UNIVERSIDAD DE VALLADOLID

E.T.S.I. TELECOMUNICACIÓN

TRABAJO FIN DE MÁSTER

MÁSTER EN INGENIERIA DE TELECOMUNICACIÓN

Escenarios de simulación de conducción basados en Unity con vehículos de combustión y eléctricos.

Autor:
D. Eduardo Loya Rodríguez

Tutor:
D. David González Ortega

Valladolid, 21 de Febrero de 2018

TÍTULO:	**Escenarios de simulación de conducción basados en Unity con vehículos de combustión y eléctricos**
AUTOR:	**Eduardo Loya Rodríguez**
TUTOR:	**D. David González Ortega**
DEPARTAMENTO:	**Departamento de Teoría de la Señal y Comunicaciones e Ingeniería Telemática**

TRIBUNAL

PRESIDENTE:	**Dña. María Ángeles Pérez Juárez**
VOCAL:	**Dña. Míriam Antón Rodríguez**
SECRETARIO:	**D. Javier Manuel Aguiar Pérez**
SUPLENTE:	**D. Miguel López Coronado**
SUPLENTE:	**D. Jaime Gómez Gil**
SUPLENTE:	**Dña. Isabel de la Torre Díez**

FECHA:

CALIFICACIÓN:

Resumen

A lo largo de los últimos años se ha podido presenciar la continua evolución del mundo de los videojuegos, la cual se ha producido no sólo a nivel gráfico o de jugabilidad, sino que también en cuanto a su finalidad, abriéndose un nuevo campo alejado de la búsqueda de la pura diversión y que busca ir más allá, permitiendo que se pueda pensar en ellos como herramienta para adquirir conocimiento o para prepararnos ante situaciones de la vida real. Es precisamente en este nuevo campo donde se sitúa el simulador de conducción que se lleva a cabo en este proyecto, desarrollado con la plataforma Unity 3D, con el que se podrán poner en práctica las habilidades de los usuarios en la conducción de una manera didáctica a través de una ruta real que recorre la ciudad de Valladolid.

Abstract

Over the last few years it has been possible to witness the continuous evolution of the world of videogames, which has occurred not only graphically or in terms of gameplay, but also in terms of its purpose, opening a new field away from the search for pure fun and that seeks to go further, allowing you to think of them as a tool to acquire knowledge or to prepare for real life situations. It is in this new field where the driving simulator is located that is carried out in this project, developed with the Unity 3D platform, which will allow to put into practice the driving skills of the users in a didactic way through a real route that runs through the city of Valladolid.

Palabras clave

Simulador, Escenario, Motor gráfico, *Unity*, *Asset*, Conducción, Eficiencia, Infracciones.

Agradecimientos

A mi familia y en especial a mis padres por haberme brindado la oportunidad de llevar a cabo mis estudios en esta área y siempre con las mejores palabras y el máximo apoyo.

A todos los amigos y compañeros que he tenido no sólo durante el máster sino desde que comenzara la carrera, por su apoyo y ayuda en todo tipo de situaciones, y en especial a aquellos que han sacado tiempo para colaborar como usuarios en las pruebas del simulador.

A David, mi tutor de este trabajo de fin de máster, por el apoyo y la confianza para realizar este trabajo; y a mis compañeros de laboratorio Julián y Javier por la ayuda que me han aportado con Unity.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

En el presente apartado se lleva a cabo una presentación del trabajo realizado en este proyecto de forma que el lector conozca las motivaciones que llevaron a su realización y los objetivos que se pretendían cumplir cuando este fue iniciado meses atrás. Del mismo modo, se describirán las fases y métodos que se van a seguir para su realización, además de los medios que se han utilizado para tal propósito y la metodología empleada para poder cumplir con la previsión temporal inicialmente marcada.

Por último, se mencionará la estructura que seguirá el resto del documento, realizando una breve introducción al contenido de los siguientes apartados.

1.1. Motivación

A lo largo de los últimos años hemos podido presenciar la continua evolución del mundo de los videojuegos, y esta se ha producido, no solo a nivel gráfico o de jugabilidad, sino que también ha ido mutando su finalidad, abriéndose un nuevo campo alejado de la búsqueda de la pura diversión y que busca ir más allá, permitiendo que se pueda pensar en ellos como herramienta para adquirir conocimiento o para prepararnos ante situaciones de la vida real. Es precisamente en este nuevo campo donde se sitúa el simulador de conducción que se llevará a cabo a lo largo de este proyecto.

Los primeros simuladores en surgir datan de mitad del siglo XX [1], empleados por aquel entonces para ayudar a los pilotos a mejorar sus habilidades. A día de hoy, múltiples empresas están invirtiendo enormes cantidades de dinero en estos simuladores.

La aparición de este tipo de simuladores representa un gran apoyo en múltiples situaciones, y tiene una serie de importantes ventajas con respecto a la utilización del uso de vehículos reales, como pueden ser:

- Ayudar a ganar confianza a personas sin experiencia en conducción y a las que realizarlo en la carretera les suponga un miedo.
- El hecho de poder simular cualquier situación que se nos pueda ocurrir y que podría resultar complicada de realizar en la realidad (simulación de condiciones de lluvia en zonas donde no es frecuente, simulación de entornos de montaña en zonas donde no hay, etc.).
- La responsabilidad con el medio ambiente en cuanto a que se evita el impacto que supone el uso de vehículos reales y las emisiones que conllevan.
- Poder realizar pruebas de todo tipo donde la seguridad de las personas no corre ningún riesgo.
- El ahorro que supone tanto a nivel de gasolina como de personas que puedan hacer falta involucrar o daños materiales que puedan sufrir los vehículos.

Por todas estas razones, se entiende que el trabajo que se ha llevado a cabo a lo largo de este proyecto tiene una aplicación real, algo que desde el punto de vista personal lo dota de gran valor.

También han sido importantes incentivos para la elección y realización de este proyecto el poder aprender y poner en práctica el desarrollo de videojuegos básicos tanto para

ordenadores como para dispositivos móviles, mundo en el que ya me embarqué cuando realicé un videojuego para el móvil en la realización del trabajo de fin de grado y que continúa captando mi atención, esta vez a través del aprendizaje de una nueva herramienta como es *Unity 3D*, basada en el lenguaje de programación C#.

1.2. Objetivos

El principal propósito de este proyecto será la realización de un escenario de conducción con ayuda de la herramienta *Unity 3D*, donde el usuario podrá realizar un recorrido que simula parte de la ciudad de Valladolid combinando tramos urbanos y de autovía, y en donde podrá poner a prueba sus habilidades de conducción, ya que dispondremos de un volante para poder realizar tales simulaciones. Se pretende analizar de esta manera la eficiencia y las estadísticas de las simulaciones realizadas por los usuarios. Se busca concienciar al usuario acerca de la importancia de las normas de tráfico a la par que se realiza una estimación sobre el consumo que el vehículo realiza, de forma que se pueda estudiar cómo afectan distintos factores como puedan ser el tráfico, condiciones meteorológicas o distintas configuraciones del propio vehículo.

Para ello se plantea el objetivo de crear una interfaz de usuario donde este realice las simulaciones, tratando de hacer la experiencia lo más real posible, no sólo a través del vehículo que el usuario utilizará sino también a través de distintas animaciones en el escenario, implementación de audio, dotación de inteligencia artificial para el resto de vehículos y peatones de la simulación, y realización de diversos *scripts* que permitan un control real de la eficiencia tanto del conductor como del vehículo.

Al finalizar la simulación todos los datos recopilados como el tiempo, infracciones realizadas, consumo y posición en cada instante serán almacenados para su análisis posterior en un fichero XML.

1.3. Estructura, fases y métodos.

Para llevar a cabo todo el trabajo que conlleva la realización de este proyecto, se realizó de manera previa un breve estudio sobre la seguridad vial, tema que ocupará el siguiente capítulo. Posteriormente se realizó una fase de familiarización con la herramienta Unity a través de diversos tutoriales que ofrece la propia página oficial y que nada en principio tienen que ver con el simulador a realizar, conociendo así las diversas posibilidades que dicha herramienta ofrece y sobre las que se habla en el tercer capítulo.

Una vez adquirido el conocimiento básico sobre la herramienta, se continuó el proceso de aprendizaje, ya enfocado al propósito del proyecto a la vez que se fue realizando este, tema del que se hablará en el cuarto capítulo y que abarcará la mayor parte del trabajo realizado.

Finalizadas las distintas pruebas y ajustes sobre el escenario, se plasma también en el cuarto capítulo todo lo necesario para poder realizar la simulación sin necesidad de tener que conocer todo el trabajo previo.

Posteriormente se hace un breve análisis en el sexto capítulo del presupuesto económico básico del trabajo desempeñado en él.

Por último, se presentan las conclusiones extraídas tras todo lo anteriormente expuesto, además de mostrar distintas líneas de desarrollo futuras cuya realización con el tiempo adecuado podría dotar de mayor valor al trabajo ya realizado.

2. Seguridad vial, simuladores y conducción eficiente

En este apartado se hablará sobre el concepto de seguridad vial, se tratarán diversas cuestiones acerca de los accidentes de tráfico y los factores que intervienen en la conducción, se introducirá el concepto de conducción eficiente, presentando las distintas formas de conseguirla y sus ventajas, y se analizará la contribución que hasta el momento han aportado los simuladores de conducción.

2.1. Seguridad vial

Nos encontramos comenzando el año 2018, o lo que es lo mismo, estamos en la recta final de esta década, la cual fue designada tiempo atrás por las Naciones Unidas como la Década de la Acción para la Seguridad Vial [2]. Con este plan, se retaba a los países que lo acogiesen a disminuir en un 50% el número de fallecidos. A continuación, y sirviéndonos de los datos oficiales en España recogidos hasta la entrada de 2017 [3] [4], veremos en qué medida se está cumpliendo y cuál es la situación actual.

2.1.1. Los accidentes de tráfico

Podemos definir este tipo de accidentes como "cualquier evento como resultado del cual el vehículo queda de manera anormal, dentro o fuera de la carretera, o que produce lesiones en las personas y daños a terceros". La Organización Mundial de la Salud (OMS) estima que cada día alrededor de 3500 personas fallecen en las carreteras, además de las decenas de millones de personas que sufren heridas o discapacidades cada año [5].

En 2016, se ha producido un aumento del 7% en los fallecidos, que no ha sido homogéneo para todas las vías ni los usuarios de las mismas, observándose un mayor aumento en las vías urbanas y afectando de manera especial a colectivos vulnerables: peatones, ciclistas y motoristas.

No obstante, seguimos siendo uno de los países de la Unión Europea con las tasas más bajas de fallecidos por millón de habitantes en accidente de tráfico, ocupando la quinta posición en el *ranking* de países con las cifras más bajas de siniestralidad, con cifras mejores que países como Alemania, Francia, Italia o Finlandia.

Además, en nuestra siniestralidad juegan un papel importante el aumento de la movilidad, el envejecimiento del conjunto de automóviles, la velocidad inadecuada, la conducción bajo los efectos del consumo de alcohol y otras drogas, la no utilización del cinturón de seguridad y de los sistemas de retención infantil y la distracción, especialmente la relacionada con el uso de dispositivos móviles.

Durante el año 2016, las diferentes policías notificaron 102.362 accidentes con víctimas. Estos accidentes ocasionaron 1.810 fallecidos en el momento del accidente o hasta 30 días después del mismo, 9.755 personas fueron ingresadas en un centro hospitalario y 130.635 resultaron heridas no hospitalizadas, según fuentes policiales.

En el año 2016 han fallecido 121 personas más que en el año anterior, lo que supone un incremento del 7% en relación al año 2015. Es importante señalar que las variaciones no han sido homogéneas, han aumentado los fallecidos pasajeros un 17%, un 6% los peatones y un 5%

los conductores. Así mismo se observan aumentos en la mayoría de los usuarios, los fallecidos en turismos aumentaron un 9% y también aumentaron un 4% los usuarios de motocicletas. Este año han fallecido 19 personas más en autobús y 9 en bicicleta al comparar con el año anterior. Por edades, aumentaron en la mayoría de los grupos de edad, observándose los mayores aumentos en el grupo de edad de 45 a 54 años que aumentó un 19% los fallecidos, el grupo de 15 a 24 que aumentó un 16%, el grupo de 55 a 64 que aumentó un 10% y el grupo de 35 a 44 que aumentó un 7%. Los mayores descensos en los fallecidos se observaron en el grupo de edad de 75 a 84 años (un 8%) y en el grupo de edad de 25 a 34 años que disminuyó un 6%.

Respecto del análisis de los factores que inciden en la seguridad, se ha realizado un análisis general de los factores concurrentes en vías interurbanas, utilizando para ello una muestra de accidentes con víctimas informados por las policías. Se observa que los factores más citados en los informes policiales son la distracción (25,0%), la velocidad inadecuada (19,2%), no mantener intervalo de seguridad (15,0%), no respetar las normas de prioridad (12,9%) y el consumo de alcohol (10,4%).

Si analizamos por otra parte cómo va variando el parque de automóviles, en la figura 1 observamos que más de las dos terceras partes son turismos. El parque de automóviles ha crecido en más de un millón de unidades en el último decenio, contando todas las categorías de vehículos. En 2016 se observa un crecimiento en el total del parque de automóviles respecto del año anterior. La mayor subida en cifras absolutas se produce en los turismos, con un saldo positivo de 521.281 vehículos, lo que supone en términos porcentuales un aumento del 2%. En términos absolutos sigue en orden la categoría de motocicletas, con 132.011 unidades más, lo que equivale a un 4% de subida interanual. Con saldo negativo destaca la categoría de ciclomotores, con un descenso del 2%. El parque de automóviles está compuesto mayoritariamente por turismos, que superan los 22 millones de unidades, lo que supone un 68% del parque de automóviles; siguen los camiones y furgonetas, con un 15% del parque total, y las motocicletas, con un 10%.

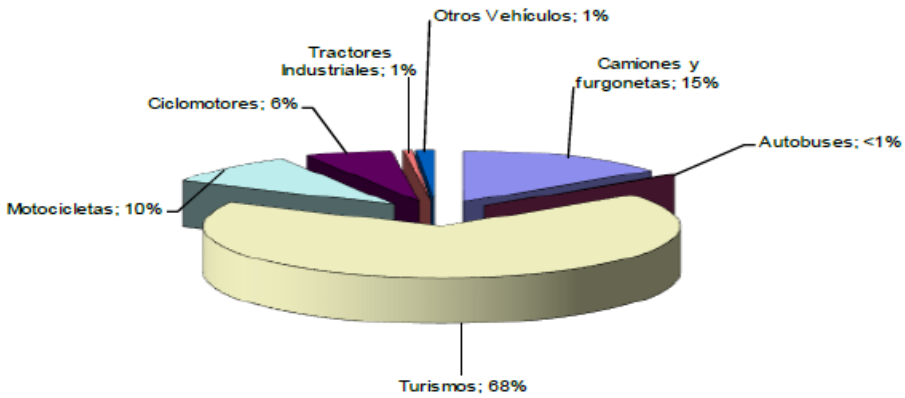

Figura 1. Distribución porcentual del parque automóvil. España, 2016

Respecto a la comparación internacional, en el año 2016, con cifras provisionales de la UE, España ocupó la quinta posición en el ranking de tasas de víctimas mortales con un valor de 39 fallecidos por millón de habitantes, por debajo de la tasa europea que fue de 51 fallecidos por millón de habitantes y dos puntos por encima de la tasa objetivo en relación a los fallecidos de la Estrategia de Seguridad Vial 2011-2020.

Los objetivos de mejora de la seguridad vial en nuestro país se plasmaron en la "Estrategia de Seguridad Vial 2011- 2020", aprobada por Consejo de Ministros de 25 de febrero de 2011, que incluye la concreción de 13 retos. La tabla 1 muestra los correspondientes trece indicadores a mejorar en 2020 relacionados con el número de accidentes, su valor en el año base, 2009, su valor en el año final, 2020, en el año de referencia de la publicación consultada, 2016, y en el año inmediatamente anterior, 2015 [6].

Algunos de estos indicadores fijados para 2020 ya se han conseguido, como es el caso de 25% menos de conductores de 18 a 24 años fallecidos y heridos graves en fin de semana.

Indicadores	Cifra basal 2009	Cifra 2015	Cifra 2016	Cifra objetivo 2020
1. Bajar de la tasa de fallecidos de 37 por millón de habitantes	59	36	39	Inferior a 37
2. Reducción del número de heridos graves en un 35%	13.923	9.495	9.755	9.050
3. Cero niños fallecidos sin sistema de retención infantil	12	5	3	0
4. 25% menos de conductores de 18 a 24 años fallecidos y heridos graves en fin de semana	730	353	361	548
5. 10% menos de conductores fallecidos mayores de 64 años	203	200	206	183
6. 30% de reducción de fallecidos por atropello	459	306	386	321
7. 1 millón de ciclistas más sin que se incremente su tasa de mortalidad	1,2	1,2	1,4	1,2
8. Cero fallecidos en turismos en zona urbana	101	61	80	0
9. 20% menos de fallecidos y heridos graves usuarios de motocicleta	3.473	2.928	3.024	2.778
10. 30% menos de fallecidos por salida de vía en carretera convencional	520	285	270	364
11. 30% menos de fallecidos en itinerarios urbanos	170	101	No disponible	119
12. Bajar del 1% los positivos en aire espirado en los controles preventivos aleatorios. Punto de corte 0,05 mg/l	6,7%	1,7%	No disponible	Inferior al 1%
13. Reducir en 50% el porcentaje de vehículos ligeros que superan el límite de velocidad en más de 20 km/hora	12,3% (autopista) 6,9% (autovía) 15,8% (conv.90)* 16,4% (conv100)**	No disponible. Estudio periódico	No disponible. Estudio periódico	6,2% (autopista) 3,5% (autovía) 7,9% (conv.90) 8,2% (conv.100)
2 En los indicadores 2, 4 y 9, se entiende por herido grave toda persona herida en un accidente de circulación cuyo estado precisa una hospitalización superior a veinticuatro horas.				
3 Niños de menos de 12 años.				
* Conv.90 hace referencia a las carreteras convencionales con límite de velocidad de 90 kmh				
** Conv100 hace referencia a las carreteras convencionales con límite de velocidad de 100 kmh				

Tabla 1. Indicadores de mejora de la conducción para el 2020 en España.

La evolución de las cifras de fallecidos por accidente de tráfico con víctimas, desde que se mantienen estadísticas, muestra a partir del año 1960 una tendencia general ascendente hasta alcanzar un máximo en el año 1989, en el que se notificaron 9.344 fallecidos. Desde entonces el número de fallecidos ha ido disminuyendo de manera más o menos acusada hasta alcanzar el mínimo de la serie histórica en el año 2013, con 1.680 fallecidos. En el último año documentado en el informe consultado, 2016, el número de fallecidos ha sido 1.810, lo que ha supuesto un aumento del 7% respecto del año 2015. Podemos ver esta evolución plasmada en forma de gráfico en la figura 2.

Figura 2. Evolución de los fallecidos en accidente de tráfico con víctimas en España.

Para mayor conocimiento sobre el resto de datos que comprenden todo lo relacionado con los accidentes de tráfico y sus cifras, se recomienda consultar la fuente ya mencionada de donde estos han sido extraídos. Aquí hemos podido ver y analizar una pequeña parte de todos esos datos, los cuales nos deben hacer entender que aún queda un gran trabajo por delante, y que cualquier herramienta o ayuda, como pueda ser la de un simulador, será necesaria y bien recibida.

2.1.2. Factores que intervienen en la circulación

Entendemos por factor de riesgo todo aquel elemento, fenómeno, condición, circunstancia o acción humana que incrementa la probabilidad de ocurrencia de un accidente. Estos factores suelen englobarse en los tres elementos generales implicados en toda situación de tráfico: el vehículo, la vía y su entorno, y el propio conductor [7].

Sin embargo, no todos estos factores tienen la misma importancia en la causa de los accidentes, puesto que a pesar de los fallos técnicos del vehículo y los derivados de factores ambientales o la vía, el factor humano es el responsable de hasta el 90% de los accidentes de tráfico.

Factor humano: Los factores humanos son la causa del mayor porcentaje de accidentes de tráfico. Dependiendo de la situación, podemos desenvolvernos como peatón, conductor, pasajero… estando todos estos roles incluidos en el factor humano.

Un factor importante relacionado con la coexistencia con el resto de personas que conforman el tráfico es la confianza. Al salir a la carretera muchas veces se asume que las personas a nuestro alrededor tienen la intención de cumplir las normas establecidas en mayor o menor medida. El problema ocurre cuando ya sea por exceso de confianza o negligencia al volante, es el factor humano el responsable de un accidente.

A continuación se enumeran algunos ejemplos de causas de accidentes debidas a este ya mencionado factor:

- Conducir bajo los efectos del alcohol, medicinas y estupefacientes es una de las mayores causas de accidentes de tráfico según la OMS [8].
- Condiciones físicas: la fatiga, somnolencia, realización de trayectos sin descanso, comidas abundantes…
- Condiciones psíquicas: depresión, actitud temeraria e imprudente, agresividad, negligencia…
- Desobedecer las señales de tráfico o su desconocimiento.
- Tiempo de percepción: Corresponde a la etapa de detención. El tiempo desde que el conductor percibe un peligro hasta que el cerebro procesa una respuesta.
- Tiempo de reacción: el tiempo entre la recepción del estímulo y la ejecución de la acción. Este tiempo de reacción se compone a su vez de unos determinados factores para su cálculo, como son: evaluación, decisión, distancia de reacción, tiempo de reacción mecánica, distancia de frenado y distancia total o distancia de detección.

Factor mecánico: hace alusión a todo lo relacionado con el vehículo, incluyendo situaciones como que este sufra una avería o que no responda adecuadamente. Mencionar que, a pesar de considerarse un factor independiente del humano, es responsabilidad del conductor el correcto funcionamiento, reglaje y puesta a punto del vehículo para minimizar el riesgo de sufrir un contratiempo que derive en un accidente vial, así como conocer el perfecto funcionamiento del mismo y la utilización de sus componentes para las diferentes maniobras. Es por ello obligatorio por ley que el vehículo supere las inspecciones técnicas pertinentes y revisiones por parte de la marca, garantizándose así el correcto funcionamiento del vehículo para que este no suponga un riesgo potencial para la conducción, así como disponer de un seguro que se haga responsable de los perjuicios ocasionados si es necesario.

Se pueden destacar dos tipos de seguridades relacionadas con este factor:

- La seguridad activa: Los elementos o sistemas que contribuyen a la seguridad activa del vehículo son aquellos que le confieren un correcto comportamiento en marcha, siendo los principales: neumáticos, dirección, suspensión, frenos, alumbrado y limpiaparabrisas.
- La seguridad pasiva: hace referencia a aquellos elementos diseñados para evitar o disminuir los posibles daños ocasionados a los ocupantes como consecuencia de un choque: carrocería, cinturón de seguridad, *airbag*, casco y reposacabezas.

Factor ambiental, la vía y su entorno: por último, considerar lo relacionado con el escenario por el cual el conductor y su vehículo han de desenvolverse y todo lo relacionado con él. Será el Estado el encargado de preservar el correcto mantenimiento de las vías públicas para que no supongan un impedimento a la hora de realizar una conducción adecuada [9]. Las vías especiales han de tener una clara señalización que no dé lugar a dudas y las situaciones

peligrosas como cambios de rasante, posible presencia de animales o tramos de alta concentración de accidentes han de ser señalizados explícitamente. Estos factores representan las exigencias a las que el conjunto conductor-vehículo debe responder y están configurados por los aspectos o elementos ambientales inalterables: la calzada o vía y el diseño de su entorno y, por otra parte, todo un conjunto de condiciones circundantes de naturaleza cambiante. Entre estos llamados elementos estables del sistema podríamos diferenciar entre la calzada o vía, y la climatología e incidencias u obstrucciones temporales: oscuridad, niebla, lluvia, nieve o hielo, obras en la vía, cruce de animales, otros vehículos y peatones, atascos, retenciones, etc. [10].

2.2. Conducción eficiente

La conducción eficiente es un nuevo modo de conducir que tiene como objetivo lograr un bajo consumo de carburante, una reducción de la contaminación ambiental, un mayor confort de conducción y una disminución de riesgos en carretera [11].

Respecto a los modos convencionales de conducción, este tipo de conducción se rige por una serie de reglas sencillas y eficaces que tratan de aprovechar las posibilidades que ofrecen las tecnologías de los motores de los coches actuales.

En España, en el sector del transporte se quema más del 60% de todo el petróleo consumido en nuestro país. De la totalidad de la energía consumida en dicho sector, el tráfico rodado consume cerca de un 80%. El vehículo automóvil consume un 15% de la energía total consumida en nuestro país. El 40% de las emisiones totales de CO_2 originadas por el consumo de energía proviene del transporte por carretera [12].

De la relevancia de estas cifras surge la necesidad de plantearse la utilización del vehículo automóvil de una forma más eficiente y racional. A lo largo de los últimos 20 años, el consumo de carburante de los coches nuevos ha ido disminuyendo progresivamente por la implantación de nuevas tecnologías, pero esto no es suficiente. La actitud del conductor y su estilo de conducción son también decisivos a la hora de reducir el consumo global de carburantes.

En Europa existe un firme propósito de desarrollar otras fuentes de energía que emitan mínimas cantidades de CO_2 a escala global. Estas son las energías renovables en general (los biocombustibles en particular), y otras energías alternativas al petróleo. Sin embargo, hoy día la capacidad de sustituir significativamente a los derivados del petróleo no es probable a corto o medio plazo. Surge entonces la necesidad de implantar nuevos programas de reducción del consumo de carburante en los coches.

2.2.1. Ventajas de una conducción eficiente

Las principales ventajas del nuevo estilo de "conducción eficiente" se pueden clasificar entre las que son para el propio conductor o a nivel global. Para el propio conductor las ventajas son: mejora del confort de conducción y disminución de la tensión, reducción del riesgo y gravedad de los accidentes, ahorro económico de combustible, menores costes de mantenimiento (frenos, embrague, caja de cambios, neumáticos y motor). Globalmente estas ventajas son: reducción de contaminación urbana que mejora la calidad del aire respirado, reducción de emisiones de CO_2 y con ello mejora de los problemas del calentamiento de la atmósfera ayudando a que se cumplan los acuerdos internacionales en esta materia, ahorro de energía a

escala nacional que incide en la balanza de pagos y reducción de dependencia energética exterior.

