deformables, cuerdas, cables, etc.

(Valve, 2012)

3.1.1.3. Unreal Engine 4

Unreal Engine 4 es un conjunto de herramientas para desarrollo de juegos, hecho por desarrolladores de juegos. Ha sido creado en C++, y permite el acceso a su código fuente, de modo que es posible personalizar y ampliar las herramientas de su editor, la física, audio, animación, renderizado, además de la interfaz de usuario. *Unreal* soporta *DirectX* 11 y 12, permitiendo muchas luces dinámicas por la escena y otras características de renderizado. Podemos ver un ejemplo de proyecto en *Unreal Engine* en la figura 3.6.

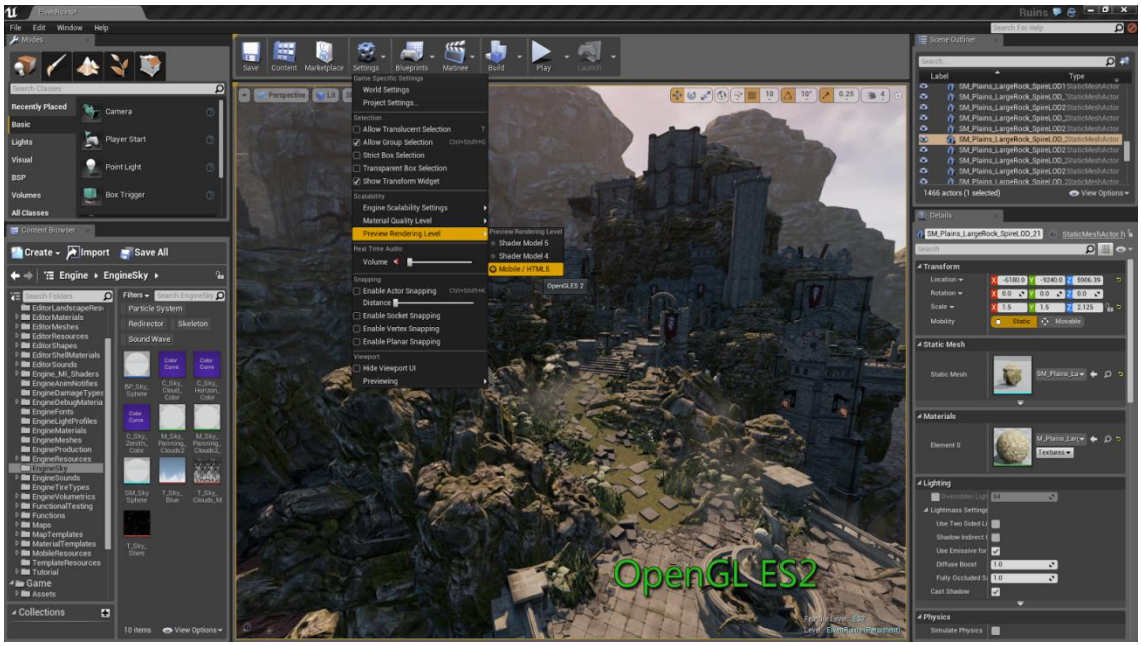

Figura 3.6. Entorno de *Unreal Engine*

Su editor *Cascade VFX*, permite una simulación detallada de fuego, nieve, polvo, entre otros efectos dinámicos de partículas. Es posible obtener muy buenos efectos de sombreado y acabados de materiales. Mediante *Blueprint Visual Scripting* es posible crear contenido jugable sin llegar a escribir ninguna línea de código, haciendo esto de una manera visual. También cuenta con un navegador de contenido para importar, organizar, buscar, etiquetar, filtrar y modificar "assets".

En cuanto a animación, la herramienta *Persona Animation* permite editar esqueletos, crear secuencias de animación y modificar las propiedades de su física. Con el conjunto de herramientas *Matinee Cinematics* es posible realizar secuencias de juego dinámico y películas.

Es posible crear ambientes con su sistema de paisaje, pintando y borrando componentes del terreno con facilidad.

Se puede navegar por las funciones de C++ directamente desde los personajes y objetos del juego, accediendo desde ahí a las líneas de código fuente. Además el código puede ser cambiado mientras el juego está en ejecución para ver esos cambios reflejados inmediatamente en el juego, sin necesidad de pausarlo. Mientras esté el juego en ejecución es posible dejar de verlo desde la cámara del jugador para observarlo desde otro punto de vista y ver que puede estar comportándose de un modo incorrecto.

Unreal ofrece integraciones con muchas de las tecnologías *middleware* líderes en la industria, entre las que se encuentran *NVIDIA*, *PhysX*, *Autodesk Gameware*, *Enlighten*, *Umbra* y *Oculus VR*.

Para el aprendizaje y/o consulta de información cuenta con una gran variedad de tutoriales hechos tanto mediante videos como con documentación escrita, una comunidad en la que se pueden plantear dudas y un lugar (*Marketplace*) donde es posible comprar proyectos hechos.

Respecto a los requisitos del sistema para desarrolladores que utilicen este motor, es necesario un sistema operativo como Windows 7 de 64 bits o Mac OS X 10.0.2. o posteriores. Además es necesario un procesador de 4 núcleos, ya sea Intel o AMD, y 8GB de memoria RAM. Respecto a tarjeta gráfica es necesario que sea compatible con DX11. Este motor de juegos podría funcionar en ordenadores con características inferiores, pero no tendría un buen rendimiento.

(Epic Games, 2015)

3.1.1.4. CryEngine 3

CryEngine es un motor de juegos desarrollado por *Crytek*. Tiene soporte para PC, *Playstation 4*, *Xbox One*, *Wii*, *Android e iOS*. Cuenta con el editor *sandBox*, que permite separar en capas un nivel de juegos y permite que trabajen varias personas en la misma capa. Cuenta con la funcionalidad WYSIWYP (*What You See Is What You Play*).

Su sistema de renderizado es muy rápido. Cuenta con un sistema de edición visual con una interfaz bastante intuitiva. Su solución para la iluminación proporciona buenos resultados, proporcionando oscurecimiento en las zonas que correspondería. Además permite el cambio de la iluminación a lo largo del día, rotando el punto desde el que se proporciona la luz, y la intensidad de esta en función del tiempo en que se va desarrollando alguna misión. Al trabajar con varios núcleos, el trabajo es repartido y por tanto se obtiene un buen rendimiento.

Permite generar vegetación muy realista, pudiendo crear bosques, campos o selvas; además de sistemas de partículas que formen fuego, humo o explosiones. Se pueden presentar muchos otros como nubes, gases, etc. de una forma muy realista. Además presenta facilidades a la hora de trabajar con automóviles, permitiendo un control intuitivo sobre ellos, los daños ocasionados o los pasajeros. Es posible generar grandes extensiones de terreno que pueden ser editadas posteriormente para crear más detalles. El agua que se obtiene tiene un gran realismo, pudiendo variar su profundidad o su comportamiento en función de la acción del viento o al estar a la orilla de la costa. Además aplica los efectos tanto de reflexión como de refracción. Mediante su método de reflexiones locales en tiempo real, cualquier tipo de superficie puede dar reflejo en sus alrededores.

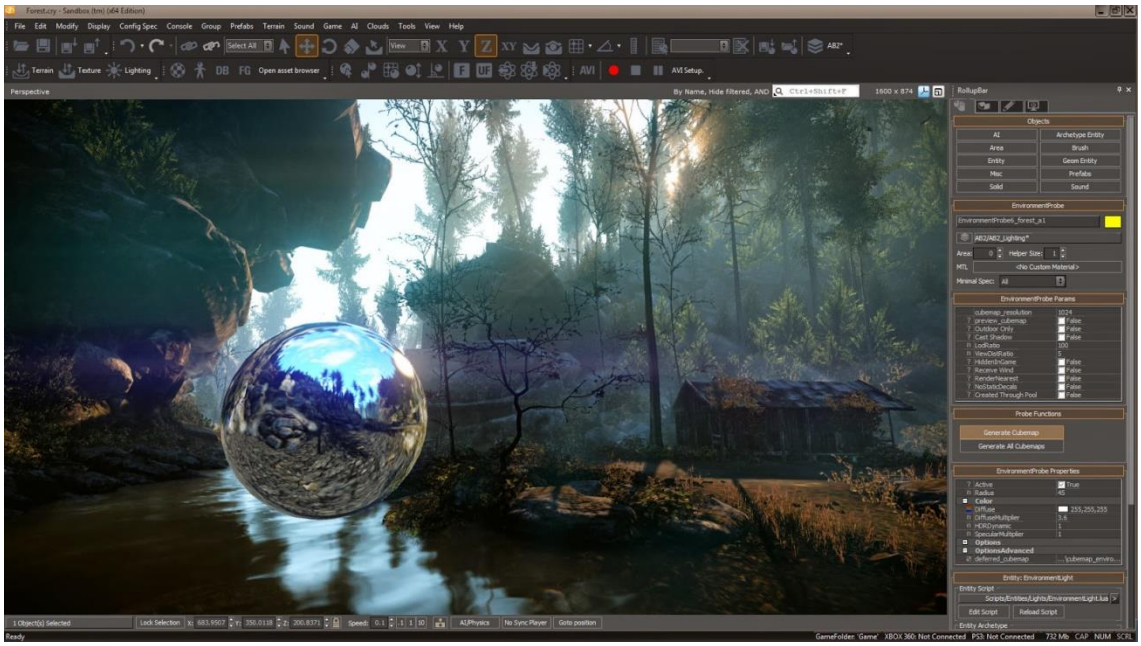

Figura 3.7. Proyecto en CryEngine

Los personajes son animados mediante *Sandbox 3*, donde podemos ver y editar los estados y las transiciones entre las animaciones del personaje. Pueden seguir caminos o subir y bajar pendientes, además de reaccionar ante ciertos cambios en el entorno. Su editor facial permite analizar audio, para de este modo ver la posición de los labios que deseamos, tomando la forma que sería necesaria para hacer ese sonido. Se les puede configurar para que muestren entre sí hostilidad, amistad u otros sentimientos, mediante características faciales.

En cuanto a la física, se puede trabajar con fuerzas como son la gravedad, las corrientes de viento, explosiones, colisiones y fricción. Es posible realizar efectos como podría ser la ondulación de una bandera, o la física de una cuerda.

Mediante *Simple DirectMedia Layer Mixer* se pueden crear prototipos de audio de una forma muy rápida, pudiendo arrastrar los sonidos a eventos, estados, etc.

Es posible trabajar con C++ o LUA. Además cuenta con un sistema visual de scripting potente que permite crear la lógica del juego de una forma intuitiva, sin necesidad de escribir nada de código.

CryEngine cuenta con documentación online para poder consultar cualquier información acerca de cómo utilizar este programa. Además cuenta con una comunidad en la que se pueden consultar los temas ya comentados por los foros o plantear nuevas preguntas.

Podemos ver un ejemplo de proyecto en este motor de juegos en la figura 3.7.

Respecto a los requisitos del sistema para desarrollar juegos con este motor, es necesario tener como sistema operativo Windows Vista, Windows 7 o Windows 8. Respecto a la CPU, los requisitos mínimos son un Intel Core Duo 2GHz o AMD Athlon 64 X2 2GHz. Es altamente recomendable un procesador multinúcleo. Además es necesaria una memoria RAM de 4GB, aunque es preferible que sea de 8GB. Por último es necesaria como mínimo una tarjeta gráfica como NVIDIA series 400 o Radeon HD 600 Series.

(Crytek, 2015)

3.1.2. Programas de modelado 3D

Aquí se tratarán diferentes programas de modelado 3D y sus características, para más adelante poder elegir cuál se adecua más a lo que deseamos.

3.1.2.1. 3ds Max

3ds Max es un software de modelado, animación y renderización en 3D perteneciente a *Autodesk*. Cuenta con soporte web técnico directo, soporte prioritario en los foros y licencias flexibles. Su entorno de trabajo se muestra en la figura 3.8.

A continuación se comentarán varias de las características que tiene respecto a animación, texturización y modelado, renderización, dinámica y efectos, interfaz de usuario, flujo de trabajo y entorno de producción.

En cuanto a animación 3D, cuenta con un secuenciador de cámara que permite cortar entre varias cámaras, recortar y reordenar clips, manteniendo intactos los originales; tiene un modificador Skin con el que crear personajes con mejor piel y deformaciones más realistas; permite generar multitudes realistas ya sea en movimiento o estáticas mediante la función *Populate*, permitiendo generar comportamientos muy convincentes al andar, correr, girar, sentarse, etc.; contiene herramientas para la manipulación de los personajes, de forma que se puede modificar su esqueleto; las trayectorias de animación se ven y editan directamente en la ventana gráfica; contiene modificadores que permiten simular efectos como podrían ser los de un fluido.

En relación a la texturización y el modelado, *3ds Max* tiene compatibilidad con *OpenSubdiv*, de forma que se puede obtener un mayor rendimiento en la ventana gráfica para mallas con muchas subdivisiones; cuenta con muchas opciones de sombreado e

interoperabilidad de sombreadores con *Maya* y *Maya LT*; permite crear una amplia gama de materiales; se pueden utilizar nubes de puntos para crear modelos precisos a partir de referencias reales; contiene herramientas para facilitar la colocación de elementos en la escena en relación con otros y para crear bordes biselados de forma que se puedan eliminar los pinzamientos; opciones de mapeado creativo de texturas (mosaico, simetría, estampación, …); permite cargar gráficos vectoriales como mapas de texturas y renderizarlos de modo que los gráficos siempre sean nítidos y claros; además contiene funciones de modelado basadas en polígonos, splines y NURBS muy eficaces para crear ciertos tipos de objetos.

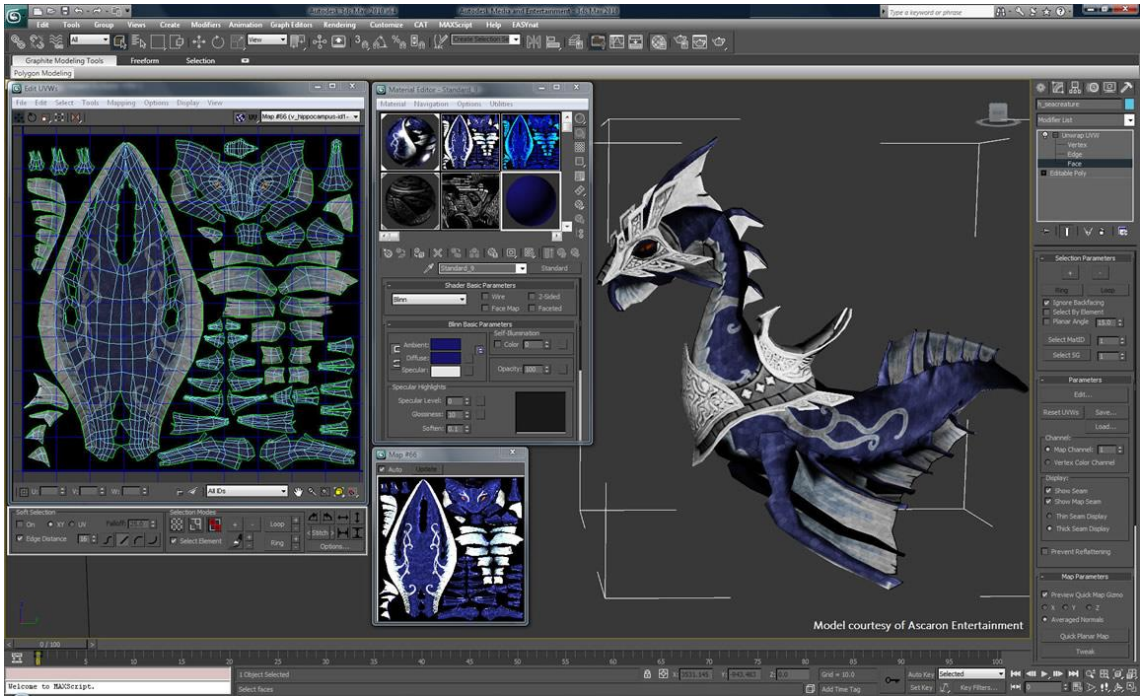

Figura 3.8. Proyecto en 3ds Max

Respecto a la renderización en 3D, *3ds Max* incorpora la renderización de *Autodesk 360* para los clientes de *Subscription*, que utiliza la computación en la nube, de forma que no se necesite emplear la capacidad de procesamiento del ordenador utilizado; su cámara física incorpora velocidad de obturación, apertura, profundidad de campo y exposición, de forma que tiene una mayor similitud con las cámaras reales; tiene compatibilidad con las mejoras de *Iray* y *mental ray*, que permite la renderización de imágenes fotorealistas; se puede mejorar la renderización con *ActiveShade*; se puede suavizar el efecto de las texturas para trabajar con menos ralentizaciones; utilizando la grabadora se pueden capturar, grabar y editar diferentes momentos de la escena, y se puede renderizar solo la parte modificada sin volver a hacerlo para toda la escena; cuenta con técnicas avanzadas de gestión de escenas y análisis multiproceso que agilizan el trabajo.

En cuanto a la dinámica y efectos, pueden ser creados efectos como agua, fuego o nieve, entre otros; se pueden obtener simulaciones de esqueletos con *mRigids*, de ropa y tejido con *mCloth*, y de fuerzas naturales con *mParticles*; efectos para el cabello (crear

rizos, ondas, cardar,…) y la piel; se pueden realizar simulaciones y análisis de iluminación con *Exposure*.

Por último, en lo relacionado con la interfaz de usuario, el flujo de trabajo y la producción, facilita la creación de nuevas herramientas que posteriormente pueden ser compartidas con el resto de usuarios; ayuda a la colaboración entre equipos mediante flujos de trabajo de animación no destructivos; su explorador de escenas y administrador de capas permiten trabajar en escenas complejas de una forma más sencilla; su espacio de trabajo *Design* facilita el acceso a las herramientas; permite la configuración de plantillas para tener una configuración de inicio predeterminada; los datos se pueden organizar en capas anidadas para una mayor facilidad en el trabajo; la interfaz de usuario es configurable, pudiendo separar un espacio para el modelado y otro para la animación con múltiples opciones diferentes; cámara flexible que permite hacer zoom para colocar los elementos fácilmente en el lugar exacto; puede importar datos desde diferentes orígenes; varias personas pueden trabajar en la misma escena simultáneamente; se pueden ampliar sus elementos mediante el kit de desarrollo de software *SDK*; además *3ds Max* se puede ampliar y personalizar mediante el lenguaje de programación *Phyton*; las funciones de *Civil View* permiten fácilmente transformar el espacio civil en un modelo 3D; permite interoperabilidad con *Autocad* y *Revit*; admite *DirectConnect* que permite intercambiar datos de diseño industrial con ingenieros que utilizan *CAD*.

3ds Max cuenta con herramientas de aprendizaje tanto para comenzar a trabajar con este programa, como para personas más avanzadas en su manejo que pueden buscar respuestas a problemas que encuentren en el desarrollo de su proyecto.

(Autodesk, 2015)

3.1.2.2. Blender

Blender es un software libre y de código abierto, para cualquier tipo de uso, que cuenta con un motor de renderizado que ofrece una representación muy realista, y con licencia permisiva para vincular con cualquier software externo. En la figura 3.9 observamos un proyecto en el entorno de trabajo de *Blender*.

Cuenta con una amplia gama de herramientas para crear modelos, que pueden ser utilizadas de una manera muy rápida mediante atajos de teclado. Además se pueden crear herramientas personalizadas y complementos, mediante el lenguaje de programación *Phyton*. Su interfaz es flexible de forma que el diseño se podrá personalizar completamente.

Su motor de renderizado permite crear materiales realistas y simular objetos reales como puede ser el vidrio. En cuanto a los esqueletos, posee herramientas que permiten la modificación de los huesos de una manera sencilla. Además contiene un conjunto de herramientas de animación con la que se pueden crear animaciones de personajes u otros elementos, incluso animaciones no lineales para movimientos independientes. Estas animaciones pueden incluir sonido.

Para el modelado de los objetos, cuenta con diferentes pinceles para esculpir, e incluso herramientas para esculpir topologías dinámicas. Sobre estos modelos se aplican texturas que pueden ser pintadas directamente sobre el modelo o utilizar texturas de imagen. Es posible realizar simulaciones muy realistas de fluidos, humo, cabello, tela, objetos rígidos y partículas como podrían ser la lluvia o las chispas.

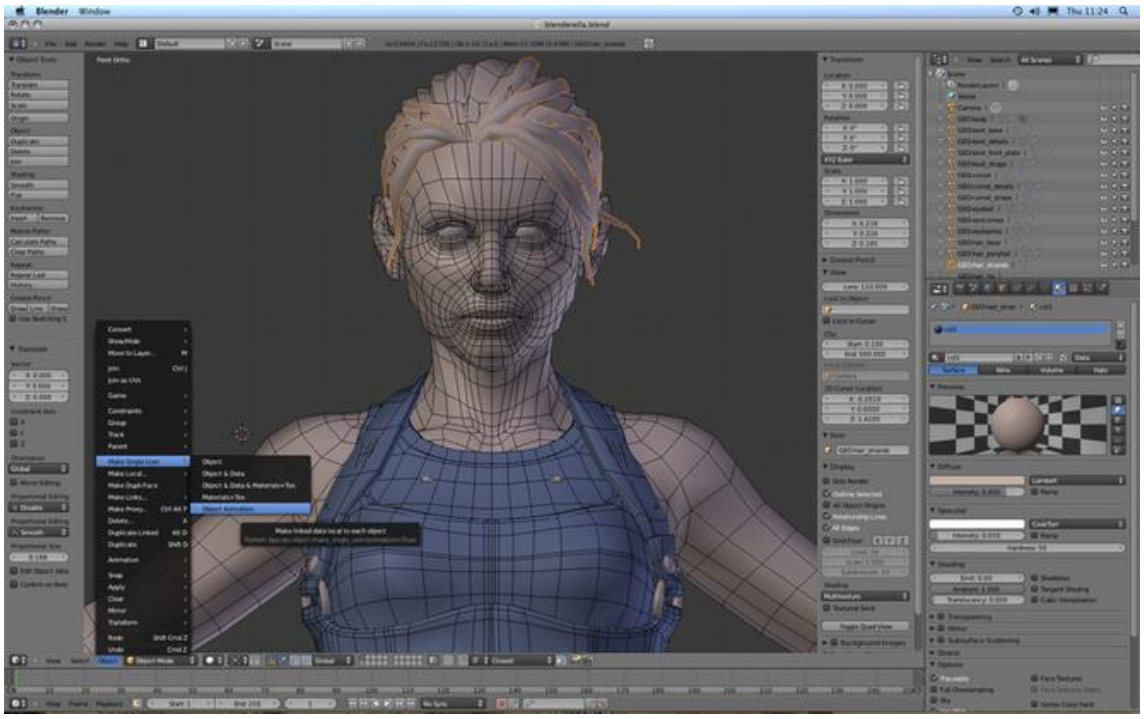

Figura 3.9. Proyecto en Blender

No es necesario exportar nada a programas ajenos, se puede hacer todo sin salir del programa. *Blender* es un motor de juego completo, por lo que se puede crear y codificar la lógica de juego en él. Los scripts se hacen mediante *Phyton*. El juego podría ser reproducido dentro del propio programa. Además permite que la cámara persiga a un objeto deseado.

Este programa cuenta con una gran variedad de extensiones creadas por su comunidad de desarrolladores, que pueden ser activadas o desactivadas fácilmente. Estas son por ejemplo para generar árboles, terreno o nubes, romper objetos, herramientas para impresión 3D, etc.

También incluye un editor de video que permite cortar y juntar videos, mezcla de audio, sincronización, control de velocidad y transiciones, entre otras funciones más avanzadas.

Permite la importación/exportación de muchos formatos diferentes. En cuanto a imagen los formatos son JPEG, JPEG2000, PNG, TARGA, OpenEXR, DPX, Cineon, Resplandor HDR, SGI Iris y TIFF. Los formatos de video soportados son AVI, MPEG y QuickTime (en OSX). Respecto a modelos 3D los formatos soportados son 3DS Studio (3DS), Collada (DAE), Filmbox (FBX), Autodesk (DXF), Wavefront (OBJ), DirectX

(X), Lightwave (LWO), Motion Capture (BVH), SVG, Standford PLY, STL, VRML, VRML97 y X3D. Además es consistente en todas las plataformas.

Blender cuenta con tutoriales online, con los que se pueden adquirir los conocimientos necesarios para trabajar con este programa.

(Blender, 2015)

3.1.2.3. Cinema 4D

Cinema 4D es un programa de modelado que permite realizar cualquier tipo de proyecto con bastante facilidad, ya que es un programa de fácil uso e intuitivo. En la figura 3.10 mostramos un ejemplo de proyecto en este programa.

Cuenta con herramientas para la creación de esqueletos y animaciones de los personajes creados. Tiene también herramientas para la edición del pelo, que permiten crearlo y editarlo de la forma deseada de una manera rápida.

Su motor físico hace sencillo realizar colisiones complejas e interacción entre objetos. El render en red permite aprovechar todos los ordenadores conectados a la misma red de forma que las animaciones pueden ser renderizadas más rápidamente.

Permite importar y exportar archivos de una gran variedad de formatos. Estos pueden ser modelados mediante herramientas de subdivisión de superficies, deformadores, modificadores, etc. Se pueden utilizar splines para extrusiones, recorridos y otras modificaciones. Contiene muchas soluciones de edición UV para el texturizado de los objetos, pudiendo obtener resultados de baja o alta resolución según deseemos. Los materiales tienen hasta 14 canales diferentes (cuyas bases suelen ser una textura o un shader), pudiendo crear otros personalizados. Para las texturas a utilizar *Cinema 4D* soporta los formatos más utilizados, entre ellos los archivos en capas PSD. Con todo esto es posible crear objetos con apariencias como podría ser el vidrio, la madera o el metal. Mediante sus pinceles podemos pintar en más de 10 canales de una vez, y mediante la tecnología *RayBrush* es posible ver los resultados en tiempo real. También se pueden pintar objetos o mallas mediante la herramienta *Pintado de Proyección*. Mediante *Proyection Man* se ha facilitado la creación de *matte painting digitales*, que son representaciones de un paisaje que crean una ilusión de un entorno que no existe en la vida real o que sería muy caro o imposible construirlo o visitarlo.

En cuanto a animación, es algo sencillo e intuitivo de realizar mediante este programa. Cualquier objeto puede hacerse animado haciendo “clic” sobre el círculo que se encuentra al lado de su nombre. Además se pueden realizar y modificar fotogramas clave fácilmente. Cuenta con una ventana *Línea de Tiempo*, la cual ofrece un gran control sobre las animaciones mediante pistas para cada parámetro animable, de forma que se puede organizar todo fácilmente. Además para poder ajustar más detalladamente la interpolación de los fotogramas, se puede utilizar el modo *F-Curves*. Con la animación no lineal se pueden construir complejas animaciones que contengan muchísimos fotogramas en complicadas jerarquías. También se pueden crear

animaciones de partículas que simulen viento, gravedad u otros muchos efectos. Respecto a la animación de personajes, cuenta con herramientas para mover los cuerpos rígidos, simular los efectos de la tela o el cabello, etc.

