UNIVERSIDAD DE VALLADOLID

E.T.S.I. TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DE TECNOLOGÍAS ESPECÍFICAS DE TELECOMUNICACIÓN

MENCIÓN EN TELEMÁTICA

Desarrollo de escenarios de simulación basados en Unity para el aprendizaje de técnicas de conducción segura y eficiente

Autor:
D. Javier Pérez Curiel

Tutor:
D. David González Ortega

Valladolid, 30 de Octubre de 2017

TÍTULO:	**Desarrollo de escenarios de simulación basados en Unity para el aprendizaje de técnicas de conducción segura y eficiente**
AUTOR:	**Javier Pérez Curiel**
TUTOR:	**D. David González Ortega**
DEPARTAMENTO:	**Departamento de Teoría de la Señal y Comunicaciones e Ingeniería Telemática**

TRIBUNAL

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

FECHA:
CALIFICACIÓN:

Resumen

En la actualidad los videojuegos son una forma de entretenimiento muy importante en la sociedad. La continua innovación y avances de la tecnología en campos la realidad aumentada (AR) o realidad virtual (VR) hacen que esta industria este ganando popularidad en la sociedad, además de crear nuevas formas de entretenimiento ofreciendo gráficos cada vez más realistas. Estos avances tienen repercusión en la educación, ya que esta industria también se beneficia del desarrollo de la tecnología. Este proyecto consiste en la creación de un escenario de simulación para la conducción, con el que poder poner en práctica los conocimientos en esta área, usando la plataforma de desarrollo Unity 3D. Este escenario sirve de gran ayuda para la práctica y entrenamiento en la conducción, donde el usuario podrá realizar la ruta indicada.

Abstract

Today video games are the main form of entertainment. Continuous innovation and advances in technology in fields such as augmented reality (AR) or virtual reality (VR) cause this industry to gain popularity in society, as well as to create new forms of entertainment offering increasingly realistic graphics. These have an impact on education, as this industry also benefits from the development of technology. This project consists of the creation of a simulation scenario for driving, with which to take advantage of the knowledge in this area, using the Unity 3D development platform. This scenario is a major help for practice and driving training, where the user can drive through the indicated route.

Palabras clave

Simulador, Escenario, Conducción, Unity 3D, Eficiencia, Infracciones, Consumo, *Realistic Car Controller*, *intelligent Traffic System*, Inteligencia artificial.

Agradecimientos

En primer lugar, a mis padres por la ayuda y dedicación que me han dado para poder realizar mis estudios de Ingeniería de Telecomunicaciones.

Gracias a mis amigos por la ayuda y apoyo moral en los momentos difíciles, tanto los compañeros de universidad como el grupo de amigos de la infancia que siguen ahí día a día.

También deseo agradecer al tutor de este trabajo de fin de grado David González Ortega por el apoyo y la posibilidad de realizar este trabajo, así como a mis compañeros de laboratorio por la ayuda en Unity.

ÍNDICE

1 Introducción 10
1.1 Introducción 10
1.2 Motivación 11
1.3 Objetivos 11
2 Seguridad vial 12
2.1 Los Accidentes de tráfico 12
2.2 Educación vial 15
2.3 Factores que intervienen en la circulación 15
2.4 Impacto de los simuladores en la educación vial 18
3 Conducción eficiente. 20
3.1 Ventajas de una conducción eficiente 20
3.2 Análisis y clasificación de Factores 20
3.3 Impacto en el consumo 23
3.4 Aspectos prácticos 25
3.5 Consejos para una conducción eficiente 26
4 Herramientas para la creacion de videojuegos 28
4.1 Concepto de videojuego 28
4.2 Géneros de videojuegos 28
4.3 Motores para producir videojuegos 30
4.3.1 Unreal Engine 4 30
4.3.2 Unity 31
4.4 Comparativa 34
4.4.1 Desarrollo 34
4.4.2 Economía 34
4.5 Elección de la herramienta de nuestro proyecto 35
5 Herramientas para el diseño 3D 36
5.1 Introducción 36
5.2 Pasos de modelado 36
5.2.1 Modelado 36
5.2.2 Sombreado 36
5.2.3 Texturizado 36

5.2.4 Animación y Rigging 37
5.2.5 Renderizado 38
5.3 Aplicaciones 38
5.3.1 Cine 38
5.3.2 Videojuegos 39
5.3.3 Arquitectura 39
5.3.4 Diseño industrial 39
5.4 Herramientas para el diseño en 3D 40
5.4.1 Autodesk 40
5.4.2 Blender 41
5.4.3 SketchUp 42
5.5 Comparativa 43
6 Simuladores 44
6.1 Introducción 44
6.2 Tipos de simuladores 44
6.3 Elementos de un simulador para la conducción 45
6.3.1 Logitech G27 Racing Wheel 45
6.3.2 Logitech Driving Force GT 46
6.3.3 CXC Motion Pro II 47
6.3.4 Most Realistic Racing Simulator 48
6.4 Simuladores 49
6.4.1 SIMESCAR 49
6.4.2 DriveSim 51
6.4.3 iRacing 52
6.4.4 Project CARS 53
6.5 rFactor 54
7 Desarrollo del proyecto 55
7.1 Planificación 55
7.2 Desarrollo 55
7.2.1 Resumen 55
7.2.2 Género 56
7.2.3 Audiencia 57
7.2.4 Flujo 57

7.2.5 Apariencia 58
7.3 Implementación 59
7.3.1 Organización 59
7.3.2 Pantallas del juego (Interfaz) 60
7.3.3 Jugador (vehículo) 64
7.3.4 Otros usuarios (IA) 66
7.3.5 Escenario 67
7.3.6 Cámara 68
7.3.7 Scripts 69
7.4 Jugabilidad 75
7.4.1 Objetivos 75
7.4.2 Acciones del usuario 76
7.4.3 Controles 76
7.4.4 Log 77
8 Presupuesto 81
8.1 Presupuesto inicial 81
8.1.1 Hardware 81
8.1.2 Software 81
8.2 Presupuesto de desarrollo 82
8.3 Presupuesto total 82
9 Conclusión y Líneas futuras 83
10 Anexos 84
10.1 Intelligent Traffic System (iTS) 84
10.1.1 Configuración de un Nuevo Vehículo (New Vehicle) 84
10.1.2 Configuración de un Sistema de Tráfico (Traffic System) 85
10.1.3 Configuración de la Física de un Vehículo (*Traffic Car Physics*) 88
10.1.4 Configuración de un Spawner 90
10.2 RealisticCarControllerV3 92
10.2.1 Player Car 92
10.2.2 Car Camera 96
11 Referencias 97

ÍNDICE DE FIGURAS

Figura 1: Parque de automóviles 14
Figura 2: Evolución de los accidentes con victimas 14
Figura 3: Factor humano 16
Figura 4: Consumo de energía en España 21
Figura 5: Consumo medio de los turismos 21
Figura 6: Consumo energético de España 23
Figura 7: Reducción de las emisiones en la conducción 24
Figura 8: Consumo a 60 Kph 24
Figura 9: Entorno Unreal Engine 4 30
Figura 10: Entorno Unity 3D 31
Figura 11: Soporte multiplataforma de Unity 32
Figura 12: Comparativa de las diferentes licencias de Unity 33
Figura 13: Diferentes licencias 33
Figura 14: Textura de madera 37
Figura 15: Animación y rigging de diferentes personajes 38
Figura 16: Película “La novia cadáver” de Tim Burton desarrollada por ordenador en 3D 38
Figura 17: Desarrollo de una habitación por ordenador en 3D 39
Figura 18: Desarrollo de un tornillo en 3D 39
Figura 19: Entorno de Autodesk 40
Figura 20: Entorno de Autodesk Maya 41
Figura 21: Entorno de Blender 42
Figura 22: Entorno de SketchUp 43
Figura 23: Logitech G27 Racing Wheel 45
Figura 24: Logitech Driving Force GT 46
Figura 25: CXC Motion Pro II 47
Figura 26: Most Realistic Racing Simulator 48
Figura 27: Simulador Simescar 49
Figura 28: Simulador Simescar AMBAR – SMCAMB 50
Figura 29: Simulador Simescar SILVER – SMCSLV 50
Figura 30: Simulador Simescar BRONZE – SMCBRZ 51
Figura 31: Características de DriveSim. 52
Figura 32: Entorno DriveSim. 52
Figura 33: Entorno iRacing 53
Figura 34: Entorno Project Cars 2 53
Figura 35: Entorno rFactor 54
Figura 36: Mapa de Valladolid representado en el escenario 55
Figura 37: Ruta de la simulación. 56
Figura 38: Diagrama de flujo de la aplicación 58
Figura 39: Apariencia del escenario 59
Figura 40: Interfaz HUD del escenario 60
Figura 41: Menú principal del videojuego 61
Figura 42: Menú de opciones del videojuego 62

Figura 43: Menú de escenarios del videojuego 62
Figura 44: Menú de identificación de la sesión del videojuego 63
Figura 45: Menú de pausa de la simulación del videojuego 63
Figura 46: Menú de comienzo de la simulación del videojuego 64
Figura 47: Diferentes modelos de la gama Lexus GS 450h 64
Figura 48: Modelo de la gama Lexus GS 450h que usara el jugador 65
Figura 49: Modelo de autobús usado en el escenario 66
Figura 50: Modelo de la gama Peugeot Bipper usado en el escenario 66
Figura 51: Modelo de la gama Peugeot 207 usado en el escenario 67
Figura 52: Modelo de ciclista usado en el escenario 67
Figura 53: Modelo de peatones usados en el escenario 67
Figura 54: Mapa virtual en Unity 3D 68
Figura 55: Componente Cámara 68
Figura 56: Visión del conductor 69
Figura 57: Script infracciones 70
Figura 58: Consumo de las diferentes marchas 70
Figura 59: Sensores que determinan el comportamiento del usuario en un semáforo 71
Figura 60: Sensores que determinan el comportamiento del usuario en un ceda el paso 72
Figura 61: Sensores que determinan el comportamiento de los intermitentes 72
Figura 62: Código de configuración de marchas automáticas 73
Figura 63: Código de configuración de marchas manuales 74
Figura 64: Código de configuración de las luces de corto y largo alcance 74
Figura 65: Código de configuración de los intermitentes 75
Figura 66: Controles del usuario por defecto 76
Figura 67: Controles del usuario definidos en Unity 76
Figura 68: Controles del usuario definidos en el RCCv3 77
Figura 69: Controles definidos al ejecutar la aplicación 77
Figura 70: Estructura del nombre del fichero de simulación (Log) 78
Figura 71: Contenido del Log en relación con los datos de la simulación 78
Figura 72: Contenido del Log en relación con los datos del jugador 79
Figura 73: Contenido del Log en relación con los datos de las infracciones 79
Figura 74: Código utilizado para el envío del fichero de simulación al servidor 80
Figura 75: Configuración general del iTS 85
Figura 76: Líneas de trafico de iTS 86
Figura 77: Conectores de trafico de iTS 87
Figura 78: Características de un vehículo en el iTS 89
Figura 79: Configuración de un vehículo en el iTS 90
Figura 80: Opciones de un Spawner 90
Figura 81. Configuración de Realistic Car Controller V3 92
Figura 82. Vehículo donde se pueden apreciar los diferentes componentes que los forman. 93
Figura 83. Configuración de las ruedas 93
Figura 84. Configuración del volante 93
Figura 85. Configuración de la suspensión 94
Figura 86. Configuración de la mecánica 94

Figura 87. Configuración de los sistemas de estabilidad 95
Figura 88. Configuración de las luces 95
Figura 89. Configuración de los sonidos 95
Figura 90. Configuración de daños 96
Figura 91. Mensaje de comprobación general del RCCv3 96

ÍNDICE DE TABLAS

Tabla 1: Tabla de indicadores de mejora de la conducción para el 2020 13
Tabla 2: Consumo de energía final 22
Tabla 3: Tabla comparativa de la clasificación de modelos Lexus GS en cuanto a emisiones 23
Tabla 4: Tabla de videojuegos desarrollados con Unreal Engine 31
Tabla 5: Tabla de videojuegos desarrollados con Unity 3D 34
Tabla 6: Comparación de las diferentes herramientas de modelado 3D [] 43
Tabla 7: Especificaciones del vehículo Lexus GS 450h usado en el proyecto 65
Tabla 8: Presupuesto hardware 81
Tabla 9: Presupuesto software 82
Tabla 10: Presupuesto desarrollo 82
Tabla 11: Presupuesto total 82

1 INTRODUCCIÓN

1.1 INTRODUCCIÓN

Desde el primer videojuego de la historia, observamos como este mundo está en continuo desarrollo e innovación. En la actualidad existen diversos dispositivos (ordenadores, portátiles, *smartphones*, *Raspberry Pi*, consolas, tabletas, realidad virtual VR) los cuales están diseñado de forma que pueden ejecutar videojuegos, por lo que este mundo ha evolucionado notablemente en los últimos anos a la par con el reciente desarrollo de la tecnología (cámaras para detectar el movimiento, software de simulación de realidad virtual, etc.).

En la actualidad, aun existiendo muchos géneros de videojuegos definidos (deportes, rol, multijugador, estrategia, plataformas,…) cabe destacar una tendencia en creación de secuelas de videojuegos ya existentes, los cuales fueron un éxito en el mercado, o fueron de gran innovación. Cada una de estas secuelas tiene mayor realismo visual gracias a los avances de hardware entre los consecutivos títulos. Por lo que cada secuela consigue crear escenarios más similares a la realidad y consigue dotar con mayor realismo a los rostros y cuerpos.

Este tipo de videojuegos trata de simular con mayor realismo la realidad proyectada gráficamente.

Los videojuegos de simulación reproducen sensaciones que en realidad no están sucediendo. Pretenden reproducir sensaciones físicas (velocidad, aceleración, percepción del entorno) y una de sus funciones es dar una experiencia real de algo que no está sucediendo para de esta forma no poner en riesgo la vida de alguien.

Además de los explicados anteriormente se intenta representar la realidad de forma precisa, aunque se creen mundos no existentes en la vida real. (Por ejemplo, en el videojuego *Need for speed* podemos observar un videojuego de carreras, las cuales son realistas, pero no están basadas en ninguna realidad ni historia).

Los primeros simuladores que ha conocido el hombre surgieron en los años 1960. Su principal misión era preparar mejor a los pilotos de aviación. Hoy en día se puede decir que son indispensables.

Se les considera pequeños juegos ya que no son reales. La función de los simuladores es aproximarse lo más posible a la realidad. El bajo coste de esta herramienta de aprendizaje ha simplificado su expansión.

Este proyecto tiene como objetivo estudiar y aprender el funcionamiento y conceptos básicos de un simulador, desarrollado con la herramienta de desarrollo 3D Unity un escenario de conducción básico que se asemeje a la realidad (en cuanto a reglas o forma de conducir), así como el estudio de las estadísticas generadas por diferentes usuarios que realicen pruebas de conducción para añadirlo a un simulador.

1.2 MOTIVACIÓN

A la hora de elegir este proyecto, se ha tenido en cuenta el actual crecimiento de dispositivos del mercado, los cuales tienen acceso a las nuevas tecnologías y son capaces de poder ejecutar aplicaciones del tipo que vamos a desarrollar.

También el poder aprender y poner en práctica el desarrollo de videojuegos básicos tanto para ordenadores como para dispositivos móviles. El uso de nuevas tecnologías como el uso de la *Kinect* (dispositivo para captar movimiento) para detector la visión del conductor o el uso de la herramienta *Unity 3D*.

Otro incentivo para elegir este proyecto es el aprendizaje de un nuevo lenguaje de programación no visto en la Carrera como es C#.

Mencionar, por último, que el principal interés y motivación a la hora de elegir este proyecto es la pasión y el entusiasmo por los videojuegos, siendo un consumidor diario de esta industria desde hace muchos desde hace más de una década.

1.3 OBJETIVOS

El principal objetivo de este proyecto es la creación de un escenario de conducción con la herramienta Unity 3D, con la que crearemos una aplicación donde el usuario podrá realizar un recorrido simulado por un entorno basado en la ciudad de Valladolid y podrá poner a prueba sus habilidades prácticas de conducción (ya que dispondremos de un elemento hardware para poder conducir el vehículo, más concretamente el volante Logitech G27). Se podrá estudiar la eficiencia y las estadísticas de la simulación realizada por el usuario, la cual será posteriormente almacenada en un servidor donde se encuentra el historial de las simulaciones realizadas de los diferentes escenarios para el departamento de GTI de la Universidad de Valladolid.

Los objetivos propuestos para desarrollar este trabajo son:

- Creación de una interfaz de usuario:
    - Menú para la selección de opciones relacionadas con el escenario y el vehículo
    - Interfaz para mostrar la información de la simulación
    - Implementación de la simulación de un conductor. Desarrollando elementos como la posición de la cámara o la creación de un entorno de simulación
- Vista en primera persona (*First Person*)
- Control del vehículo, así como la implementación de las leyes físicas relacionadas con él.
- Creación de scripts para el cálculo de la eficiencia de la conducción.
- Creación de scripts para la comunicación con el servidor.
- Implementación de animaciones en el escenario.
- Implementación del audio en el escenario.
- Implementación de la inteligencia artificial (*artificial intelligent*, AI) para los diferentes vehículos de la simulación (coches, bus, ciclistas).

2 SEGURIDAD VIAL

2.1 LOS ACCIDENTES DE TRÁFICO

El número de accidentes de tráfico con víctimas aumentó un 7% en el año 2015, en España [1], en comparación con el año anterior. Este es el tercer año consecutivo en el que se observa un incremento en el número de accidentes con víctimas. Este hecho está en parte relacionado con la mejora del nivel de notificación de los accidentes ocurridos en vías urbanas y, especialmente, de aquellos accidentes de menor gravedad en los que no hay personas fallecidas u hospitalizadas [1].

Analizaremos los resultados relacionados con los accidentes de tráfico ocurridos en España en el año 2015 [1]. Analizando con más detalle la distribución del número de fallecidos, hay ciertos aspectos destacables.

En primer lugar, una evolución en esa distribución en autopistas y autovías, con una reducción del 4% en estos dos tipos de vía, y en carreteras convencionales, con un aumento del 1%.

En cuanto al tipo de accidente, se observan las mayores reducciones del número de fallecidos por las salidas de la carretera (5%), las colisiones frontales (7%) y las colisiones laterales y frontolaterales (7%). Por su parte, hay aumentos importantes en las colisiones traseras y múltiples (17%) y en los accidentes clasificados en la categoría de otros tipos (16%), donde se incluyen aquí, entre otros accidentes, las caídas o las colisiones contra obstáculos o elementos de la vía.

El número de fallecidos también ha evolucionado de manera diferente según el medio de desplazamiento. En concreto, hay aumentos en los ciclomotores, con 3% más de fallecidos; motocicletas, con un 15% más de fallecidos; y peatones, con un 9% más. Por el contrario, hay reducciones en todos los demás tipos de vehículos. En el caso de los ocupantes de turismos, esta reducción se ha situado en un 4% [2].

En cuanto a la edad de los fallecidos, se observan aumentos en los fallecidos en los grupos de edad de 15-34 y 65-84 años, y reducciones en los restantes. La mayoría de accidentes con víctimas y heridos no hospitalizados ocurren en vías urbanas (65% y 62% del total, respectivamente). El mayor número de fallecidos se presentan en carreteras convencionales (57% del total, 78% si restringimos el análisis a las vías interurbanas). El 76% de los accidentes se producen en días laborables, contabilizándose en ellos el 66% de los fallecidos. En cuanto al tipo de accidente, aunque las colisiones laterales, traseras y múltiples suponen más de la mitad de los accidentes, son las salidas de la vía las que provocan un mayor porcentaje de fallecidos.

A pesar de que en el 77% de los accidentes con víctimas está implicado un turismo, los fallecidos en este tipo de vehículo suponen el 41% del total. Los peatones son los usuarios más vulnerables, como prueba el hecho de que estén implicados en un 14% de accidentes con víctimas, pero supongan el 22% del total de fallecidos. En términos de lesividad, les siguen las motocicletas, implicadas en el 25% de los accidentes con el 19% de fallecidos. En relación con la edad, se observa que el 32% de los fallecidos tenían una edad comprendida entre 35 y 54 años. En el 69% de los accidentes con víctimas estaba implicado al menos un varón.

Los objetivos de mejora de la seguridad vial en nuestro país se plasmaron en la "Estrategia de Seguridad Vial 2011-2020" [3], aprobada por Consejo de Ministros de 25 de febrero de 2011, que incluye la concreción de 13 retos. La Tabla 1 muestra los correspondientes indicadores, su valor en el año base, 2009, su valor en el año final (objetivo), 2020, en el año de referencia, 2015, y en el año anterior, 2014.

La tabla 1 muestra los trece indicadores a mejorar en 2020 relacionados con el número de accidentes. Algunos de estos indicadores fijados para 2020 ya se han conseguido, como es el caso de 30% menos de fallecidos en accidentes urbanos.

INDICADORES	CIFRA BASAL 2009	CIFRA 2014	CIFRA 2015	OBJETIVO 2020
1. Bajar la tasa de fallecidos a 37 por millón de habitantes	59	36	36	Inferior a 37
2. Reducción del número de heridos graves en un 35%	13.923	9.574	9.495	9.050
3. Cero niños fallecidos sin sistema retención infantil	12	2	5	0
4. 25% menos conductores de 18 a 24 años fallecidos y heridos graves en fin de semana	730	360	353	548
5. 10% menos de conductores fallecidos mayores de 64 años	203	213	200	183
6. 30% de reducción de fallecidos por atropello	459	310	306	321
7. 1 millón de ciclistas más sin que se incremente su tasa de mortalidad	1,2	1,6	1,2	1,2
8. Cero fallecidos en turismos en zona urbana	101	71	61	0
9. 20% menos de fallecidos y heridos graves usuarios de motocicleta	3.473	2.870	2.928	2.778
10. 30% menos de fallecidos por salida de vía en carretera convencional	520	277	285	364
11. 30% menos de fallecidos en itinerarios interurbanos	170	99	101	119
12. Bajar del 1% los positivos en aire espirado en los controles preventivos aleatorios.	6,7%	1,7%		Inferior al 1%
13. Reducir en 50% el % 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% (conv.100) **	No disponible	No disponible	6,2% (autopista) 3,5% (autovía) 7,9% (conv.90) 8,2% (conv.100)

**** Conv.90 hace referencia a las carreteras convencionales con límite de velocidad de 90 kmh***

***** Conv.100 hace referencia a las carreteras convencionales con límite de velocidad de 100 kmh***

Tabla 1: *Tabla de indicadores de mejora de la conducción para el 2020*

Procedemos a analizar cómo va variando el parque de automóviles. Se entiende como parque de automóviles los mencionados en la figura 1. Observamos que más de las dos terceras partes son turismos.

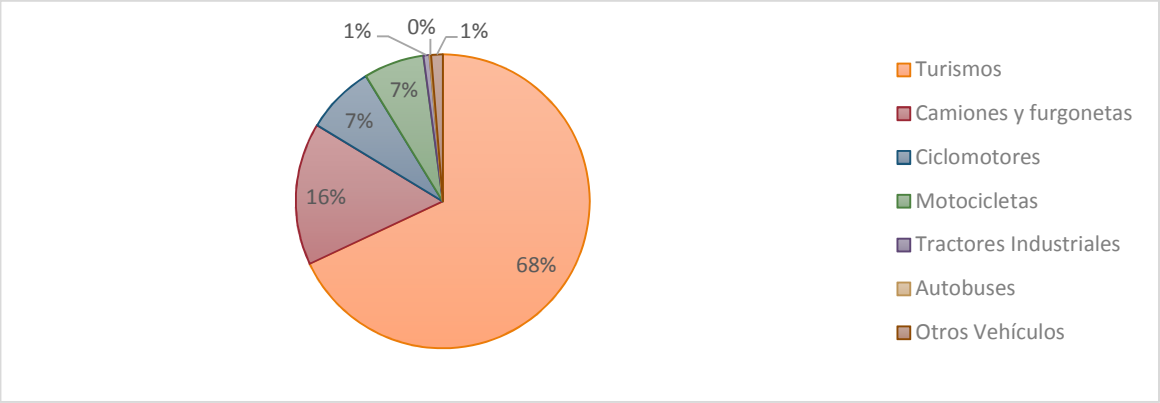

Figura 1: *Parque de automóviles*

En cuanto al número de accidentes con víctimas dependiendo de las vías obtenemos lo mostrado en la figura 2.

En el año 2015, el 35% de los accidentes de tráfico con víctimas se registraron en las vías interurbanas, alcanzando la cifra de 34.558 accidentes, y en ellos se produjeron el 74% de las víctimas mortales, 1.248 fallecidos. El índice de letalidad en 2015 en las vías interurbanas fue de 2,3%, un valor superior a la letalidad para el conjunto de las vías que fue de 1,2%.

En el año 2015 se ha producido un fallecido más por accidente de tráfico en las vías interurbanas al comparar con el año 2014, los heridos hospitalizados disminuyeron un 2% y los heridos no hospitalizados lo hicieron en un 1%.

