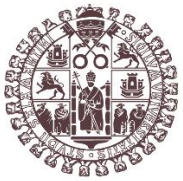


VNiVERSiDAD D SALAMANCA

Escuela politécnica superior de zamora

Grado en Ingeniería Mecánica

Creación de una aplicación 3D Interactiva de un Vehículo Radiocontrolado

Nombre: Bárbara María De Arriba Mangas

Departamento: Construcción y Agronomía

Área: Expresión Gráfica

Tutor: Manuel Pablo Rubio Cavero

Fecha de Adjudicación: noviembre 2018

Fecha de Presentación: septiembre 2019

A todos aquellos que confiaron en mí

y nunca me dejaron rendir.

Índice General

Introducción ____ *12*

1. Objetivos ____ **13**

2. Realidad Virtual, entornos 3D y sus ventajas e inconvenientes ____ **13**

3. Referencias ____ **15**

Capítulo 1: Memoria Técnica ____ *16*

1. Vehículos radiocontrolados ____ **17**

2. Piezas y conjuntos del vehículo ____ **18**

a. Sistema de suspensión ____ 18

b. Sistema de transmisión ____ 18

c. Chasis ____ 19

d. Ruedas ____ 19

e. Motor, emisora, servos y deposito gasolina ____ 20

f. Carrocería, protectores laterales, parachoques y alerón ____ 20

g. Otros elementos ____ 20

3. VR inmersiva y no inmersiva ____ **21**

4. Simulación del vehículo RC en realidad virtual ____ **21**

5. Referencias ____ **22**

Capítulo 2: Software y Hardware empleados. ____ *23*

1. 3DS Max (2018) ____ **24**

a. Introducción al programa ____ 24

b. Interfaz del programa ____ 24

c. Modelado ____ 25

d. Materiales ____ 26

e. Iluminación ____ 26

f. Animación ____ 26

g. Renderización ____ 27

2. De 3DS Max a Unreal Engine 4: DATASMITH ____ 27

a. Exportación e Importación ____ 28

3. Unreal Engine 4 ____ 30

a. Terminología ____ 30

b. Inicio ____ 33

c. Herramientas y editores ____ 33

4. Unreal Engine 4 y la Realidad Virtual ____ 36

a. Herramientas y dispositivos para VR ____ 36

b. HTC Vive ____ 38

c. UE4 y HTC Vive ____ 41

5. Referencias ____ 43

Capítulo 3: Desarrollo del proyecto ____ 44

1. Montaje del vehículo ____ 45

a. Montaje en realidad virtual inmersiva. ____ 47

b. Montaje en realidad virtual no inmersiva ____ 61

c. Materiales para las piezas ____ 68

d. Entorno y materiales empleados ____ 72

e. Iluminación ____ 79

f. Sonido ____ 80

2. Conducción del vehículo ____ 81

a. Importación del vehículo ____ 82

b. Entorno para la conducción y materiales empleados ____ 87

c. Iluminación ____ 90

3. Referencias ____ 91

Conclusiones ____ ***92***

Índice de Ilustraciones

Ilustración 1 - VR inmersiva VS. VR no inmersiva ____ *13*

Ilustración 2 - VR en medicina ____ *14*

Ilustración 3 - Imagen real del vehículo simulado ____ *17*

Ilustración 4 - Amortiguador, Trapecio y barra estabilizadora. ____ *18*

Ilustración 5 - Diferenciales ____ *18*

Ilustración 6 – Chasis ____ *19*

Ilustración 7 – Rueda ____ *19*

Ilustración 8 - Motor emisora, servos y depósito ____ *20*

Ilustración 9 - Carrocería, parachoques y alerón ____ *20*

Ilustración 10 - Placa de anclaje trasera ____ *20*

Ilustración 11 - Interfaz 3DS Max ____ *25*

Ilustración 12 - Barra de tiempo, deslizador y keyframe ____ *27*

Ilustración 13 - Descarga de Unreal Datasmith ____ *28*

Ilustración 14 - Export 3DS Max ____ *29*

Ilustración 15 - Exportar .UDATASMITH ____ *29*

Ilustración 16 - Importación en Unreal Engine ____ *29*

Ilustración 17 - Jerarquía de un proyecto ____ *30*

Ilustración 18 - Skeletal Mesh Actor ____ *31*

Ilustración 19 - Camera Actor ____ *32*

Ilustración 20 - Buscador de proyectos ____ *33*

Ilustración 21 - Editor de nivel 34

Ilustración 22 - Editor de materiales 34

Ilustración 23 - Sistema de nodos 35

Ilustración 24 - Editor Blueprint 35

Ilustración 25 - HTC Vive 37

Ilustración 26 - Primer modelo Samsung Gear VR 37

Ilustración 27 - Google Cardboard 38

Ilustración 28 - HTC Vive 39

Ilustración 29 - Casco HTC Vive y elementos 39

Ilustración 30 - Controladores 39

Ilustración 31 - Configuración Área de juego 40

Ilustración 32 – SteamVR 41

Ilustración 33 - Modo de juego en Preferencias del Editor>Mapas y Modos 42

Ilustración 34 - Componentes a agregar en peón VR 42

Ilustración 35 – Eye Level VS. Floor Level 43

Ilustración 36 - Animación inicial 45

Ilustración 37 - Desplazamiento de llave hasta fotograma 0. 46

Ilustración 38 - Herramienta Group. 46

Ilustración 39 - Conjuntos del vehículo. 47

Ilustración 40 - Plantilla VR UE4. 48

Ilustración 41 - Entorno de la plantilla. 48

Ilustración 42 - Ejemplo de materiales creados por UE4 _____ *48*

Ilustración 43 - Ruta para llegar a asignaciones de acción. _____ *49*

Ilustración 44 - Controles del mando y asignaciones de acción. _____ *49*

Ilustración 45 - Asignaciones de eje. _____ *50*

Ilustración 46 - Ejes panel táctil del Trackpad. _____ *50*

Ilustración 47 - Localización BP Pick Up Cube _____ *51*

Ilustración 48 - Blueprint Pick Up Cube _____ *51*

Ilustración 49 - Blueprint MotionController. _____ *52*

Ilustración 50 - Ejemplos de programación en Blueprint MotionController. _____ *52*

Ilustración 51 - Piezas y conjuntos separados. _____ *53*

Ilustración 52 - Colisión en malla estática. Amortiguador. _____ *53*

Ilustración 53 - Simular físicos. _____ *54*

Ilustración 54 - Coche original y referencia importados y visibles. _____ *55*

Ilustración 55 - Comparación posiciones. _____ *55*

Ilustración 56 - Comparación de orientaciones _____ *56*

Ilustración 57 - Puerta AND. _____ *56*

Ilustración 58 - Visibilidades de las piezas original y referencia. _____ *57*

Ilustración 59 - Visibilidad de las piezas. _____ *57*

Ilustración 60 - Nodos que se unen en EventTick. _____ *58*

Ilustración 61 - Plataforma cilíndrica. _____ *58*

Ilustración 62 - Chasis unido a plataforma. _____ *59*

Ilustración 63 - Programación giro de la plataforma. _____ *59*

*Ilustración 64 - Menú Ayuda.*_____ *60*

Ilustración 65 - Blueprint Widget. _____ *60*

Ilustración 66 - Programación Menú Ayuda. _____ *61*

Ilustración 67 - Plantilla First Person. _____ *61*

*Ilustración 68 - Ruta BP FirstPersonCharacter.*_____ *62*

Ilustración 69 - Comparativa arma y manos. _____ *62*

Ilustración 70 - Programación Pick Up parte 1. _____ *63*

Ilustración 71 - Programación Pick Up parte 2. _____ *64*

Ilustración 72 - Programación Pick Up parte 3. _____ *64*

*Ilustración 73 - Programación Drop.*_____ *65*

Ilustración 74 - Programación Event Tick _____ *65*

*Ilustración 75 - Ubicación de Held Object Location.*_____ *65*

Ilustración 76 - Control de Pick Up y Drop. _____ *66*

Ilustración 77 - Cubo cogido por personaje. _____ *66*

Ilustración 78 - Conjunto creado. _____ *67*

*Ilustración 79 - Movimiento jugador*_____ *67*

*Ilustración 80 - Material Carrocería.*_____ *69*

*Ilustración 81 - Material Rough_2*_____ *69*

Ilustración 82 - Material Goma_Neumático _____ *69*

*Ilustración 83 - Material Metal_Chrome*_____ *70*

Ilustración 84 - Material MetalBlackPlain 70

Ilustración 85 - Material Metal ChromeFast 70

Ilustración 86 - Material Esponja filtro de aire 71

Ilustración 87 - Material Plástico Verde 71

Ilustración 88 - Material Rough1 71

Ilustración 89 - Habitación garaje 72

Ilustración 90 - Creación Material. 73

Ilustración 91 - Material Suelo. 73

Ilustración 92 - Material Paredes y Techo. 74

Ilustración 93 - Mesas Madera. 74

Ilustración 94 - Banco de trabajo. 75

Ilustración 95 - Armario de Madera. 75

Ilustración 96 - Herramientas. 76

Ilustración 97 - Panel de Herramientas. 76

Ilustración 98 - Creación de las ventanas. 77

Ilustración 99 - Puerta de interior. 77

Ilustración 100 - Puerta Garaje. 78

Ilustración 101 - Lámparas. 78

Ilustración 102 - Garaje. 78

Ilustración 103 - Garaje 2. 79

Ilustración 104 - Complejidad de la luz. 80

Ilustración 105 - Colocación sonido ambiente _____ *81*

Ilustración 106 - Sonido Ding. _____ *81*

Ilustración 107 - Plantilla Vehículo _____ *82*

Ilustración 108 - Posición Ejes en huesos. _____ *82*

Ilustración 109 - Ubicación de los huesos. _____ *83*

Ilustración 110 - Jerarquía de huesos. _____ *83*

Ilustración 111 - Giro solidario hueso y rueda. _____ *84*

Ilustración 112 - BP Ruedas Delanteras VS. Traseras. _____ *84*

Ilustración 113 - Blueprint de animación. _____ *85*

Ilustración 114 - Vehículo y cámara (BP del vehículo). _____ *85*

Ilustración 115 - Vehicle Setups _____ *86*

Ilustración 116 - Físicas de la malla esquelética. _____ *86*

Ilustración 117 - Materiales del vehículo en malla esquelética. _____ *86*

Ilustración 118 - Controles conducción. _____ *87*

Ilustración 119 - Pared de ladrillo. _____ *88*

Ilustración 120 - Suelo césped. _____ *88*

Ilustración 121 - Pista en 3DS Max y en UE4. _____ *89*

Ilustración 122 - Terreno rugoso. _____ *89*

Ilustración 123 - Rampas metálicas. _____ *90*

Ilustración 124 - Circuito de conos. _____ *90*

Introducción

1. Objetivos

Este trabajo es desarrollado con el fin de obtener una aplicación utilizando la realidad virtual (VR) en la que se pueda realizar el montaje de un vehículo de radiocontrol, así como su conducción en una pista.

Para ello, previamente se han estudiado los programas empleados que más adelante se mencionarán, se han diseñado los entornos, modificado los materiales, y programado las diferentes tareas que se verán en la aplicación. Se va a tratar de realizar unos entornos realistas para que el usuario se acerque todo lo posible a la realidad a través de la aplicación.

También será necesario aprender sobre los sistemas de RV, el entorno 3D, los sistemas que se utilizan en la actualidad y cómo funcionan, y sus beneficios e inconvenientes.

2. Realidad Virtual, entornos 3D y sus ventajas e inconvenientes

La VR se compone de dos entornos fundamentales que se conectan entre sí mediante una interfaz: el entorno virtual y el entorno del usuario. Este intercambio de información entre usuario y entorno virtual se realiza con la ayuda del tacto, la visión o el sonido.

Habrá dos tipos de VR; la **inmersiva**, en la que, a través de mandos, guantes, cascos y otros dispositivos, se registra la posición y rotación de las diferentes partes del cuerpo del usuario para facilitar la interacción tridimensional con un entorno creado por ordenador; y la **no inmersiva**, que empleará la pantalla de ordenador para acercar al usuario a diferentes espacios y poder moverse e interactuar en ellos. El alto coste de los dispositivos inmersivos ha facilitado el desarrollo de dispositivos más sencillos y que utilizan la VR no inmersiva, como videoconsolas o juegos de ordenador a través de Internet.

Ilustración 1 - VR inmersiva VS. VR no inmersiva

Los inicios de la VR se remontan a la II Guerra Mundial, con la creación de simuladores de vuelo para entrenamiento de bombarderos. Por tanto, aunque inicialmente se tuvieran fines militares, a lo largo de las décadas se ha aplicado la VR a diferentes sectores como la educación, el entretenimiento, la medicina y psicología, la formación, y por supuesto, el sector de los videojuegos que quizás sea el más conocido.

Debido a la aplicación en la formación y preparación, la VR hace posible que el usuario pueda enfrentarse a situaciones reales sin poner en peligro a nadie. Es esta una de sus principales **ventajas**. La versatilidad de esta tecnología hace que se haya implantado en centros educativos, universitarios y que sea especialmente útil en arquitectura o ingeniería, pero sobre todo en medicina. De este modo, y uniendo robótica y VR, se ayudará a la formación de los estudiantes para adquirir diferentes habilidades, poder explorar y conocer órganos y partes del cuerpo humano y aprender a utilizar dispositivos complejos. Los beneficios en este campo son numerosos, destacando el poder evitar la práctica real con pacientes, reducir los costes, reproducir la anatomía del paciente con exactitud, etc. En la actualidad, se incluyen dispositivos de VR en consultas de enfermos de Alzheimer, y en consultas de psicología.

Ilustración 2 - VR en medicina

Cabe mencionar que la tecnología de VR también cuenta con **desventajas**, sobre todo el alto coste ya mencionado de equipos de VR inmersiva (que ya está siendo solucionado actualmente), la necesidad de mejora en cuanto a la resolución, y los diferentes problemas sociales que pueden desencadenarse si el usuario llegase a preferir un mundo virtual al mundo real.

3. Referencias

[1] www.filmora.wondershare.com/es/virtual-reality/disadvantages-virtual-reality.html

[2] www.ehowenespanol.com/ventajas-desventajas-realidad-virtual-info_89195/

[3] www.baboonlab.com/blog/noticias-de-marketing-inmobiliario-y-tecnologia-1/post/realidad-virtual-y-medicina-usos-y-aplicaciones-27

Capítulo 1:

Memoria Técnica

1. Vehículos radiocontrolados

Un vehículo radiocontrolado (RC) es un vehículo realizado a escala y que es conducido por el usuario mediante un aparato de radio de manera inalámbrica (también denominado enlace hertziano).

En el RC se tienen en cuenta tres aspectos fundamentales:

- **Electrónica**. Emite las señales desde el mando receptor a la antena emisora, y viceversa.
- **Electricidad**. Necesaria para ciertos componentes eléctricos y electrónicos.
- **Mecánica**. Transforma en movimiento mecánico las señales eléctricas.

Esta industria, denominada industria del **automodelismo**, incluye desde juguetes hasta modelos profesionales, existiendo competiciones en diferentes categorías. Parámetros como la escala (desde 1:6 hasta 1:87), el tipo de motor (motor eléctrico, motor térmico), o la tracción van a determinar la clasificación de estos vehículos.

El proyecto del que se parte se titula ***"Diseño y simulación del funcionamiento de los sistemas mecánicos de suspensión y transmisión de un vehículo de radio control a escala 1:8"*** de Enrique Manuel Silva González. Para su realización, se midieron las piezas una a una, reflejando las medidas en el ordenador en el programa *Solid Edge V20*. El ensamblaje de las piezas tuvo lugar en el entorno conjunto del mismo programa, para terminar realizando la animación realista en 3DS Max. Esta animación es lo que fue proporcionado para comenzar con el proyecto que se desarrolla en esta memoria.

Ilustración 3 - Imagen real del vehículo simulado

2. Piezas y conjuntos del vehículo

En este apartado se procede a enumerar los elementos que forman el vehículo, según su función. En total se cuenta con 31 conjuntos.

a. Sistema de suspensión

Este sistema, encargado de mantener el contacto de las ruedas con el suelo, está formado por amortiguadores, trapecios y barra estabilizadora. Esta última se encarga de evitar que las ruedas interiores pierdan tracción y apoyo en las curvas.

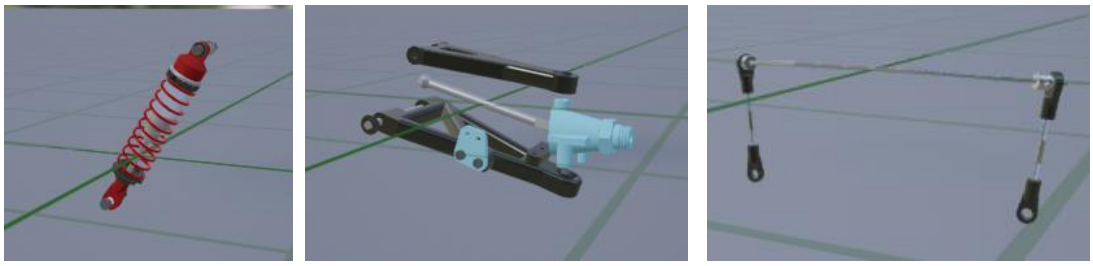


Ilustración 4 - Amortiguador, Trapecio y barra estabilizadora.

b. Sistema de transmisión

Formado por los diferenciales, que permiten que tanto las ruedas interiores como las exteriores giren lo que les corresponde exactamente y no pierdan adherencia; los ejes de transmisión, tanto el delantero como el trasero están conectados al diferencial central; los palieres y el mecanismo de transmisión Cardam.