Para añadir el audio, se puede ver el gráfico del sonido junto a las líneas de tiempo de las animaciones para poder sincronizarlas entre sí fácilmente. Este audio además puede ser enlazado a objetos determinados.

Figura 3.10. Proyecto en Cinema 4D

Cinema 4D además cuenta con una gran cantidad de herramientas de iluminación. Es posible controlar el brillo, el color, el contraste, el tono y densidad de las sombras, etc. (incluso se puede medir en Lumen o Candelas). Mediante su flujo de trabajo lineal y la tecnología *HyperThreading* y *Multinúcleos* puede tardar poco tiempo en conseguir buenos resultados. Además se pueden crear archivos de color o sombras para ajustar posteriormente en el destino. Tiene exportación directa de sus capas a *Adobe Photoshop*, *Adobe After Effects*, *Final Cut Pro*, *Nuke*, *Shake*, *Fusion* y *Motion*. Se puede utilizar una iluminación global de forma que se obtiene un aspecto realista ya que la luz rebota en los objetos. Las cámaras que utiliza pueden tener efecto estéreo definiendo valores como la separación entre los ojos y la convergencia.

Este programa también cuenta con documentación para aprender a manejarlo, preguntas frecuentes con sus correspondientes respuestas y otros tipos de apoyo.

(Maxon, 2015)

3.1.2.4. Maya

Maya es un software perteneciente a *Autodesk*. Es un software de modelado, simulación, animación y renderizado en 3D. Un ejemplo de proyecto en Maya se muestra en la figura 3.11.

En cuanto a dinámica y efectos, cuenta con muchas herramientas y módulos de extensión con un amplio abanico de posibilidades, para poder hacer cualquier tipo de objeto que se desee muy realista. Por ejemplo, para añadir espuma y burbujas a líquidos de forma que podemos obtener mucho realismo y muchos detalles, además de poder controlar el comportamiento de los líquidos con una simulación en caché o con un objeto animado con lo que se pueden crear olas y otros muchos efectos. También se pueden crear efectos atmosféricos como humo y niebla y efectos de atmósferas, explosivos, líquidos viscosos, nubes, etc. Mediante *Maya NHair* se puede crear cabello además de otros efectos dinámicos. También es posible pintar, cortar, plegar o estirar ropa, entre otras opciones. Contiene dinámica de cuerpos rígidos y flexibles, y se pueden poner restricciones de movimiento a objetos como bisagras o muelles.

Respecto a la animación 3D, mediante su sistema multiproceso es posible distribuir el cálculo entre los núcleos y procesadores gráficos del equipo utilizado. Proporciona ayuda para detectar cuellos de botella en escenas y poder solucionarlos. Cuenta con un sistema de animación por capas no destructivo que funciona con atributos. Los personajes animados pueden adquirir un aspecto articulado y natural, y pueden ser enlazados entre sí. Estos pueden ser reutilizados al igual que las animaciones para ahorrar tiempo, transfiriendo información por ejemplo de su piel a otro modelo. Es posible presentar varias tomas de cámara en una secuencia de animación.

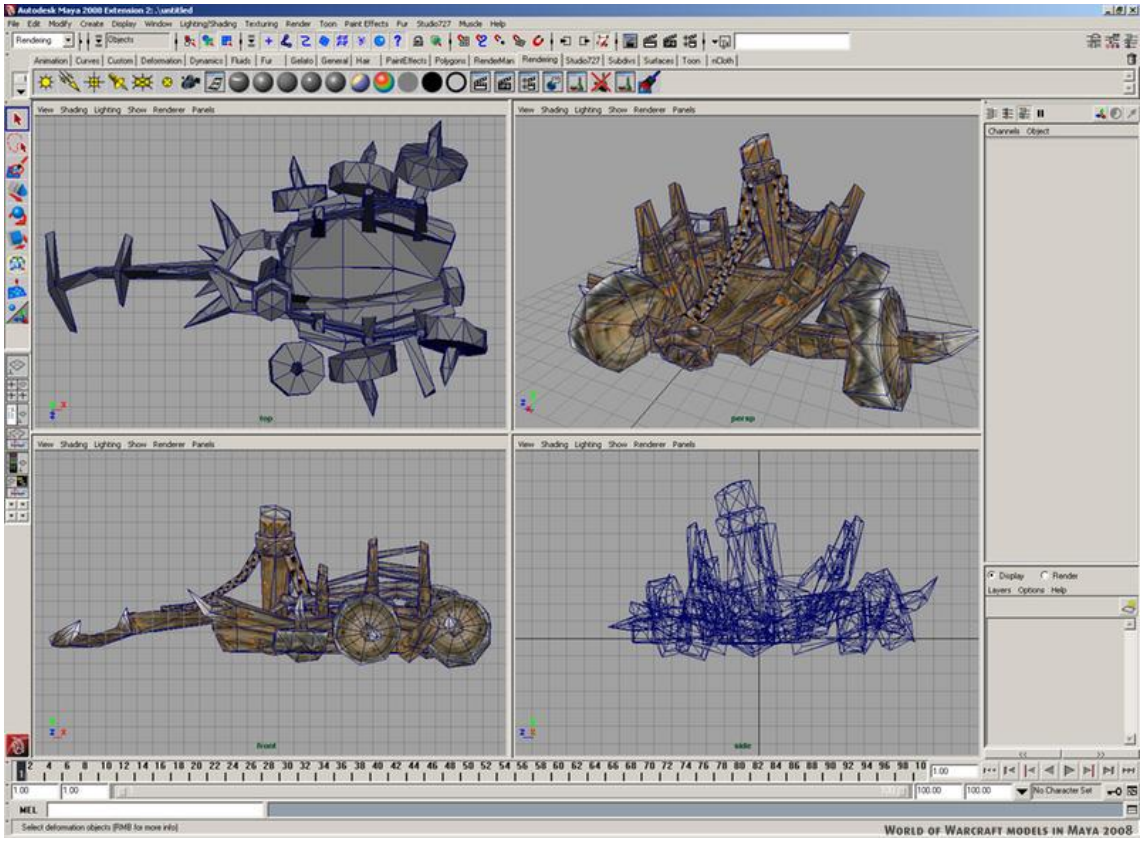

Figura 3.11. Proyecto en Maya

Para el modelado 3D cuenta con unas herramientas de esculpido que pueden ser manejadas de una forma intuitiva, y permitiendo hacer muchos detalles con una buena

resolución. Las mallas pueden ser optimizadas para crear deformaciones muy definidas, y los polígonos pueden ser modelados de un modo fiable, siendo sencillo por ejemplo crear biselados. *Maya* además tiene compatibilidad con *OpenSubdiv*, de forma que se pueden utilizar arquitecturas CPU y GPU paralelas para mejorar el rendimiento al dibujar. Un algoritmo de despliegue multiproceso y los flujos de trabajo de selección permiten crear y editar de una manera rápida mallas de UV complejas obteniendo unos resultados de alta calidad.

Es posible escribir módulos de extensión en *Maya Embedded Language* (MEL) o *Phyton*. Las aplicaciones individuales y módulos de extensión se pueden ejecutar desde su línea de comandos en C++, Phyton o .NET. Cuenta con herramientas de gestión de datos y escenas que facilitan el trabajo a la hora de tener grandes conjuntos de datos. Permite diagnosticar rápidamente las rutas de archivos rotas y corregirlas.

La generación y edición de materiales es bastante sencilla e intuitiva. Su interfaz de edición de nodos hace más fácil la conexión, organización y el trabajo con sombreados. Tiene integrado el renderizador *mental ray*, que presenta funciones fotorrealistas avanzadas. Es posible crear detalles naturales realistas mediante sus herramientas basadas en pinceles. Además, mediante *Toon Shader*, podemos transformar escenas 3D en dibujos animados.

Cuenta con soporte en foros, soporte web técnico y con licencias flexibles.

(Autodesk, 2015)

3.2. Comparación de tecnologías empleadas

En este apartado compararemos las diferentes opciones de tecnologías que podríamos utilizar para la realización del simulador, y finalmente se elegirán las empleadas.

3.2.1. Motores de juego

En la tabla 3.1. se muestra una comparación entre los diferentes motores de juego comentados en el apartado anterior.

	Unity 3D	Source Engine	Unreal Engine	CryEngine
Plataformas de desarrollo	Windows y OSX	Windows, OSX y Linux	Windows y OSX	Windows, OSX y Linux
Lenguajes de programación	C#, JavaScript y Boo	C++	C++, C# y Cg	Lua
Aplicaciones para	iOS, Android, Windows Phone 8, Blackberry, Windows, OSX, Linux, Web Player, Oculus	Windows, OSX, Linux, Xbox, Xbox 360 y PS3	Windows, OSX, iOS, Android, Xbox One, PS4	Xbox One, PS4, Wii U, PC, Android, iOS

	Rift, PS3, PS4, PlayStation Vita, PlayStation Mobile, Xbox One, Xbox 360, Wii U			
Sistema de scripting visual	Sí, con *Play Maker*	No	Sí, con *BluePrints*	Sí, con *Flow Graph*
Código fuente disponible	No	No	Sí	Sí (solo para desarrolladores comerciales)
Comunidad activa	Sí	Sí	Sí	Sí
Recursos y tutoriales	Abundantes	Poco	Abundantes	Abundantes
Popular (1-5)	5	3	5	4
Crear herramientas personalizadas	Sí	Sí	Sí	Sí
Posibilidad de uso gratuito	Sí (para programar en Windows) Versión Pro: 1500$ o 75$/mes	Sí	Sí (se debe aportar un 5% de las ganancias si el producto supera los 3.000$/trimestre)	No (subscripción de 9.90€/mes)
Facilidad de uso	Alta	Media	Media	Baja

Tabla 3.1. Comparativa de motores de juego

Teniendo en cuenta la información de la tabla (y la descrita anteriormente con las características), los motores más interesantes son Unity 3D y Unreal Engine:

Respecto a Unity 3D, entre sus ventajas podemos encontrar que es uno de los motores con mejores condiciones de licencia en la industria de los videojuegos, puede adquirirse en su versión gratuita, o la versión Pro por 1.500$ o 75$ mensuales. Es fácil de usar con una interfaz intuitiva. y es compatible con otras plataformas de juego. Tiene una gran comunidad la cual proporciona mucho apoyo a la hora de trabajar con Unity, además de que la curva de aprendizaje es inferior a la de otros motores más difíciles de manejar y es una opción muy utilizada por desarrolladores. Es compatible con una gran variedad de formatos de archivos, y permite trabajar tanto en 2D como en 3D Otra de sus ventajas es que cuenta con una biblioteca de Assets muy grande, donde podemos descargar o comprar archivos. Por otro lado tenemos algunas desventajas como pueden ser la limitación de sus herramientas, por lo que muy frecuentemente hay que crear algunas propias. Además el tiempo requerido para hacer un juego con efectos complejos es muy alto. No tiene características de modelado en tiempo real.

En cuanto a *Unreal Engine*, entre sus ventajas podemos encontrar que es utilizado por una gran variedad de desarrolladores, por lo que el apoyo de la comunidad es muy alto. Tiene muchos tutoriales tanto en formato de videos como de texto, y en sus actualizaciones va añadiendo nuevas herramientas útiles por tanto la cantidad actual que

posee es muy grande. Además es compatible con bastantes plataformas. Este ha sido abierto al público recientemente, teniendo que pagar únicamente en el caso de crear algo con mucho éxito. Sus capacidades gráficas son muy altas. Sus desventajas son pocas, una de ellas podría ser que algunas de las herramientas son difíciles de aprender a manejar.

Teniendo en cuenta todos estos aspectos nos decantamos por ***Unity 3D*** ya que tiene más facilidades a la hora de aprender a utilizarlo (ya que nunca antes había trabajado con un motor de juegos), y dispone de todas las herramientas necesarias para crear nuestro simulador.

3.2.2. Programas de modelado 3D

A continuación, en la tabla 3.2, se muestra una comparativa de los diferentes programas de modelado 3D comentados anteriormente. En ella se resaltan en color verde las características más positivas y en color rojo las más negativas (siendo el color negro neutro).

	3ds Max	Blender	Cinema 4D	Maya
Primer mercado	**Juegos**	**Tiempo real**	**Movimiento y diseño**	**Juegos y películas**
Precios para usuarios finales	**5.000€**	**Gratuito**	**800€**	**2.500€**
Subscripción anual	**Sí**	**No**	**No**	**No**
Plataformas de desarrollo	**Windows**	**Windows, Linux y MacOS**	**Windows y MacOS**	**Windows, Linux y MacOS**
Idiomas	**Francés e inglés**	**Muchos**	**Muchos**	**Inglés**
Uso en industrias	**Mucho**	**Poco**	**Bastante**	**Mucho**
Tiempo de aprendizaje	**Medio**	**Medio-alto**	**Rápido**	**Medio-alto**
Soporte para usuario	**Bueno**	**Comunidades**	**Muy bueno**	**Bueno**
Interfaz	**Limpia y potente**	**Rápida No intuitiva**	**Limpia e intuitiva**	**Flexible y potente No intuitiva**
Documentación	**Buena**	**Buena**	**Muy buena**	**Muy buena**
Formatos importar/exportar	**Muchos**	**Bastante**	**Bastante**	**Muchos**
Renderizado	**Muy bueno (Interno y Mental Ray)**	**Bueno (Interno)**	**Bueno (Interno)**	**Muy bueno (Interno y Mental Ray)**
Texturas	**Muy bueno**	**Bueno**	**Muy bueno**	**Muy bueno**
Herramientas de animación	**Muy bueno**	**Bueno**	**Bueno**	**Muy bueno**
Modelado	**Muy bueno**	**Bueno**	**Muy bueno**	**Muy bueno**
Modificadores	**Muy bueno**	**Bueno**	**Bueno**	**Muy bueno**

Herramientas para partículas, ropa, etc.	**Muy bueno**	**Bueno**	**Bueno**	**Muy bueno**
Utilizad para encontrar trabajo	**Muy bueno**	**Regular**	**Bueno**	**Bueno**

Tabla 3.2. Comparativa de programas de modelado 3D

3ds Max es uno de los programas de modelado 3D más utilizado y conocido. Tiene unas características muy buenas aunque su precio es bastante elevado. Además cuenta con una interfaz intuitiva que permite que su curva de aprendizaje no sea demasiado alta.

La ventaja de *Blender* respecto a sus oponentes es que es un programa completamente gratuito. Por el contrario, en algunos aspectos tiene menor calidad que otros programas como *3ds Max* y *Maya*. Además, su interfaz es muy poco intuitiva y requiere una curva de aprendizaje bastante más grande que en los otros casos.

Maya tiene unas características similares a 3ds Max, pero es un programa que está más orientado a su utilización para crear efectos visuales y películas. Su interfaz es poco intuitiva.

Respecto a *Cinema 4D*, es más rápido de aprender a utilizar que todos los demás y tiene unas características bastante buenas, aunque *3ds Max* le supera en algunas de ellas.

Para concluir, se ha decidido utilizar ***3ds Max*** para realizar este trabajo, ya que es un programa que ofrece muchas características positivas, unas herramientas potentes, interfaz intuitiva, documentación disponible para aprendizaje, etc. Además es uno de los programas más utilizados en la actualidad, y puede ser muy útil a la hora de buscar trabajo como desarrollador de videojuegos (en el caso de este trabajo se utiliza para el desarrollo de un simulador) u otras opciones de trabajo que cuenten con modelado 3D.

4. Desarrollo del simulador de conducción

Este simulador se ha creado para contribuir a la seguridad vial. Para ello se han realizado diferentes entornos de conducción en los que el usuario deberá manejar el coche de una forma correcta y sin cometer infracciones. Contamos con una aplicación en la que al ejecutarla aparecerá una solicitud de nombre y tras ello un menú para la selección del tipo de conducción (automática, con levas o manual mediante la palanca de cambios) y los diferentes escenarios. Una vez que se seleccione un escenario se entrará dentro de él para poder realizar la simulación.

4.1. Menú inicial

Como se ha mencionado anteriormente, al ejecutar la aplicación, lo primero que aparece es una solicitud para que se introduzca el nombre del usuario. Tras introducir el nombre y pulsar aceptar se muestran diferentes botones para seleccionar el modo de conducción y acceder al escenario deseado. Una vez que seleccionamos el escenario que deseamos y entramos en él se da la opción de iniciar o de volver al menú de selección. Además una vez que terminemos el recorrido de un escenario (en las escenas que tienen fin) se proporcionarán las opciones de volver al menú o cambiar de usuario, de modo que en el segundo caso se vuelve a solicitar el nombre del nuevo usuario y tras introducirlo se volvería al menú de selección. Además, una vez conduciendo, el simulador puede ser pausado, presentándose las opciones de continuar o de volver al menú inicial. Dentro de este menú podemos salir del simulador de dos modos: pulsando la tecla “Escape” o bien pulsando el botón “Salir”. El nombre de usuario es almacenado en los *PlayerPrefs* mediante una variable “name”, y una vez que queremos cambiar de usuario eliminamos esta variable, creando una nueva posteriormente con el nuevo usuario. *PlayerPrefs* es una clase que permite guardar información recuperable aun cuando deje de correr la aplicación. En nuestro caso son utilizados para poder guardar variables que podamos compartir entre distintas escenas. Cada vez que salimos de la aplicación eliminaremos la variable asociada al nombre del usuario. También se utilizan *PlayerPrefs* para guardar tanto el tipo de conducción como la escena que ha sido seleccionada. Esto lo hacemos en unas variables llamadas *marcha* y *scene*, respectivamente. Estos *PlayerPrefs* serán utilizados posteriormente para almacenar en un fichero *XML* los datos referentes a la conducción, además de para indicar qué escena debemos cargar y cuál debe ser el modo de conducción que debemos indicar al coche.

Los botones y mensajes se realizan mediante la función *OnGUI* utilizando etiquetas (*GUI.Label*) y botones (*GUI.Button*). También se ha utilizado un *GUI.Toolbar* para las opciones de conducción, ya que este permite seleccionar únicamente uno de los botones que estén asociados a ello, de modo que si seleccionamos otro, se deseleccionaría el anteriormente elegido. Además se ha proporcionado un sonido para los botones al ser pulsados, cargando en *audio.clip* el sonido, y reproduciéndolo mediante *audio.Play( )*. Utilizando *GUI.Style*, en el código se ha cambiado el aspecto del texto de los botones y etiquetas, poniendo en negrita y color negro las etiquetas, y en negrita y color blanco la letra de los botones. También se ha añadido la propiedad de

que al pasar el cursor por encima de los botones se cambie la letra a azul claro, y que cuando un botón del *toolbar* se encuentre seleccionado, su texto se muestre en amarillo, para diferenciarlo. Todo esto se ha realizado con las opciones *hover*, *normal*, *onNormal*, *onHover*, etc.

En la figura 4.1 podemos observar el diagrama de flujo del funcionamiento de este menú.

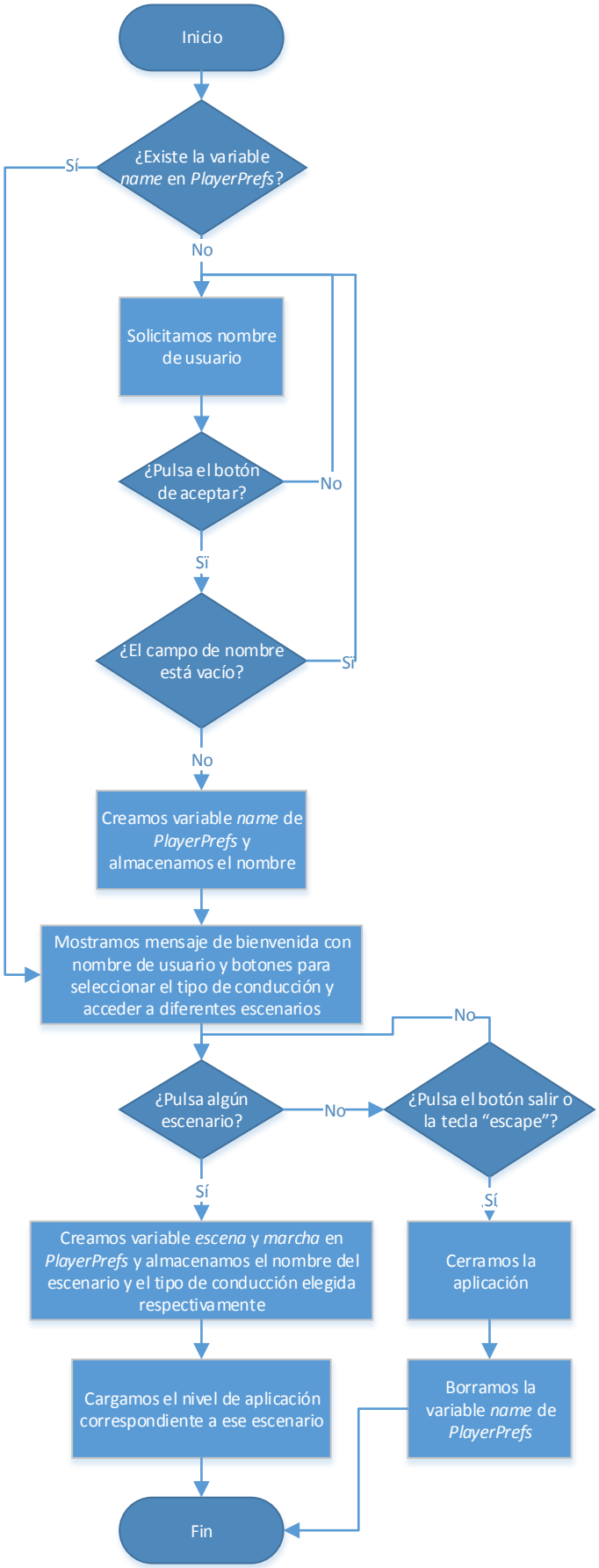

Figura 4.1. Diagrama de flujo de menú de inicio

4.2. Escenarios

Este simulador se compone de 6 escenarios diferentes. En este apartado se comentará cómo es cada uno de los escenarios y cómo han sido creados.

4.2.1. Escenario 1

El primero es un cruce simple de dos carreteras de un carril por sentido, en el que hay un paso de peatones, por el cual empezará a cruzar un peatón en el momento en el que el coche esté lo suficientemente cerca del paso de peatones. También vendrá un coche procedente del lado izquierdo del cruce, hacia el que debemos girar. En la figura 4.2 se muestra cómo sería la escena al iniciar.

Figura 4.2. Escenario 1 desde la perspectiva del conductor

Este escenario se ha realizado utilizando *Unity*. La carretera está hecha mediante una serie de cubos en los que se ha aplicado la textura correspondiente al asfalto (excepto en el tramo del paso de peatones y la intersección, que se han aplicado texturas diferentes), y la acera se ha hecho del mismo modo pero utilizando una textura correspondiente al pavimento de una acera. En la figura 4.3 se muestra la información correspondiente a uno de estos cubos, que se mostraría en el inspector. En este caso la textura se ha realizado directamente arrastrando la deseada hacia el objeto de la escena en el que la queremos aplicar. Además incluye algunos modelos 3D que han sido descargados, como algún banco, árbol, edificio, fuente y una señal de stop. Todos los elementos cuentan con sus correspondientes *Collider* para no ser atravesados (para ello deben tener la opción *isTrigger = false*) y para detectar si algún elemento choca con ellos *(*para ello deben tener la opción *isTrigger = true*), además de un *Rigidbody* para proporcionarles la fuerza de gravedad, peso, etc. Los componentes *Collider* definen la forma de un objeto para propósitos de colisiones físicas y son invisibles. Por otro lado los *Rigidbody* permiten a los objetos actuar bajo el control de la física. En la figura 4.4

podemos observar los componentes *Collider* y *Rigidbody* que tendría asociados cada señal de tráfico. De este modo pueden recibir diferentes tipos de fuerzas y actuar en consecuencia. También se ha añadido el cielo, mediante un *Skybox Material*, un elemento que haga de sol mediante una luz direccional y un terreno de hierba sobre el que se encuentra la escena.

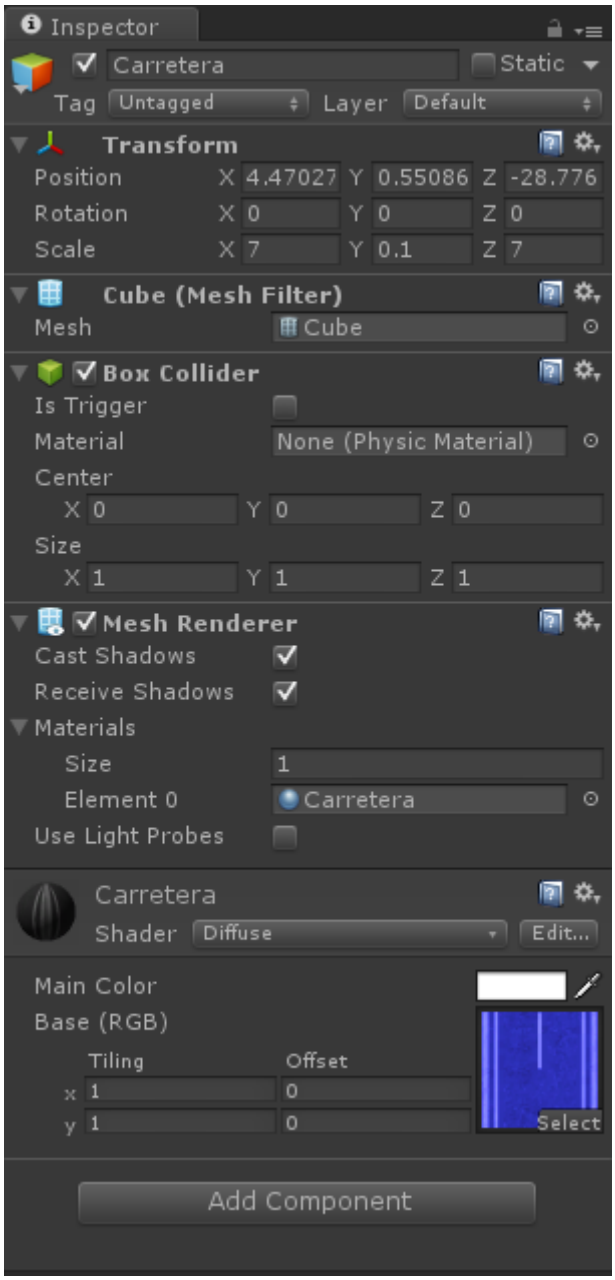

Figura 4.3. Inspector de un cubo correspondiente a un tramo de carretera

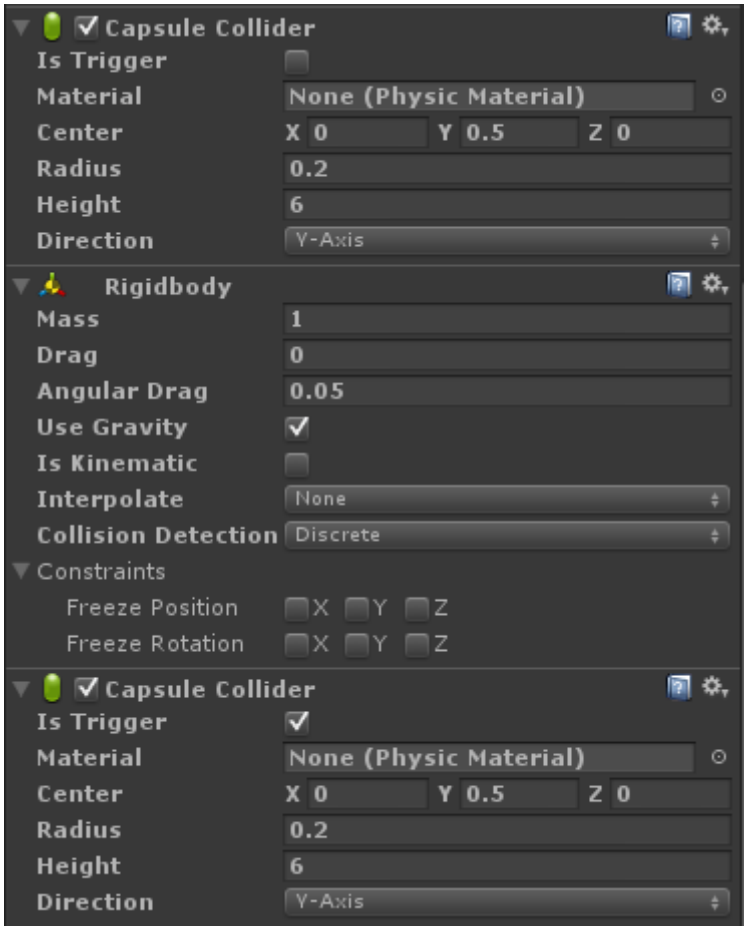

Figura 4.4. Componentes *Collider* y *Rigidbody* de una señal

En este escenario el conductor debe seguir la carretera hasta la intersección, y posteriormente girar a la izquierda (le será indicado mediante una imagen con una flecha en esa dirección). Para realizar la conducción correctamente el conductor no deberá subirse a la acera, deberá dejar pasar al peatón por el paso de cebra, sin atropellarlo, y deberá parar en la intersección ya que esta tiene una señal de stop. Debe tenerse en cuenta que todas estas acciones serán registradas y almacenadas en un fichero XML.