El número de fallecidos se ha reducido de 1,4 a 0,5 en el periodo 2006-2015. Esta reducción equivale a un millón de fallecidos menos en una década.

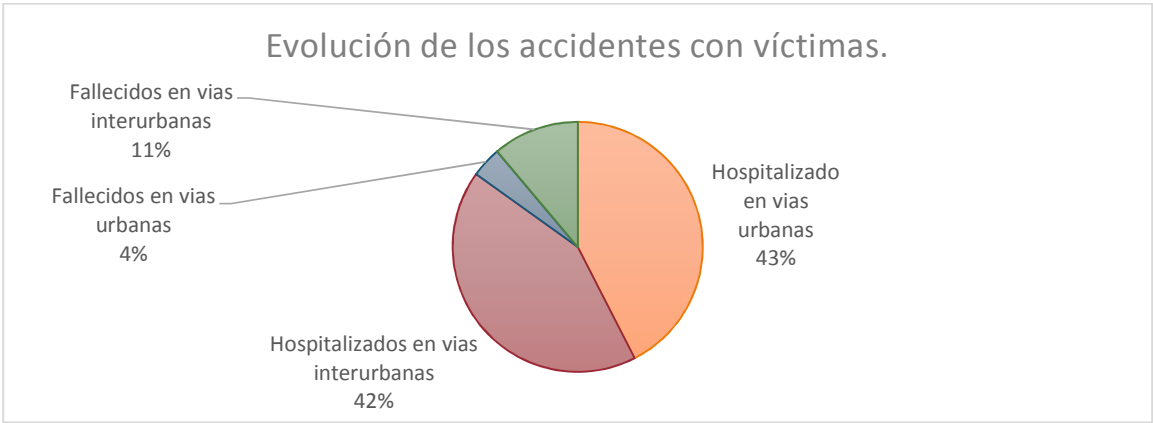

Figura 2: *Evolución de los accidentes con victimas*

2.2 EDUCACIÓN VIAL

Siempre se ha pensado que la educación vial es una materia relacionada con el mundo de la escuela y solo en ella y en ese periodo de la vida, las personas se educan vialmente [4]. En el mejor de los casos, los adultos vuelven a tener contacto con ella cuando obtienen el permiso de conducción en la autoescuela.

El concepto ha cambiado y se considera que la educación vial es un proceso que se inicia a la edad más temprana posible pero que su educación se prolonga a lo largo de toda la vida. Los objetivos que pretende alcanzar es que los ciudadanos circulen con seguridad y fluidez por las vías públicas en cualquier situación y circunstancia, no solo como conductores, sino también como peatones o usuarios de los transportes públicos o privados.

2.3 FACTORES QUE INTERVIENEN EN LA CIRCULACIÓN

Existen tres tipos de factores que intervienen en la conducción. Estos son el factor humano, el factor mecánico, y el factor ambiental.

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 [4].

Entre algunas de las medidas adoptadas por la Comisión Europea en materia de Seguridad Vial relacionadas con estos factores están:

- Regulación de los límites de velocidad.
- Educación vial.
- Mejora de las infraestructuras viarias.
- Medidas sobre dispositivos de seguridad activa de los vehículos.

2.3.1.1 Factor humano

El ser humano es el principal desencadenante de los accidentes de tráfico situándose por encima de los otros dos factores. La figura 3 muestra una situación donde un vehículo (centro de la imagen) detecta otro vehículo (blanco, arriba de la imagen) que se incorpora a la vía, y reaccionará acorde a ello para evitar un accidente. Dependiendo de sus necesidades, el hombre en el sistema vial se desenvuelve como:

- **Peatón**: Se consideran peatones a las personas que transitan a pie por las vías, calles, caminos y carreteras y aceras. También se consideran peatones a las personas con capacidades especiales. El peatón es considerado el elemento más importante y a la vez más frágil dentro del sistema vial.
- **Pasajero**: Es la persona que transita a pie por vías, calles, caminos, carreteras, etc. Son las personas con capacidades especiales que transitan igualmente en vehículos especiales manejados por ellos o terceros.
- **Ciclista**: Es la persona que conduce una bicicleta, y, como tal, es responsable de la movilización de la misma.

- **Conductor**: Es la persona legalmente facultada para conducir un vehículo. El conductor influye en la seguridad vial dependiendo de su experiencia y de su respuesta ante situaciones inesperadas.

Figura 3: *Factor humano*

Otro factor desencadenante en los accidentes respecto al factor humano es el tiempo, el lapso entre que el conductor percibe una eventualidad y reacciona ante ella. El Tiempo Percepción/Reacción se divide en [5]:

- **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:

    - **Evaluación**: Es la fase de compresión de la situación, la interpretación de riesgo o peligro que se corre en un evento determinado. Comienza cuando finaliza el tiempo de percepción y termina una vez procesada la información. Gran cantidad de accidentes tienen como causas errores en esta etapa de evaluación. Su duración aproximada es de 0,5 segundos.
    - **Decisión**: Es la toma de elección de la maniobra más conveniente ante un evento. En esta etapa se resuelve si es conveniente modificar la velocidad, la dirección o la aceleración. El cerebro envía orden de ejecución al grupo de músculos correspondientes. La duración media de esta etapa es de 0,2 segundos.
    - **Distancia de reacción**: El tiempo de reacción depende de varios factores, entre otros son: La atención del conductor momentos antes de la frenada, el consumo de drogas, generalmente alcohol, que afecta aun en dosis mínimas, la edad y actividad física del conductor, la experiencia y pericia y el cansancio o sueño, entre otros. La distancia de reacción será la distancia recorrida en este intervalo de tiempo.

- **Tiempo de reacción mecánica**: Comienza al terminar la etapa de respuesta y finaliza cuando el vehículo empieza a responder a distintas acciones. Este tiempo corresponde al transcurrido desde que el conductor toma una decisión hasta que el vehículo se detiene. Su duración media es de 0,5 segundos.
- **Distancia de frenado**: La distancia de frenado es la distancia que recorre el vehículo hasta su detención completa. Esta varía según la calzada, carga del vehículo, los neumáticos, tipo de freno, la peripecia del conductor y la velocidad. El tiempo es de 0,5 a 0,15 segundos aproximadamente.
- **Distancia total o distancia de detección**: La distancia total o distancia de detención es la suma de la distancia recorrida durante el tiempo de percepción, más la distancia recorrida durante el tiempo de reacción, más la distancia de frenado. Cuando se duplica la velocidad en un vehículo, se duplica la distancia de reacción y se cuadriplica la distancia de frenado.

2.3.1.2 Factor mecánico

Este factor es el relacionado con el vehículo, que incluye situaciones como que el vehículo tenga alguna avería o que no responda adecuadamente. En este apartado podemos destacar 2 tipos de seguridades [6]:

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. Los principales son:

- **Neumáticos:** Debe evitarse la subida a bordillos o escalones, que pueden provocar deformaciones en las llantas y cortes o roturas en los neumáticos. Cualquier neumático con cortes profundos o deformaciones irregulares debe sustituirse.
- **Dirección:** Deben evitarse los golpes en las ruedas con bordillos o baches, pues pueden deformar la suspensión o la dirección.
- **Suspensión** Debe acudirse al mecánico para que realice una revisión de la suspensión cuando aprecien ruidos o golpeteos localizados en la parte baja del vehículo o exista un balanceo excesivo en curvas.
- **Frenos**: Debe revisarse semanalmente el nivel del líquido de frenos, hacer su sustitución cada dos años y revisar, al menos una vez al año, el estado de los discos de freno, latiguillos y bombines.
- **Alumbrado:** Obligación de llevar en el vehículo un juego de lámparas de repuesto.
- **Limpiaparabrisas**: No debe ponerse en funcionamiento el limpiaparabrisas con la luna seca ya que puede deteriorarla.

La seguridad pasiva: Los elementos de seguridad son aquellos diseñados para evitar o disminuir los posibles daños ocasionados a los ocupantes como consecuencia de un choque.

- **Carrocería**: Cumple la misión de proteger a los ocupantes, pues la deformación progresiva de la parte delantera y trasera de la carrocería permite absorber gran parte de la energía del choque.
- **Cinturón de seguridad:** La colocación correcta del cinturón permite salvar un gran porcentaje de vidas en los accidentes de tráfico.
- ***Airbag***: Es un sistema de protección que permite evitar lesiones más graves. La protección se consigue hinchando una bolsa de aire en milésimas de segundo, después del impacto.

- **Casco:** Dependiendo del tipo de vehículo puede ser obligatorio su uso. Su función principal es frenar la deceleración brusca que sufre el cerebro dentro de la cavidad craneal.
- **Reposacabezas**: Son los elementos fundamentales en la protección de la persona frente al latigazo cervical, siempre que se ajusten a la altura de la persona que vaya sentada.

2.3.1.3 Factores ambientales

La acción del conductor hay que situarla en un escenario real, soporte físico del sistema de tráfico; este no es otro que la vía y su entorno. Representa las exigencias a las que el conjunto conductor-vehículo debe responder y está configurado 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 los elementos "estables" del sistema podríamos considerar los siguientes [7]:

- **La calzada o vía**: incluyendo su planteamiento, construcción, trazado, pavimentación, anchura, resistencia al deslizamiento, número de carriles, la pendiente, el peralte, así como su explotación, mantenimiento y rehabilitación.
- **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.

    - **Lluvia**: Sobre el pavimento se forma una película lubricante que facilita el deslizamiento del vehículo. Debe reducirse la velocidad y aumentar el espacio entre vehículos. Después de circular por una vía mojada se deben recuperar los frenos, dando varios toques cortos y suaves.
    - **Niebla**: Se debe conectar el alumbrado de cruce y antiniebla. Se debe disminuir la velocidad y aumentar la distancia con el vehículo precedente.
    - **Viento**: Aumenta el riesgo de desplazamiento o vuelco. Se debe aminorar la marcha, sujetar firmemente el volante y extremar la atención en los pasos de zonas protegidas a desprotegidas.
    - **Hielo**: Supone una pérdida total de adherencia. El frenado deberá ser muy ligero. En caso de pérdida de control, no debe frenarse y debe levantarse el pie del acelerador y girar el volante hacia donde vaya la parte trasera del coche.
    - **Nieve**: Es necesario el uso de cadenas, realizar movimientos suaves, utilizando marchas altas. Es recomendable cuando la vía este cubierta por nieve, conducir por las marcas de los neumáticos de los vehículos que ya han pasado por esa vía, esto permitirá que tu coche circule mejor. Es recomendable circular por las marcas de neumáticos de los otros vehículos ya que facilita la adherencia.

2.4 IMPACTO DE LOS SIMULADORES EN LA EDUCACIÓN VIAL

Aunque a lo largo de la historia la industria del automóvil ha hecho grandes cambios en los vehículos, la base no ha cambiado, siguen teniendo ruedas y usando mayoritariamente combustibles fósiles. Aun así, la industria del automóvil es una de las que más ha evolucionado y de las que más ha intervenido en el desarrollo de otras tecnologías, como en nuestro caso, en los simuladores, los cuales permiten ahorrar grandes cantidades de dinero a la industria.

Actualmente los simuladores cada vez cobran más importancia en la industria. Muchos avances han sido desarrollados gracias a estos, puesto que la implementación real hubiera sido demasiado costosa para poder hacer pruebas o realizar cambios.

Los simuladores reducen tanto los gastos de ensayos y pruebas, como el tiempo de desarrollo de prototipos.

Cada vez que se diseña un componente, además de verificar que sus proporciones son correctas y que es posible el fabricarlo en serie, se realizan ensayos para comprobar todo tipo de cosas, desde su interacción con otros elementos, si gusta o no al público al que va dirigido, pruebas en las que se quiere saber su durabilidad, cómo se rompe y por qué, etc. Estas pruebas se conocen como los ensayos destructivos ya que, para su estudio, irremediablemente la pieza se rompe, y siempre es mejor realizar simulaciones que romper piezas físicas.

3 CONDUCCIÓN EFICIENTE.

El estilo de conducción influye, en gran medida, en el consumo de cualquier automóvil y determinadas costumbres aumentan el gasto de combustible, incluso provocan el deterioro prematuro de los vehículos. El exceso de agentes contaminantes en la atmósfera es uno de los mayores problemas a los que nos enfrentamos en la actualidad. Las emisiones nocivas que provienen de los coches son las causantes de los altos nivel de contaminación actuales. El transporte utiliza mayoritariamente combustibles fósiles, los cuales producen elevadas emisiones de CO_2. Este incremento provoca el llamado "efecto invernadero". Por ser estas emisiones las causantes de un gran porcentaje de la contaminación, cada vez se le da más importancia a la conducción eficiente, a la fabricación de vehículos ecológicos y a la sensibilización de la población [8].

3.1 VENTAJAS DE UNA CONDUCCIÓN EFICIENTE

Entre las ventajas de realizar una conducción eficiente están:

- **Mejora el confort de conducción y disminuye la tensión**: Para conducir de manera eficiente hay que evitar frenazos bruscos y acelerones, realizar el cambio de marchas de manera adecuada, es decir, sin dejar que las revoluciones sean muy grandes. De esta manera, los ruidos que proceden del motor se disminuyen. Este tipo de conducción lleva a la tranquilidad que evita los estados de estrés producidos por el tráfico con lo que se reduce el riesgo y la gravedad de los accidentes [9].
- **Ahorro económico de combustibl**e: El comportamiento del conductor influye sobre el consumo del carburante del vehículo. Hemos de tener especial cuidado al arrancar el coche o cuando utilizamos el acelerador. Debemos ser capaces de anticiparnos a las situaciones del tráfico con el fin de frenar lo menos posible. Mantener una velocidad adecuada y constante hará que el consumo se mantenga. Además, esto generará menos costes en el mantenimiento del vehículo (frenos, embrague, caja de cambios, motor, neumáticos,…), pues están sometidos a un esfuerzo menor.
- **Reducción de contaminación urbana que mejora la calidad del aire respirado**. La emisión de gases contaminantes se asocia a enfermedades respiratorias, problemas oculares, enfermedades cardiovasculares y jaquecas. La reducción de estos gases contribuye además a mejorar los problemas de calentamiento de la atmósfera.

3.2 ANÁLISIS Y CLASIFICACIÓN DE FACTORES

La energía petrolífera es agotable y aunque existen diversas fuentes de energías, ninguna puede suplir al petróleo en la actualidad. La constante reducción del consumo de los motores de los vehículos nuevos no impide el aumento constante del consumo global de energía en el sector de transporte, que es el primer consumidor a nivel nacional.

La demanda española de petróleo ha crecido enormemente desde 1965. El consumo de petróleo en España creció un 4,5% anualmente, casi el doble que la tasa mundial. En 1980, España, superaba por un 50% de consumo de petróleo a la media global [10].

Para minimizar esta dependencia de petróleo, se ha elaborado un modelo energético con una reducción del 23% del consumo de energía para el 2030. Un 70% de esta energía deberá provenir de fuentes renovables para el 2020 y el 100% en 2030 [10].

El transporte es el principal consumidor de energía final, con un consumo final de 41,3%. De este porcentaje, el 37,4% está destinado al transporte y el resto al conjunto de la industria, sector residencial, el comercio y los servicios. El principal problema al que se enfrenta el transporte es que cuenta con un alto coste económico y es muy dependiente del exterior, debido en gran parte a la importación del crudo.

El transporte por carretera representa cerca del 80% del total en el sector del transporte. El consumo de energía de los vehículos privados representa en España alrededor del 50% del total de los consumos del transporte por carretera. El porcentaje restante corresponde fundamentalmente al tráfico de mercancías (alrededor del 47%), y, con una participación mínima, al transporte colectivo de pasajeros (un 3%) [10]. En la figura 4 observamos el consumo de petróleo en los diferentes usos de los vehículos, los vehículos privados y los de mercancías son los que más consumen.

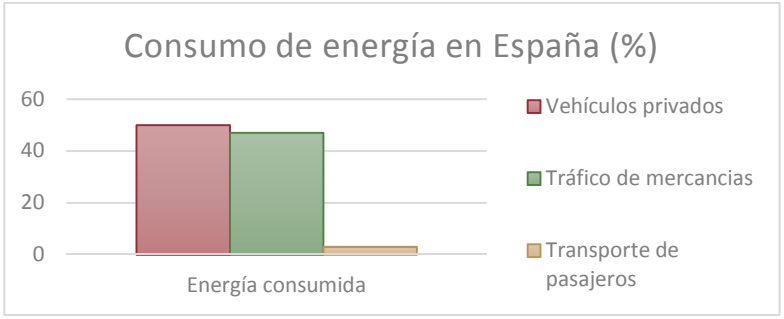

Figura 4: *Consumo de energía en España*

El continuo consumo de energía en el transporte provoca serias consecuencias económicas y sociales, tales como: el efecto invernadero, el ruido u otros daños al medio ambiente, atascos, accidentes y empobrecimiento de la calidad de vida y de los servicios. En la figura 5, vemos como como lo turismos de gasolina consumo en más combustible que los diésel.

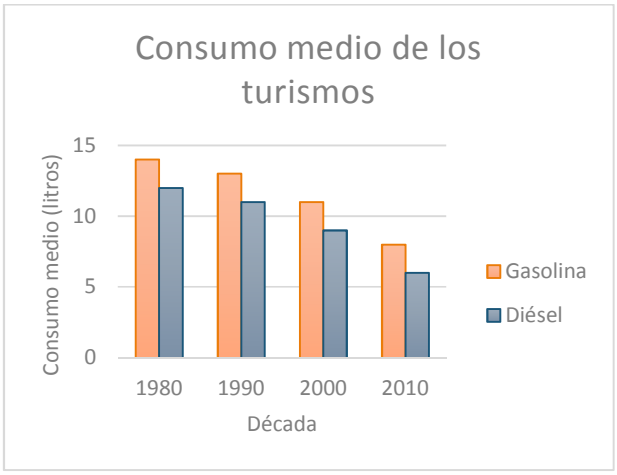

Figura 5: *Consumo medio de los turismos*

Observamos que en estas casi 4 décadas se ha reducido considerablemente el consumo de los motores de los turismos, por lo que el siguiente paso será el reducir el tipo de energía usada para el transporte.

En la tabla 2 podemos observar el porcentaje de las diferentes energías usadas en España en los años 2015 y 2016, siendo el petróleo casi la mitad de estas energías.

	2015		**2016**		**2016/2015**
	(ktep*)	Estructura (%)	(ktep)	Estructura (%)	Variación (%)
Carbón	1.515	1,8	1.340	1,6	-11,5
Productos petrolíferos	40.323	47,7	41.266	48,1	2,3
Gas	13.218	15,6	13.446	15,7	1,7
Electricidad	19.955	23,6	20.115	23,4	0,8
Energías renovables	5.287	6,2	5.385	6,3	1,9
• **Biomasa y otros**	*3.955*	*4,6*	*4.011*	*4,7*	*1,4*
• **Biogás**	*59*	*0,1*	*38*	*0,0*	*-35,1*
• **Biogás térmico**	23	0,0	23	0,0	-0,0
• **Calor útil de la cogeneración**	36	0,0	15	0,0	-57,4
• **Biocarburantes**	*977*	*1,2*	*1.023*	*1,2*	*4,7*
• **Solar térmica**	*277*	*0,3*	*293*	*0,3*	*5,8*
• **Geotérmica**	*19*	*0,0*	*19*	*0,0*	*3,1*
Total consumo final energético	80.299	94,9	81.551	95,0	1,6
Carbón	*0*	*0,0*	*0*	*0,0*	
Productos petrolíferos	*3.874*	*4,6*	*3.879*	*4,5*	*0,1*
Gas	*436*	*0,5*	*445*	*0,5*	*2,0*
Total consumo final no energético	4.310	5,1	4.324	5,0	2,1
Total	84.609	100,0	85.875	100,0	3,7

**** Ktep es una unidad de energía que significa kilo tonelada equivalente de petróleo***

Tabla 2: *Consumo de energía final*

La figura 6 muestra el consumo energético en España en el año 2015, los datos corresponden a los mencionados en la tabla 2.

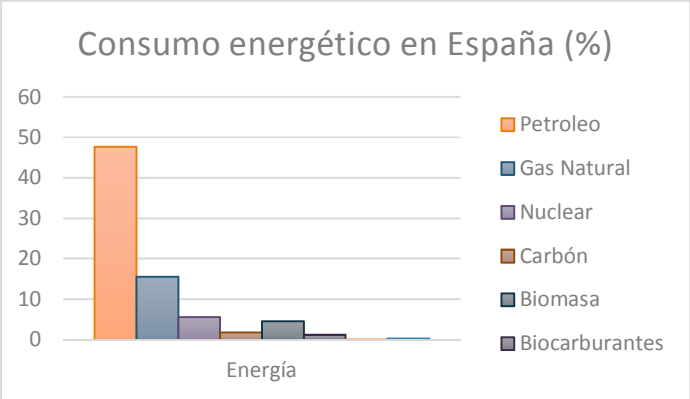

Figura 6: *Consumo energético de España*

3.3 IMPACTO EN EL CONSUMO

Un coche usado eficiencientemente puede ahorrar hasta un 50% de las emisiones de CO_2 con respecto a un coche ineficiente e inapropiado. Hay que tener en cuenta que un vehículo medio a lo largo de su vida puede emitir a la atmósfera más de 30 toneladas de CO_2. Además, obtendremos un ahorro considerable en el gasto de combustible y de emisiones contaminantes [10].

La Base de Datos de Consumos y Emisiones de Coches Nuevos, (a la que se puede acceder a través de la página web de IDAE, Instituto para la Diversificación y Ahorro de la Energía [11] directamente) está actualizada por los representantes de las marcas de coches. En esta página web podemos consultar la clasificación energética de cualquier vehículo que este en su base de datos. A partir de esta información se elabora el etiquetado comparativo (se comparan vehículos del mismo segmento o tamaño) y la clasificación de los vehículos nuevos con criterios de eficiencia energética. Vamos a comparar los diferentes modelos del vehículo que usaremos en nuestra simulación, el Lexus GS450H [11].

MODELO	CONSUMO	EMISIONES (GCO_2/KM)	CLASIFICACIÓN
LEXUS GS 250 NEUMÁTICOS 17P	8,9	207	E
LEXUS GS 250 NEUMÁTICOS 18P	9	209	E
LEXUS GS 250 NEUMÁTICOS 19P	9,1	211	E
LEXUS GS 300H NEUMÁTICOS 17P	4,4	104	A
LEXUS GS 300H NEUMÁTICOS 18P	5	115	A
LEXUS GS 450H	6,1	141	A
LEXUS GS 450H NEUMÁTICOS 18P	6,2	145	A
LEXUS[12] GS F NEUMÁTICOS 19P	11,2	260	G

Tabla 3: *Tabla comparativa de la clasificación de modelos Lexus GS en cuanto a emisiones*

La tabla 3 contiene información del consumo, las emisiones y la clasificación energética de los modelos de la gama *Lexus GS*. Observamos que, como era de esperar, los híbridos de gasolina (H) son los vehículos que mejor clasificación tienen, obteniendo una A [12]. En cambio, los de gasolina (250 y F) obtienen unas notas bastantes desfavorables, E y G respectivamente, dejando claro que en cuanto a emisiones y respeto con el medio ambiente, los coches de gasolina no serán el futuro. Existen ya diversos países que tienen fecha límite para la fabricación de este tipo de vehículos.

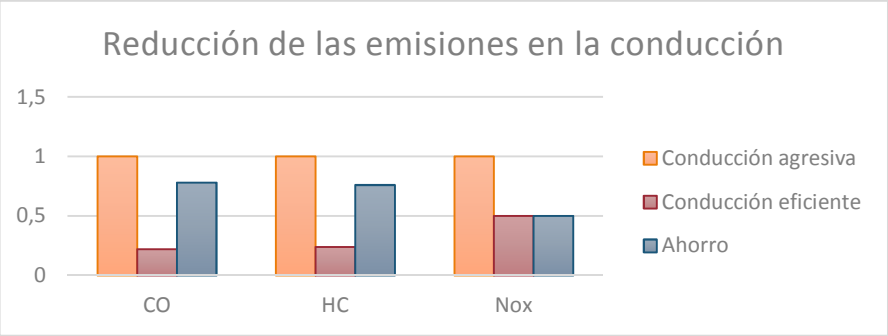

Figura 7: *Reducción de las emisiones en la conducción*

La figura 7 contiene información sobre la reducción de las emisiones en la conducción en cuanto a la forma de conducir (eficiente o agresiva). Podemos ver que una conducción eficiente puede llegar incluso a reducir el 78% de las emisiones de gases expulsadas por un vehículo. Observamos que el porcentaje de emisiones baja entre un 50% y un 78% dependiendo del tipo de motor (Monóxido de carbono, CO, Óxido de nitrógeno, Nox, Hidrocarburos, HC), por lo que la conducción eficiente es una gran ayuda para frenar el número de emisiones actuales a corto plazo.