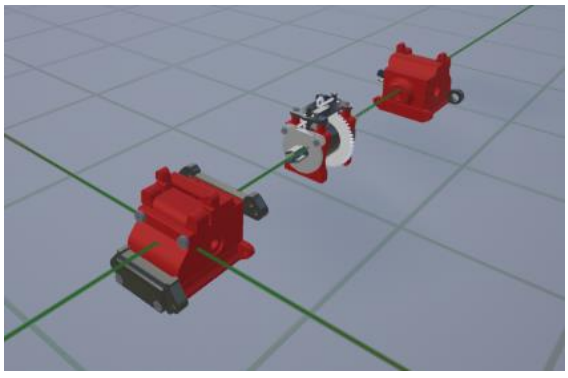


Ilustración 5 - Diferenciales

c. Chasis

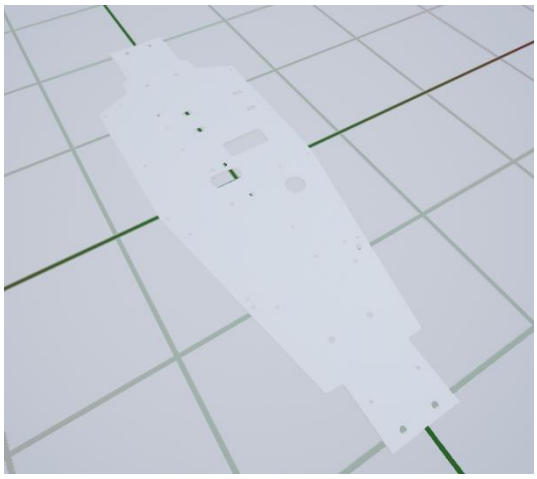


Ilustración 6 – Chasis

d. Ruedas

Cada rueda se compone de la llanta y el neumático, dos piezas individuales agrupadas en un mismo conjunto.

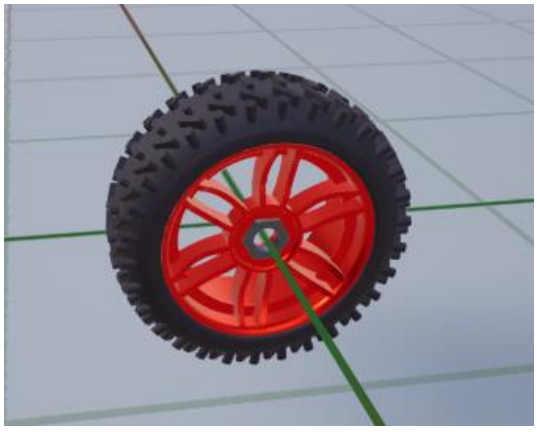


Ilustración 7 – Rueda

e. Motor, emisora, servos y deposito gasolina

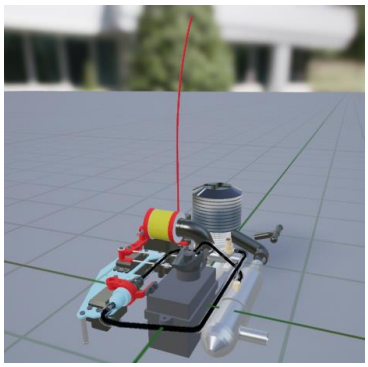


Ilustración 8 - Motor emisora, servos y depósito

f. Carrocería, protectores laterales, parachoques y alerón

Elementos de protección y aerodinámica, esenciales para el vehículo.

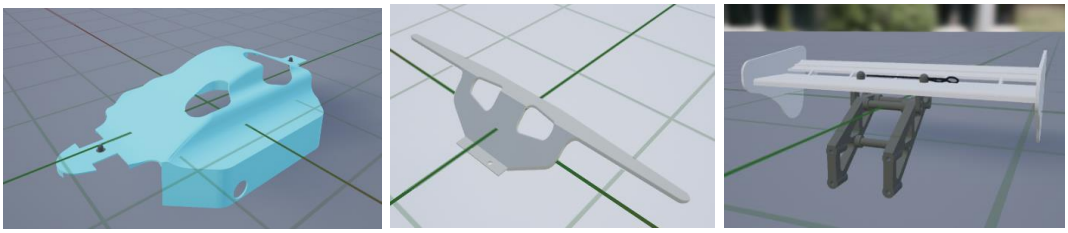


Ilustración 9 - Carrocería, parachoques y alerón

g. Otros elementos

Como las cañas de inclinación, o las placas de anclaje delantera y trasera a las que se unen los amortiguadores.

Ilustración 10 - Placa de anclaje trasera

3. VR inmersiva y no inmersiva

Como ya se había mencionado en la introducción, la VR puede ser de dos tipos: inmersiva y no inmersiva. En ambas, el objeto principal es que el usuario conecte con un entorno complejo, creado por ordenador, e interactúe con el mismo. Esta interacción será tridimensional y en tiempo real.

En la VR ***inmersiva***, el usuario dispone de un conjunto de dispositivos adicionales con los que va a poder interactuar en el entorno. Dichos dispositivos registran en tiempo real la ubicación y rotación del cuerpo del usuario, y serán explicados detalladamente más adelante.

En cuanto a la VR ***no inmersiva***, lo principal es que permite la interacción, pero no es necesario utilizar dispositivos adicionales que no sean un ordenador, la pantalla, el teclado o el ratón.

4. Simulación del vehículo RC en realidad virtual

La aplicación que se va a explicar y desarrollar va a constar de diferentes niveles. En ellos, se diferenciará montaje y conducción, y su vez se podrá elegir si se van a emplear los dispositivos para hacer la experiencia inmersiva, o si por el contrario se realizará de forma no inmersiva, mediante teclado y ratón. Se va a partir de un modelo de vehículo RC previamente diseñado en 3D.

Para completar esta aplicación, ha sido necesario emplear diferentes programas, ya que se modelan y editan las piezas y el entorno en 3DS Max (2018), se editan imágenes en Adobe Photoshop CC 2018, o se utiliza Unreal Engine (UE4) versión 4.21 como motor de programación en tiempo real, entre otras tareas que se mencionarán más adelante.

El resultado al que se quiere llegar es obtener una simulación muy cercana a la realidad en la que se puedan manipular y observar las diferentes piezas del vehículo, realizar su montaje, y crear un entorno en el que el vehículo se pueda conducir.

5. Referencias

[1] web.archive.org/web/20120326075345/http://tecnica.carbi.net/newpage.html

Capítulo 2: Software y Hardware empleados.

1. 3DS Max (2018)

a. Introducción al programa

Autodesk 3DS Max es un software para el modelado 3D, empleado en videojuegos, arquitectura, animación, etc. Permite también la animación de los objetos creados y la renderización. Una de las principales ventajas de esta herramienta es la interacción con *AutoCAD*, permitiendo importar y vincular archivos *.dwg* y *.dxf*. Este tipo de archivos corresponden a programas como el ya mencionado *AutoCAD*, *Architectural Desktop* o *Viz Render*, que tienen un gran desarrollo en el campo de la arquitectura.

Cuando se comienza un proyecto en 3DS Max, se suelen seguir unos determinados pasos o secuencia:

- **Recopilar** información sobre el proyecto a realizar y que sirva como ayuda visual.
- **Modelar** los elementos, creando, modificando o importando archivos de otros programas para así conseguir los elementos de nuestro entorno o escena.
- Realizar la **iluminación** y **texturas**, añadiendo **materiales**, puntos de luz y texturas para obtener una mayor sensación de realidad.
- **Renderizar** la escena.
- **Postproducción**, con programas de edición de vídeo e imágenes.

b. Interfaz del programa

3DS Max va a contar con diferentes menús, barras de herramientas y opciones, y también con la posibilidad de tener varias ventanas en las que se colocarán las diferentes vistas: alzado, planta, perspectiva, vista de cámara concreta, etc. Entre las barras y menús, destacan:

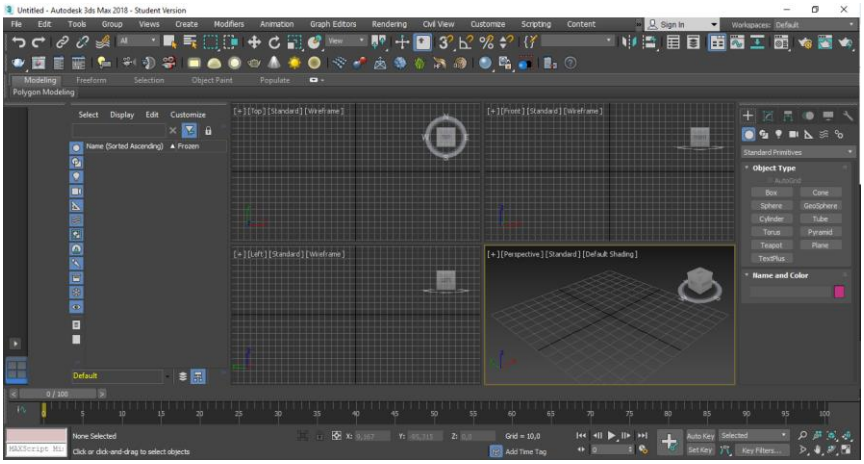


Ilustración 11 - Interfaz 3DS Max

- *Barra de menús*: con todas las opciones del programa
- *Menú Edit*: barra de edición (deshacer, rehacer, clonar, seleccionar, propiedades…)
- *Menú Tools*: o herramientas, como capas, luces, alineaciones, simetrías…
- *Menú View*: cambios en las vistas, cuadrículas, etc.
- *Menú Create*
- *Modifiers*
- *Render*: para renderizar la escena
- *Materials*: abre el editor de materiales
- *Control de reproducción de animación y regulador de tiempo*

c. Modelado

Mediante el modelado, se podrán realizar todos los objetos del mundo real, ya que todos se pueden descomponer en objetos con formas más básicas. 3DS Max permite realizar desde las formas más básicas, como una caja, hasta todo tipo de formas, líneas, textos, terrenos, operaciones booleanas, muros, follaje, elementos arquitectónicos o escaleras, entre otros.

Además, se podrán ejecutar operaciones de selección, transformaciones, rotaciones, movimientos, clonaciones, modificaciones, etc.

d. Materiales

Junto con la iluminación, forman una parte muy importante para dotar de realismo a los objetos. Su creación se basa en los mapas o imágenes de referencia, junto con el modificador Mapa UVW para poder visualizarlos correctamente. 3DS Max va a permitir crear un número ilimitado de materiales y visualizarlos sin necesidad de renderizar la escena.

e. Iluminación

En 3DS Max existen tres tipos de luces. Las luces estándar y las fotométricas están incorporadas al programa, y el tercer tipo lo proporcionan plugins como por ejemplo V-Ray.

Las luces estándar pueden generar cualquier iluminación, como una lámpara o el sol. También se pueden crear sombras, modificar parámetros e incluir efectos especiales, además de incluir parámetros de foco y luces direccionales.

No obstante, 3DS Max ofrece una iluminación global mediante un sistema de luz diurna proporcionando dos fuentes de luz: el cielo y el sol. Este sistema es denominado *Daylight System*.

f. Animación

Mediante la animación se podrá dotar de movimiento a los objetos, modificando su posición, orientación, tamaño, etc. El inicio y fin del movimiento se lleva a cabo mediante los keyframes, o puntos de partida clave donde existen estas modificaciones del objeto. Para crear una animación sólo se necesitan los keyframes, ya que el resto de los fotogramas los genera 3DS Max automáticamente.

Para crear un keyframe, hay dos opciones: *Key auto*, que lo hace de forma automática cuando el objeto es modificado de alguna manera a lo largo de la línea o regulador de tiempo; y mediante *Set key*, en la que el usuario define el keyframe.

Ilustración 12 - Barra de tiempo, deslizador y keyframe

A lo largo de la barra de tiempo tendrá lugar la animación de la escena, y el deslizador permite desplazarse a través de los fotogramas. El tiempo puede configurarse ya que por defecto se crean 100 fotogramas. La cantidad de fotogramas y otros muchos parámetros se podrán modificar en *Time Configuration*.

g. Renderización

El objetivo del renderizado es poder ver la iluminación y su influencia en las geometrías. Es necesario ver sus efectos y poder así modificar la iluminación y texturas para poder acercarse más aún a la realidad. Hay dos motores integrados en 3DS Max, Default Scanline Render y Mental Ray; y también existen otros que funcionan como plugins, como V-Ray.

2. De 3DS Max a Unreal Engine 4: DATASMITH

Una vez visto el software de 3DS Max hay que conectar los archivos con Unreal Engine 4, para poder disponer de ellos en este último. La manera para pasar escenas y objetos al motor de programación es mediante DATASMITH.

Unreal Datasmith es un conjunto de herramientas en forma de plugin para Unreal Engine que hace que el proceso de importación y manipulación de datos de fuentes de CAD sea más eficiente.

A través de la aplicación *Epic Games Launcher* y accediendo a la pestaña *Biblioteca*, junto a *Unreal Engine*, se llega a un menú llamado *Beta de Unreal Studio*. Es en este menú donde se podrá descargar *Unreal Datasmith* e instalarla en el motor.

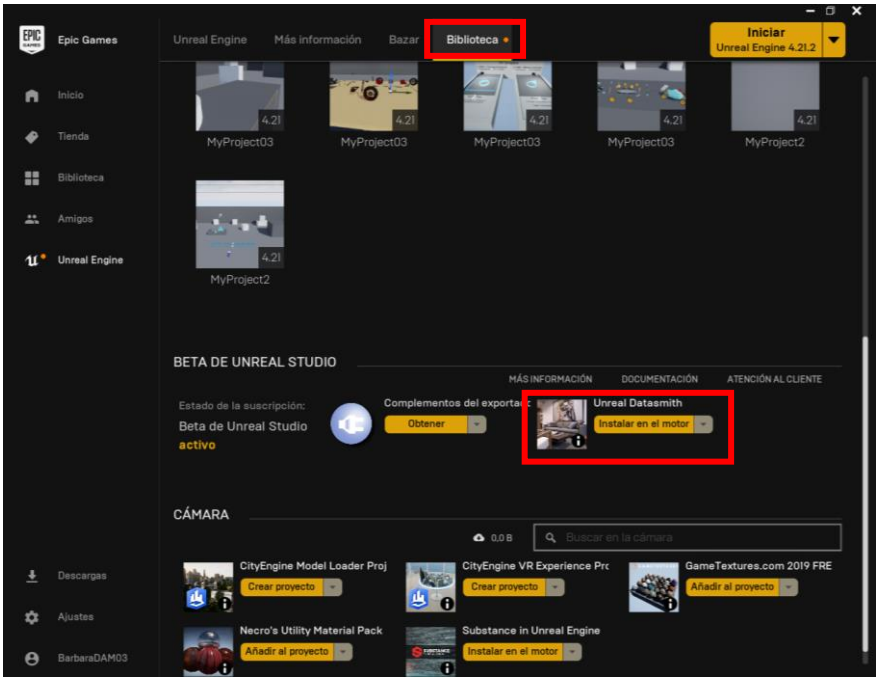


Ilustración 13 - Descarga de Unreal Datasmith

a. Exportación e Importación

Ambos procesos mediante Datasmith son muy sencillos y se van a describir brevemente a continuación.

- <u>Exportación de archivos 3DS Max</u>

Para exportar archivos en formato Datasmith desde 3DS Max, habrá que acceder a *File > Export > Export* y seleccionar en el tipo de archivo *.UDATASMITH*.

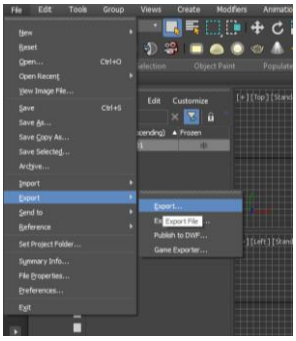


Ilustración 14 - Export 3DS Max

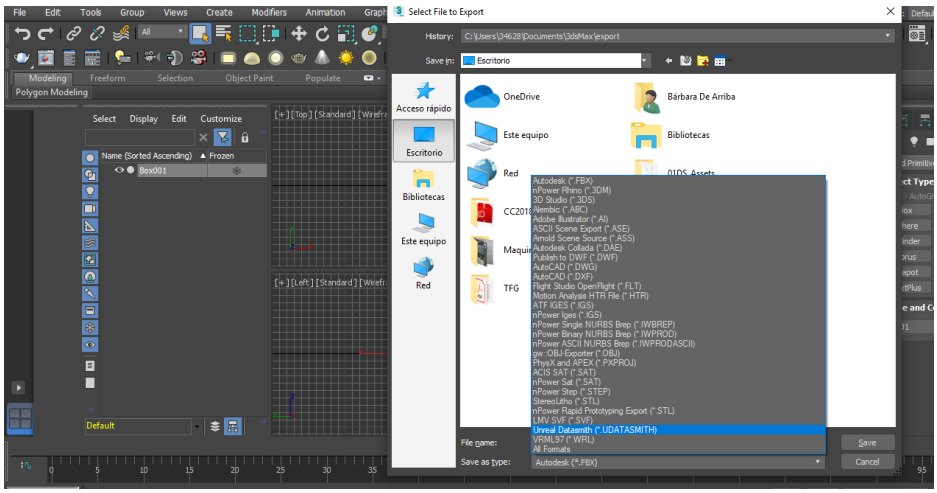


Ilustración 15 - Exportar .UDATASMITH

Con la exportación se generan archivos *.udatasmith* que ya podrán ser importados.