4.2.2. Escenario 2

El segundo escenario se compondrá de un circuito cerrado con 4 rotondas de 3 carriles cada una, interconectadas mediante carreteras rectas de dos carriles por sentido, de las cuales podemos observar una vista superior en la figura 4.5. En este caso habrá una variedad de coches que circularán por el escenario de una forma cíclica. Los coches del escenario circularán a diferentes velocidades, por lo tanto en los tramos en que un coche alcance a otro que va más despacio, lo adelantará (más adelante veremos cómo funcionan estos coches).

Para la realización de las rotondas y carreteras por las que están conectadas se ha utilizado el programa de modelado 3ds Max. Para ello se han creado diferentes materiales que se correspondan a las diferentes texturas que necesitamos.

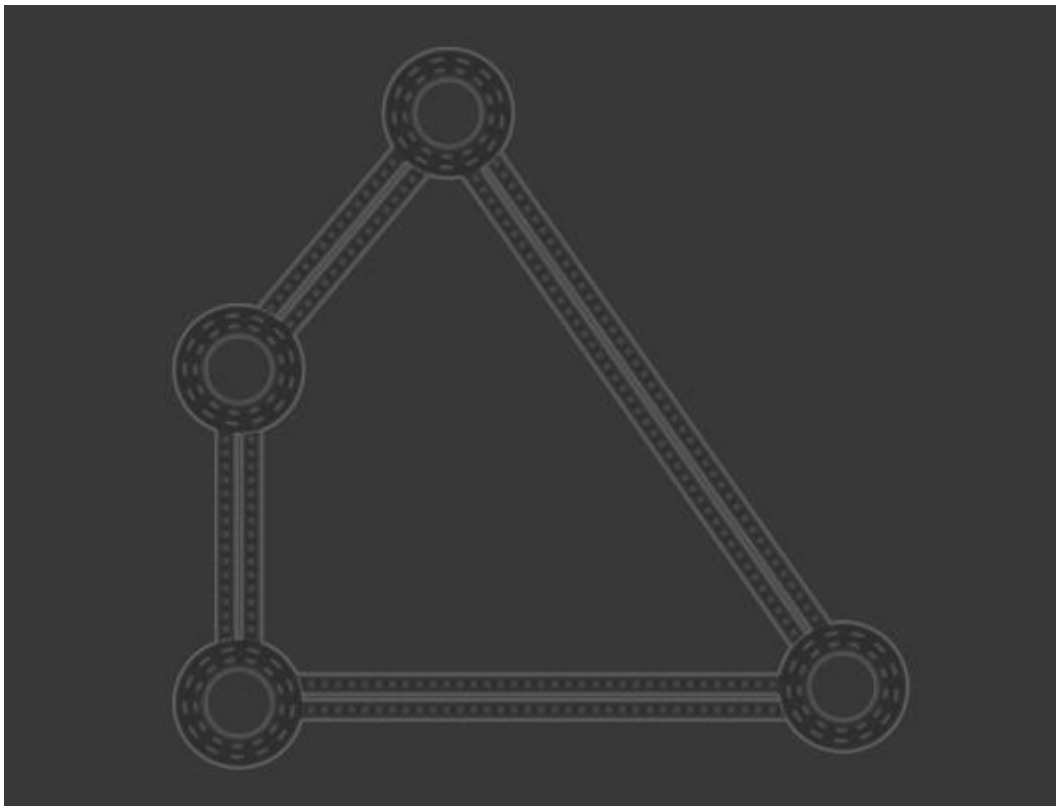

Figura 4.5. Vista superior de la carretera correspondiente al escenario 2

En el caso de la carretera, se han creado objetos de tipo *Box* de diferentes longitudes (pero misma altura y anchura para todas las carreteras). Se ha utilizado un material de tipo *Standard* (ya que utilizaremos una única textura). En *Maps* se ha elegido *Diffuse Color* → *Bitmap* y a continuación se ha seleccionado la textura correspondiente a una carretera de dos carriles con línea discontinua intermedia. A continuación se ha arrastrado el material creado hacia sus correspondientes objetos. Para lograr que el material se adapte al tamaño de la carretera creada se ha aplicado un modificador llamado *UVW Map* (para lo cual primero convertimos nuestro objeto en un polígono editable), de manera que con el elemento *Gizmo* que contiene, podemos modificar el tamaño y forma de la textura de manera que quede del modo deseado en el objeto.

En el caso de las rotondas, estas han sido creadas mediante un objeto de tipo *Tube* con un número elevado de lados para obtener un aspecto redondeado. En este caso se ha utilizado un material de tipo *Multi/Sub-Object*, ya que necesitamos diferentes texturas para diferentes tramos de la rotonda (la línea exterior de la rotonda debe ser continua en los tramos que no hay salidas ni entradas, discontinua en los tramos de entrada y sin línea en los tramos de salida). En este caso elegimos 3 sub-materiales de tipo *Standard*, que procedemos a crear del mismo modo que hicimos con el de la carretera, asignando a cada uno la textura correspondiente (aunque todas con 3 carriles). Cada uno de estos sub-materiales tiene asignado un ID diferente, para poder elegir

dónde se utiliza uno u otro. Este material creado se arrastra hacia nuestros objetos correspondientes a las rotondas, que posteriormente convertimos en polígonos editables. En este caso, la textura no debe aplicarse con una proyección normal, sino que debe seguir la forma correspondiente a la rotonda. Para ello debemos utilizar un modificador diferente, en este caso se utiliza *UVW Xform*, y mediante *U Tile*, *V Tile* y *W Tile* (y en el caso de que sea necesario se modifican los offset) adaptamos la textura al tamaño de nuestro objeto. Al tener un polígono editable, ahora podemos elegir seleccionar por polígonos, y en cada uno de ellos podemos elegir el ID que queremos que utilice del material, de forma que la textura utilizada en ese polígono será la correspondiente a ese ID. En la figura 4.6 vemos un zoom hacia una de las rotondas, donde podemos observar cómo en diferentes lugares son necesarias diferentes texturas.

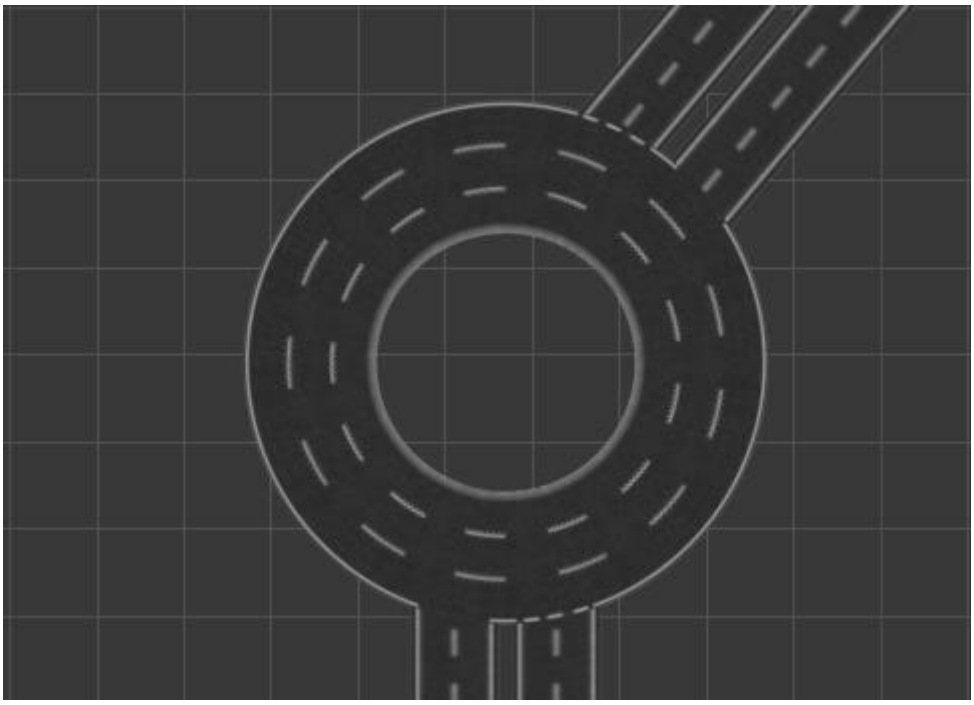

Figura 4.6. Rotonda con diferentes texturas en función de entradas/salidas

Figura 4.7. Escenario 2

Tras conectar las carreteras y rotondas del modo deseado, se ha importado el objeto a Unity. Ahí se le ha añadido un terreno con textura correspondiente a hierba y se han creado algunos montículos con textura más rocosa, un cielo y el sol (estos últimos del mismo modo que en el escenario anterior). Además se han situado las correspondientes señales de ceda el paso en todas las rotondas y algunas esculturas de adorno (con sus correspondientes *Collider* y *Rigidbody*). También se han situado los coches que durante la simulación estarán en funcionamiento haciendo el recorrido correspondiente. La figura 4.7 muestra una vista desde un punto externo de cómo sería el escenario.

En este escenario se realizará una vuelta completa a él, tomando la siguiente salida de las rotondas, ya que en este caso solo tienen por la que se entra y otra, que es por la que se debe continuar. Aquí se deberá realizar una vuelta completa hasta regresar al punto de inicio.

4.2.3. Escenario 3

El tercer escenario está compuesto por una serie de carreteras más largas, de longitudes entre 0.5 y 2.5 km de longitud, interconectadas mediante rotondas de 2 y 3 carriles. Las carreteras son de dos tipos también: un carril por sentido y dos carriles por sentidos. Estas carreteras y rotondas están realizadas en 3ds Max de un modo similar al anterior escenario, con la diferencia de que ahora las carreteras tienen curvas. Estas curvas se han realizado poco a poco modificando la posición de los vértices de los polígonos editables y algunas mediante partes de objetos *Tube*. La figura 4.8 muestra una parte del escenario en la que las carreteras presentan curvas, mientras que en la figura 4.9 podemos ver desde una perspectiva superior y muy alejada (para que quepan todas las carreteras) cómo sería el conjunto del escenario.

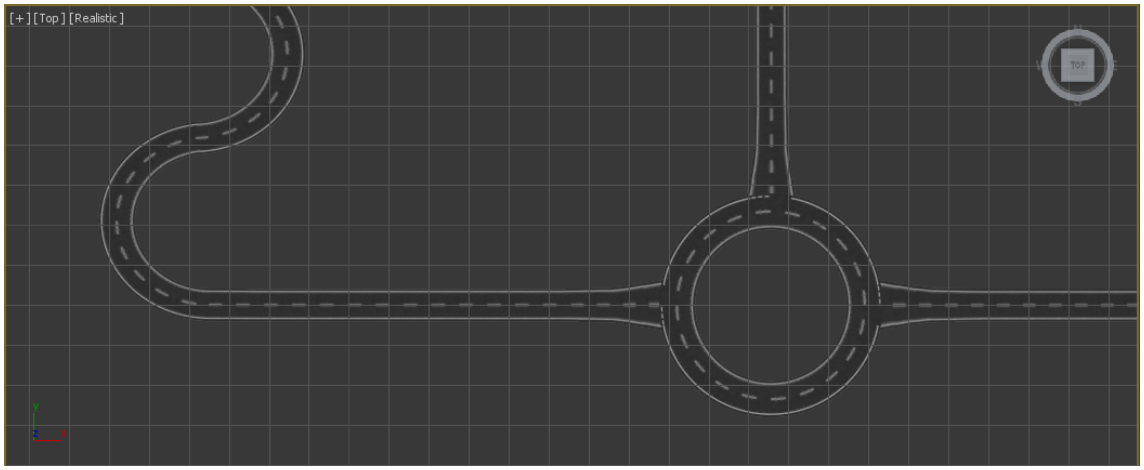

Figura 4.8. Zoom de una parte del escenario 3

Al igual que en los casos anteriores, este escenario ha sido importado a Unity, donde se le han añadido los demás elementos como señales, terreno, cielo, sol, etc. En este caso se incluye más variedad de señales (de las cuales varias han sido creadas

mediante 3ds Max), como son señales de límite de velocidad, de aviso de rotondas o de curvas peligrosas.

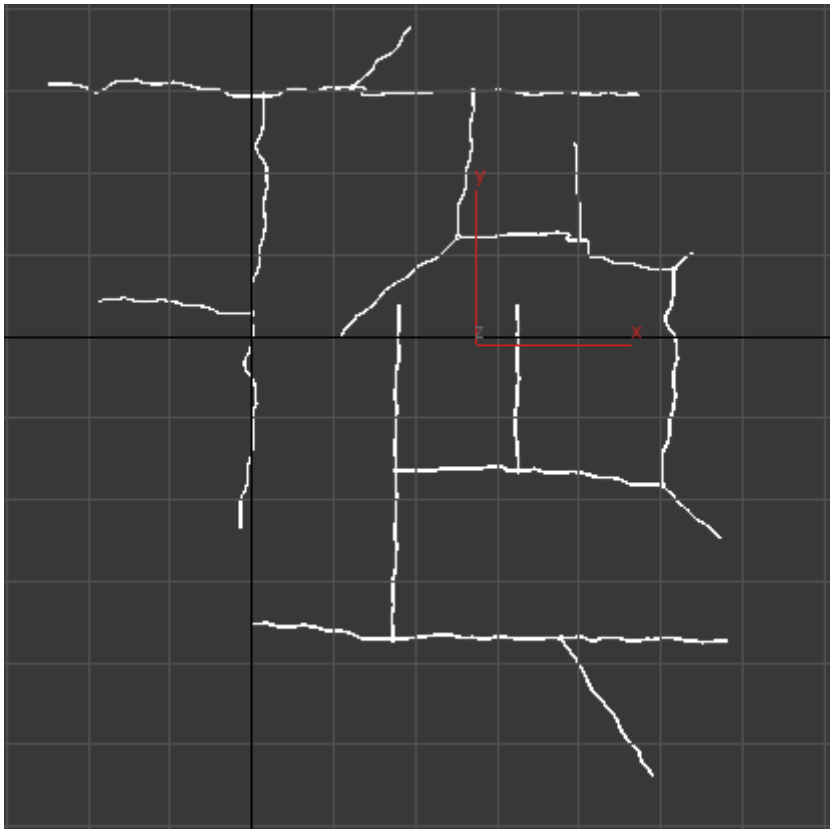

Figura 4.9. Vista superior de las carreteras del escenario 3

En este escenario se va guiando al conductor mediante imágenes, indicando la salida que debe tomar en cada rotonda hasta llegar al final del recorrido. Esta imagen se mostrará unos metros antes de llegar a la rotonda, y desaparecerá una vez que tomemos la salida correcta. Iremos encontrando otros coches en las rotondas del recorrido, donde se deberán respetar las prioridades y circular de un modo correcto para no colisionar con ellos. De la primera rotonda saldrá un coche por la misma salida que nosotros, que irá disminuyendo la velocidad en función de la nuestra, y que por tanto deberemos adelantar. En rotondas posteriores en algunas ocasiones se incrementa el número de coches con los que nos encontraremos dentro de la rotonda. Este recorrido sería el más largo de todos tardándose en realizar aproximadamente unos 20 minutos. Simula un recorrido interurbano.

4.2.4. Escenario 4

Este escenario se ha desarrollado de un modo muy diferente al resto. Para ello se ha utilizado un plugin para Unity llamado *Road & Traffic System* el cual permite utilizar diferentes tramos de carretera prediseñados e irlos conectando entre sí mediante nodos. El fin de este plugin es permitir a los usuarios crear y simular redes de tráfico dinámicas para cualquier clase de vehículo de una manera más rápida. Es posible personalizar las piezas utilizadas, y cuenta con rotondas, intersecciones con semáforos, etc. Además tiene sus propios tutoriales en formato vídeo para aprender a manejarlo correctamente, y

un foro donde se pueden plantear las cuestiones que nos surjan. La figura 4.10 muestra parte del inspector del objeto que se utiliza para colocar las piezas de carretera.

Figura 4.10. Inspector de Traffic System con tramos a utilizar

Este escenario incluye carreteras de 1, 2 y 3 carriles por sentido, además de intersecciones en 'T' e intersecciones de 4 carriles señalizadas mediante semáforos. También incluye rotondas. Todo esto para los diferentes tamaños de carril.

Para realizarlo utilizamos las herramientas *Ankle* y *Edit* y las herramientas correspondientes para colocar las piezas en el lugar deseado y realizar conexiones en el sentido que se necesite (conexiones que serán utilizadas posteriormente para guiar a los coches con inteligencia artificial). En la figura 4.11 se muestra el editor de conexiones de este plugin.

A esta escena como al resto, se le han añadido el terreno, cielo y sol correspondientes. Además se ha añadido algunos edificios realizados mediante *3ds Max* con cubos, aplicando diferentes texturas a las paredes y al tejado, y ajustándolas mediante el modificador *UVW Map*.

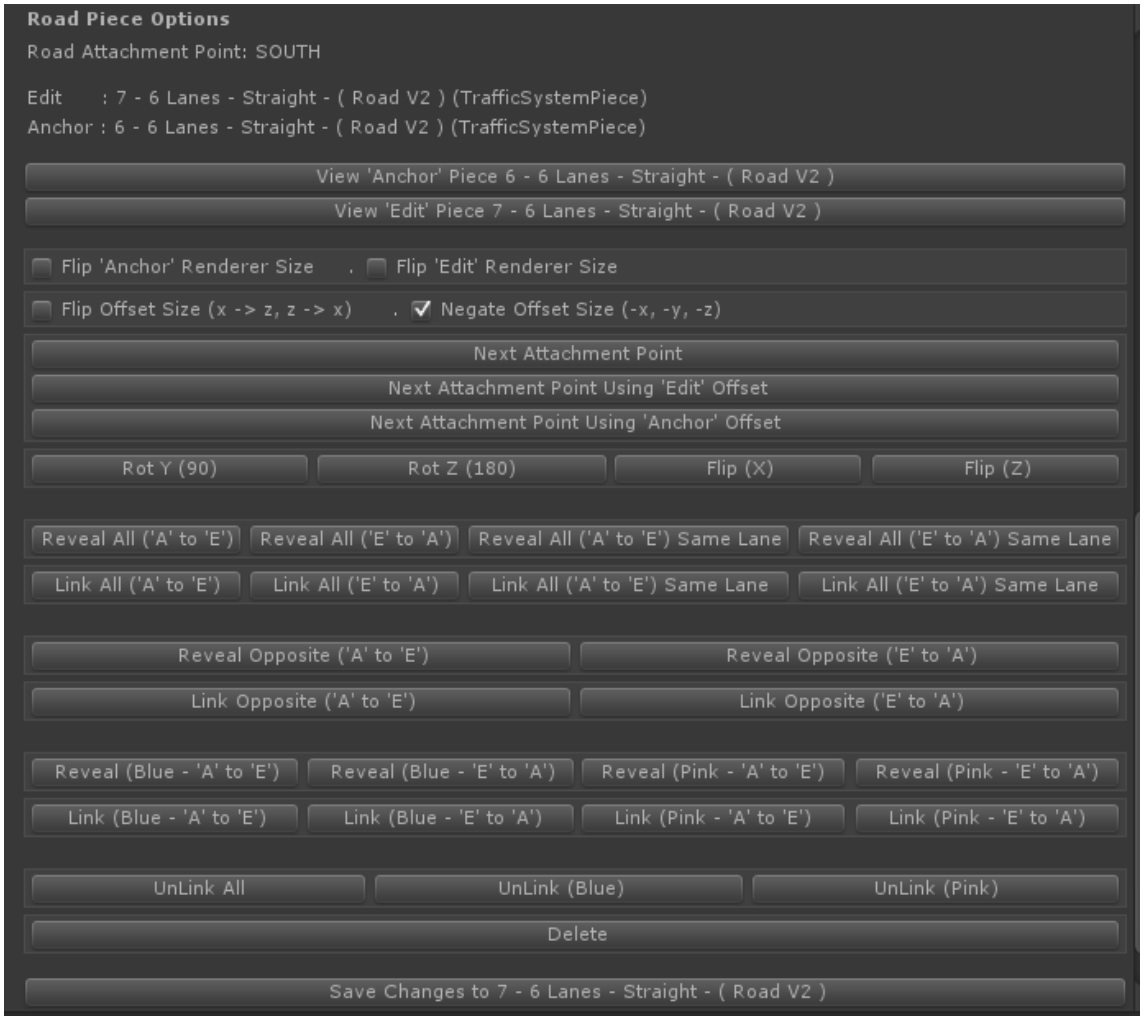

Figura 4.11. Editor de conexiones de Traffic System Road

También se han creado imágenes para que vayan apareciendo durante la simulación de este escenario, al acercarse a una rotonda o intersección, para indicar al usuario qué dirección debe tomar. Estas aparecen en la esquina superior derecha. Del mismo modo que en el escenario anterior, estas desaparecerán una vez que hayamos tomado la salida correcta. La figura 4.12 muestra un ejemplo de un usuario conduciendo en una escena, en la que podemos observar en la esquina superior derecha la indicación de la dirección que debe tomar.

Figura 4.12. Escena 4 con indicación de dirección a tomar (en la esquina superior derecha)

La simulación finalizará cuando el usuario llegue a un punto en concreto del recorrido que se ha tomado como punto final.

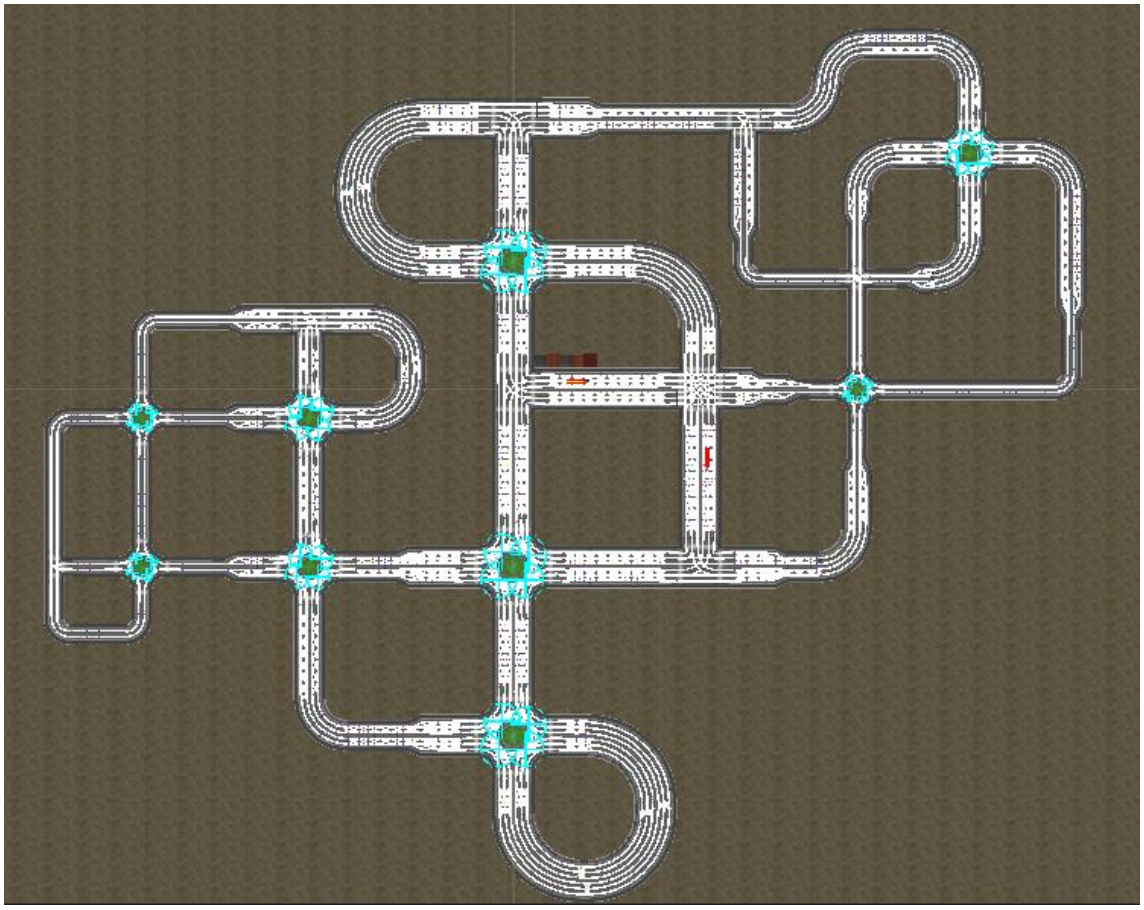

Figura 4.13. Vista superior del escenario 4

En la figura 4.13 podemos observar el escenario desde una vista superior, donde se pueden ver las carreteras de diferente número de carriles (según anchura), las rotondas (son las que tienen un círculo verde en medio) y otras intersecciones en T y de

4 carriles. La figura 4.14 muestra una intersección de 4 carriles con semáforos, en la que además se ven los nodos y las líneas que los conectan. Este escenario simularía la conducción en una zona urbana.