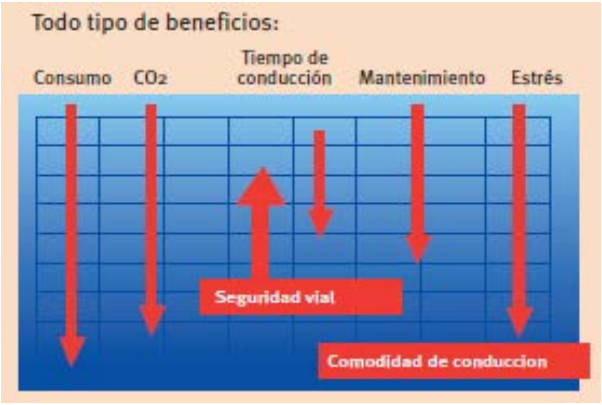

Figura 3. Beneficios de la conducción eficiente.

En la figura 3 se pueden ver algunas de estas ventajas, a continuación se comentan algunas de ellas más detalladamente.

- **Mejora del confort:** Además de todos los sistemas de mejora del confort que incorporan los vehículos modernos, se puede hacer que el viaje sea aún más cómodo mediante la conducción eficiente. Se trata de evitar acelerones y frenazos bruscos, con lo que los ruidos correspondientes procedentes del motor se pueden eliminar, mantener una velocidad media constante, realizar el cambio de marchas conveniente que mantenga funcionando el motor de forma regular, etc. Ante todo, la conducción eficiente es un estilo de conducción impregnado de tranquilidad y que evita el estado de estrés producido por el tráfico al que están sometidos los conductores, sobre todo en ciudad. En la figura 4 se puede ver en qué porcentaje puede aumentar el confort al realizar una conducción eficiente.

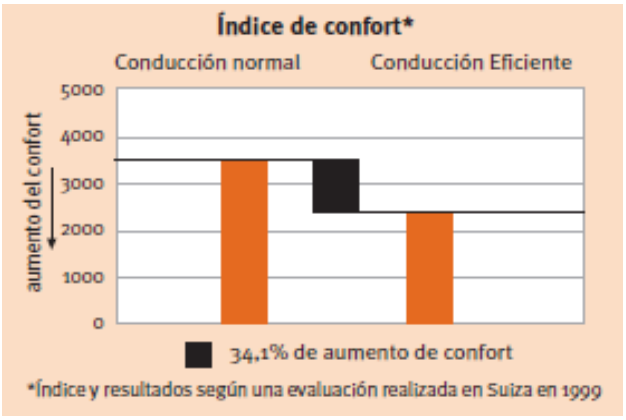

Figura 4. Mejora del confort gracias a la conducción eficiente.

- **Aumento de la seguridad:** El enorme progreso de las tecnologías ha permitido que los automóviles que hoy se conducen incluyan una serie de elementos que velan por la seguridad de los ocupantes. Pero aun así, las cifras de accidentes de tráfico no se reducen lo suficiente.

La conducción eficiente afecta a la seguridad al tener como principales enseñanzas: mantener una distancia de seguridad superior a la habitual, para tener mayor tiempo de reacción en caso de incidencias en el tráfico; mantener una velocidad media constante, para reducir la velocidad punta que puede llegar a alcanzarse en un determinado recorrido y conducir con anticipación y previsión manteniendo siempre un adecuado campo visual.

Estudios realizados en países europeos donde la conducción eficiente lleva tiempo implantada demuestran reducciones en las cifras y gravedad de los accidentes de tráfico.

- **Menor consumo:** El conductor, con su comportamiento, tiene una gran influencia sobre el consumo de carburante en el vehículo. Deberá tener especial cuidado en el arranque del vehículo, la utilización del acelerador, el uso de las marchas de forma adecuada o la anticipación frente a situaciones imprevistas del tráfico. Intentará también mantener una velocidad constante y adecuada a cada situación, para que su consumo se mantenga dentro de los niveles que marca la conducción eficiente, optimizando de esta forma el gasto de carburante. Se ha evaluado que con la conducción eficiente se puede ahorrar en torno a un 11,7% del carburante, tal y como se aprecia en la figura 5.

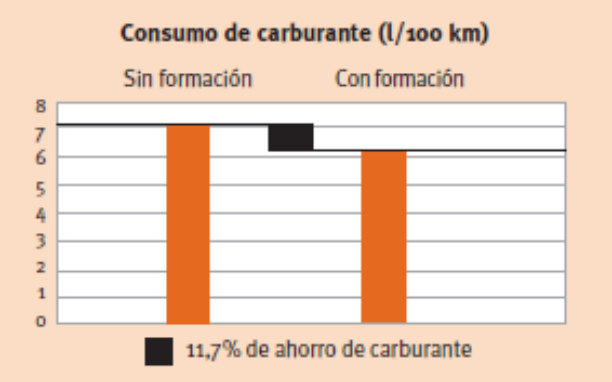

Figura 5. Ahorro en el consumo gracias a la conducción eficiente.

- **Menor coste:** El efecto de reducción del consumo está asociado a un menor coste de combustible y, a su vez, a un menor coste en mantenimiento del vehículo. Las pautas impuestas por la conducción eficiente provocan que todos los elementos del vehículo estén sometidos a un esfuerzo inferior al que soportarían en el caso de la conducción tradicional. Por ejemplo, la relación de marchas adecuada evita someter a la caja de cambios a esfuerzos innecesarios, y la anticipación y el uso del freno motor minimizan el desgaste del sistema de frenado.

- **Disminución de emisiones:** La reducción en el consumo de carburante lleva asociado directamente la reducción de emisiones contaminantes a la atmósfera. La contaminación atmosférica produce enfermedades. Agentes contaminantes como óxidos de carbono y de nitrógeno, hidrocarburos y partículas, se asocian a enfermedades como las dificultades respiratorias, los problemas oculares, las enfermedades cardiovasculares y las jaquecas. También corroen materiales y atacan a todo tipo de vegetación. En la figura 6 se puede observar en qué medida disminuye la emisión de cada uno de estos agentes contaminantes.

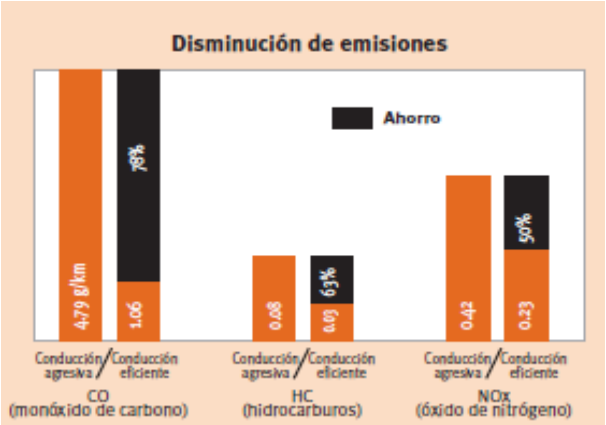

Figura 6. Disminución de emisiones gracias a la conducción eficiente.

2.2.2. Conceptos asociados y principales reglas de la conducción eficiente

Al fin de optimizar su conducción, y lograr dominar la conducción eficiente, estas son las principales claves a tener en cuenta:

- Circular en la marcha más larga posible y a bajas revoluciones.
- Mantener la velocidad de circulación lo más uniforme posible.
- En los procesos de aceleración, cambiar de marcha: entre 2.000 y 2.500 revoluciones en los motores de gasolina, y entre 1.500 y 2.000 en los motores diésel.
- En los procesos de deceleración, reducir de marcha lo más tarde posible.
- Realizar siempre la conducción con anticipación y previsión.
- Recordar que mientras no se pisa el acelerador, manteniendo una marcha engranada, y una velocidad superior a unos 20 km/h, el consumo de carburante es nulo.

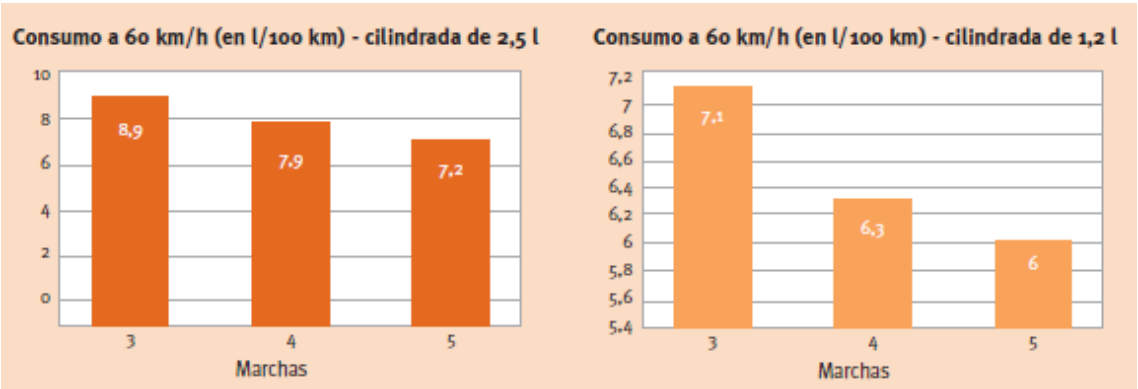

Figura 7. Diferencias en el consumo en función de la marcha a la que se consume.

Tal y como vemos en la figura 7, cuanto más larga es la marcha con la que se circula, siempre por encima de un número mínimo de revoluciones del motor, menor consumo de carburante; y, a mayor cilindrada, mayor impacto en el consumo tiene el circular en una marcha más larga. Por otra parte, en la figura 8 podemos apreciar cómo influye en el consumo la velocidad a la que se circula en una determinada marcha.

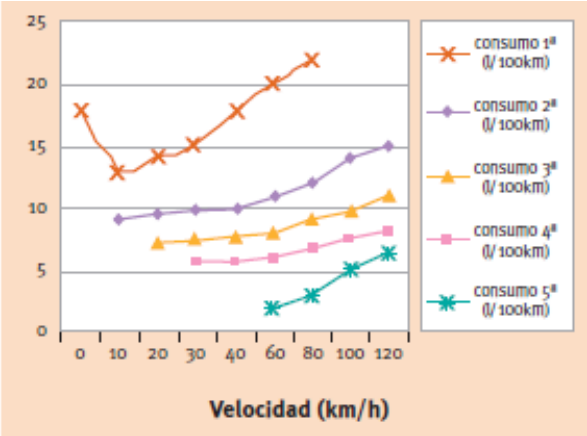

Figura 8. Diferencias en el consumo en función de la velocidad.

Aplicando las anteriores reglas, se efectuará un menor número de cambios de marcha. En pruebas realizadas, se ha comprobado que circulando lo más posible en las marchas más largas se obtiene un ahorro comparativo del orden del 20% en el número de cambios realizados, lo que significa un ahorro en el uso del embrague, de los frenos, de la caja de cambios y del motor. Se logra también con esta técnica un cambio de actitud en la conducción, creando un estilo de conducción menos agresivo, basado en la anticipación y en la previsión, que repercute en un menor grado de estrés para el conductor, y en una reducción del número de accidentes como indican las cifras de los países europeos en los que está plenamente implantada la conducción eficiente.

Una recomendación importante a tener en cuenta por los conductores formados en las técnicas de la conducción eficiente consiste en llevar el control del consumo del vehículo a lo largo del tiempo [13]. Este control se realizará mediante anotaciones de los kilómetros recorridos y litros de carburante consumidos cada vez que se procede a llenar el depósito. Esta sencilla actuación incrementa la eficacia de las técnicas de la conducción eficiente en el ahorro de carburante y logra conservar la actitud de ahorro evitando que se pierda con el transcurso del tiempo. Resulta además de mucha utilidad a la hora de realizar detecciones de averías al alertar sobre variaciones significativas de consumo de carburante [14].

En la tabla 2 [15] se puede observar el impacto que diferentes factores tienen en el consumo extra de carburante.

Eco parámetros	Consumo extra
Incorrecto arranque y cambio de marchas del vehículo	+25%
Incorrecta elección de la marcha de forma prolongada	+20%
Aire acondicionado encendido	+ [10% - 20%]
Ventanillas abiertas a 120 km/h	+7%
Presión de los neumáticos	+6%
Exceso de peso	+5% / 100 kg
Uso de accesorios en montacargas	+ [7.5% - 39%]
Uso del sistema de alumbrado	+ 2%
Uso de motor gasolina sobre diésel	+10%
Coche con antigüedad superior a los 25 años	+95%

Tabla 2. Impacto en consumo de los diversos factores relacionados.

2.3. Contribución de los simuladores de conducción

En los últimos años la industria del automóvil ha experimentado múltiples cambios y evoluciones, interviniendo además en el desarrollo de otras tecnologías como es en nuestro caso en los simuladores, los cuales permiten ahorrar grandes cantidades de dinero a la industria [16].

Actualmente los simuladores cada vez cobran más importancia en la industria dado que muchos avances han sido desarrollados gracias a estos. Los simuladores no solo reducen los gastos de ensayos y pruebas, sino también otros aspectos como el tiempo de desarrollo de prototipos.

Cada vez que un componente es diseñado, no vale con comprobar que sus proporciones sean adecuadas y que pueda realizarse su fabricación en serie, sino que se realizan todo tipo de ensayos, desde si gusta al público al que va dirigido, hasta su durabilidad o su interacción con otros elementos. Estas pruebas se conocen como ensayos destructivos dado que para su estudio irremediablemente la pieza se rompe, y siempre es mejor realizar simulaciones que romper piezas físicas.

Como ya se explicase en la introducción y motivación, los simuladores aportan también la posibilidad de repetir los escenarios de una simulación de forma indefinida. Además, es posible simular situaciones de la vida real que serían prácticamente imposibles de simular de forma voluntaria en un entorno real, al menos sin poner en riesgo a personas y vehículos por igual. Los simuladores permiten además realizar estudios de cómo ciertos factores pueden influir en las capacidades de conducción, tales como enfermedades, el estado físico del conductor, el consumo de medicamentos, drogas o alcohol, los efectos psicológicos o la edad.

Por último, es importante mencionar que los simuladores permiten recopilar datos de forma simple. En este proyecto que nos ocupa se ha creado un sistema que permite que estos datos sean almacenados y procesados por un servidor remoto. A nivel de usabilidad, también permite a los usuarios acceder a información de las simulaciones que han realizado y compararlas con datos medios de otros usuarios, e incluso en algunos simuladores acceder a la grabación completa de su simulación. Esto permite que un conductor pueda ver qué fallos o infracciones ha cometido durante su conducción y en qué campos debe mejorar.

2.3.1. Simuladores de conducción existentes en el mercado

En la actualidad existen varios simuladores de uso profesional implantados en el mercado. A continuación, podemos ver un breve resumen de las características de algunos de ellos:

- **DriveSIM:** desarrollado por el Instituto Tecnológico de Castilla y León para la empresa Arisoft S.A. Está programado en Unity para ser utilizado en Windows, habiendo sido finalista de los Unity Awards en 2014, edición en la que se erigió vencedor el popular juego Hearhstone [17].

  Dispone de tramos urbanos, de carretera, autopista y de montaña; permite establecer distintos tipos de climatología; su tráfico y peatones están dotados de Inteligencia Artificial; permite distintas configuraciones del vehículo a utilizar: gasolina o diésel, manual o automático, cilindrada del motor o tipo de tracción; dotado de adaptación de las señales de tráfico al país; y disponible en varios idiomas: Inglés, Español, Italiano, Alemán, Portugués, Francés, Árabe…

Además contempla el gasto de combustible y emisión de gases, convirtiéndose en uno de los simuladores más completos en ese aspecto.

Dado que está realizado con Unity al igual que el nuestro, este simulador es un ejemplo perfecto de lo que se puede llegar a lograr si se invierte en personal y la dedicación adecuada [18].

- **SmartSIM:** desarrollado por INDRA para la CNAE (Confederación Nacional de Autoescuelas). Dispone de una variedad de escenarios: circuitos de maniobras, urbano/interurbano, autovía…; permite establecer distintos tipos de meteorología; físicas realistas del tráfico y peatones, etc.

  Para su uso es necesaria una cabina especializada, que se compone de: asiento regulable, cinturón de seguridad, volante, pedales, cuadro de instrumentos completo… Y permite que varias cabinas funcionen al mismo tiempo controladas por una posición de instructor o que el instructor pueda conducir cualquier vehículo del tráfico para interactuar con el vehículo del alumno.

  Además dispone de reporte en tiempo real (el sistema evalúa toda acción del usuario incluyendo *Eye-Tracking*, grabando, en la base de datos del alumno, todo tipo de errores, los cuales generan puntuación, alertas y mensajes visuales para alertar al alumno), grabación de ejercicios, informes, gráficos y telemetría.

- **Simescar:** desarrollado en parte por la empresa española Simumak, que cuenta con presencia en multitud de países y está especializada en la creación de simuladores [19]. Destaca porque está especializada no solo en la conducción de coches, camiones, motocicletas o incluso avionetas sino también en el uso de maquinarias pesadas como excavadoras, carretilleras elevadoras, grúas, etc.

  Además de contar con las características de los anteriormente mencionados simuladores, dispone del software Sócrates, que permite al instructor gestionar los perfiles de todos los alumnos y mantener un seguimiento de su progreso.

Estos tres simuladores que se acaban de mencionar no son más que un pequeño ejemplo de la implantación que están teniendo en los últimos años a nivel profesional dadas las múltiples ventajas ya mencionadas que aportan frente a la conducción convencional.

3. Herramientas disponibles para la realización del simulador

A lo largo de este apartado se expondrán las distintas opciones de las que uno dispone cuando quiere enfrentarse al reto de hacer un simulador como el que se ha llevado a cabo en este proyecto.

Será necesario utilizar un motor gráfico que nos dé todas las herramientas necesarias para crear la lógica de los eventos y la física de los elementos; y, aunque ya hemos mencionado previamente haber optado por *Unity*, veremos cómo no era la única gran opción. Así mismo, se mencionarán también los programas de diseño 3D de los cuales uno se puede ayudar para mejorar el aspecto del trabajo.

Por último, también se analizarán las múltiples tecnologías y lenguajes disponibles para el desarrollo de la plataforma web de almacenamiento y procesamiento de datos, aunque de una forma mucho más breve dado que se ha utilizado una plataforma ya anteriormente creada por el grupo de investigación GTI para tal fin.

3.1. Motores gráficos

Es indiscutible que la escena de videojuegos ha crecido de forma exponencial durante los últimos años. Cada vez son más las personas que se animan a crear su propio videojuego usando las infinitas herramientas y ayudas que se pueden encontrar en Internet [20].

Ahora es más fácil encontrar todo el material necesario para empezar en el mundo del diseño y la programación de videojuegos. Existen desde programas que te ahorran el tedioso trabajo de crear el código hasta motores gráficos de grandes compañías que se pueden encontrar de forma gratuita y sin ningún tipo de limitación a la hora de dejar volar nuestra imaginación.

Hoy en día existen una amplia gama de motores gráficos destinados a la creación de videojuegos, pero en cuanto a los utilizados para PC y consolas de última generación hay 5 que se reparten gran parte del mercado: *Source 2, Unreal Engine 4, Unity 5, Cryengine y Ubiart Framework*; además de otras buenas opciones si uno quiere empezar desde abajo con algo más fácil de utilizar como *Stencyl* o *GameMaker*.

3.1.1. Source 2

Source es un motor de videojuegos desarrollado por la empresa *Valve Corporation* para las plataformas *Microsoft Windows* (32 y 64 bits), *Mac OS X, GNU/Linux, Xbox One, Xbox 360, PlayStation 3* y *PlayStation 4*. Desde que debutase en 2004 con el popular videojuego *Counter-Strike*, ha evolucionado constantemente, dando lugar a numerosos éxitos más como *Half Life, Left 4 Dead* o *Dota* [21].

Parte del éxito de *Source* se basa en que fue creado para ir evolucionando poco a poco mientras la tecnología avanza, al contrario que los cambios de versión bruscos de sus competidores. Esto se vuelve especialmente relevante cuando se considera que está ligado a *Steam*, el cual baja las actualizaciones automáticamente lo que hace que nuevas versiones del motor puedan llegar a toda la base de usuarios instantáneamente [22].

Creado completamente en C ++, el motor *Source* está diseñado teniendo en cuenta la extensibilidad, la flexibilidad y el rendimiento. Con licencia de *Source* se dispone de acceso a todo el código fuente que *Valve* usa para construir el motor, así como al de sus juegos. Esto permite a su personal de desarrollo dedicar su tiempo a realizar el diseño de su juego en lugar de comenzar desde el principio. Además cuenta con una herramienta propia para el diseño de mapas llamada *Valve Hammer Editor*, cuya interfaz podemos ver en la figura 9.

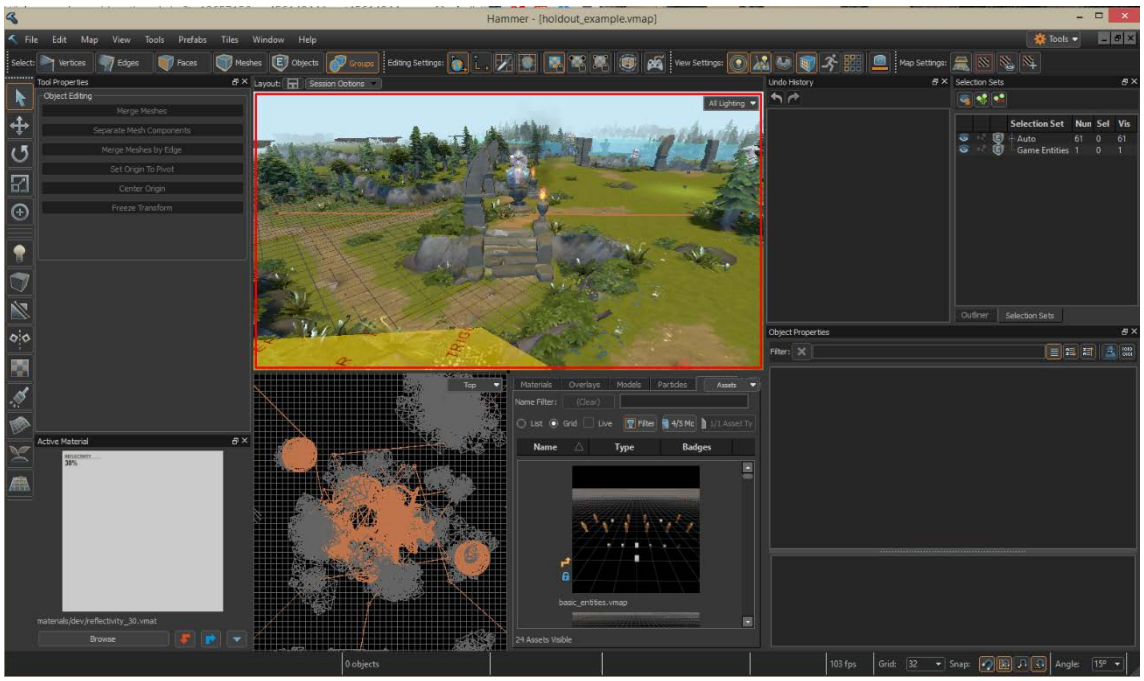

Figura 9. Interfaz del Valve Hammer Editor.

Algunas de las características de este motor, de las que ellos mismos en su web se enorgullecen [23] son:

- Soporte escalable multi-procesador.
- Motor de física eficiente en ancho de banda para red.
- Mapas de iluminación y sombras dinámicos pre-computados.
- *Renderizado* HDR (*High Dynamic Range*).
- *Renderizado* con *DirectX* en *Windows* y *Xbox* y con *OpenGL* en *Linux*, *MacOS* y *Android*.

3.1.2. Unreal Engine 4

Unreal Engine es un motor de juego de PC y consolas creado por la compañía *Epic Games* en 1998, y es considerado como el motor más potente por los expertos, empleándose en el desarrollo de juegos de alto coste económico, denominados comúnmente como juegos Triple A, juegos como *Unreal Tournament, Tom Clancy's Rainbow Six: Vegas, America's Army, Red Steel, Gears of War, BioShock, Star Wars Republic Commando* o *Batman: Arkham Asylum* [24].

La versión actual, *Unreal Engine* 4, está diseñada para las plataformas *Microsoft Windows*, *macOS, Linux, SteamOS, HTML5, iOS, Android, Nintendo Switch, PlayStation 4, Xbox One SteamVR/HTC Vive, Oculus Rift, PlayStation VR, Google Daydream, OSVR* y *Samsung Gear VR*.

Desde el 2 de marzo de 2015, a través de un comunicado oficial [25], el motor *Unreal Engine* 4 está disponible para todo aquel que lo desee de forma gratuita, al igual que todas las actualizaciones que se lancen de él. Según se puede leer en el comunicado en la página oficial: "Puedes descargar el motor gráfico y usarlo para cualquier cosa en el desarrollo de videojuegos, educación, arquitectura, visualización de realidad virtual, cine y animación. Si cualquiera de estos proyectos se comercializa de forma oficial *Epic Games* obtendría el 5 % de los beneficios de la obra cada trimestre cuando este producto supere sus primeros 3000 dólares".

Creado en C++, permite el acceso a su código fuente de modo que es posible personalizar y ampliar las herramientas de su editor, la física, audio, animación y *renderizado*. Una funcionalidad interesante es que permite navegar por las funciones de C++ directamente desde los objetos del juego accediendo desde ahí al código fuente. Esto hace que sea posible cambiar el código mientras el juego está en ejecución, pudiendo así observar los resultados a los cambios de forma inmediata. En la figura 10 podemos ver una imagen del entorno de este motor gráfico.