- Importación de archivos en Unreal Engine

En Unreal Engine, importar archivos Datasmith en muy sencillo. Tan sólo habrá que seleccionar la opción y el archivo que queremos importar.

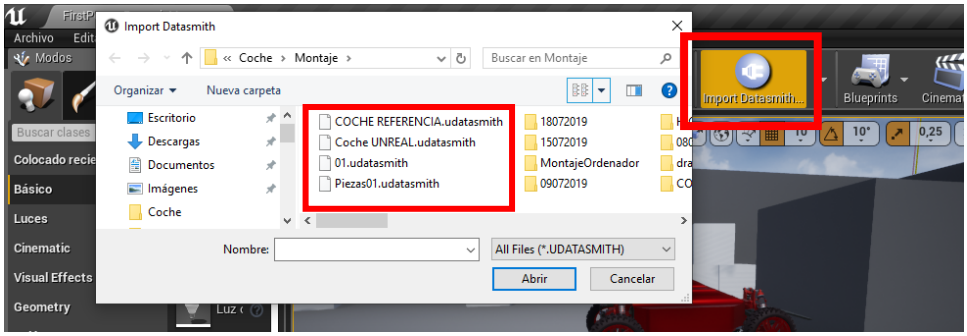


Ilustración 16 - Importación en Unreal Engine

Una vez que los archivos estén importados, habrá que dejar que el programe cargue los materiales y degradados, y después ya se podrá trabajar con dichos archivos.

3. Unreal Engine 4

Unreal Engine 4 (UE4) es un conjunto de herramientas de desarrollo para trabajar con tecnología en tiempo real. Este motor de videojuegos fue creado por Epic Games. UE4 es utilizado en la actualidad por muchos desarrolladores de juegos, y su código está escrito en C++. Actualmente se trabaja con su versión 4, que trabaja con numerosas plataformas tales como Microsoft Windows, macOS, Linux, SteamOS, HTML5, iOS, Android, PlayStation 4, Nintendo Switch, Xbox One SteamVR/HTC Vive, Oculus Rift, PlayStation VR, Google Daydream, OSVR y Samsung Gear VR.

a. Terminología

Esta sección es importante debido a que en UE4 se emplean términos específicos de este programa.

Proyecto

Un **proyecto** es una unidad autónoma con el contenido y código que forman un juego o aplicación individual, y coincide con un conjunto de directorios en el disco del ordenador en el que se está trabajando.

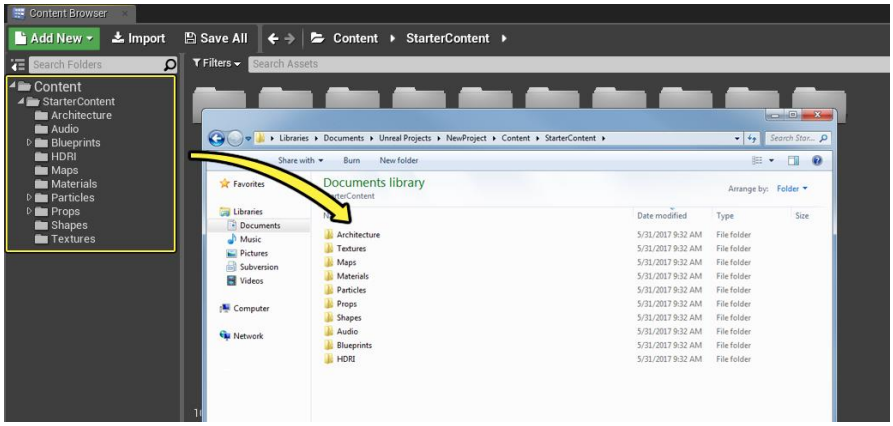


Ilustración 17 - Jerarquía de un proyecto

Objetos y actores

Los **objetos** son los componentes básicos de UE4 y contienen todas las funcionalidades esenciales para que la aplicación se desarrolle de manera correcta.

Un **actor** es un objeto que se coloca en un nivel. Este tipo de objeto puede ser rotado, trasladado o escalado, y también se pueden generar y destruir a través del juego. Los actores pueden ser de diferentes tipos.

Algunos ejemplos de actores son *StaticMeshActor*, *CameraActor* y *PlayerStartActor*. Para crear un nivel, se colocan actores en un mapa, se mueven para crear un entorno, y se modifican sus propiedades para verlos o para que se comporten de una forma específica.

StaticMeshActor hace referencia a un tipo simple de actor que muestra una malla. A pesar de que por defecto son estáticos, este comportamiento puede ser editado. Un *SkeletalMeshActor* es un tipo de actor con una malla animada de manera que su geometría se podrá deformar mediante animaciones creadas en aplicaciones de animación 3D.

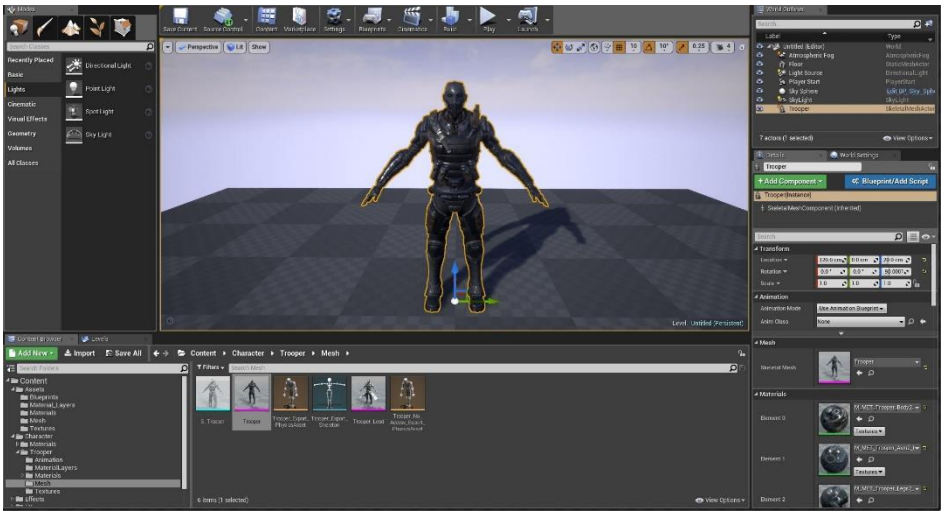


Ilustración 18 - Skeletal Mesh Actor

Con un *CameraActor* se establecerá un punto de vista para el juego, mientras que con *PlayerStartActor* se coloca el punto donde el juego va a comenzar.

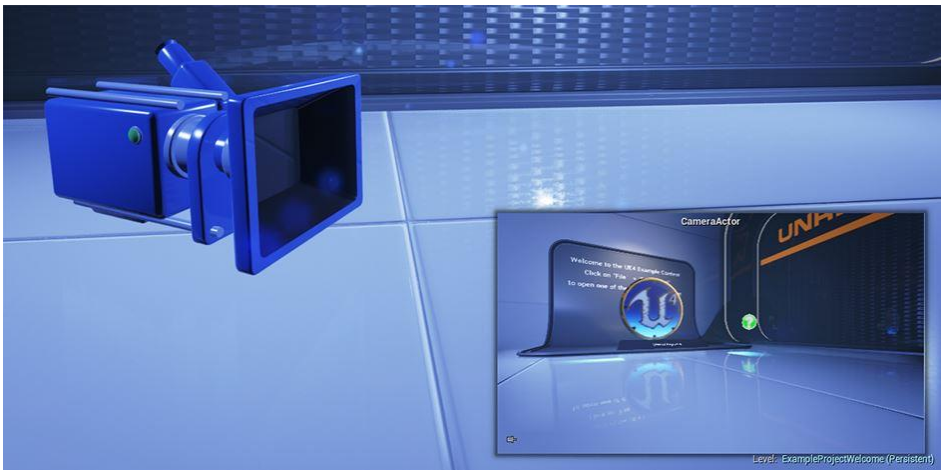


Ilustración 19 - Camera Actor

Clases

Las **clases** son las que definen el comportamiento y propiedades de objetos y actores dentro del juego que se está creando. Pueden crearse en Blueprints o en C++, y además tienen jerarquía. Esto quiere decir que una clase adquiere la información de su clase primaria o padre, y que a su vez transmitirá esta información a sus hijos.

Componentes

Un **componente** es una herramienta que da funcionalidad al actor al que se agrega, ya que no pueden existir por sí mismos.

Peones y personajes

Los **peones** son subclases de actores y funcionan como avatares dentro del juego. Los **personajes** son subclases de los peones, cuyo objetivo es ser usado como el personaje para el jugador.

Niveles y mundo

Un **nivel** o mapa es un área de juego definida por el jugador. Son creados, visualizados y modificados mediante la colocación y transformación de los actores y sus propiedades.

El conjunto o lista de diferentes niveles cargados forman el **mundo**.

b. Inicio

Cuando se inicia UE4, se abre una pestaña en la que se pueden elegir diferentes características, o si el proyecto va a partir de una plantilla, va a ser un proyecto en blanco o un proyecto ya guardado en disco.

Ilustración 20 - Buscador de proyectos

c. Herramientas y editores

En UE4 se trabaja con diferentes tipos de Editores para controlar y diseñar los objetos de los proyectos. Los más empleados se describen a continuación.

Editor de Niveles

Es el editor principal para construir los niveles de juego. En él, se define el área de juego mediante actores, geometrías, planos, Blueprints, etc.

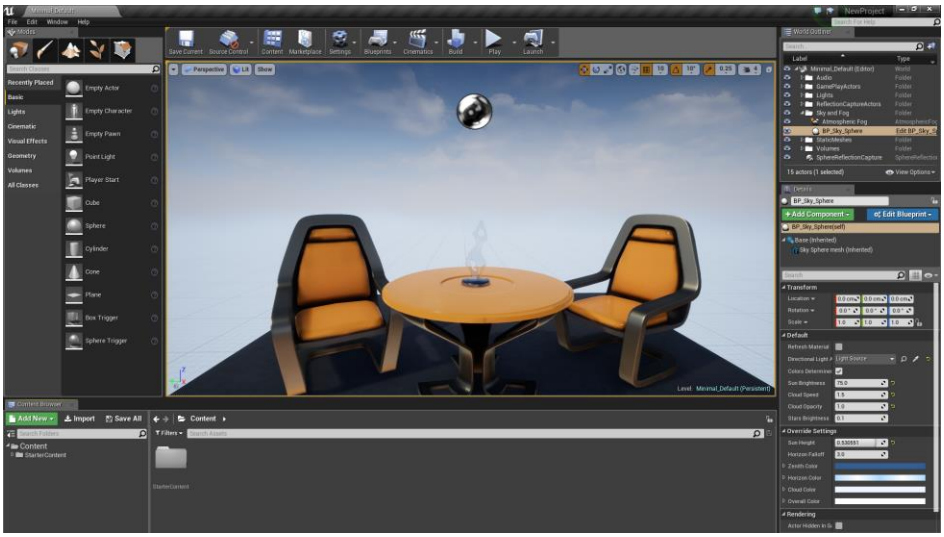


Ilustración 21 - Editor de nivel

Con el menú de la izquierda se pueden agregar elementos, desde los más básicos como actores, peones, cubos, esferas, etc, hasta componentes de luz, efectos visuales, volúmenes o geometrías.

Editor de materiales

En este nivel se crean y editan los materiales activos que serán aplicados a una malla para controlar su aspecto en el juego. Es una interfaz gráfica basada en nodos que permite aplicar los materiales a las mallas estáticas y esqueléticas.

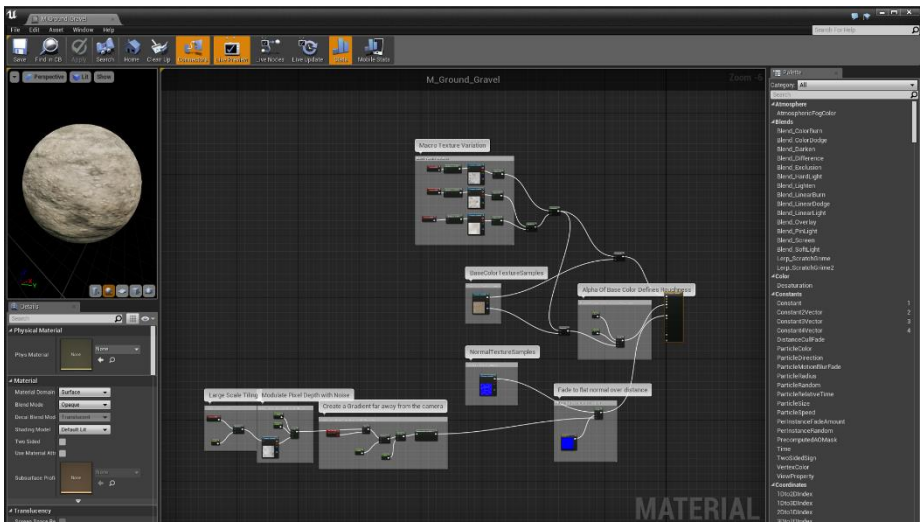


Ilustración 22 - Editor de materiales

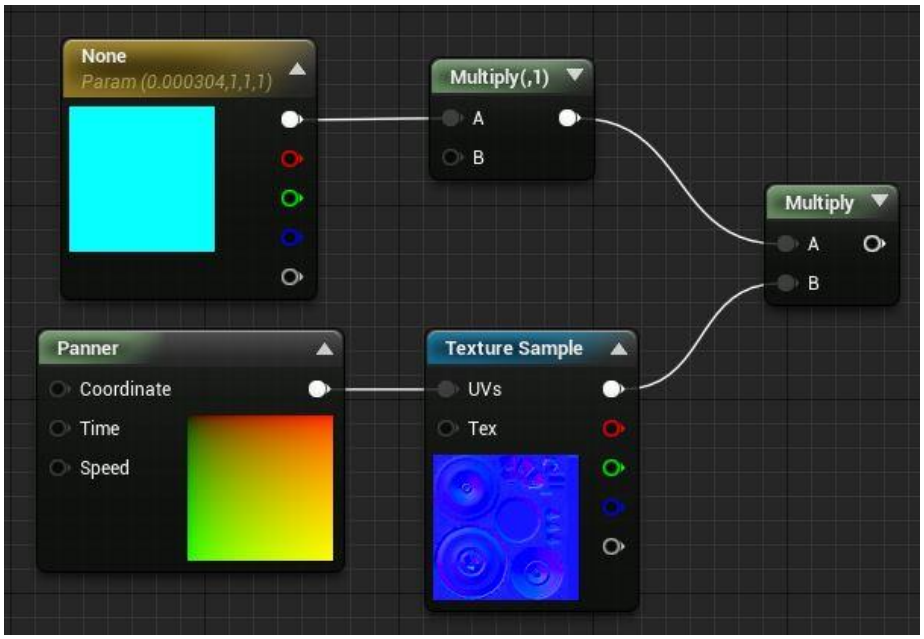


Ilustración 23 - Sistema de nodos

Editor de Blueprints

En este editor se trabaja y modifican los Blueprints, que son activos especiales usados para crear tipos de actores y eventos del nivel sin tener que escribir código C++. Mediante la unión de nodos, eventos, funciones y variables se podrán crear todos los elementos de la aplicación.

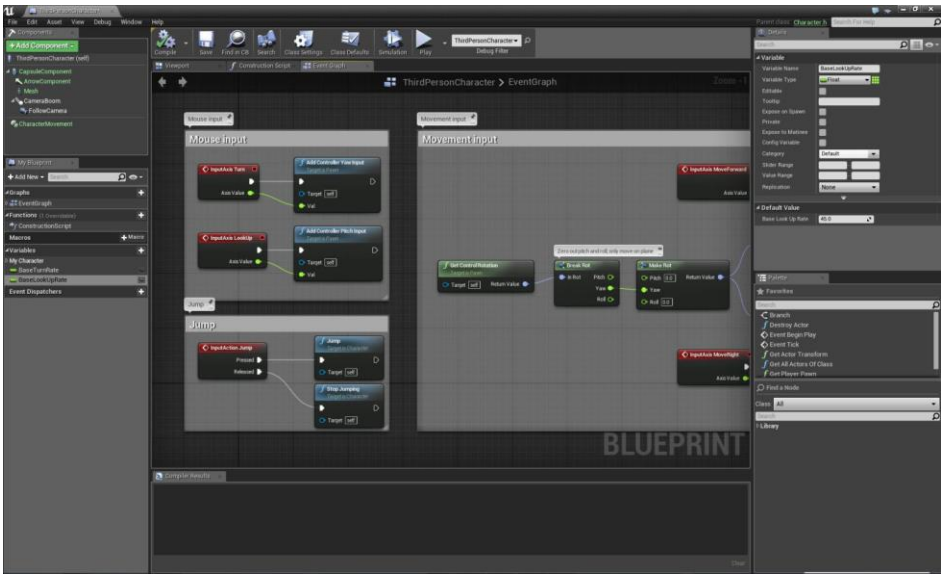


Ilustración 24 - Editor Blueprint

El comportamiento de los actores podrá ser controlado creando Blueprints para cada uno de ellos, a parte del Blueprint del nivel. Estos Blueprints tendrán la *ventana gráfica*, para manejar propiedades del actor; el *script*; y el *gráfico del evento*, cuyo funcionamiento es igual que el de los Blueprint de nivel, pero con algún evento y función particular.