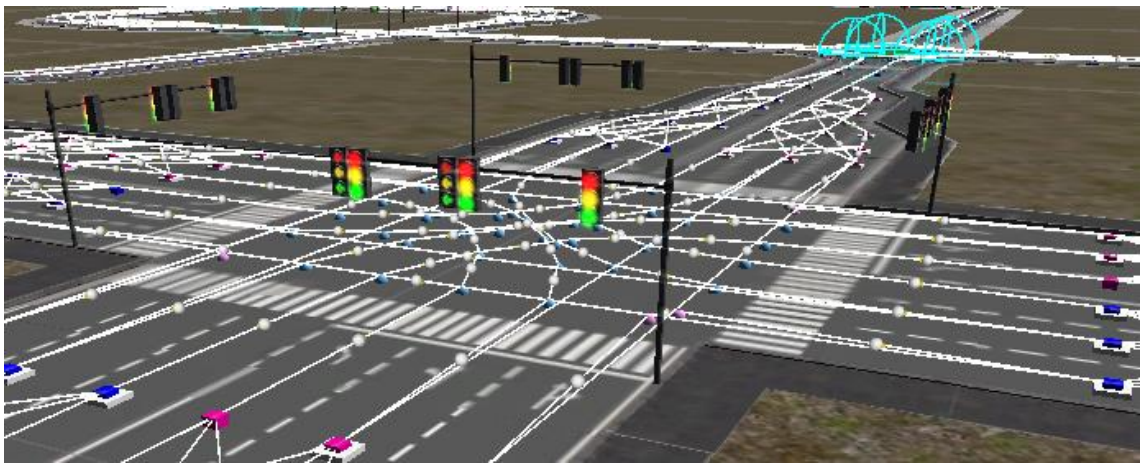

Figura 4.14. Intersección con semáforos (los nodos y las líneas que los conectan no son visibles durante la simulación)

Para la utilización de este *plugin* ha sido necesaria una modificación de los códigos proporcionados ya que estos generaban algunos errores al depurar en *Monodevelop* (entorno de desarrollo utilizado por *Unity*). Esto se comentará en el Anexo I.

4.2.5. Escenario 5

Como escenario 5 se ha utilizado el mismo entorno que en el escenario 3, con la diferencia de que en este caso, una vez que el usuario llega al "punto final" de este escenario, se le lleva de nuevo al punto de inicio sin que esto sea demasiado perceptible a ojos del usuario (continúa una carretera igual), de modo que la conducción se mantiene fluida. En él se pretende que el usuario pueda conducir de manera ininterrumpida todo el tiempo que desee. La simulación únicamente finalizará cuando el usuario pulse "Escape", que se ha utilizado como tecla para pausar la simulación, donde podrá elegir continuar o salir al menú.

4.2.6. Escenario 6

Para el escenario 6 se ha utilizado el entorno utilizado en el escenario 4. En este caso se guía al conductor por las diferentes carreteras, de modo que en cierto momento el conductor vuelve a pasar por el punto de inicio. En lugar de mostrar un mensaje para salir, no aparecerá nada, de manera que el conductor seguirá siendo guiado una y otra vez dando tantas vueltas por el escenario como desee. Al igual que en el escenario 5, el conductor podrá elegir cuándo salir de esta escena pulsando la tecla "Escape" y seleccionando la opción "Salir al menú".

4.3. Elementos

En este apartado se comentarán los distintos elementos que forman parte del simulador de conducción, su modo de funcionamiento y cómo han sido realizados.

4.3.1. Coche a conducir

Respecto al coche que se conducirá en este simulador, trataremos acerca de los componentes que se le han tenido que añadir para que este tenga un funcionamiento correcto. Además se verá un poco cómo funcionaría teóricamente la fuerza que se le ejerce al motor y cuál sería el consumo del automóvil en función de su velocidad para las diferentes marchas en las que nos podamos encontrar. Por último se verá cómo se ha implementado el coche para su funcionamiento y conducción.

4.3.1.1. Componentes físicos del coche

Para hacer el coche que se va a conducir, se han tenido que añadir una serie de componentes al modelo 3D de un coche. Uno de ellos es el *Rigidbody* (mostrado en la figura 4.15), necesario para obtener un comportamiento lo más real posible, en el que definimos la masa del coche, la resistencia que presenta al viento e indicamos que utilice la fuerza de la gravedad.

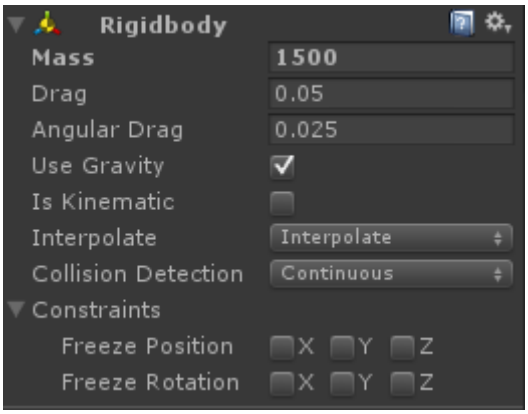

Figura 4.15. Componente Rigidbody

También se han añadido dos *Box Collider* (uno para la parte superior del coche y otro para la parte inferior) con la opción *Is Trigger* desactivada, de modo que el coche no sea atravesado por otros objetos; y otros dos con ella activa, para detectar que está siendo atravesado por algo. Estos elementos indicados en el inspector, se observan en la figura 4.16.

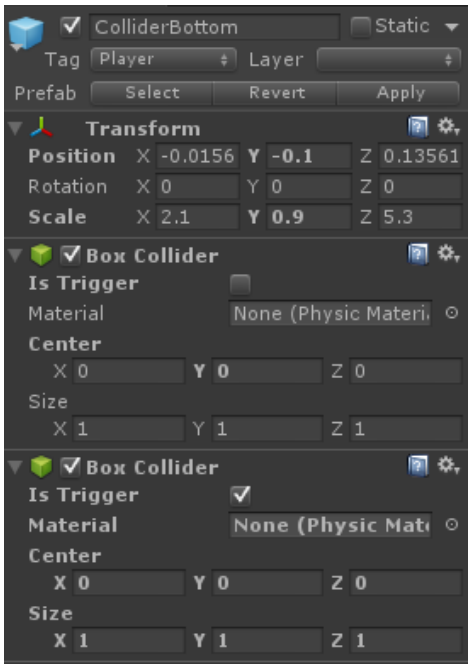

Figura 4.16. Objeto con dos *Box Collider* para la parte inferior del automóvil

A cada rueda del coche se le ha añadido un *Wheel Collider* (cuya correspondencia en el inspector se muestra en la figura 4.17), en el cual definimos el peso de la rueda, el radio y la suspensión, entre otros valores como valores de fricción.

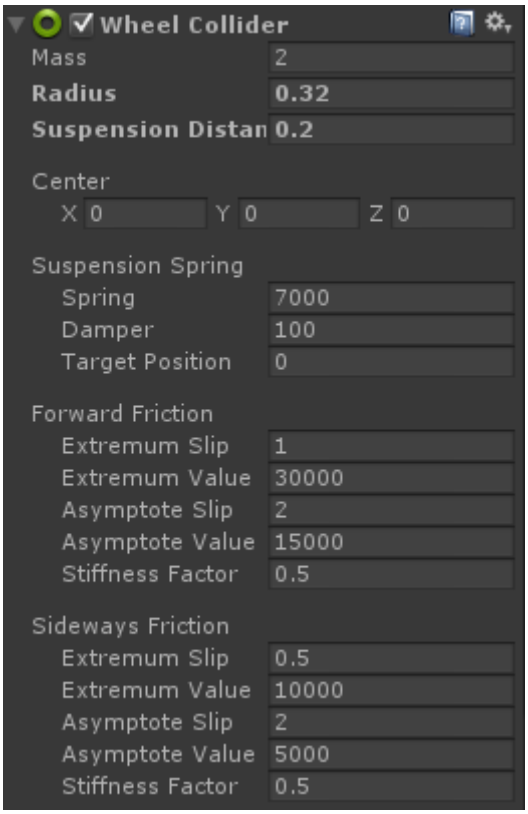

Figura 4.17. Componente Wheel Collider

Además el coche cuenta con una serie de cámaras: una para la visión normal durante la conducción, otra que apunta hacia atrás para utilizarla como retrovisor y otras dos, una mirando hacia la izquierda y otra hacia la derecha, de manera que el conductor pueda activarlas cuando desee mirar a los lados. La figura 4.18 muestra un ejemplo de cada una de estas vistas.

Figura 4.18. Cámara mirando hacia la izquierda, hacia el frente y hacia la derecha, sucesivamente

Por último se ha añadido una fuente de sonido, con el sonido de un motor, que será configurada para que suene de manera cíclica y en función de lo revolucionado que se encuentre el coche. Su correspondencia en el inspector se observa en la figura 4.19.

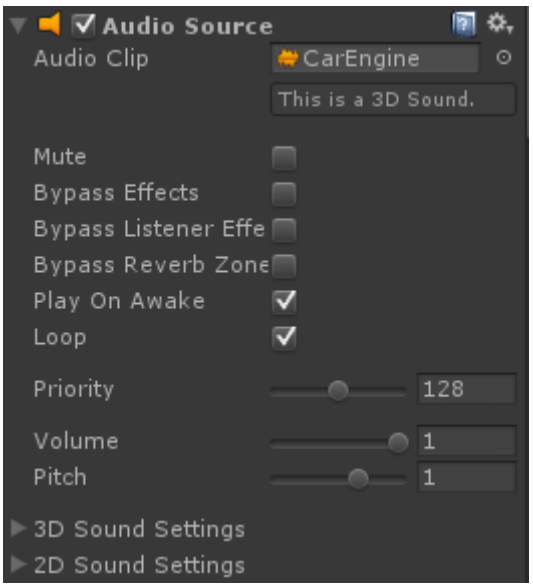

Figura 4.19. Componente de sonido

4.3.1.2. Funcionamiento teórico del coche

Aquí se tratarán algunos aspectos sobre el funcionamiento teórico del motor de un coche, que serán utilizados en el coche del simulador.

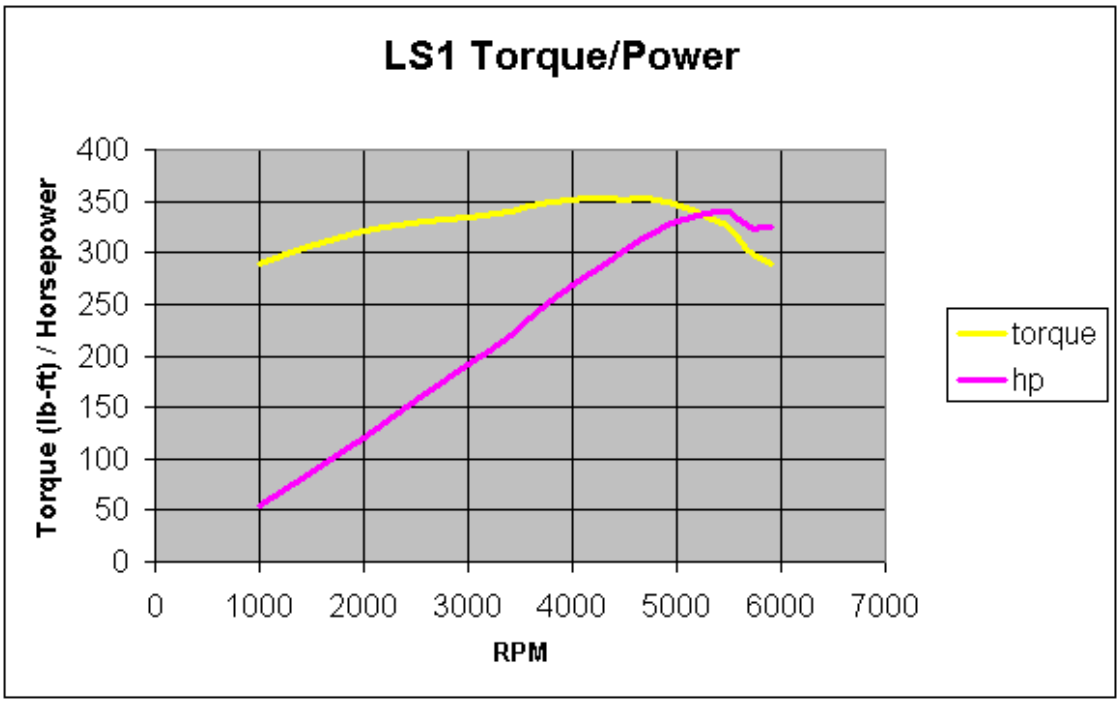

Figura 4.20. Torque en función de las RPM

El torque del motor del coche o también llamado momento, es la tendencia de una fuerza a rotar un objeto alrededor de un eje. Este depende de las revoluciones del

motor, concretamente del modo que se representa en la imagen. Esta dependencia se muestra en el gráfico de la figura 4.20. De este modo en función de lo revolucionado que esté el coche tendremos una cantidad de potencia diferente. Esto será reflejado en el desarrollo del coche del simulador con una curva (*AnimationCurve*).

Para calcular la fuerza total que se le aplica al motor debemos tener en cuenta otros factores: ratio diferencial (número de revoluciones de salida de transmisión por cada vuelta de la rueda), ratio de la marcha (relación entre el número de dientes del engranaje de entrada y el de salida, por tanto diferente para cada marcha siendo un valor más elevado para marchas bajas y un valor menor en marchas más altas), revoluciones por minuto (RPM) de la rueda, la curva del torque mostrada anteriormente y el par motor del automóvil.

- Ratio de transmisión = Ratio de marcha [marcha] * Ratio diferencial
- RPM del motor = RPM de la rueda * Ratio de transmisión
- Potencia total del motor = Curva torque (RPM del motor) * Par motor
- Fuerza total del motor = Potencia total del motor * Ratio de transmisión

Un ejemplo de valores que podrían tomar para un coche real serían:

- Ratio diferencial = 3.42
- Ratio de la marcha [1ª] = 2.66
- Ratio de la marcha [2ª] = 1.78
- Ratio de la marcha [3ª] = 1.30
- Ratio de la marcha [4ª] = 1
- Ratio de la marcha [5ª] = 0.74
- Ratio de la marcha [Marcha atrás] = 2.90

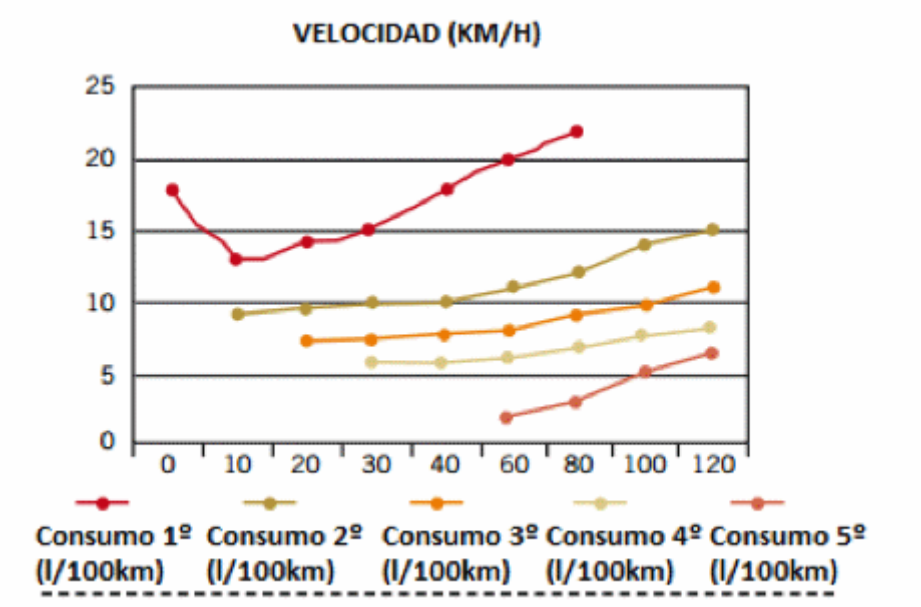

Figura 4.21. Consumo del automóvil en función de su marcha y velocidad (Autoexpress, 2015)

En cuanto al consumo del automóvil, este depende de cuál sea el coche que conduzcamos, por tanto, para este simulador vamos a tomar un ejemplo concreto de consumos en función de la marcha y la velocidad, pero que podrían ser diferentes para otro coche. El ejemplo tomado es el mostrado en la figura 4.21.

4.3.1.3. Implementación del automóvil

Se ha diseñado un automóvil con 3 opciones de modo de conducción: cambio de marchas automático, cambio de marchas mediante levas y cambio de marchas manual, utilizando la palanca de cambios y el embrague. Todo esto se ha diseñado para que pueda ser conducido tanto mediante el teclado del ordenador como con el periférico Logitech G27 (volante, levas, pedales y palanca de cambios).

Para nuestro automóvil hemos utilizado las fórmulas mostradas en el apartado anterior para calcular el torque que aplicamos al motor finalmente. Se han probado diferentes valores para ver cuáles se adaptarían mejor al funcionamiento correcto de un coche. Finalmente se han tomado diferentes valores en algunos de los escenarios para tener un comportamiento algo distinto (ya que cualquier coche no funcionaría exactamente igual que otro en ese aspecto), dentro de valores razonables.

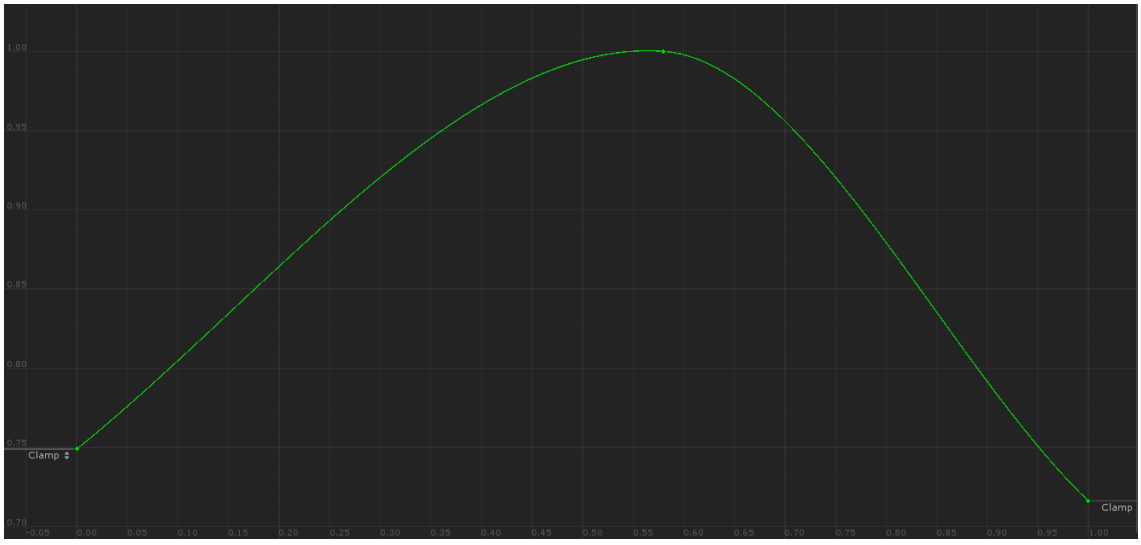

Figura 4.22. *AnimationCurve* utilizada para el motor (eje Y va de 0.7 a 1 y eje X de 0 a 1)

Para la curva mencionada con anterioridad, se ha utilizado un elemento llamado *AnimationCurve*, que como podemos observar en la figura 4.22, toma una forma similar a la vista teóricamente. Como se menciona en el título de la imagen, el eje X de esta gráfica va de 0 a 1, y esto representa las distintas revoluciones a las que puede ir el coche, siendo el 0 equivalente a 0 rpm y el 1 equivalente a 6000 rpm que es el máximo que hemos admitido. Respecto al eje Y representa el valor del torque, que siendo 1 se toma el máximo, en 0.7 se tomaría el 70% y así sucesivamente para cualquier punto.

En el coche que conduciremos, existirá un registro del consumo de combustible. Por tanto, al implementarlo se han creado también diferentes gráficas de consumo en función de la marcha en la que nos encontremos. Esto también se ha realizado con los

elementos llamados *AnimationCurve*. Estas curvas se han realizado tomando las formas correspondientes a las curvas de consumo teóricas mostradas en la gráfica del apartado anterior.

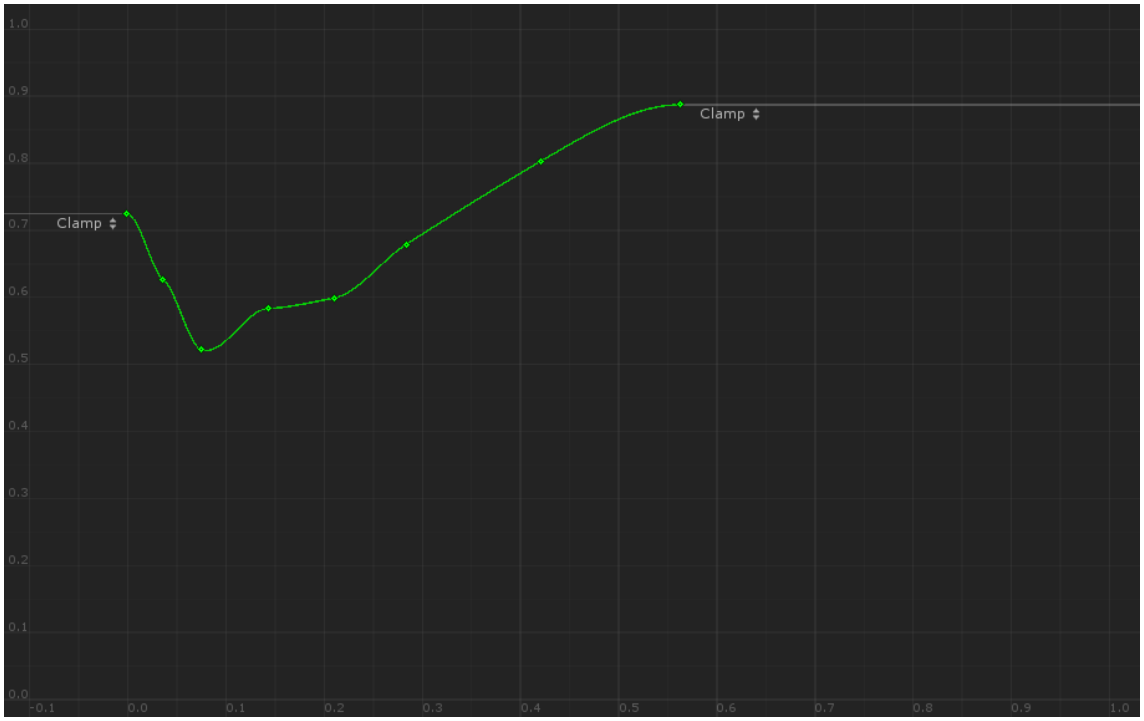

Figura 4.23. AnimationCurve para consumo en marcha 1

Las curvas son evaluadas en función de la velocidad a la que vaya el automóvil. El eje X representaría la velocidad del coche, representando de 0 a 1 el rango de velocidades que puede tomar nuestro automóvil, en este caso de 0 a 140 km/h. El eje Y representará el consumo del automóvil, donde el 0 equivaldría a 0 litros/100km y el 1 a 25 litros/100km (los puntos intermedios representarían el porcentaje correspondiente). La figura 4.23 muestra la *AnimationCurve* utilizado para el consumo en la marcha primera. En la figura 4.24 se observan las correspondientes a las marchas segunda, tercera, cuarta y quinta (aunque en este caso se ha hecho un zoom en la zona de cada curva, por lo que las dimensiones realmente no son las mismas)

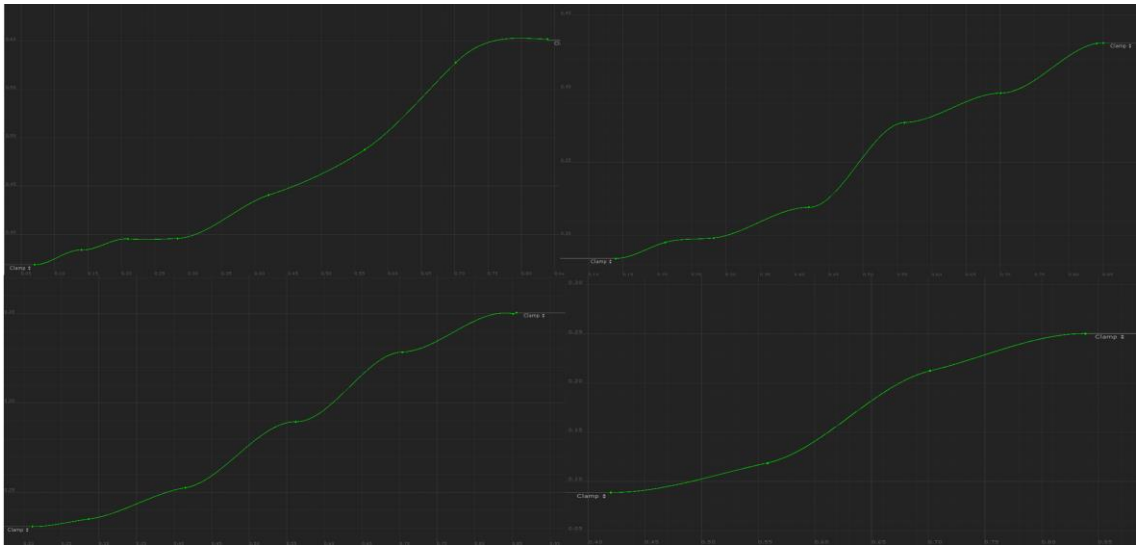

Figura 4.24. Curvas de consumo correspondientes a las marchas 2ª, 3ª, 4ª y 5ª (tener en cuenta que están a diferentes escalas ya que es un zoom en la zona donde se encuentra la curva)

Aunque en la imagen no veamos en qué rango de valores está cada curva, se debe tener en cuenta que se ha seguido el modelo de la gráfica teórica del apartado anterior, de modo que las marchas más altas tienen un consumo menor y las más bajas un consumo superior.

A continuación trataremos sobre el funcionamiento del coche en cuanto a scripts. Tiene asociado uno al coche en general y otro a cada una de las ruedas (el de las ruedas es diferente al del coche). La figura 4.25 muestra el diagrama de flujo del coche a conducir, mientras que la figura 4.26 muestra el diagrama de flujo de las propias ruedas.