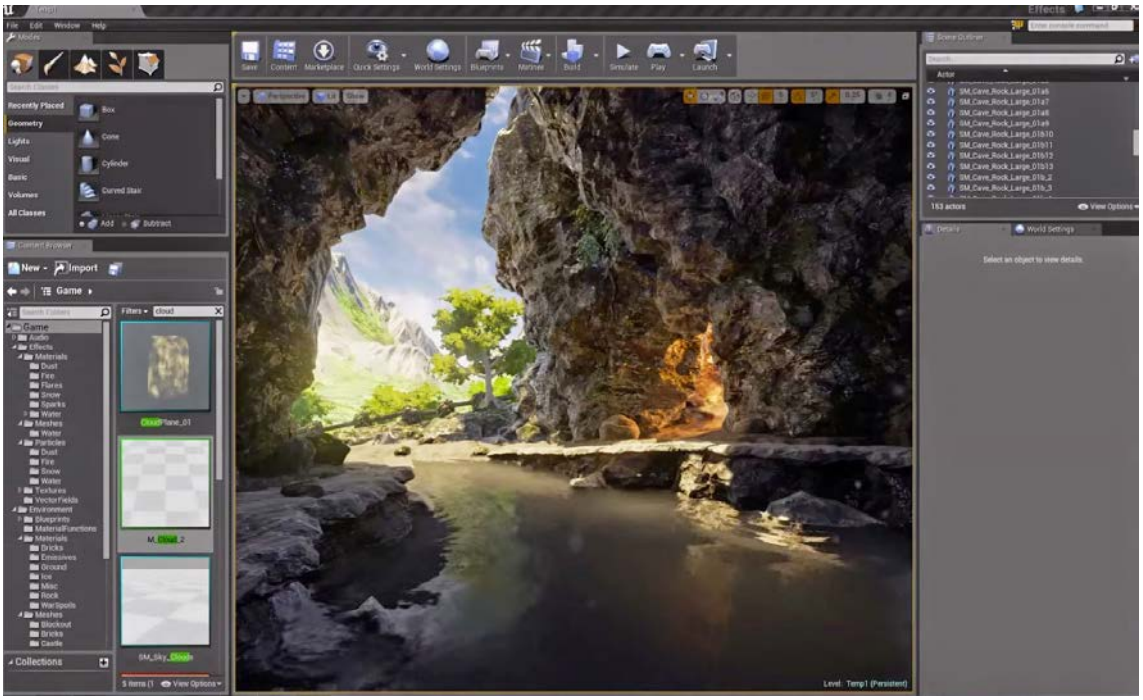

Figura 10. Entorno de Unreal Engine 4.

En esta su última versión hasta la fecha, *Unreal* soporta *DirectX 11* y *Direct X 12*. Además, cuenta con una herramienta de *Scripting Visual* denominada *Blueprint*, que permite construir la lógica de los scripts mediante bloques relacionados entre sí.

3.1.3. Unity 5

Unity es una plataforma de desarrollo flexible y potente para la creación de juegos y aplicaciones gráficas multiplataforma tanto bidimensionales como tridimensionales [26]. En realidad dispone de dos motores de física por separado, *BoxPhysics* para 2D y *NvidiaPhys* para 3D. Es una plataforma de desarrollo para *Windows, OS* y *Linux*, que permite crear juegos para estas plataformas además de para *XBox One, PlayStation4, Wii, iPhone, Android* o *Windows*

Phone. Algunos de los videojuegos desarrollados con *Unity* son: *Battlestar Galactica Online* [27], *Hearthstone: Heroes of Warcraft y Assassin's Creed Identity*. En la figura 11 podemos ver una imagen de todos los logos de las plataformas a las que Unity permite exportar sus aplicaciones.

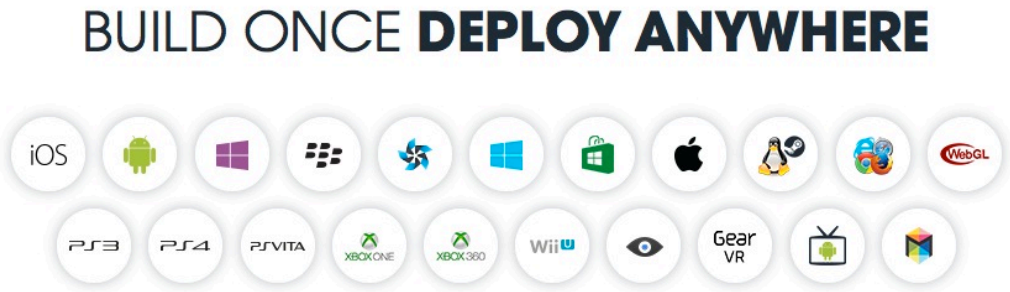

Figura 11. Plataformas a las que Unity permite exportar las aplicaciones.

Como ya se ha mencionado en alguna ocasión a lo largo de esta memoria, *Unity* ha sido el motor elegido, y, aunque en apartados posteriores hablaremos de nuevo sobre algunas de sus características y cualidades, es interesante mencionar aquí otras de ellas.

Unity tiene compatibilidad con una gran variedad de herramientas de diseño 3D sobre las que hablaremos también a continuación (*Blender, 3ds Max, Maya, Softimage, Modo, ZBrush, Cinema 4D, Cheetah3D, Adobe Photoshop*). Si aplicamos en algún objeto usado en el videojuego algún cambio usando estas herramientas, el objeto se actualizará automáticamente en *Unity* sin necesidad de reimportarlo.

MonoDevelop**,** la implementación de código abierto de .NET *Framework*, es el *entorno de desarrollo integrado* (IDE) proporcionado con Unity. Un IDE combina la operación familiar de un editor de texto con características adicionales para depurar y gestionar otras tareas de proyecto. MonoDevelop es instalado por defecto con Unity, aunque hay una opción para excluirlo en las Ventanas de instalación.

En cuanto al lenguaje, los programadores pueden utilizar *UnityScript* (un lenguaje personalizado inspirado en la sintaxis *ECMAScript*), C# o Boo (que tiene una sintaxis inspirada en Python). Es además compatible con Visual Studio para el desarrollo del código del videojuego, así como un editor de código propio, e incluye *Unity Asset Server*, una solución de control de versiones para todos los *assets* de juego y scripts.

En noviembre de 2010 se lanzó el *Unity Asset Store* que es un recurso disponible en el editor de *Unity*. Más de 150.000 usuarios de *Unity* pueden acceder a la colección de más de 4.400 paquetes de *Assets* en una amplia gama de categorías, incluyendo modelos 3D, texturas y materiales, sistemas de partículas, música y efectos de sonido, tutoriales y proyectos, paquetes de *scripts*, extensiones para el editor y servicios en línea.

Unity cuenta además con la herramienta de *debugging Profiler*, incluida por defecto desde la versión 5 siendo una de sus características más importantes de cara al desarrollo, pues permite depurar la aplicación y analizar el rendimiento en términos de carga sobre el procesador y la tarjeta gráfica, detectando el rendimiento de la aplicación en distintas áreas como en la memoria utilizada, la velocidad de ejecución de los *scripts* o para ver el tiempo de *renderizado*.

En cuanto a licencias y coste, *Unity* dispone de 4 licencias de software diferentes. La básica para uso personal con la cual comenzar a desarrollar aplicaciones en *Unity*. La licencia Plus con la capacidad de editar la interfaz, informes de ejecución y acceso al software educativo para la formación durante 1 mes. La versión Pro con soporte *premium*, acceso al código de la herramienta y beneficios ilimitados y, por último, la versión *Enterprise*, con características para modo multijugador y análisis personalizables. Esta última versión es la usada por empresas que se dedican al desarrollo de aplicaciones en *Unity*. El precio de estas licencias citadas anteriormente se puede observar en la figura 13. La licencia más básica es de uso gratuito, mientras que las licencias Plus y Pro son de pago mensual (35$ y 125$). La licencia Enterprise, usada en empresas, varía en función del número de licencias requeridas y el uso de la herramienta, por lo que la empresa que la desee solicitar deberá ponerse en contacto con *Unity*. Todos estos precios y características de cada licencia pueden ser consultados en su página oficial [28]. Por nuestra parte, comentar que la utilizada, como cabía imaginar, ha sido la personal.

A lo largo del capítulo correspondiente al desarrollo del proyecto podremos ver muchas otras características de este motor así como distintas imágenes relativas a la interfaz y a alguno de sus escenarios.

3.1.4. CryEngine

CryEngine es un motor de juego creado por la empresa alemana desarrolladora de software *Crytek*, originalmente un motor de demostración para la empresa *Nvidia*, que al demostrar un gran potencial se implementa por primera vez en 2006 en el videojuego *FarCry* [29]. A partir de este momento y dado su éxito, la totalidad de los derechos de *CryEngine* son adquiridos por la distribuidora de videojuegos *Ubisoft*. Desde entonces han existido las versiones 2 y 3 bajo las que se publicaron algunos videojuegos como *Crysis* o *Robinson* respectivamente, hasta llegar a la más actual, *CryEngine V*, presentada en 2016 [30].

Este motor tiene soporte para PC, *PlayStation 4, Xbox One, Wii, Android* e *iOS*. Cuenta además con el editor *SandBox*, que permite que varias personas trabajen de manera simultánea en un mismo escenario dividiendo en capas el entorno.

Es posible trabajar tanto con LUA como con C++ como lenguajes para el código. Además, cuenta con un sistema visual de *scripting* muy potente similar al de *Blueprint* mencionado de *Unreal Engine*, el cual da la posibilidad de crear la lógica del juego de una manera más intuitiva sin la necesidad de escribir nada de código. En la figura 12 podemos observar cómo es su interfaz.

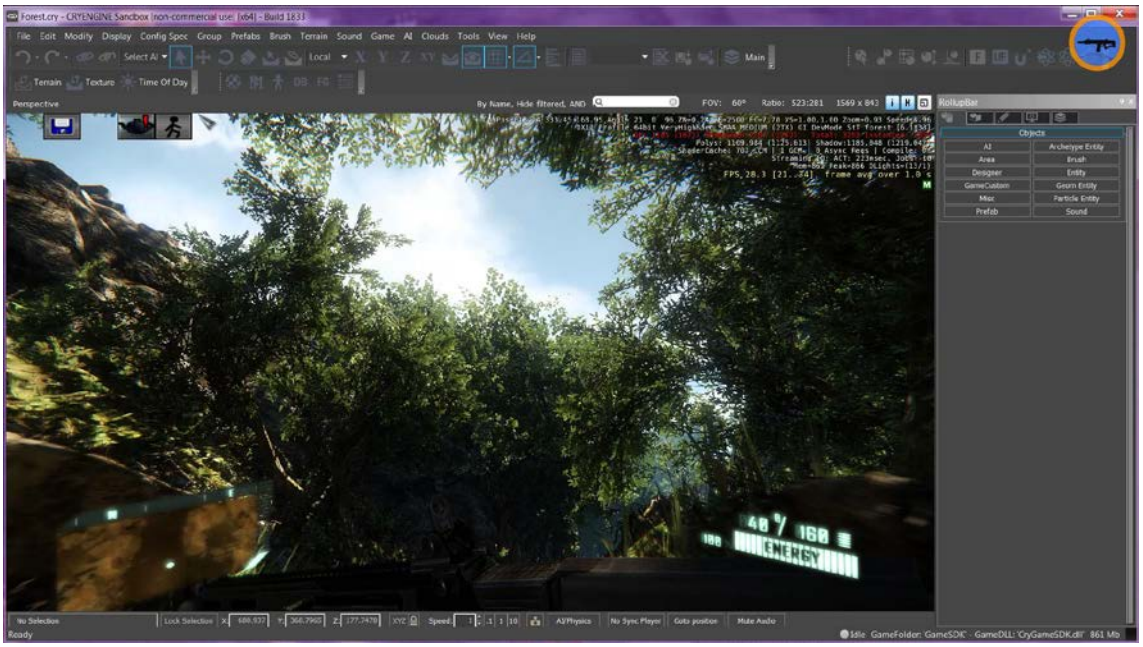

Figura 12. Interfaz de CryEngine.

Dispone de licencia gratuita para proyectos sin ánimos de lucro, además de versiones de pago para aquellos que quieran sacarle mayor rendimiento al motor.

3.1.5. Ubiart Framework

Es un motor de videojuego en 2.5D desarrollado por *Ubisoft Montpellier*. Su función es organizar gráficos vectoriales 2D animados dentro de un videojuego sin necesidad de códigos extensos. *Ubisoft* es la mayor compañía de videojuegos que existe actualmente. Aunque tiene proyectos de una magnitud inabarcable para la mayoría de estudios, como desarrollar al mismo tiempo juegos con altísimos presupuestos como *Assassin's Creed, The Division, The Crew* y *Watch Dogs*, parece que ha encontrado un nuevo canal por el que apostar, el de los juegos con presupuestos menores en el que la mayor baza es la destreza de sus artistas, ya sean guionistas, animadores, artistas conceptuales o compositores.

Las características del motor permiten, además de la facilidad y flexibilidad a la hora de crear animaciones, ejecutar lo que vemos en pantalla comiendo pocos recursos y requiriendo la intervención de pocos artistas gráficos. Es un motor preparado para la era *FullHD* y con un rendimiento de 60 *frames* por segundo. Además permite crear imágenes, escenarios y animaciones en muy alta resolución partiendo de conceptos en muy pequeña resolución. *UbiArt Framewrok* parte de transformar dibujos y animaciones hechas prácticamente a mano y dotarlas de animaciones en 2.5 dimensiones y posteriormente armar una obra de arte hecha videojuego. El motor gráfico en la actualidad ha sorprendido por fascinantes entregas, entre las que destacan: *Rayman Origins*, *Rayman Legends*, *Child Of Light* y *Valiant Hearts*. En la figura 13 se puede ver una de las escenas del mencionado videojuego de *Rayman Origins*.

Figura 13. Imagen del videojuego Rayman Origins desarrollado con Ubiart.

Uno de los grandes inconvenientes de este motor es que no podremos usarlo para algo que no sea un juego de plataformas en 2D o 2.5D, aunque el mayor inconveniente de todos es que tendremos que pasar por conseguir la autorización de *Ubisoft* para poder darle un uso personal, a no ser que la propuesta de convertirlo en *Open Source* se haga realidad [31]. A pesar de ello se postula como uno de los grandes competidores del mercado de cara al futuro [32].

3.1.6. Comparativa y elección

En el presente trabajo se ha llevado a cabo un exhaustivo estudio de algunos de los motores de juegos más importantes del momento [33] [34]. Tras haber comentado algunas de sus características, comentaremos ahora el motivo de nuestra elección, teniendo en cuenta siempre nuestro enfoque: crear un simulador en un tiempo limitado y de forma unipersonal.

Esta situación personal nos hace tener que descartar a la mayoría de motores incluso aunque sean más potentes o más eficientes. Además, al necesitar un motor 3D, tenemos que descartar todos aquellos que no lo sean, y debemos escoger uno cuya curva de aprendizaje permita realizar el proyecto en un número de horas limitado, lo cual nos hace descartar a un motor como *CryEngine* o *Unreal*. Deberemos escoger por otra parte uno cuya licencia sea gratuita.

En cuanto al lenguaje, C++ es un lenguaje potente, pero de menor nivel que por ejemplo C#, el cual posee nociones muy similares a Java siendo un lenguaje tipado de alto nivel con unas funcionalidades muy diversas.

En el mundo del desarrollo no existe una herramienta para todo, pero sí que hay una herramienta para cada problema, lo difícil es tomar la decisión. En nuestro caso es importante mencionar que la decisión de trabajar con *Unity* fue ya proporcionada de forma previa, no cabiendo lugar a una alternativa, pero tras el análisis realizado se llega a la conclusión de que esta decisión fue acertada y que muy probablemente hubiese sido el motor elegido en el caso de haber podido tomar esa decisión de manera abierta.

En cualquier caso, y a pesar de tener una decisión ya hecha, se considera interesante incluir la tabla 3 donde podemos ver a modo de síntesis las características de los motores

anteriormente mencionados, excluyendo *Ubiart* aún menos contrastado debido a su licencia, de forma que podamos comparar rápidamente sus características de un solo vistazo [35].

Nombre	2D/3D	Lenguaje	Motor físico	Características avanzadas	Licencia	Datos de interés
Source 	3D	C#	Motor propio	Forward rendering	Comercial	
Unreal Engine 	2D y 3D	Unreal Script	PhysX	Deferred rendering, occlusion culling, parallax mapping, Per Object Motion Blur	Comercial. Versión libre y pago de royalties sobre producto vendido al superar un umbral de facturación.	Motcon, el motor gráfico más potente del momento según los expertos
Unity 3D 	2D y 3D	C#, Boo, JS.	PhysX	Forward y Deferred rendering, Occlusion Culling, light mapping, light probing, LOD discreto, edición de terrenos, sistema de partículas	Comercial. Versión básica gratuita y versiones PRO y plataformas móviles y consolas con un pago único sin royalties	Motor con mayor número de desarrolladores registrados del momento
CryEngine 	3D	C#	Motor propio	Occlusion culling, Deferred rendering, parallax mapping, Per Object Motion Blur	Comercial. Es gratis para uso no comercial	CryENGINE Cinebox - Herramienta para reducir el Time to Market de los creadores de Contenido

Tabla 3. Comparativa de los distintos motores gráficos.

3.2. Programas de modelado 3D

A grandes rasgos, cuando hablamos de diseño 3D nos referimos tanto a la creación tridimensional de piezas, objetos o estructuras, empleado generalmente en ingeniería y arquitectura, como a la generación de imágenes en 3D relacionadas con el mundo multimedia y la animación 3D.

Los gráficos 3D se generan mediante un proceso de cálculos matemáticos sobre formas geométricas tridimensionales creadas por herramientas de diseño 3D (las cuales veremos más adelante), con el objetivo de representar visualmente el objeto.

Estos cálculos requieren de una gran carga computacional y de procesado por lo que se necesitan computadores de grandes capacidades, como la aceleración gráfica 3D. Esto hoy en

día es posible gracias a las tarjetas gráficas y procesadores del mercado. Estos dispositivos están formados por uno o más procesadores (*Graphic Processing Unit, GPU*) diseñados especialmente para acelerar las operaciones a realizar, las cuales suponen reproducir imágenes tridimensionales sobre una pantalla bidimensional y, de esta forma, liberar de carga de proceso a la unidad central de proceso (*Central Processing Unit, CPU*) del ordenador.

Los pasos del modelado 3D se pueden dividir de una forma sencilla en los siguientes:

- Modelado: consiste en dar forma a los objetos individuales que luego serán usados en la escena.
- Sombreado: mediante esta técnica se define cómo se comportarán las caras de un polígono cuando es iluminado por una fuente de luz.
- Texturizado: para incrementar el detalle y el realismo de los modelos creados se pueden añadir texturas. Se trata de una imagen que se coloca en las caras del polígono.
- Animación: Una vez creados los objetos, se pueden animar mediante transformaciones básicas o bien usando otras técnicas más avanzadas como esqueletos, deformadores o transformaciones dinámicas.
- *Renderizado*: consiste en generar una imagen 2D o animación a partir de la escena creada. Este proceso necesita una gran capacidad de cálculo.

Aunque *Unity* cuenta con un primitivo editor de formas geométricas, no es suficiente en muchas ocasiones, ya que la aplicación de texturas, mapas de sombreado o mapas de normales se hace de forma exterior. Salvo para casos muy simples o concretos, se ha optado en este proyecto por emplear modelos creados por la comunidad con licencia gratuita no comercial ya sea del *Asset Store* principalmente o de páginas ajenas. De este modo se logra agilizar de manera notable el proceso al no ser necesario modelar uno a uno los objetos que se deseen incluir en los escenarios.

Cada software tiene sus ventajas y desventajas frente a los demás, pero la posibilidad de realizar un trabajo de calidad no depende de esto, sino de los conocimientos, la creatividad, y no tanto del software. El mejor programa 3D será aquel con el cual el usuario se encuentre más cómodo a la hora de trabajar, teniendo en cuenta su forma y filosofía de trabajo. A continuación se exponen de manera muy breve algunos de los principales programas, como son *Autodesk 3ds Max, Autodesk Maya, Blender, SketchUp* y *Cinema 4D*.

3.2.1. Autodesk 3ds Max

Anteriormente *3D Studio Max*, es un software de modelado, animación y *renderizado* 3D perteneciente a *Autodesk*, que salió a la venta por primera vez en 1990 para DOS [36].

3ds Max, con su arquitectura basada en *plugins*, es uno de los programas de animación 3D más utilizado, especialmente para la creación de videojuegos, anuncios de televisión, en arquitectura o en películas [37]. Se ha utilizado para desarrollar títulos como *Tomb Raider, Red Alert, Diablo* y *Warcraft* [38].

Permite realizar tareas de modelado, mapeado UV, texturizado, *rigging, weighting*, animación, simulación de partículas, *scripting*, *renderizado*, composición y post-producción.

Respecto a la animación 3D, cuenta con un sistema de secuencia de cámara que permite cortar entre varias cámaras, recortar y reordenar clips, manteniendo intactos los originales. Permite

crear personajes con pieles muy realistas mediante el modificador *Skin*. Mediante la función *Populate*, permite crear multitudes tanto dinámicas como estáticas, además de permitir generar comportamientos y animaciones como andar, correr, girar, sentarse, etc.

En cuanto al *renderizado*, *3ds Max* funciona con la mayoría de los *renderizadores* más importantes, como *Arnold, V-Ray, Iray* y *Mental ray*, para crear escenas de gran calidad e imágenes impactantes. Además incorpora el renderizado de elementos mediante la computación en la nube, de forma que no se necesite emplear la capacidad de procesamiento del ordenador utilizado.

Los datos pueden organizarse en capas anidadas para una mayor facilidad en el trabajo. *3ds Max* se puede ampliar y personalizar mediante el lenguaje de programación *Phyton*. Existen otras funciones más específicas como *Civil View* que permite fácilmente transformar el espacio civil en un modelo 3D o *DirectConnect* que permite intercambiar datos de diseño industrial con aquellos que utilicen herramientas CAD. En la figura 14 podemos ver cómo es su entorno.

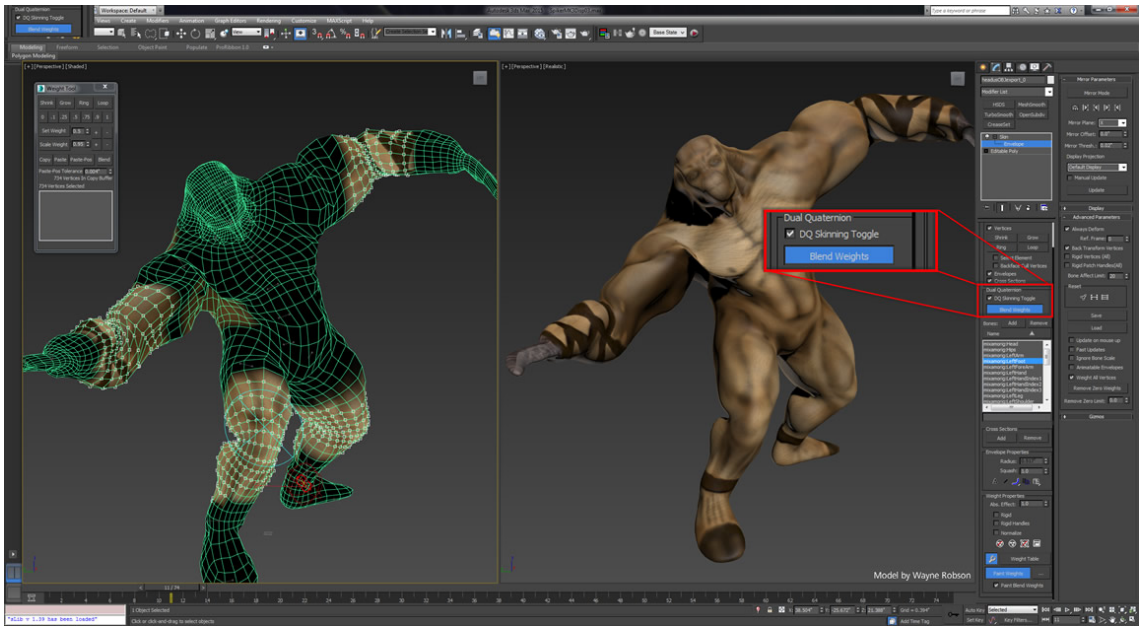

Figura 14. Editor 3ds Max.

Cuenta con herramientas de aprendizaje tanto para comenzar a trabajar como para personas más avanzadas en su manejo. Destacar también el precio de su licencia, que es de 242 € la mensual o 1936 € la anual [39].

3.2.2. Autodesk Maya

También conocido como *Maya*, es un programa informático dedicado al desarrollo de gráficos 3D por ordenador, efectos especiales y animación. Surgió a partir de la evolución de *Power Animator* y de la fusión de *Alias* y *Wavefront*, dos empresas canadienses dedicadas a los gráficos generados por ordenador [40]. Más tarde *Silicon Graphics* (ahora SGI), el gigante informático, absorbió a *Alias-Wavefront*, que finalmente fue absorbida por *Autodesk* dueña de *3d Studio Max*, por la cantidad de 182 millones de dólares.

Maya se caracteriza por su potencia y las posibilidades de expansión y personalización de su interfaz y herramientas. MEL (*Maya Embedded Language*) es el código que forma el núcleo de *Maya* y gracias al cual se pueden crear scripts y personalizar el paquete.

El programa posee diversas herramientas para modelado, animación, renderizado, simulación de ropa y cabello, dinámicas (simulación de fluidos), etc, dado que se divide en diferentes módulos integrados en el entorno de simulación unificada *Maya Nucleus* para aportar características adicionales más personalizables y versátiles: *Maya nParticles, Maya Fluid Effects, Maya nCloth y Maya nHair*. En la figura 15 podemos ver su entorno.