4. Unreal Engine 4 y la Realidad Virtual

Debido a la rápida evolución de las tecnologías de realidad aumentada, virtual y mixta, UE4 permite construir equipo, activos y flujo de trabajo con diversas herramientas, como el escalado de escenas simples muy detalladas, entornos y personajes. UE4 está diseñado para la alta exigencia de aplicaciones como juegos AAA[1], filmación y visualización fotorreal, y es elegido por marcas líderes mundiales para dar vida a sus historias.

Para el desarrollo de una aplicación de VR van a ser necesarios dispositivos y software específicos.

a. Herramientas y dispositivos para VR

Los **controladores de movimiento** (motion controller) son dispositivos que controlan el juego mediante el uso de acelerómetros y otros sensores que rastrean el movimiento y proporcionan la información.

Los **cascos o gafas**, también llamados HMD (head-mounted display) pueden ser de dos tipos, diferenciando los que llevan pantalla de los que son una carcasa en la que se introduce un smartphone.

Particularizando para UE4, los HMD que soporta son los siguientes:

[1] Juego AAA: es un juego que destaca en todos sus apartados. Generalmente un juego triple AAA es un juego que tenga A en jugabilidad, A en gráficos y A en sonido.

- *HTC Vive*: es el sistema más avanzado y el que se utilizará en este proyecto. Son el resultado de la unión de HTC (hardware) y Valve (software SteamVR).

Ilustración 25 - HTC Vive

- *Samsung Gear VR*: desarrollado por Samsung Electronics en colaboración con Oculus VR, funciona introduciendo modelos de smartphone de Samsung de alta gama, con o sin controlador de mano.

Ilustración 26 - Primer modelo Samsung Gear VR

- *Oculus Rift* (por Oculus VR)
- *Play Station VR*: desarrolladas por Sony como complemento para ciertos videojuegos en Play Station 4 y Play Station 4 Pro.
- Otros modelos, como las Open *Source Virtual Reality (OSVR)* o los dos tipos desarrollados por Google: *Google VR* y *Google Cardboard*.

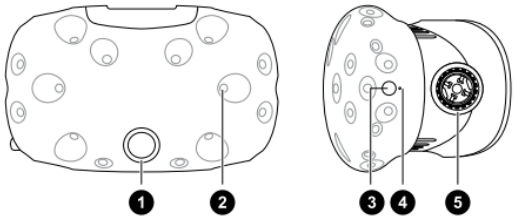
Ilustración 27 - Google Cardboard

b. HTC Vive

Este sistema, que debido a su potencia permite mirar en cualquier dirección sin fricciones, es uno de los más avanzados en la actualidad. El HTC Vive tiene una gran adaptabilidad, permitiendo el desarrollo del juego en una habitación grande o en una más pequeña (se sugiere de 2 m x 1.5 m). Se podrá interactuar con cualquier objeto del entorno virtual mediante sus controladores o mandos inalámbricos.

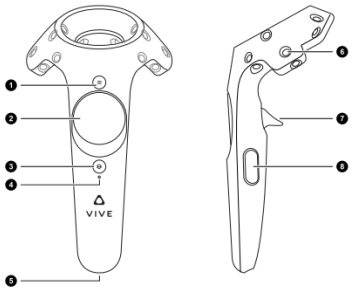
Cuenta con dos mandos o controladores inalámbricos, dos estaciones base, el casco y una caja de enlace para conectar este último al PC. Por esto, el mayor inconveniente del Vive es la multitud de cables, tomas de corriente y ranuras a conectar.

Ilustración 28 - HTC Vive



1	Lentes de la cámara
2	Sensor de seguimiento
3	Botón de casco
4	Luz de estado
5	Perilla para control de distancia del lente

Ilustración 29 - Casco HTC Vive y elementos



1	Botón de menú
2	Trackpad (Panel táctil)
3	Botón de sistema
4	Luz de estado
5	Puerto micro USB
6	Sensor de seguimiento
7	Disparador
8	Botón de agarre

Ilustración 30 - Controladores

Las bases habrán de ser colocadas a cierta altura del suelo y con una distancia de separación entre ellas, de forma que se sitúen en esquinas opuestas. Estas bases son conectadas a la red. El casco será conectado a una placa con entrada USB, HDMI y de audio, y ésta también se conectará a la red y a su vez al PC. Los mandos funcionan con baterías, por lo que son inalámbricos, y sólo habrá que encenderlos para que tanto el casco como las bases los detecten, ya que estarán vinculados al primero.

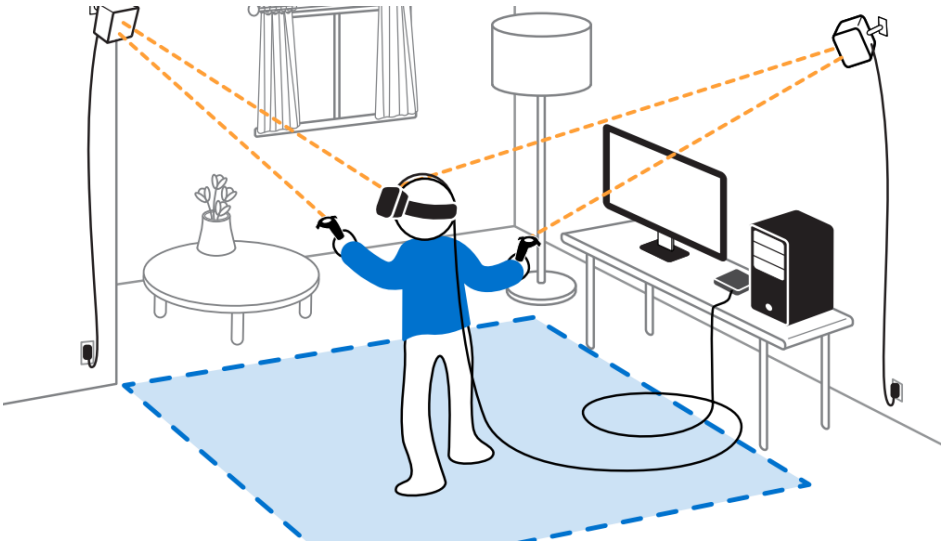


Ilustración 31 - Configuración Área de juego

Cualquier información adicional o ayuda podrá ser consultada en el manual que se incluye como *Guía para el usuario*. Con el montaje realizado de forma correcta, el siguiente paso es descargar Steam y SteamVR, para después calibrar los controladores y el casco en el área de juego establecida.

Otro aspecto importante para tener en cuenta será si el PC con el que se va a trabajar está listo para la VR. Se puede descargar un software de comprobación para el PC desde la página web de HTC Vive, pero los requisitos básicos para HTC Vive son los siguientes:

- GPU: NVIDIA GeForce GTX 970 / AMD Radeon R9 290 equivalente o superior
- CPU: Intel i5-4590 / AMD FX 8350 equivalente o superior
- Memoria: 4GB+ RAM
- Salida de vídeo: HDMI 1.4 o DisplayPort 1.2 o posterior

- Puerto USB: 1 puerto USB 2.0 o superior
- Sistema operativo: Windows 7 SP1 o superior

Ordenador del laboratorio de realidad virtual (E.P.S. Zamora). El equipo con el que se ha trabajado reúne las siguientes características:

- CPU: Intel i7
- GPU: NVIDIA GeForce RTX 2070
- Memoria RAM: 16GB
- Sistema operativo: Windows 10
- Puertos USB y salida HDMI

c. UE4 y HTC Vive

Para que UE4 detecte el sistema HTC Vive tan sólo habrá que abrir SteamVR cuando ya se haya conectado el casco al PC y a la red. Se muestran unos iconos en verde cuando SteamVR detecta que el elemento está conectado.

Ilustración 32 – SteamVR

Ya en UE4, si se parte de un proyecto nuevo, será necesario crear una carpeta donde guardar los Blueprints de modo de juego y de peón, además de configurar el modo de juego en preferencias del editor – mapas y modos para que por defecto cargue el Blueprint creado de VR. Habrá que hacer lo mismo con el peón de VR.

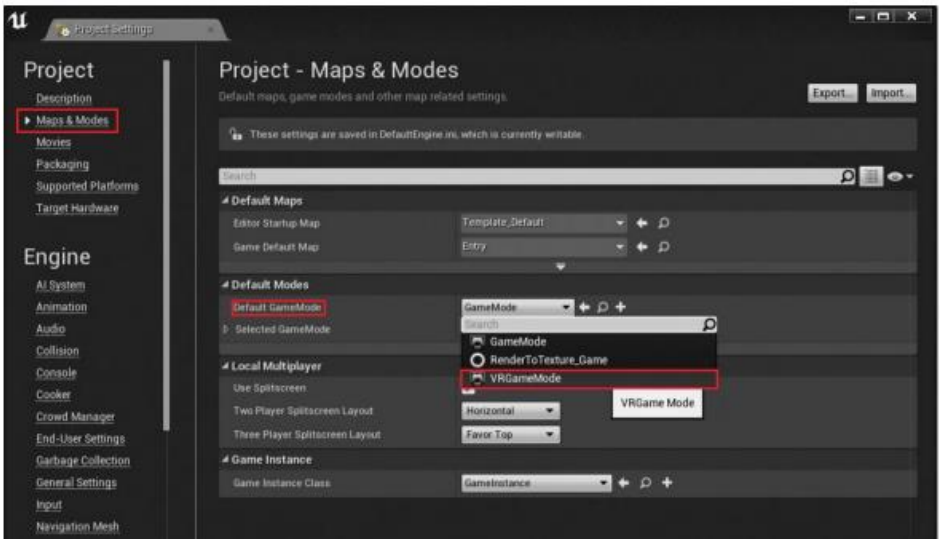


Ilustración 33 - Modo de juego en Preferencias del Editor>Mapas y Modos

Se deberá abrir la configuración del peón y agregar un componente de escena, uno de cámara, y dos para los controles de movimiento de las manos izquierda y derecha.

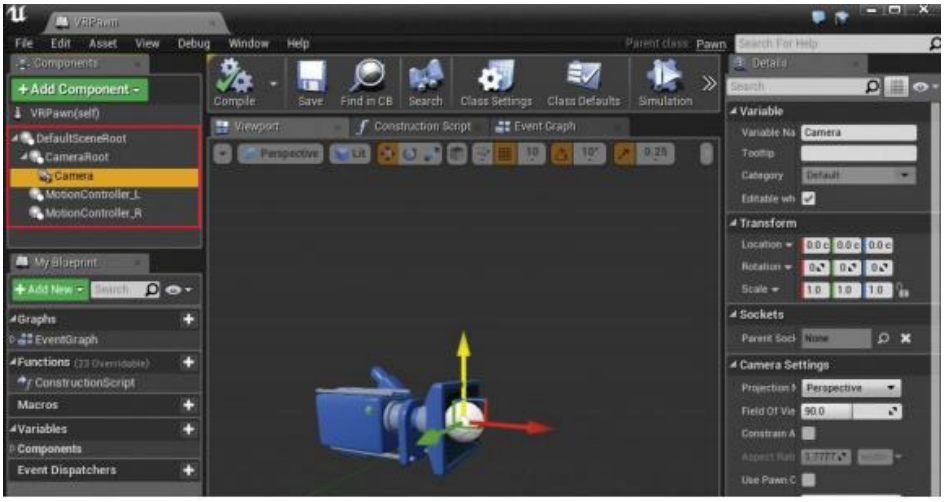


Ilustración 34 - Componentes a agregar en peón VR

En el gráfico del evento se deberá unir el evento Begin Play con una función llamada Set Tracking Origin, para establecer el punto de vista desde el que comienza el juego: sentado (Eye Level) o de pie (Floor Level). La diferencia entre ambas es que *Eye Level* ubica la cámara haciendo que el punto de vista apunte hacia dentro del monitor en el área de juego, mientras que *Floor Level* la ubica en el centro del espacio sobre el suelo.

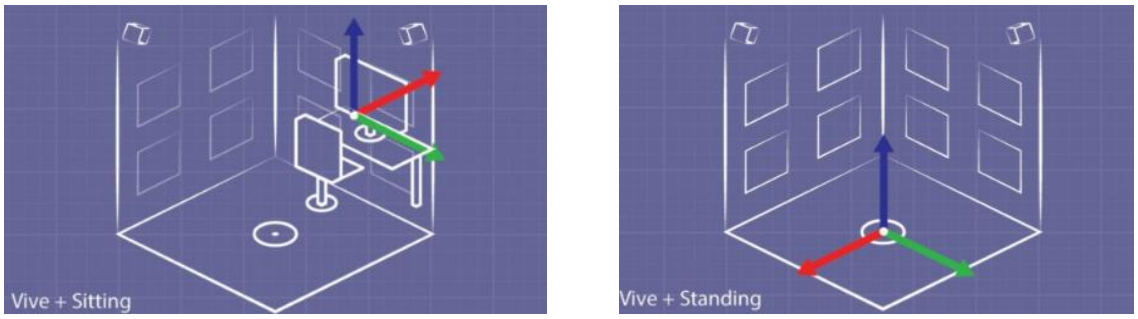


Ilustración 35 – Eye Level VS. Floor Level

5. Referencias

[1] www.foro3d.com/f112/manual-de-3d-studio-max-8-por-el-instituto-tecnologico-de-durango-60725.html#post528649

[2] docs.unrealengine.com/en-US/GettingStarted/Terminology

[3] docs.unrealengine.com/en-US/GettingStarted/SubEditors

[4] es.wikipedia.org/wiki/Unreal_Engine#Unreal_Engine_4

[5] es.wikipedia.org/wiki/Realidad_virtual#Controladores

[6] images-na.ssl-images-amazon.com/images/I/A14rqSnQyyL.pdf

[7] androidpit.es/pc-realidad-virtual-requisitos

[8] es.gizmodo.com/htc-vive-analisis-al-mejor-casco-de-realidad-virtual-1769110462

[9] Mitch McCaffrey, Chapter 1. Terminology and Best Practices de Unreal Engine VR Cookbook, pp. 17 – 20 y 25.

[10] Mitch McCaffrey, Chapter 2. Head Mounted Display Setup de Unreal Engine VR Cookbook, pp. 46 – 53.

Capítulo 3: Desarrollo del proyecto

Para explicar el desarrollo de la aplicación hay que tener en cuenta que ésta tiene diferentes partes. Como ya se ha mencionado, se realiza el montaje del vehículo por conjuntos de piezas tanto en realidad virtual inmersiva, como en realidad virtual no inmersiva. A continuación, se realiza una simulación de la conducción, en ambos tipos de realidad virtual también. Es por esto que este capítulo se va a dividir para así poder diferenciar cada parte y no mezclar las tareas que se han llevado a cabo.

Es importante mencionar de dónde se ha partido para comenzar este proyecto. Se proporcionó un archivo de escena de 3DS Max, en el cual se reproducía un vídeo del montaje por conjuntos.

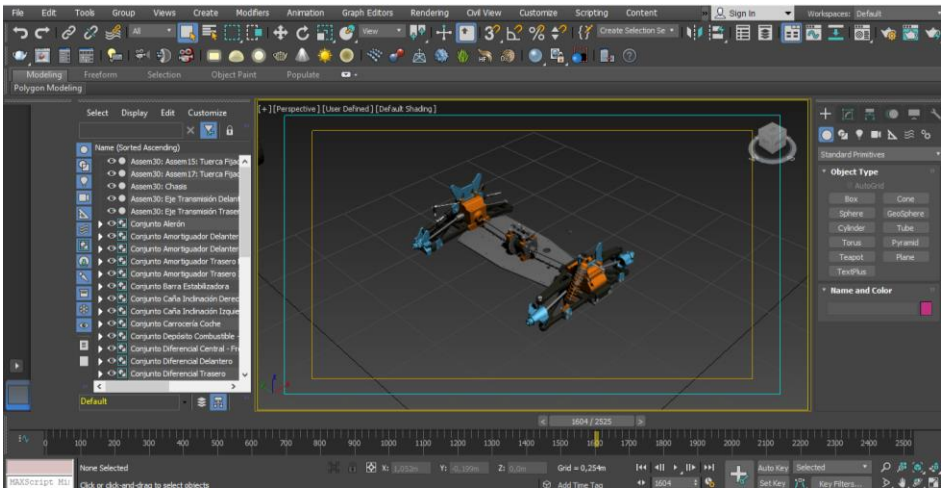


Ilustración 36 - Animación inicial

1. Montaje del vehículo

El primer paso fue eliminar las animaciones y hacer visibles todas las piezas desde el principio, usando **3DS Max**. Para ello, se eliminaron las llaves (keys) que hacían aparecer cada pieza, y se desplazaron las llaves que definen la posición final de cada pieza hasta el inicio de la barra de tiempo. De este modo, todas las piezas del vehículo eran visibles desde el principio. Esto se hace debido a que si no se mueven las llaves, el archivo que se exporta en Datasmith es el del fotograma inicial, por lo que las piezas que tengan la posición definida en otro fotograma no serían exportadas.