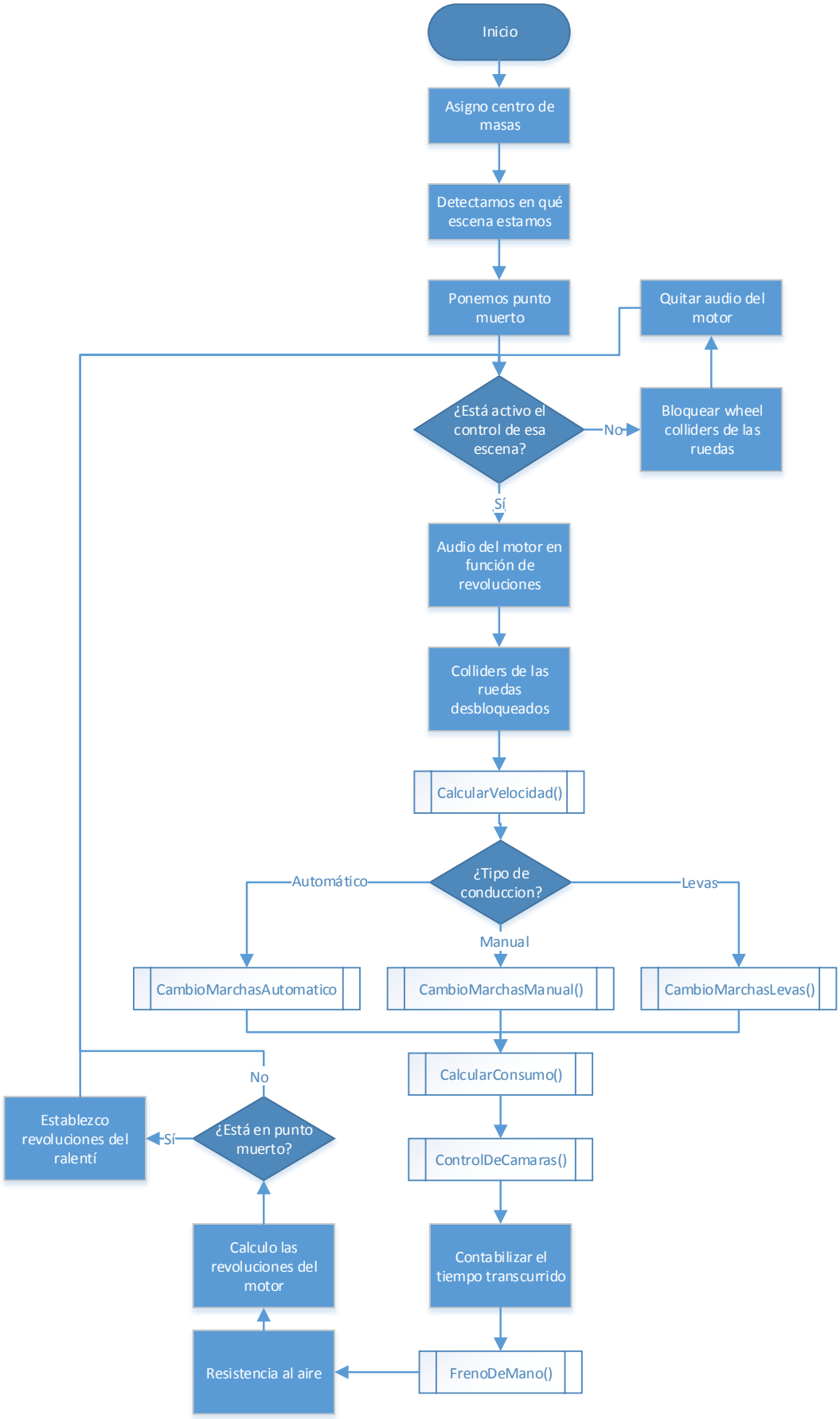

Figura 4.25. Diagrama de flujo del coche a conducir

La función *CalcularVelocidad( )* se encarga de calcular la velocidad actual a la que se está moviendo el automóvil, además de la distancia total que ha recorrido hasta el momento y la velocidad media que ha llevado durante el recorrido.

La función *CalcularConsumo( )* calcula el consumo actual del automóvil, además del consumo medio que se ha tenido durante la simulación y el consumo total acumulado en el recorrido. Para ello evalúa las curvas de consumo en función de la marcha en la que se encuentre y la velocidad que lleve en cada momento.

La función *ControlDeCamaras( )* se encarga de cambiar la profundidad de las cámaras que miran hacia la derecha y hacia la izquierda, para que cuando estas se activen se muestren por encima de la cámara principal, y cuando no, vuelva a mostrarse la principal.

La función *FrenoDeMano( )* indica si se está pulsando el control correspondiente al freno de mano o no. Del funcionamiento del freno de mano se encarga luego el script asociado a las ruedas del automóvil.

Las funciones *CambioMarchasAutomatico( ), CambioMarchasManual( ), CambioMarchasLevas()* se encargan del cambio de las marchas, que se realiza de un modo diferente dependiendo del modo de conducción en el que estemos. En el caso de automático las marchas están configuradas para cambiarse automáticamente en función de las revoluciones a las que estemos: si el coche supera las 2.000 rpm aumentamos a la siguiente marcha y si el coche baja de las 1.200 rpm reducimos la marcha. El cambio a punto muerto y marcha atrás se deberá realizar manualmente mediante las levas. Para el caso de las levas está configurado para que las marchas las cambie el conductor subiendo y bajando con las levas. Por último en el caso manual las marchas serán cambiadas pisando el embrague y metiendo la marcha que se desee con la palanca de cambios (o en el caso de utilizar el teclado con las teclas correspondientes a ello).

Además mediante la función *OnGUI( )* mostramos por pantalla gráficamente la marcha en la que se encuentra el automóvil, la velocidad a la que va y las revoluciones que lleva en cada momento. La velocidad y las revoluciones se muestran mediante un círculo simular al de los automóviles, con un pivote que gira sobre el punto medio indicando en cada momento la actual.

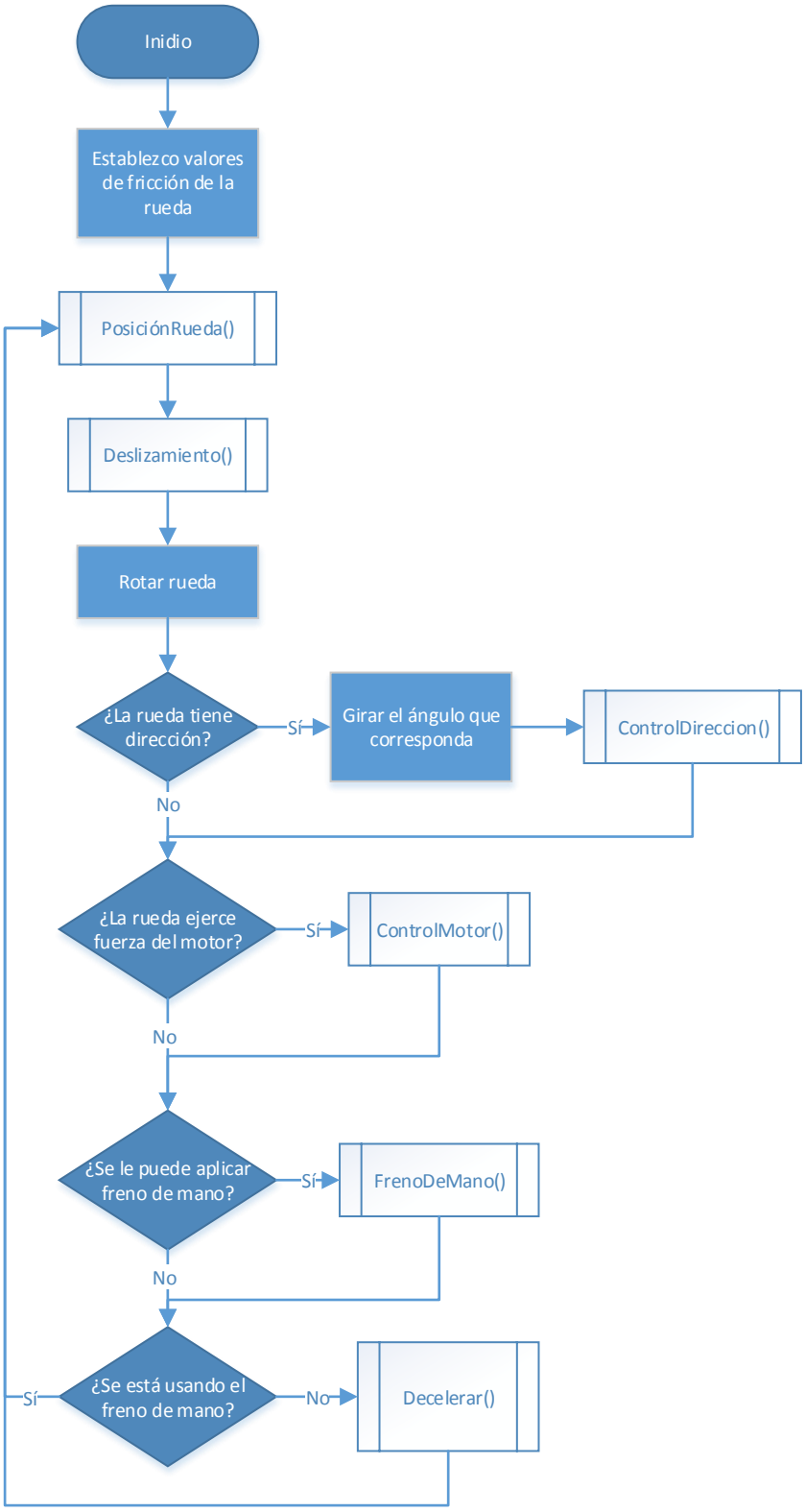

Figura 4.26. Diagrama de flujo de las ruedas

La función *PosicionRueda()* se encarga de mantener la rueda en la posición correcta teniendo en cuenta la suspensión del automóvil.

La función *ControlDireccion( )* sirve para girar las ruedas encargadas de la dirección del coche, para ello se modifica el *steerAngle* de los *wheelCollider*. Para esto además se ha utilizado una curva que haga que el efecto del giro del volante sea más suave a velocidades más elevadas para facilitar la conducción. Para hacerlo se ha utilizado otra *AnimationCurve* como se muestra en la figura 4.27. En la figura, el eje X representaría la velocidad del automóvil mientras que el eje Y sería la cantidad de giro que se ejerce con el volante.

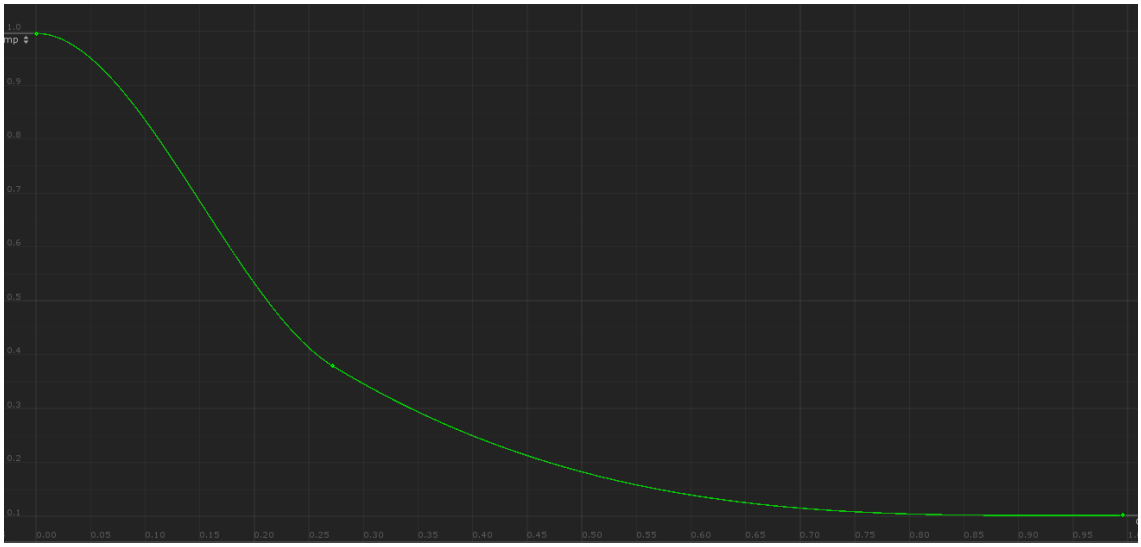

Figura 4.27. *AnimationCurve* para el giro del volante

En cuanto a *ControlMotor()* se encarga de calcular y modificar el *motorTorque* de los *WheelColliders*, utilizando la curva mencionada anteriormente del motor, los valores de los ratios y la entrada de aceleración que se esté dando.

La función *FrenoDeMano()* hace que si el freno de mano se encuentre activo, las ruedas asociadas al motor patinen. Esto se hace cambiado la *WheelFrictionCurve*, que es una curva asociada a los *Wheel Collider* que asignamos a las ruedas, y define la fricción de estas con el suelo.

La función *Deslizamiento()* cambia de una manera suave las curvas de fricción en función de si el coche está yendo marcha adelante, marcha atrás o está girando.

La función *Decelerar()* se encarga de que si no se está acelerando ni frenando, el automóvil vaya perdiendo velocidad poco a poco, como sucedería en un coche real.

4.3.2. Coches con inteligencia artificial (IA)

En este apartado se tratará sobre cómo se han realizado los coches con inteligencia artificial, es decir, que se mueven sin ser controlados por nadie (al contrario que el coche a conducir).

4.3.2.1. Componentes físicos de los coches IA

Para realizar los coches con inteligencia artificial, se ha utilizado varios modelos 3D, a los que se les han añadido una serie de componentes, al igual que para el coche a

conducir, para poder lograr un correcto funcionamiento. Lo primero sería el *Rigidbody*, definiendo de nuevo la masa del coche, la resistencia que presenta al viento e indicando que deben utilizar la fuerza de la gravedad.

También se han añadido dos *Box Collider* (el de la parte superior del coche y el de la parte inferior) con la opción *Is Trigger* desactivada, de modo que el coche no sea atravesado por otros objetos, y además otros dos *Box Collider* idénticos a los anteriores pero con esa opción activa, para que podamos detectar si algo está chocando con nuestros coches. Esto será necesario a la hora de guardar datos para reconocer si nuestro coche ha colisionado con otros.

De nuevo, a cada rueda del coche se le ha añadido un *Wheel Collider*, en el cual definimos el peso de la rueda, el radio y la suspensión, entre otros valores como valores de fricción.

Estos automóviles también cuentan con una fuente de sonido de modo que al pasar por el lado del coche conducido por el usuario, podríamos tener una sensación igual a la que se produce en la realidad cuando un coche pasa al lado del nuestro en la carretera.

4.3.2.2. Implementación de los coches IA

Los coches IA, al igual que el coche a conducir, tienen atribuidos 2 scripts, uno para el coche en general, y otro para cada una de las ruedas. La figura 4.28 muestra el diagrama de flujo de los coches IA.

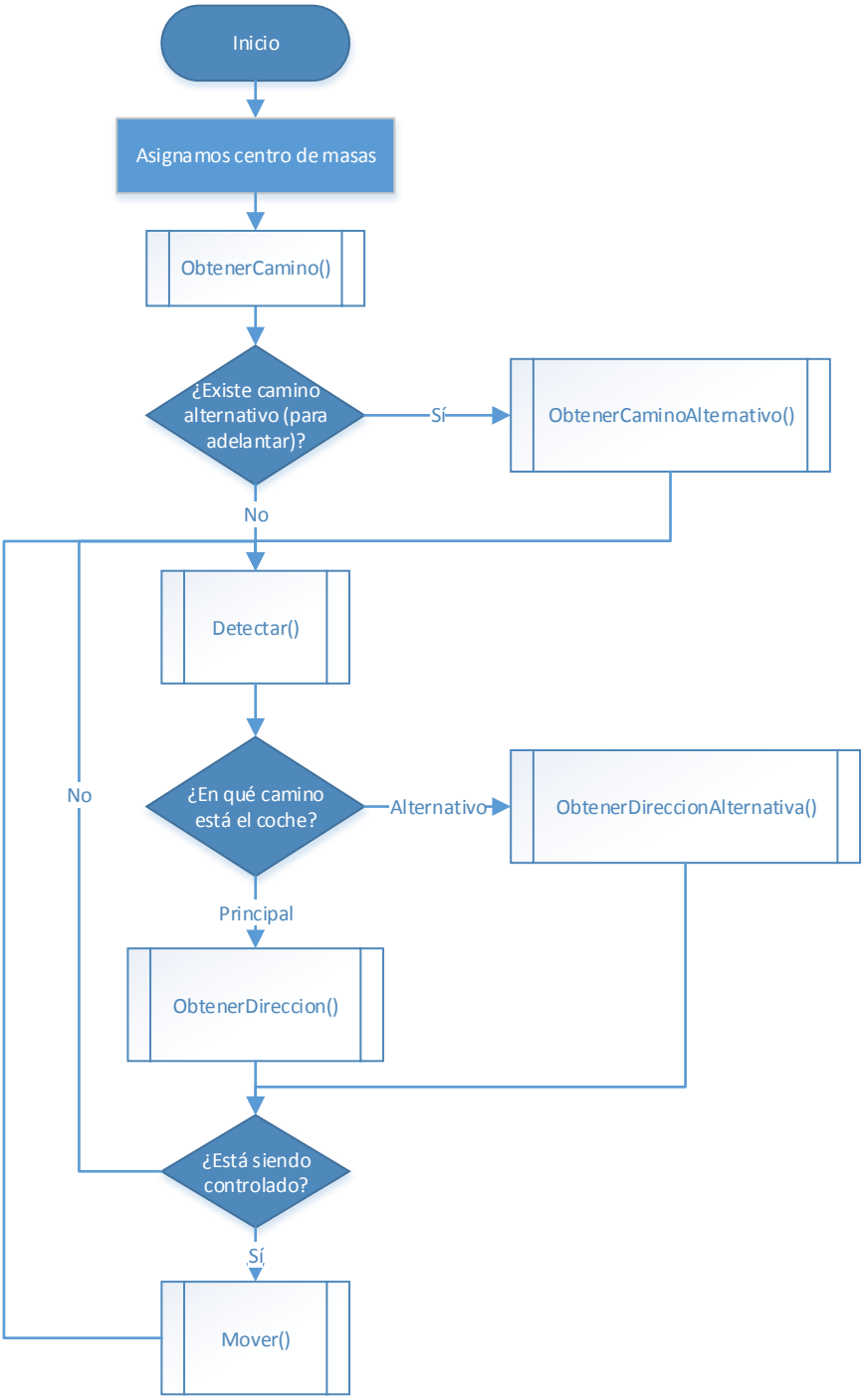

Figura 4.28. Diagrama de flujo coche IA

La función *Detectar()* se encarga de ver si hay algún otro coche cerca de nuestro IA para actuar en consecuencia. Podemos observar el diagrama de flujo de esta función en la figura 4.29. Para ello se lanzan una serie de *Raycast* desde diferentes lugares del coche en diferentes direcciones y ángulos, calculados previamente. Estos *Raycast* son líneas que se trazan desde diferentes puntos del coche, con las cuales podemos detectar si algo se interpone en su camino. Estos *Raycast* son mostrados en la figura 4.30.

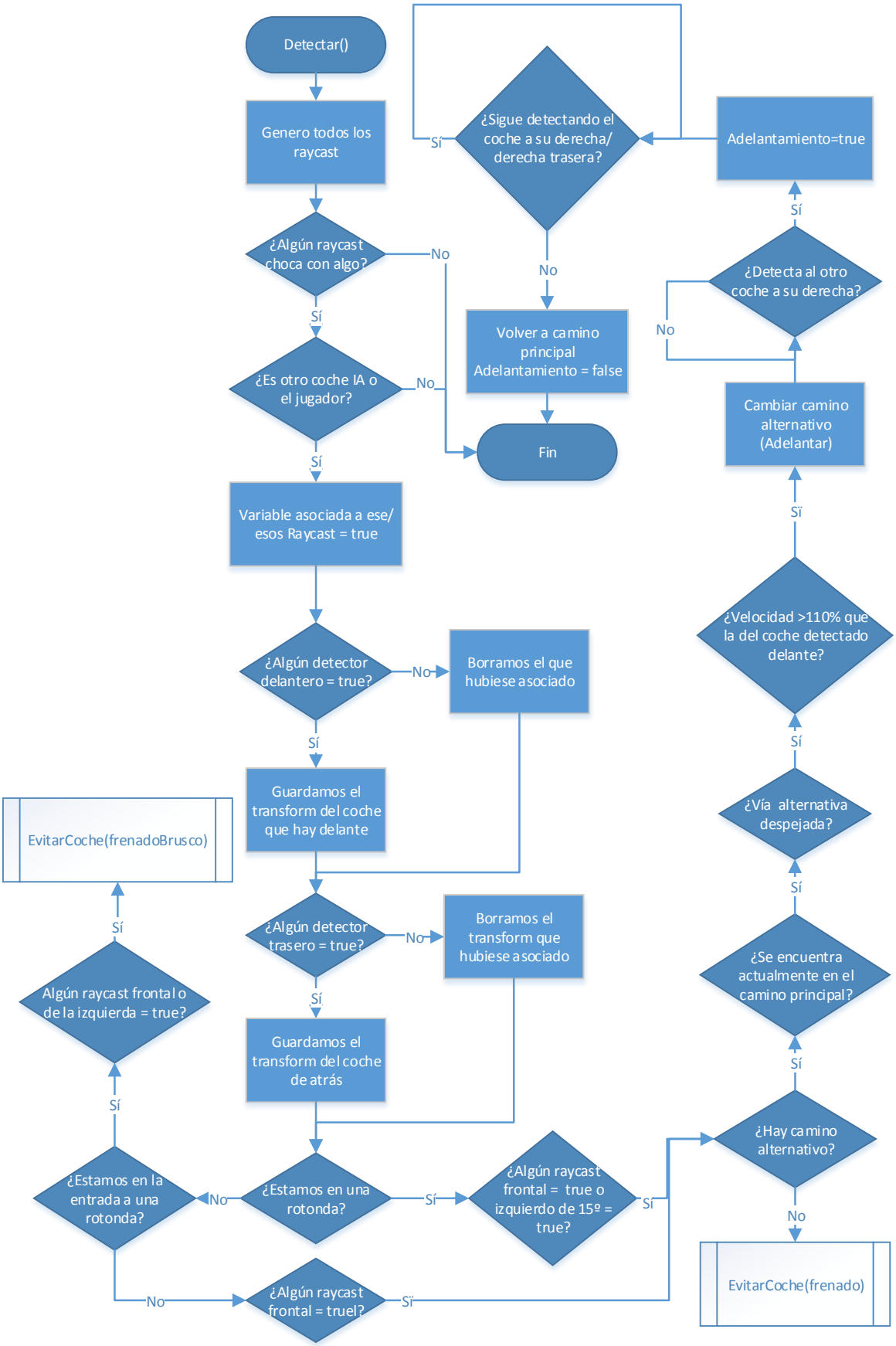

Figura 4.29. Función Detectar() de coches IA

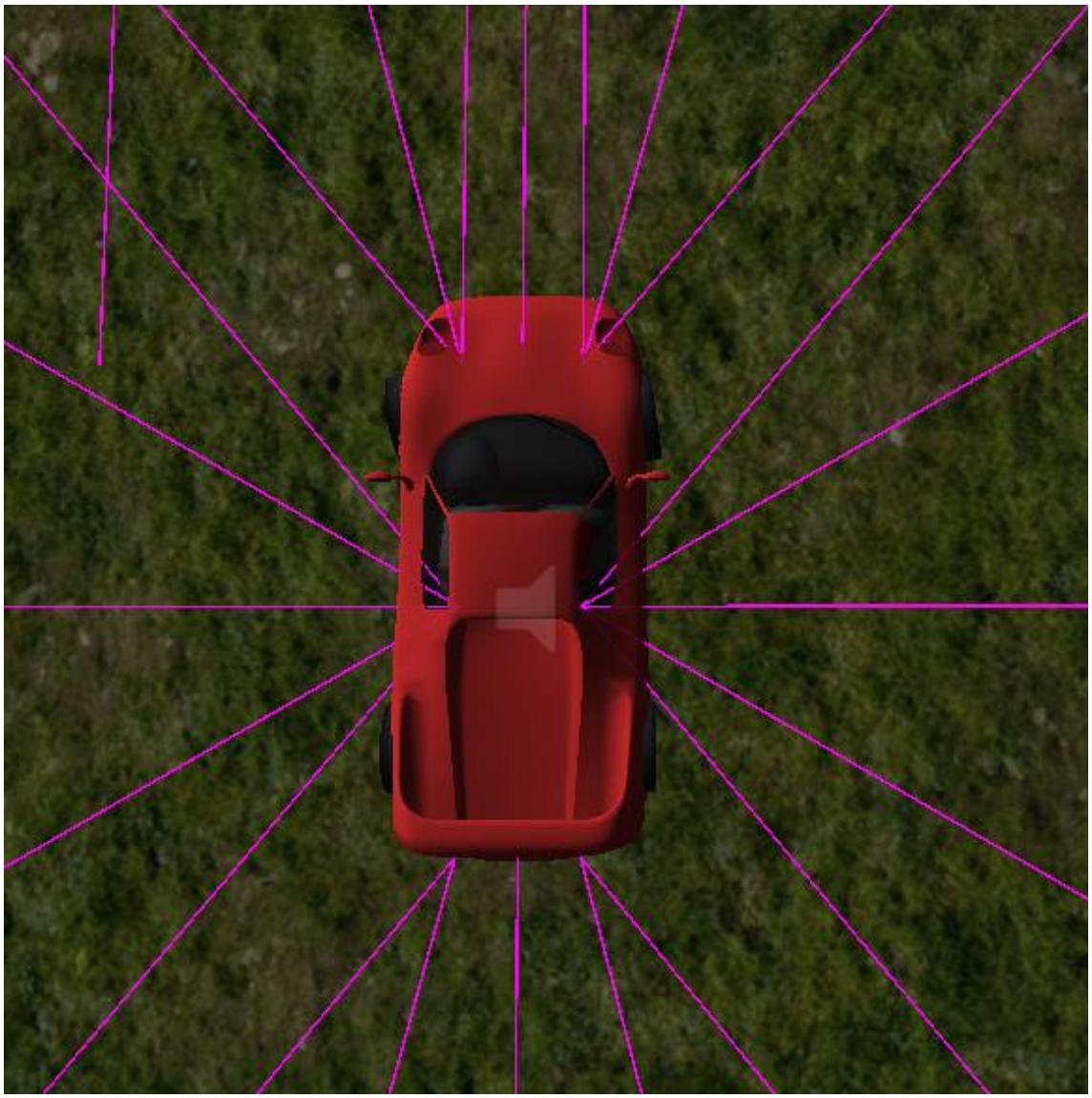

Figura 4.30. Coche IA con los diferentes *Raycast* (líneas rosas)

Las funciones para evitar coches utilizadas en *Detectar( )*, toman diferentes valores de frenado ya que si estamos a la entrada de una rotonda y vienen coches, el que va a entrar debe frenar del todo, mientras que si lo que tenemos es un coche delante que va más despacio solamente hay que frenar un poco. Lo que se hace es dejar de aplicar fuerza al motor (hacerla nula) y aplicar una fuerza de frenado equivalente al valor que pasemos a la función.

Volviendo al diagrama de flujo general del coche IA, las funciones de obtener el camino o el camino alternativo toman los elementos hijo de un camino formado por puntos (creado previamente en la escena), en los cuales cada hijo es uno de los puntos siguientes del camino. A cada coche le es atribuido el camino y en algunos casos el camino alternativo que le corresponde. A estos caminos se les ha añadido un script para que se muestre en color en la escena (ya que si no solo tendríamos objetos vacíos) y de este modo poder ver mejor por dónde va, como se puede observar en la figura 4.31.