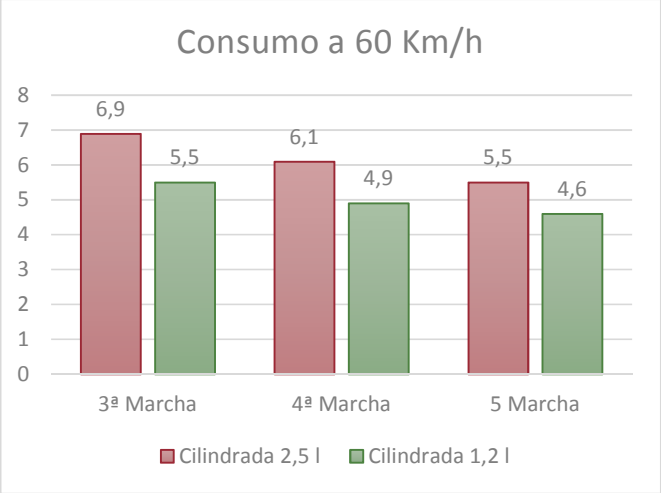

Figura 8: *Consumo a 60 Kph*

La figura 8 contiene una comparación del consumo a 60 kilómetros hora de las marchas tercera, cuarta y quinta. Observamos como a 60 km/h el consumo disminuye con marchas más largas, usando, por tanto, menos carburante y gastando menos recursos fósiles.

El etiquetado energético permite que el usuario considere la eficiencia energética como un factor más para tener en cuenta en la decisión de compra de un coche y al mismo tiempo, promueve el ahorro energético y la protección medioambiental.

3.4 ASPECTOS PRÁCTICOS

Hay que intentar circular con las marchas más largas posibles y procurar no frenar bruscamente para no tener que acelerar posteriormente. Vamos a mencionar algunos aspectos prácticos para los diferentes tramos de carreteras [13].

1) **En tramos con pendientes**:
   - Si son descendientes debemos levantar el pie del acelerador sin reducir de marcha y dejar bajar al coche por su propia inercia. si la aceleración no se mantuviera aceleraríamos lo justo para conseguir la velocidad de crucero pretendida.
   - Nunca hay que bajar la pendiente con el coche en punto muerto, pues además de incrementar el consumo y la contaminación, resulta extremadamente peligroso.
   - Si las pendientes fuesen ascendentes hay que procurar circular en la marcha más elevada posible, aunque tengamos que pisar más el acelerador.
2) **En las curvas**:
   - Antes de entrar en la curva hay que adaptar la velocidad del vehículo, se hará exactamente igual que en cualquier deceleración.
   - Levantaremos el pie del acelerador y dejaremos rodar el coche por su propia inercia. Si fuera necesario, reduciríamos a la marcha que precisemos para tomar la curva.
3) **En las glorietas:**
   - De la misma manera, en las **glorietas** tenemos que adaptar la velocidad del vehículo. Hay que realizar un reconocimiento a la entrada de la **glorieta** y anticiparse a las características de la misma:
     - Anchura de la calzada.
     - El tamaño de la glorieta
     - La existencia de otro vehículo en circulación o en espera.
4) **Conducción en caravana**:
   - Procuraremos por todos los medios circular en la marcha más larga posible que nos permita mantener la distancia de seguridad con los vehículos que nos preceden. Evitaremos en todo caso acelerar para después tener que volver a frenar, así evitaremos desgastes que no son necesarios para nuestro vehículo, además de ahorrar combustible y contaminar menos. Como consecuencia de esto, los vehículos que circulan detrás del nuestro podrán hacer lo mismo y habrá más fluidez en el tráfico.
5) **Paradas durante la marcha**.
6) Para paradas superiores a 60 segundos hay que apagar el motor. El coche parado funciona a ralentí y aunque el consumo no es muy alto, existe. Si se computan todas las paradas, el consumo es elevado. **Adelantamientos**:
   - Adelantar tiene que ser algo funcional, es decir, si adelantamos para ganar únicamente un puesto o dos, la ganancia en tiempo es prácticamente nula pero el consumo de combustible es alto.
   - El adelantamiento ha de ser siempre seguro, sin comprometer a otros conductores. Es importante que haya suficiente espacio y tiempo para realizarlo adecuadamente. Puede hacerse

a una velocidad relativamente alta y con marchas similares, si hemos elegido bien el momento y el lugar.

7) **Otras alternativas**:
   - Existe en Europa el firme propósito de reducir la contaminación que provocan los vehículos.

Dos puntos importantes además de la conducción eficiente para la reducción de la contaminación son:

- Por un lado, la creación de vehículos eléctricos que a pesar de que ya son parte de nuestro presente aún no se han instalado completamente. Estos coches provocan una menor polución, por lo que son una alternativa menos contaminante, y además son más silenciosos con lo que también consiguen reducir la contaminación acústica. Estos vehículos funcionan con baterías eléctricas que se alimentan con corriente eléctrica en lugar de gasolina/gasóleo. Para recargarlos solo necesitaríamos enchufar la batería a un enchufe convencional pudiéndose recargar en cualquier momento sin necesidad de tener que esperar a que la batería se cargue por completo.
- Por otro lado, los biocombustibles, que son combustibles obtenidos de manera renovable a partir de restos orgánicos. A menudo son mezclados con otros combustibles en pequeñas proporciones, como 5% o 10%, proporcionando un combustible útil pero limitado de gases de efecto invernadero. Los cuatro más conocidos son:
  - **Biodiesel**: Es un biocombustible que se fabrica a partir de grasa animal o aceites vegetales, que pueden ser usados o sin usar. Se suele utilizar girasol, soja etc. Puede ser puro o mezclado con gasoil en cualquier proporción en motores diésel.
  - **Bioetanol o etanol de biomasa**: es un alcohol que se obtiene a partir de maíz, sorgo, caña de azúcar o remolacha. Permite sustituir las gasolinas.
  - **Biogás**: Es el resultado de la fermentación de los desechos orgánicos.
  - **Biomasa**: Es la materia orgánica que se origina en un proceso biológico, espontáneo o provocado, utilizable como fuente de energía.

3.5 CONSEJOS PARA UNA CONDUCCIÓN EFICIENTE

Los consejos más importantes para lograr una conducción eficiente [14] son:

- **Arranque y puesta en marcha**: arrancar el motor sin pisar el acelerador. Iniciar la marcha inmediatamente después del arranque. En motores turboalimentados, esperar dos o tres segundos antes de iniciar la marcha [14].
- **Primera marcha:** Usarla solo para el inicio de la marcha. Cambiar a 2ª a los 2 segundos o 6 metros, aproximadamente.
- **Aceleración y cambios de marchas:** Acelere de forma progresiva, sin pisar el pedal a fondo. En motores de gasolina, cambie entre las 1.500 y 2.500 revoluciones; en los diésel, entre las 1.300 y 2.000. Visto desde el lado de la velocidad, cambia a 2ª marcha a los 2 segundos o 6 metros, aproximadamente. Cambia a 3ª marcha a partir de unos 30 km/h. Cambia a 4ª marcha a

partir de unos 40 km/h. Cambia a 5ª marcha por encima de unos 50 km/h. Hay que acelerar de forma ágil tras la realización del cambio.

- **Utilización de las marchas:** Circular lo más posible en las marchas más largas y a bajas revoluciones. Es preferible circular en marchas largas con el acelerador pisado en mayor medida que en marchas cortas con el acelerador menos pisado. El uso más eficiente del pedal acelerador tiene lugar entre el 50 y el 70% de su recorrido. En ciudad, siempre que sea posible, hay que utilizar la 4ª y la 5ª marcha, respetando siempre los límites de velocidad
- **Velocidad de circulación:** Mantenerla lo más uniforme posible; buscar fluidez en la circulación, evitando los frenazos, aceleraciones y cambios de marchas innecesarios.
- **Deceleración**: Levantar el pie del acelerador y dejar rodar el vehículo con la marcha engranada en ese instante. Frenar de forma suave con el pedal del freno. Reducir de marcha lo más tarde posible, con especial atención en las bajadas. Circulando por encima de unos 20km/h con una marcha metida y sin pisar el acelerador, el consumo de carburante es nulo. En cambio, a ralentí, el motor del coche consume entre 0,5 y 0,7 litros/hora.
- **Detención:** Siempre que la velocidad y el espacio lo permitan, hay que detener el coche sin reducir previamente de marcha. En otros casos, usar el freno de motor para reducir velocidad.
- **Paradas:** En paradas prolongadas (por encima de 60 segundos), es recomendable apagar el motor.
- **Pendientes:** En las subidas, conviene retrasar –en lo posible– la reducción de marchas y acelerar ligeramente. En las bajadas es más eficiente circular en marchas largas y rodar por inercia, evitando siempre cualquier situación de riesgo.
- **Anticipación y previsión:** Conducir siempre con una adecuada distancia de seguridad y un amplio campo de visión que permita ver 2 o 3 vehículos por delante. En el momento en que se detecte un obstáculo o una reducción de la velocidad de circulación en la vía, levantar el pie del acelerador para anticipar las siguientes maniobras.
- **Seguridad:** En la mayoría de las situaciones, aplicar las reglas de la conducción eficiente contribuye al aumento de la seguridad vial. Sin embargo, obviamente, existen circunstancias que requieren acciones específicas distintas, para que la seguridad no se vea afectada.

4 HERRAMIENTAS PARA LA CREACION DE VIDEOJUEGOS

4.1 CONCEPTO DE VIDEOJUEGO

Un videojuego es una aplicación orientada al entretenimiento que, a través de dispositivos denominados mandos, *gamepad*, *joystick* o controles, permite simular experiencias sensoriales y físicas. Los videojuegos se diferencian de otras formas de entretenimiento en que los usuarios deben involucrarse activamente con el contenido.

Las máquinas creadas específicamente para alojar videojuegos se conocen como *arcade* o recreativas. Hasta hace algunos años, era posible encontrarlas en muchos establecimientos como bares o salones de juego. Aunque con el avance de la tecnología, los videojuegos fueron dirigidos a videoconsolas y ordenadores.

Existen infinidad de videojuegos distintos entre sí, ya sea por su complejidad a la hora de jugarse o por su calidad de los gráficos, por lo que, al igual que en otros campos como el cine donde se clasifican las películas según géneros, podemos clasificar los videojuegos.

4.2 GÉNEROS DE VIDEOJUEGOS

Entre los diferentes géneros de videojuegos destacan:

- **Acción:** Este tipo de videojuegos requieren que el jugador haga uso de sus reflejos, puntería y habilidad, a menudo en un contexto de combate o de superación de obstáculos y peligros. Este amplio género se encuentra emparentado con otros géneros de gran popularidad, como *shooters* y plataformas, entre otros.
- **Plataformas**: Este tipo de videojuegos gira en torno a desafíos de tipo físico, que exigen un nivel de precisión para avanzar a través de la trama, ya sea para superar unas estructuras, o para poder enfrentarnos a un enemigo. El juego de plataformas por excelencia es Súper Mario Bros., desarrollado y publicado por Nintendo en el año 1985.
- ***Shooter***: Este tipo de videojuegos, como su nombre indica, se basan en disparar, aunque no por ello cabe referirse solo a los juegos que presentan armas, sino también a juegos donde el personaje lanza ataques, ya sea en forma de proyectil o de rayo (entre otras muchas posibilidades).
- **Rol Playing Game (RPG)**: Este tipo de videojuegos está dedicado al mundo del rol. Suelen confundirse con los juegos de aventuras, pero su foco son los personajes y su evolución a lo largo de la historia. Una de sus características principales es la utilización del término nivel para referirse al grado de experiencia de sus protagonistas y no a los diferentes mundos y escenarios que deben atravesar.
- **Simuladores:** Este tipo de videojuegos están dedicados a la simulación como su nombre indica. Existen diversos tipos de simuladores. Se podría decir que existe un simulador para cualquier campo que deseamos simular como vidas alternativas, simuladores de carreras o simuladores de cocina.
- **Aventuras gráficas**: Este tipo de videojuegos tiene la principal característica de que la historia es la protagonista. Deben ser construidos en base a una rica narrativa, que atrape poco a poco al jugador en el mundo, y le transmita la necesidad de resolver una serie de misterios. Tuvieron un

gran éxito a mediados de los 90, ya que, por aquellas fechas, los ordenadores no eran capaces de ejecutar juegos de alta definición o que consumieran gran cantidad de recursos. Este tipo de juegos era perfecto para aquellos ordenadores, ya que ofrecían grandes horas de entretenimiento sin requerir de muchos recursos.

- **Deportes**: Este tipo de videojuegos, como su nombre indica, hace alusión al mundo deportivo. Si bien su nombre parece decirlo todo, existe una línea muy delgada entre este género y el de simulación. Un juego de deportes refleja fielmente las reglas de la disciplina original, pero no a niveles milimétricos, sino que se vale de ciertas licencias, como avanzar más rápidamente el tiempo que en la realidad.
- **Estrategia**: Este tipo de videojuegos se caracterizan por la necesidad de manipular a un numeroso grupo de personajes, objetos, haciendo uso de la inteligencia y la planificación, para lograr los objetivos. Aunque existen diversos tipos, la mayoría son de temática bélica. Existen dos grandes subgéneros, estrategia en tiempo real (RTS, *real-time strategy*), y estrategia por turnos (TBS, *turn based strategy*).
- ***Sandbox***: Este tipo de videojuegos, muchas veces confundido con los mundos virtuales, son aquellos en los cuales se comienza prácticamente desde cero, creando prácticamente todo lo necesario para avanzar y transformar un mundo virtual propio. Son videojuegos no lineales, porque no tienen una línea de juego apenas definida, el orden de las acciones permite la mayor libertad.
- **Dragones y Mazmorras (*Dungeons & Dragons*, D&D)**: Este tipo de videojuegos derivado del mundo de dragones y mazmorras, es decir, rol puro, se basa en juegos de rol por asaltos, donde el jugador se enfrenta a los enemigos controlados por la inteligencia artificial. El jugador puede pausar el juego para tomar decisiones o indicar las acciones a tomar a los personajes. Estos personajes realizarán las mismas en un determinado número de asaltos y según la probabilidad (dados).
- **Rol Multijugador Masivo Online (*Massive Multiplayer Online Rol Playing Game*, MMORPG):** Este tipo de videojuegos rol multijugador masivo en línea son una variante de los juegos RPG donde el jugador podía crearse un personaje para desarrollar una aventura. Con la diferencia de que están centrados en un mundo multijugador, es decir, el jugador compite o es ayudado por otros jugadores.
- **Campo de batalla multijugador online (*Multiplayer Online Battle Arena*, MOBA)**: Este tipo de juegos de campo de batalla multijugador en línea se basa en dos equipos de jugadores que compiten entre sí. Cada jugador toma el control de un solo personaje a través de una interfaz de tercera persona, en un mapa simétrico, donde ambos equipos compiten por los mismos objetivos, como destruir la base del oponente.

Existen algunos géneros de videojuegos menos globales, como de cartas, de *point and click*, de *puzzles* etc., pero la lista presentada engloba los principales juegos del mercado y los que cuentan con el mayor número de usuarios.

4.3 MOTORES PARA PRODUCIR VIDEOJUEGOS

4.3.1 Unreal Engine 4

Unreal Engine 4 es una herramienta para el desarrollo de videojuegos hecha por y para desarrolladores de juegos. Desde videojuegos para dispositivos móviles en 2D hasta grandes títulos de consolas, *Unreal Engine 4* ofrece todo lo que se necesita para empezar, aprender y crecer en este campo.

Figura 9: *Entorno Unreal Engine 4*

La tecnología de *Unreal Engine* genera cientos de videojuegos además de películas en 3D, simuladores de entrenamiento, etc. En los últimos 15 años se han desarrollado miles de compañías dedicadas a desarrollar aplicaciones usando este motor. Podemos observar el entorno de *Unreal* en la figura 9.

4.3.1.1 Versiones

Se han presentado diferentes versiones de Unreal Engine a lo largo del tiempo. La primera, ***Unreal Engine 1***, hizo su aparición en 1998. Esta primera generación de *Unreal Engine* integraba *renderizado*, detección de colisiones, inteligencia artificial, visibilidad, opciones para redes y manipulación de archivos de sistema en un motor bastante completo.

La segunda versión ***Unreal Engine 2*** hace su aparición en 2002. Esta generación pasó por una reescritura completa del código del núcleo y del motor de *renderizado*. Muchos otros elementos del motor fueron actualizados, con mejoras, agregando soporte para *PlayStation 2*, *GameCube* y *Xbox*. Hubo una mejora de esta versión, ***Unreal Engine 2.5***, mejorando el rendimiento y agregando físicas para vehículos, editor de sistema de partículas para el *UnrealEd* y soporte 64-bits en 2004.

El ***Unreal Engine 3*** aparece en 2006, diseñado para PC con soporte *DirectX 9/10, Xbox 360 y PlayStation 3*. Su motor reescrito soporta técnicas avanzadas como alta definición de renderizado (HDRR), mapeo (*normal mapping*) y sombras dinámicas. Incluye componentes para herramientas complementarias al igual que las anteriores versiones del motor. En 2009 se publicó una versión gratuita del *Unreal Engine 3:* ***Unreal Development Kit*** para permitir a grupos de desarrolladores amateur realizar juegos

Desde 2015, 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. 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".

4.3.1.2 Videojuegos desarrollados con Unreal

En la tabla 4 podemos ver diferentes videojuegos desarrollados con Unreal Engine y las plataformas soportadas por estos [15].

Año	Titulo	Plataforma
2009	Borderlands	OS X, Microsoft Windows, PlayStation 3, Xbox 360
2012	Borderlands 2	Linux, OS X, Microsoft Windows, PlayStation 3, Xbox 360
2014	Borderlands 2 Vita	PlayStation Vita
2014	Borderlands: The Pre-Sequel	Linux, OS X, Microsoft Windows, PlayStation 3, Xbox 360
2015	Rocket League	Nintendo Switch
2016	Ark: Survival Evolved	Microsoft Windows, OS X, Linux, PlayStation 4, Xbox One, Steam
2016	Street Fighter V	Microsoft Windows, PlayStation 4
2015	Tekken 7	Arcade, Microsoft Windows, PlayStation 4, Xbox One

Tabla 4: *Tabla de videojuegos desarrollados con Unreal Engine*

4.3.2 Unity

Unity 3D es un motor de videojuego multiplataforma creado por *Unity Technologies*. *Unity* está disponible como plataforma de desarrollo para Microsoft Windows, OS X y Linux. La plataforma de desarrollo tiene soporte de compilación con diferentes tipos de plataformas. A partir de su versión 5.4.0, ya no soporta el desarrollo de contenido para navegador a través de su plugin web, en su lugar se utiliza *WebGL*. *Unity* tiene dos versiones: *Unity Professional* (pro) y *Unity Personal* [16]. En la figura 10 podemos observar como es el entorno de trabajo de *Unity 3D*.

Figura 10: *Entorno Unity 3D*

4.3.2.1 Características

Unity cuenta con gran variedad de compatibilidad de herramientas de diseño 3D (*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.**

Utiliza el motor gráfico *OpenGL* (para sistemas operativos de sobremesa, Windows, Mac y Linux), *Direct3D* (propio de Windows), *OpenGL ES* (para dispositivos moviles), e interfaces propietarias (Consolas como Wii).

Unity cuenta con las siguientes características:

- Soporte multiplataforma como se puede observar ver la figura 11, entre estas plataformas se incluyen las siguientes:.
  - Sistemas operativos de sobremesa (*Windows*, *Linux* o *iOS)*.
  - Dispositivos móviles *Android* (Actualmente version *8.0 Oreo*).
  - Soporte para consolas(*PS4*, *Nintendo WiiU*, *XBoxOne*, , la *Nintendo Switch*)
  - Dispositivos de realidad virutal, *Samgsum Gear VR*, *Google CardBoard* o *HTC Vive*, entre otros.
  - Plataformas de videojuegos como *Steam*.
  - Dispositivos de *Smart TV* como *tvOS*.
- Soporte para mapeado de relieve, mapeado de reflejos, mapeado por paralaje, oclusión ambiental en espacio de pantalla, sombras dinámicas utilizando mapas de sombras, *render* a textura y efectos de post-procesamiento de pantalla completa.
- Lenguaje *ShaderLab* para la creación de sombreadores. Pueden escribirse *shaders* en tres formas distintas: *Surface shaders*, *Vertex* y *Fragment shaders* o como *shaders* de función fija.
- El soporte integrado para *Nvidia*, el motor de física *PhysX* (a partir de Unity 3.0) con soporte en tiempo real para mallas arbitrarias y sin piel, *ray casts* gruesos, y las capas de colisión.
- El *scripting* viene a través de Mono. El script se basa en Mono, la implementación de código abierto de *.NET Framework*. Los programadores pueden utilizar *UnityScript* (un lenguaje personalizado inspirado en la sintaxis *ECMAScript*), *C# o Boo* (que tiene una sintaxis inspirada en *Python*).
- Compatible con *Visual Studio* para el desarrollo del código del videojuego, así como un editor de código propio.
- *I*ncluye *Unity Asset Server*, una solución de control de versiones para todos los *assets* de juego y *scripts*.

Figura 11: *Soporte multiplataforma de Unity*

4.3.2.2 Licencia:

La figura 12 contiene información acerca de las diferentes licencias que se pueden adquirir en *Unity*. Como se puede observar *Unity* cuenta con 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 *multiplayer* y análisis personalizables, esta versión es la usada por empresas que se dedican al desarrollo de aplicaciones en *Unity*.

Figura 12: *Comparativa de las diferentes licencias de 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. 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.

Figura 13: *Diferentes licencias*

4.3.2.3 Videojuegos desarrollados con Unity

En la tabla 5 se muestran algunos videojuegos que están desarrollados con Unity y las plataformas soportadas de cada uno de ellos.

Año	Titulo	Plataforma
2008	Three Kingdoms Online	Windows, Linux (Navegador)
2011	Battlestar Galactica Online	Windows, Linux (Navegador)
2013	Hearthstone: Heroes of Warcraft	Windows, Linux, iOS, Android
2014	Windows, Linux (Navegador)	Android, iOS
2015	Assassin's Creed Identity	Android, iOS

Tabla 5: *Tabla de videojuegos desarrollados con Unity 3D*

4.4 COMPARATIVA

Con la versión de *Unity 5* en el mercado y tras volverse *Unreal Engine 4* gratuito, nos surgen las siguientes preguntas: ¿cuál es mejor? ¿cuál es más eficaz a la hora de desarrollar? Vamos a compararlos en base a 2 criterios, desarrollo y economía.

4.4.1 Desarrollo

En el caso de *Unity3D*, el lenguaje de programación *C#* es mucho más sencillo que el lenguaje de programación *C++* usado en *Unreal*, ya que dispone de algunas librerías que te facilitan muchísimo la labor de la programación y es más rápido de programar, sobre todo a base de *scripting*.

Por el lado opuesto, tenemos otra gran evidencia que da un gran punto de ventaja a *Unreal Engine* y es que, aunque tardes más en hacer el desarrollo, es más óptimo y gasta menos recursos.

Por otro lado, existe un *plugin* externo en *Unity* denominado *PlayMaker*, el cual ya tiene una versión gratuita, para una persona que no tenga un nivel avanzado de programación o que esté empezando.

En cuanto a desarrollo, ambas herramientas se encuentran bastante a la par.

4.4.2 Economía

Esta parte suele ser muy importante ya que necesitamos conocer cuánto queremos invertir en nuestro software. Por supuesto, cuanto más nos ahorremos, ya sea para uso personal como para uso empresarial, mejor.

Para empezar, *Unity3D* sigue teniendo una licencia de *Pro* (la cual es de pago) y otra gratuita. En la gratuita, al alcanzar 100.000$ debes cambiarte a la licencia Pro. En la licencia Pro, también tenemos sub-pagos. Además, si necesitamos implementar en dispositivos *Android*, *iOS* o Videoconsolas, necesitamos pagar por estos servicios. Podemos comprar la licencia con un solo pago (1500€) o en mensualidades, pero cuando se actualice la versión no necesitaremos comprar ninguna licencia. Por el contrario, si pagamos la licencia total, al actualizar la versión no tendríamos versión de pago.

Unreal Engine, por otro lado, directamente es gratuito, pero, al ganar 3.000€ o más de beneficio, ellos se llevarán un 5% de lo que ganes.

En el caso de una desarrolladora grande que gana mucho, por ejemplo, la desarrolladora de *Clash of Clans* (un juego para dispositivos móviles del que se filtró su ganancia, ascendiendo a más de cinco

millones de dólares) un 5% de los beneficios es bastante superior a los 1500€ que deberíamos pagar por *Unity3D* o por los pagos mensuales de *Unity3D*, en definitiva.