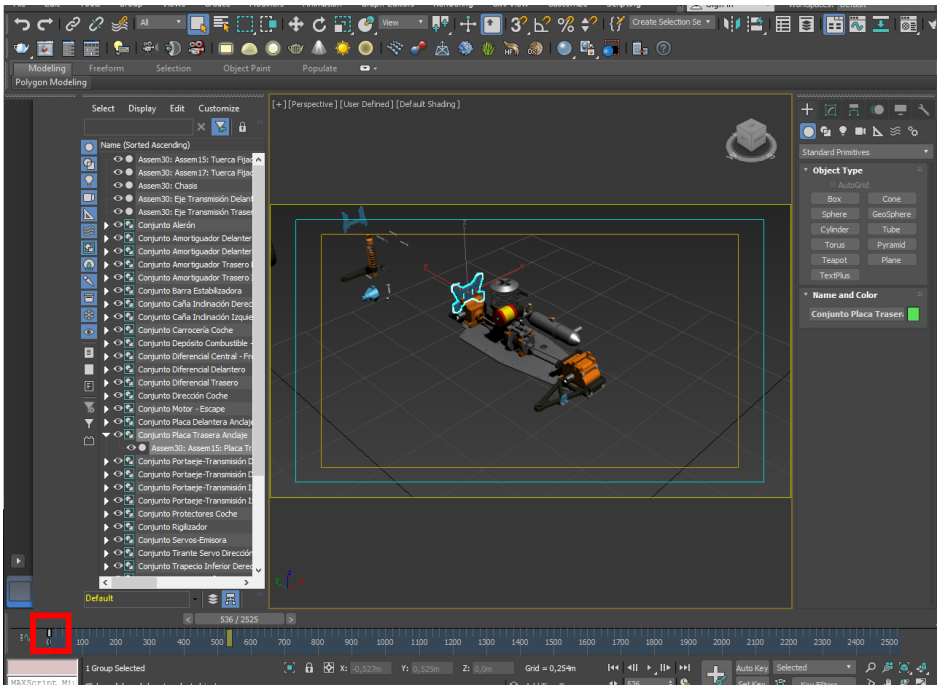


Ilustración 37 - Desplazamiento de llave hasta fotograma 0.

A continuación, se definen los conjuntos que van a componer el montaje del vehículo. Mediante la opción *Group* se agrupan diferentes piezas y subconjuntos que van a formar cada parte, como el alerón, las ruedas, los amortiguadores, etc. Estos conjuntos fueron establecidos tras estudiar la mejor forma para realizar el montaje en realidad virtual, de manera que se puedan montar piezas relativamente grandes para que el resultado no fuera una aplicación larga, si no que fuera lo más interactiva, sencilla y atractiva posible.

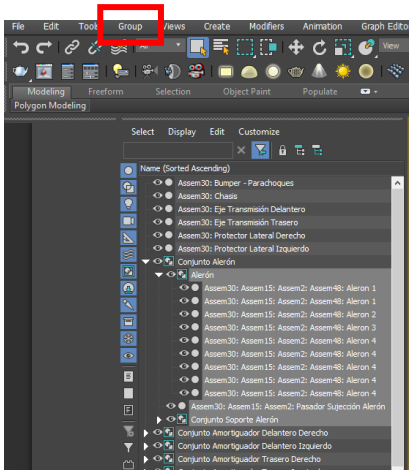


Ilustración 38 - Herramienta Group.

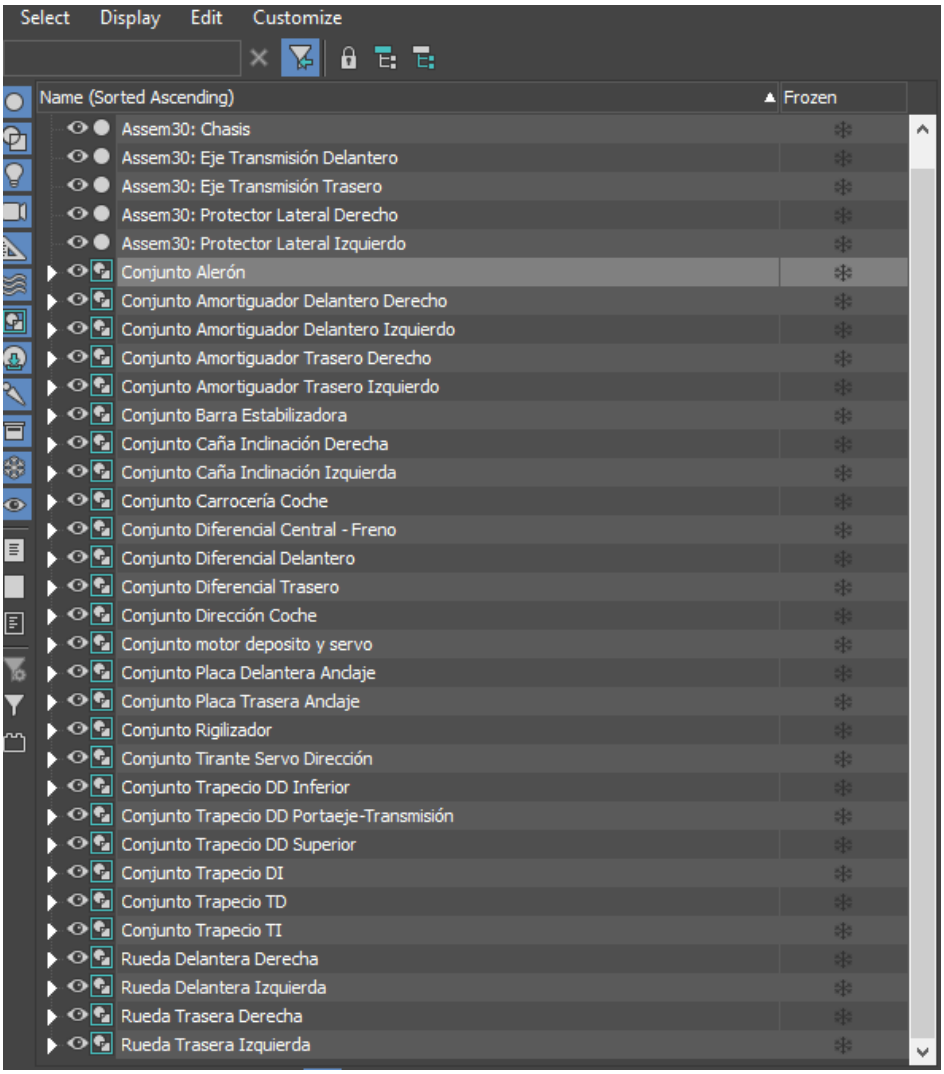


Ilustración 39 - Conjuntos del vehículo.

Se procede a la exportación del archivo y sus conjuntos mediante el formato DATASMITH como ya se había comentado, y a la importación de éste en **Unreal Engine 4**.

a. Montaje en realidad virtual inmersiva.

Ya en UE4, no se abre un proyecto nuevo si no que aprovechando que el programa cuenta con algunas plantillas, se inicia la de realidad virtual. En esta plantilla, conectando el sistema HTC Vive, el usuario se puede desplazar por un entorno muy sencillo, y con los mandos, coger, mover y lanzar cubos.

Ilustración 40 - Plantilla VR UE4.

Ilustración 41 - Entorno de la plantilla.

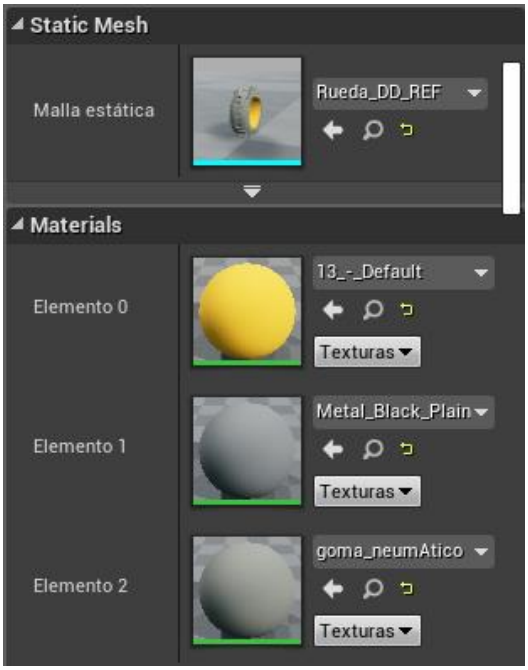


Ilustración 42 - Ejemplo de materiales creados por UE4

Lo primero que se hace es importar el archivo Datasmith del vehículo separado por conjuntos. Al importarlo, UE4 proporciona unos materiales basándose en los materiales que se habían definido en 3DS Max *(Ilustración 35)*. Estos materiales serán modificados y explicados más adelante. El entorno también será modificado más adelante, por lo que el archivo es importado en un lugar cualquiera del entorno de la plantilla.

Uno de los aspectos fundamentales es conocer los controles del juego establecidos en la plantilla. Para conocerlos, se acude a *Ajustes del proyecto*, en el menú *Editar*, y dentro de estos ajustes, a *Motor* y *Entrada*. Se encuentran dos tipos de asignaciones: **asignaciones de acción** y **asignaciones del eje**.

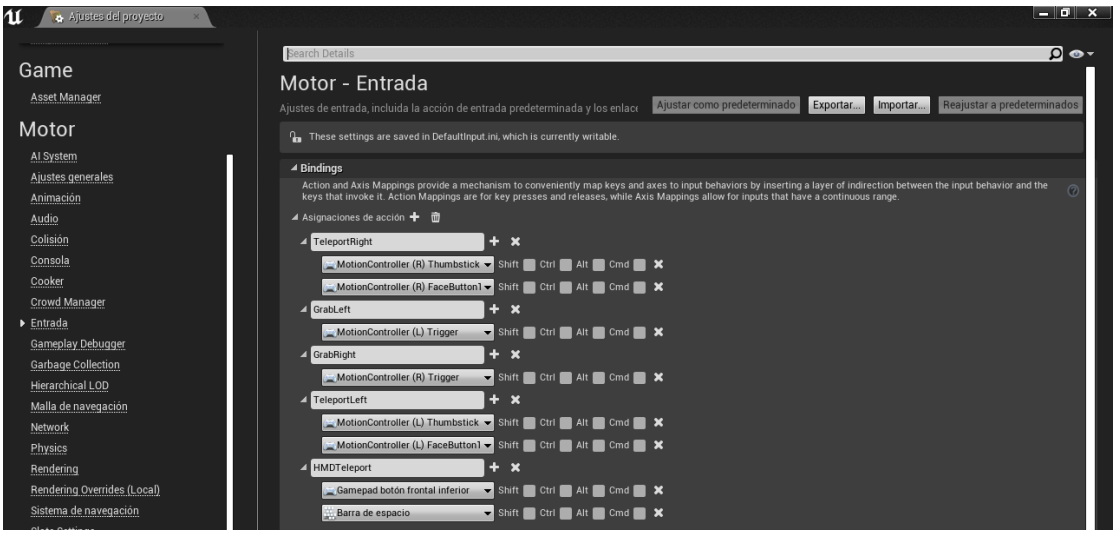


Ilustración 43 - Ruta para llegar a asignaciones de acción.

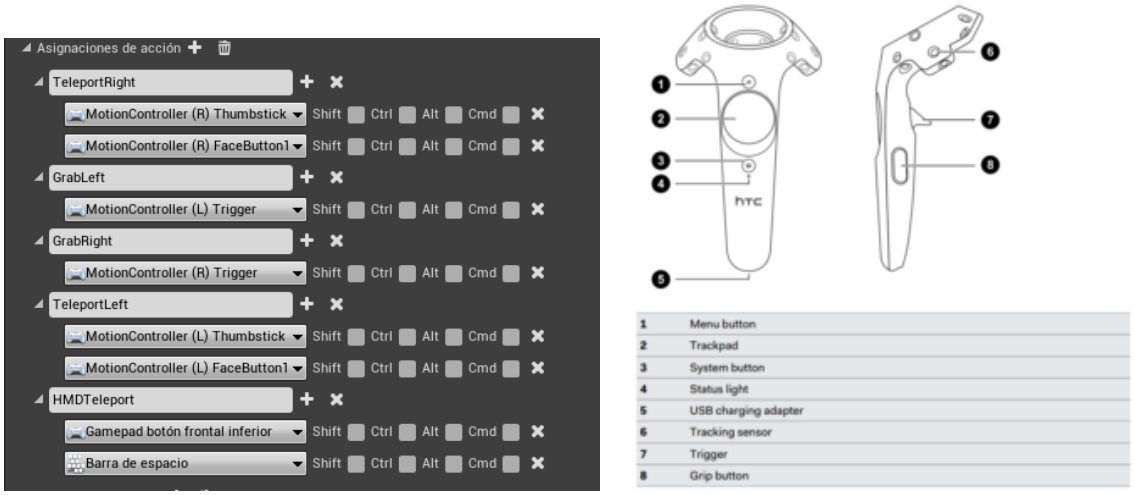


Ilustración 44 - Controles del mando y asignaciones de acción.

Los botones y acciones que interesan para este proyecto debido a que son los que se utilizan, serán:

- ***Teleport Right***: para moverte por el entorno con el mando derecho, se utilizará el *Trackpad* mediante una pulsación y apuntando con el mando. Cuando se tenga el lugar al que se quiere llegar, se suelta el botón y automáticamente el jugador ya se habrá desplazado.
- ***Teleport Left***: igual que el anterior, pero con el controlador de la mano izquierda.
- ***Grab Right***: mediante el gatillo del mando (*Trigger*) se podrán coger y dejar objetos con el mando derecho.
- ***Grab Left***: lo mismo, con el mando izquierdo.

En cuanto a las asignaciones del eje, la plantilla trae las siguientes:

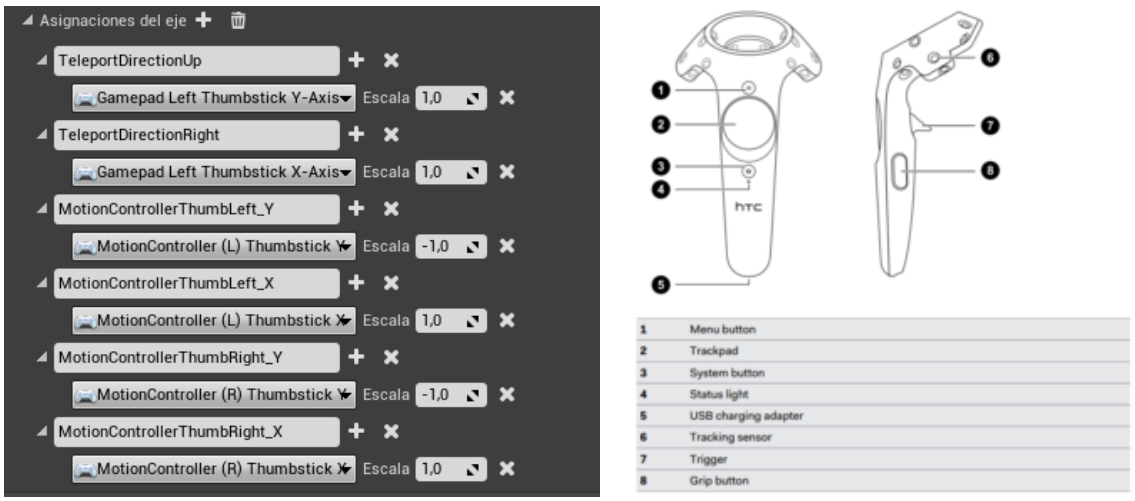


Ilustración 45 - Asignaciones de eje.

El Trackpad es un botón circular que además es táctil. De este modo, se comporta como si además de botón, fuera un joystick.

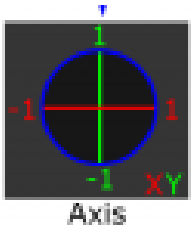


Ilustración 46 - Ejes panel táctil del Trackpad.

- ***MotionControllerThumbRight_X***: dentro del mando derecho, recoge el movimiento en el eje X.
- ***MotionControllerThumbRight_Y:*** movimiento en eje Y.
- ***MotionControllerThumbLeft_X:*** eje X, mando izquierdo.
- ***MotionControllerThumbLeft_Y:*** eje Y, mando izquierdo.

Cabe mencionar que se podrán añadir nuevas asignaciones, tanto de acción como de eje, en cualquier momento pulsando el botón "+", y también eliminar las que se deseen.

En esta plantilla, hay unos Blueprints ya creados. El Blueprint Pick Up Cube, es el que permite coger y dejar los cubos. Para acceder a él, habrá que seguir la ruta que se muestra en la ilustración 36.

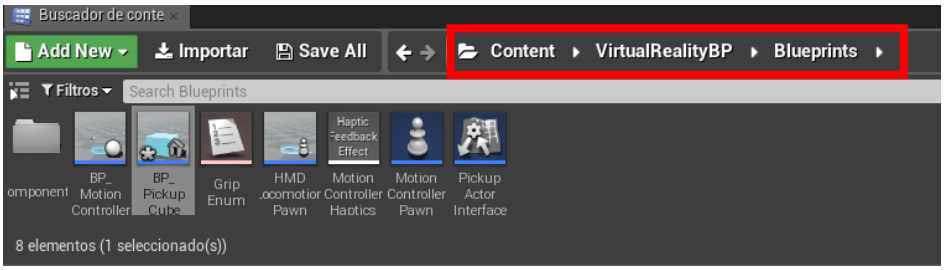


Ilustración 47 - Localización BP Pick Up Cube

Dentro de este Blueprint, se encuentra la programación para poder coger y poder soltar un cubo cualquiera del entorno. Para ambas acciones, se han creados dos eventos personalizados: *Evento Pickup*, para coger, y *Evento Drop*, para soltar. Con la función Ajustar simular físicos lo que se hace es que se pueda coger cualquier componente de malla estática que esté simulando físicos. Después, se colocará la función *AttachToComponent* o *DetachFromActor*, que lo que ordenan es que el cubo se una a la mano del actor, y que se suelte del mismo, respectivamente.