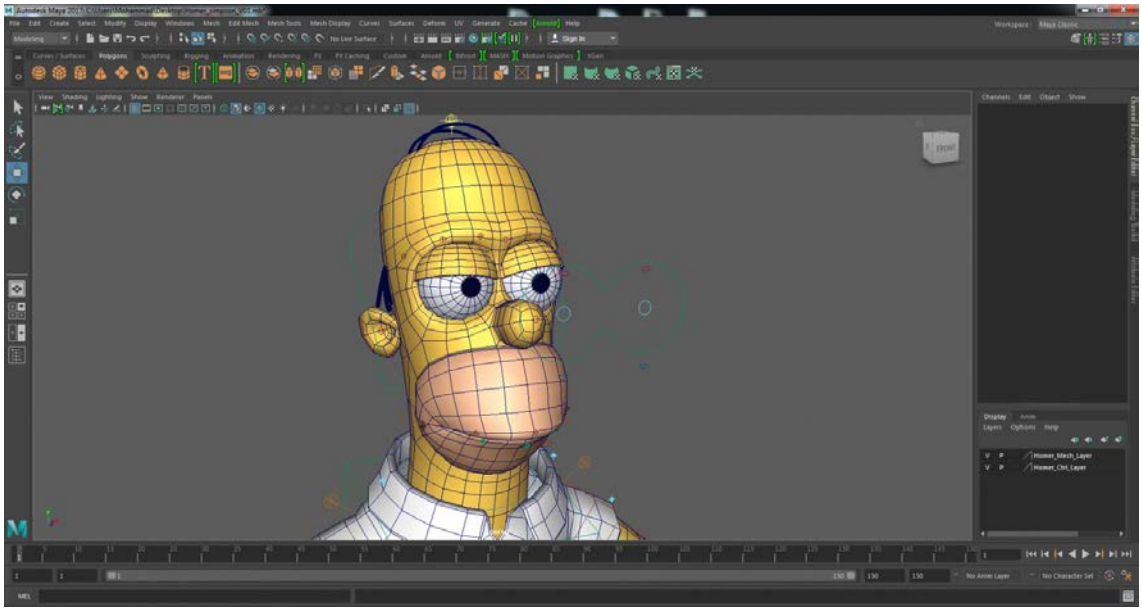

Figura 15. Editor Autodesk Maya.

Además, *Maya* es el único software de 3D acreditado con un *Oscar* gracias al enorme impacto que ha tenido en la industria cinematográfica como herramienta de efectos visuales, con un uso muy extendido debido a su gran capacidad de ampliación y personalización.

El precio de su licencia, al igual que para *3ds Max* también de *Autodesk*, es de 242 € la mensual o 1936 € la anual, de lo que podemos sacar la conclusión de que al menos para la empresa propietaria de ambos, ninguno está en prestaciones por debajo del otro a pesar de sus diferenciadas características [41].

3.2.3. Blender

Blender es un software libre, dedicado especialmente al modelado, iluminación, renderizado, animación y creación de gráficos tridimensionales [42]. En *Blender* se pueden desarrollar videojuegos ya que posee un motor de juegos interno y tiene además licencia permisiva para vincular con cualquier software externo [43].

El programa fue inicialmente distribuido de forma gratuita pero sin el código fuente, con un manual disponible para la venta, aunque posteriormente pasó a ser software libre. Actualmente es compatible con todas las versiones de *Windows, Mac OS X, GNU/Linux (Incluyendo Android), Solaris, FreeBSD* e *IRIX*.

Es la solución más asequible para *freelances* ya que a pesar de ser gratuito permite llevar a cabo la mayoría de las funciones de *Autodesk 3ds Max*. Como desventaja tiene una curva de aprendizaje complicada. Su enfoque está destinado más a la animación de video en tiempo real y modelado tridimensional.

Permite crear herramientas personalizadas y complementos mediante el lenguaje de programación *Phyton*, y su interfaz es flexible de forma que su diseño se pueda personalizar completamente. En la figura 16 podemos ver cómo es esta interfaz.

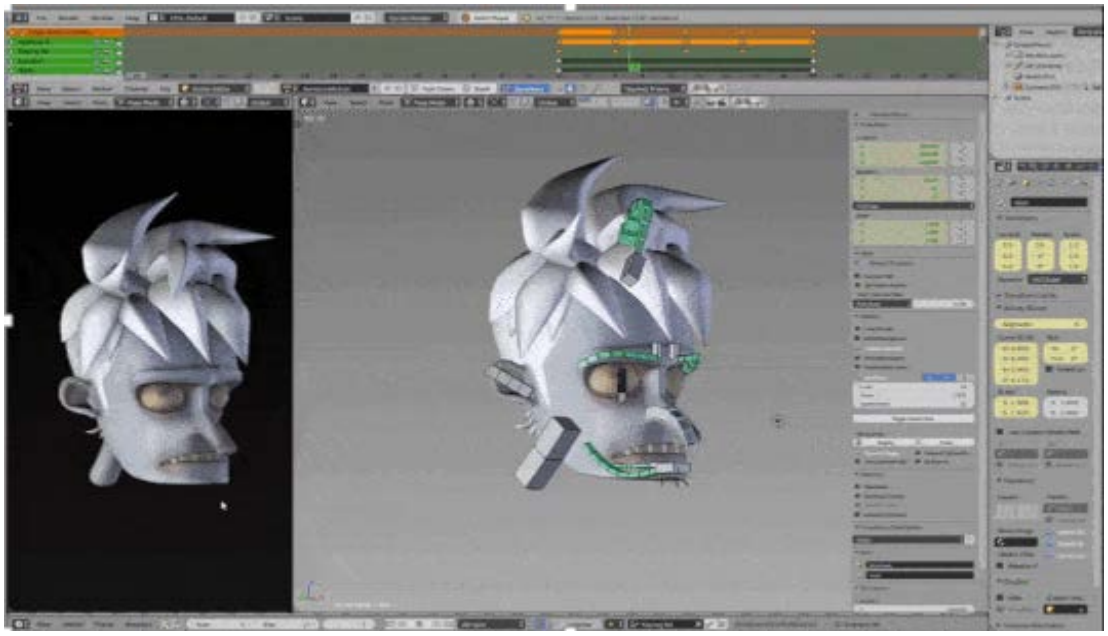

Figura 16. Interfaz para la animación con Blender.

El programa cuenta con una gran variedad de extensiones creadas por su comunidad de desarrolladores, que pueden ser activadas o desactivadas fácilmente: generación de árboles esculpido del terreno, simulación de rotura de objetos, herramientas para impresión 3D, etc. También incluye un editor de video que permite cortar y juntar videos, mezcla de audio, sincronización y control de velocidad y transiciones, además de otras funciones avanzadas. Permite también la importación y exportación de muchos formatos diferentes, y cuenta con tutoriales *online* con los que poder adquirir los conocimientos necesarios para trabajar con el programa.

3.2.4. SketchUp

Anteriormente *Google SketchUp*, es un programa de diseño gráfico y modelado en tres dimensiones basado en caras [44]. Más pensado para entornos de arquitectura, ingeniería civil, diseño industrial y diseño escénico aunque también utilizado para videojuegos o películas. Es un programa desarrollado por *Last Software*, empresa adquirida por *Google* en 2006 y finalmente vendida a *Trimble* en 2012 [45].

Las extensiones para *SketchUp* se hacen con el lenguaje de programación interpretado y orientado a objetos llamado Ruby. Fue diseñado con el objetivo de que pudiera usarse de una manera intuitiva y flexible. El programa incluye en sus recursos un tutorial en vídeo para ir aprendiendo paso a paso cómo se puede ir diseñando y modelando el propio entorno.

SketchUp permite conceptualizar y modelar imágenes en 3D de edificios, coches, personas y cualquier objeto o artículo que imagine el diseñador o dibujante. Además el programa incluye una galería de objetos, texturas e imágenes listas para descargar. Su principal característica es poder realizar diseños en 3D de forma sencilla. Cinema 4D

Cinema 4D es un software de creación de gráficos y animación 3D desarrollado originariamente para *Commodore Amiga* por la compañía alemana *Maxon* en 1996, y portado posteriormente a plataformas *Windows* y *Macintosh* (OS 9 y OS X) [46]. Es una herramienta más enfocada a la creación de contenido tridimensional para el *post-renderizado* de video, efectos especiales para cine, televisión, publicidad, etc. Algunas de las compañías que hacen uso de este software son: *ABC, Blizzard Entertaiment, BMW, CNN, Fox, NBC, Siemens, Sony Pictures y The Walt Disney Company*.

En la figura 17 podemos ver cómo es el editor de *Cinema 4D*.

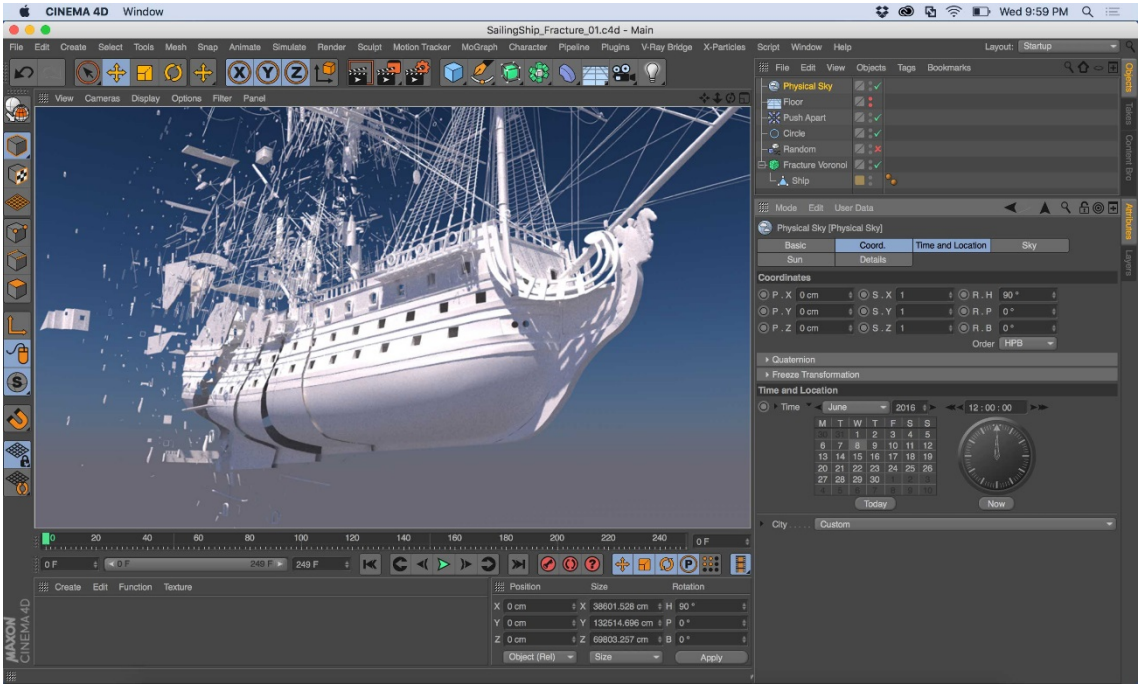

Figura 17. Editor de Cinema 4D.

Permite modelado (primitivas, *splines*, polígonos), texturizado y animación. Sus principales virtudes son una muy alta velocidad de *renderizado*, una interfaz altamente personalizable y flexible, y una curva de aprendizaje (comparado con otros programas de prestaciones profesionales similares) muy vertical; en poco tiempo se aprende mucho.

Una de las características más destacadas de *Cinema 4D* es la modularidad. Partiendo de una versión básica de *Cinema 4D* pueden añadirse módulos especializados independientes en función de las necesidades del proyecto a realizar: *Advanced Render, Dynamics, Mocca, Hair, ThinkingParticles* o *MoGraph*.

3.2.5. Comparativa

En la tabla 4 podemos ver una pequeña comparación encontrada en un artículo profesional de enseñanza entre las diferentes alternativas de software de modelado y diseño gráfico, el cual a pesar de no incluir al antes mencionado *SketchUp* nos puede ofrecer una manera sencilla de establecer los motivos que podamos tener a la hora de decantarnos por un software u otro [47].

	Autodesk 3ds max	**Autodesk Maya**	**Blender**	**Cinema 4D**
Enfoque principal	Videojuegos	Videojuegos Cine	Modelado Animación	Televisión Publicidad
Precio de la licencia	1936 € al año	1936 € al año	Gratuito	3600 € al año
Plataformas				
Curva de aprendizaje	< 2 meses	< 3 meses	< 3 meses	< 1 mes
Interfaz	Estilo CAD, sencilla y potente	Flexible y potente, no muy intuitiva	No sigue los estándares de la industria	Sencilla e intuitiva
Calidad general	Excelente	Excelente	Buena	Buena
Herramientas de animación	Muy buena	Excelente	Buena	Buena
Calidad de pintura	Buena	Muy buena	Baja	Excelente
Calidad de modelado	Excelente	Muy buena	Buena	Muy buena
Dinámica / Cuerpos rígidos	Muy buena	Excelente	Muy buena	Excelente
Cuerpos blandos	Muy buena	Muy buena	Buena	Buena
Calidad de pelo y ropa	Muy buena	Muy buena	Buena	Muy buena
Sistema de partículas	Excelente	Muy buena	Buena	Excelente
Calidad de fluidos	Muy buena	Muy buena	Muy buena	Buena
Calidad de sombras	Excelente	Excelente	Muy buena	Buena
Scripting	Buena	Excelente	Muy buena	Buena

Tabla 4. Comparativa de los distintos programas de modelado 3D.

3.3. Desarrollo de la plataforma web

Como se ha comentado previamente, este simulador incorpora una plataforma web de almacenamiento y procesamiento de datos. A pesar de no haber sido desarrollada en exclusividad para este proyecto sino que era una plataforma ya levantada de manera previa para su uso en otros proyectos anteriores del grupo de investigación de la universidad, resulta

interesante conocer cómo está creada esta aplicación web así como las alternativas existentes.

Es posible desarrollar una plataforma web tanto usando un *framework* como partiendo de un gestor de contenidos (*CMS – Content Management System*).

En el desarrollo de software, un *framework* o infraestructura digital, es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software. Un *framework* simplifica el desarrollo de las aplicaciones ya que automatiza muchos de los patrones utilizados para resolver las tareas comunes. Además, un *framework* proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. Por último, un *framework* facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas.

Por su parte, un gestor de contenidos es una aplicación (generalmente web) que proporciona la capacidad de que múltiples usuarios con diferentes niveles de permisos puedan gestionar de forma conjunta el contenido, datos o información de un sitio web, aplicación web o de una intranet. Por gestión de contenidos se entiende la capacidad de crear, editar, archivar, publicar, colaborar, informar o distribuir contenido, archivos o información de una aplicación.

Con un *framework* se dispone de mucha más flexibilidad y un mayor control sobre el proyecto, mientras que con un CMS uno tiene que adherirse a como esté estructurada la aplicación y, por decirlo de alguna manera, construir sobre ella. Esto puede llegar a limitar las posibilidades de un CMS, pero a su vez nos proporciona un enorme número de funcionalidades ya programadas (*out of the box),* lo que puede repercutir en un enorme ahorro de tiempo [48].

A la hora de seleccionar la tecnología a utilizar para el desarrollo de la plataforma web, se optó por centrarse en gestores de contenido, ya que esta plataforma iba a ser un añadido al núcleo del proyecto (el simulador) y por lo tanto debía ser algo que proporcionara el mayor contenido directamente sin tener que desarrollarlo punto por punto, lo cual hace descartar los *frameworks* dadas las explicaciones anteriormente aportadas [49].

Se decidió optar por *Drupal* frente a otros también conocidos y de uso extendido como *Liferay* o *Concrete5*.

3.3.1. Drupal

Drupal es un sistema de gestión de contenidos libre, modular, multipropósito y muy configurable que permite publicar artículos, imágenes, archivos y que también ofrece la posibilidad de otros servicios añadidos como foros, encuestas, votaciones, blogs y administración de usuarios y permisos. *Drupal* es un sistema dinámico: en lugar de almacenar sus contenidos en archivos estáticos en el sistema de ficheros del servidor de forma fija, el contenido textual de las páginas y otras configuraciones son almacenados en una base de datos y se editan utilizando un entorno Web.

Es un programa de código abierto, con licencia *GNU/GPL*, escrito en PHP, combinable con *MySQL*, desarrollado y mantenido por una activa comunidad de usuarios. Destaca por la calidad de su código y de las páginas generadas, el respeto de los estándares de la web, y un énfasis especial en la usabilidad y consistencia de todo el sistema.

El diseño de *Drupal* es especialmente idóneo para construir y gestionar comunidades en Internet, también destaca por su flexibilidad y adaptabilidad, así como la gran cantidad de

módulos adicionales disponibles, hace que sea adecuado para realizar muchos tipos diferentes de sitio web.

Actualmente *Drupal* está siendo adoptado por más del 71% de las 100 mejores universidades a nivel mundial: instituciones como *Harvard, Stanford, Cal Berkeley.* Otras instituciones importantes como la Universidad de Carolina del Norte y *Cornell* han seguido el ejemplo [50].

Estas son algunas características de *Drupal* con la que está ayudando en el proceso de estandarización en instituciones de Educación Superior:

- Contenido flexible: Puede definir campos personalizados que podrán ser utilizados tanto si se almacenan los datos de esos campos en *SQL, NoSQL* o si se utiliza almacenamiento remoto.
- Integración con otros sistemas: *Drupal* ofrece una gran variedad de servicios web, como *View Data Export, RESTful Web Services* y *Web Service Client*. *Drupal* puede configurarse como *end-point* de servicios (recibir y devolver datos a sistemas de terceros) o configurarse para llamar a otros servicios (recibir datos de aplicaciones como sistemas de gestión de eventos). Para las instituciones que publican aplicaciones móviles nativas, estas propiedades de servicios web pueden entregar contenido a las aplicaciones nativas de forma transparente. Algunas instituciones utilizan *Drupal* como fuente principal de contenido en sus aplicaciones para móviles.
- Gestión de cuentas de usuario: La gestión de usuarios web en las universidades puede ser muy compleja, a menudo con registros duplicados cuando se administran usuarios a través de múltiples sitios web y otros sistemas de información. *Drupal* puede asociar sistemas de bases de datos de usuarios, como el Directorio Activo de Microsoft o LDAP utilizando el módulo LDAP (*Lightweight Directory Access Protocol*). O bien, el módulo CAS puede utilizarse para convertir a *Drupal* en la base de datos principal de los registros de usuarios.
- Multi-sitio: *Drupal* ofrece varias maneras para configurar en una sola instalación la ejecución de varios sitios web. El enfoque multi-sitio tradicional es el más flexible, proporcionando la capacidad de almacenar contenido para cada sitio web en diferentes bases de datos y/o compartir tablas de *MySQL* como la tabla de usuarios de *Drupal*. Otro enfoque es utilizar el módulo *Domain Access*, que permite a Drupal cargar diferentes permisos de acceso a contenido y brindar diferentes configuraciones dependiendo del dominio (URL) por el que se accede.
- Interfaz de usuario responsiva: *Drupal* 8 ya cuenta con temas responsivos por defecto, ofreciendo compatibilidad con los dispositivos móviles sin necesidad de instalar un nuevo tema.
- *Testing* automático del código: Un nuevo entorno de *testing* automatizado, con más de 30.000 tests incluidos por defecto, permite realizar tests para integración continua de todos los parches al núcleo de *Drupal* y a los módulos contribuidos.

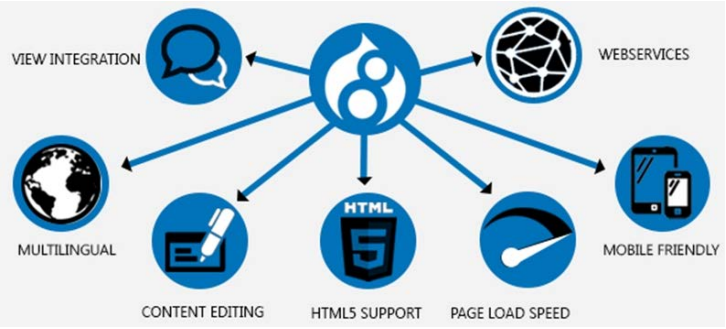

Figura 18. Características de Drupal 8.

Comentar para finalizar que *Drupal* ha intentado evolucionar en cada una de sus nuevas versiones, de tal manera que en su versión 8 ha decidido utilizar componentes de *Symfony* para su desarrollo y pretende orientar su paradigma de programación de uno estructurado a uno orientado a objetos siguiendo la línea por la que van las aplicaciones modernas.

4. Desarrollo del proyecto

A lo largo de esta sección veremos cómo se ha llevado a cabo el proyecto que nos ocupa, analizando las distintas fases por las que ha ido pasando y tratando de explicar cómo se han llevado a cabo las distintas acciones y el porqué de ellas, para lo cual se desarrollaran algunos conceptos sobre el motor Unity y sus funcionalidades. Una vez hecho esto se mostrará el resultado de la aplicación y veremos cómo es la forma de interactuar con ella, de forma que cualquier futuro usuario solo con leer esta sección fuese capaz de poder hacer uso del simulador.

4.1. Planificación

Este proyecto comenzó a llevarse a cabo en Septiembre de 2017 y desde entonces hasta la fecha de su ahora publicación en Febrero de 2018 ha ido pasando por distintas fases de desarrollo:

1. Análisis: En esta primera fase, llevada a cabo junto con el tutor de este proyecto, se trató de detallar qué se quería hacer y cómo hacerlo, determinando cuál sería la tecnología utilizada y cuáles eran los objetivos a conseguir al término de este.
2. Diseño: En esta fase se lleva a cabo el diseño del conjunto del escenario y los objetos en él contenidos.
3. Implementación: En esta fase, por momentos realizada de forma simultánea a la anterior, se lleva a cabo la programación de las funciones relativas al comportamiento del escenario.
4. Pruebas: En esta fase, llevada a cabo una vez todo lo anterior ha finalizado y tras haber sido todo testeado a nivel interno junto con el tutor, se comprueban las funcionalidades implementadas y con la ayuda de varios usuarios se toman valores de diferentes simulaciones realizadas.
5. Documentación: En esta última fase se documenta el trabajo realizado así como todo tipo de información que se considera interesante con relación a él.

4.2. Creación del escenario

El proyecto, como ya se ha mencionado en alguna ocasión, se basa en la creación de un escenario con la herramienta Unity 3D para el estudio y el aprendizaje de técnicas de conducción segura y eficiente. Para ello se desarrolla un escenario de conducción por donde conducirán los usuarios y que permite estudiar su nivel de seguridad y eficiencia en la conducción.

Este escenario estará basado en un tramo de la ciudad de Valladolid, mostrado en la figura 19, el cual a lo largo de sus 21 kilómetros de longitud combina tramos urbanos con tramos de autovía, teniendo tanto su salida como su meta la Escuela Técnica Superior de Ingenieros de Telecomunicaciones.

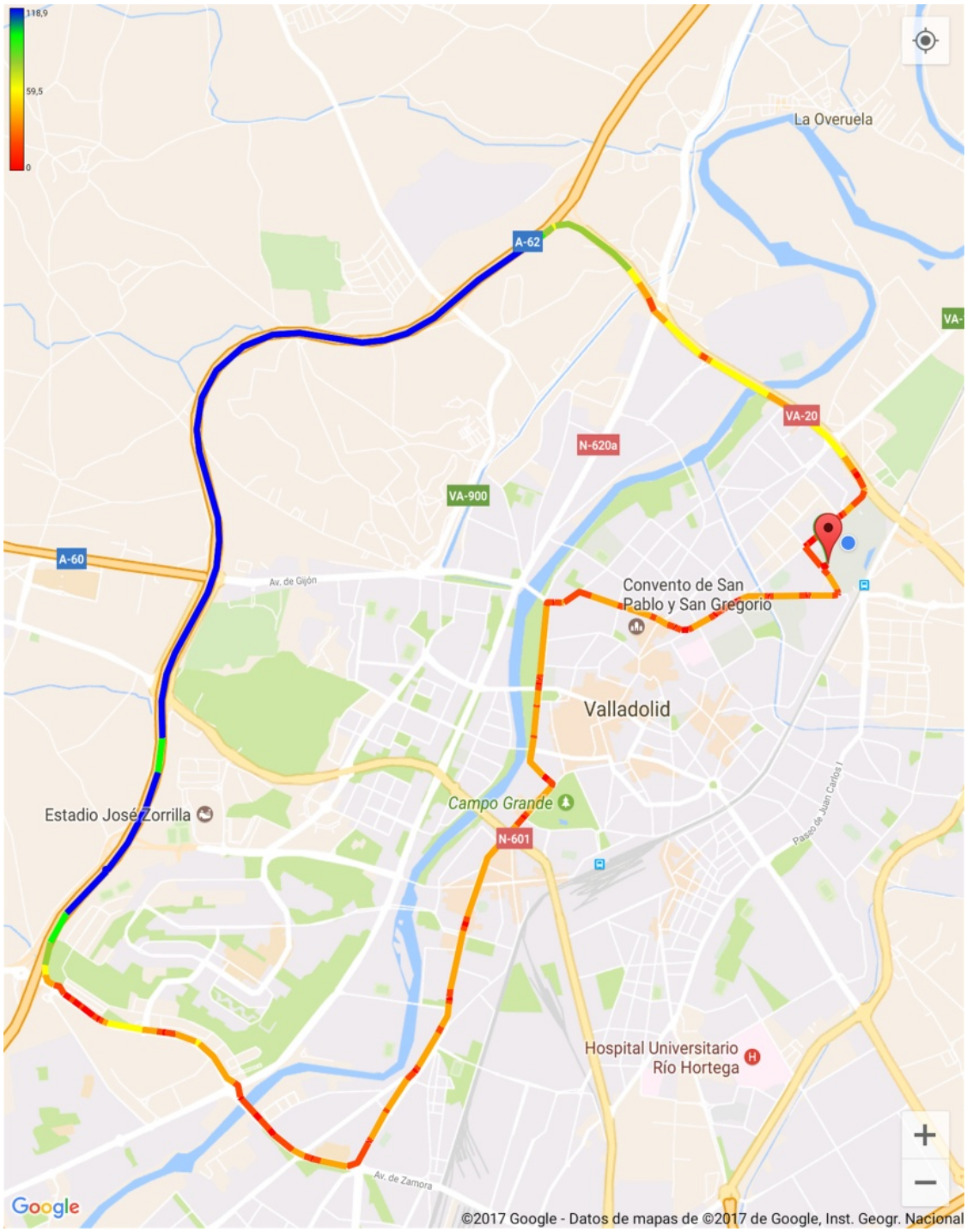

Figura 19. Imagen de Google Maps con la ruta de la simulación.

Para ello se reproducirá esta zona de la ciudad con los diferentes cruces de carreteras por donde el usuario deberá conducir, habiéndose puesto el foco en reproducir lo más fielmente posible todas las distancias y longitudes de las distintas carreteras así como su número de carriles, y no tanto en los elementos estéticos ajenos a las vías por las que se circula, ya que el tiempo era limitado y se consideró como algo de menor prioridad. En esta ruta, se reproducirán los diferentes elementos que podemos encontrar durante la conducción, como las diferentes indicaciones viales (como pueden ser señales o semáforos) o los diferentes usuarios que transitan la vía (como ciclistas, automóviles o autobuses) o peatones.