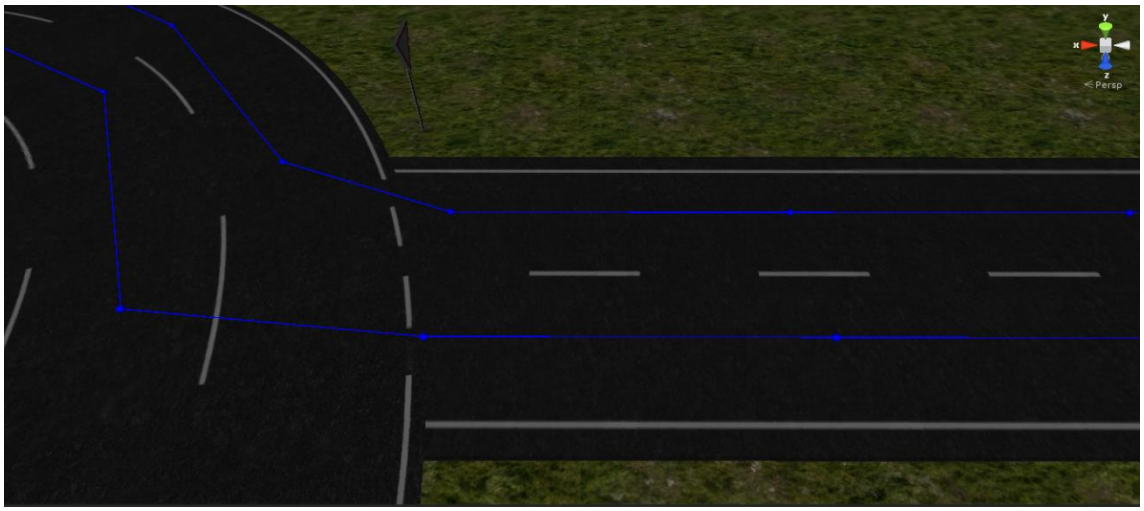

Figura 4.31. Fracción de escenario con caminos para los coches IA

Las funciones de obtener dirección sirven para girar las ruedas en función de dónde se encuentre el siguiente punto al que debe ir nuestro coche IA e ir obteniendo cuál es el punto al que tocaría llegar, y en el caso de que estemos en el último punto del recorrido podría terminar ahí o si queremos que haga un recorrido cíclico volvería al primer punto (esto lo determinaremos en cada coche IA mediante una variable booleana).

Por último la función *Mover()* se encarga de determinar la velocidad en función del radio de las ruedas y las vueltas por minuto que dan. Proporciona la máxima fuerza al motor siempre que estemos por debajo de la velocidad máxima que establezcamos (la podemos configurar independientemente para cada coche IA). Si alcanzamos la velocidad máxima deja de hacer fuerza el motor y se aplica una fuerza de frenado.

Para detectar si estamos en una rotonda o en una entrada de rotonda, contamos con *Colliders* en cada una de ellas con una etiqueta correspondiente a lo que es (rotonda, entrada, etc.). Por tanto mediante variables *booleanas* que sean *true* cuando entramos y pasen a ser *false* cuando salimos, podemos conocerlo. A otros coches con inteligencia artificial o el propio coche conducido por el usuario (u otros elementos como puede ser una señal de tráfico) también los distinguen leyendo cuáles son sus etiquetas.

4.3.3. Personas

Para crear personas animadas se utiliza un modelo 3D de una persona, que debe tener un esqueleto asociado de modo que podemos articular ese modelo como una persona real. Además necesitamos animaciones, que una vez aplicadas al modelo 3D hagan que este se comporte con el movimiento deseado.

Primero, una vez tenemos un modelo 3D, necesitamos indicar que el tipo de animación que deseamos es “humanoide”, como se muestra en la figura 4.32.

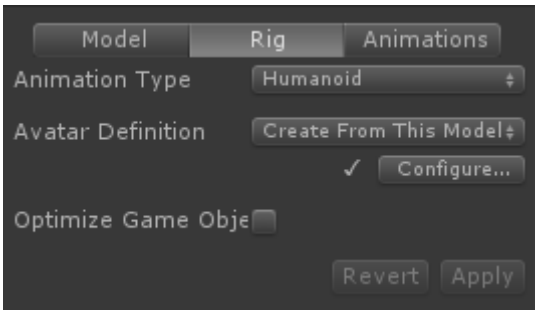

Figura 4.32. Inspector del modelo 3D de la persona

A continuación de esto, en *Configure* hay que comprobar que todos los nodos están en verde y el esqueleto está bien configurado (si no no funcionarán correctamente las animaciones que se apliquen a posteriori). Si no lo están hay que configurarlo correctamente. En la figura 4.33 se muestra el resultado del esqueleto correctamente configurado. Además en la figura 4.34 podemos ver cómo se mostraría el esqueleto diseñado sobre el modelo 3D de la persona a utilizar.

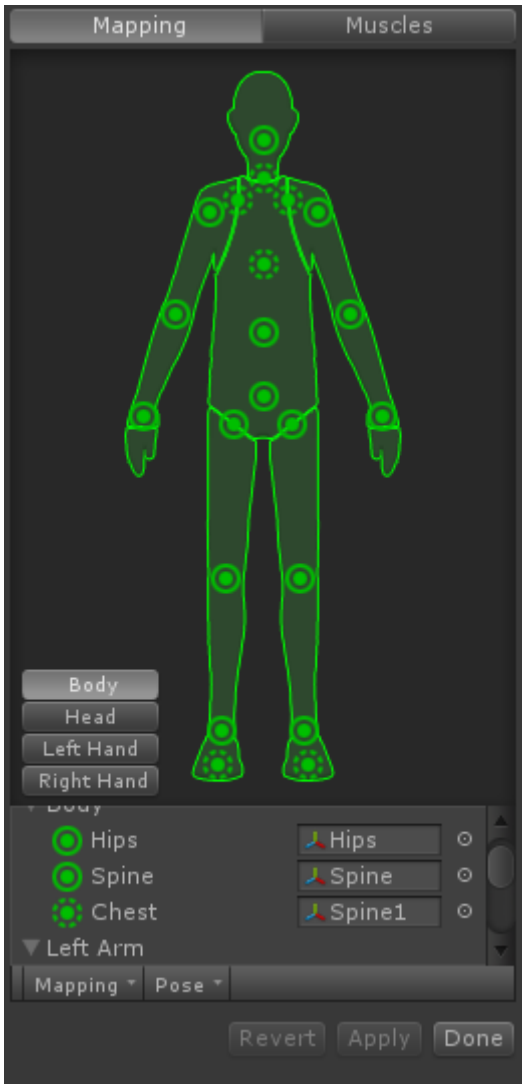

Figura 4.33. Esqueleto correctamente configurado

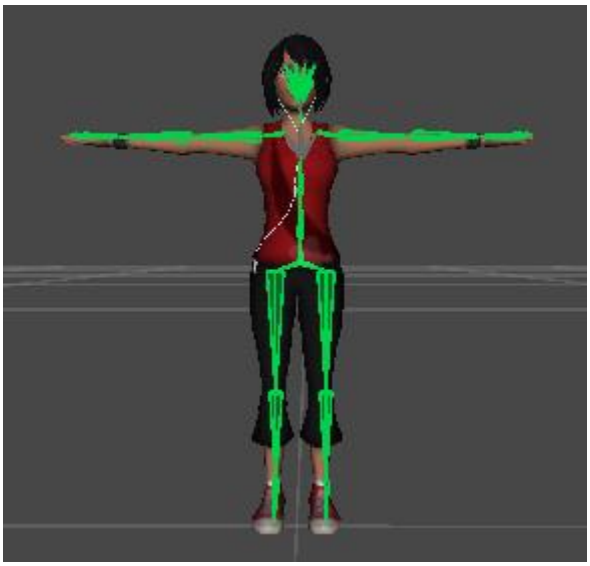

Figura 4.34. Esqueleto sobre el modelo 3D

Las diferentes animaciones que se le vayan a asignar deben ser configuradas del mismo modo, para que el esqueleto del modelo 3D y el de la animación coincidan (como antes, hay que seleccionar que es de tipo humanoide y configurar el esqueleto de manera idéntica). Además en la pestaña *Animations* del inspector debemos configurar cómo deseamos que sea la animación, si queremos que se repita en bucle, la velocidad a la que queremos que se reproduzca, etc.

En este caso se van a utilizar 3 animaciones diferentes: *Idle*, *WalkForward* y *Death*, siendo la primera una animación para cuando está de pie pero sin avanzar, la segunda para caminar hacia delante y la tercera para cuando es atropellado por un coche.

Una vez configuradas todas las animaciones, es necesario crear un animador, que debemos añadir como componente al modelo 3D que deseamos animar, de forma que aparezca en su inspector. Este componente se muestra en la figura 4.35.

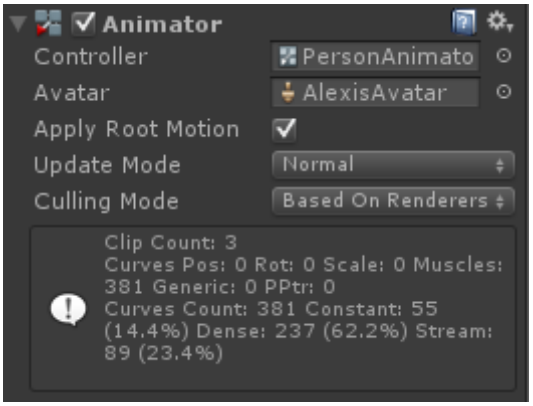

Figura 4.35. Componente animador en modelo 3D

El animador debe ser configurado de manera que cambie entre las diferentes animaciones en el momento en que lo deseemos. Para ello creamos diferentes estados

(*Idle*, *Walking* y *death*) a los que asignamos la animación correspondiente, e interconectamos entre sí los estados entre los que puede haber transiciones (solo en las direcciones que pueden ocurrir). Las transiciones que hemos utilizado en nuestro personaje se muestran en la figura 4.36.

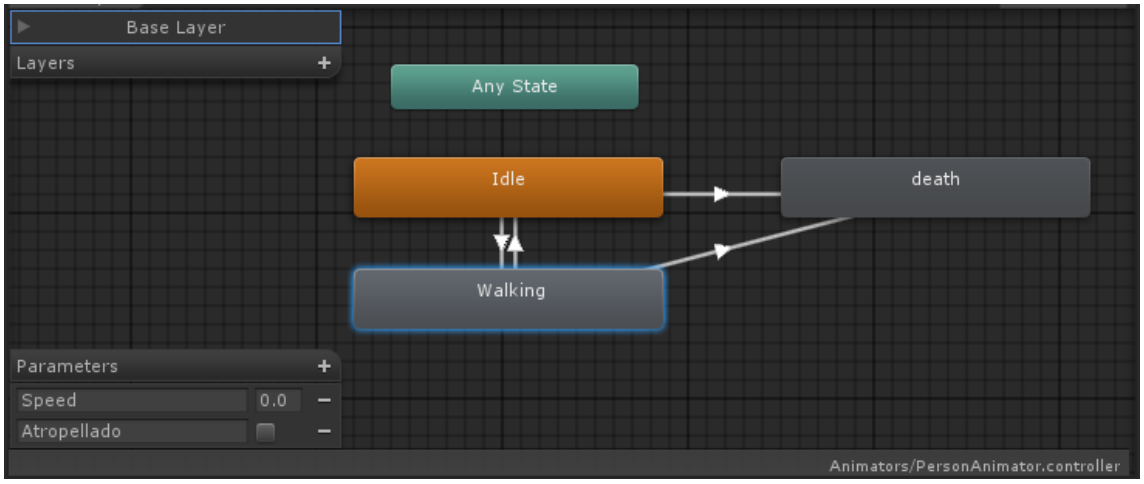

Figura 4.36. Transiciones en el animador

Además se deben crear los diferentes parámetros de los que dependan las transiciones entre los diferentes estados, en este caso la velocidad a la que camina la persona (mediante una variable de tipo *float*) y si ha sido atropellado o no (utilizamos una variable *booleana*). En función de estos parámetros editamos las transiciones, añadiendo condiciones acerca de qué debe suceder.

En el caso de pasar del estado *Idle* a *Walking*, esto se hace si la velocidad de la persona es superior a 0.1 (se está moviendo) y *Atropellado* = *false* (no ha sido atropellado). La transición inversa se haría si la velocidad es menor que 0.1 (se deja de mover) y no ha sido atropellado. Para pasar al estado *death* la única condición es que haya sido atropellado. La configuración de la transición de *Idle* a *Walking* se puede observar en la figura 4.37.

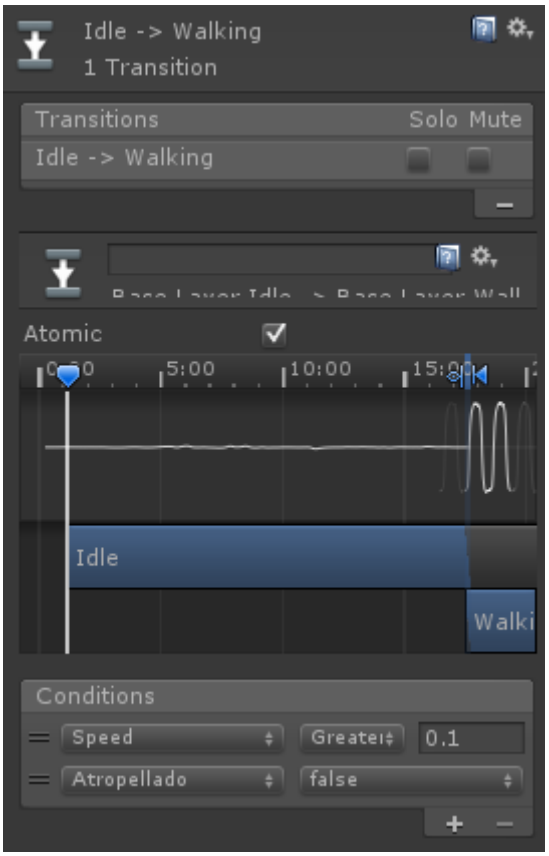

Figura 4.37. Transición de Idle a Walking

Los modelos de personas que se utilicen en el simulador tienen además que tener un componente *Rigidbody* para que actúen como un cuerpo real, de modo que activamos la opción de que use la gravedad, y configuramos su peso y otros parámetros del modo deseado. Además debe contar con dos componentes *Collider* diferentes, uno con la opción *Is Trigger* desactivada de modo que el elemento no pueda ser atravesado por otros, y otro con la opción activada, para que se pueda detectar si algún otro elemento ha chocado con el cuerpo de la persona (ver si ha sido atropellada).

Para que todo esto funcione correctamente debemos asignar a la persona un Script que defina su comportamiento. En nuestro caso, solo se ha utilizado una persona en el escenario 1.

A continuación, en la figura 4.38 se muestra el diagrama de flujo del funcionamiento de la persona, para lo cual se debe tener en cuenta que la persona pasa a estar activa cuando el coche que conducimos pasa por un detector (que comprueba que lo que está atravesándolo es el jugador, es decir, tiene como etiqueta "Player"). Esto ocurre cuando el automóvil está suficientemente cerca del paso de peatones. Antes de ello la persona se encontrará en estado *Idle*.

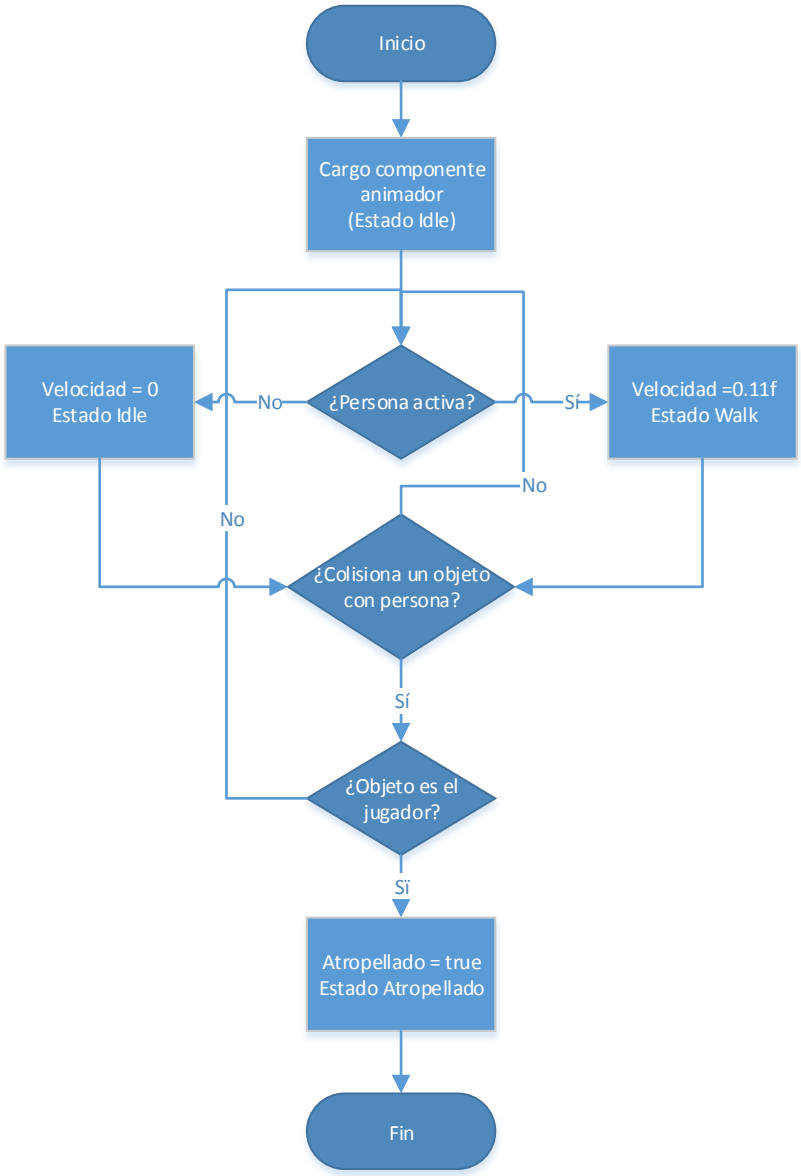

Figura 4.38. Diagrama de flujo de Persona

4.3.4. Controladores de juego

Para el desarrollo de las simulaciones, en cada escenario contamos con diferentes controladores de juego que se encargan entre otras cosas de activar o desactivar los coches IA, controlar el tiempo de la simulación, etc.

En cada escena hay un objeto que se encarga del control general del juego, del cual se muestra un diagrama de flujo en la figura 4.39.

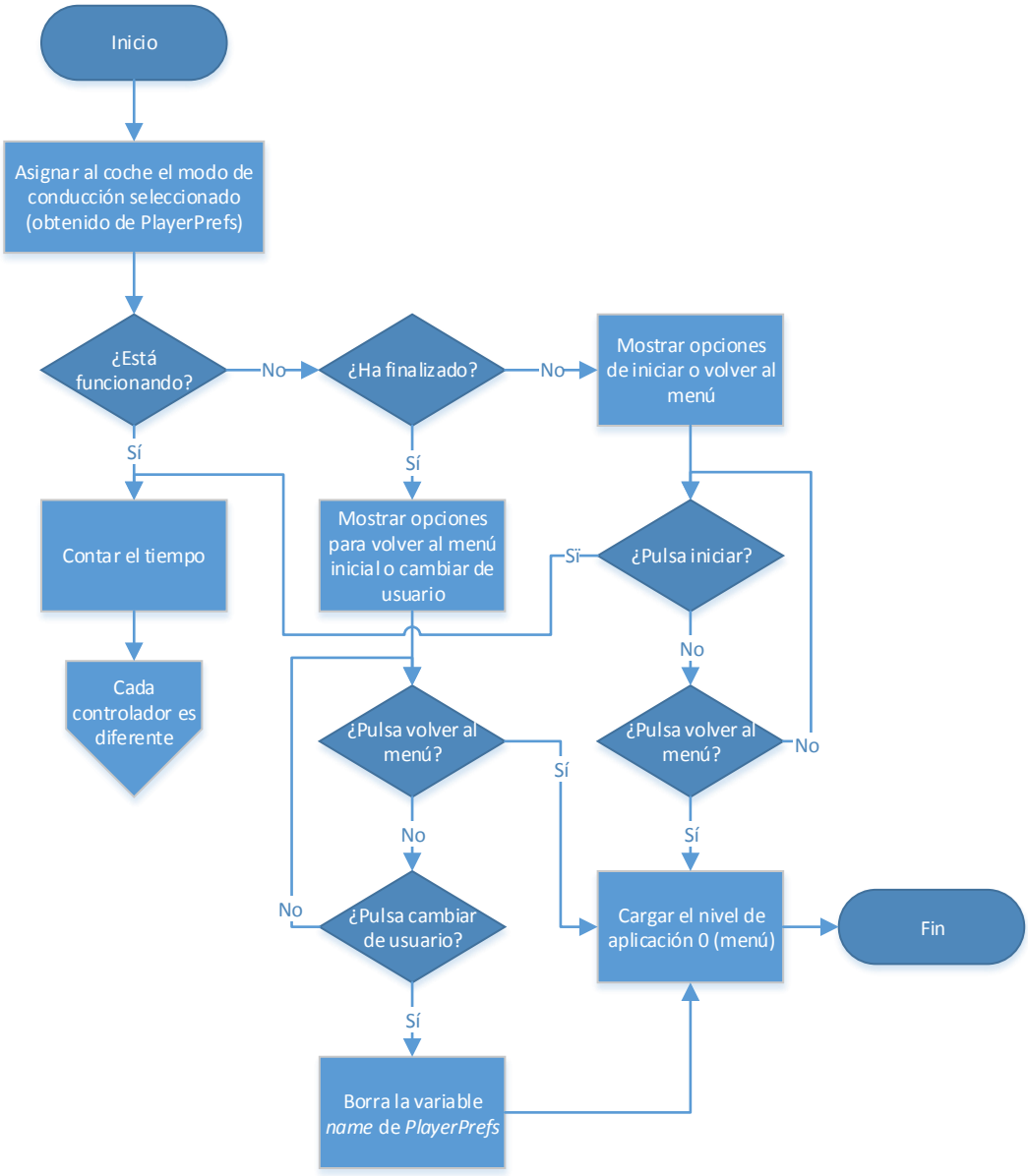

Figura 4.39. Controlador de juego

A partir de la figura con forma de pentágono, cada controlador de juego realiza diferentes funciones, ya que cada escenario funciona de distinto modo. En todos los casos si el usuario pulsa la tecla escape durante la simulación, esta será pausada y se ofrecerán las opciones de salir al menú o continuar. Además todos los controladores se encargan de borrar la variable "name" de *PlayerPrefs* si cerramos la aplicación desde cualquier escena.

En la primera escena una vez que empieza a contar el tiempo se activa el coche IA que circula en esa misma escena, y una vez que se llega cerca del punto donde debe tomar una dirección, se encarga de mostrar una imagen indicando la dirección. En el caso del segundo escenario, una vez que comienza a contar el tiempo se activan todos los coches de este. En el tercer, cuarto, quinto y sexto escenarios, no es este controlador el que se encarga de controlar los coches IA, sino otros que serán mencionados posteriormente. Cabe destacar que en el caso del quinto escenario, este controlador

además se encarga de llevar todos los coches IA a su posición inicial cada vez que el usuario ha dado una vuelta completa al recorrido.

Además en las diferentes escenas existen otros objetos vacíos que forman parte del control del juego ya que mediante scripts detectan qué es lo que pasa a través de ellos y actúan en consecuencia.

En el primer escenario contamos con uno para destruir el coche automático una vez que llega a su fin. Lo único que hace es que cuando algo atraviesa este objeto (lo detecta con un *Collider* con la opción *isTrigger* activa), comprueba si es un coche IA viendo si su etiqueta se corresponde con "CocheAutomatico" y si es así, destruye el "game object" correspondiente a ese coche, mediante la función *Destroy( )*. También tenemos otro objeto de ese tipo para que cuando el coche que conducimos llegue a cierto punto, se nos indique la dirección que debemos tomar. Para ello detecta si el objeto que lo atraviesa es el jugador, comprobando si tiene la etiqueta "*Player*". Si es así pone una variable booleana llamada *girar* a *true*, de modo que el controlador general de la escena muestra una imagen con una flecha indicando que se debe girar hacia la izquierda. Otro objeto de este tipo se encuentra al final del recorrido. Ese se encarga de indicar al control de juego general que el jugador (con etiqueta "Player") ha llegado al final del recorrido, activando su variable booleana *isFinished* y desactivando la de *isRunning*. De este modo parará la simulación y se mostrarán las opciones de "Volver al menú" o "Cambiar de usuario". También tenemos un objeto de control para que cuando el jugador esté cerca del paso de peatones, se active la persona que está al lado (llamando al script encargado de controlar a la persona) y esta comience a cruzarlo. Por último tenemos otro controlador antes de la señal de stop, detecta si el jugador para en el stop o no. Si lo hace, llamamos a la función correspondiente a guardar datos enviándole un mensaje de que el jugador ha parado correctamente en el stop, y si no lo hace enviamos un mensaje de que no ha parado.

En el segundo escenario tenemos, al igual que en el primero, un objeto al final del recorrido que se encarga de detectar si el jugador ha llegado ahí. Si es así hace lo mismo que en el primer escenario. Aquí además tenemos *Colliders* en todas las rotondas y en todas sus entradas, para que cuando los coches con inteligencia artificial estén en una rotonda o en la entrada a una, lo sepan y actúen en consecuencia. Esto se diferencia poniéndole etiquetas a los distintos *Collider* (en este caso hemos utilizado los nombres de etiqueta *Rotonda* y *EntradaRotonda*).

En el tercer escenario como en los anteriores contamos con el detector del final del recorrido. También como en el caso anterior contamos con los *Collider* de las rotondas y las entradas de las rotondas para el funcionamiento correcto de los coches IA. Además tenemos una serie de objetos "activadores" (tantos como el número de rotondas en la escena, en este caso 12), que utilizan el mismo script (pero con diferentes objetos atribuidos) y están etiquetados como "*ActivarX*", siendo X el número de rotonda para saber qué coches activar en cada momento (los que vayan por esa rotonda). Estos se encuentran un tramo antes de que el conductor llegue a la siguiente rotonda, y una

vez que detectan que el coche ha llegado a ese punto, hacen que se muestre una imagen por pantalla indicando la salida que deben tomar en la rotonda correspondiente, ponen una variable booleana que indica que eso está activo a *true* (esto será utilizado posteriormente) y activan un temporizador correspondiente a ese tramo. En cada uno se van activando los coches que circularán por esa rotonda, unos en cuanto se llega a este detector y otros los activamos algún segundo más tarde utilizando el temporizador. En las salidas de cada rotonda que se corresponden con la que debería tomar el coche, tenemos una serie de "desactivadores" etiquetados como "*SalirRotonda*" (a los que asignamos el "activador" que le corresponde) que se encargan de que cuando el coche del usuario pase por ellos, ponen la variable de su objeto "activador" correspondiente de nuevo a *false*, y por tanto se deja de mostrar la imagen de la dirección que se debía tomar. En la figura 4.40 vemos las diferentes imágenes que van apareciendo a lo largo del recorrido del escenario 3.