4.5 ELECCIÓN DE LA HERRAMIENTA DE NUESTRO PROYECTO

A la hora de realizar nuestro proyecto necesitamos elegir la herramienta de desarrollo que usaremos para crearlo. Entre las posibilidades existentes, nos centraremos en dos de ellas (las dos que hemos tratado anteriormente). Se puede observar que los juegos desarrollados en Unreal tienden a ser de mayor calidad gráfica. Esto no se debe a la herramienta, ni a que *Unreal* haga juegos de gran calidad gráfica y *Unity* no, sino a que *Unreal* tiene una mayor precisión y variedad de desarrollo que se adapta mejor a estas características. Por ejemplo, a la hora de desarrollar una cinemática, *Unity 3D* no posee un elemento o característica que te permita indicar por dónde circula la cámara, sin embargo, *Unreal* sí.

Este, a primera vista, parece un detalle insignificante, pero si observamos que la mayoría de los *triple A* (Juego de nombre en el mercado) de la actualidad, ya no se centran tanto en desarrollar historia y tramas, como hace algún tiempo (las cuales eran lo mejor del juego), sino que son sagas ya asentadas en el mercado, con historia ya definida y semejante entre ellos, en los cuales lo que prima de una entrega a otra es desarrollar una mayor calidad visual. Por esto mismo, entenderemos que los desarrolladores prefieran *Unreal* por este elemento extra.

En nuestro caso, nos decidiremos a utilizar *Unity* por varias razones:

- La primera de ellas es que el grupo de investigación que acoge este proyecto trabaja con esta herramienta. Por ello, es el principal recurso que nos puede ofrecer para desarrollar nuestro proyecto. La segunda de ellas, y también muy importante, es que, aunque para la generación de este escenario de conducción se haya partido de cero, se dispone de una serie de *Asset* de conducción y otros proyectos anteriores del grupo, los cuales se basan en el mismo material usando los mismos recursos y herramientas. Por lo que, a la hora de reutilizar algo o usarlo de apoyo, es beneficioso utilizar la misma herramienta.
- Otra característica que se aleja de las otras dos, las cuales están relacionadas con el entorno de trabajo, es que para nuestro simulador no necesitamos las calidades gráficas excepcionales que nos ofrece Unreal, sin desprestigiar a *Unity en este aspecto.* Unity es una herramienta más fácil de utilizar y conocer teniendo en cuenta que el desarrollador no es diseñador gráfico.

5 HERRAMIENTAS PARA EL DISEÑO 3D

5.1 INTRODUCCIÓN

Antes de explicar qué es y cómo se generan, mencionar que una imagen en 3 dimensiones (3D) se diferencia de una imagen en 2 dimensiones (2D) principalmente por la forma en que ha sido generada.

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 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.

5.2 PASOS DE MODELADO

5.2.1 Modelado

El **modelado** consiste en dar forma a objetos individuales, los cuales se usarán posteriormente en las escenas de la aplicación. Las dos técnicas de modelado más usadas son:

- **Modelos representados por polígonos**: En la actualidad, posiblemente el sistema más utilizado por ordenador para representar un objeto en 3D. Todos los objetos generados están constituidos por muchos polígonos formando una malla, por ejemplo, un cubo está formado por 6 cuadrados, mientras que un balón de fútbol se compone de 12 pentágonos y 20 hexágonos. Naturalmente, cuando mayor sea el objeto generado, más compleja será su composición, como una catedral.
- **Modelos definidos por sus curvas matemáticas**: Actualmente, hay otros sistemas de modelado donde el usuario trabaja con superficies curvas definidas matemáticamente en lugar de polígonos, es decir, se trabaja con la función matemática de la circunferencia en lugar de polígonos.

5.2.2 Sombreado

Como su nombre explica, la fase de **sombreado** se refiere a dotar de sombras y luces al objeto en 3D. Existen diversos tipos de sombras. Gracias a este paso, se consigue un resultado similar la realidad de una forma eficaz, ya que se elimina una sombra falsa entre el suelo y el objeto. El problema de estas sombras es que su cálculo ralentiza bastante.

5.2.3 Texturizado

Muchos objetos no pueden definirse con un único color superficial. Ya queramos definir el terreno del suelo, la madera de los muebles o el estampado de una camisa, necesitamos diferentes colores con una

distribución a veces geométrica y otras completamente azarosa. Por eso recurrimos a las texturas. Un ejemplo de textura se puede observar en la figura 14.

Figura 14: *Textura de madera*

Se escaneamos un trozo de madera y guardamos la imagen resultante, podemos aplicarlo a cualquier objeto. De esta forma, el objeto parecerá que es de madera (depende también de la calidad de la imagen). Esto es lo que entendemos por textura. Este tipo de textura se conoce como textura *Bitmap* (mapa de bits). Si la calidad de la textura no es muy buena, podremos apreciar los píxeles de la imagen. Existen sistemas de texturizado para evitar este problema, los cuales aportan diversos beneficios:

- Resolución siempre óptima (evitando ver píxeles).
- Imitación de la naturaleza (corteza de un árbol o las llamas).
- En ningún momento percibimos fenómenos de repetición.
- Los cálculos que el ordenador tiene que realizar son más rápidos que cuando se aplica un mapa de bits muy grande.

Existen 4 procedimientos básicos para aplicar una textura:

- **Planar**: Si aplicamos este sistema en un objeto veremos que en la cara donde intervenimos aparece la textura perfectamente definida, pero en las adyacentes aparece proyectada longitudinalmente.
- **Cúbico**: Si tenemos que texturizar un armario, lo haríamos mediante una aplicación cúbica, proyectándose la textura en las 6 direcciones de las caras de un cubo.
- **Cilíndrico**: Si queremos ponerle la etiqueta a una botella de vino usaremos una proyección cilíndrica.
- **Esférico**: para aplicar la textura de los mares y continentes a la bola terrestre, este sería el procedimiento idóneo.

5.2.4 Animación y Rigging

La **animación** consiste en dotar de vida a los objetos creados. Estos no tienen por qué ser un ser vivo, también podemos aplicar movimiento y animación a un trozo de césped para similar el efecto del aire. Aun así, el cambio más significativo en cuanto a animación equivale a dotar a personas y animales con movimiento de forma que podamos expresar sus sentimientos y comportamiento, como podemos ver en la figura 15.

Figura 15: *Animación y rigging de diferentes personajes*

Para esto último, entra en escena un elemento denominado ***rigging***, es decir, la preparación del personaje para que sea animado construyendo un esqueleto con capacidad de moverse, deformarse y cambiar de expresión según nuestras necesidades.

5.2.5 Renderizado

El proceso de **renderizado** se desarrolla con el objetivo de generar, en un espacio 3D formado por estructuras poligonales, una simulación realista del entorno, del comportamiento tanto de luces, texturas y materiales (agua, madera, metal, plástico, tejidos, etc.) como también de los comportamientos físicos (animación).

Renderizar es un proceso que hay que hacer siempre al finalizar la edición. Siempre que se modifica algo, aunque el cambio sea mínimo hay que renderizar.

5.3 APLICACIONES

A continuación, mostramos ejemplos de los diferentes usos del diseño 3D llevados a cabo.

5.3.1 Cine

El realismo de los elementos creados es algo que solo la impresión 3D es capaz de conseguir. Se ha optado por esta tecnología para, por ejemplo, la creación de objetos utilizados por los protagonistas de alguna de las películas reciente de cine como *Guardianes de la Galaxia, El señor de los anillos* o *Star Wars*.

Figura 16: *Película "La novia cadáver" de Tim Burton desarrollada por ordenador en 3D*

En la película de animación de *Tim Burton*, La novia cadáver (figura 16), la impresión 3D fue la gran protagonista. Ayudó a la creación de los personajes y escenarios, cada uno de ellos tenía cientos de expresiones y características distintas.

5.3.2 Videojuegos

La industria de los videojuegos se basa y rige por los elementos 3D para poder generar productos. En la actualidad está ganando importancia la realidad virtual (*virtual reality*, VR) con la que podemos "sumergirnos" dentro del propio videojuego.

5.3.3 Arquitectura

El hecho de tener la capacidad de diseñar y saber utilizar programas de modelado en 3D es una gran ventaja para cualquier diseñador. Este tipo de software permite generar una percepción más amplia sobre lo que se está creando y agrega realismo al momento de presentar las opciones a los clientes.

El modelado en 3D permite generar ilustraciones fotorrealistas y maquetas que son convincentes, así como ampliar el conjunto de habilidades para los planos de manualidades para los productos y los diseños ambientales, además de permitir presentar diseños de 360 grados de envases, propuestas de muebles y s hasta de edificios enteros para los arquitectos. Podemos observar un ejemplo en la figura 17.

Figura 17: *Desarrollo de una habitación por ordenador en 3D*

5.3.4 Diseño industrial

Actualmente, el diseño en 3 dimensiones está evolucionando muchísimo, introduciendo nuevas mecánicas al diseñar, desarrollar y fabricar productos. Un gran ejemplo de ellos son las nuevas tecnologías de impresión 3D, las cuales nos permite fabricar ya en casa desde un pequeño tornillo (figura 18) hasta un coche de juguete totalmente funcional.

Figura 18: *Desarrollo de un tornillo en 3D*

5.4 HERRAMIENTAS PARA EL DISEÑO EN 3D

5.4.1 Autodesk

Autodesk 3ds Max es un software de creación de gráficos y animación 3D desarrollado por *Autodesk*, en concreto la división *Autodesk Media & Entertainment* (anteriormente *Discreet*). Creado inicialmente por el *Grupo Yost* para *Autodesk*, salió a la venta por primera vez en 1990 para *DOS* [17].

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 [18].

Podemos ver el entorno de *Autodesk* en la figura 19 y el de *Autodesk Maya* en la figura 20.

Figura 19: *Entorno de Autodesk*

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 [19].

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, etc.

Además, *Maya* es el único software de 3D acreditado con un premio 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.

Figura 20: *Entorno de Autodesk Maya*

5.4.2 Blender

Blender es una herramienta para la creación de modelos 3D gratuita y de código abierto. Soporta todos los pasos de la creación 3D: modelado, *rigging*, animación, simulación, renderizado, composición y movimiento, además de edición de video y generación para la creación de videojuegos [20].

Blender es un programa informático multiplataforma, dedicado especialmente al modelado, iluminación, renderizado, animación y creación de gráficos tridimensionales. También sirve para la composición digital utilizando la técnica procesal de nodos, edición de vídeo, escultura (incluye topología dinámica) y pintura digital. En *Blender*, además, se pueden desarrollar videojuegos, ya que posee un motor de juegos interno.

Blender es un proyecto público creado por cientos de personas de todas las partes del mundo, por estudios y artistas personales entre otros. 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*. Podemos ver el entorno de *Blender* en la figura 21.

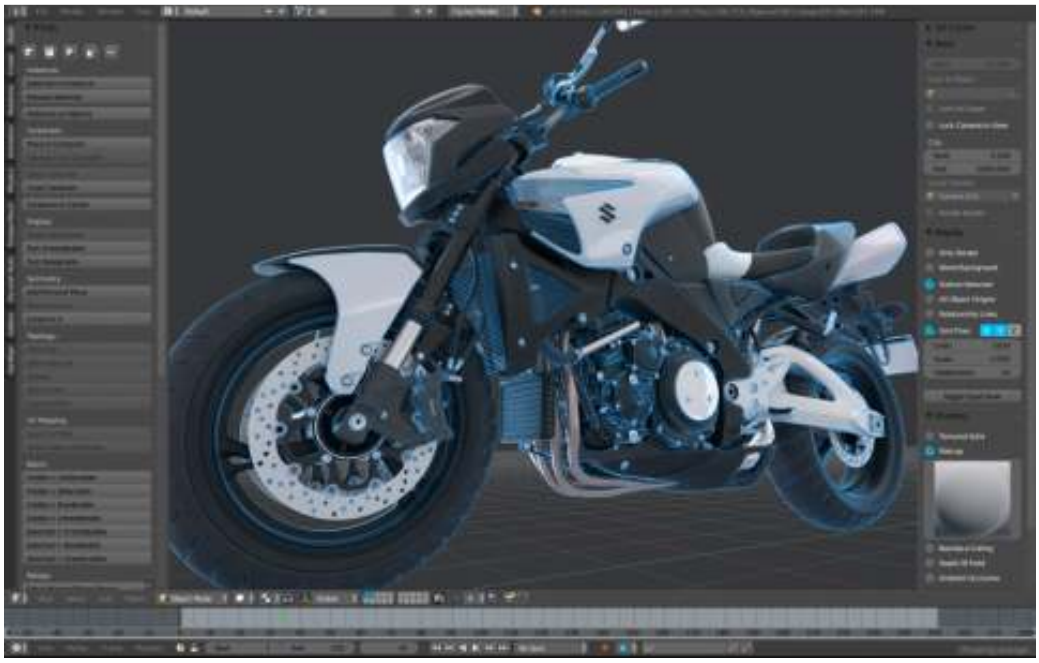

Figura 21: *Entorno de Blender*

5.4.3 SketchUp

SketchUp (anteriormente *Google-SketchUp*) es un programa de diseño gráfico y modelado en tres dimensiones (3D) basado en caras. Para entornos de arquitectura, ingeniería civil, diseño industrial, diseño escénico, videojuegos o películas. Es un programa desarrollado por *@Last Software*, empresa adquirida por *Google* en 2006 y finalmente vendida a *Trimble* en 2012 [21].

Hay una razón por la que *SketchUp* es sinónimo de un software amable e indulgente de modelado en 3D. No sacrificamos la facilidad de uso por la funcionalidad. Comienza dibujando líneas y figuras. Empuja y tira de las superficies para convertirlas en figuras en 3D. Estira, copia, gira y pinta para construir lo que quieras.

Su principal característica es poder realizar diseños en 3D de forma sencilla. El programa incluye, entre sus recursos, un tutorial en vídeo para aprender paso a paso cómo se puede diseñar y modelar el propio entorno. Permite conceptualizar y modelar imágenes en 3D de edificios, coches, personas y cualquier objeto o artículo que imagine el diseñador o dibujante, además de que el programa incluye una galería de objetos, texturas e imágenes listas para descargar.

Figura 22: *Entorno de SketchUp*

5.5 COMPARATIVA

En la tabla 6 observamos una pequeña comparación entre los dos productos de Autodesk y Blender. Blender, a diferencia de los otros 2, es una herramienta de trabajo gratuita y que contiene mayor número de idiomas disponibles, sin embargo, Autodesk se hizo un hueco en la industria, aun no contando con estas características.

	AUTODESK 3DS MAX 9	**AUTODESK MAYA 8.5**	**BLENDER 3D 2.45**
Primer mercado	Viz - Games	Games - Film	VIZ - Real-time
Precio para el usuario final	5,00 €	2,50 €	FREE
Suscripción anual	Sí	Sí	No
Plataformas	Windows	Windows, MacOS, Linux	Windows, MacOS, Linux
Entorno personal de aprendizaje (PLE)	No	Sí	No disponible
Periodo de prueba	Sí	Sí	No disponible
Idiomas	Francés e inglés	Inglés	Muchos
Cambio de idiomas	No	No	Todos incluidos
Uso en la industria	Muy bueno	Muy bueno	Bajo
Tecnología	Antigua	Antigua	Antigua, renovándose

Tabla 6: *Comparación de las diferentes herramientas de modelado 3D [22].*

6 SIMULADORES

6.1 INTRODUCCIÓN

Un simulador es una herramienta, de carácter general informático, que reproduce un sistema. Los simuladores reproducen sensaciones y experiencias que en la realidad pueden llegar a suceder [23].

Un simulador pretende reproducir tanto las sensaciones físicas (velocidad, aceleración, percepción del entorno) como el comportamiento de los equipos de la máquina que se pretende simular. Para simular las sensaciones físicas, se utilizarán complejos mecanismos comandados por potentes ordenadores que mediante modelos matemáticos consiguen reproducir sensaciones de velocidad y aceleración. Para reproducir el entorno exterior, se emplean proyecciones de bases de datos de terreno, también denominado entorno sintético.

6.2 TIPOS DE SIMULADORES

Existen diferentes tipos de simuladores:

- **Simulador de conducción**: permiten a los alumnos de autoescuela enfrentarse con mayor seguridad a las primeras clases prácticas, además de permitirles practicar de manera ilimitada situaciones específicas (aparcamientos, incorporaciones desde posiciones de escasa visibilidad, conducción en condiciones climatológicas adversas, etc.).
- **Simulador de carreras**: es el tipo de simulador más popular; se puede conducir un automóvil, motocicleta, camión.
- **Simulador de vuelo o de aviones**: permite dominar el mundo de la aviación y pilotar aviones, helicópteros, etc.
- **Simulador de vida o de dinámica familiar**: permite controlar una persona y su vida.
- **Simulador de negocio**: permite representar un entorno empresarial. Es posible jugar diferentes roles dentro de las funciones típicas de un negocio.
- **Simulador político**: permite representar un entorno político.
- **Simulador de redes**: permite simular redes. Ejemplos: *Omnet++, ns2, ns3* (usado en una asignatura de Grado).
- **Simulador clínico médico**: permite realizar diagnósticos clínicos sobre pacientes virtuales. El objetivo es practicar, con pacientes virtuales, casos clínicos, bien para practicar casos muy complejos, preparando al médico para cuando se encuentre con una situación real, o bien para poder observar cómo un colectivo se enfrenta a un caso clínico, para poder sacar conclusiones de si se está actuando correctamente, siguiendo el protocolo de actuación establecido.
- **Simulador musical**: permite reproducir sonidos con un instrumento de juguete.

6.3 ELEMENTOS DE UN SIMULADOR PARA LA CONDUCCIÓN

¿Qué podemos hacer en un simulador? Aquí se abre un abanico que podemos decir que es casi infinito. Podemos utilizar un simulador para conducir por placer sin ninguna "prisa" o para pilotar a nivel competitivo. Pero esto no es lo único, en el mundo de la simulación puedes desde diseñar tu coche hasta retransmitir una carrera, pasando por conducir el coche fantástico. Las posibilidades son muchas.

Estos simuladores te acercan a la experiencia de conducir a gran velocidad, tal y como lo hacen las estrellas del automovilismo.

Preparar una pista o una carrera no es cosa fácil. Muchos menos conseguir un *auto ad hoc*. Pero eso no quiere decir que no puedas vivir la experiencia de conducir a gran velocidad, incluso desde la sala de tu casa. Los siguientes elementos de los simuladores nos permiten vivir con mucho realismo la experiencia de competir.

6.3.1 Logitech G27 Racing Wheel

El dispositivo usado en nuestro proyecto será el *Logitech G27 Racing Wheel* para PC, podemos ver la apariencia de este dispositivo en la figura 23. 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 [24].

Figura 23: *Logitech G27 Racing Wheel*

El *G27* destaca por la poderosa fuerza de su mecanismo y su sistema de *Forcé Feedback* con motor dual para simular con precisión toda la acción de las carreras. El *G27* viene con un set de tres pedales de acero (aceleración, freno y embrague) resistentes que ofrecen una experiencia muy real.

La palanca de cambios cuenta con 6 velocidades. El volante forrado manualmente con piel y suave al tacto cuenta con un par de paletas en la parte posterior para hacer los cambios en caso de que no domines la palanca de velocidades. Su precio es de 150 dólares, alrededor de 130 €.

6.3.2 Logitech Driving Force GT

El dispositivo *Logitech Driving Force GT*, que se puede observar en la figura 24, cuenta con una palanca de cambios de 6 marchas integrada en el módulo del volante, donde tenemos 3 componentes de dirección además de 8 botones. Este volante cuenta además con la característica de 900 grados de giro. El *Driving Force GT* es compatible con PlayStation 2, PlayStation 3 y PC [25].

Figura 24: *Logitech Driving Force GT*

Cuenta con las siguientes características:

- **Disco de ajuste en tiempo real con 24 posiciones**: Ajuste al instante y con toda precisión de los frenos, el sistema de tracción y otras funciones, para un control sin precedentes del rendimiento del coche.
- Driving Force GT cuenta con una rotación de **900 grados del volante**: Dos giros y medio del volante, como en los coches de verdad.
- **Tecnología *Force Feedback:*** Transmite la sensación de cada centímetro de carretera para posibilitar un control máximo y ofrecer una experiencia automovilística inigualable. Los motores del *Driving Force GT* cuentan con una novedosa tecnología para aumentar el realismo en los juegos de conducción.
- **Pedales de freno y acelerador**: Respuesta precisa con pedales de aceleración y freno de increíble realismo.
- **Palanca de cambio secuencial**: Marcha a marcha para medir mejor sus movimientos.
- **Aro recubierto de goma**: Sujeción cómoda durante carreras intensas.
- **Volante macizo de una sola pieza**: Conducción con la confianza de que el volante no crujirá ni se deformará durante maniobras agresivas.
- **Optimizado para los juegos más populares**: Se ve y siente en el juego cada movimiento del volante, sin interferencias ni demoras.
- **Codificación óptica de máxima precisión**: Fiabilidad y precisión una carrera tras otra.

Como desventajas de este volante, hay que mencionar los pedales sin embrague y poco realistas, ya que se puede apreciar que están hechos de plástico y no dan efecto de una conducción realista.

Su precio es de 120 dólares, alrededor de 105 €.

6.3.3 CXC Motion Pro II

Forbes calificó a este simulador como el mejor en cuanto a experiencias de conducción *indoor*. *CXC Simulations*, fundada por Chris Considine, es la casa creadora de este y otros simuladores. Pilotos como Lewis Hamilton, Patrick Longs y Graham Rahal han realizado pruebas en sus simuladores e incluso han impartido cursos virtuales a través de ellos [26]. Podemos observar este simulador en la figura 25.

Figura 25: *CXC Motion Pro II*

El *Motion Pro II* accede a las pistas y circuitos de *iRacing, Assetto Corsa*, *Project CARS* y *rFactor*, con lo que se tiene una librería bastante amplia. Además, ofrece opciones de conectividad para participar en carreras en tiempo real contra corredores en otros países.

La diferencia entre el *Motion Pro II* y una consola común radica en su cabina (asiento, pedales, volante, palanca) que entrega una experiencia sumamente real. Por si fuera poco, cuenta con tres pantallas de 55 pulgadas con sonido *surround* y un sistema de movimiento muy preciso.

Su precio es de 50000 dólares, alrededor de 45000 €.

6.3.4 Most Realistic Racing Simulator

Pareciera un poco pretencioso, pero este es considerado el simulador más realista y por ello Hammacher Schlemmer no tuvo otra opción que nombrar *Most Realistic Racing Simulator* a este simulador. *Ford Motor Company* usa este sistema para hacer pruebas y ofrecer experiencias de manejo [27]. Podemos observar este simulador en la figura 26.

Figura 26: *Most Realistic Racing Simulator*

El simulador usa servo-actuadores lineares que causan un efecto de suspensión, un chasis monocasco de fibra de vidrio que genera la sensación de rodamiento, paradas y rotación 360º a 5G de aceleración. Este simulador reproduce fielmente las condiciones de manejo y movimientos que se sentirían en, digamos, una curva a 200 Km/h.

El volante cuenta con palancas de cambios en la parte posterior, tiene pedales de aceleración, freno y *embrague* y tiene un sistema de *feedback-force* que genera fuerzas *10X* que te hace pensar que realmente estas en las pistas que podrás apreciar en HD en el monitor triple de 108 pulgadas. Tiene un repertorio de 12 vehículos de F1, GT y F3 que podrás oír a través de un sistema de sonido de 500 vatios de potencia.

Su precio es de 120000 dólares, alrededor de 105000 €.

6.4 SIMULADORES

A continuación, vamos a detallar diversos simuladores que se encuentran en el mercado [28].

6.4.1 SIMESCAR

El *Simescar* es un simulador de conducción de automóvil que, junto con nuestra herramienta de software *Socrates*, podemos observarlo en la figura 27, cumple dos objetivos principales [29]:

- Formación de nuevos conductores mediante prácticas esquematizadas y la repetición de ejercicios específicos para el desarrollo de buenos hábitos de conducción.
- Concienciación para conductores noveles o experimentados mediante prácticas de conducción a gran velocidad, consecuencias de las distracciones al volante, y consecuencias de la conducción bajo los efectos del alcohol.

Figura 27: *Simulador Simescar*

6.4.1.1 Características del software:

Las características principales del software de Simescar son:

- Posibilidad de interacción en tiempo real con la simulación mediante el puesto de Instructor.
- Simulación de conducción en estado de embriaguez para concienciación en Seguridad Vial.
- Variedad de escenarios: cruces, autopista, ciudad, puerto de montaña, glorietas, pista de pruebas y adelantamientos.
- Variedad de puntos de inicio y de entornos específicos dentro de cada escenario seleccionado.
- Conducción con condiciones meteorológicas con intensidad modificable: soleado, nocturno, amanecer, atardecer, lluvia, niebla y tormenta.
- Parametrización de intensidad de agresividad del resto de coches y/o peatones del entorno.
- Gestión de usuarios y simuladores gracias a la herramienta de software *Socrates.*

6.4.1.2 Características del hardware:

Simescar cuenta con 3 modelos de simuladores. Estos modelos se detallan a continuación.

Modelo SIMESCAR AMBAR – SMCAMB, este modelo se puede observar en figura 28.