El usuario tendrá además la opción de realizar una ruta guiada. Con esta opción, al acercarse a una intersección se mostrará el camino a seguir.

A la hora de implementar este recorrido en nuestro proyecto en *Unity*, encontramos como gran inconveniente que la herramienta no permita importar mapas de forma real, algo de gran importancia para escenarios de este tipo. Dado que se buscaba una gran precisión en cuanto a las distancias, se optó por utilizar no una sino varias imágenes del recorrido extraídas con *Google Maps,* y situarlas de forma correcta como texturas de *GameObjects* llamados *Planes*. En la figura 20 podemos ver el resultado de nuestro escenario tras aplicar esta técnica.

Figura 20. Planos situados sobre el escenario para crear el recorrido.

Una vez plasmado el recorrido a seguir sobre nuestro proyecto, el siguiente paso fue comenzar con la creación de las carreteras. Para ello se contemplaron dos posibilidades, el *asset Easy Roads* y el *asset Road Architect System*, de los que se habla a continuación en el apartado relativo a los *assets* empleados.

Una vez realizadas las carreteras, el siguiente paso fue proceder a situar los semáforos y señales que deberían controlar posteriormente el tráfico. Para ello se utilizó el *asset* gratuito

que incluye *Unity* de *Environment*, además de algunas señales pertenecientes al anteriormente mencionado *Road Architect*, algunas del *asset Traffic System* e imágenes descargadas de la red.

Una vez finalizado el entorno, se incluyeron el tráfico con la ayuda del *asset iTS (Intelligent Traffic Sytem)* y los peatones con ayuda del *asset Pedestrian System*. La realización del coche y de todo aquello concerniente a él se realizó con el *asset Realistic Car Controller V3*.

4.3. Desarrollo del escenario y assets empleados

En la sección anterior se ha explicado de forma breve cómo se ha realizado el escenario de este proyecto. En este capítulo se explicará con mayor detalle cómo son los *assets* utilizados, las posibilidades que ofrecen, para qué han sido empleados y cómo se trabaja con ellos.

4.3.1. Creación de las carreteras, Road Architect.

Para la creación de las carreteras del escenario se ha utilizado el *asset Road Architect*. Este *asset* en la actualidad ya no está disponible en el *Asset Store*, ya que quedó obsoleto. Esto significa que no se permiten nuevas compras del paquete y que solo los usuarios que ya compraron o descargaron el paquete antes de que se desaprobara pueden descargarlo. En la mayoría de los casos, la desaparición del paquete ocurre porque el editor ya no puede o no quiere soportar el paquete [51].

El propio *Unity* sugiere buscar paquetes alternativos, pero dado que disponíamos del *Road Architect*, hemos decidido hacer uso de él por los motivos que a continuación se exponen [52].

El primero y fundamental es que es muy intuitivo, y su curva de aprendizaje es corta. La forma de crear una carretera una vez está el *asset* implementado es la siguiente: seleccionamos el *GameObject* del sistema *Road Architect* recién creado y hacemos clic en el botón "*Add Road*" (Agregar carretera), ubicado en la ventana del inspector del sistema Road Architect que podemos ver en la figura número 21. A continuación, mientras se selecciona un objeto de carretera, se agregan los nodos de esta en el terreno mientras mantenemos presionado ctrl y haciendo clic con el botón izquierdo en la ubicación del terreno. En la figura 22 podemos ver el resultado de crear una carretera y situar 2 nuevos nodos. Con tan solo seguir estos pasos podemos realizar una carretera, una solución más rápida que la de algunas de sus alternativas y que dado que nuestro recorrido es de gran extensión y disponemos de poco tiempo, ha hecho que nos hayamos decantado por este *asset*.

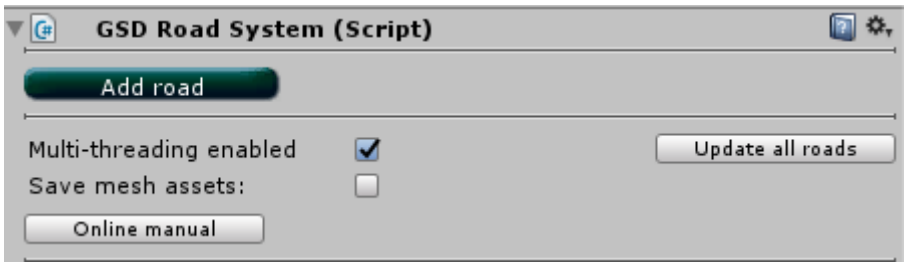

Figura 21. Interfaz de Road Architect para la creación de una nueva carretera.

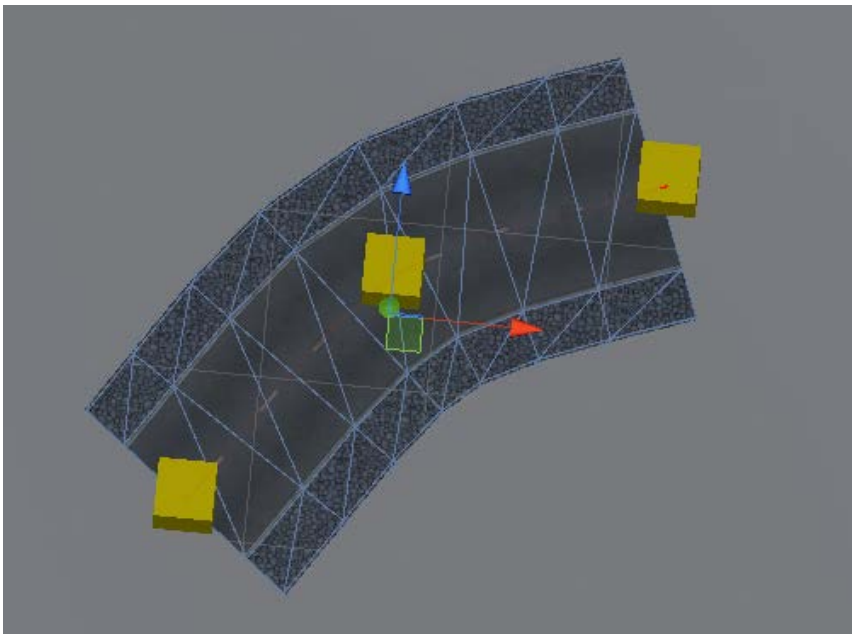

Figura 22. Fragmento de carretera creado con Road Architect.

Además de permitirnos una rápida creación de carreteras, este *asset* dispone de otras características que nos son de gran utilidad como son la creación de puentes y la inclusión de objetos en medio o a los lados de la carretera.

Para crear un puente debemos seleccionar el nodo de la carretera donde el puente debe comenzar. En la parte superior de la ventana del inspector del nodo, marcamos éste como "*Bridge Start*" (Inicio del puente). A continuación seleccionamos el nodo de la carretera donde finalizará el puente y lo marcamos como "*Bridge End*" (Puente final). Tras ello seleccionando el nodo de inicio hacemos clic en "*Open Wizard*" (Abrir Asistente) y accedemos de esta manera a una gran posibilidad de puentes de gran calidad como podemos ver en la figura 23.

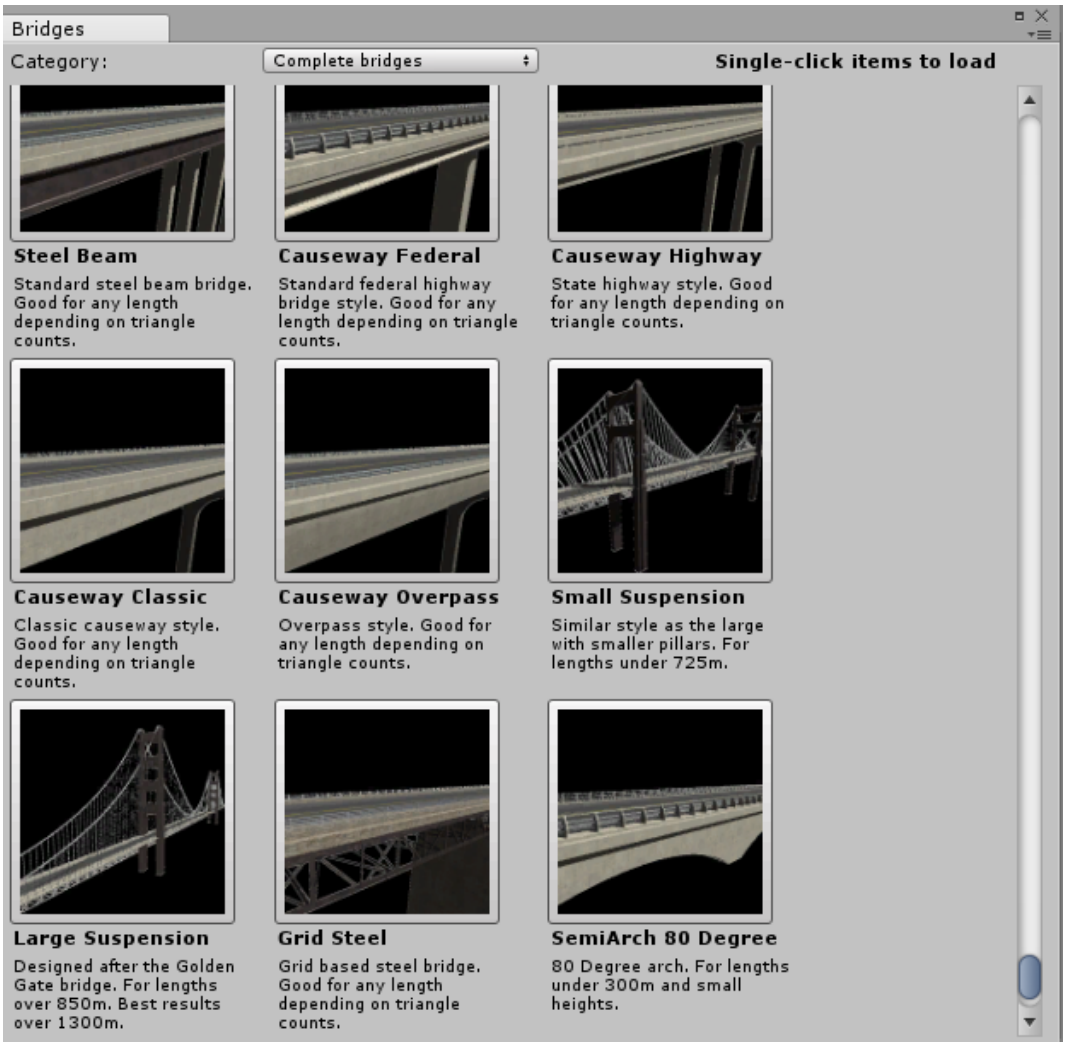

Figura 23. Puentes ofrecidos por Road Architect.

Para incluir objetos en la carretera, deberemos marcar el nodo donde queremos que se sitúen, y al igual que hiciésemos para poner un puente, hacer clic en "*Open Wizard*", y seleccionando a continuación "*Extrusion Objects*" (Objetos prominentes) o "*Edge Objects*" (Objetos al borde). De esta manera podremos añadir de una forma sencilla elementos como guardarraíles, vallas, aceras, barreras o farolas entre otros.

Como grandes desventajas tenemos la creación de intersecciones, rotondas y el número de carriles con los que contará nuestra carretera. La creación de intersecciones se postulaba como algo sencillo y una de sus grandes funcionalidades, bastaría con arrastrar el nodo de una carretera sobre el nodo de otra distinta para que éstas se fusionaran dando lugar a una nueva intersección; sin embargo, el modo en que esto se produce no ha sido como se esperaba, siendo queja de muchos usuarios y se cree también que uno de los motivos de la desaparición del paquete, ya que produce una intersección que altera las carreteras afectadas, de estética cuestionable y con poca variedad de intersecciones. Para la realización de intersecciones se optó por superponer de forma manual dos carreteras, editando a mano la zona de la superposición.

En cuanto a las rotondas, el *asset* directamente no las incluía, habiendo tenido que buscarlas en otros paquetes para su incorporación al escenario. Por último, su mayor desventaja o al menos en lo relativo a este proyecto, es la cantidad de carriles con que se puede crear una

carretera: 2, 4 o 6; lo cual ha obligado a tener que llevar modificaciones poco estéticas para los casos en que se necesitaba carreteras de 1 o 3 carriles.

A pesar de las desventajas mencionadas, el hecho de que estas afecten principalmente a la parte estética del proyecto, algo que como se ha explicado no es prioritario, y el hecho de que sea un paquete que permita una implementación rápida, han llevado a que sea el *asset* escogido. En cualquier caso, se entiende que si está ya obsoleto y no disponible en *Unity* es por algo, y de tratarse de un proyecto que requiera mayores soluciones estéticas o para el que el tiempo no fuese tan limitado o el recorrido tan largo, se optaría por otro distinto como pudiera ser *Easy Roads*, descartado para este proyecto [53].

4.3.2. Implementación del tráfico, iTS

Intelligent Traffic System es una aplicación desarrollada para *Unity3D* que simula tráfico. Este sistema controla los vehículos para simular tráfico en un área definida. Dado que el grupo de investigación GTI con el que se ha realizado el proyecto ya había adquirido de manera previa este *asset*, cuyo coste es de 70€, se ha decidido que sea el encargado de gestionar nuestro tráfico.

El *asset*, además de gestionar el tráfico, nos permite crear nuevos vehículos, funcionalidad que no hemos utilizado ya que incluye algunos básicos por defecto, cuyo funcionamiento es correcto pero de estética básica aunque suficiente para el propósito de este proyecto.

Controlado por el script *TSMain Manager*, el *asset* presenta el aspecto que podemos ver en la figura 24. Para agregar una nueva línea de tráfico, deberemos seleccionar el botón "*Lines*" (Líneas), con la pestaña secundaria "*Line*" y la opción "*Create*" (Crear) como se muestra en la figura antes mencionada. Tras ello, basta con situar el ratón donde desea iniciar el carril y presionar el botón izquierdo, y arrastrar hasta el punto donde desea que la línea finalice.

Figura 24. Interfaz del script TS Main Manager.

La forma de eliminar estas líneas de tráfico también es bastante sencilla, basta con seleccionar la pestaña "*Lines*", con la pestaña secundaria "*Line*" y la opción "*Remove*" (Eliminar) y hacer clic con el botón izquierdo en el carril que desea quitar. Se pueden también enlazar líneas entre sí, esto es útil para posibilitar los adelantamientos y el cambio de carril. Para crear la unión entre dichas líneas, deberemos seleccionar la pestaña "*Line Connectors*" (Conectores de líneas) y la pestaña secundaria "*Connector*" y seleccionar la opción "Create". En la figura 25 podemos ver cómo se ha generado el tráfico en una intersección haciendo uso de estas opciones.

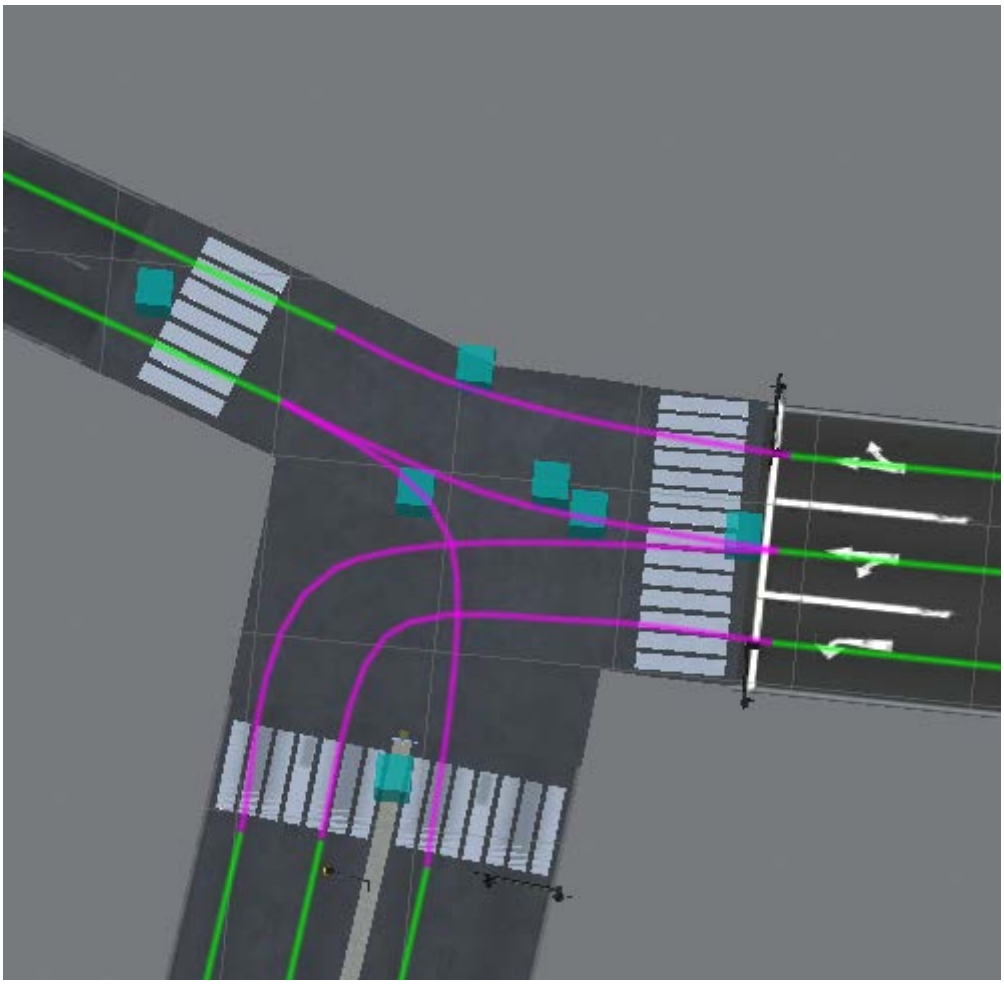

Figura 25. Diseño del tráfico con iTS.

Una vez creado nuestro tráfico, este *asset* nos permite controlar en cada línea qué tipo de vehículos habrá (coches, autobuses, bicis …), la velocidad máxima a la que irán, la densidad de tráfico que habrá, cómo deben hacer el cambio de línea, etc., además de alguna funcionalidad añadida que no se ha utilizado como agregar semáforos a los objetos del juego para controlar el flujo de tráfico en algunos puntos seleccionados de un carril o conectores, útil para controlar el tráfico en cruces y algunos otros puntos en el sistema de tráfico, pero que como se decía se ha decidido no emplear y por contra gestionar los semáforos de otra forma que más adelante se explica.

Es por estas características mencionadas un *asset* muy interesante y fácil de emplear y que hace valer el precio que su adquisición cuesta, pero también se han encontrado algunas limitaciones. El sistema de tráfico viene con una física para los vehículos aparentemente fácil de configurar y de código abierto, por lo que podría modificarse para adaptarla a las distintas necesidades que uno pueda tener, pero es aquí donde han surgido las limitaciones. El *script* que controla esta configuración, llamado *TSSimpleCar*, viene con una configuración por defecto de velocidad máxima 50 km/h, 2500 revoluciones por minuto, fuerza del motor de 80 así como otras muchas características. Como se ha dicho, este *script* es de código abierto, pero lograr la funcionalidad de que los vehículos tengan velocidad máxima de 120 km/h que para este proyecto se necesitaba, no es tan sencillo como cabe imaginar, ya que no vale con modificar el parámetro mencionado que lo controla, sino que en los cálculos relativos al

funcionamiento del coche hay otros muchos parámetros que influyen y que para nuestra desgracia no vienen explicados de una forma clara. De hecho, para tratar de lograr el citado objetivo de una velocidad máxima superior tuvimos que ponernos en contacto con el desarrollador del *asset* el cual nos contestó en repetidas ocasiones. En estas conversaciones nos habló de cómo, aparte del parámetro de M*ax Speed* (máxima velocidad), intervienen otros como *Max Aceleration* (máxima acelaración), los valores de *Stiffness* de los *wheelcolliders* (que hacen referencia a la rigidez de giro en las ruedas), el valor de *body drag (*arrastre corporal, a mayor velocidad más tarda en acelerar), el valor *motor torque* (fuerza del motor) o el valor *brake torque* (fuerza del motor). Finalmente se consiguió el objetivo deseado y los coches del escenario alcanzan velocidades de hasta 130km/h en tramos de autovía, sin embargo, el comportamiento de estos deja en ocasiones que desear, dado que tienen un comportamiento aleatorio que es casi imposible de controlar, haciendo que, por ejemplo, circulen a velocidades muy bajas en algunas ocasiones o que incluso lleguen a pararse eventualmente.

Podríamos decir por tanto que es un *asset* muy fácil de emplear y con muchas funcionalidades de gran utilidad aunque con algunas limitaciones.[54].

Para este proyecto se probó como alternativa el *asset Road & Traffic System*. Este *asset* permite no solo la creación del tráfico sino también de las carreteras y los peatones, permitiendo unificar en uno solo los tres utilizados en este proyecto. Además, tras las pruebas que se hicieron, las posibilidades que ofrece están dotadas de una mayor calidad, especialmente en lo relativo a las carreteras, donde ya pudimos ver los inconvenientes que el *asset Road Architect* tenía. El inconveniente de este *asset* es que su curva de aprendizaje es mucho más elevada, y una vez que se domina su empleo no permite una agilidad tan elevada como la de *iTS*. No obstante, para proyectos donde el recorrido de carreteras y tráfico no sea elevado o proyectos en los que se disponga de más tiempo y se quiera un toque más profesional, se recomienda su uso frente a *iTS* [55].

4.3.3. Implementación de los peatones, Pedestrian System.

Para la implementación de los peatones se ha utilizado el *asset Pedestrian System*. Este sistema, de uso gratuito, fue desarrollado por el *asset Road & Traffic System*, ya antes mencionado y con funcionalidades completas. De hecho la funcionalidad de peatones es la misma para ambos *assets*, y dado que decidimos no emplear este último para el tráfico, nos bastará con utilizar *Pedestrian System*, ahorrando así los 20€ que cuesta *Road & Traffic System*.

El método de empleo es muy sencillo. Una vez se ha importado en el proyecto, el aspecto que presenta su configuración es el que vemos en la figura 26.

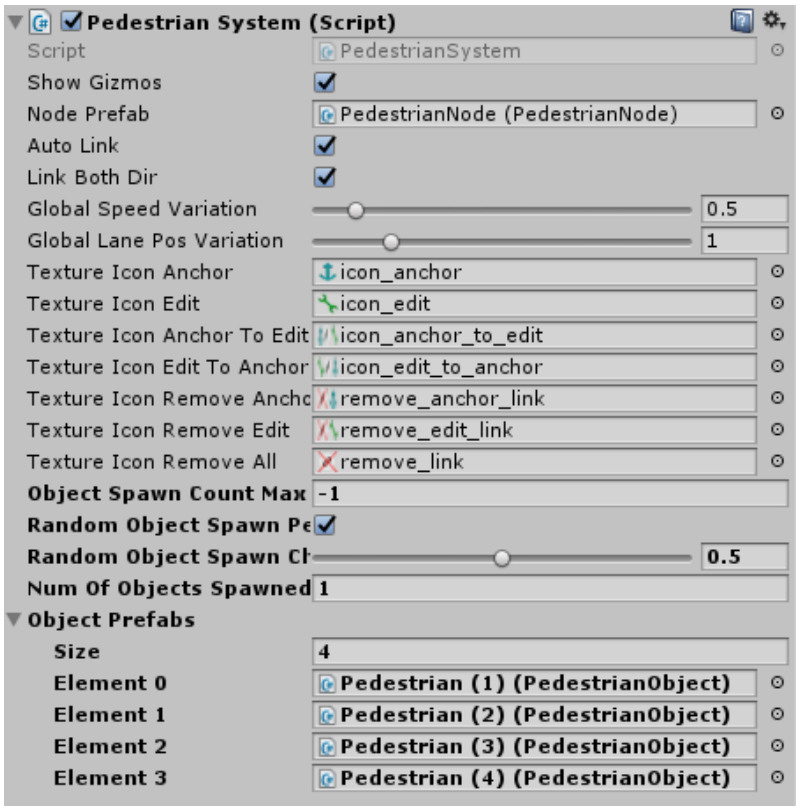

Figura 26. Interfaz del script PedestianSystem.

Para crear una línea de tráfico de peatones, bastará con que vayamos situando distintos nodos con el "*Generate Node*" (Generar nodo) y uniéndolos entre sí. Además, para que su uso sea más sencillo, se tienen las opciones para que, de forma automática, cada nodo se una con el anterior "*Auto Link*" y hacer que entre ambos los peatones circulen en ambos sentidos "*Link Both Dir*", aunque esto se puede hacer también de forma manual indicando a los peatones entre qué nodos queremos que circulen y en qué sentidos. Además el *asset* permite la opción de controlar la cantidad de peatones que queremos que circulen por cada trayecto y la aleatoriedad de sus movimientos. En la figura 27 podemos ver de blanco unas de estas líneas de tráfico de peatones sobre el proyecto.

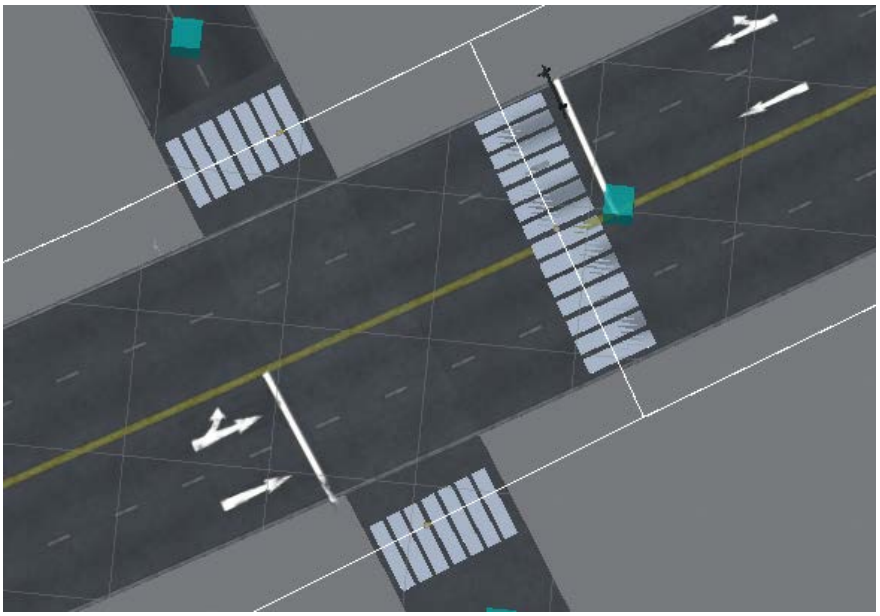

Figura 27. Tráfico de peatones creado con PedestrianSystem.