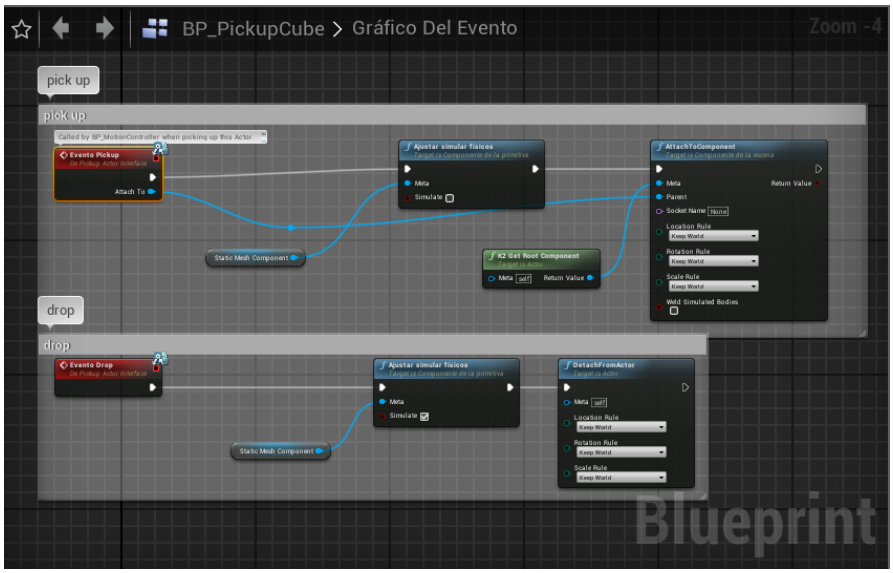


Ilustración 48 - Blueprint Pick Up Cube

Dentro del Blueprint *MotionController*, se puede encontrar programación en la que no se va a entrar en detalles, puesto que es un nivel bastante avanzado. Acciones como el poder teletransportarse, la vibración de los mandos cuando se detecta un objeto, la vibración cuando, al tener cogido un objeto impacta con otro que está simulando físicos, o la animación de las manos pueden encontrarse en este Blueprint programadas (ilustración 43).

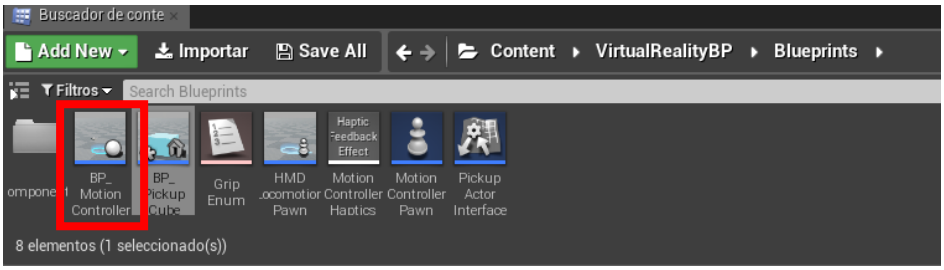


Ilustración 49 - Blueprint MotionController.

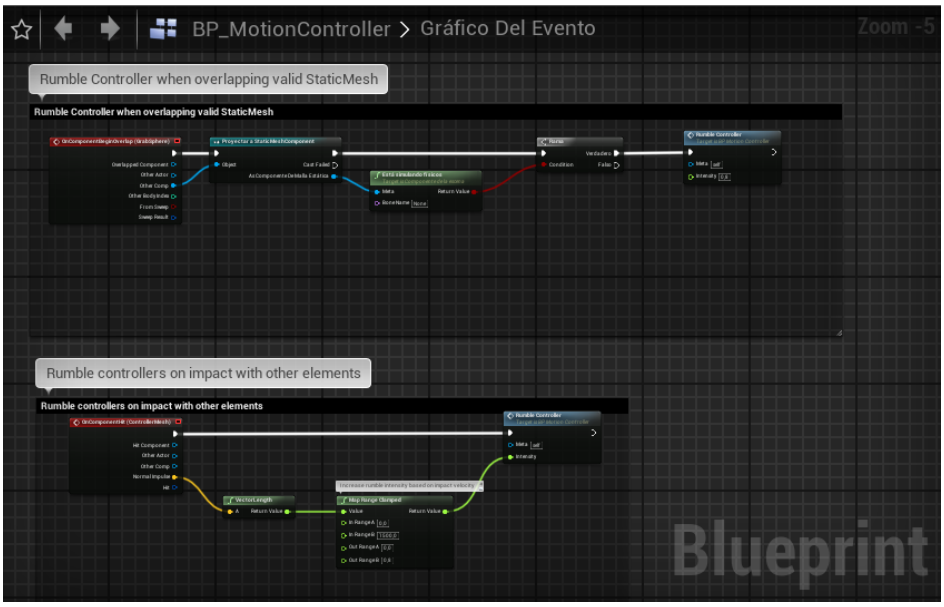


Ilustración 50 - Ejemplos de programación en Blueprint MotionController.

Una vez que se tienen los conjuntos importados, hay que tener en cuenta que se importan ensamblados por lo que habrá que separarlos todos para así poder manipular los conjuntos individualmente.

Ilustración 51 - Piezas y conjuntos separados.

Cuando son importados, por defecto UE4 no les pone **colisión**, por lo que los objetos no pueden simular físicos. Es por esto por lo que habrá que seleccionar, uno por uno, todos los conjuntos y piezas, entrar en su malla estática y otorgarles una colisión. Para hacerlo lo más realista posible, se elige una colisión personalizada mediante *Auto Convex Collision*, en lugar de una colisión cúbica, o esférica.

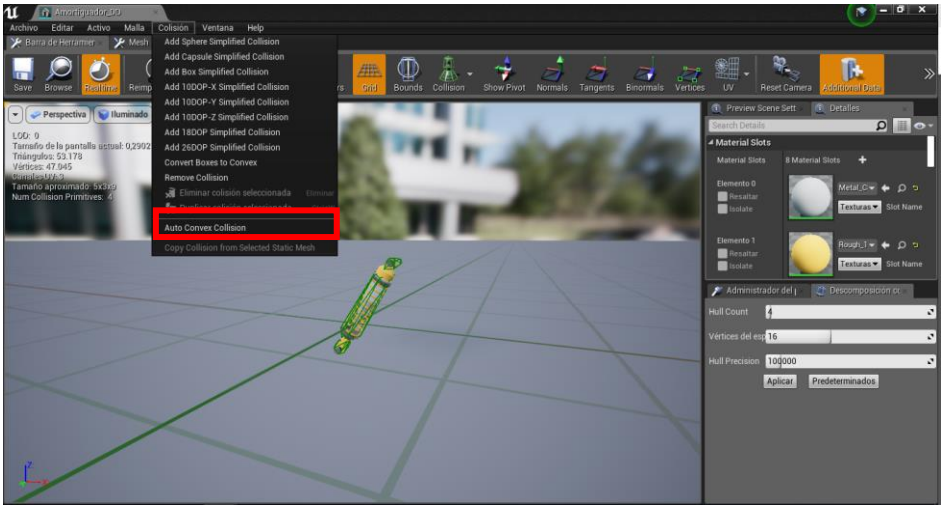


Ilustración 52 - Colisión en malla estática. Amortiguador.

Una vez que la malla estática tiene una colisión, el objeto ya puede simular físicos. Para activar esta opción, habrá que acudir al menú *Physics*, y marcar la casilla *Simular Físicos*.

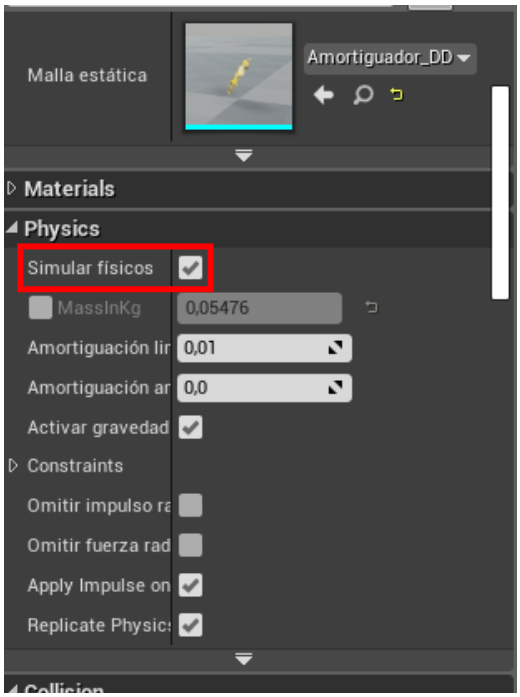


Ilustración 53 - Simular físicos.

Se repetirá este proceso en todos y cada uno de los objetos y conjuntos que forman el despiece del vehículo. Aprovechando que se ha mencionado el menú *Physics*, otra de las casillas es Activar Gravedad. UE4 es una herramienta muy potente y es capaz de simular de una manera muy realista el comportamiento de los objetos. De esta forma, si esta casilla es activada, cuando el juego se inicie, todos los objetos que tengan esta casilla marcada se verán sometidos a la fuerza de la gravedad. Así, si un objeto es colocado en el aire, al comenzar el juego precipitará hasta que choque con otro objeto, o con el mismo suelo. También se puede introducir un parámetro que le otorgue una masa personalizada a cada objeto mediante *MassInKg*.

El objetivo es lograr tener los conjuntos por separado, poder cogerlos, y colocarlos en la posición adecuada en el orden correcto para poder obtener el vehículo montado. Es por esto por lo que se importa una copia más del vehículo, que se denominará *Referencia*. La única pieza visible desde el inicio del vehículo referencia será el chasis, estando el resto ocultas, y a partir de él se irán colocando los conjuntos en la posición adecuada. Las piezas estarán colocadas al lado del chasis de referencia, de modo que si, por ejemplo, el usuario coge el conjunto de diferenciales y los acerca a la posición correcta, queden colocados de manera definitiva.

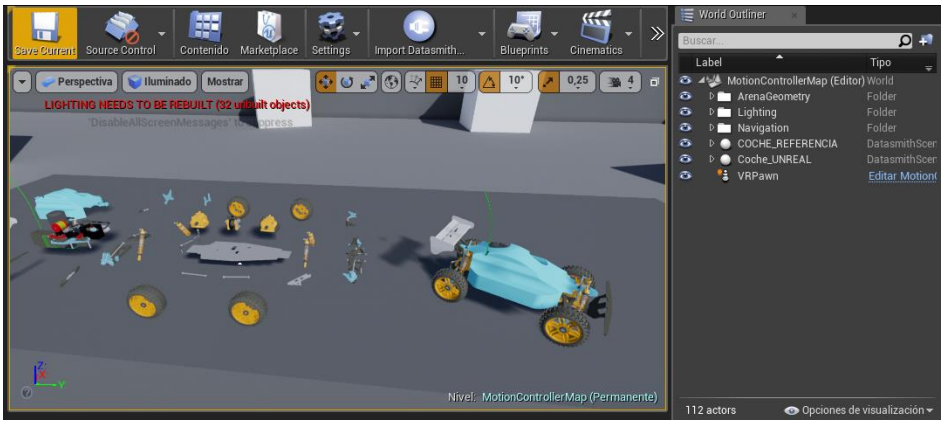


Ilustración 54 - Coche original y referencia importados y visibles.

Para lograr esto, se utiliza una programación basada en la visibilidad de las piezas originales y las de referencia. Cuando la pieza original que es cogida por el usuario tenga una localización y un ángulo aproximados al que tiene la pieza referencia, automáticamente la pieza referencia que estaba oculta se pondrá visible, y la pieza original que estaba visible cambiará a oculta.

Esta programación se realizará en el Blueprint de nivel como se muestra a continuación.

Posición: mediante *GetActorLocation* se obtiene la posición de la pieza original y la de referencia. Éstas se restan, dando como resultado un vector del que se obtendrá el valor mediante *VectorLength*. Este valor será comparado con un valor numérico.

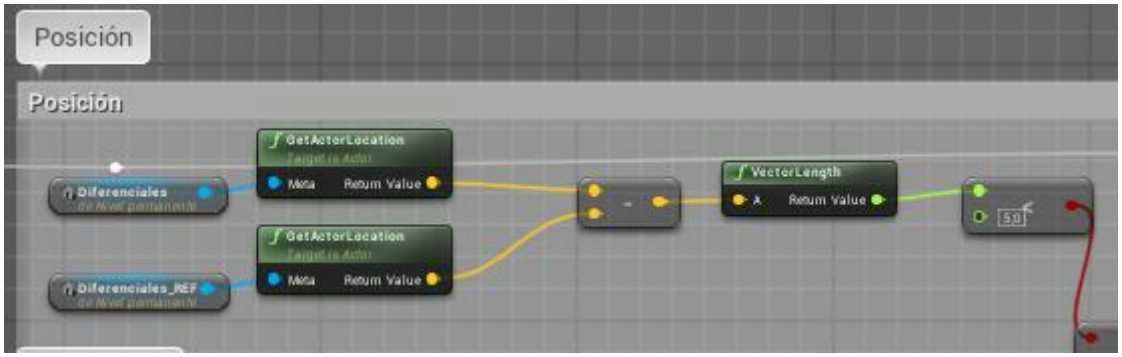


Ilustración 55 - Comparación posiciones.

Orientación: con la comparación de las orientaciones se podrá controlar que la pieza se coloque cuando el usuario hace el giro correcto. Para ello, se obtiene la rotación de las piezas con *GetActorRotation* y el vector de avance del actor en el espacio con

GetForwardVector. De nuevo se restan estos vectores, se obtiene su longitud y se compara con un valor numérico.

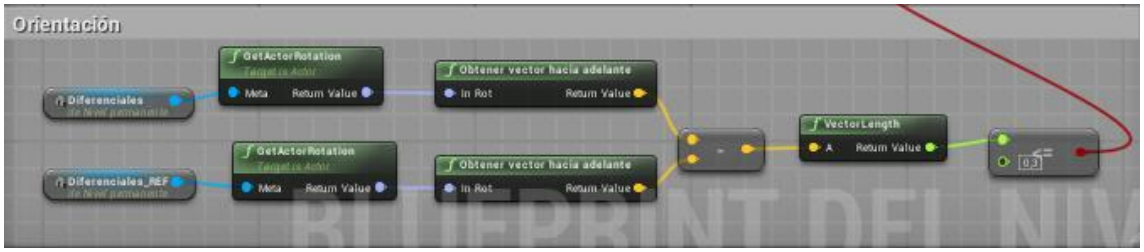


Ilustración 56 - Comparación de orientaciones

Estas dos comparaciones arrojan una variable de tipo booleana (en color rojo). Por esto, resulta sencillo utilizar una puerta lógica AND, que lo que hace es una multiplicar las variables.

Var. 1	Var. 2	AND
0	0	0
0	1	0
1	0	0
1	1	1

Tabla 1 - Puerta lógica AND.

La puerta lógica AND lo que hace es que sólo en el caso de que las dos variables sean verdad (valor 1) tendrá como salida un valor 1, por lo que se tienen que cumplir ambas condiciones.

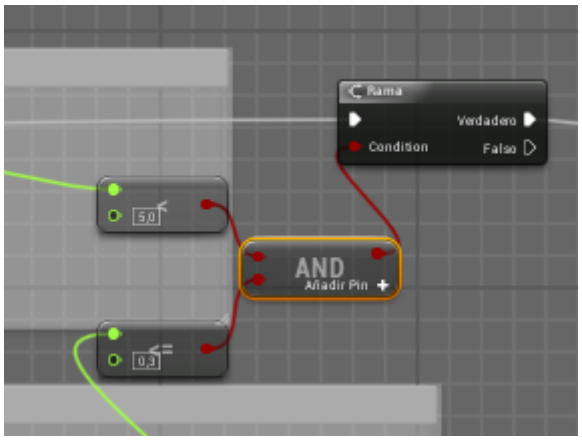


Ilustración 57 - Puerta AND.

Esta variable resultado se lleva a una *Rama (Branch).* Como se observa tiene dos nodos de salida, verdadero y falso. En este caso sólo se utiliza el nodo verdadero, y de ahí se ajustan las visibilidades.

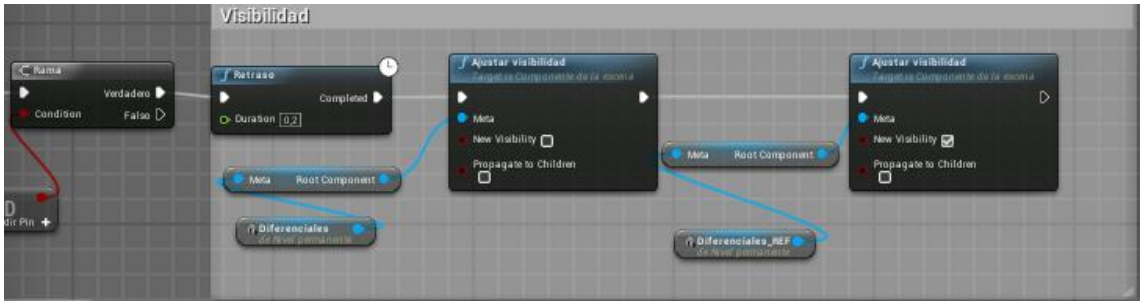


Ilustración 58 - Visibilidades de las piezas original y referencia.

Mediante el nodo *SetVisibility* se ajusta la visibilidad. Como se había comentado antes, la pieza original, la que se coge, estaba visible y tras cumplir las condiciones se vuelve oculta; mientras que la pieza referencia que estaba oculta, se vuelve visible, por lo que se marca el tick en la casilla. Además, se introduce ante de esto un *Delay* o Retraso de 0,2 segundos.