- Controles y mandos originales de automoción: llave de arranque con 4 posiciones, freno de mano de palanca, manetas de luces y limpiaparabrisas, claxon, pedales y luces de emergencia.
- Cinturón de seguridad con sistema de bloqueo.
- Palanca de cambios manual de 5 velocidades. Posibilidad de conducir con transmisión automática durante la simulación.
- Tres monitores de 24” de HD para campo de visualización de 135°.
- Sonido 2.0 con salida para auriculares.

- Carcasa resistente de madera y metal con una estructura modular que permite su fácil manipulación y transporte.
- Posibilidad para plegar las pantallas laterales y separar el simulador en diferentes módulos independientes.

Figura 28: *Simulador Simescar* ***AMBAR*** *– SMCAMB*

Modelo SIMESCAR SILVER – SMCSLV, este modelo se puede observar en la figura 29.

- Columna de dirección con *force feedback* y volante de turismo real.
- Pedalera de coche turismo real con motores de vibración integrados en cada pedal para recrear efecto de *ABS* y punto de fricción del embrague.
- Manetas de vehículo real: bocina, intermitentes, luces y limpiaparabrisas integrados en columna de dirección.
- Cinturón de seguridad con sistema de bloqueo.
- Palanca de cambios manual de 5 velocidades.
- Tres monitores de 32” de *HD* para campo de visualización de 135º y un monitor de 10” para cuadro de instrumentación.
- Posibilidad de incorporar una plataforma de movimiento *2DOF* que recrea los movimientos de alabeo y cabeceo para generar sensaciones de aceleración, frenada, vibración y paso por curva.
- Sonido envolvente 5.1, con posibilidad de incorporar auriculares.
- Carcasa resistente de fibra.

Figura 29: *Simulador Simescar* ***SILVER*** *– SMCSLV*

Modelo SIMESCAR BRONZE – SMCBRZ, este modelo se puede observar en la figura 30.

- Columna de dirección con *force feedback* y volante de turismo real
- Pedalera de turismo real con motores de vibración integrados en cada pedal para recrear efecto de ABS y punto de fricción del embrague.
- Manetas de vehículo real: bocina, intermitentes, luces y limpiaparabrisas integrados en columna de dirección.
- Cinturón de seguridad sensorizado con sistema de bloqueo.
- Palanca de cambios manual de 5 velocidades.
- Tres monitores de 32" de *HD* para campo de visualización de 135º y un monitor de 10" para cuadro de instrumentación.
- Posibilidad de incorporar una plataforma de movimiento *2DOF* que recrea sensaciones de aceleración, frenada, vibración y paso por curva.
- Sonido envolvente 5.1, con posibilidad de incorporar auriculares.
- Carcasa resistente de fibra.

Figura 30: *Simulador Simescar* ***BRONZE*** *– SMCBRZ*

6.4.2 DriveSim

DriveSim es un simulador de conducción para autoescuelas, centros de formación, cursos de prevención de riesgos laborales o cualquier tipo de evento. Este programa de simulación ha sido especialmente diseñado como una herramienta educativa para conductores [30].

Este simulador de conducción nos va a permitir practicar en un entorno seguro, absolutamente libre de riesgos. Además, podremos repetir los ejercicios tantas veces como deseemos y conocer cuáles han sido nuestros errores durante la conducción gracias a un sistema de corrección adaptado a la normativa de diferentes países. Podemos ver las características de *DriveSim* en la figura 31.

Figura 31: *Características de DriveSim.*

Se trata de un software de simulación que ha sido especialmente diseñado como herramienta educativa. Cada vez más centros de todo el mundo apuestan por esta novedosa formación para sus conductores: una forma de aprender y practicar interactiva, divertida y, lo más importante, eficiente. Un simulador para autoescuelas con el que los alumnos aprenden a la vez que se divierten.

DriveSim es un simulador de conducción completo que nos va a permitir experimentar multitud de situaciones y repetirlas tantas veces como queramos. *DriveSim* es perfecto para ser utilizado en eventos de seguridad vial, cursos de prevención de riesgos laborales o cursos de amaxofobia. Podemos ver su apariencia en la figura 32.

Figura 32: *Entorno DriveSim.*

6.4.3 iRacing

iRacing es el líder de la simulación online de carreras. Desarrollado desde el comienzo para centrarse en las competiciones y el Racing, *iRacing* organiza *hosts* de usuarios y pistas oficiales en carreras alrededor de todo el mundo. Utilizando las últimas tecnologías para recrear nuestra línea de expansión de los coches de carreras y los circuitos [31].

La generación de software de simulación de *iRacing* es usada por pilotos profesionales además de usuarios y jugadores aficionados. Aun así, *iRacing* es un simulador de conducción puro. Si estás buscando la mejor competición de *racing* online o prefieres competir contigo mismo y el cronómetro, *iRacing* es una buena elección.

Para comenzar, todo lo que necesitas es un ordenador, un controlador para la conducción (vistos en apartados anteriores) y una conexión a Internet. Podemos ver su apariencia en la figura 33.

Figura 33: *Entorno iRacing*

6.4.4 Project CARS

Project CARS es una galardonada serie de juegos de carreras, en la que los mejores coches correrán bajo condiciones extremas para ofrecerte la mejor experiencia de conducción [32].

Creado por jugadores, probado por pilotos de primera categoría del equipo *SMS-R*, es la opción definitiva de los profesionales de *eSports*. *Project CARS* captura la esencia de las carreras reales en el juego de carreras más hermoso, intenso, auténtico y técnicamente más avanzado del planeta.

Project CARS 2 cuenta con todo tipo de vehículos (como GT, turismos, prototipos de resistencia o exóticos supercoches) para ofrecerte la gama más completa de carreras automovilísticas y una libertad total para elegir qué quieres conducir y dónde quieres hacerlo, en cualquier momento y lugar. Podemos ver su apariencia en la figura 34.

Figura 34: *Entorno Project Cars 2*

Sus principales características son:

- Más de 170 coches licenciados por las mejores marcas.
- La mayor lista de circuitos jamás vista en un videojuego de carreras, con circuitos de hielo y tierra.
- Nuevos tipos de vehículos y clases automovilísticas, como *Rallycross*, *IndyCar* y *Oval*.
- Cambios dinámicos de horas del día, condiciones climáticas y estacionales.

- Nuevo modo de campeonatos en línea.
- *LiveTrack 3.0* con condiciones dinámicas de superficies que afectarán al rendimiento y el manejo del vehículo, y que irán modificando el circuito durante el fin de semana de una carrera.
- Pensado para eSports desde el minuto cero, con clasificación completa y funcionalidad de transmisiones en directo.
- Física de neumáticos de última generación, inteligencia artificial (*artificial intelligent*, AI) avanzada y control intuitivo de mando.
- Compatibilidad con *RV*, 12K, 21:9 y triple pantalla.

6.5 RFACTOR

rFactor es indudablemente uno de los mejores simuladores de coches que podemos encontrar, si lo que queremos es una simulación minuciosamente realista y un apartado gráfico fiel, también, a la realidad de los vehículos. Está centrado en recrear dinámica, acústica y otros apartados con el máximo realismo [33].

Uno de los detalles clave de este título es que, más allá de los contenidos incluidos en él, podemos encontrar decenas de MODs con modelos adicionales con los que disfrutar en las carreras, o circuitos en los que experimentar sus sensaciones de conducción. Nos ofrece también la posibilidad de conducir con condiciones meteorológicas activas, que evolucionan en tiempo real y es uno de los únicos que cuenta con eventos de 24 horas en modo multijugador con equipo de asistencia. Este simulador requiere una suscripción para poder disfrutar de él. Podemos ver su apariencia en la figura 35.

Figura 35: *Entorno rFactor*

7 DESARROLLO DEL PROYECTO

7.1 PLANIFICACIÓN

En la planificación del proyecto hemos destacado varias fases de desarrollo:

1. **Análisis**: En esta fase detallaremos qué hacer, cómo hacerlo, con qué herramientas de desarrollo de software, qué limitaciones tendrá nuestro simulador y qué objetivos tendremos que conseguir con este proyecto.
2. **Diseño**: En esta fase, diseñaremos el escenario, tanto el vehículo a utilizar como los objetos utilizados.
3. **Implementación**: En esta fase programaremos el comportamiento del escenario, con la programación de las funciones utilizadas para la conducción e inteligencia artificial.
4. **Pruebas**: En esta fase se comprueban las funcionalidades implementadas y con la ayuda de varios usuarios se toman valores de diferentes simulaciones realizadas.
5. **Documentación:** En esta fase se documenta el trabajo realizado, cómo se ha realizado y demás información necesaria para entender el objetivo del Trabajo Fin de Grado.

7.2 DESARROLLO

7.2.1 Resumen

El proyecto se basará 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 desarrollaremos un escenario de conducción por donde conducirá 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 36. Este tramo abarcará desde la E.T.S. de Ingenieros Industriales hasta alrededores de la Plaza San Juan.

Figura 36: *Mapa de Valladolid representado en el escenario*

Para ello reproduciremos a escala esta zona de la ciudad (salvo por elementos estéticos, ya que no disponemos de los mismos modelos) con los diferentes cruces de carreteras, por donde el usuario deberá conducir.

Figura 37: *Ruta de la simulación.*

En esta ruta, mostrada en la figura 37, 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 la opción de realizar una ruta guiada, con esta opción, al acercarse a una intersección se mostrará el camino a seguir. Si el usuario, por los motivos que sean, no va a la dirección indicada, la ruta se recalculará.

7.2.2 Género

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.

Dentro de este género, pertenece al subgénero de simuladores de conducción. Un simulador de conducción es un videojuego en el que se imitan competencias de los vehículos y la conducción real. El objetivo de este tipo de simuladores es el de recorrer una distancia o ruta hacia un determinado sitio, con el extra de una serie de normas, en nuestro caso, las reglas físicas y de conducción.

El propósito de estos simuladores de conducción es imitar de manera realista a vehículos reales. Estos pueden calcular el recorrido físico de la suspensión, el trabajo del motor o la fricción de los neumáticos con la carretera.

Año tras año, estos juegos simulan mejor a la realidad, por lo que compiten entre ellos para lograr el mayor realismo posible. Se necesita un dispositivo de conducción, es decir, un volante para mayor realismo.

7.2.3 Audiencia

La creciente implantación de los videojuegos en la actualidad no ha estado exenta de controversias, por las críticas arrojadas, por sus escenas o contenido, por parte de los medios de comunicación. Los principales consumidores de videojuegos son los menores de 25 años.

En cuanto al contenido nuestro videojuego, este no presenta escenas censuradas ni contenido ilegal, por lo que, en cuanto a este ámbito, no existe una edad restringida para su uso.

En cuanto a los conocimientos necesarios para su uso, se necesita poder usar un volante, por lo que cualquier persona que pueda usarlo podrá realizar simulaciones. Partiremos, por lo tanto, de una edad de 8 años para poder usarlo.

Por otro lado, nuestro videojuego está orientado a la realización de pruebas de conducción, siendo sus usos el aprendizaje en las autoescuelas, sin la necesidad de tener que realizar las pruebas en coches reales, y la implementación de prototipos realizados por empresas de automoción. Para este último caso no habría restricciones, ya que sería para un uso interno y privado de la empresa. Por el contrario, para el uso del simulador en las autoescuelas, cualquier persona que tenga edad para poder sacarse el carnet de conducir podrá usar este simulador con el escenario desarrollado. Actualmente puedes empezar las clases en las autoescuelas a la edad de 16 (aunque necesites permiso paterno y no puedas presentarte al examen práctico hasta los 18).

Una vez vistos todos los aspectos, podemos catalogar nuestra aplicación para una edad recomendada de 16 años (para el uso educativo) en autoescuelas y edad mínima de 8 años (para uso personal).

7.2.4 Flujo

La aplicación comienza cuando se ejecuta el fichero “.exe”, tras esto se mostrará el menú principal. 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 38.

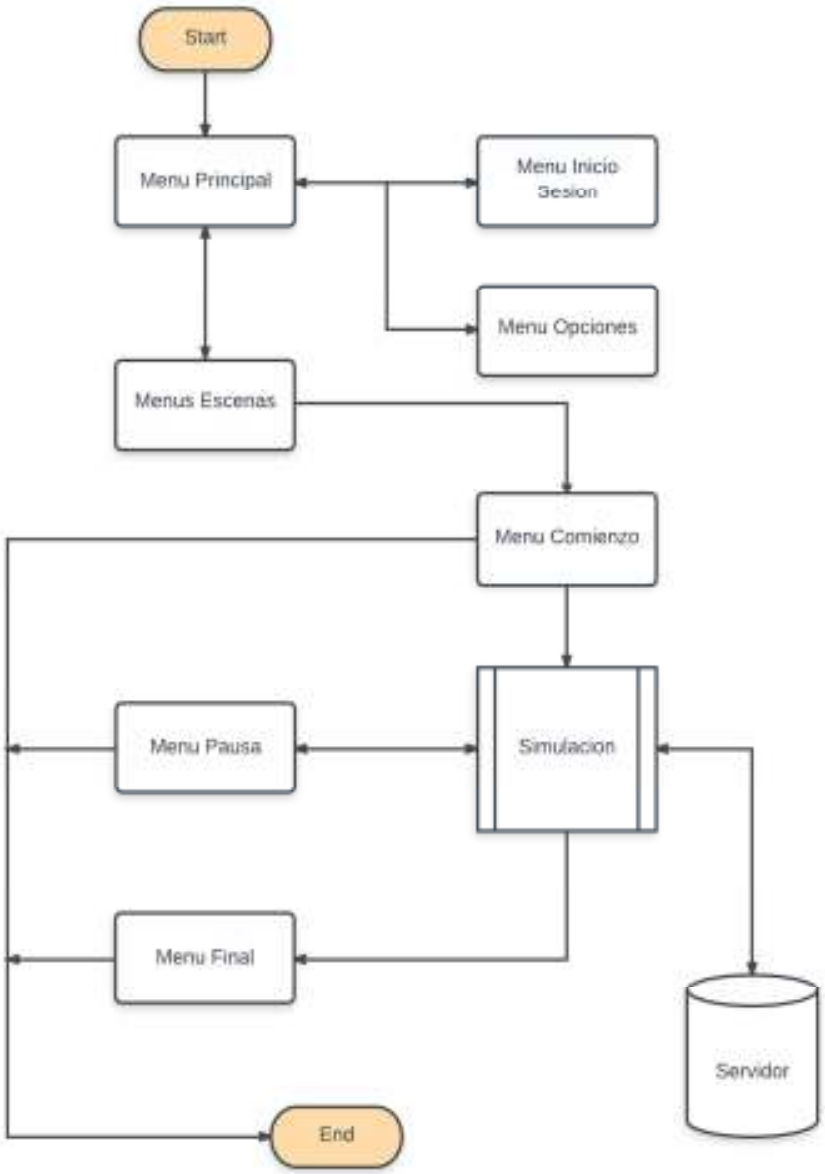

Figura 38: *Diagrama de flujo de la aplicación*

7.2.5 Apariencia

La apariencia de nuestro proyecto deberá mostrar realismo y semejanza con el tramo diseñado. Dicho esto, tenemos que decir que, ya que no somos diseñadores gráficos, no dispondremos de los mismos modelos, como edificios, señales, vallas, bancos, etc. que existen en el escenario real.

La importancia de reproducir un tramo de ciudad es la reproducción del mismo conjunto de vías, para que, de esa forma, el usuario pueda realizar un recorrido del mismo modo que en la realidad, dando así sensación de realismo y, lo que es más importante, logrando que la sensación de conducción sea la mejor posible. Podemos observar la apariencia de nuestro escenario en desarrollo en la figura 39.

Figura 39: *Apariencia del escenario*

7.3 IMPLEMENTACIÓN

7.3.1 Organización

Nuestra aplicación contiene una gran cantidad de apartados: modelos 3D para el entorno de la simulación, realización de *scripts* para la aplicación de las leyes físicas, el comportamiento del vehículo, la inteligencia artificial, el cálculo de parámetros o las transiciones entre pantallas, las diferentes interfaces de menús y conexión con el servidor donde se enviarán los datos de la simulación.

Al ser un proyecto tan amplio de desarrollar, no hemos definido una cronología de los diferentes cambios a lo largo del tiempo, ya que cada elemento a desarrollar tiene varios factores implicados, por ejemplo, realizar un semáforo para la circulación necesitará de un modelo 3D, unas texturas básicas para diferenciar sus estados, un *script* para poder cambiar el estado del semáforo, necesitaremos realizar otro *script* para detener los vehículos de inteligencia artificial que circulen por la vía y, por último, detectar si el usuario ha pasado cuando el semáforo estaba en rojo.

Aun así, es posible dividir la estructura del desarrollo proyecto en las siguientes secciones, en las que explicaremos cómo se ha ido desarrollando la aplicación:

- Interfaces
- Jugador
- Otros usuarios
- Escenario
- Cámara
- Scripts

7.3.2 Pantallas del juego (Interfaz)

7.3.2.1 HUD

En los videojuegos, se denomina *HUD* (visualización cabeza-arriba) 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 40.

Figura 40: *Interfaz HUD del escenario*

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

- **Velocímetro**: Es un indicador que mide el valor de la velocidad media del vehículo, 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.
- **Reloj cuentarrevoluciones**: Es un 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**: Estos indicadores indican 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.
- **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.
- **Frenos**: El freno de disco es un sistema de frenado usado normalmente para ruedas de vehículos. Este indicador se encenderá al frenar el vehículo.

7.3.2.2 Menu Principal

Al iniciar la aplicación, se mostrará un menú principal (figura 41), el cual dispondrá de las siguientes opciones:

- **Escenarios**: En esta sección se accederá a otro menú que nos mostrará los escenarios en los que se podrá conducir.
- **Opciones**: En esta sección podremos modificar las opciones del vehículo, como el tipo de tracción y las opciones del escenario como la iluminación o el cielo.
- **Identificación**: En esta sección el usuario podrá validarse en la aplicación para recoger datos acerca de las simulaciones que realice. Estas estadísticas se podrán observar en la página web del servidor
- **Menús de simulación**: Los otros tres menús a desarrollar son los menús de dentro de la simulación. Estos dos menús serán los siguientes:
    - **Menú de inicio**: Esta sección será la mostrada inicialmente al usuario al iniciar la simulación. Este menú contiene opciones para la configuración de un sistema de sensores corporales, 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 trabajo).
    - **Menú de pausa**: Este menú será el mostrado al pausar el jugador la simulación, contendrá los detalles de infracciones realizadas por el usuario hasta el momento, así como el consumo del vehículo utilizado. El jugador podrá reanudar la marcha.

Figura 41: *Menú principal del videojuego*

7.3.2.3 Menú Opciones

En este menú, figura 42, el usuario podrá modificar el comportamiento del escenario y el vehículo podrá definir el tipo de marchas (automático, manual o levas), el tipo de tracción (tracción delantera FWD, tracción trasera RWD o tracción a las cuatro ruedas AWD), el cielo (soleado, ocaso o noche), así como elegir si prefiere la opción de modo guiado (donde se mostrará la ruta a seguir) o la notificación de las infracciones (se mostrará un mensaje por pantalla cuando se cometa una infracción).

Figura 42: *Menú de opciones del videojuego*

7.3.2.4 Menú de Escenarios

En este menú, figura 43, el usuario podrá elegir la escena que desea simular. Actualmente solo se encuentra una escena, pero está implementado para más por si se quiere añadir otras escenas.

Figura 43: *Menú de escenarios del videojuego*

7.3.2.5 Menú de Identificación

En este menú, figura 44, el usuario podrá identificarse en la aplicación. Deberá estar registrado en el servidor para poder acceder.

Figura 44: *Menú de identificación de la sesión del videojuego*

7.3.2.6 Menú de Pausa

Este menú, figura 45, es el mostrado cuando se pausa la simulación. Contiene la información sobre las infracciones y sobre el consumo. También contiene botones para poder continuar la simulación o finalizarla.

Figura 45: *Menú de pausa de la simulación del videojuego*

7.3.2.7 Menú Comienzo

Este menú, figura 46, será el mostrado al inicio de la simulación. Se usará para sincronizar los sensores fisiológicos usados para monitorizar al conductor durante las simulaciones.

Figura 46: *Menú de comienzo de la simulación del videojuego*

7.3.3 Jugador (vehículo)

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 47.

Figura 47: *Diferentes modelos de la gama Lexus GS 450h*

El modelo del vehículo en Unity que usará el jugador se puede ver en la figura 48.

Figura 48: *Modelo de la gama Lexus GS 450h que usara el jugador*

Este modelo implementará las características del vehículo mostradas en la tabla 7 [34]:

CARACTERÍSTICA	ESPECIFICACIÓN
Máxima potencia	345
Tracción	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 NOX (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	5,9
Coeficiente de resistencia aerodinámico (CD)	0.27
Peso en vacío (Kg)	1920
Peso máximo admisible (Kg)	2335

Tabla 7: *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. Para observar cómo configurar el vehículo, se puede consultar el Anexo referente al *Realistic Car Controller V3*.

7.3.4 Otros usuarios (IA)

La inteligencia artificial está implementada gracias a la ayuda del *Asset intelligent Traffic System* (iTS), el cual nos permite implementar rutas. Para conocer más detalles sobre este apartado, podemos ir al Anexo correspondiente.

En nuestro caso, hemos implementado un sistema de tráfico basado en líneas y conectores, los cuales recorrerán los diferentes tramos urbanos de la simulación.

En cuanto a la inteligencia artificial (AI), hemos desarrollado unos pocos tipos de vehículos (automóviles, autobús, bicicletas) y peatones, que recorrerán de manera autonómica el escenario. Estos elementos interferirán en la conducción, provocando que el usuario deba adaptar su forma de conducir a la de los demás usuarios de la vía. A continuación, podemos observar los diferentes vehículos:

- **Autobús**: Este vehículo se implementará por la ruta que transitan los autobuses urbanos de la ciudad. Podemos ver el modelo en Unity en la figura 49.

Figura 49: *Modelo de autobús usado en el escenario*

- **Peugeot Bipper:** Este vehículo será uno de los dos principales que representarán automóviles. Podemos ver el modelo en Unity en la figura 50.

Figura 50: *Modelo de la gama Peugeot Bipper usado en el escenario*

- **Peugeot 207:** Este vehículo será uno de los dos principales que representarán automóviles. Podemos ver el modelo en Unity en la figura 51.

Figura 51: *Modelo de la gama Peugeot 207 usado en el escenario*

- **Ciclistas:** Estos usuarios de la vía recorrerán tanto carreteras como tramos de carriles para bicicletas. Podemos ver el modelo en Unity en la figura 52.

Figura 52: *Modelo de ciclista usado en el escenario*

- **Peatones:** Serán los viandantes de la simulación. Podemos ver los modelos en Unity en la figura 53.

Figura 53: *Modelo de peatones usados en el escenario*

7.3.5 Escenario

El escenario realizado es un tramo de la ciudad de Valladolid, el cual está escalado para poder realizar todos los cruces que existen en la realidad. En cuanto a los edificios y demás escenografía, hay que tener en cuenta que, aunque el escenario está intentando simular la realidad, al no disponer de los modelos de los edificios, estos no guardan similitud con los de la ruta real. Aun así, la intención de este escenario es el estudio de la conducción y la escenografía queda, por lo tanto, en segundo plano.

Figura 54: *Mapa virtual en Unity 3D*

7.3.6 Cámara

En nuestro proyecto tendremos diversas cámaras desarrolladas en Unity.

Figura 55: *Componente Cámara*

La cámara principal de la simulación será la cámara interior del vehículo (*Hood Camera*), situada en la posición del conductor del vehículo. Esta cámara nos aportara realismo y el poder simular la sensación de conducción como si fuéramos el conductor real del vehículo

Desde la cámara podremos observar la interfaz *HUD* 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 56: *Visión del conductor*

Para los retrovisores necesitaremos aplicar al espejo (*Mesh*) la textura deseada. Una vez asignada, seleccionamos la cámara que deseemos que se muestre en el espejo y en el parámetro *Target Texture* indicamos la textura de renderizado que aplicamos previamente al espejo. Con esto conseguiremos que en el espejo se muestre la visión de la cámara. Puesto que el modelo de nuestro vehículo no tiene espejos independientes, hemos optado por agregar un plano en el espejo donde proyectamos la cámara.

7.3.7 Scripts

En este proyecto hemos creado una gran cantidad de scripts para muchas funciones, como pueden ser la interacción entre los diferentes vehículos de la vía, la configuración de nuestro propio vehículo, el cálculo y notificación de las diferentes infracciones cometidas a lo largo de la simulación o el envío de información al servidor que aloja lo datos.

Para crear un Script seleccionamos Assets **> Create > C# Script.**

Existen diversas funciones básicas de *Unity que* nos servirán para poder seleccionar distintos componentes de un objeto, así como *scripts*, también dentro de objetos hijos contenidos en el objeto padre [35].

A continuación, explicaremos cómo se han desarrollado los *scripts* más influyentes en nuestro proyecto.