En cuanto a los peatones implementados, se han obtenido del asset gratuito de Environment ya antes mencionado. En la figura 26 anteriormente mostrada, podemos ver que se han incluido hasta 4 modelos de peatones distintos.

Se puede concluir con que es un *asset* muy sencillo de utilizar, pero, dado que no lo hemos integrado junto al resto del *Road & Traffic System*, ha presentado algún problema a la hora de convivir con *RoadArchitect* para la gestión común de tráfico y peatones en los semáforos. De la forma en que esta gestión se ha realizado y los inconvenientes acarreados, se hablará en la siguiente sección.

4.3.4. Creación del entorno

En esta sección se explicará cómo se han creado el resto de elementos que aún no se han mencionado, además del "sistema GPS" que guía al conductor y la forma en que se gestionan los semáforos y su interacción con vehículos y peatones.

Algunos de estos elementos son los siguientes:

- Señales en el suelo: como pueden ser las flechas que en la figura 27 aparecen, o las de bus o intersecciones que también encontramos. Estas han sido incorporadas gracias al *asset LowPolyStreetPack*, de descarga gratuita.
- Pasos de cebra, árboles y algunas señales: tomados del *asset Environment*.
- Rotondas: tomadas también de *LowPolyStreetPack,* ya que el *asset* empleado para la realización de las carreteras no las incluía.
- Semáforos: tomados del *asset iTS.*

Una vez implementado todo el escenario, para que el usuario pueda desenvolverse por él de manera adecuada siguiendo la trayectoria que queremos estudiar, deberemos guiarlo a través de él. Para ello, se ha implementado un sistema que trata de simular el de un GPS, colocando distintos sensores a lo largo del escenario los cuales, al pasar el usuario por ellos, hacen que se le indique cómo debe continuar a través de una de las imágenes de la figura 28, la cual

aparecerá en la parte superior de la pantalla durante la ejecución, ayudando así al usuario a mantener el rumbo correcto.

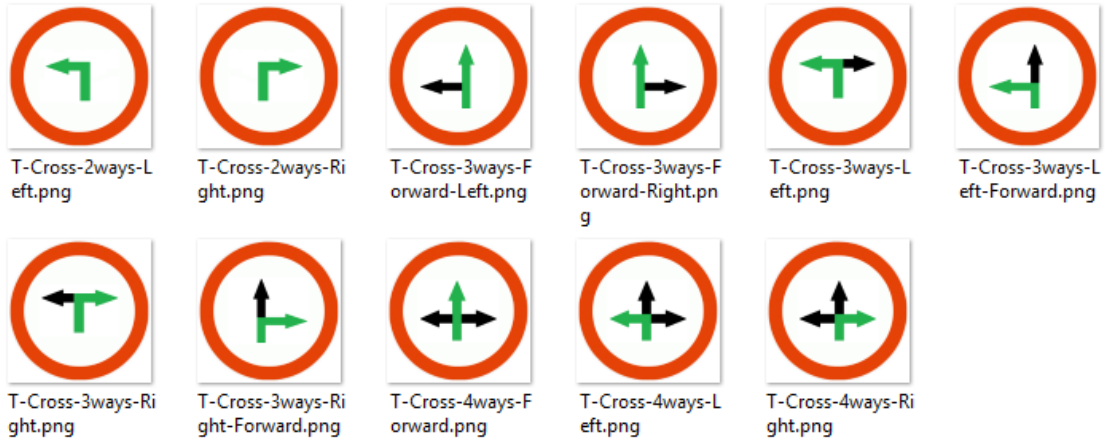

Figura 28. Imágenes utilizadas para indicar al conductor la ruta a seguir.

Quedaría por tanto explicar cómo se gestionan los semáforos y la forma en que vehículos y peatones responden a ellos. Para que su compresión sea más fácil nos apoyamos en la figura 29. En ella se puede ver el ejemplo de un paso de peatones controlado por un semáforo y, hasta 4 sensores que intervienen en su funcionamiento.

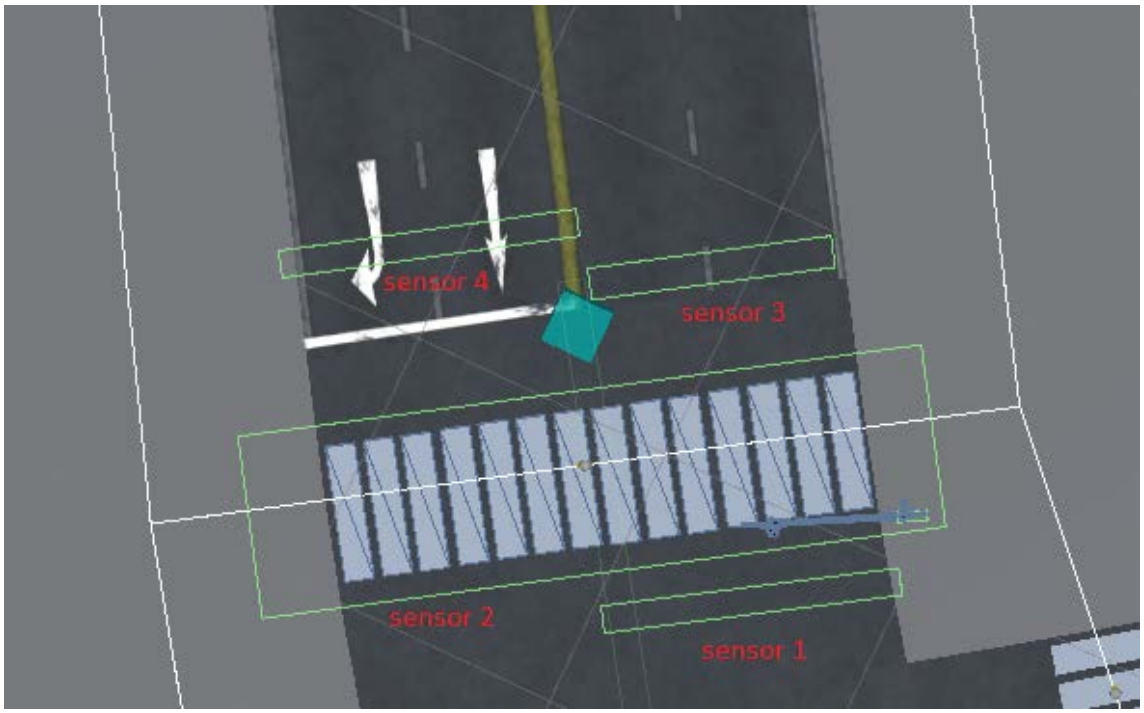

Figura 29. Sensores que regulan el tráfico en los semáforos.

El funcionamiento, diseñado a partir de numerosos *scripts*, es el siguiente: cuando los vehículos que circulan en el mismo sentido al nuestro y, por tanto, a los que afecta el semáforo, llegan al sensor 1, evalúan si este está en verde para poder continuar. Por su parte, los vehículos que vienen en dirección contraria, y a los que el semáforo no afecta, al entrar al

sensor 4 evaluarán si hay peatones sobre el sensor 2 y, de no haberlos, cruzarán. Los peatones, siempre en movimiento, al entrar al sensor 2 evaluarán si pueden pasar o no en función también de cómo esté el semáforo. El sensor 3 servirá para medir si nuestro vehículo pasa por él con semáforo en rojo, habiendo cometido en esos casos una infracción.

Leyendo el párrafo anterior, uno entiende que no se ha dicho nada que no sea el funcionamiento cotidiano de un cruce, pero el problema surge en la forma en que los peatones, recordemos que son del *asset Road & Traffic System*, responden a esta lógica, diseñada con los vehículos de *iTS*, ya que lo que los peatones hacen al llegar al llamado sensor 2 cuando el semáforo está en verde para los vehículos no es parar de moverse, sino moverse en el sitio sin avanzar al siguiente nodo, lo cual hace que a veces se desplacen en círculo y que un vehículo que pase cerca los pueda atropellar. La forma de solventar este incidente fue alejar los nodos de peatones de los cruces y ampliar el sensor 2. Sin embargo, esto no otorga un efecto 100% real ya que los peatones no esperan junto al semáforo sino a dos metros de él. Para lograr una solución igual de efectiva pero de mayor calidad estética se recomienda de nuevo implementar todo con un único *asset* en caso de tener que realizar un trayecto de menor longitud o de disponer del tiempo adecuado.

4.3.5. Desarrollo del jugador, Realistic Car Controller V3.

El jugador tendrá como principal vehículo el modelo de Lexus GS 450h. Las especificaciones de los diferentes acabados de esta gama se pueden ver en la figura 30 [56].

Figura 30. Diferentes modelos de la gama Lexus GS 450h.

El modelo del vehículo en Unity que usará el jugador se puede ver en la figura 31.

Figura 31. Modelo de la gama Lexus GS 450h que usará el jugador.

Este modelo implementará las características del vehículo mostradas en la tabla 5 [57]:

Característica	Especificación
Máxima potencia (CV)	345
Tracción	Delantera
Cilindrada (CC)	3456
Tipo de combustible	Gasolina
Consumo urbano (Litros/100 km)	6.7
Consumo en carretera (Litros/100 km)	5.5
Consumo medio (Litros/100 km)	6.1
CO	266.6
HC	28.9
Emisiones de NO_X (g/km)	0.017
Nivel de emisiones	Euro 6
CO_2 ciclo urbano (g/km)	156
CO_2 carretera (g/km)	129
Emisiones de CO_2 (g/km)	141
Velocidad máxima (km/h)	250
Aceleración 0-100 km/h (s)	5.9
Coeficiente de resistencia aerodinámico (CD)	0.27
Peso vacío (kg)	1920
Peso máximo admisible (kg)	2335

Tabla 5. Especificaciones del vehículo Lexus GS 450h usado en el proyecto.

Configuraremos estos valores en las variables del *script* del *Asset Realistic Car Controller* (*RCCv3*), de forma que adaptemos el comportamiento del vehículo al del Lexus GS 450h. De todos los *scripts* de este *asset*, *RealisticCarControllerV3.cs* es el principal. En este script, tal y como podemos apreciar en la figura 32, tendremos 8 categorías principales: ruedas, volante, suspensión, configuración mecánica, estabilidad, luces, sonidos y daños.

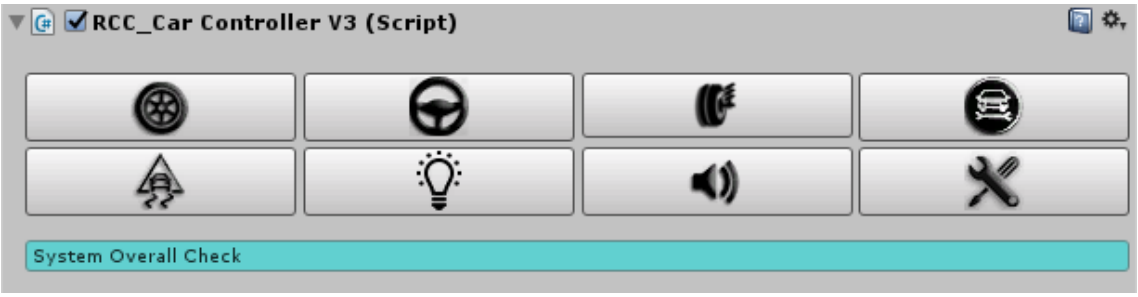

Figura 32. Interfaz del script RCC_Car Controller V3.

Todos los vehículos comparten una configuración global almacenada en *RRC Settings*. Una vez configurada y adaptada la apariencia de nuestro vehículo, procedemos a configurar las características del vehículo en el *RCC*. Una vez hecho esto, se modificará el comportamiento del *script*, que contiene el comportamiento del motor del vehículo así como las marchas, revoluciones o manejo, para adaptarlo a las siguientes funcionalidades de nuestro proyecto:

- Comportamiento de los intermitentes: El script deberá desarrollar el comportamiento de los indicadores de dirección del vehículo. En nuestro caso bastará con activar o desactivar los intermitentes dependiendo de qué dirección hayamos seleccionado, por lo que en nuestro *script* detallaremos qué hacer cuando esté activado uno u otro y qué hacer cuando no esté activado ninguno. Hay que añadir además una fuente de sonido y la pista de audio correspondiente al sonido de los indicadores, de forma que cuando se activen, tengamos una notificación sonora de estos.
- Comportamiento del cambio de marchas: el escenario ofrece 3 tipos de cambio de marchas, de forma automática, manual o por levas. Para ello habrá que ajustar el *script* para cada uno de los tipos. Para el automático, que no hace uso del embrague, las marchas deberán aumentarse o reducirse de la misma forma que funcionaría un vehículo real automático. Para cambio manual, el conductor necesitará pisar el embrague e introducir la marcha deseada, teniendo en cuenta además el funcionamiento lógico de un coche real, donde no podemos por ejemplo pasar de cuarta a marcha atrás. Por último, para realizar cambio de marchas por levas, el conductor utilizará las levas del volante para poder incrementar o reducir las marchas. En la figura 33 podemos ver un fragmento del código utilizado para los cambios de marcha manual. En este caso, se comprobará que el usuario haya metido la marcha a la vez que presionaba el pedal del embrague. En caso de que quiera meter marcha atrás o primera marcha a una velocidad elevada, se le notificará que debe reducir primero la velocidad.

```
// Modo manual
else if (drivePref == 1){
    if (Input.GetButtonDown("Primera") && (Input.GetAxis("Embrague")<-0.5)){
        if (speed < 20)
            StartCoroutine("ChangingGear", 0);
        else
            textUIController.ShowInfo("No se puede cambiar a Primera a tanta velocidad", (float)1);
    }
    else if (Input.GetButtonDown("Segunda") && (Input.GetAxis("Embrague")<-0.5))
        StartCoroutine("ChangingGear", 1);

    else if (Input.GetButtonDown("Tercera") && (Input.GetAxis("Embrague")<-0.5))
        StartCoroutine("ChangingGear", 2);
    else if (Input.GetButtonDown("Cuarta") && (Input.GetAxis("Embrague")<-0.5))
        StartCoroutine("ChangingGear", 3);
    else if (Input.GetButtonDown("Quinta") && (Input.GetAxis("Embrague")<-0.5))
        StartCoroutine("ChangingGear", 4);
    else if (Input.GetButtonDown("MAtras") && (Input.GetAxis("Embrague")<-0.5)){
        if (speed < 5)
            StartCoroutine("ChangingGear", -1);
        else
            textUIController.ShowInfo("No se puede cambiar a Marcha atras a tanta velocidad", (float)1);
    }
}
```

Figura 33. Fragmento de código para la configuración de marchas manuales.

- Comportamiento del motor: deberá ser similar al de un vehículo real, asignando la velocidad y aceleración del vehículo haciendo intervenir factores como la fuerza del motor, la velocidad y marcha actual, el sonido del motor o si el usuario está realizando alguna acción como pudiera ser frenar o acelerar.

Por otra parte, el consumo del coche vendrá calculado por el *script consumo.cs*. Este cálculo dependerá del tipo de marcha y velocidad que tenga el vehículo. En la figura 34 se puede ver una gráfica que muestra el consumo con relación a la velocidad [58].

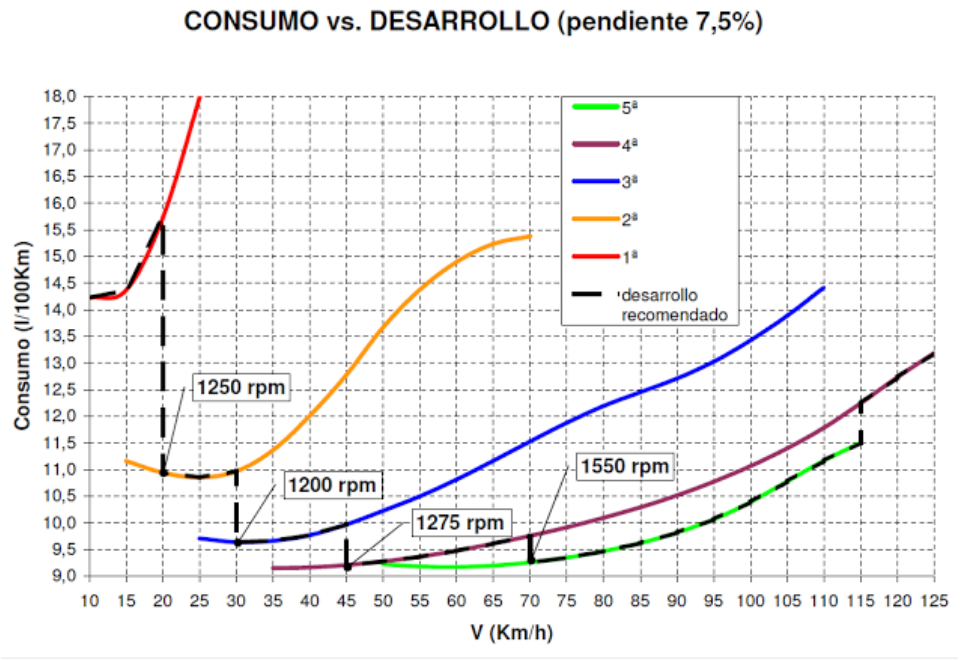

Figura 34. Consumo con relación a la velocidad.

En la gráfica están marcadas líneas discontinuas que indican cuál sería la velocidad ideal para cambiar de marcha de manera más eficiente con la que, por lo tanto, gastaríamos menos combustible. Estas curvas de consumo serán las que tratemos de representar en el *script*. Cada

marcha se puede representarse como una ecuación matemática que, en nuestro caso, haremos lineal, ya que de esta forma es más fácil de calcular y reflejan el consumo que necesitamos calcular. Para ello se introduce un apartado en el *script* donde calculamos el consumo dependiendo de la marcha actual del coche.

Por último, es interesante mencionar la cámara interior del vehículo, que será la que trate de proporcionarnos esa sensación de estar en él. Esta cámara (*Hood camera*) está situada en la posición del conductor del vehículo. Desde la cámara, tal y como se aprecia en la figura 35, podremos observar tanto la carretera como la interfaz *HUD (*visualización cabeza-arriba) del vehículo, la cual nos mostrará en todo momento los diferentes indicadores necesarios para poder informar al usuario de las variables del coche durante la conducción.

Figura 35. Visión desde la cámara en la posición del conductor.

Para realizar los retrovisores se aplica al espejo la textura deseada y, una vez asignada, se selecciona la cámara que se quiere mostrar en él, indicando en el parámetro "*Target Texture"* la textura de *renderizado* que se aplicó previamente. De esta forma, se consigue que en el espejo se muestre la visión de la cámara. Dado que el vehículo no tiene espejos independientes, se ha optado por agregar un plano en el espejo donde se proyecta la cámara.

4.3.6. Sistema de infracciones

Como ya se ha mencionado, la finalidad de este proyecto es didáctica, y a través de él se trata de concienciar a los usuarios de las ventajas que una conducción responsable tiene frente a una que no lo es. Una forma de lograr este resultado es analizando qué hace mal el conductor durante la simulación, y, para ello, se han creado una serie de infracciones que el simulador controla cuando se cometen. Estas infracciones pueden verse en la figura 35, la cual muestra una parte de la pantalla que el jugador puede ver al pausar la simulación.

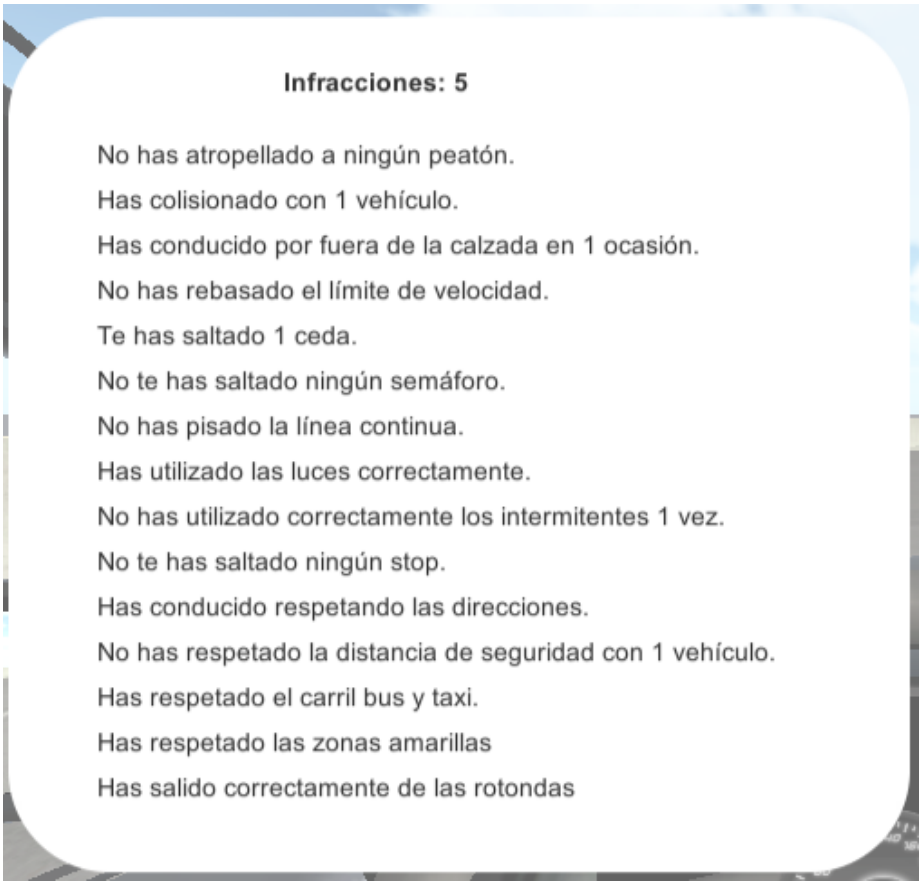

Figura 36. Infracciones en la conducción controladas por el simulador.

A continuación se explicará el funcionamiento e implementación de cada una de estas infracciones.

- Atropellar a peatones: los peatones tendrán un "*Collider*" que detecte cuándo ha colisionado el conductor contra ellos añadiendo de esta forma una infracción de este tipo al resultado final.
- Colisionar con vehículos: misma lógica que la infracción anterior aunque esta vez aplicada al resto de vehículos del escenario.
- Conducir por fuera de la calzada: se han colocado sensores fuera de las carreteras que detecten cuándo se sale el conductor.
- Rebasar el límite de velocidad: se incluye un *script* a nuestro coche que detecte la velocidad a la que circula en todo momento. Se ha decidido que cuente como infracción una vez cada 7 segundos, ya que si no se limita de alguna manera este hecho, el conducir por encima de la velocidad durante 10 segundos podría llegar a contar como cientos de infracciones.
- Ceda el paso: se ha implementado un sistema que hace uso de tres sensores, como el de la figura 36, donde podemos ver uno de los ceda el paso que controlan la incorporación a la autovía.

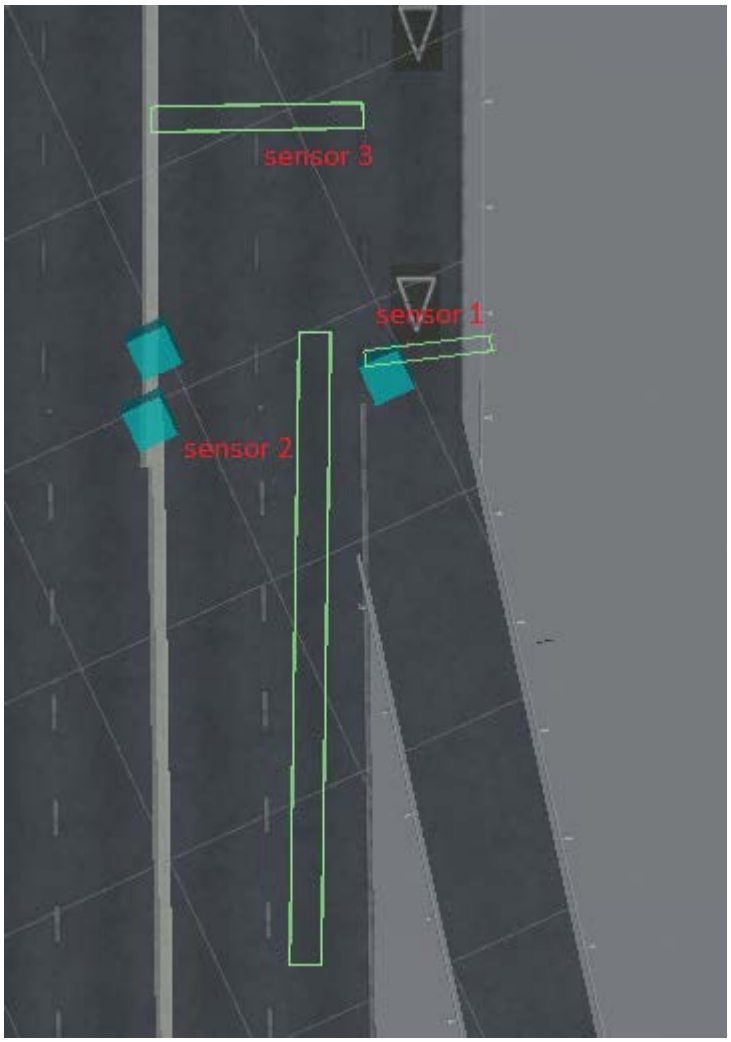

Figura 37. Sensores que intervienen en el control de los ceda el paso.

Cuando el conductor entra al sensor 1, se comprueba si hay algún vehículo en la zona del sensor 2. De ser así, cuando el conductor pase por la zona del sensor 3, acumulará una infracción de este tipo.