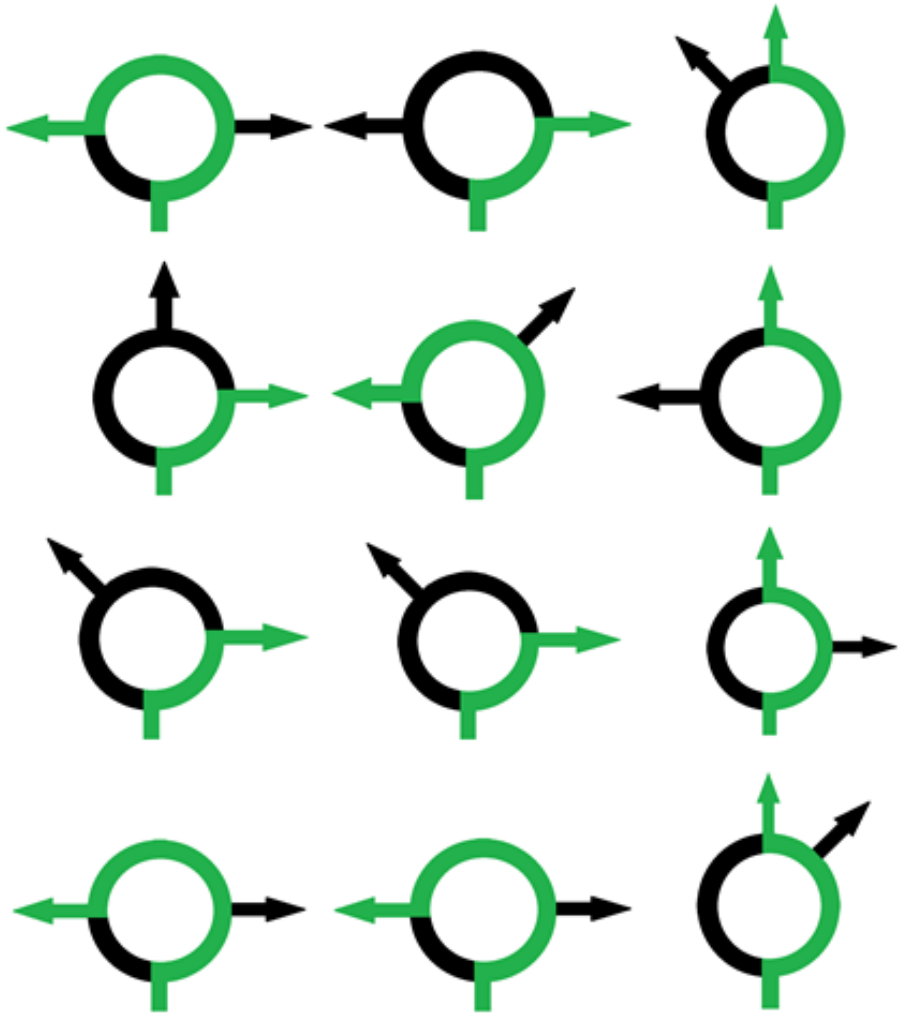

Figura 4.40. Indicadores del camino a seguir correspondientes al escenario 3

En el cuarto escenario tenemos el objeto del final como en los anteriores. Además tenemos una serie de objetos "activadores" y "desactivadores" al inicio y al fin de un tramo en el que se deba tomar una decisión de dirección, solo que en este caso solo se encargan de activar y desactivar la imagen que muestra el camino a tomar, tanto

en intersecciones con semáforos como en rotondas. La figura 4.41 muestra algunas de las imágenes de sus indicadores.

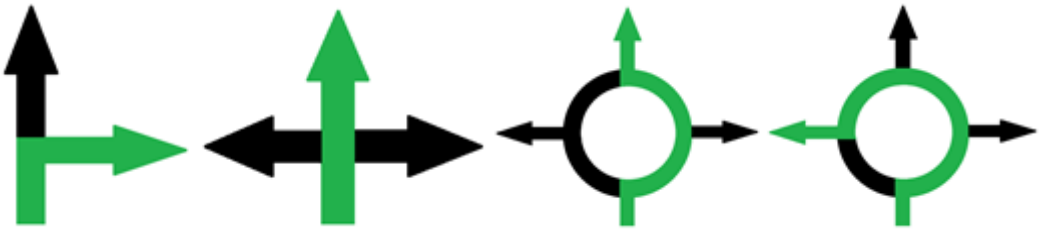

Figura 4.41. Ejemplos de indicadores del escenario 4

En este escenario además utilizamos otra serie de detectores en los semáforos que se encargan de detectar si el usuario pasa un semáforo en rojo. En ese caso, llamaríamos a la función correspondiente a guardar datos enviándole un mensaje de que el jugador se ha saltado un semáforo en rojo, para que esto quede registrado en el log.

En los escenarios 5 y 6 se utiliza lo mismo que en los escenarios 3 y 4 respectivamente. La diferencia en el 5 es que se ha eliminado el detector del final del recorrido y se ha sustituido por un objeto que se encarga de transportarnos a la posición inicial (sin que apenas se note el cambio) de forma que podemos conducir de forma continua e ininterrumpida, ya que volveríamos a hacer el recorrido tantas veces como deseemos. En el 6 lo que cambia es que también se ha eliminado el detector del final y se han añadido algunos tramos más de recorrido para volver a la posición inicial y que se pueda volver a hacer el recorrido también de forma continua. Para ello se han modificado un poco los scripts porque por algunos de los activadores y desactivadores se pasa varias veces antes de volver al inicio y es necesario saber en qué momento debe mostrar cada indicación diferente.

4.4. Guardar datos

Para el almacenamientos de datos sobre la conducción en ficheros XML se han utilizado dos scripts: *SavePositionScript.cs* y *ColliderCollectDataScript.cs*.

El script *SavePositionScript.cs* se encarga de guardar los datos de la simulación de la escena que seleccionemos en dos ficheros *.xml*, uno con los datos iniciales y otro con la información generada durante simulación. Para ello primero en la función *Awake( )*, la cual se llama cuando la instancia del script está siendo cargada, se comprueba si existe el directorio en el que deseamos guardar los datos, y si no es así, se crea. Para no tener problemas con la ubicación del directorio, se utiliza como tal: *Application.dataPath+ "/infoProyecto"*. De este modo, sea cual sea el ordenador en el que se utilice, se creará un directorio llamado "infoProyecto" en la ubicación en la que se encuentre nuestra aplicación, dentro de la carpeta de datos que se genera para la aplicación.

En la función *Start( )* creamos el primer fichero *XML* con los datos iniciales, en el cual se incluye el nombre de la persona que ha utilizado el simulador (se le pedirá con

anterioridad), el nombre de la escena y el modo de conducción que ha seleccionado (automático, manual o con levas). Además se anotará la fecha y hora de inicio, y la posición inicial en la que se encuentra el vehículo a controlar de la escena. Este fichero tiene la apariencia que se muestra en la figura 4.42.

```
<?xml version="1.0" encoding="utf-8"?><!DOCTYPE DatosDeSeguimiento
 [
 <!ENTITY locations SYSTEM "log-42167.4777175798.xml">
 ]>
<Simulacion>
  <DatosIniciales>
    <Jugador>Rebeca</Jugador>
    <EscenaSimulada>Escenario 3</EscenaSimulada>
    <ModoConduccion>Automatico</ModoConduccion>
    <FechayHoraInicio>2015-06-12T11:27:54.8239172+02:00</FechayHoraInicio>
    <PosicionInicial>
      <x>135.1486</x>
      <y>0.7354</y>
      <z>-224.2515</z>
    </PosicionInicial>
  </DatosIniciales>
  <tracking>&locations;</tracking>
</Simulacion>
```

Figura 4.42. Fichero XML con los datos de inicio

Mediante la función *CollectData(string Event)* escribiremos el otro fichero *XML* en el que se almacenará el nombre del evento (qué es lo que sucede en ese instante), el tiempo en el que se produce, la posición del automóvil (en coordenadas (x, y, z)), la velocidad en ese momento y la velocidad media durante la simulación (en km/h), la distancia recorrida hasta el momento (en metros), las revoluciones por minuto (RPM), la marcha actual, el consumo en ese instante y el medio durante la simulación (litros/100km) además del consumo total a lo largo de la simulación (en litros). Esta función será llamada pasándole como evento "Actualizacion automática" cada cierta cantidad de segundos, en este caso hemos definido *interval*=1 (s). La figura 4.43 muestra un ejemplo de la información que se guarda en uno de los eventos de este tipo. Cuando la simulación finalice, esta función será llamada con el nombre de evento "Simulacion finalizada". Además esta función será llamada por el script *ColliderCollectDataScript.cs* cuando se produzca algún incidente o hecho relevante durante la simulación.

```
<Evento xmlns="Actualizacion automatica">
  <Tiempo>0h 0m 16s</Tiempo>
  <Posicion>
    <x>-440.3710</x>
    <y>2.1177</y>
    <z>113.4264</z>
  </Posicion>
  <Velocidad>
    <Actual(km/h)>51</Actual(km/h)>
    <Media(km/h)>33.8375931</Media(km/h)>
  </Velocidad>
  <DistanciaRecorrida(m)>125.01</DistanciaRecorrida(m)>
  <RPM>1502</RPM>
  <Marcha>4</Marcha>
  <Consumo>
    <Actual(l/100km)>6.15</Actual(l/100km)>
    <Medio(l/100km)>8.40</Medio(l/100km)>
    <Total(litros)>0.0101287821</Total(litros)>
  </Consumo>
</Evento>
```

Figura 4.43. Evento "Actualización automática" del fichero XML

El script *ColliderCollectDataScript.cs* en su función *Update( )* envía información sobre eventos ocurridos a la función *CollectData(string Event)* del otro script. Estos eventos incluyen incidencias como subir o bajar de una acera, chocar con una señal, coche o edificio, atropellar a un peatón o saltarse una señal de stop. Además a cada objeto se le atribuye una etiqueta a través de la que se pueda distinguir qué tipo de objeto es (coche, señal, peatón, etc.), y utilizarlo para decidir qué tipo de evento se ha producido.

Para que todo esto funcione correctamente el script *SavePositionScript.cs* se añade a un objeto vacío de la escena, y *ColliderCollectDataScript.cs* a los diferentes elementos de los cuales queremos que se almacenen eventos ocurridos.

En las escenas en las que se ha utilizado el plugin Road & Traffic System, no se ha puesto a los coches IA el script que hemos asignado en los demás casos, ya que en este caso los objetos son instanciados (no están en la escena inicialmente) y ese script tiene que estar asociado al objeto encargado de guardar datos de la escena. De este modo, para no modificar el script que teníamos (ya que también se podría hacer de ese modo), se ha creado otro script que ha sido asignado al coche que conducimos, de modo que es este el que detecta a los demás coches (los que tengan la etiqueta "CocheAutomatico") y hace la llamada a la función correspondiente si colisiona con algún otro coche.

5. Manual de usuario

En este apartado se comentarán cuáles son los controles a utilizar para el simulador, además de cómo se deberá proceder a la hora de realizar simulaciones con él.

5.1. Controles

Para el manejo de este simulador es posible utilizar dos tipos de controles diferentes. Por un lado puede ser controlado mediante el teclado y por otro con el periférico Logitech G27.

5.1.1. Teclado

En la figura 5.1 se muestran los diferentes controles del teclado y cuál es la función de cada uno.

Figura 5.1. Controles del teclado

1. Acelerar (Alternativa: tecla *W*)
2. Frenar (Alternativa: tecla *S*)
3. Embrague
4. Girar a la derecha (Alternativa: tecla *D*)
5. Girar a la izquierda (Alternativa: tecla *A*)
6. Aumentar marcha
7. Reducir marcha
8. Mirar hacia la derecha
9. Mirar hacia la izquierda
10. Freno de mano
11. Marcha primera
12. Marcha segunda
13. Marcha tercera
14. Marcha cuarta
15. Marcha quinta
16. Marcha atrás
17. Punto muerto

5.1.2. Logitech G27

En este apartado se verán los controles que se utilizarán mediante el periférico *Logitech G27* y las funciones de cada uno. La figura 5.2 lo muestra gráficamente.

Figura 5.2. Controles del periférico Logitech G27

1. Acelerar
2. Frenar
3. Embrague
4. Volante (girar)
5. Aumentar marcha
6. Reducir marcha
7. Mirar hacia la derecha
8. Mirar hacia la izquierda
9. Freno de mano
10. Marcha primera
11. Marcha segunda
12. Marcha tercera
13. Marcha cuarta
14. Marcha quinta
15. Marcha atrás
16. Punto muerto

5.2. Manejo del simulador

En este apartado se comentará cómo funciona el simulador, comentando tanto el menú que se presentará al inicio como el manejo del automóvil una vez estemos en uno de los escenarios.

5.2.1. Inicio, pausas y salir de la aplicación

Al ejecutar la aplicación aparecerá una ventana en la que se solicitará el nombre del usuario (como se muestra en la figura 5.3). Es necesario introducir un nombre ya que si no no se permitirá avanzar. A continuación se deberá pulsar el botón de *Aceptar*.

Figura 5.3. Inicio de la aplicación. Introducir nombre

Una vez hecho esto se llevará al usuario al menú de inicio, en el que se le dará la bienvenida y se le mostrarán las opciones del modo de conducción que desee: cambios de marcha automáticos, cambios utilizando levas, o cambio de marcha manual utilizando la palanca de cambios. Además se muestran las opciones de los diferentes escenarios en los que podría entrar. También incluye un botón para salir de la aplicación si se desea. Otra opción para salir de la aplicación es pulsando la tecla "Escape". Este menú de inicio es el que podemos observar en la figura 5.4.

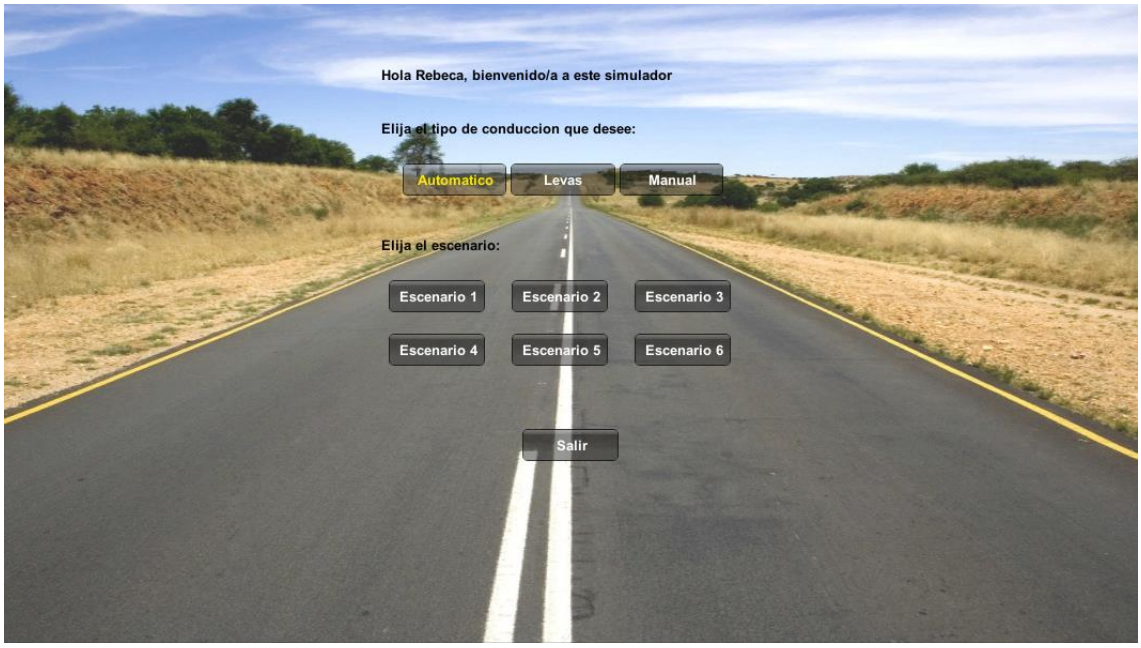

Figura 5.4. Menú: seleccionar escenario

Una vez que pulsemos sobre el escenario que deseamos ejecutar, accederemos a él y se mostrarán las opciones de inicio o volver al menú inicial, como se observa en la figura 5.5.

Figura 5.5. Inicio de una escena

Si pulsamos el botón de iniciar, el tiempo comenzará y ya se podrá conducir el automóvil. El tiempo de conducción se indicará en la esquina superior derecha. Si en cambio se pulsa la opción de volver al menú inicial, volveríamos a la pantalla en las que se muestran las opciones del modo de conducción y escenarios.

Este simulador también incluye una opción de pausado una vez que estamos dentro de una escena de conducción. Esto se puede hacer pulsando la tecla “Escape”. En el caso de hacerlo, todo el simulador se pausará, tanto el automóvil conducido por el propio usuario, como el resto de coches que circulan por la escena, y se mostrará un mensaje que dice: “Ha pausado el simulador, ¿qué desea hacer?”, y a continuación se presentarán dos botones con las opciones de continuar (también realizable volviendo a pulsar la tecla “Escape”) y la opción de volver al menú inicial, por si no se desea continuar con la simulación. Esto se muestra en la figura 5.6.

Figura 5.6. Simulador pausado

Como también se puede observar en la figura 5.6., se muestra gráficamente por pantalla tanto la marcha en la que se encuentra el coche como la velocidad (esquina inferior derecha) y las revoluciones (a la izquierda de la velocidad) que lleva a cada instante. Estos indicadores funcionan mediante el giro del pivote rojo, que siempre señalará la velocidad y las revoluciones que llevemos actualmente. La marcha se indicará con el número correspondiente a la marcha en la que se encuentre (abajo en el centro). En el caso de estar en punto muerto se mostrará una "X" y en el caso de estar marcha atrás se mostrará una "R". Podemos ver un zoom en la parte de los indicadores en la figura 5.7.

Figura 5.7. Zoom en la parte de los indicadores

Además, en la pantalla de la escena, arriba a la derecha se indica el tiempo que se lleva conduciendo y, en el caso de que tengamos que tomar una salida concreta (en rotondas o intersecciones), aparecerá una imagen indicando el camino que se debe tomar, como podemos ver en la figura 5.8. Esa imagen desaparecerá una vez se haya tomado el camino correcto.

Figura 5.8. Escenario durante la simulación con indicador de dirección

5.2.2. Control del automóvil

El automóvil a manejar presenta 3 posibilidades diferentes de conducción: automático, con levas o manual.

En todos estos casos los controles funcionan igual (son los indicados en el apartado 5.1.), excepto a la hora de cambiar las marchas.

En el caso de elegir automático, no será necesario el uso del embrague, ni las de las marchas 1ª, 2ª, 3ª, 4ª, 5ª ni marcha atrás. Las que se utilizarán serán las de aumentar/reducir marcha (las teclas *P* y *O* en el teclado, y las levas en el periférico *Logitech G27*). Al inicio de la simulación el coche se encontrará en punto muerto. Será necesario pulsar el control correspondiente a "Aumentar marcha" para indicar que se desea ir hacia delante, o el de "Reducir marcha" para indicar que se desea ir marcha atrás. Una vez estemos en marcha atrás, si pulsamos el control "Aumentar marcha" pasaremos a punto muerto, y si lo volvemos a pulsar estaremos marcha adelante (se meterá automáticamente la primera marcha". En el caso de estar marcha adelante, si pulsamos el control "Reducir marcha" pasaremos a punto muerto, y si lo volvemos a pulsar estaríamos en marcha atrás. Una vez estemos marcha adelante con la primera marcha metida, el propio simulador se encargará de ir cambiando a la marcha adecuada durante la conducción.

En el caso de elegir cambio mediante levas, los pasos entre marcha atrás – punto muerto – marcha adelante serán los mismos que en el caso comentado anteriormente (opción automática). La diferencia estará en que una vez que estemos en marcha hacia adelante, los cambios entre marchas no se realizarán solos, sino que se deberán ir pulsando los controles de "Aumentar marcha" o "Reducir marcha" para subir o bajar de marcha respectivamente.

El caso de control manual es el más complejo de todos, ya que en este caso se deberá utilizar el embrague, mientras indicamos a qué marcha deseamos cambiar. En este caso para dar marcha atrás será necesario pulsar el control de "Embrague" (tecla *Z* en el teclado o pedal de embrague en el *Logitech G27*) a la vez que pulsamos el de "Marcha atrás" (mediante la tecla 0 en el teclado o metiendo la marcha correspondiente en la palanca de cambios de *Logitech G27*). Para punto muerto no será necesario pulsar el embrague, simplemente el control correspondiente a punto muerto (tecla *X* en el teclado o la correspondiente posición en la palanca de cambios). En el caso de las marchas 1ª, 2ª, 3ª, 4ª y 5ª se deberá pulsar el embrague a la vez que el control correspondiente a esa marcha (teclas del 1 al 5 en el teclado o la posición correspondiente a esas marchas en la palanca de cambios).

Como se ha comentado anteriormente, el resto de controles funcionan igual en cualquiera de los casos (acelerar, frenar, mirar a la derecha/izquierda, girar y freno de mano), pulsando los controles correspondientes a cada una de las acciones.

6. Pruebas realizadas

Se han realizado una serie de pruebas con diferentes usuarios para analizar el funcionamiento del simulador y los resultados que se han obtenido durante estas pruebas. Para ello, se ha realizado una encuesta previa a los usuarios en la que se recogen datos como su edad, su estado de somnolencia en el momento de la simulación, la probabilidad de quedarse dormido en ciertas situaciones y su experiencia como jugadores de videojuegos. En la tabla 6.1 se recogen datos acerca de la edad, la experiencia en conducción y la experiencia en videojuegos de los usuarios.

	Usuarios						
	1	**2**	**3**	**4**	**5**	**6**	**7**
Edad	23	27	25	21	26	25	42
Años con carnet de conducir	5	4	7	3	8	7	13
Jugador de videojuegos	Sí	Sí	Sí	Sí	Sí	Sí	No
Experiencia como jugador (0-10)	6	7	5	7	9	6	0
Experiencia en juegos de coches (0-10)	3	5	7	6	8	2	0

Tabla 6.1. Datos acerca de la edad y experiencia de los usuarios

En la tabla 6.2 tenemos información acerca de la probabilidad de que los usuarios se queden dormidos en diferentes situaciones de la vida real, siendo 0 una probabilidad nula o muy baja, y 3 una probabilidad alta. Concretamente se ha utilizado como cuestionario el *Epword Sleepiness Scale* (ESS), un instrumento eficaz para medir la somnolencia diurna. (Johns, M.W., 2006)

Situación	**Usuarios**						
	1	**2**	**3**	**4**	**5**	**6**	**7**
Sentado y leyendo (0-3)	1	1	2	1	0	0	1
Viendo la televisión (0-3)	1	1	1	1	1	0	1
Sentado, inactivo en un lugar público (ej. Cine, teatro, etc.) (0-3)	0	0	0	0	0	0	0
Como pasajero de un coche en un viaje de 1 hora sin paradas (0-3)	1	0	1	0	1	0	0
Estirado para descansar al mediodía cuando las circunstancias lo permiten (0-3)	1	1	2	1	1	1	1
Sentado y hablando con otra persona (0-3)	0	0	0	0	0	0	0
Sentado tranquilamente después de una comida (sin alcohol) (0-3)	0	1	2	2	2	1	1
En un coche parado por el tráfico unos minutos (0-3)	0	0	0	0	1	0	0
Puntuación total (máx. 24)	**4**	**4**	**8**	**5**	**6**	**2**	**4**

Tabla 6.2. Resultados de *Epword Sleepiness Scale* (ESS) en los usuarios

Según el resultado de este test, tenemos que si la puntuación total se encuentra entre 1 y 6, se está durmiendo lo suficiente, entre 7 y 8 estaría en la media, y a partir de 9 se encontraría en un estado muy somnoliento, por lo que sería recomendable visitar un

médico. En el caso de nuestros resultados tenemos que todos los usuarios excepto el 3 están durmiendo lo suficiente, mientras que el usuario 3 se encuentra en la media. A partir de estos resultados obtenemos que, de las personas que han realizado las pruebas, el usuario 3 es el que cuenta con una mayor somnolencia diurna, seguido del usuario 5.

La tabla 6.3. muestra información acerca del estado de somnolencia de cada uno de los usuarios a la hora de comenzar a realizar las simulaciones. Para ello se han realizado el *Karolinska Sleepiness Scale* (KSS) y el *Standford Sleepiness Scale* (SSS) para evaluar la somnolencia subjetiva en el momento. (Johns, M.W. & Hocking, B., 2009)

	Usuarios						
	1	2	3	4	5	6	7
Estado de somnolencia en el momento de realizar la simulación (KSS) (1-10)[1]	7	3	3	4	5	5	4
Estado en ese momento (SSS) (1-7)[2]	3	2	2	2	2	3	2

Tabla 6.3. Estado de somnolencia de los usuarios antes de la conducción. Cuestionarios KSS y SSS

De los resultados de estos cuestionarios podemos observar que el usuario que se encontraba en un mayor estado de somnolencia antes de comenzar a realizar las pruebas de la simulación es el usuario 1. En el segundo lugar se encontraría el usuario 6 y en tercero el usuario 5.

A la hora de realizar pruebas, estas han sido realizadas en el escenario 2 primero en modo automático y después manual, a continuación se ha ejecutado el escenario 4, también en modo automático y manual. Después se ha conducido en el escenario 3 en modo manual, y finalmente se ha hecho en el escenario 6 durante 20 minutos en modo manual. En la figura 6.1. podemos observar a un usuario probando el simulador (utilizando el controlador *Logitech G27 Racing Wheel*).

[1] 1 = Extremadamente en alerta.
2 = Muy en alerta.
3 = En alerta.
4 = Prácticamente en alerta.
5 = Ni en alerta ni dormido.
6 = Está notando algunos signos de somnolencia.
7 = Con sueño, pero sin hacer esfuerzos por mantenerse despierto.
8 = Con sueño, y haciendo un leve esfuerzo por mantenerse despierto.
9 = Con mucho sueño, realizando grandes esfuerzos por mantenerse despierto.
10 = Extremadamente soñoliento, se queda dormido todo el tiempo.

[2] 1 = Se siente activo y vital, alerta, completamente despierto.
2 = Funcionamiento alto, pero no máximo, capacidad de concentrarme.
3 = Relajado, despierto, no completamente alerta, reactivo.
4 = Un poco apagado, no al máximo, disminuido.
5 = Apagado, empieza a perder el interés por estar despierto.
6 = Soñoliento, prefiero estar acostado, luchando con el sueño, confuso y aturdido.
7 = Casi en sueño, comienzo del sueño inmediato, incapacidad de permanecer despierto.

Figura 6.1. Usuario realizando pruebas de conducción

En la tabla 6.4. se recoge información acerca de la conducción de cada uno de los usuarios en todos los escenarios mencionados. En ella se encuentra el tiempo empleado en realizar la simulación, la distancia recorrida, la velocidad media que se ha llevado, el consumo medio de combustible y el consumo total; todos estos parámetros para cada una de las escenas y para cada uno de los usuarios. Podemos observar que algunos usuarios tienen más distancia recorrida (lo que aumenta el total de consumo final), ya que en ocasiones se han salido del carril por el que deben circular, recorriendo más espacio. En esta tabla podemos observar las relaciones entre todos los parámetros y comparar el comportamiento entre cada uno de los diferentes usuarios en los mismos escenarios, así como el comportamiento de un mismo usuario en distintos escenarios o utilizando distinto tipo de conducción.