7.3.7.1 Infracciones

El script Infracciones.cs se centrará en la correcta manipulación y almacenamiento de las diferentes infracciones en una estructura de datos. También almacenará los datos de la conducción. En la figura 57 observamos la declaración de estas dos estructuras. La primera de ellas 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.

```
// 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 57: *Script infracciones*

7.3.7.2 Consumo

El script Consumo.cs calculará el consumo del vehículo del jugador. Este cálculo dependerá del tipo de marcha y velocidad que tenga el vehículo. En la figura 58, vemos las gráficas de consumo con relación a la velocidad [36].

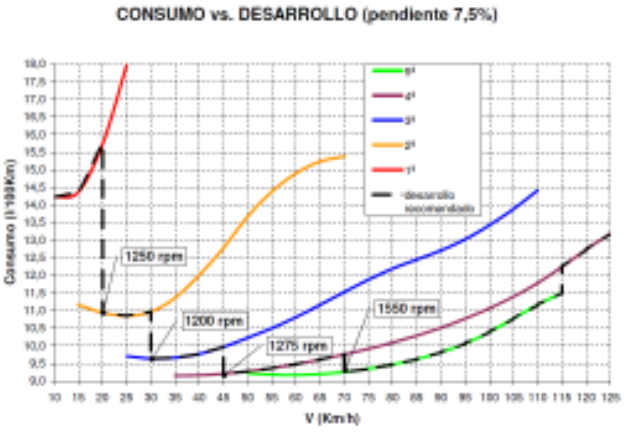

Figura 58: *Consumo de las diferentes marchas*

En la gráfica podemos observar (líneas discontinuas) cuál sería la velocidad para cambiar de marcha de manera más eficiente. Siguiendo este esquema conduciríamos de forma más eficiente y, por lo tanto, gastaríamos menos combustible.

Una vez vista la gráfica de las curvas de consumo tenemos que implementarlas. Cada marcha se puede representarse como una ecuación matemática (en nuestro caso usaremos ecuaciones de rectas, ya que son fáciles de calcular y reflejan el consumo que necesitamos calcular). Introducimos un apartado en el *script* donde calculamos el consumo dependiendo de la marcha actual del coche.

7.3.7.3 Sensores

Usaremos una serie de funciones de Unity, que permiten calcular cuándo un objeto entra en contacto con otro (*Colliders*).

Gracias a esta serie de funciones, podemos detectar cuándo el jugador u otro vehículo entra, sale o simplemente está dentro de una zona. A esta funcionalidad, añadimos la de los *Tags*.

Los *Tags* son una serie de etiquetas que podemos asignar a cualquier objeto. Un objeto padre puede tener una etiqueta mientras que cualquier de los objetos hijos contenidos en él pueden tener etiquetas diferentes. Por último, hay que destacar que podemos "buscar" dentro de una jerarquía de objetos si existe algún objeto hijo ya sea por nombre, etiqueta o *script* contenido en alguno de ellos.

De esta forma, y combinando estas tres características (*Collider*, *Tags* y *GetComponent*), se han desarrollado una serie de *scripts* que permiten desde detectar si has chocado con otro objeto (como otro vehículo), si te has saltado una ceda el paso (porque existía otro vehículo cruzando la intersección que tenía preferencia) o si te has saltado un semáforo que estaba en rojo.

Vamos a detallar alguno de los comportamientos de estos scripts:

- **Semáforos**: En la figura 59, se muestra el comportamiento del semáforo con paso de cebra, como podemos observar, el *Prefab* (este término se refiere a una agrupación de objetos que se pueden reutilizar en bloque) está compuesto de 4 "sensores". El primero de ellos (1) sirve para hacer parar a los vehículos que lleguen al semáforo. Los sensores (1) y (2) contienen código para determinar, a su vez, si el usuario ha pasado el semáforo en rojo, notificando la infracción. El sensor (3) activa una variable si algún peatón se encuentra en el paso de cebra. Por último, los sensores (1) y (4) obligan a los vehículos de IA a parar si algún peatón está cruzando.

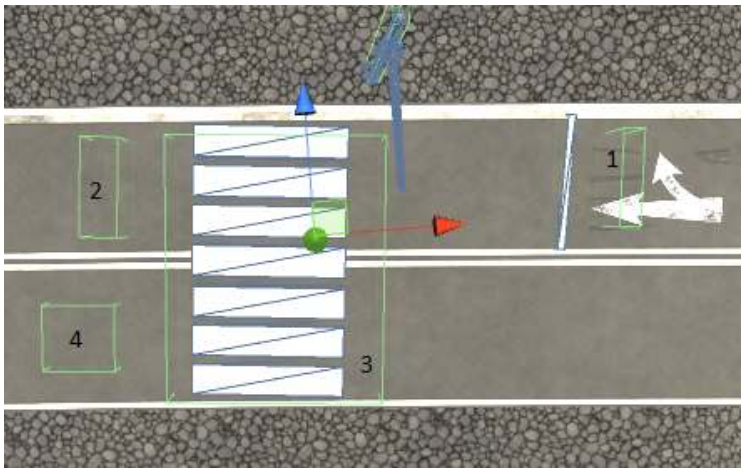

Figura 59: *Sensores que determinan el comportamiento del usuario en un semáforo*

- **Cedas el paso**: En la figura 60 se muestra el comportamiento de una regla de ceda el paso. Como podemos observar, el *Prefab* está compuesto de 6 "sensores". Los sensores (1) y (2) contienen código para determinar si el usuario se está acercando a la intersección. Los sensores (5) y (6) activan una variable si algún vehículo se encuentra dentro de ese sensor. Los sensores (3) y (4) notificarán la infracción de ceda el paso, tras comprobar, en el momento de abandonar

el sensor (2), si existía algún vehículo en los sensores (5) y (6), y dependiendo el destino de la intersección, comprobar si el vehículo interfiere en nuestro recorrido.

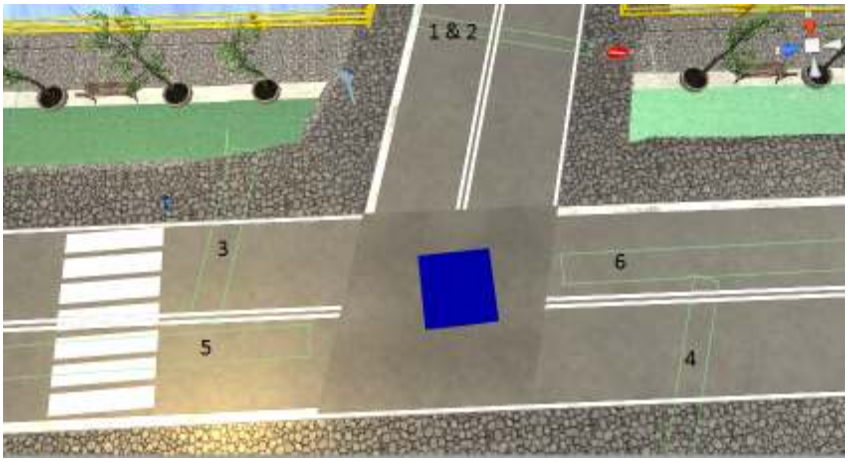

Figura 60: *Sensores que determinan el comportamiento del usuario en un ceda el paso*

- **Intermitentes**: En la figura 61, podemos ver el comportamiento de sensores como sigue. Disponemos hasta un total de 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, que 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).

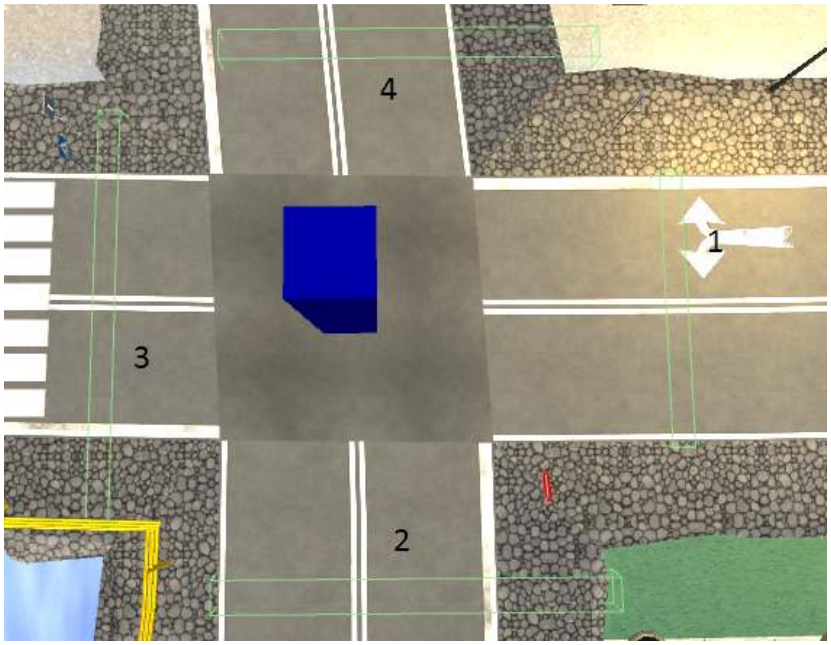

Figura 61: *Sensores que determinan el comportamiento de los intermitentes*

7.3.7.4 RealisticCarControllerv3

Realistic Car Controller es un Asset dedicado a la configuración del vehículo que usara el jugador, está formado por una serie de *scripts,* los cuales implementan desde el funcionamiento básico del vehículo (marchas, velocidad, fuerza del motor, luces,…) hasta la configuración de los controles usados para desplazarse.

Para más información sobre este Asset se puede consultar al Anexo correspondiente al final de la documentación.

De todos sus *scripts*, RealisticCarControllerV3.cs es el *script* principal del *Asset*. Contiene el comportamiento del motor del vehículo, así como las marchas, revoluciones o manejo. Modificaremos el comportamiento de este *script* para adaptar las siguientes características:

- **Comportamiento del cambio de marchas**: Nuestro script contendrá 3 tipos de cambio de marchas: automático, manual y por levas. Deberemos ajustar el *script* a cada tipo de cambio. Para cambio automático no necesitaremos hacer uso del embrague, por lo que las marchas deberán incrementarse o reducirse, de la misma forma que funcionaria un vehículo real automático Para cambio manual, el usuario deberá poder cambiar de marcha cuando pise el embrague e introduzca la marcha deseada, teniendo en cuenta que no se debe poder cambiar de quinta a marcha atrás o que, si estamos yendo marcha atrás, no podemos introducir una marcha hasta detenernos completamente. Para cambio de marchas por levas el usuario podrá incrementar o reducir las marchas con la ayuda de las levas del volante, los cuales de configurarán para ello. En la figura 62, mostramos el código de este *script*. Este fragmento tras comprobar que se seleccionó automático en las opciones de configuración incrementará o reducirá la marcha del vehículo dependiendo de las revoluciones de este. También cambiara la dirección del vehículo al meter marcha atrás o primera marcha.

```
// Marchas automaticas
if(drivePref == 0){
    if(currentGear < totalGears - 1 && !changingGear){
        if(speed >= (gearSpeed[currentGear] * .35f) && FrontLeftWheelCollider.rpm > 0){
            if(!semiAutomaticGear)
                StartCoroutine("ChangingGear", currentGear + 1);
            else if(semiAutomaticGear && direction != -1)
                StartCoroutine("ChangingGear", currentGear + 1);
        }
    }
    if(currentGear > 0){
        if(!changingGear){
            if(speed < (gearSpeed[currentGear - 1] * .25f) && direction != -1){
                StartCoroutine("ChangingGear", currentGear - 1);
            }

        }
    }
    // Cambio de Primera a Marcha atras
    if (Input.GetButtonDown("MAtras") && currentGear == 0){
        if (speed < 5){
            StartCoroutine("ChangingGear", -1);
        }
    }
    // Cambio de Marcha atras a primera
    if (Input.GetButtonDown("Primera") && direction == -1){
        if (speed < 5){
            Debug.Log("Primera");
            StartCoroutine("ChangingGear", 0);
        }
    }
}
```

Figura 62: *Código de configuración de marchas automáticas*

La figura 63, equivale al fragmento de código para los cambios de marcha en 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 63: *Código de configuración de marchas manuales*

- **Comportamiento del motor**: El *script* deberá estar configurado para que el comportamiento sea similar al de un motor de un vehículo real, es decir, deberá asignar la velocidad y aceleración del vehículo de forma que intervengan factores como la fuerza del motor, la velocidad actual, el sonido del motor, la marcha actual o si el usuario está realizando alguna acción como acelerar.
- **Comportamiento de las luces**: El *script* deberá desarrollar el comportamiento de las diferentes luces del vehículo. En el caso de luces de carretera, se modificarán la distancia, intensidad ángulo de visión e icono indicador de la HUD, dependiendo de si el jugador activa las luces de corto alcance o las luces de largo alcance. Las luces de freno situadas en la parte trasera deberán activarse cuando el usuario frene. Del mismo modo, al seleccionar la marcha atrás, las luces traseras deberán adecuarse para informar a los demás usuarios de la vía de la intención de realizar un desplazamiento hacia atrás. En la figura 64, vemos los diferentes estados de las luces, que se accionaran al pulsar el botón (como vemos en la primera línea de código).

```
if(Input.GetKeyDown(RCCSettings.highBeamHeadlightsKB)){
    // Luces de corto alcance
    if (lightCount == 0){
        shortlonglight.activeshortLight();
        lightCount = 1;
    }
    // Luces de largo alcance
    else if(lightCount == 1){
        shortlonglight.activelongLight();
        lightCount = 2;
    }
    // Sin luces
    else{
        shortlonglight.disableLight();
        lightCount = 0;
    }
}
```

Figura 64: *Código de configuración de las luces de corto y largo alcance*

- **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. A su vez, dentro de este código, las luces de intermitencia se activarán y desactivarán cada cierto tiempo, dando la sensación de intermitencia. En la figura 65 observamos el código que usaremos para activar o desactivar las luces de intermitencia. A este *script* hay que añadirle 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.

```
// intermitente Derecho
if(Input.GetKeyDown(RCCSettings.rightIndicatorKB)){
    if(indicatorsOn != IndicatorsOn.Right){
        indicatorsOn = IndicatorsOn.Right;
        indicatorR = true;
        indicatorlight.activeRight();
    }
    else {
        indicatorsOn = IndicatorsOn.Off;
        indicatorR = false;
        indicatorlight.disableLight();
    }
}

// Intermitente Izquierdo
if(Input.GetKeyDown(RCCSettings.leftIndicatorKB)){
    if(indicatorsOn != IndicatorsOn.Left){
        indicatorsOn = IndicatorsOn.Left;
        indicatorL = true;
        indicatorlight.activeLeft();
    }
    else {
        indicatorsOn = IndicatorsOn.Off;
        indicatorL = false;
        indicatorlight.disableLight();
    }
}
```

Figura 65: *Código de configuración de los intermitentes*

7.4 JUGABILIDAD

En esta sección detallaremos las diferentes características de la jugabilidad del usuario.

7.4.1 Objetivos

Nuestro proyecto tendrá diferentes objetivos en cuanto a jugabilidad:

- El usuario tendrá como primer objetivo un menú simple con las diferentes opciones a tomar.
- El usuario podrá iniciar sesión en el servidor.
- El usuario podrá modificar las opciones del vehículo.
- El usuario podrá modificar las opciones de la simulación.
- El usuario podrá elegir una simulación para realizarla.
- Al entrar en la simulación, el usuario deberá conducir el vehículo del jugador.
- El usuario podrá elegir la opción de ruta guiada, donde se le mostrará el itinerario a seguir.
- El usuario podrá detener la simulación en cualquier momento.
- Al final de la simulación, se le mostrarán al usuario los diferentes resultados obtenidos.
- El resultado de la simulación deberá ser enviado al servidor.

7.4.2 Acciones del usuario

El usuario podrá realizar las siguientes opciones:

- Iniciar la aplicación.
- Identificarse en el simulador.
- Configurar opciones del vehículo.
- Seleccionar un escenario de simulación.
- Conducción del vehículo por el escenario.
- Consulta de las estadísticas (infracciones y datos del vehículo) obtenidas.

7.4.3 Controles

El dispositivo de control que usará el usuario para poder realizar la simulación por el escenario será el dispositivo Logitech G27 (detallado en apartados anteriores). Los controles de este dispositivo usados en este escenario por defecto se muestran en la figura 66.

Figura 66: *Controles del usuario por defecto*

Estos controles los podemos definir en Unity en la sección: Settings > Inputs. Podemos observarles en la figura 67.

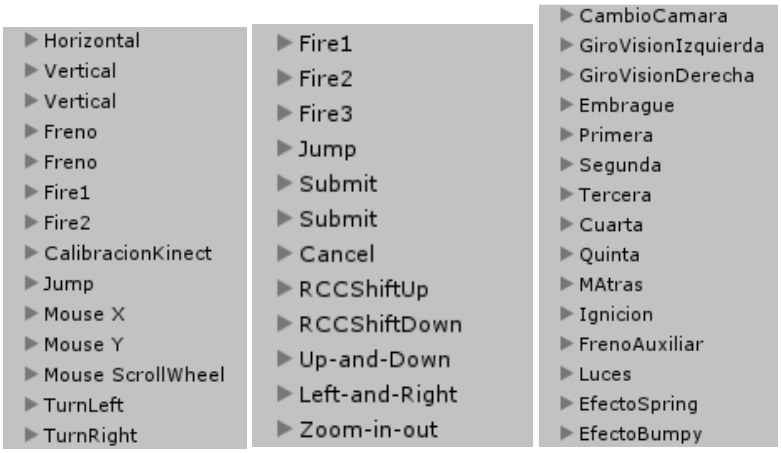

Figura 67: *Controles del usuario definidos en Unity*

También es posible configurar los controles 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 68.

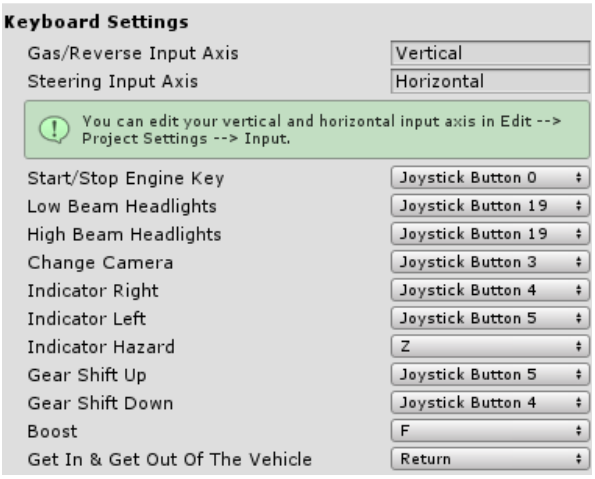

Figura 68: *Controles del usuario definidos en el RCCv3*

Aun así, una vez compilado el simulador, al iniciarlo, se pueden configurar los diferentes controles. De esta forma, aunque el usuario no disponga de un dispositivo de conducción como el que hemos especificado o, si, por el contrario, no está de acuerdo con la asignación, puesto que conduciría mejor con otra asignación, podrá modificarla a su gusto. Podemos ver la configuración de estos controles en la figura 69.

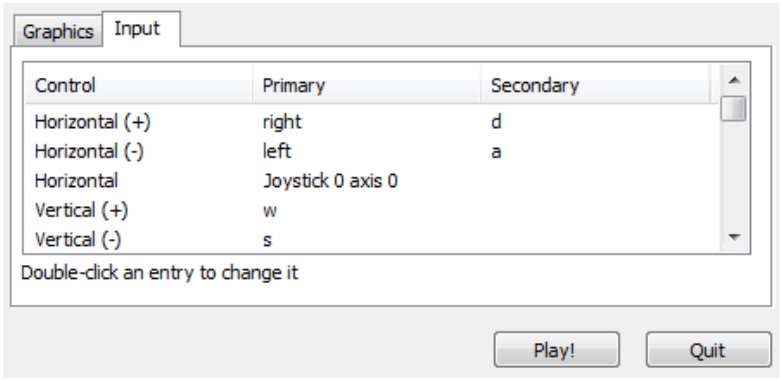

Figura 69: *Controles definidos al ejecutar la aplicación*

7.4.4 Log

Al terminar la simulación, se generará un fichero de datos, el cual tendrá los datos de cada simulación. Posteriormente, será enviado al servidor, por lo que deberá adaptarse a una estructura específica para que el servidor pueda entender el fichero.

En la figura 70, podemos observar cómo está compuesto el nombre del fichero. Este nombre estará formado por el identificador del escenario, la fecha y hora de la simulación, el usuario y las opciones de simulación elegidas.

El primer requisito de este fichero es que el nombre debe mantener la siguiente estructura:

```
// ID del escenario
string escenario = "S9";
// Fecha
string fechaAMD = System.DateTime.Now.Year.ToString() + "-" + System.DateTime.Now.Month.ToString() + "-" + System.DateTime.Now.Day.ToString();
// Hora
string instanteHMS = System.DateTime.Now.Hour.ToString() + "-" + System.DateTime.Now.Minute.ToString() + "-" + System.DateTime.Now.Second.ToString();
// Usuario que hace de jugador
string usuario;
if (PlayerPrefsX.GetBool("usuario_conectado"))
    usuario = PlayerPrefs.GetString("playerUsername");
else
    usuario = "Anonymous-user";
// Tipo de log
string tipo = "SIMULACION";
// Opciones de simulacion
string opciones = "";
if (PlayerPrefs.GetInt("DRIVE") == 0)
    opciones += "A";
else if (PlayerPrefs.GetInt("DRIVE") == 1)
    opciones += "M";
else if (PlayerPrefs.GetInt("combio_marchas") == 2)
    opciones += "L";
else
    opciones += "X";
opciones += "-";
if (PlayerPrefsX.GetBool("modo_gps"))
    opciones += "G";
else
    opciones += "L";
opciones += "-";
if (PlayerPrefsX.GetBool("infracciones"))
    opciones += "I";
else
    opciones += "N";
// Nombre de fichero resultante
string final = escenario + "_" + fechaAMD + "_" + instanteHMS + "_" + usuario + "_" + tipo + "_" + opciones;

return final;
```

Figura 70: *Estructura del nombre del fichero de simulación (Log)*

Dentro del fichero se pueden apreciar tres partes definidas:

- En la figura 71 vemos **datos de la simulación**. Estos datos serán el identificador del escenario, el consumo medio de la simulación, el consumo total y el tiempo que se ha tardado en completar la simulación.

```
datos = "{" + System.Environment.NewLine;
sumarelemento("id_simulacion", (9).ToString(), true);
sumarelemento("consumo_medio", script_consumo.consumoMedio.ToString(), true);
sumarelemento("consumo_total", script_consumo.consumoTotal.ToString(), true);
sumarelemento("tiempo_total", Time.time.ToString(), true);
```

Figura 71: *Contenido del Log en relación con los datos de la simulación*

- En la figura 72, observamos los datos de **posición del jugador**. Dichas posiciones se calcularán con una frecuencia configurable (típicamente de pocos segundos), y almacenará las coordenadas del usuario, así como el instante de tiempo en el que se realizaron las muestras. Estos datos podrán ser útiles para realizar un seguimiento del usuario a lo largo del recorrido. Almacenaremos el número de muestras guardadas y llamaremos al siguiente método recorriendo la estructura de datos recogidos.

```
/*
 *  Metodo sumarmuestraBBDD
 *      Añade una muestra a la cadena de trexto de la simulacion
 */
void sumarmuestraBBDD(int i)
{
    datos += "{" + System.Environment.NewLine;
    sumarelemento("instante", script_infrac.muestras[i].instante.ToString(), true);
    sumarelemento("posicion_x", script_infrac.muestras[i].posicion_x.ToString(), true);
    sumarelemento("posicion_y", script_infrac.muestras[i].posicion_y.ToString(), true);
    sumarelemento("posicion_z", script_infrac.muestras[i].posicion_z.ToString(), true);
    sumarelemento("velocidad", script_infrac.muestras[i].velocidad.ToString(), true);
    sumarelemento("rpm", script_infrac.muestras[i].rpm.ToString(), true);
    sumarelemento("marcha", script_infrac.muestras[i].marcha.ToString(), true);
    sumarelemento("consumo_instantaneo", script_infrac.muestras[i].consumo_instantaneo.ToString(), true);
    sumarelemento("consumo_total", script_infrac.muestras[i].consumo_total.ToString(), false);
    if (i == script_infrac.im - 1)
    {
        datos += "}" + System.Environment.NewLine;
    }
    else {
        datos += "}," + System.Environment.NewLine;
    }
}
```

Figura 72: *Contenido del Log en relación con los datos del jugador*

- En la figura 73, vemos las **infracciones** cometidas por el usuario durante la simulación. Estas infracciones tendrán una serie de parámetros obligatorios, como el identificador del tipo de infracción (es decir, el identificador que determinará si se ha saltado un semáforo o si ha golpeado a otro vehículo).

  Al igual que en el apartado anterior, también se guardará la posición del usuario en el instante de la infracción, de forma que se pueda representar la ubicación de las infracciones cometidas, el instante de la infracción y también un mensaje, el cual nos dará información como el tipo de infracción que se ha cometido.