- Saltarse un semáforo: controlado por dos sensores como ya se viera en la sección correspondiente al entorno donde se trató este tema, añadirá una infracción de este tipo cada vez que el conductor pase un semáforo en rojo.
- Pisar la línea continua: este tipo de líneas llevarán un sensor que detecten cada vez que el conductor atraviesa una de ellas.
- Utilizar las luces correctamente: no utilizada finalmente en este proyecto dado que no se ha implementado el hecho de que pueda anochecer mientras el usuario conduce, está diseñada para detectar cuándo el usuario no activa las luces habiendo poca luz en el escenario.
- Utilizar incorrectamente los intermitentes: en este caso, tal y como se puede apreciar en la figura 38, se dispone de hasta 4 sensores por cada entrada a una intersección. El sensor 1 será el encargado de detectar al jugador cuando se acerque a la intersección. Este sensor notificara a los otros 3 sensores si el jugador tiene activado algún

intermitente, y en caso afirmativo, qué intermitente es el que está activado. Los sensores 2, 3 y 4 identifican las posibles salidas de la intersección: izquierda, recto y derecha. Cuando el usuario entre en alguna de ellas, gracias a la información proporcionada por el sensor 1, se podrá notificar de un mal uso de los intermitentes por tener activado el que no corresponde o de no activar el intermitente requerido en caso de que no estuviera ninguno activado.

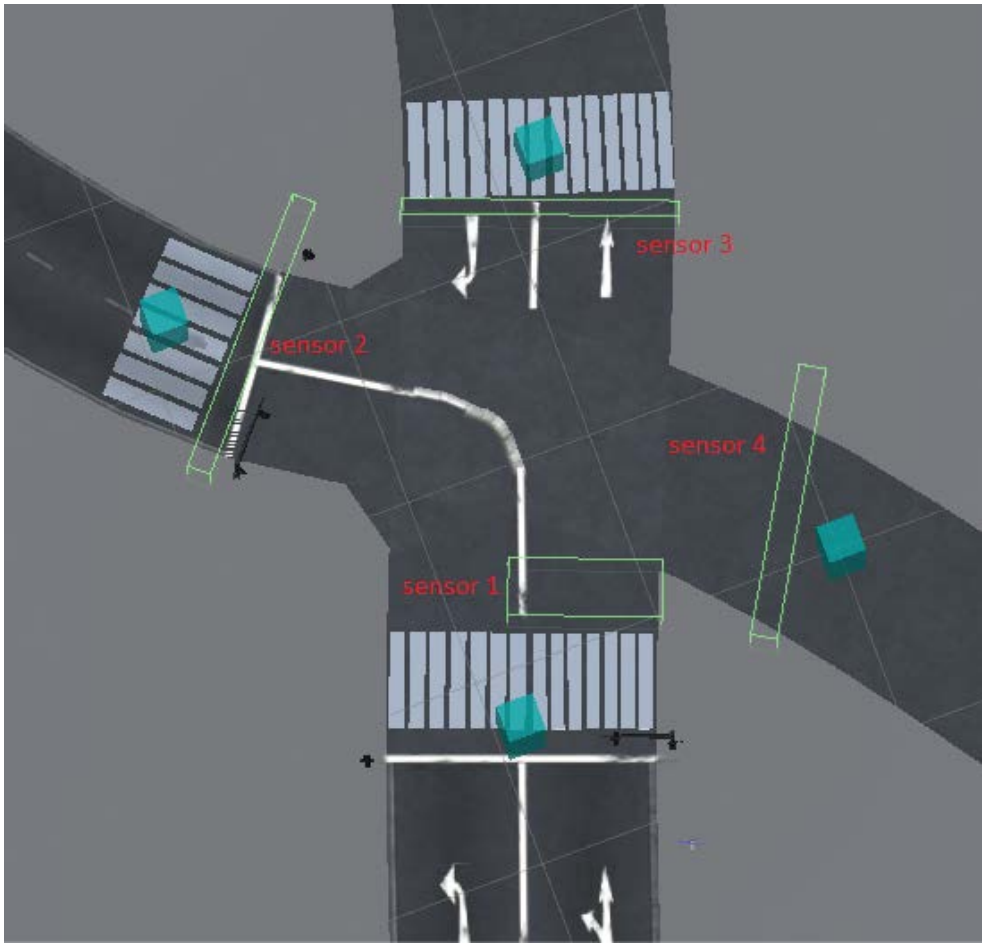

Figura 38. Sensores que intervienen en el control de los intermitentes.

- Saltarse un stop: en este caso habrá un sensor en la zona en que el vehículo debe permanecer parado que compruebe que la velocidad del coche fue de 0 en algún momento a lo largo de su paso por ese sensor, detectando de esta manera si se ha detenido correctamente o añadiendo una infracción en caso contrario.

- Conducir respetando las direcciones: habrá sensores en las entradas de los carriles que vienen en dirección contraria a la del conductor que detecten si éste pasa por ahí y circula, por tanto, en dirección contraria.

- Respetar la distancia de seguridad con los vehículos: cuando un conductor circula, debe mantener una distancia respecto al vehículo que tiene delante que le permita reaccionar a tiempo en caso de que algo inesperado ocurra. Para controlar que el conductor efectivamente mantenga esta distancia, mediante el script *DetectorVehiculoDistancia.cs* se calcula que el conductor mantenga una distancia de 0.5 metros por cada km/h de velocidad que lleve, debiendo mantener, para que todo

sea correcto, por ejemplo una distancia de al menos 50 metros con el coche de delante si se circula a una velocidad de 100 km/h.

- Respetar el carril bus y taxi: a lo largo del recorrido por el tramo urbano, habrá zonas en que un carril esté destinado a la circulación únicamente de autobuses y taxis sobre él, habiéndose situado detectores en tales zonas que indiquen cuando el conductor utiliza estos carriles de forma inadecuada.
- Respetar las zonas amarillas: Un conjunto de líneas amarillas entrecruzadas recuerda a los conductores la prohibición de permanecer parados sobre esa zona. Apoyándonos en la figura 38 podemos entender mejor el funcionamiento de esta norma.

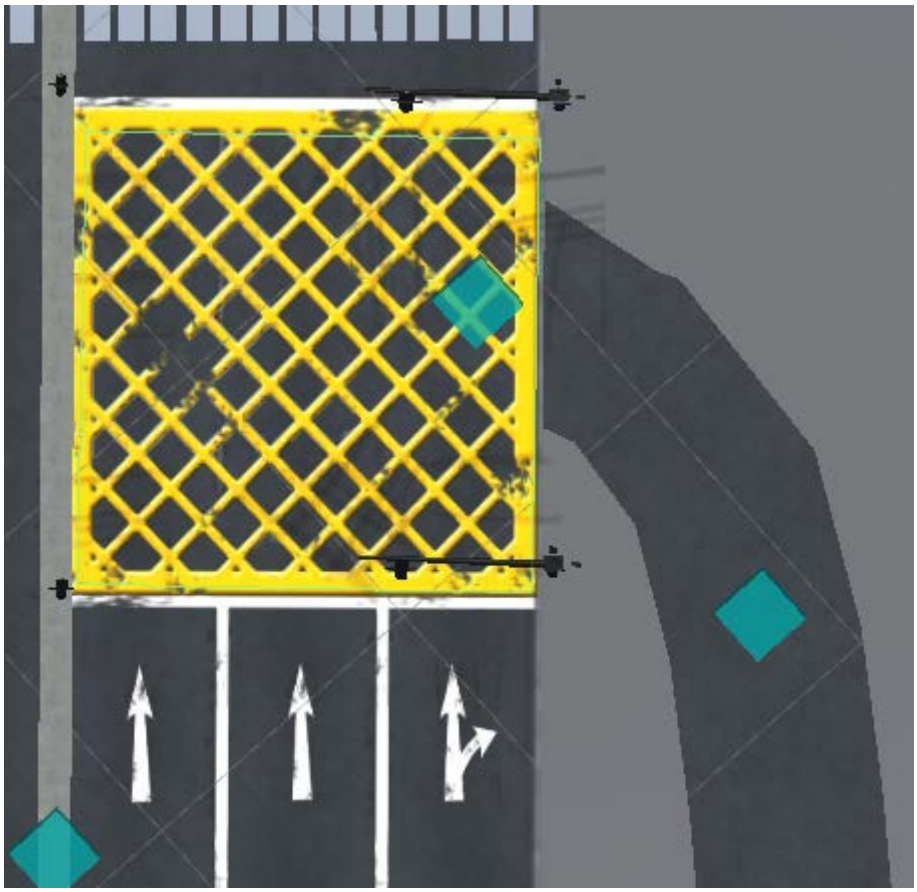

Figura 39. Tramo del recorrido con zona amarilla.

Como acabamos de mencionar, un vehículo no debe obstaculizar nunca estas zonas, destinadas en la mayoría de casos a la incorporación de vehículos que provienen de otro carril. En una situación como la de la figura 39, si el conductor se encuentra con que el semáforo 1 está en verde pero el semáforo 2 está en rojo, no deberá invadir esta zona amarilla, ya que sino imposibilitaría la incorporación de los vehículos provenientes del carril derecho. Para controlar esta infracción se ha situado un sensor sobre esta zona que identifique si el vehículo ha alcanzado una velocidad nula en esta zona, empleando un razonamiento inverso al sensor situado en las zonas donde había una señal de stop.

- Salir correctamente de las rotondas: aunque en la realidad no suele ser la práctica habitual para muchos conductores, uno siempre debe abandonar una rotonda desde el carril derecho, con independencia del número de carriles que tenga la vía a la que uno se incorpora. Para controlar que esto se haga así y el conductor no salga de la rotonda desde uno de los carriles centrales, se sitúan dos sensores que controlen que esto no ocurra, añadiendo una infracción de este estilo en caso contrario.

Este sistema de infracciones no sólo avisa al conductor de cuando está cometiendo una de ellas, sino que también las almacena junto a otros datos de la conducción. En la figura 40 observamos el fragmento del script *infracciones.cs* donde aparece la declaración de estas dos estructuras.

```
// Estructura de infraccion
public struct infraccion
{
    public float instante;
    public float id_infraccion;
    public float posicion_x;
    public float posicion_y;
    public float posicion_z;
    public string observaciones;
};

// Estructura de muestra
public struct muestra
{
    public float instante;
    public float posicion_x;
    public float posicion_y;
    public float posicion_z;
    public float velocidad;
    public float rpm;
    public float marcha;
    public float consumo_instantaneo;
    public float consumo_total;

};
```

Figura 40. Fragmento del script infracciones.cs

La primera de estas estructuras corresponde a la que almacenará las infracciones cometidas, está compuesta por el tiempo en el que se produjo la infracción, el identificador de la infracción, las coordenadas del jugador en ese instante y un texto de observaciones. La segunda recoge los datos de posición del jugador, esta estructura almacenará el instante de tiempo de la muestra, las coordenadas del jugador, la velocidad actual, las revoluciones por minuto y los consumos instantáneos y total.

4.4. Flujo del escenario, apariencia y menús

La aplicación comienza cuando se ejecuta el fichero ".exe", de ahora en adelante ejecutable. Tras esto se mostrará el menú principal por el que el usuario podrá desplazarse entre las diferentes pantallas, así como comenzar un escenario o interactuar con el servidor. Este flujo está representado en la Figura 41.

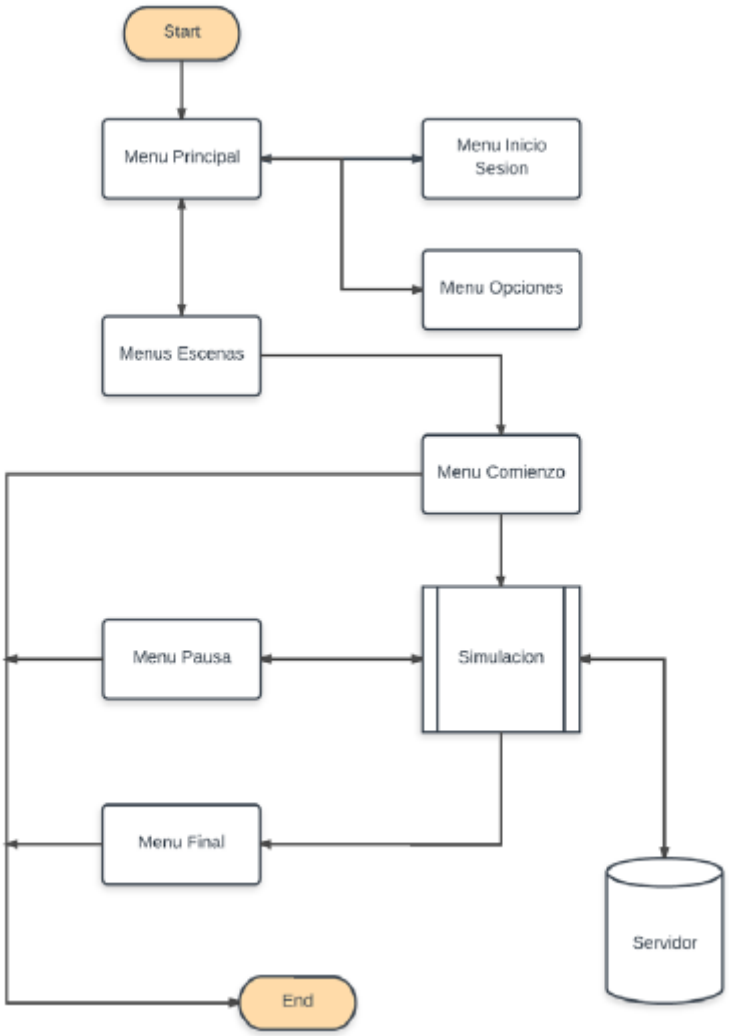

Figura 41. Diagrama de flujo de la aplicación.

La apariencia del proyecto deberá mostrar realismo y semejanza con el tramo diseñado, aunque, como se ha mencionado en alguna ocasión, esta no es la mayor prioridad dado que no dispondremos de los mismos modelos que existen en el escenario real. No obstante, se dota al escenario de los elementos suficientes como para que alguien que conozca el recorrido en la vida real, sea capaz de identificar en cada momento en qué parte está.

Una vez visto el flujo de la aplicación, es importante mostrar la apariencia de cada uno de estos menús y pantallas del juego así como las posibilidades que ofrecen.

4.4.1. Menú principal

Al iniciar la aplicación se mostrará un menú cuyo aspecto podemos ver en la figura 42 y que dispone de las opciones siguientes a través de sus botones:

- Comenzar: este botón nos hará acceder al menú de escenarios.
- Opciones: este botón nos hará acceder al menú de opciones.
- Login: este botón nos hará acceder al menú de identificación.

- Conectar sensores: contiene opciones para la configuración de un sistema de sensores fisiológicos inalámbricos, que miden las constantes del conductor (la aplicación que detecta las contantes y el código para la implementación en Unity, fue desarrollado por otro compañero del grupo de investigación).

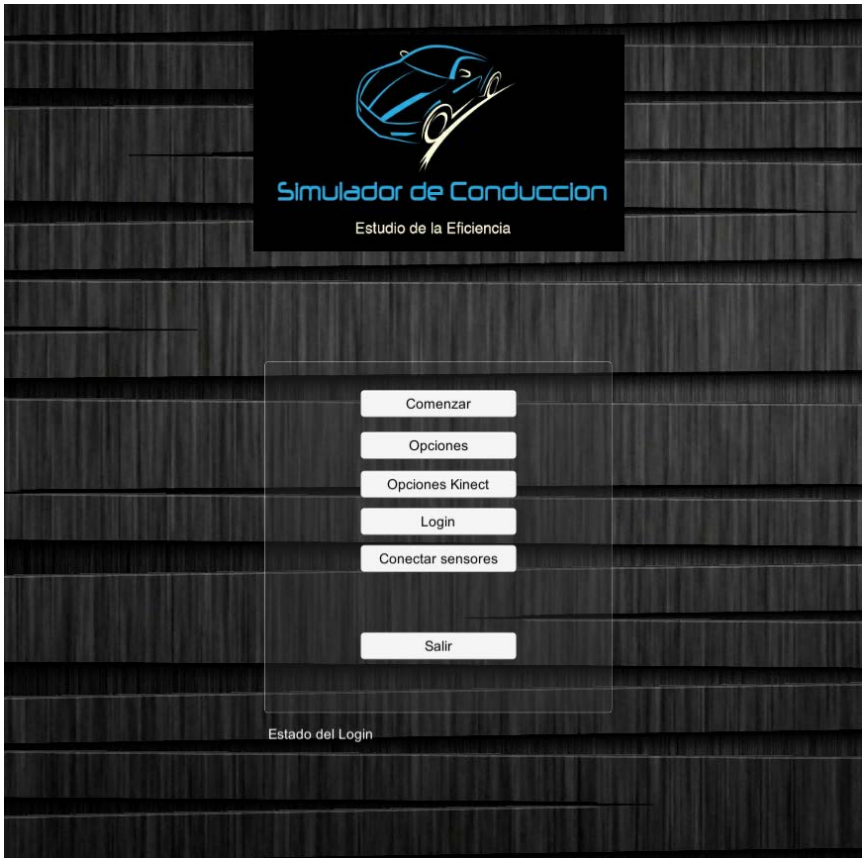

Figura 42. Menú principal.

4.4.2. Menú opciones

En este menú, cuyo aspecto podemos ver en la figura 43, el usuario podrá definir si quiere o no que se le vaya indicando el camino a seguir (modo guiado), si quiere que se le notifiquen las infracciones a medida que las va cometiendo y las estadísticas de su conducción, si va a hacer uso de un volante para la simulación ya que ,como hemos explicado, podría realizarse también por teclado, el tipo de marchas que va a utilizar (automático, manual o levas), el tipo de tracción (tracción delantera FWD, tracción trasera RWD o tracción a las cuatro ruedas AWD), el color del coche o el instante (día, noche u ocaso).

Figura 43. Menú opciones.

4.4.3. Menú escenarios

En este menú, cuyo aspecto puede verse en la figura 44, el usuario podrá elegir la escena que desea simular. Actualmente solo se encuentra una escena, pero está implementado para más por si se quieren añadir otras escenas.

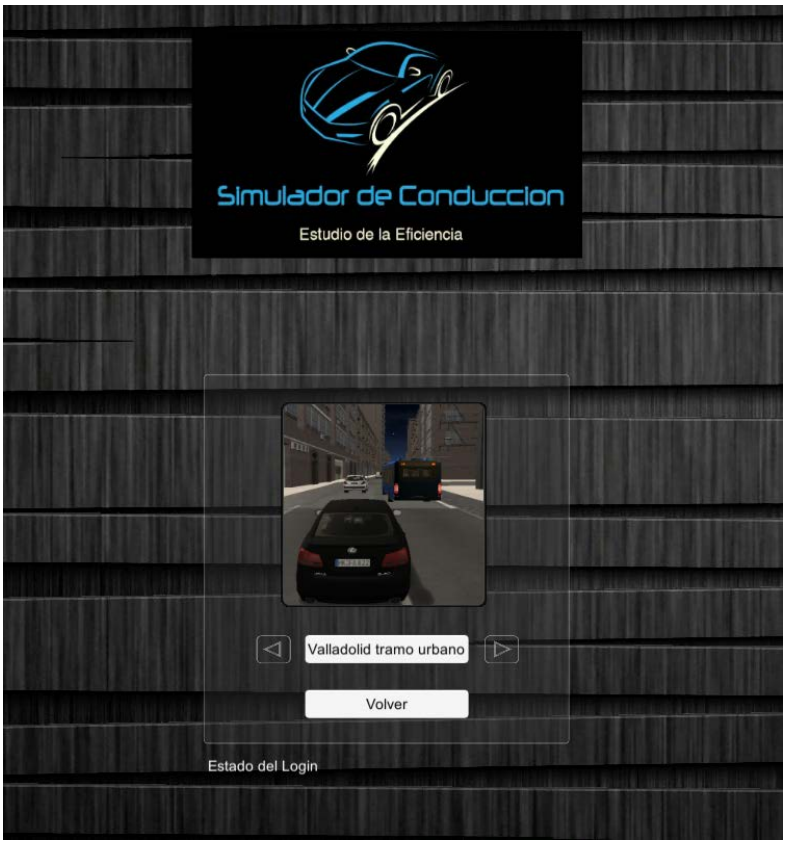

Figura 44. Menú escenarios.

4.4.4. Menú de identificación

En esta sección el usuario podrá validarse en la aplicación para recoger datos acerca de las simulaciones que realice. Deberá estar registrado en el servidor para poder acceder. Estas estadísticas se podrán observar en la página web del servidor. Podemos ver su aspecto en la figura 45.

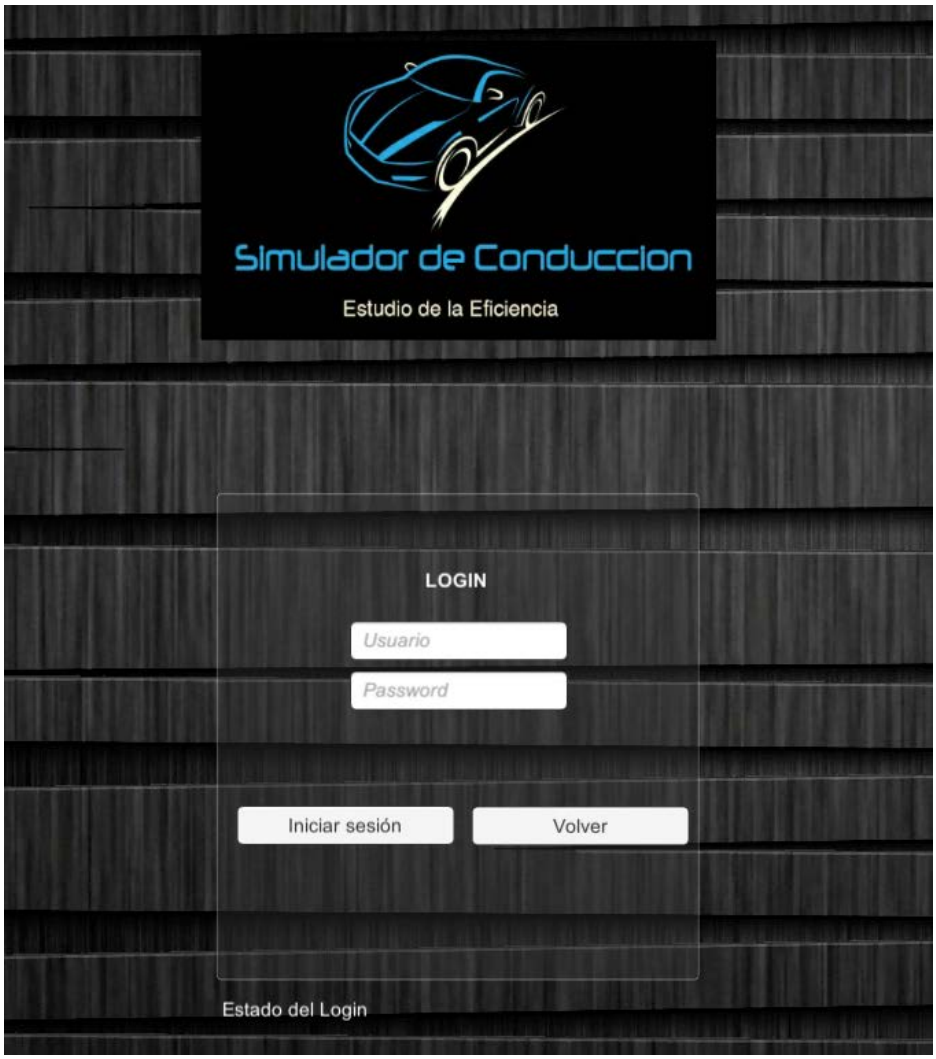

Figura 45. Menú de identificación.

4.4.5. Menú comienzo

Este menú, cuyo aspecto se muestra en la figura 46, será el mostrado al inicio de la simulación. Se utiliza para sincronizar los sensores fisiológicos usados para monitorizar al conductor durante las simulaciones.

Figura 46. Menú al comienzo de la simulación.

4.4.6. Menú pausa

Este menú, cuyo aspecto vemos en la figura 47, será el que se muestre cuando se pause la simulación. Muestra la información relativa tanto a las infracciones como al consumo, además de contener los botones que ofrecen la posibilidad de continuar la simulación o finalizarla, ya sea guardando los datos o sin guardarlos.

Figura 47. Menú pausa.

4.4.7. HUD

Por último, una vez presentado el aspecto de los distintos menús, es importante mostrar el aspecto que tendrá la aplicación durante su uso. En los videojuegos, se denomina *HUD* a la información que en todo momento se muestra en pantalla. Durante la simulación de nuestro videojuego, esta información viene dada en forma de iconos y números. Podemos ver nuestra HUD en la figura 48.

Figura 48. HUD.

El *HUD* en nuestro caso mostrará indicaciones con respecto al vehículo:

- Velocímetro: es el indicador que mide el valor de la velocidad media del vehículo en el último segundo, ya que el intervalo en el que medimos la magnitud, por lo general, es pequeño y se aproxima a la rapidez instantánea. El usuario podrá ver por pantalla la velocidad del vehículo.
- Cuentarrevoluciones: es el indicador que mide las revoluciones por minuto del motor. En pantalla se indicarán las revoluciones de la marcha actual, así como la marcha en cada momento.
- Intermitentes: revelan a los demás usuarios de la vía la intención del usuario de cambiar la dirección de la marcha, ya sea para incorporarse a otro carril del mismo sentido o para girar en una intersección.
- Luces: estos indicadores nos informan sobre el tipo de luces que tenemos encendidas. Las luces de corto alcance son para alumbrar distancias cortas, mientras que las luces de largo alcance son para iluminar distancias mayores. También existen indicadores de luces antiniebla delanteras o traseras.
- Frenos: el freno de disco es un sistema de frenado usado normalmente para ruedas de vehículos. Este indicador se encenderá al frenar el vehículo.
- ABS: El sistema antibloqueo de ruedas, es un indicador utilizado en automóviles que hace variar la fuerza de frenado para evitar que los neumáticos resbalen con el suelo.

4.5. Modo de empleo: jugabilidad

En este capítulo se van a explicar, una vez ya creado el proyecto y lanzado el ejecutable, las opciones de las que el usuario dispone a la hora de elegir un escenario así como los controles que ha de utilizar para la conducción, no sin antes situar el género y la audiencia para la que está destinado.

Nuestro proyecto pertenecerá al género de simulación. Este género se caracteriza por recrear situaciones o actividades del mundo real, dejando al jugador tomar el control de lo que ocurre. La simulación se basa en pretender un alto grado de verosimilizad.