		Escenario 2		Escenario 4		Escenario 3	Escenario 6
		Automático	Manual	Automático	Manual	Manual	Manual
Usuario 1	Tiempo	1m 18s	1m 17s	8m 10s	7m 18s	18m 29s	20m
	Distancia recorrida (m)	1.338,77	1.334,80	3.975,46	3.942,59	23.984,50	9.250,33
	Velocidad media (km/h)	70,63	70,98	29,89	32,83	83,09	27,85
	Consumo medio (l/100km)	5,15	6,94	9,69	8,35	7,05	8,74

	Consumo total (l)	0,063856	0,087914	0,239292	0,3045714	1,582793	0,636051
Usuario 2	Tiempo	1m 30s	1m 31s	7m 26s	9m 0s	22m 43s	20m
	Distancia recorrida (m)	927,67	1.063,03	4.009,22	4.902,82m	23.917,53	10.653,38
	Velocidad media (km/h)	41,12	46,74	32,98	33,08	63,45	32,08
	Consumo medio (l/100km)	7,77	7,47	8,38	10,34	5,46	10,80
	Consumo total (l)	0,063377	0,076748	0,224970	0,449557	1,184733	1,031815
Usuario 3	Tiempo	1m 29s	1m 17s	6m 42s	7m 57s	20m 35s	20m
	Distancia recorrida (m)	1.194,01	1.281,80	4.061,95	4.057,96	23.888,62	10.975,75
	Velocidad media (km/h)	51,96	66,59	36,70	31,05	70,17	33,05
	Consumo medio (l/100km)	6,46	7,77	7,83	10,35	5,47	8,73
	Consumo total (l)	0,06856354	0,106032	0,179057	0,358042	1,235850	0,923990
Usuario 4	Tiempo	1m 27s	1m 28s	7m 22s	8m 6s	17m 54s	20m
	Distancia recorrida (m)	1.078,98	1.114,79	4.015,93	4.012,39	24.002	9.463
	Velocidad media (km/h)	46,71	50,16	33,12	30,14	80,89	28,50
	Consumo medio (l/100km)	6,70	6,75	9,06	9,53	5,86	10,34
	Consumo total (l)	0,065871	0,072331	0,213667	0,289042	1,347250	0,768167
Usuario 5	Tiempo	1m 37s	1m 28s	8m 5s	8m 50s	18m 43s	20m
	Distancia recorrida (m)	958,56	910,59	4.029,13	4.311,31	23.925,72	9.788,92
	Velocidad media (km/h)	40,92	39,89	30,20	29,57	76,98	29,51
	Consumo medio (l/100km)	7,78	9,20	9,80	10,67	6,70	11,39
	Consumo total (l)	0,071569	0,079656	0,291613	0,424329	1,513507	0,937693
Usuario 6	Tiempo	1m 29s	1m 28s	7m 24s	8m 32s	20m 35s	20m
	Distancia recorrida (m)	1.106,17	982,25	4.075,72	4.066,34	23.924,24	10.170,87
	Velocidad media (km/h)	48,39	43,43	33,34	28,94	70,12	30,59
	Consumo	6,90	7,99	9,02	8,80	5,45	10,10

	medio (l/100km)						
	Consumo total (l)	0,070928	0,080637	0,214048	0,325041	1,228337	0,898139
Usuario 7	Tiempo	1m 20s	1m 29s	8m 5s	11m 40s	22m 23s	20m
	Distancia recorrida (m)	1.369,27	1.132,21	4.005,49	4.639,29	25.803,48	6.851,56
	Velocidad media (km/h)	69,76	48,24	30,14	24,04	69,57	20,63
	Consumo medio (l/100km)	2,89	7,56	9,58	10,31	8,06	9,38
	Consumo total (l)	0,039935	0,082484	0,254339	0,493272	2,029233	0,674103

Tabla 6.4. Datos de la conducción de los usuarios

Para mostrar parte de esta información de un modo más intuitivo, en la figura 6.2 se muestra un gráfico con el consumo medio de cada uno de los usuarios en cada escenario.

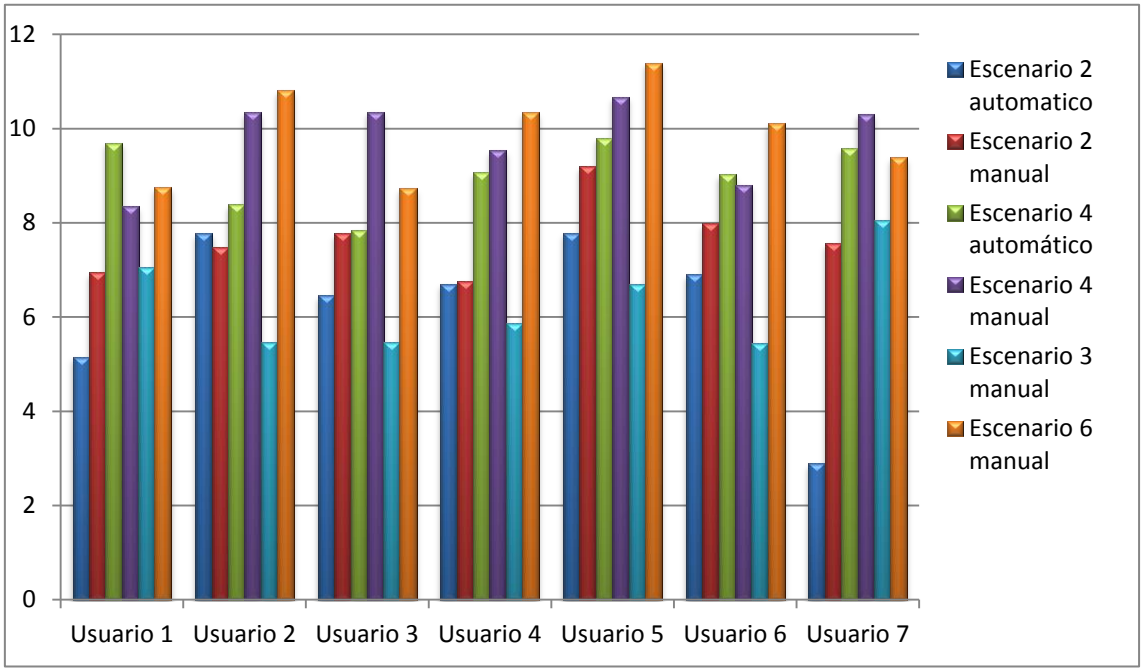

Figura 6.2. Consumo medio de los usuarios en cada escenario

En el gráfico podemos observar cómo por norma general, el consumo utilizando el modo de cambios automático ha sido menor que en modo manual, aunque hay alguna excepción. Esto es debido a que para ahorrar combustible es recomendable llevar poco revolucionado el automóvil, por lo que es bueno ir en marchas altas, mientras que los usuarios habituamos a cambiar algo más tarde. Además, teniendo en cuenta que los escenarios 2 y 3 serían recorridos interurbanos, mientras que el resto se asemejarían a un recorrido urbano, podemos observar que el consumo conduciendo por ciudad es bastante más elevado que conduciendo por carretera.

El usuario 5, a pesar de ser el más experimentado en videojuegos, especialmente de coches, es el que peores resultados ha obtenido en cuanto al consumo, ya que ha sido el que más ha tenido en todas las escenas, excepto en el escenario 3. Además cabe destacar que aunque por lo general se han obtenido mayores cantidades de consumo en modo automático que en manual, el usuario 4 ha tenido un consumo parecido en ambos casos (comparando entre los mismos escenarios).

Para el estudio de la velocidad media llevada por cada uno de los usuarios en estos escenarios, en la figura 6.3 se muestra un gráfico representando estos datos. En él podemos ver como la velocidad media en los escenarios interurbanos es mucho más elevada que en los urbanos.

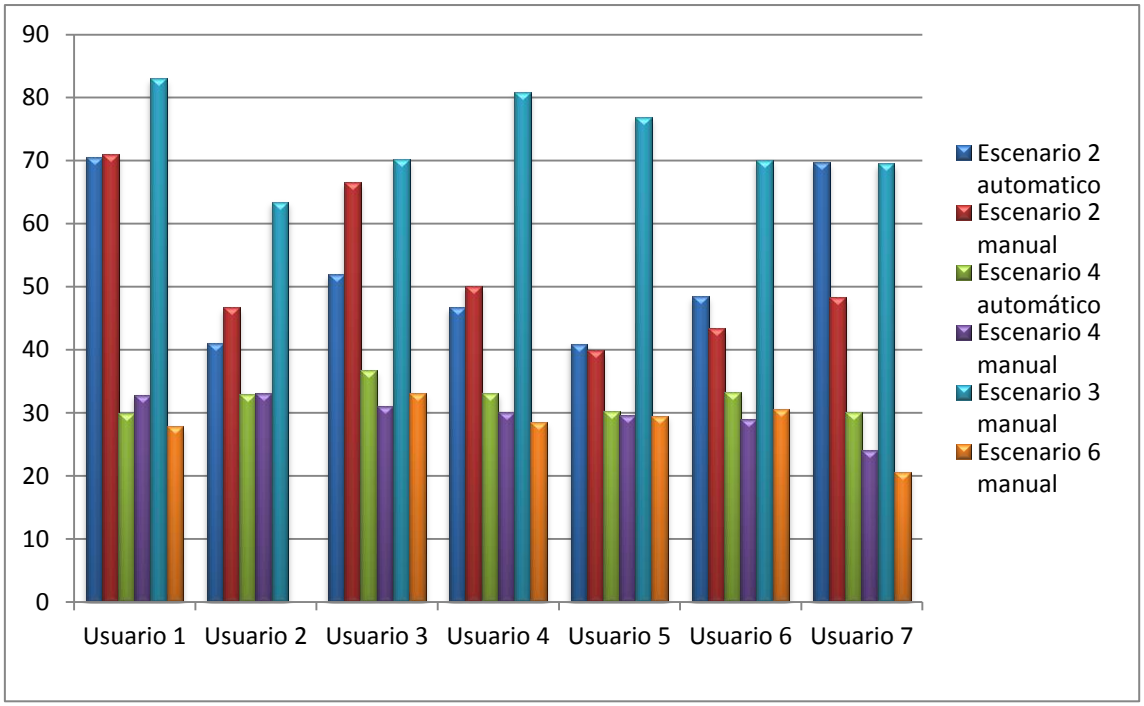

Figura 6.3. Velocidad media de los usuarios en cada escenario

Respecto a la velocidad, el usuario 1 es el que más velocidad ha llevado en los recorridos interurbanos, aunque a su vez ha llevado velocidades bajas en recorridos urbanos. Aun así su consumo no presenta grandes diferencias entre un comportamiento y otro, ya que tanto en escenarios urbanos como interurbanos ha mantenido un consumo alrededor de la media del resto de usuarios. Esto es debido a que influye mucho no sólo la velocidad, si no las revoluciones que se han llevado durante el recorrido, y en este caso se han mantenido en un punto medio.

A continuación, para medir el grado de dispersión de los datos de velocidad de cada usuario respecto a su valor promedio, en la tabla 6.5. se mostrarán las desviaciones típicas (s) en cuanto a la velocidad. Estas se han calculado mediante la fórmula:

$$s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}$$

Tenemos que x_i es el valor de la velocidad en cada instante, $\bar{x}$ es la media de las velocidades y n es el número de muestras. Para hacerlo se han extraído todos los valores de velocidad de cada uno de los ficheros de log creados.

	Escenario 2		Escenario 4		Escenario 3	Escenario 6
	Automático	Manual	Automático	Manual	Manual	Manual
Usuario 1	24,91	26,95	21,06	21,03	28,41	20,63
Usuario 2	14,80	21,64	21,20	29,64	23,19	22,07
Usuario 3	21,48	29,32	24,81	23,78	23,24	26,06
Usuario 4	17,61	21,56	24,06	23,42	26,38	22,58
Usuario 5	16,23	21,65	17,66	20,14	22,57	29,29
Usuario 6	19,51	17,75	22,45	26,39	21,08	28,13
Usuario 7	17,36	14,06	19,96	23,68	34,39	21,59

Tabla 6.5. Desviación típica de la velocidad de los usuarios

Aunque hay bastantes excepciones, por lo general los usuarios han tenido una mayor desviación típica en los casos en los que la conducción ha sido manual.

Además en la tabla 6.6 podemos ver las infracciones cometidas por cada uno de los usuarios en las diferentes escenas.

		Escenario 2		Escenario 4		Escenario 3	Escenario 6
		Automático	Manual	Automático	Manual	Manual	Manual
Usuario 1	Choques con coches	0	0	0	5	4	8
	Semáforos en rojo	0	0	0	2	0	0
Usuario 2	Choques con coches	0	0	0	4	4	8
	Semáforos en rojo	0	0	0	2	0	2
Usuario 3	Choques con coches	0	0	3	2	2	4
	Semáforos en rojo	0	0	0	0	0	1
Usuario 4	Choques con coches	0	0	1	2	1	7
	Semáforos en rojo	0	0	0	0	0	2
Usuario 5	Choques con coches	0	1	3	6	1	2
	Semáforos en rojo	0	0	1	1	0	1
Usuario 6	Choques con coches	0	0	1	2	1	4
	Semáforos en rojo	0	0	0	0	0	5
Usuario 7	Choques con coches	0	0	0	4	5	10
	Semáforos en rojo	0	0	0	1	0	4

Tabla 6.6. Infracciones cometidas por los usuarios

En cuanto a las infracciones cometidas por los usuarios, el usuario 7, que no tenía experiencia en videojuegos, es el que más tiene en total. A continuación estarían el usuario 1 y el 2, que aunque tienen algo de experiencia en videojuegos, son de los usuarios que menos años llevan con el carnet de conducir. Además, en el usuario 1 ha podido influir el hecho de ser el usuario en mayor estado de somnolencia a la hora de realizar la simulación.

Por último, después de que hubiesen hecho todas las pruebas, se ha hecho una encuesta a los usuarios acerca de diferentes factores del simulador. En la tabla 6.7. se muestra una media de las opiniones de los usuarios.

Facilidad de interacción (jugabilidad) (0-10)	Similitud a la conducción real (0-10)	Utilidad para formación de conductores (0-10)
8,5	7,5	7

Tabla 6.7. Opiniones de los usuarios acerca del simulador

También se ha preguntado a los usuarios sobre las diferencias entre que la vista sea desde dentro del coche o desde fuera al conducir, y lo que más se ha destacado es que la conducción desde dentro del coche es más real, aunque desde fuera resulta más fácil conducirlo.

7. Conclusiones y líneas futuras

En este apartado se tratarán las conclusiones obtenidas con la realización de este TFG, además de las posibles líneas futuras por las que se podría continuar trabajando en este proyecto.

7.1. Conclusiones

Tras realizar este Trabajo de Fin de Grado, se ha logrado crear un simulador para la conducción en el que se pueden aprovechar las ventajas que comentamos al inicio de poder utilizarlo para contribuir a la seguridad vial. De este modo, se ha logrado simular la conducción en diferentes escenarios y tipos de vías (urbanas e interurbanas), con diferentes cantidades de tráfico circulando por estas vías y diferentes comportamientos de los vehículos, de modo que nos podemos situar frente a situaciones muy diversas. Como se deseaba, los usuarios pueden ejecutar estas simulaciones sin someterse a los posibles peligros que habría en una carretera real, y pueden hacerlo tantas veces como deseen. Esto en cambio no sería posible utilizando un automóvil real en la carretera. Además los datos de las simulaciones son almacenados en ficheros, de los cuales podemos extraer información relevante para poder proporcionar una realimentación y mejorar en los aspectos que el usuario esté haciendo mal. Aquí se almacena información como la velocidad que lleva el usuario en cada momento, el consumo que tiene, la posición en la que se encuentra, la marcha, las revoluciones, etc. Además se calcula continuamente la velocidad y el consumo medios que ha llevado el usuario durante los recorridos. A través de estos datos de consumo, se puede mentalizar a los usuarios acerca de la conducción eficiente, ya que tras sus simulaciones podrán ver cuál es el consumo que han tenido a lo largo de la simulación, y se puede estudiar y probar los modos en los que obtendríamos un mejor rendimiento en el consumo de combustible. En este caso se pueden comparar los datos obtenidos utilizando un cambio de marchas manual con los obtenidos con un cambio de marchas automático y ver si se logra un consumo tan bajo en modo manual.

De este modo el simulador puede ser utilizado para el aprendizaje tanto de los controles de un automóvil mediante los pedales, el volante y la palanca de cambios utilizada de *Logitech G27*, como para adquirir conocimientos sobre cómo reaccionar ante diferentes situaciones que se nos puedan plantear en la carretera como cruces, pasos de peatones, rotondas con otros coches, intersecciones con semáforos, etc. Todo esto de un modo seguro, y con una realimentación que permitirá mejorar tanto en el control como en la eficiencia de la conducción.

Mediante la aportación de las pruebas realizadas con los usuarios, vemos que este simulador presenta una buena facilidad de interacción, pero sería mejor poder adaptarlo para que su similitud con la conducción real fuese más elevada y así poder utilizarlo de un modo más eficiente para la formación de conductores. Además el modo de visión del coche a conducir será más eficiente teniendo la vista desde dentro del automóvil para conseguir un mayor realismo.

7.2. Líneas futuras

Como líneas futuras de este trabajo, se podría hacer una mejora del simulador, aumentando el realismo en la conducción, tanto por parte del automóvil que lleva el usuario como de los coches con inteligencia artificial que circulan por el escenario.

En cuanto a los coches con inteligencia artificial, se podrían mejorar los detectores que utilizan y las condiciones de actuación en función de lo que detecten, de modo que puedan actuar en situaciones más complejas de la manera que lo haría cualquier usuario.

Además se podrían crear nuevos escenarios con otros elementos de las carreteras como podrían ser túneles, tramos con otro tipo de señales que deban ser reconocidas para actuar en consecuencia de su significado, y generar situaciones en las que habría un alto riesgo de accidente, ya sea por la peligrosidad del tramo de la vía o por el mal comportamiento de algún otro vehículo.

Otra opción posible sería crear un escenario en el que el usuario deba aparcar, tanto en batería como en línea.

Respecto al almacenamiento de datos se podría añadir más información como tiempos de reacción ante situaciones peligrosas preparadas en el simulador, u otros aspectos sobre el uso que da el usuario a los distintos controles.

Por último, con el simulador se podrían comprobar los efectos que tiene sobre la conducción el estado en el que se encuentre el usuario, como en situaciones de embriaguez, somnolencia, haber consumido algún tipo de sustancia que pueda afectar al estado del conductor, como podría ser algún tipo de medicina.

8. Presupuesto económico

Para la realización de este TFG, como se ha mencionado anteriormente, se ha trabajado con el periférico Logitech G27 Racing Wheel, además del ordenador utilizado y un *plugin* de *Unity* llamado *Road & Traffic System*. Para el desarrollo del proyecto ha sido necesario utilizar los programas Unity (que se ha utilizado en su versión Pro) y 3ds Max. Teniendo en cuenta sus precios, el gasto en hardware y software sería de:

▪ Logitech G27 Racing Wheel:	359,00 €
▪ Licencia Unity Pro (6 meses):	404.89 €
▪ Licencia 3ds Max (6 meses):	1200,00 €
▪ Plugin Unity Road & Traffic System:	22,50 €
▪ Ordenador portátil Asus X550L:	700,00 €
Total:	2686,39 €

En cuanto a las horas dedicadas al desarrollo de este trabajo, se calculan unas 185 horas. Teniendo en cuenta que el sueldo de un desarrollador de videojuegos amateur puede ser sobre 22€/hora, se calcularían unos 4070€.

El gasto total ascendería a 2686,39 + 4070 = 6756,39 €.

9. Bibliografía

Airsoft (2015). Simuladores de conducción. Para el aprendizaje y práctica de la conducción de vehículos de dos y cuatro ruedas. Disponible en: *https://www.dropbox.com/s/5awfovq32c0s5s1/ARISOFT_PresentacionSIMULADORES.pdf*

American Psycological Association (APA). Artificial Intelligence (n.d.). Dictionay.com Unanbridged. Disponible en: *http://dictionary.reference.com/browse/artificial intelligence*

Autodesk (2015). 3ds Max. Disponible en: *http://www.autodesk.es/products/3ds-max/overview*

Autodesk (2015). Software de modelado y animación en 3D. Maya. Disponible en: *http://www.autodesk.es/products/maya/overview*

Autoexpress (2015). Cómo ir más lejos con menos. Disponible en: *http://autoexpress.es/conduccion-eficiente-como-llegar-mas-lejos-con-menos-3-parte-durante-la-conduccion*

Blender (2015). Home of the Blender proyect. Disponible en: *http://www.blender.org/*

Cebolla Cebolla, Castell (2012). 3D Studio Max 2012: curso práctico. Ed. Ra-Ma.

Cecu (2015). La conducción eficiente. Disponible en: *http://www.cecu.es/especiales/conduccion/secciones/conduccion_eficiente.html*

Crytek (2015). CryEngine Features. Disponible en: *http://cryengine.com/features*

D.G.T. (2013). Las principales cifras de la siniestralidad vial. España 2013. Disponible en: *http://www.dgt.es/Galerias/seguridad-vial/estadisticas-e-indicadores/publicaciones/principales-cifras-siniestralidad/Siniestralidad_Vial_2013.pdf*

D.G.T. (2014). Seguridad vial. Cuestiones de seguridad vial, conducción eficiente, medioambiente y contaminación. Disponible en: *http://www.dgt.es/Galerias/seguridad-vial/formacion-vial/cursos-para-profesores-y-directores-de-autoescuelas/XVII-Curso-de-Profesores/Seguridad-vial-Ed.-2014.pdf*

DriveSim (2015). Driving Software Simulator. Disponible en: *http://drivesimsimulator.com/es/*

Drive Square (2015). Driving Simulators for Training, Drunk Simulator. Disponible en: *http://www.drivesquare.com/de/*

Epic Games (2015). What is Unreal Engine 4. Unreal Engine Features. Disponible en: *https://www.unrealengine.com/unreal-engine-4*

Fisher, D.L., Rizzo, M., Caird, J., Lee, J.D. (2011). Handbook of driving simulation for engineering, medicine and psychology.

Forward Development (2015). City Car Driving. Disponible en: *http://citycardriving.com*

Garage Games (2015). Torque 3D. Disponible en: *http://www.garagegames.com/products/torque-3d/overview/overview*

Geig, Mike (2014). Unity Game development in 24 hours. Ed. Sans.

Jeep (2015). Gear Ratio Calculator. Disponible en: *http://www.grimmjeeper.com/gears.html*

Johns, M.W. (2006). A new method for measuring daytime sleepiness: The Epworth sleepiness scale (ESS). Try this: Best Practises in Nursing Care to Older Adults. Disponible en: *http://consultgerirn.org/uploads/File/trythis/try_this_6_2.pdf*

Johns, M.W. & Hocking, B. (2009). What is excessive daytime sleepiness. Sleep Deprivation: Causes, Effects and Treatment, Nova Science Publishers. Disponible en: *http://www.mwjohns.com/wp-content/uploads/2008/09/johns-2009-what-is-eds.pdf*

Logitech Support (2015). Logitech Gaming Software. Disponible en: *http://support.logitech.com/software/gaming-software*

Maxon (2015). Cinema 4D Studio. Disponible en: *http://www.maxon.net/es/products/cinema-4d-studio.html*

Monster, Marco (2003). Car Physics for Games. Disponible en: *http://www.asawicki.info/Mirror/Car%20Physics%20for%20Games/Car%20Physics%20for%20Games.html*

Microsoft (2015). MSDN. Learn to Develop with Microsoft Developer Network. Disponible en: *https://msdn.microsoft.com/*

Ray Johnston, Chevy (2013). Unity Patterns. Disponible en: *http://unitypatterns.com/*

Shiva Technologies SAS (2014). Shiva 1.9. Cross Platform Game Development Tool. Disponible en: *http://www.shivaengine.com/*

STISIM Drive (2015). Car Driving Simulator & Simulation Software. Disponible en: *http://www.stisimdrive.com/*

TF3DM (2015). 3D Models for free. Disponible en: *http://tf3dm.com/*

Torque 3D (2015). Torque 3D. Disponible en: *http://torque3d.org/*

TurboSquid (2015). 3D Models for Professionals. Disponible en: *ww.turbosquid.com*

Unity (2015). Unity. Disponible en: *http://unity3d.com/*

Unity (2015). Unity Asset Store. Disponible en: *assetstore.unity3d.com/*

Unity (2015). Unity Community. Disponible en: *http://forum.unity3d.com/*

Unity Technologies (2015). Unity Manual. Disponible en: *http://docs.unity3d.com/Manual/index.html*

Valve (2012). Source Engine Features. Valve Developer Community. Disponible en: *https://developer.valvesoftware.com/wiki/Source_Engine_Features*

Vlaked, W. P. (2005). The use of simulators in basic driver training. In Humanist TFG Workshop on the Application of New Technologies to Driver Training, Brno, Czech Republic. Disponible en: *www. escope. info/download/research_and_development/HUMANISTA_13Use. pdf*

Anexo I. Plugin Road & Traffic system.

Como se ha comentado con anterioridad, para el desarrollo de los escenarios 4 y 6, se ha utilizado como herramienta un *plugin* de *Unity* llamado *Road & Traffic system*.

Este *plugin* generaba una serie de errores al hacer *debug* en *Monodevelop* (entorno de desarrollo utilizado por *Unity*), a pesar de que en el propio Editor no los daba. Estos errores eran debidos al modo en el que se hace la sobrecarga de métodos, ya que en C# no se permiten especificadores de parámetros predeterminados en la definición de una función, es decir, no se puede asignar valores por defecto a una variable de entrada para que si al llamar a la función no se le ha asignado un valor, tome dicho valor por defecto.

Un ejemplo que generaría errores sería:

```
public TrafficSystemNode GetNextNode( TrafficSystemVehicle a_vehicle, bool a_checkLocalConnectedNode = true, List<TrafficSystemNode> a_blockedNodes = null )
{
        …
}
```

En este ejemplo no se podrían asignar ahí los valores *true* del segundo parámetro y *null* del tercero. Para solucionar el problema se ha tenido que cambiar por:

```
public TrafficSystemNode GetNextNode( TrafficSystemVehicle a_vehicle, bool a_checkLocalConnectedNode)
{
        return GetNextNode(a_vehicle,a_checkLocalConnectedNode,null);
}

public TrafficSystemNode GetNextNode( TrafficSystemVehicle a_vehicle)
{
        return GetNextNode(a_vehicle,true,null);
}

public TrafficSystemNode GetNextNode( TrafficSystemVehicle a_vehicle, bool a_checkLocalConnectedNode, List<TrafficSystemNode> a_blockedNodes )
{
        …
}
```

De este modo si se dan dos parámetros en la llamada de la función, se devolverá una llamada a la función en la que el tercer parámetro será puesto a *null* por defecto, y si

solo se especifica un parámetro en la llamada el segundo será puesto a *true* y el tercero a *null*. No ha sido necesario hacer el caso para cuando no se especifica ningún parámetro, ya que no hay ninguna llamada a la función en los scripts.