```
/*
 *  Metodo sumarinfraccionBBDD
 *      Añade una infraccion a la cadena de trexto de la simulacion
 */
void sumarinfraccionBBDD(int j)
{
    datos += "{" + System.Environment.NewLine;
    sumarelemento("instante", script_infrac.infracciones[j].instante.ToString(), true);
    sumarelemento("id_infraccion", script_infrac.infracciones[j].id_infraccion.ToString(), true);
    sumarelemento("posicion_x", script_infrac.infracciones[j].posicion_x.ToString(), true);
    sumarelemento("posicion_y", script_infrac.infracciones[j].posicion_y.ToString(), true);
    sumarelemento("posicion_z", script_infrac.infracciones[j].posicion_z.ToString(), true);
    sumarelemento("observaciones", script_infrac.infracciones[j].observaciones.ToString(), false);
    if (j == script_infrac.ik - 1)
    {
        datos += "}" + System.Environment.NewLine;
    }
    else
    {
        datos += "}," + System.Environment.NewLine;
    }
}
```

Figura 73: *Contenido del Log en relación con los datos de las infracciones*

Una vez finalizada la simulación, es decir, tras alcanzar el punto de destino, se generará el fichero de log de la simulación realizada y, posteriormente, se enviará al servidor, el cual reconocerá el usuario, el número de la simulación y demás valores del nombre y contenido del fichero.

```
/*
 *  Metodo GrabarLog
 *      Envía el fichero Log generado a la base de datos
 */
void GrabarLog (string datos)
{
    // Varaible para controlar la ruta de los Logs
    string directoriologs = Application.dataPath + "/Logs";
    if (!Directory.Exists(directoriologs))
    {
        Directory.CreateDirectory(directoriologs);
    }

    // Varaible para controlar el nombre del fichero de simulacion
    string nombreFichero = obtenerNombreLogSIMULACION();

    // Montamos la ruta con el fichero escribimos los datos de texto, limpiamos y cerramos
    StreamWriter sw = new StreamWriter(Application.dataPath + "/Logs" + "/"+nombreFichero+ ".log");
    sw.Write(datos, true);
    sw.Flush();
    sw.Close();

    // Mostramos en Unity los datos generados
    // Debug.Log(datos);