En cuanto a la audiencia, a pesar del contenido violento que alguien podría atribuir al hecho de que se puedan atropellar peatones o colisionar con otros vehículos, la realidad es que el simulador no presenta escenas de mal gusto o contenido ilegal, por lo que en ese sentido no existe una edad restringida para su uso.

Los conocimientos necesarios para su empleo no son muy elevados, basta con la capacidad de poder manejar un volante, sin la necesidad ni mucho menos de ser poseedor de permiso de conducir, por lo que este aspecto tampoco limita de gran manera el público objetivo.

Una vez vistos todos los aspectos, podría recomendarse nuestro simulador para una edad recomendada de 16 años a 22 años para el uso educativo en autoescuelas, pero siendo apto para cualquier público a partir de una edad mínima de 9 años por debajo de la cual se considera que no se tienen las capacidades suficientes para su utilización.

Al iniciar la aplicación, el conductor podrá identificarse en el simulador, configurar las opciones del vehículo, seleccionar un escenario de simulación, conducir por él y consultar las estadísticas obtenidas.

En cuanto a los controles, aunque cabría la opción de utilizar el simulador con el teclado, que ha sido la opción utilizada durante su implementación para lograr mayor agilidad a la hora de hacer pruebas y cambios, el objetivo es que el usuario utilice el dispositivo Logitech G27 Racing Wheel para PC, cuyo coste es de 250€ [59] . Este dispositivo cuenta con 3 pedales (embrague, freno y acelerador) un volante con 6 botones más los 2 de levas y un módulo para una palanca de cambios de hasta 6 marchas y 12 botones. El Logitech G27 es compatible con PC y consolas como PlayStation. Los controles de este dispositivo usados en este escenario por defecto se muestran en la figura 49.

Figura 49. Controles establecidos en el dispositivo Logitech.

Estos controles pueden configurarse desde el *Asset Realistic Car Controller RCCv3*, el cual dispone de un recurso denominado *RCC_Settings* donde podemos asignar los diferentes controles del vehículo. Podemos observar estos controles en la figura 50.

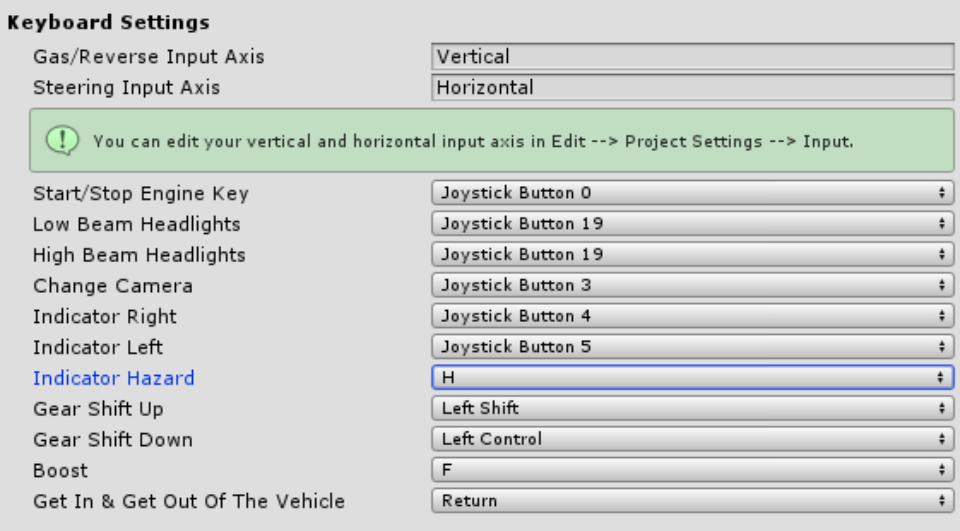

Figura 50. Configuración de controles en RCC Settings.

También se pueden definir los controles en Unity en la sección *Settings -> Inputs*, añadiendo cuantas posibilidades queramos. Además, una vez compilado el simulador, se pueden configurar a nuestro gusto los controles al iniciarlo. De esta forma, aunque el usuario no disponga del dispositivo Logitech, o si dispone de él pero quiere configurarlos a su gusto, puede modificarlos a su parecer. En la figura 51 podemos ver la configuración de los controles cuando se inicia la aplicación.

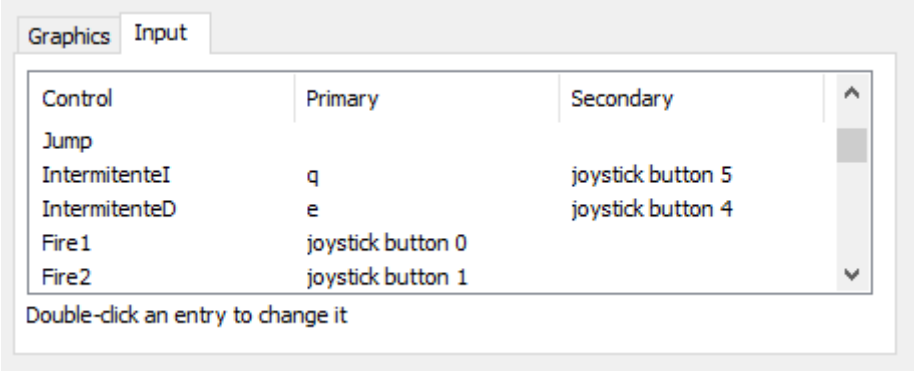

Figura 51. Configuración de controles en Settings Input.

4.6. Registro de la simulación

Una vez el usuario haya finalizado el recorrido de la simulación, nos interesará almacenar el contenido de esta con el fin de poder analizar qué fallos se cometen, en qué situaciones, tiempos, consumo etc. Para ello se generará un fichero de datos, el cual posteriormente será enviado al servidor, por lo que deberá adaptarse a una estructura específica para que el servidor pueda entender el fichero.

Uno de estos requisitos será el nombre del fichero, que estará formado por el identificador del escenario, la fecha y hora de la simulación, el usuario y las opciones de simulación elegidas. En dicho fichero se almacenará también la posición del conductor a lo largo de la simulación con una frecuencia configurable, lo cual permite realizar un seguimiento de este. Almacenará también las distintas infracciones que antes hemos ido mencionando.

De esta forma y gracias a este fichero, se podrá consultar siempre que se quiera cualquier dato relacionado con la simulación.

5. Presupuesto

En este apartado se tratará de analizar el gasto que este proyecto hubiese acarreado de haber tenido que responder económicamente a los costes del hardware y software empleados así como del personal.

5.1. Hardware

Engloba los gastos acarreados por el ordenador el cual se ha utilizado para el desarrollo del proyecto y el volante utilizado para las pruebas, excluyendo del mismo gastos secundarios como el proyector y pantalla utilizado al considerarse prescindibles así como el dispositivo Kinect que se tenía pensado utilizar para monitorizar al usuario pero que finalmente no ha sido empleado.

- Ordenador: necesario para la realización del proyecto. No hace falta decir que son múltiples los modelos y precios de ordenadores que uno puede utilizar, además del hecho de que su uso no sería exclusivo a lo largo de su vida para este proyecto; pero calculando un precio medio de 2000€ y que se ha empleado lo que podría calcularse como un cifra del orden de una octava parte de su vida completa (uso de 6 meses suponiendo una vida media de 4 años), tendríamos que el coste asociado equivaldría a 250€, entendiéndose tal y como se ha explicado que la inversión podría llegar a considerarse como la del total del precio de no disponerse de él.
- Volante Logitech G27 junto con pedales: necesario para la realización de las pruebas así como para el testing adecuado del proyecto. Su precio en el mercado ronda los 250 €. En este caso, y dado que el poseer uno de manera previa se antoja mucho menos común que para el caso del ordenador, consideraremos el total de su valor.
- Infraestructura: muchas veces omitida a la hora de realizar este tipo de presupuestos, debemos recordar que disponer de un puesto en un laboratorio donde poder realizar nuestro trabajo conlleva un gasto asociado, ya sea por el consumo de luz o por el hecho de tener que pagar un alquiler si de una oficina hablásemos. Calcular de manera precisa esta cantidad sería realmente complicado, pero de una manera simbólica y que trata de ser aproximada, y partiendo de la base de que no ha habido que pagar alquiler por el lugar donde el trabajo se ha realizado, se otorgará un valor de 50€.

Hardware	Coste (€)
Ordenador	250 €
Volante Logitech G27	250 €
Infraestructura	50 €
TOTAL	**550€**

Tabla x. Presupuesto Hardware

5.2. Software

Como ya se ha venido comentando a lo largo de este documento, la herramienta para el desarrollo del proyecto es Unity. Este software es gratuito si no se usa con fines comerciales, y, dado que nuestro proyecto está inicialmente diseñado para investigación y por tanto no

generará beneficios económicos, podremos usar la licencia gratuita. Sí deberemos incluir los gastos asociados a algunos de los assets de pago utilizados así como el de algunos modelos empleados.

- Assets: Dentro del universo Unity hemos hecho uso de varios assets, siendo algunos de ellos gratuitos, pero cabe mencionar dos que no lo son y que han sido utilizados para el desarrollo de nuestro escenario. Estos son iTS (Intelligent Traffic System), utilizado para dotar de inteligencia artificial al resto de vehículos, y RCC_V3 (Realistic Car Controller Version 3), utilizado para el desarrollo del vehículo que se utilizará en las simulaciones. El precio de estos assets es de 70€ en el caso de iTS y 50€ en el caso de RCC_V3, precio que se incluirá en los presupuestos en su totalidad.
- Modelos 3D: Para la generación del entorno se necesita de modelos 3D. Dado que al aspecto de diseño gráfico no se le ha dado una gran importancia sino que nos hemos centrado en la parte más técnica, la mayoría de los modelos han sido bastante sencillos y descargados de páginas gratuitas dedicadas a tal fin. Únicamente un modelo ha sido descargado a través del Asset Store, el modelo de autobús urbano utilizado, cuyo precio es de 10 €.

Software	Coste (€)
Asset iTS	70 €
Asset RCC V3	50 €
Modelo 3D	10 €
TOTAL	**130€**

Tabla x. Presupuesto Software.

5.3. Desarrollo del proyecto

Este apartado incluirá el coste asociado a la cantidad de horas empleadas, las cuales, repartidas en un cálculo aproximado de 20 semanales durante los dos primeros meses del proyecto, 25 durante los dos siguientes y 30 horas semanales en el último mes, hacen un total de 500 horas. A pesar de que pudiera valorarse de distinta manera el coste de una hora dedicada al análisis de la de una dedicada al testing o a la programación, y de que el abanico de posibles costes es amplio, estimaremos un valor medio de 11 € la hora.

Desarrollo del proyecto	Coste (€)
TOTAL	**5500€**

Tabla x. Presupuesto del desarrollo del proyecto.

5.4. Presupuesto total

Realizando la suma de todos los valores anteriormente explicados, obtenemos que el coste total de realización del proyecto sería de 6180 €. Se estima que de tener que realizarse este mismo proyecto de nuevo con los conocimientos de la herramienta ahora adquiridos, podrían ahorrarse hasta 100 horas de trabajo para llegar al mismo fin, pero dado que esto supondría la contratación de una persona con experiencia lo cual conlleva más dinero por hora de trabajo, el cálculo final sería de nuevo una cifra aproximada a los 6180 € ya mencionados.

Presupuesto	Coste (€)
Hardware	420 €

Software	130 €
Desarrollo del proyecto	5500 €
TOTAL	**6180 €**

Tabla x. Presupuesto total del proyecto

6. Conclusiones y líneas futuras

En este proyecto se ha realizado un escenario básico de conducción que recorre una parte de la ciudad de Valladolid combinando tramos urbanos y de autovía. Para su realización ha sido necesario refrescar algunos conceptos relativos a la seguridad vial y la conducción eficiente, que a pesar de ser poseedor del carnet de conducir quedaban ya difusos, lo que da una muestra de la realidad de muchos de los conductores que circulan actualmente por nuestras carreteras. Ha sido también necesario poner en práctica un nuevo lenguaje de programación hasta ahora no utilizado a lo largo del grado ni del máster como es C#, aunque cabe decir que dadas sus similitudes con Java no ha sido de gran dificultad.

Este proyecto ha permitido además la posibilidad de adquirir conocimientos básicos con la herramienta Unity 3D, además de adentrarse en el universo de los motores gráficos y los programas de modelado. Así mismo, este tipo de proyectos permiten a quienes los realizan aprender a trabajar de una forma distinta a la que uno acostumbra en las distintas asignaturas, ya que no se dispone de un profesor que sea continuamente quien va guiando al alumno y explicando lo que debe hacer, sino que este debe aprender a buscar las soluciones por sí mismo, algo muy importante de cara al futuro y la inclusión en el mundo laboral.

Uno de los mayores desafíos de este proyecto ha sido recrear la inteligencia artificial del resto de vehículos y usuarios de la vía. El mayor problema encontrado ha sido lograr que los vehículos pudieran responder de forma correcta tanto en los tramos urbanos como en los de autovía, así como lograr que la conducción del coche fuera realista también para todo el abanico de posibilidades y velocidades.

Para poder llevar a cabo todo los objetivos propuestos ha sido necesario priorizar las tareas a realizar, lo que supone que existan algunas características y opciones que hubiera sido interesante implementar pero que no ha sido posible debido a la falta de tiempo. Como líneas futuras para mejorar este proyecto se realizan las siguientes propuestas:

- Mejorar la estética del escenario añadiendo edificios, árboles, bancos y demás elementos existentes en la vida real y de los que en este proyecto se ha prescindido dada la cantidad de tiempo que su inclusión conllevaría.
- Incluir todas las alturas reales del escenario ya que, aunque se han mantenido con rigor las distancias, no hay desniveles en el terreno y se conduce siempre por llano.
- Disponiendo de mayor número de modelos 3D, poder elegir entre más modelos de vehículos distintos para conducir y distintos colores de estos.
- Implementar la opción de que el motor tenga un comportamiento a elegir entre eléctrico, gasolina y diésel.
- Implementar diferentes condiciones meteorológicas como pudiera ser la lluvia o nieve, y el efecto de asfalto mojado o congelado.
- Incluir el control de más infracciones, como por ejemplo adelantamientos sin dar el intermitente, lo cual conllevaría la inclusión de una gran cantidad de lógica y una gran cantidad de sensores repartidos a lo largo de todo el escenario.
- Sería interesante incluir un modo de conducción que simulara el estado de embriaguez para así concienciar a los conductores de lo peligroso que es conducir bajo los efectos

del alcohol. Esto podría realizarse utilizando el *Force Feedback* del volante reduciendo así el control del mismo.

- Incluir un sistema de posición en tiempo real, mediante el cual se pueda ver en todo momento la posición del vehículo respecto al escenario.

Bibliografía

[1] Bowers, P. (2013). Boeing 377 Stratocruiser. Enero 12, 2018, de The Aviation History On-Line Museum. Sitio web: http://www.aviation-history.com/boeing/377.html

[2] Ban Ki-moon. (2010). Decenio de Acción para la Seguridad Vial 2011-2020. Enero 13, 2018, de ONU. Sitio web: http://www.un.org/es/sg/messages/2010/roadsafetyday2010.html

[3] DGT. (2017). Tablas estadísticas 2016. Enero 13, 2018, de DGT. Sitio web: www.dgt.es/es/seguridad-vial/estadisticas-e-indicadores/accidentes-30dias/tablas-estadistica

[4] DGT. (2017). Las principales cifras de la Siniestralidad Vial España 2016. Enero 13, 2018, de DGT. Sitio web: www.dgt.es/Galerias/seguridad-vial/estadisticas-e-indicadores/publicaciones/principales-cifras-siniestralidad/Las-principales-cifras-2016.pdf

[5] OMS. (2017). Accidentes de tránsito. Enero 13, 2018, de OMS. Sitio web: http://www.who.int/violence_injury_prevention/road_traffic/es/

[6] Valcárcel, J. (2010). Estrategia de Seguridad Vial 2011-2020. Enero 13, 2018, de DGT. Sitio web: http://www.dgt.es/Galerias/seguridad-vial/politicas-viales/estrategicos-2011-2020/doc/estrategico_2020_003.pdf

[7] Universitat de Valencia. (2011). Las causas de los accidentes de tráfico: factores de riesgo. Enero 13, 2018, de Universitat de Valencia. Sitio web: https://www.uv.es/sfpenlinia/cas/64las_causas_de_los_accidentes_de_trfico_factores_de_riesgo.html

[8] OMS. (2015). Alcohol. Enero 13, 2018, de OMS. Sitio web: http://www.who.int/mediacentre/factsheets/fs349/es/

[9] Universitat de Valencia. (2011). El factor ambiental: la vía y su entorno. Enero 13, 2018, de Universitat de Valencia. Sitio web: http://www.uv.es/sfpenlinia/cas/643el_factor_ambiental_la_va_y_su_entorno.html

[10] Donate-López, C., Espigares-Rodríguez, E., Jiménez-Moleón, J. J., del Castillo, L., de Dios, J., Bueno-Cavanillas, A., & Lardelli-Claret, P. (2007). Efecto de las circunstancias ambientales sobre el riesgo de defunción de los conductores de vehículos de dos ruedas de motor implicados en accidentes de tráfico. *Gaceta Sanitaria*, *21*(3), 197-203.

[11] Varcárcel, J. Conducción eficiente. Enero 13, 2018, de DGT. Sitio web: http://www.dgt.es/PEVI/documentos/catalogo_recursos/didacticos/did_adultas/Conduccion_eficiente.pdf

[12] IDAE. (2006). Manual de conducción eficiente. Enero 13, 2018, de IDAE. Sitio web: www.apegr.org/images/descargas/Manual_conduccion_eficiente.pdf

[13] Fraile, C. (2017). 10 claves para conducir de forma eficiente. Enero 13, 2018, de DGT. Sitio web: http://revista.dgt.es/es/educacion-formacion/conducir-mejor/2017/0213-Conduccion-eficiente-diez-claves-para-consumir-menos.shtml#.Wm9BDOeCHIV

[14] IDAE. (2015). La conducción eficiente. Enero 13, 2018, de IDEA. Sitio web: http://www.idae.es/uploads/documentos/documentos_10297_TREATISE_Conduccion Eficiente_A2005_A_f3817bad.pdf

[15] IDAE. (2011). La conducción eficiente: un nuevo estilo de conducción que logra importantes ahorros de carburante, reducción de emisiones y que mejora la seguridad. Enero 13, 2018, de IDAE Sitio web: http://www.idae.es/publicaciones/la-conduccion-eficiente-un-nuevo-estilo-de-conduccion-que-logra-importantes-ahorros-de-carburante

[16] Pardillo Mayora, J. M., & Jurado Piña, R. (2008). Aplicación de simuladores en la formación de los conductores.

[17] DriveSim. Enero 13, 2018, de DriveSim. Sitio web: http://drivesimsimulator.com/

[18] Indra. (2015). SmartSim Simulador de conducción de automóviles. Enero 13, 2018, de Indra. Sitio web: https://www.indracompany.com/sites/default/files/triptico_smartsim_simulador_conduccion_automovil_v0716_baja_esp_.pdf

[19] Simumak. Enero 13, 2018, de Simumak. Sitio web: https://www.indracompany.com/sites/default/files/triptico_smartsim_simulador_conduccion_automovil_v0716_baja_esp_.pdf

[20] Droiden, Z. (2017). Si quieres hacer tus propios juegos, estos son los mejores motores que vas a encontrar. Enero 13, 2018, de VidaExtra. Sitio web: https://www.vidaextra.com/listas/si-quieres-hacer-tus-propios-juegos-estos-son-los-mejores-motores-que-vas-a-encontrar

[21] Crossley, R. (2011). Valve on Source and studio culture. Enero 13, 2018, de Develop-online. Sitio web: http://www.develop-online.net/interview/valve-on-source-and-studio-culture/0117030

[22] Valve. (2012). Source Engine Features. Enero 23, 2018, de Valve Sitio web: https://developer.valvesoftware.com/wiki/Source_Engine_Features

[23] Valve. (2012). Source Information Sheet. Enero 23, 2018, de Valve. Sitio web: www.valvesoftware.com/SOURCE_InfoSheet.pdf

[24] Epic Games. (2014). Unreal Engine. Enero 23, 2018, de Epic Games. Sitio web: https://www.unrealengine.com/en-US/what-is-unreal-engine-4

[25] Sweeney, T. (2015). If you love something, set it free. Enero 23, 2018, de Epic Games. Sitio web: https://www.unrealengine.com/en-US/blog/ue4-is-free

[26] Unity. (2014). Unity 5 Announced at GDC 2014, Pre-order Begins. Enero 23, 2018, de Unity. Sitio web: https://unity3d.com/es/company/public-relations/news/unity-announces-unity5

[27] Made with Unity. Enero 23, 2018, de Unity. Sitio web: https://unity.com/madewith

[28] Unity Store. Enero 23, 2018, de Unity. Sitio web: https://store.unity.com/es

[29] CryTek. (2018). CryEngine. Enero 23, 2018, de CryTek. Sitio web: https://www.cryengine.com/

[30] CryTek. (2013). CryEengine 3 Equipped for Development on Sony Computer Entertainment's PlayStation4. Enero 23, 2018, de CryTek. Sitio web: https://web.archive.org/web/20130609043421/http://www.crytek.com/news/cryengine--3-equipped-for-development-on-sony-computer-entertainment--s-playstation-4

[31] Lherot, A. (2017). UbiArt Framework, el motor con el que se construyen los sueños. Enero 23, 2018, de HobbyConsolas. Sitio web: https://www.hobbyconsolas.com/opinion/ubiart-framework-motor-que-construyen-suenos-4251

[32] Solis, E. (2017). UbiArt Framework: el arte convertido en videojuego. Enero 23, 2018, de Missing Number. Sitio web: http://www.missingnumber.com.mx/ubiart-framework-videojuego/

[33] Catá, J. (2017). Comparativa de motores gráficos para videojuegos. Enero 23, 2018, de La web del programador. Sitio web: https://www.lawebdelprogramador.com/pdf/4756-COMPARATIVA-DE-MOTORES-GRAFICOS-PARA-VIDEOJUEGOS.html

[34] Contreras, D. (2011). Motores gráficos. Enero 23, 2018, de Calaméo Sitio web: http://es.calameo.com/read/004154185295b115fbd5c

[35] Moltó, A. C., Pellicer, J. L., & Pérez, J. V. M. (2012). Gráficos a la máxima potencia: Una comparativa entre motores de juego. *3 c TIC: cuadernos de desarrollo aplicados a las TIC*, *1*(2), 3.

[36] 3ds Max. Enero 23, 2018, de Autodesk. Sitio web: https://www.autodesk.es/products/3ds-max/overview

[37] Comparación de 3ds Max 2018 con versiones anteriores. Enero 23, 2018, de Autodesk. Sitio web: https://www.autodesk.es/products/3ds-max/compare

[38] Autodesk. (2015). History of Autodesk 3ds Max. Enero 23, 2018, de Wayback Machine. Sitio web: https://web.archive.org/web/20151024145611/http://area.autodesk.com/maxturns20/history#h1988

[39] Suscripción de 3ds Max. Enero 23, 2018, de Autodesk. Sitio web: https://www.autodesk.es/products/3ds-max/subscribe

[40] Maya. Enero 23, 2018, de Autodesk. Sitio web: https://www.autodesk.es/products/maya/overview

[41] Maya suscripción. Enero 23, 2018, de Autodesk. Sitio web: https://www.autodesk.es/products/maya/subscribe?plc=MAYA&term=1-YEAR&support=ADVANCED&quantity=1

[42] Blender features. Enero 23, 2018, de Blender. Sitio web: https://www.blender.org/features/

[43] Steam. (2017). Blender. Enero 23, 2018, de Steam. Sitio web: http://store.steampowered.com/app/365670/Blender/

[44] Iscar. (2018). SketchUp Pro2018. Enero 23, 2018, de Iscar. Sitio web: https://www.iscarnet.com/sketchup/funcionalidades-sketchup-pro/

[45] SketchUp Pro - New in 2018. Enero 23, 2018, de SketchUp. Sitio web: https://www.sketchup.com/products/sketchup-pro/new-in-2018

[46] Maxon. (2018). Cinema 4D. Enero 23, 2018, de Maxon. Sitio web: https://www.maxon.net/es/productos/cinema-4d/cinema-4d/

[47] Saint-Moulin, B. (2017). 3D softwares comparisons table. Enero 23, 2018, de TDT 3D Sitio web: http://www.tdt3d.be/articles_viewer.php?art_id=99

[48] About Drupal. Enero 23, 2018, de Drupal. Sitio web: https://www.drupal.org/about

[49] Drupal 7, más fácil y potente que nunca. Enero 23, 2018, de Drupal. Sitio web: https://www.drupal.org/drupal-7.0/es

[50] Seed. (2017). Por qué Drupal es el CMS adecuado para las instituciones de Educación Superior. Enero 23, 2018, de Seed. Sitio web: http://www.seedem.co/es/blog/por-que-drupal-es-el-cms-adecuado-para-las-instituciones-de-educacion-superior

[51] Unity. (2018). Road Architect. Enero 23, 2018, de Unity. Sitio web: https://assetstore.unity.com/packages/tools/terrain/road-architect-15739

[52] Unity. (2014). Road Architect Manual. Enero 23, 2018, de Unity.

[53] Intelligent Traffic System. Enero 23, 2018, de Unity. Sitio web: https://assetstore.unity.com/packages/templates/systems/its-intelligent-traffic-system-23564

[54] Road & Traffic System. Enero 23, 2018, de Wired Developments Sitio web: http://wireddevelopments.com/trafficsystem/

[55] Pathing Pedestrian System. Enero 23, 2018, de Unity. Sitio web: https://assetstore.unity.com/packages/templates/systems/pathing-pedestrian-system-46124

[56] Lexus. Enero 23, 2018, de Lexus. Sitio web: https://www.lexusauto.es/car-models/rx/#hero

[57] Descubra el GS. Enero 23, 2018, de Lexus. Sitio web: https://www.lexusauto.es/car-models/gs/#Introduction

[58] Martinez, R. (2013). ¿Cómo conducir con pendientes para minimizar el consumo?. Enero 23, 2018, de Como consumir menos. Sitio web: http://www.comoconsumirmenos.com/2013/01/como-conducir-con-pendientes-para_11.html

[59] G27 Racing Wheel Support. Enero 23, 2018, de Logitech Sitio web: http://support.logitech.com/en_us/product/g27-racing-wheel