Para establecer que las piezas referencia estén ocultas, habrá que desmarcar la casilla *Visible* en el menú *Rendering*, como se muestra en la siguiente ilustración. Esto hubo que hacerlo con todas las piezas y conjuntos pertenecientes a *Referencia*.

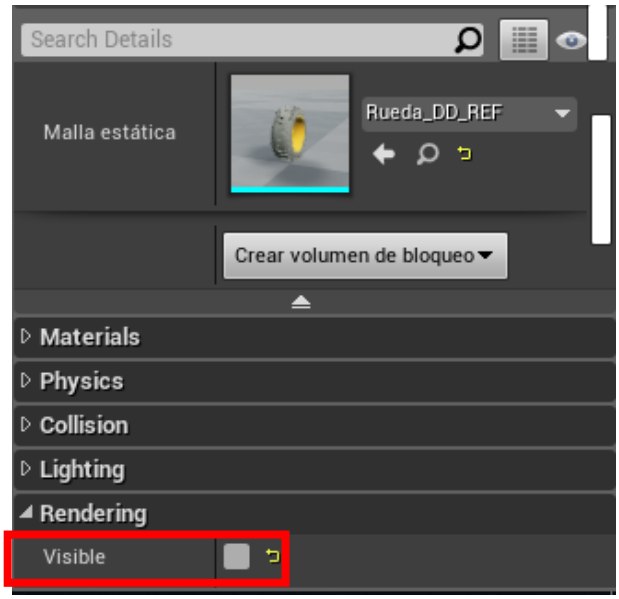


Ilustración 59 - Visibilidad de las piezas.

Cabe mencionar que los nodos han de ser conectados a un *EventTick* (Evento Marcar) para que puedan funcionar correctamente.

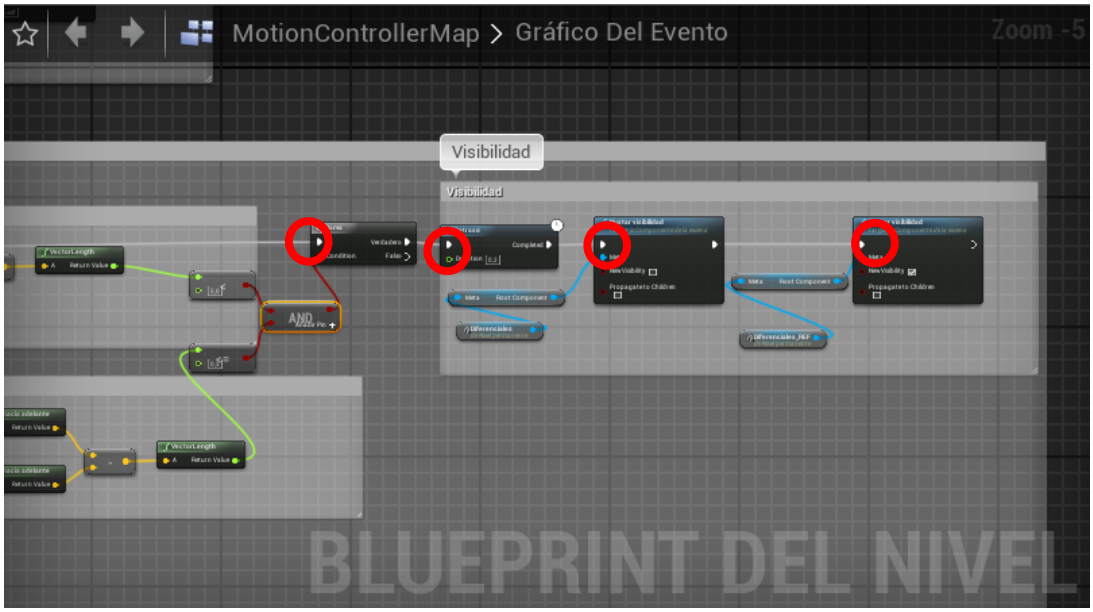


Ilustración 60 - Nodos que se unen en EventTick.

Este modo de programación será repetido para todos y cada uno de los elementos que quieran cogerse, colocarse y formar parte del vehículo montado.

Para hacer más fácil el montaje del coche, y evitar que el jugador se esté desplazando, se diseña una plataforma en la que se encontrará inicialmente el chasis a partir del que se montarán todas las piezas. Esta plataforma será cilíndrica y giratoria.

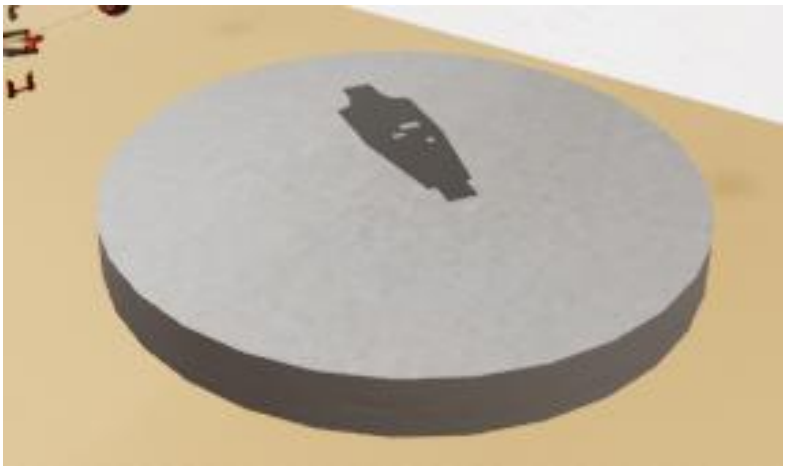


Ilustración 61 - Plataforma cilíndrica.

Para que la plataforma gire, y a su vez lo haga el coche, lo que se hace es acoplar el chasis al cilindro de la plataforma.

Ilustración 62 - Chasis unido a plataforma.

La programación del giro se realizará en el Blueprint del nivel. Para ello, se crea un evento personalizado llamado *Rotate 90º*. Se obtiene el actor de la plataforma (*Cylinder*) y su rotación mediante *GetActorRotation*. La variable resultado se expande para obtener los diferentes valores de rotación en X, Y y Z, y en el eje Z se sumará 90. Con *SetActorRotation*, se establece la nueva ubicación.

Esta rotación será controlada con el Grip del mando de la mano derecha, por lo que este nodo llamará al evento *Rotate 90º*.

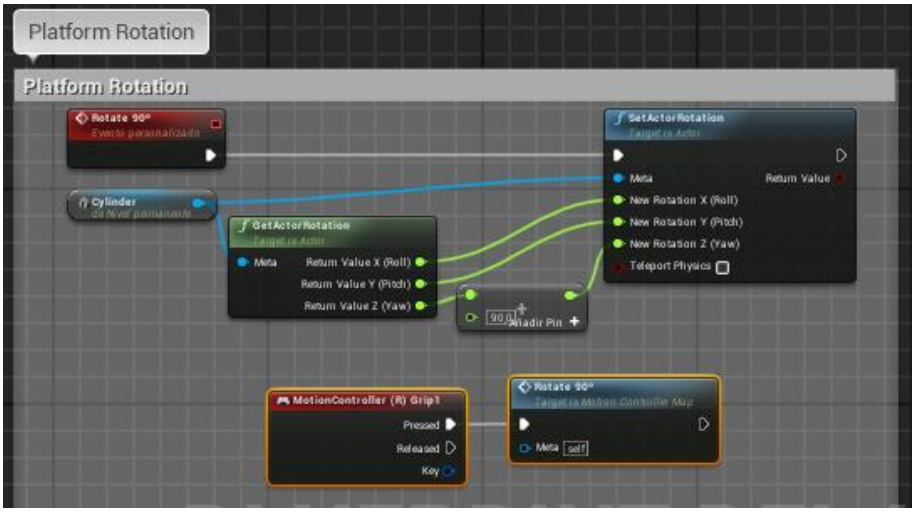


Ilustración 63 - Programación giro de la plataforma.

Por último, y utilizando el programa Adobe Photoshop CC 2018 para crear la imagen, se introduce un menú de ayuda en el que se explica el montaje paso por paso del vehículo.

Este menú constará de varias imágenes numeradas y de un texto explicativo en el que se proporcionan las normas para el correcto montaje. Para que aparezca este menú, habrá que mantener pulsado el botón *Grip* del mando de la mano izquierda. Cuando se deje de presionar, el menú desaparecerá.

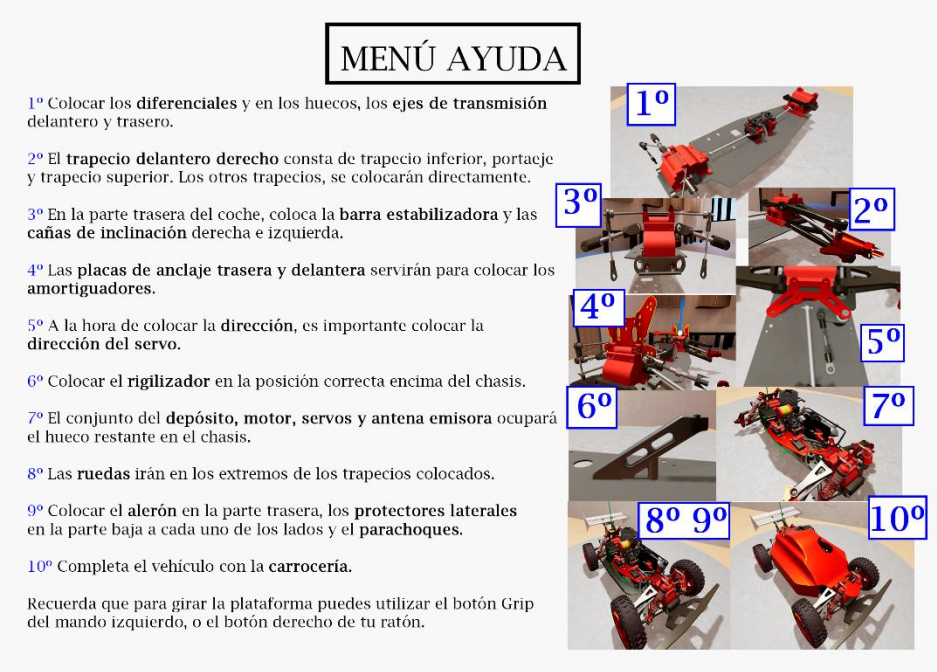


Ilustración 64 - Menú Ayuda.

Para ello, lo primero es crear en el Content Browser – User Interface un Blueprint Widget, al que se agregará una imagen. En este caso, la imagen es la que se muestra en la Ilustración 64, que tendrá que ser importada a UE4. Después de esto, habrá que realizar una asignación de acción que sea *Ayuda*, y definir qué botón se empleará para utilizarla.

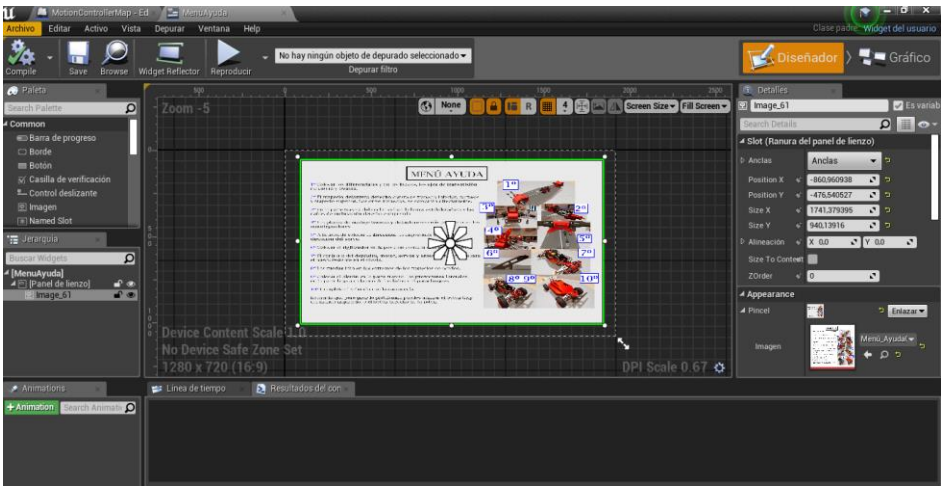


Ilustración 65 - Blueprint Widget.

Ya en el Blueprint del nivel se conectará la acción Ayuda cuando se presione el botón con la llamada al widget creado y obteniendo el controlador del reproductor. Mediante

AddtoPlayerScreen se podrá obtener en pantalla la imagen. Para que la imagen desaparezca al dejar de pulsar, se emplea *RemoveFromParent*.

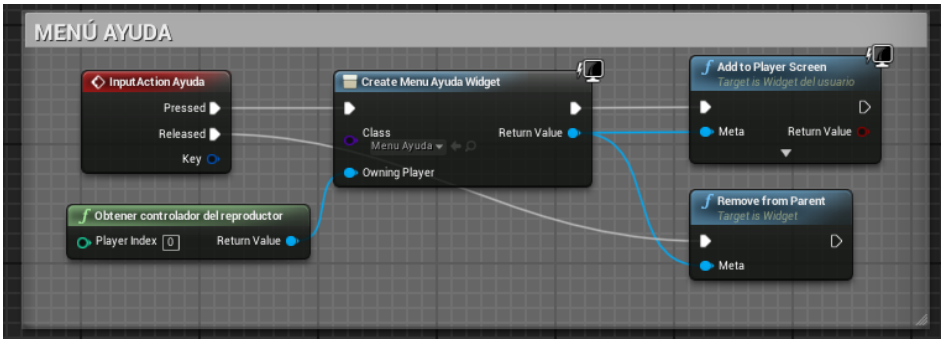


Ilustración 66 - Programación Menú Ayuda.

Así, ya se habría obtenido el montaje del vehículo en realidad virtual inmersiva.

b. Montaje en realidad virtual no inmersiva

Para realizar el montaje en RV no inmersiva, se partirá de la plantilla de UE4 que frece un juego en primera persona *First Person*.

Ilustración 67 - Plantilla First Person.

En ella, el jugador dispone de una pistola que lanza proyectiles con los que derribar los cubos que se encuentran en el entorno. Se utiliza esta plantilla debido a que se considera el mejor modo de juego para poder interactuar con las piezas y realizar el montaje del vehículo.

Una vez abierta la plantilla, se realizarán una serie de modificaciones. La primera consiste en eliminar la pistola y las manos del jugador de la pantalla. Para ello, habrá que acceder

al Blueprint llamado *FirstPersonCharacter*, siguiendo la ruta que se indica en la siguiente ilustración.

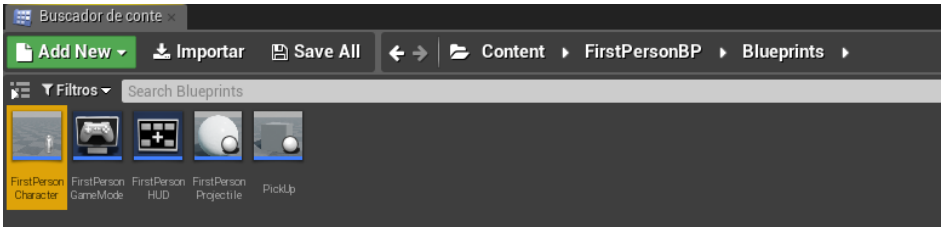


Ilustración 68 - Ruta BP FirstPersonCharacter.

En la ventana gráfica del Blueprint, y tal como se ve en la imagen de la izquierda, la pistola y los brazos se ven más allá de la cámara. Lo que se hace es seleccionar estas mallas, y desplazarlas hacia atrás.

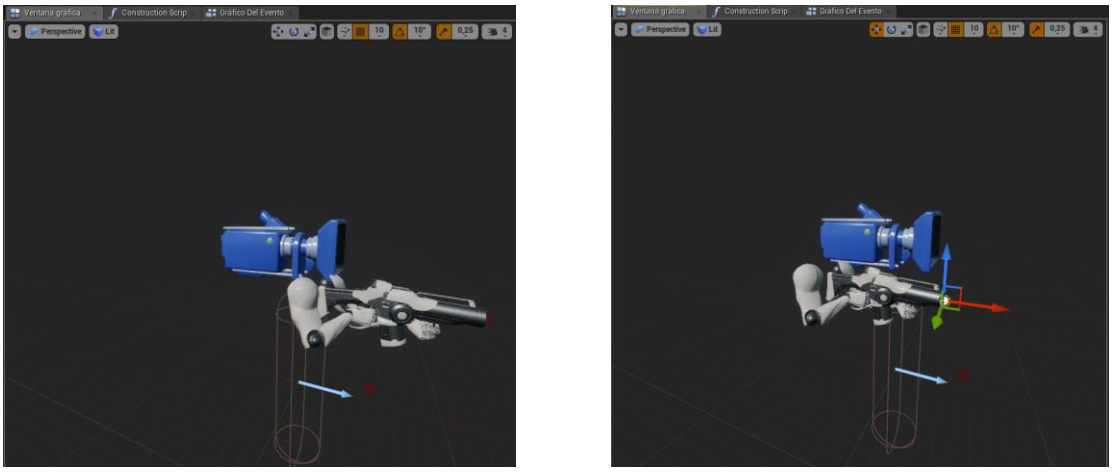


Ilustración 69 - Comparativa arma y manos.

Además, se desmarcará la casilla de visibilidad, por lo que estas mallas quedarán totalmente ocultas en la pantalla del juego.