    Debug.Log("Partida guardada en Log local");
    //script_final.LogMuestrasOFFGuardado = true;
}
```

Figura 74: *Código utilizado para el envío del fichero de simulación al servidor*

8 PRESUPUESTO

8.1 PRESUPUESTO INICIAL

8.1.1 Hardware

- **Ordenador personal**: Se va a requerir de un ordenador para la realización del proyecto. Contando que un ordenador portátil tiene una vida media de 4 años, y que se va a hacer uso de él durante 28 semanas (suponemos 7 meses de trabajo), su uso se calcula:

  100% ---------- 192 semanas

  X % ---------- 28 semanas x = (28*100) /192 = 14,6% de uso.

- **Volante *Logitech G27*:** Se requerirá para la realización de las pruebas, así como para comprobar que según vamos desarrollando el proyecto funciona correctamente (*testing*) un dispositivo para la conducción. Su precio es de 125€ aproximadamente. Al igual que hemos calculado antes, suponemos que este dispositivo tiene una vida de 4 años
- **Dispositivo *Kinect***: Se requerirá este dispositivo para monitorizar al usuario y de esta forma determinar dónde está mirando a la hora de conducir. El precio de este dispositivo es de 100€. Al igual que hemos calculado antes, suponemos que este dispositivo tiene una vida de 4 años

HARDWARE	COSTE (€)
ORDENADOR PERSONAL	500
VOLANTE G27	125
KINECT	100
TOTAL	725 €

Tabla 8: *Presupuesto hardware*

8.1.2 Software

- ***Unity***: Necesitaremos una herramienta para el desarrollo del proyecto. En nuestro caso y por las razones explicadas en el apartado de herramientas de desarrollo, usaremos *Unity*. Este *software* es gratuito si no se tiene un interés comercial. Puesto que nuestro proyecto será desarrollado inicialmente para tareas de investigación, no generará beneficios, por lo que por el momento podremos usar la licencia gratuita.
- ***Asset***: Dentro de *Unity*, haremos uso de diversos *Asset* (los cuales detallaremos en el apartado de Anexos). Dentro de este apartado hemos hecho uso de dos grandes *Asset*, utilizados para el desarrollo de nuestro vehículo (*Realistic Car Controller V3*) y para la implementación de la inteligencia artificial (*Intelligent Traffic System*). El precio de estos dos Assets es de 50€ y 75€ respectivamente. Ambos son de un único pago.
- **Modelos 3D:** Para la generación del entorno, necesitaremos modelos en 3D proporcionados por herramientas como *Blender*. Puesto que no somos diseñadores gráficos, ni nuestra habilidad para el desarrollo de estos modelos es buena, hemos descargado modelos de las páginas

gratuitas dedicadas a ellos. Hemos obtenido de estas páginas todos los modelos utilizados en el proyecto menos uno. Este modelo que no hemos descargado gratuitamente es el modelo del autobús urbano utilizado. Este modelo le hemos comprado también desde el *Asset Store*, de *Lowpoly_Master*, con un precio de 10€. Cabe destacar el buen trato y atención de esta empresa a la hora de la implementación de su modelo.

SOFTWARE	COSTE (€)
UNITY	0
ASSET – RCCV3	50
ASSET – ITS	75
MODELOS 3D	10
TOTAL	135 €

Tabla 9: *Presupuesto software*

8.2 PRESUPUESTO DE DESARROLLO

Los costes referidos al desarrollo del proyecto por parte de un ingeniero son los siguientes, contando que el proyecto desarrollado se ha realizado en un tiempo estipulado de 450h.

CATEGORÍA	TIEMPO	COSTE (€)
ANÁLISIS	60h	10€/h
DISEÑO	90h	10€/h
PROGRAMACIÓN	250h	12€/h
DOCUMENTACIÓN Y PRUEBAS	50h	8€/h
TOTAL	450h	4900€

Tabla 10: *Presupuesto desarrollo*

8.3 PRESUPUESTO TOTAL

	COSTE (€)
HARDWARE	725 €
SOFTWARE	135 €
DESARROLLO	4900 €
TOTAL	5760 €

Tabla 11: *Presupuesto total*

9 CONCLUSIÓN Y LÍNEAS FUTURAS

En este proyecto hemos realizado un escenario básico de conducción, en concreto, una ruta dentro la ciudad de Valladolid, para un simulador. Gracias a esta oportunidad he podido aprender otro lenguaje de programación como es C#, complementario a los vistos en el grado. Además, he adquirido conocimientos básicos de la herramienta Unity 3D.

Cabe destacar que el escenario ha sido realizado por una sola persona, por lo que, contando con un equipo de trabajo de 4 personas, y una vez conocida la herramienta a utilizar, este escenario podría ser mejorado con mayor facilidad. Aun así, Unity permite generar aplicaciones de forma sencilla, con conocimientos básicos de la herramienta.

Uno de los mayores desafíos de este proyecto ha sido recrear la inteligencia artificial de los demás vehículos y usuarios de la vía. El mayor problema encontrado, no por su complejidad sino por su dedicación y formación, ha sido la recreación de la ciudad, ya que se necesita mucho tiempo, dedicación y, lo más importante, se necesitan modelos en 3D para poder recrear el escenario. Un diseñador gráfico especializado en la creación de modelos 3D sería de gran ayuda ya que no se estaría limitado a usar los modelos encontrados por Internet de forma gratuita. En este sentido, también ayudaría a la modificación de algunas formas utilizadas, adaptándolas mejor y detallándolas.

Como líneas futuras de este proyecto, cabe destacar las comentadas a continuación.

- Este escenario recrea un tramo de la ciudad de Valladolid, por lo que se podría ampliar para abarcar mayor distancia de la ciudad e incluso conectar este tramo urbano con algún tramo interurbano, por ejemplo, para llegar un pueblo situado a 15 km. La duración de este recorrido ampliado sería de unos 20min.

- Otra línea similar consistiría en desarrollar diferentes escenarios de la ciudad incluyendo, por ejemplo, tramos por travesía o más tramos urbanos. Estos escenarios se podrían implementar para poder acceder a ellos desde el menú de escenas.

- Contando con un mayor número de modelos 3D, podría elegirse el vehículo a conducir. De esta forma, en lugar del vehículo Lexus, podría conducirse otro tipo de vehículo como un autobús u otro modelo, como la Peugeot Bipper utilizada.

- Para algunos modelos de vehículo de jugador, se podría elegir implementar la opción de seleccionar el tipo de motor: de gasolina, gasoil o eléctrico. De esta forma, podemos concretar más el modelo del vehículo y su tipo de consumo.

- Se podrían añadir mayor número de opciones de configuración como pueden ser la selección del color del vehículo entre una gama de colores mediante una paleta de colores (RGB 256 bits).
- También se podrían implementar diferentes condiciones meteorológicas adversas como lluvia, conducción sobre asfalto mojado o incluso nieve.

- Por último, se podría incluir la implementación de un menú general que adapte todos los diferentes escenarios a desarrollar, así como un informe detallado de las diferentes preferencias que el escenario debería tener para poder configurarlo desde este menú general.

10 ANEXOS

10.1 INTELLIGENT TRAFFIC SYSTEM (ITS)

Intelligent Traffic System (iTS) es una aplicación desarrollada para *Unity 3D* que simula tráfico. Este sistema de tráfico es usado para simular determinados eventos.

Para acceder al menú del iTS seleccionamos la opción **GameObject -> iTS**. Dentro de este menú, encontramos las opciones:

- **Traffic Lights**: Esta opción tiene a su vez las opciones:
    - **Add traffic light script to selected**: Permite añadir un script del tipo *traffic light* al objeto seleccionado
    - **Group**: Permite hacer un grupo de *traffic lights* para simplificar la administración de ese grupo.
- **Simple Vehicle**: Tiene un submenú con la opción *Add Empty*. Esta opción añade un vehículo vacío a la escena y nos muestra una pantalla de configuración para añadir las propiedades del vehículo (*body*, ruedas, etc.)
- **iTS Manager**: Tiene tres submenús:
    - **Add iTS Manager to the scene**: Esta opción añade una instancia del *iTS Manager* a un nuevo *GameObject* si no existe aún. En caso de que exista, nos saldrá un mensaje informándonos.
    - **Add Traffic spawner to the scene**: Añade un nuevo *Traffic Spawner* a la escena. Este *Spawner* es necesario para poder crear un sistema de tráfico.
    - **Add iTS Traffic volume to the scene**: Si un *GameObject* está seleccionado, el *script* de volumen será añadido dentro de un nuevo objeto contenido dentro del objeto seleccionado.

10.1.1 Configuración de un Nuevo Vehículo (New Vehicle)

Para añadir un nuevo vehículo a nuestra librería de vehículos usamos la opción *Add Empty* del menú.

Cuando seleccionamos esta opción, aparecerá una ventana requiriendo un modelo 3D del vehículo, el cual tendremos que arrastrar hasta ahí.

El siguiente paso es ajustar la posición y rotación del modelo en caso de que sea necesario (es mejor ajustar estos parámetros en una herramienta de modelado 3D como *Blender* para corregirlos antes de importarlos en Unity). Es decir, es mejor retocar la posición de la figura para que una vez introducida esté ya centrada.

El siguiente paso es asignar las ruedas. En este paso es necesario seleccionar todos los *GameObject* de los cuales se compone la rueda, como pueden ser las llantas o los frenos.

A continuación, la ventana mostrará el radio de las ruedas traseras y delanteras. El asistente ya habrá calculado un radio óptimo acorde a los modelos de ruedas introducidos anteriormente.

Posteriormente añadimos los *Colliders* al modelo. Para ello podemos crear un *Collider* de tipo *Mesh*, que se ajuste a la forma (consumirá más), de tipo *Box Collider* (al ser un cubo tiene que calcular menos polígonos) o de cualquiera de los otros *Colliders* disponibles.

El siguiente paso es añadir las luces de freno. Para ello seleccionamos todos los *GameObjects* que representan estas luces y luego seleccionamos el material correspondiente de la lista en caso de tener más de un material.

El último paso es guardar nuestro modelo. Este paso añadirá un *Prefab* del modelo dentro de la carpeta */Resources/Car* del proyecto.

10.1.2 Configuración de un Sistema de Tráfico (Traffic System)

Para la configuración del tráfico existen diversos ajustes y parámetros que pueden ser modificados si se necesitan. Estas opciones estarán dentro del objeto denominado *iTSManage* y las podemos observar en la figura 75.

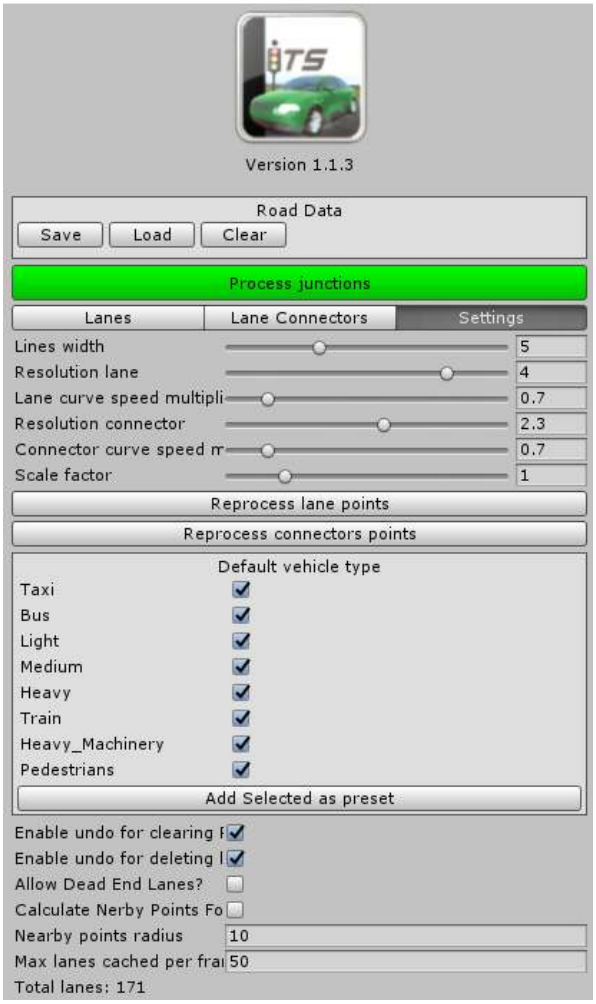

Figura 75: *Configuración general del iTS*

La opción de procesar las líneas (*Process Junctions)*, comprueba todas las líneas establecidas en el escenario para poder usar el sistema de tráfico. Tiene las siguientes opciones:

- ***Lines width***: Anchura visual de las líneas mientras estemos en modo *edit*.
- ***Resolution lane***: Distancia entre puntos de las líneas. cuanto mayor sea este valor, menos puntos serán usados.
- ***Lane curve speed multiplier***: Multiplicador de la velocidad máxima en las curvas que el sistema calcula cuando las líneas y conectores son procesados usando el botón "*Process junctions*".
- ***Resolution connector***: Distancia entre cada punto de los conectores.
- ***Connector curve speed multiplier***: Multiplicador de velocidad máxima en las curvas que el sistema calcula cuando las líneas y conectores son procesados usando el botón "*Process junctions*".
- ***Reprocess lane points***: Reprocesa las líneas de tráfico. Debe ser presionado cuando una línea es modificada para aplicar los cambios.
- ***Reprocess connector points***: Reprocesa los conectores de tráfico. Debe ser presionado cuando una línea es modificada para aplicar los cambios.
- ***Default vehicle type***: Tipos de vehículo por defecto al crear las líneas.
- ***Add selected as preset***: Añade los tipos de vehículos actuales como una nueva configuración cuando una línea o conector está seleccionado.
- ***Max Lanes cached per frame***: Cantidad de líneas que el editor podría procesar en cada *frame.* Esta variable es usada para detectar las líneas que son actualmente visibles en la escena.

10.1.2.1 Líneas (Lanes)

Usaremos líneas para determinar el recorrido que realizarán los vehículos.

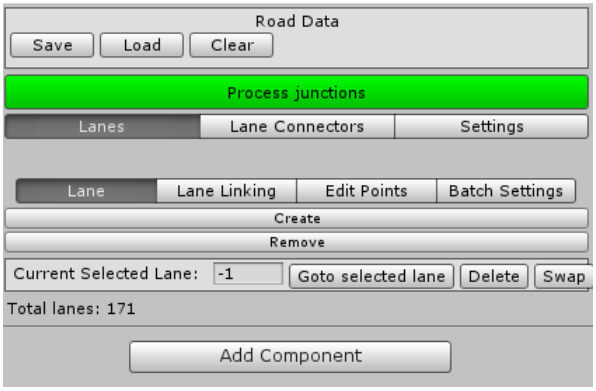

Figura 76: *Líneas de tráfico de iTS*

La interfaz se muestra en la figura 76. Para añadir líneas de tráfico, usaremos la opción de crear (*Create*) líneas dentro del *script*. Una vez elegida esta opción, haremos clic para insertar puntos de la línea. Tendremos que configurar los diferentes vehículos que circulan por el tramo, así como la velocidad del mismo.

Podemos eliminar líneas. Para ello seleccionamos la opción borrar (*Remove*) y seleccionamos la línea deseada.

Podremos también modificar los diferentes puntos de una línea (*Edit Points*), ya sea para modificar la línea si se quiere realizar otro recorrido, o simplemente para retocar la ruta de los vehículos.

Podemos ajustar las diferentes opciones de configuración de las líneas:

- ***Width***: Ancho de la línea.
- ***Max Speed***: Velocidad máxima de la línea.
- ***Max Density***: El porcentaje de ocupación para la colocación de vehículos en ella. Si está a 1 el 100% de esta línea debería estar ocupada cuando se colocan los vehículos iniciales. Si está a menos de 0.5, es decir, el 50%, se colocarán vehículos para el 50% de la ocupación.
- ***Vehicle Type***: Tipos de vehículos que pueden circular por la línea.
- ***Presets***: Los vehículos disponibles por defecto de la línea.

10.1.2.2 Conectores (Connectors)

Usamos los conectores para unir las diferentes líneas, de esta forma crearemos las intersecciones.

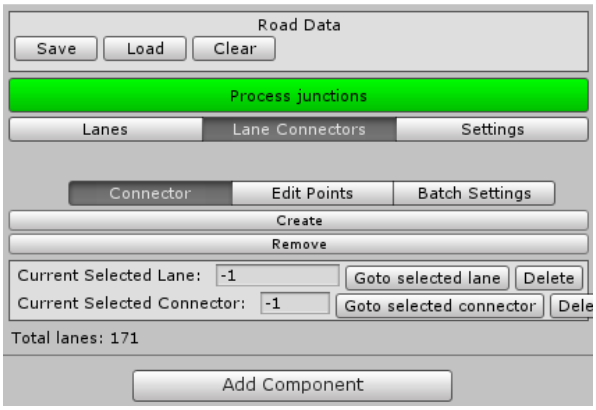

Figura 77: *Conectores de tráfico de iTS*

La interfaz se muestras en la figura 77. Para añadir conectores, usaremos la opción de crear conectores dentro del *script*. Una vez elegida esta opción, haremos clic en un punto exterior de una línea (color verde) y arrastraremos hasta otro punto de otra línea a unir (color azul).

Podemos eliminar conectores. Para ello, seleccionamos el conector deseado y, posteriormente, lo eliminamos.

Podremos también modificar los diferentes puntos de un conector, ya sea para modificar el conector para realizar otro recorrido, o simplemente para retocar la ruta de los vehículos.

Podemos ajustar las diferentes opciones de configuración de los conectores, que son:

- ***Direction***: Dirección que el vehículo tomará en la intersección. Al crearlo detecta automáticamente si será un giro a la izquierda, derecha o continuará recto. Es necesario para poder utilizar las luces de intermitencia de los vehículos.
- ***Forced stop***: Habilitada, los vehículos realizarán una parada antes de cruzar el conector. Sirve para simular señales de Stop.
- ***Pass priority***: Prioridad de paso entre las diferentes intersecciones que finalicen en la misma línea.

- ***Vehicle Type***: Tipo de vehículos que pueden circular por este conector.
- ***Presets***: Los vehículos por defecto que se asignan al conector.

10.1.3 Configuración de la Física de un Vehículo (*Traffic Car Physics*)

Cada vehículo que generemos tendrá una serie de características físicas, podemos observarlas en la figura 78. Estas características son las siguientes:

- ***Front wheels***: Este *Array* contiene todos los objetos padre de las ruedas delanteras.
- ***Back wheels***: Este *Array* contiene todos los objetos padre de las ruedas traseras.
- ***Additional brake wheels***: Este *Array* contiene todos los objetos padre de las ruedas adicionales como pueden ser las del tráiler de los camiones.
- ***Gear Ratio***: Ajuste de la ratio de marchas de un vehículo.
- ***Current Gear***: Marcha actual del vehículo.
- ***Engine Torque***: Fuerza total del motor del vehículo.
- ***Max Engine RPM***: Revoluciones máximas por minuto.
- ***Min Engine RPM***: Revoluciones mínimas por minuto.
- ***Max acceleration***: Factor de aceleración del vehículo. A mayor valor, más rápido acelerará.
- ***Brake torque***: Fuerza máxima de los frenos
- ***Brake Lights***: Este *Array* contiene las luces de freno del vehículo
- ***Brake intensity rate***: Controla cómo de rápido cambian las luces de freno al ser o no ser usadas.
- ***Enable Disable Brake Light***: Esta opción hace que se desactive el *renderizado* de las luces al encenderlas y apagarlas.
- ***Brake Lights Shader***: Sombra asignada si la opción anterior no es usada.
- ***Property name***: Nombre de la propiedad que debería ser usada para encender o apagar las luces.
- **Super Simple Physics**: Esta opción habilita la física súper simple, que no usa *Colliders* de ruedas.
- **Turn speed**: La velocidad máxima de giro para el modo súper simple.
- **Car Grip**: El gripado en el modo súper simple.
- **Max Speed**: La velocidad máxima del modo súper simple.
- **Max Speed to turn**: La velocidad máxima para girar el coche en el modo súper simple.
- **UpsideDown**: Esta variable informa sobre si el coche está volcado.
- **Crashed Smokes**: Este *Array* mantiene una o más partículas para similar humo del motor cuando el coche se daña.
- **Turn right light**: Este objeto contiene todos los intermitentes derechos del vehículo.
- **Turn left light**: Este objeto contiene todos los intermitentes izquierdos del vehículo.
- **Center of Mass**: Centro de masas del vehículo.

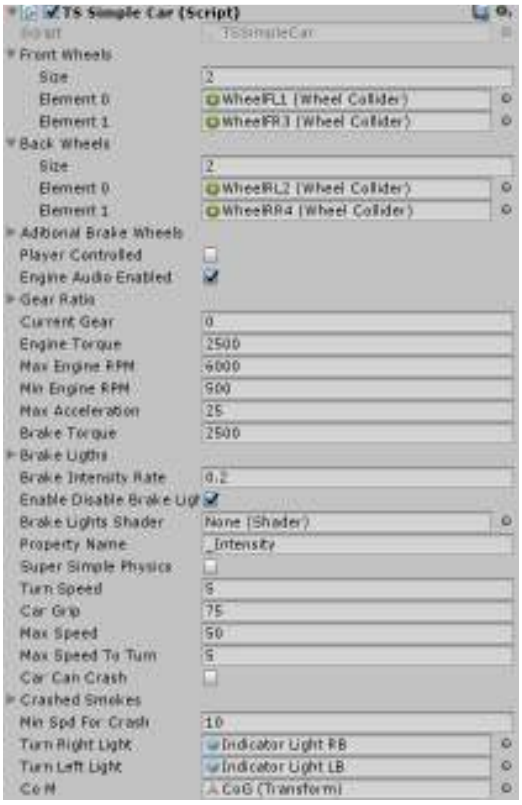

Figura 78: *Características de un vehículo en el iTS*

También el *script* de ajustes contiene las siguientes propiedades mostrado en la figura 79.

- ***Front Wheels***: En este *Array* se asignarán los *Transform* de las ruedas delanteras, usado para calcular el centro entre ambas.
- ***LOOKAHEAD_CONST***: Distancia por defecto entre el *Waypoint* actual y el vehículo.
- ***My Vehicle Type***: Tipo de vehículo.
- ***Player Sensor***: *Box Collider* usado para detector a otros vehículos.
- ***Min Player Sensor Length***: Distancia mínima del sensor de jugador cuando el vehículo está detenido o va lento.
- ***Length Margin***: Distancia entre los vehículos de IA.
- ***Length MarginMin***: Distancia mínima entre los vehículos de IA.
- ***Length MarginMax***: Distancia máxima entre los vehículos de IA.
- ***My Lane Offset Min***: *Offset* mínimo que el vehículo puede tener. Asigna variaciones a la posición de los vehículos de la línea de forma que no parezca que siguen la misma línea.
- ***My lane offset Max***: *Offset* máximo que el vehículo puede tener. Asigna variaciones a la posición de los vehículos de la línea de forma que no parezca que siguen la misma línea.
- ***Player Tag***: Etiqueta del vehículo del jugador.
- ***Player Detected Sound Audio Source***: Fuente de audio reproducido cuando se detecta el vehículo del jugador.

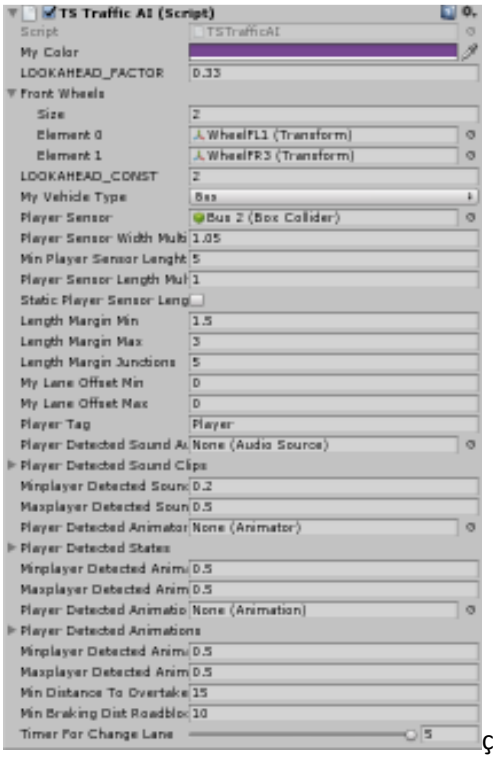

Figura 79: *Configuración de un vehículo en el iTS*

10.1.4 Configuración de un Spawner

Un Spawner es un generador de vehículos. Estos vehículos se generarán dentro de la zona del *Spawner*, dentro de las líneas trazadas para crear el sistema de tráfico. En la figura 80, observamos las opciones de configuración para la generación de vehículos.

Figura 80: *Opciones de un Spawner*

- ***Cars (Array):*** Este es el *Array* de vehículos que queremos que se generen en nuestra escena. Estos vehículos necesitan estar configurados con el *iTS*, como vimos en apartados anteriores.
- ***Amount*****:** La cantidad de vehículos que queremos en la escena.
- ***Max Distance***: Es la distancia máxima de generación de vehículos, equivale a la línea verde exterior del *Spawner*.
- ***Offset*****:** Es la distancia mínima de la generación de vehículos, equivale a la línea verde interior. En el espacio entre el *Offset* y *Max Distance,* los vehículos se generan.
- ***Closer Range*****:** Es el rango donde los vehículos pasan a la física súper simple. El objetivo es ahorrar en el funcionamiento con los coches más lejanos usando una física que consume menos CPU.
- ***Refresh Time*****:** Es el tiempo de refresco del generador, para buscar puntos de procesamiento dentro del área de generación.
- ***Manager*****:** La referencia al *GameObject iTS Manager*, la cual viene asignada por defecto automáticamente.
- ***Unused Cars Position*****:** Es el *Vector3* de posición donde los vehículos son relocalizados cuando no están siendo usados en la escena, para su posterior recolocación. Por defecto es el 50000, 50000, 50000.
- ***Respawn if upside down*****:** Esta opción habilita la recolocación de un vehículo si se genera boca arriba.
- ***Respawn upside-down time*****:** Es el tiempo que necesita un vehículo que no está colocado boca arriba para volverse a recolocar.
- ***Respawn Altitude*****:** La altitud de generación de los vehículos.
- ***Disable Multi-threading*****:** Esta opción habilita el proceso de recogida de los puntos de generación.
- ***Global Point Offset*****:** Es un *Offset* global útil si tu escena es muy grande y comienzas a tener puntos en vuelo provocando problemas con las físicas.
- ***Cars created per frame*****:** La cantidad de vehículos que el *Spawner* comprueba si están fuera del rango de generación para deshabilitarlos en cada ciclo de actualización.
- ***Initial not spawning radius*****:** Este es el radio del *Spawner* que debería comprobar sus coordenadas, para evitar que los vehículos inicialmente se generen dentro del área. Útil para evitar que los vehículos se generen encima del jugador.
- ***Transform spawning check*****:** Es una lista de coordenadas que serán comprobadas por el *Spawner* cuando va a generar un vehículo. Esto evitaría generar un vehículo dentro del radio de estas coordenadas, evitando generar muchos vehículos juntos y repartiéndolos de mejor forma por la escena.

10.2 REALISTICCARCONTROLLERV3

Realistic Car Controller V3 (*RCC*v3) es un *Asset* para el desarrollo y optimización de la conducción en la herramienta *Unity*. Este *Asset* estará formado básicamente por una serie de *scripts* encargados de asignar los valores al vehículo, así como de gestionar las diferentes variables usadas en él (*Colliders*, *Transforms*, *Lights*…).

Cada vehículo tiene su propio *RCCV3 Script*, por lo tanto, cada vehículo es responsable de las propiedades asignadas en su propio *Script*. Podemos observar la interfaz en la figura 81.

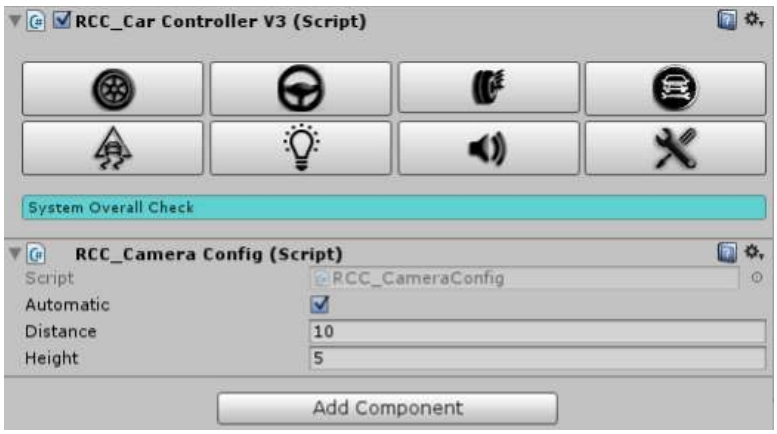

Figura 81. *Configuración de Realistic Car Controller V3*

En el Script principal del RCC, llamado *RCC_Car_Controller_V3.cs* tendremos 8 categorías principales detalladas a continuación:

- Ruedas (*Wheels*).
- Volante (*Steering*).
- Suspensión (*Suspensions*).
- Configuración mecánica (*Mechanic Configuration).*
- Estabilidad (*Stability*).
- Luces (*Lights*).
- Sonidos (*Sounds*).
- Daños (*Damages*).

Todos los vehículos comparten una configuración global almacenada en *RRC Settings*. Podemos acceder a esta configuración mediante: **Tools -> BoneCracker Games -> Realistic Car Controller -> RCC Settings**.

También podremos modificar sonidos, materiales de terreno, etc., en la opción: **Tools -> BoneCracker Games -> Realistic Car Controller -> Configure Ground Materials**.

10.2.1 Player Car

En nuestro caso, el coche diseñado para el jugador será un Lexus. Para ello obtenemos las texturas materiales y modelo descargándolo en una Web gratuita.

Una vez tenemos nuestro modelo de coche dentro de *Unity* es hora de personalizarlo acorde a nuestras expectativas (velocidad, tamaño, luces), gustos (color) y aficiones (AWD, FWD, etc.).

Una vez configurada y adaptada la apariencia de nuestro vehículo, así como ordenados los diferentes elementos que forman el coche (debemos seguir siempre la misma estructura en cuanto a los elementos almacenados en la jerarquía de nuestro proyecto, de tal forma, que sean para nosotros lo más cómodos, rápidos y accesibles), procedemos a configurar las características del vehículo en el *RCC*. Para ello, observamos los diferentes valores que pueden tomar cada una de las 8 categorías principales nombradas en el apartado anterior. Podemos observar el vehículo con componentes en la figura 82.

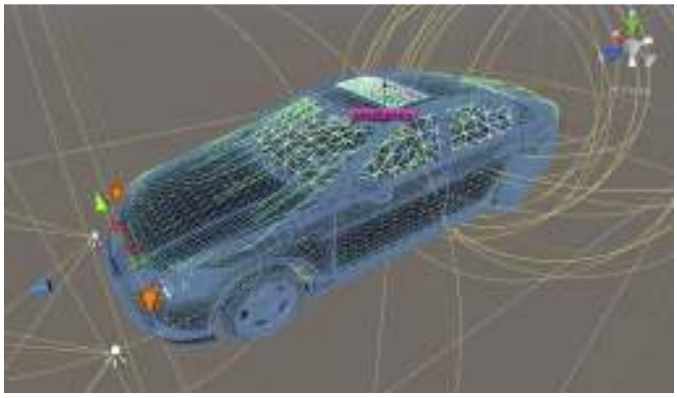

Figura 82. *Vehículo donde se pueden apreciar los diferentes componentes que los forman.*

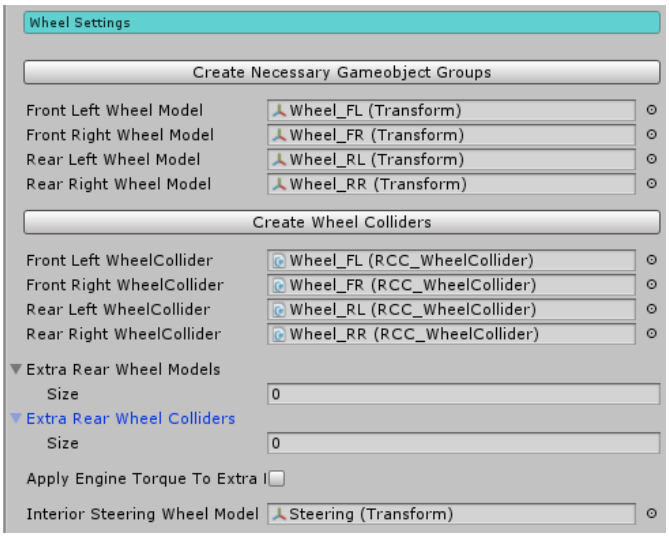

Figura 83. *Configuración de las ruedas*

Ruedas (*Wheels*): En la figura 83, observamos esta configuración. En esta categoría introducimos todo lo relacionado con las ruedas del vehículo, su componente ***Transform*** (componente que almacena la posición, rotación y escala) y su *Collider* (componente usada para producir colisiones entre los objetos, es decir, dotarlas de un físico real). También introduciremos modelos y *Colliders* de tamaño extra. Por último, introducimos el volante (*Steering*). Todos estos elementos servirán para que, al girar el coche, interactúen, dotando de realidad a la conducción.

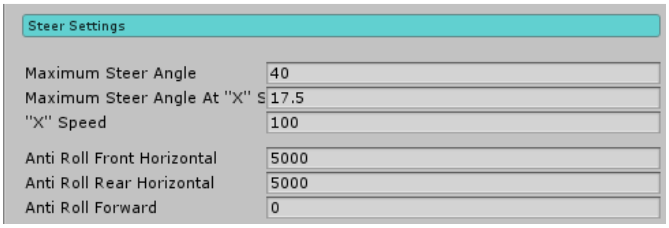

Figura 84. *Configuración del volante*

Volante (*Steering*): En esta categoría introducimos las características que forman parte del volante, como el ángulo de giro el ángulo de giro dependiente de la velocidad. Podemos observar las características en la figura 84.

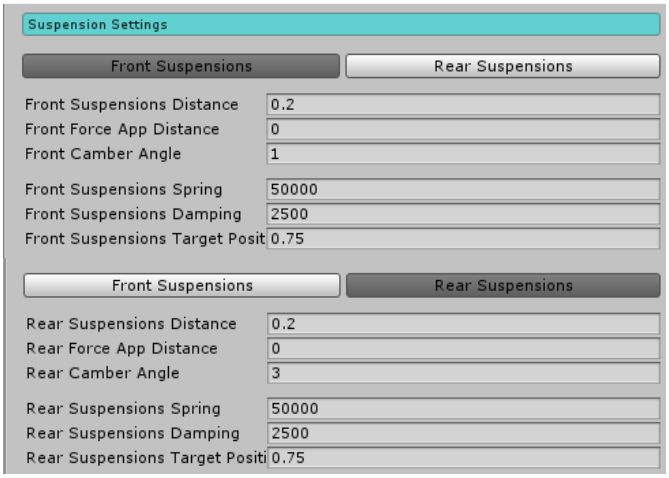

Figura 85. *Configuración de la suspensión*

Suspensión (*Suspensions*): En esta categoría configuraremos la suspensión frontal y trasera del vehículo. Ambas suspensiones poseen las mismas propiedades a configurar. Esta categoría se muestra en la figura 85.

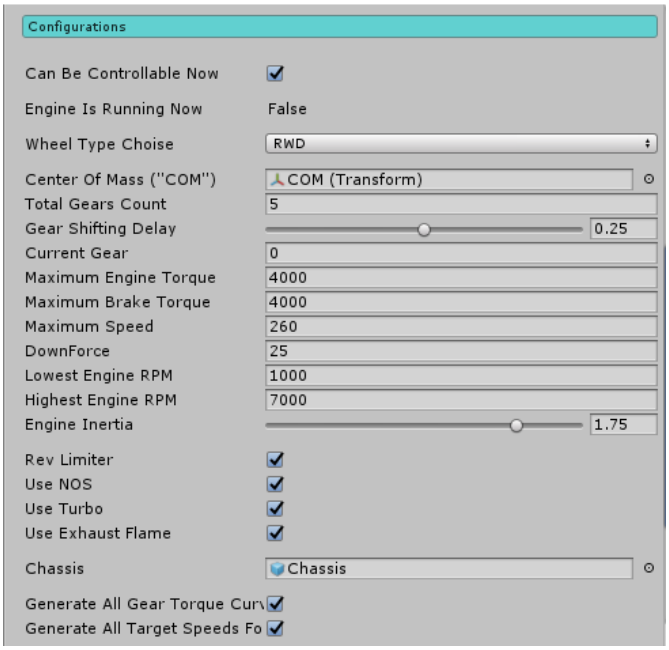

Figura 86. *Configuración de la mecánica*

Configuración mecánica (*Mechanic Configuration*): Esta categoría contiene variables para que el vehículo pueda ser conducido, el tipo de tracción de las ruedas, como observamos en la figura 86. También contiene todo lo relacionado con las marchas del vehículo: número de marchas, revoluciones máximas y mínimas, el uso de elementos extras como el óxido nitroso (*NOS)* o *Turbo*. También podemos deseleccionar la opción *Generate All Gear Torque Curve* para poder configurar las curvas de las marchas a nuestro antojo.

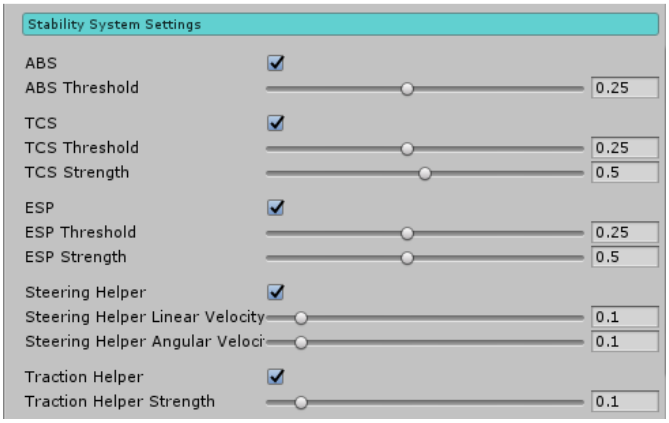

Figura 87. *Configuración de los sistemas de estabilidad*

Estabilidad (Stability): En esta configuración ajustaremos las características del sistema de antibloqueo de ruedas (*ABS*), el sistema de control de tracción (*Traction Control System, TCS*), el control de la estabilidad (*Electronic Stability Program, ESP*), la ayuda con el volante o la ayuda con la tracción. Podemos observar estos sistemas en la figura 87

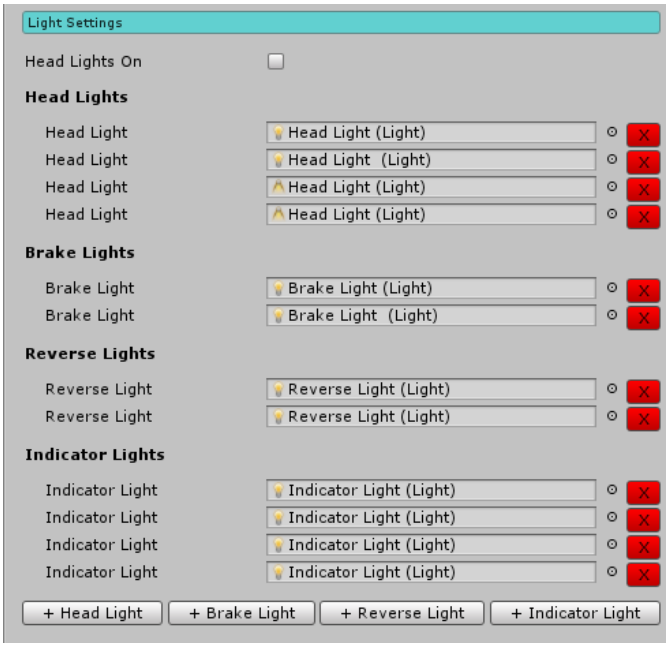

Figura 88. *Configuración de las luces*

Luces (*Lights*): En esta categoría introduciremos el número de luces que poseerá nuestro vehículo. Estas pueden ser luces delanteras (*Head Lights*), luces traseras (*Reverse Lights*), luces de freno (*Brake Lights*) o intermitentes para indicar la dirección (*Indicator Lights*). Encontramos la configuración de luces en la figura 88.

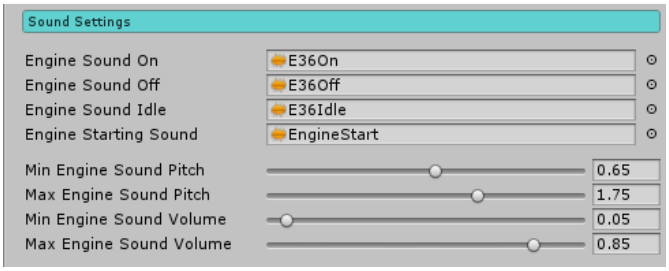

Figura 89. *Configuración de los sonidos*

Sonidos (*Sounds*): Observamos en la figura 89, la categoría los diferentes sonidos que vamos a usar en la simulación para el sonido del motor del vehículo (apagado, encendido, continuo, arranque).

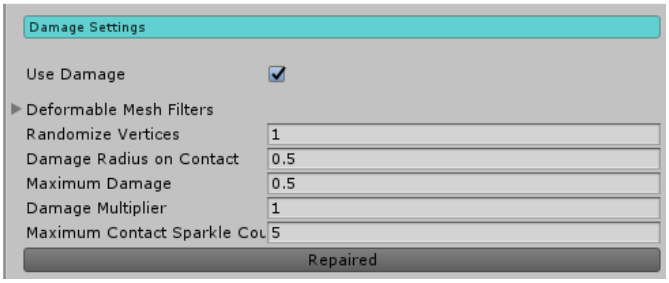

Figura 90. Configuración de daños

Configuración de los daños (*Damage Settings*): En esta categoría, mostrada en la figura 90, podemos indicar si queremos que los daños que pueda sufrir nuestro vehículo tengan impacto físico (deformaciones en el vehículo) y, en caso de estar activado, configuraremos el resultado de estos daños.

En el caso de que alguna componente o categoría del vehículo esté mal configurada, aparecerá un error en la comprobación del sistema (*System Overall Check*). Este error nos indicará qué categoría y componente precisa ser configurada. En caso de no existir errores, no nos aparecerá nada más que la etiqueta de comprobación del sistema, como observamos en la figura 91.

Figura 91. Mensaje de comprobación general del RCCv3

10.2.2 Car Camera

Para la simulación de la conducción usaremos como base de nuestra cámara el Script RCC_Camera.cs. Este *Script* configurará la cámara para que emule una visión realista de nuestro vehículo en diversos aspectos, como los siguientes:

- Los giros son más dinámicos. De esta forma, cuando el vehículo gire, eliminaremos la sensación de que el vehículo se está quieto y gira el plano en vez de él.
- Cámara marcha atrás: El *script* usará una posición de la cámara para cuando el vehículo está funcionando normalmente y cambiará de posición cuando activemos la marcha atrás, de tal forma que veremos el vehículo desde la parte frontal, teniendo una visión en el sentido que circula nuestro vehículo en todo momento.
- Podremos intercambiar cámaras mediante la tecla "C" (configurada por defecto). Esta opción permite cambiar entre las diferentes cámaras del Asset.

11 REFERENCIAS

[1] Las principales cifras de la Siniestralidad Vial en España en 2015
http://www.dgt.es/Galerias/seguridad-vial/estadisticas-e-indicadores/publicaciones/principales-cifras-siniestralidad/Las-principales-cifras-2015.pdf

[2] Dirección General de Tráfico Seguridad Vial
http://www.dgt.es/es/seguridad-vial/

[3] Estrategia de seguridad vial 2011-2020 http://www.dgt.es/Galerias/seguridad-vial/politicas-viales/estrategicos-2011-2020/doc/estrategico_2020_003.pdf

[4] Factores de riesgo en los accidentes de trafico
http://www.uv.es/sfpenlinia/cas/64las_causas_de_los_accidentes_de_trfico_factores_de_riesgo.html

[5] Conducción vehículo, factor humano
https://es.slideshare.net/paulbarce/transito-factor-humano

[6] Factor mecánico, seguridad activa y pasiva del vehículo
https://www.cea-online.es/blog/128-seguridad-activa-y-pasiva-del-vehiculo

[7] Factor ambiental, la vía y su entorno
http://www.uv.es/sfpenlinia/cas/643el_factor_ambiental_la_va_y_su_entorno.html

[8] DGT: Conducción eficiente
http://www.dgt.es/PEVI/documentos/catalogo_recursos/didacticos/did_adultas/Conduccion_eficiente.pdf

[9] Consejos para la conducción eficiente
https://www.ecointeligencia.com/2013/09/consejos-conduccion-eficiente/

[10] Cuida tu coche con la conducción eficiente
http://www.cochesdemarca.com/cuida-tu-coche-conoce-la-conduccion-eficiente/

[11] Clasificación de los diferentes modelos de vehículos
http://coches.idae.es/portal/basedatos/marcamodelo.aspx

[12] Guía de vehículos de venta en España con indicación de consumo y emisiones de CO2
http://coches.idae.es/PDF/GuiaFinalN.pdf

[13] Aspectos prácticos de la conducción eficiente
http://www.dgt.es/PEVI/documentos/catalogo_recursos/didacticos/did_adultas/Conduccion_eficiente.pdf

[14] Técnicas para una conducción eficiente
http://www.idae.es/uploads/documentos/documentos_10297_TREATISE_ConduccionEficiente_A2005_A_f3817bad.pdf

[15] Videojuegos creados con Unreal Engine
https://es.wikipedia.org/wiki/Anexo:Videojuegos_que_usan_Unreal_Engine

[16] Unity 3D
https://unity3d.com/es/unity

[17] Herramienta para el diseño 3D: Autodesk
https://www.autodesk.es/

[18] Herramienta para el diseño 3D: Autodesk 3ds Max
https://es.wikipedia.org/wiki/Autodesk_3ds_Max

[19] Herramienta para el diseño 3D: Autodesk Maya
https://es.wikipedia.org/wiki/Autodesk_Maya

[20] Herramienta para el diseño 3D: Blender
https://www.blender.org/

[21] Herramienta para el diseño 3D: Sketchup
https://www.sketchup.com/es

[22] Comparativa de herramientas para el diseño 3D
http://www.mundogeek.com/tutoriales/60-general/262-comparativa-blender-vs-maya

[23] Simulador
https://es.wikipedia.org/wiki/Simulador

[24] G27 Racing Wheel
http://support.logitech.com/en_us/product/g27-racing-wheel

[25] Driving Force GT
http://gaming.logitech.com/es-roam/product/driving-force-gt-gaming-wheel

[26] Motion Pro II Racing Simulator
https://www.cxcsimulations.com/products/motion-pro/

[27] The Most Realistic Racing Simulator
http://www.hammacher.com/Product/Default.aspx?sku=12677

[28] Los mejores simuladores de conducción
http://motorbit.com/los-mejores-simuladores-de-conduccion/?pais=

[29] Simulador Simescar
http://simumak.com/es/simescar

[30] Simulador DriveSimulator
http://drivesimsimulator.com/

[31] Simulador iRacing
http://www.iracing.com/

[32] Simulador ProjectCars
http://www.projectcarsgame.com/

[33] Simulador rFactor
http://www.rfactor.net/

[34] Lexus GS 450
http://www.lexusauto.es/car-models/gs/gs-450h/#Introduction

[35] Scripting en Unity
https://unity3d.com/es/learn/tutorials/s/scripting

[36] Gráficas de consumo
http://www.comoconsumirmenos.com/2013/01/como-conducir-con-pendientes-para_11.html