En el gráfico del evento, la plantilla cuenta con diversas programaciones que controlarán el salto del jugador, su movimiento si se conecta un sistema HMD, o el lanzamiento de proyectiles. Es en este Blueprint donde se realiza la programación para coger y soltar objetos.

Para la acción de coger objetos, se crea un evento personalizado llamado *evento Pick up*. Utilizando la cámara proporcionada por la plantilla, se obtendrá su localización en el

mundo con *GetWorldLocation* y el vector de avance con *GetForwardVector*. Mediante la función *LineTraceByChannel* se va a crear una línea cuya función es hacer un seguimiento de colisión a lo largo de esta línea dada y devolver el primer golpe de bloqueo encontrado. Es decir, en este proyecto se va a emplear como detector del objeto que se va a coger. La función tendrá como ubicación inicial el valor de *GetWorldLocation*, y como valor final la suma de valores de *GetWorldLocation* y *GetForwardVector* (previamente multiplicado por un valor que simboliza el alcance o la distancia que hay hasta el objeto para que pueda ser detectado).

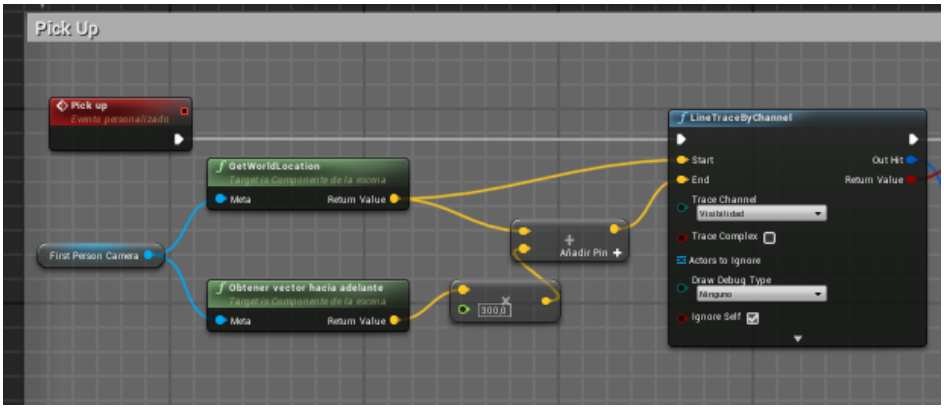


Ilustración 70 - Programación Pick Up parte 1.

La función cuenta con otros nodos tal y como se puede ver, que permiten que se pueda ver la línea trazada, ignorar ciertos actores que se coloquen en una matriz, o ignorar a uno mismo. Estos nodos no tendrán relevancia en el proyecto y no serán utilizados.

A continuación, *LineTraceByChannel* tiene tres nodos de salida: el blanco conecta las diferentes funciones; el rojo ofrece un resultado booleano que indicará cierto si hubo golpe con un objeto y falso si no lo hubo; el azul da el resultado de las propiedades de rastreo, por lo que mediante el nodo *HitResult* extrae los datos. Por ello, el valor booleano será llevado a una rama (*Branch*), y si es verdadero que hubo golpe con un objeto la programación continuará hacia otra rama. De la función *HitResult* se utilizará el nodo *HitComponent* que se refiere al componente golpeado por la línea. Con el nodo *IsSimulatingPhysics* se comprobará si el objeto golpeado tiene activada las colisiones y está simulando físicos. En caso de no ser así el objeto no podrá ser cogido.

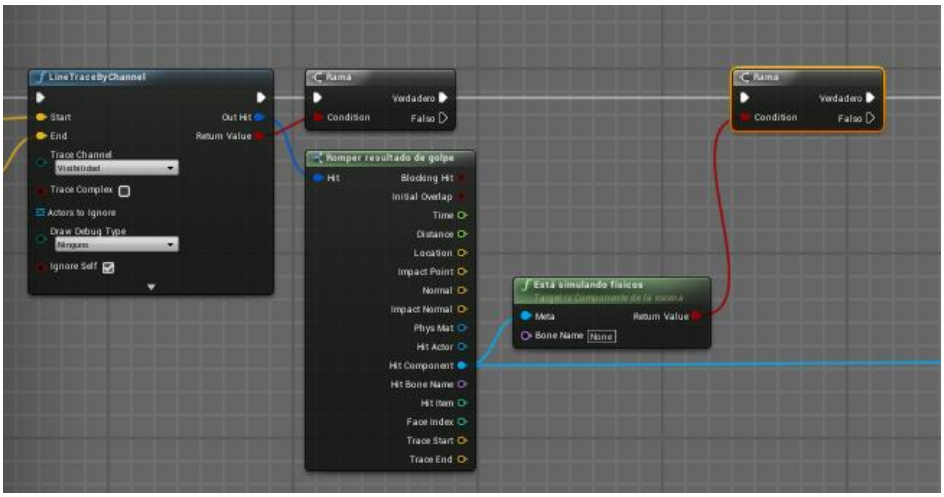


Ilustración 71 - Programación Pick Up parte 2.

Tras esto, se debe crear un componente controlador de físicos *Physics Handle*, en *añadir componente.* Con la función *Grab Component at Location* se agarra el componente especificado en una ubicación determinada sin restringir la rotación. Esta localización será el valor resultado de obtener la localización del componente golpeado. Por último, se crea una función de tipo booleano (*Is Holding Object*) que indica si se está sujetando o no el objeto. Como se está programando para coger un objeto, se marcará la casilla de esta función booleana.

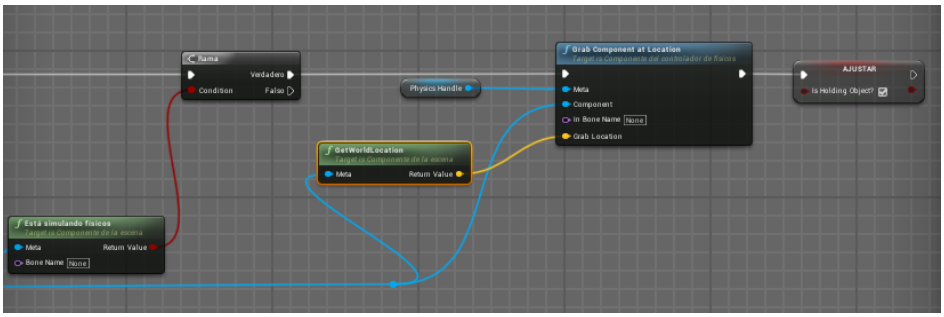


Ilustración 72 - Programación Pick Up parte 3.

El evento personalizado *Drop* controlará la acción para soltar un objeto. Para ello se empleará la opción de liberar componente (*ReleaseComponent*), controlado por el controlador de físicos, y unido a la función booleana *Is Holding Object*, que en este caso no tendrá la casilla marcada.

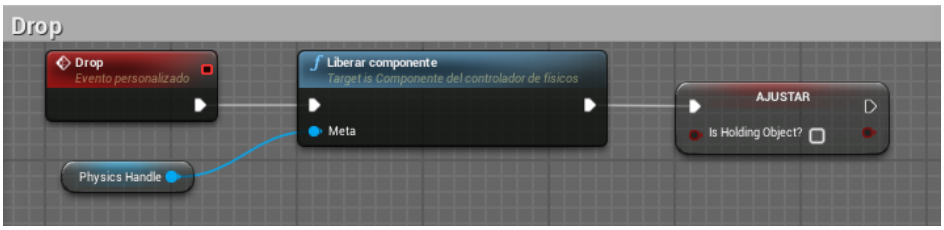


Ilustración 73 - Programación Drop.

El evento marcar, que comprueba en cada fotograma del juego, ha de ser unido a la función *Set Target Location* para establecer la ubicación de destino del componente controlador de físicos (*PhysicHandle*). La nueva ubicación será la establecida por un nuevo componente de escena denominado *Held Object Location.* Este componente de escena habrá que acoplarlo a la cámara de primera persona y refleja la ubicación a la que será cogido un objeto y por tanto dónde aparecerá en la pantalla.

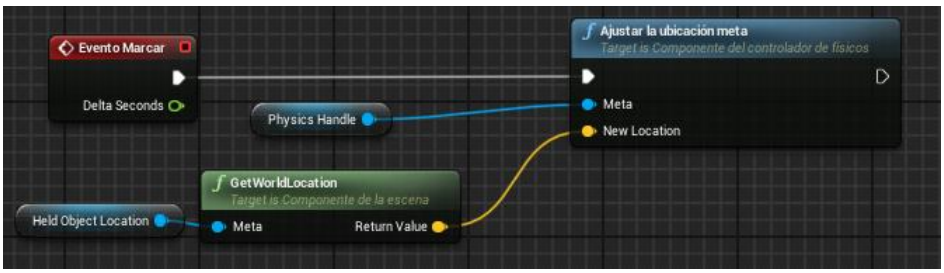


Ilustración 74 - Programación Event Tick

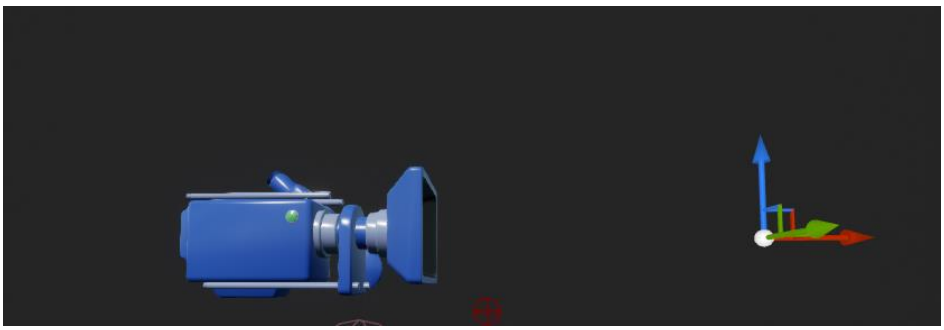


Ilustración 75 - Ubicación de Held Object Location.

Por último, la acción de coger y soltar objetos se realizará presionando el botón izquierdo del ratón. Relacionando este botón y la función booleana Is Holding Object en una rama se podrán simular estas acciones de forma que, si Is Holding Object es falso, o su casilla no está marcada, se procede a coger el objeto (*Pick Up*); y si es verdadero, que significa que ya hay un objeto cogido, habrá que soltarlo al presionar el botón (*Drop*).

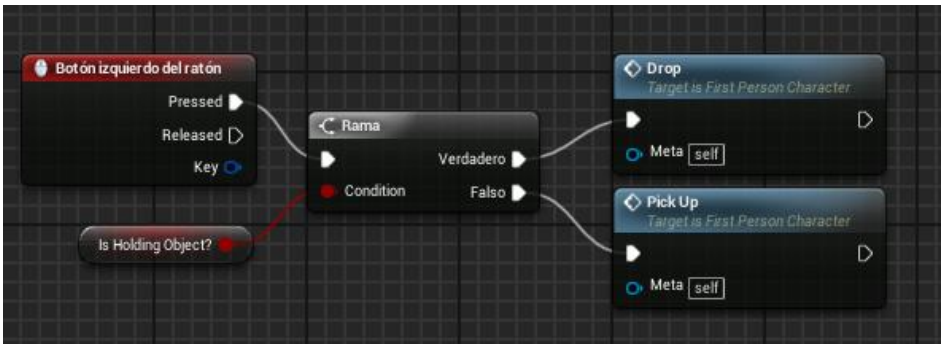


Ilustración 76 - Control de Pick Up y Drop.

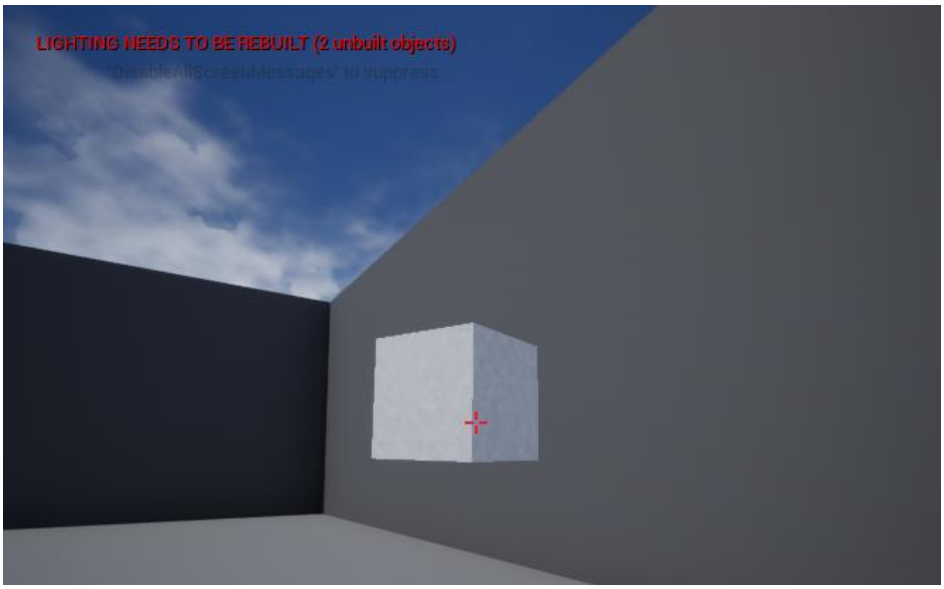


Ilustración 77 - Cubo cogido por personaje.

Se procede a la importación del vehículo por piezas y el vehículo que servirá de referencia. La programación para el montaje será la misma que en realidad virtual inmersiva. Cabe mencionar que habrá que repetir todo el proceso: importación, colisiones, edición de materiales y programación, aunque en la memoria no se vuelva a describir.

Debido a las dimensiones de algunas piezas, y de que esta aplicación es más inexacta que la de realidad virtual inmersiva, se unen algunos componentes en 3DS Max para obtener conjuntos de mayor tamaño que se puedan manipular de una mejor forma. Estas piezas que se unen serán los diferenciales, los ejes de transmisión delantero y trasero, el rigidizador de chasis, la dirección, el tirante servo de la dirección, la barra estabilizadora y las cañas de inclinación derecha e izquierda. Se agrupan los elementos, se exportan en

Datasmith y se importa el archivo a UE4. Es importante importar este grupo como malla estática, al igual que el resto de las piezas. Se importará dos veces, debido a que si se importa una vez y se duplica las características que se le apliquen a la malla serán las mismas para el duplicado. Es decir, para las piezas, la malla debe tener colisiones y simular físicos, mientras que para la referencia no puede tener colisiones ya que su visibilidad va a estar oculta y no tendría sentido que al colocar la pieza chocase con algo que no se puede ver.

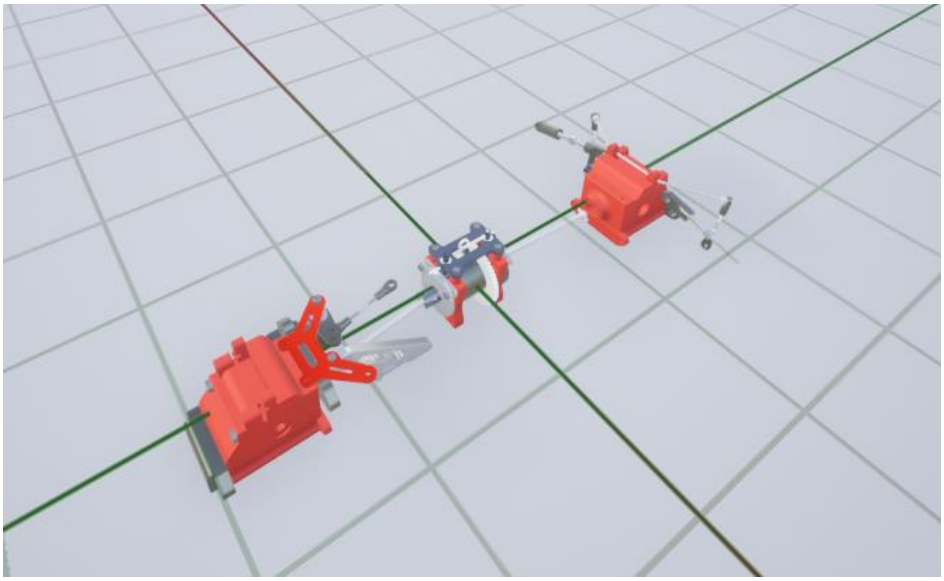


Ilustración 78 - Conjunto creado.

En cuanto a los controles, la aplicación se va a desarrollar con teclado y ratón en un ordenador. Además de los controles ya mencionados, en este proyecto se encontrarán los siguientes:

- **Movimiento del jugador**: el movimiento hacia adelante y hacia atrás se realizará con las teclas *subir* y *W*, y con *bajar* y *S*, respectivamente. El movimiento a la derecha y a la izquierda será con *D* y con *A*, respectivamente.

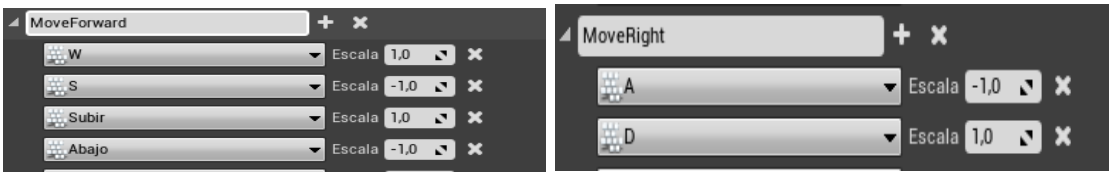


Ilustración 79 - Movimiento jugador

- **Salto**: el salto se realizará con la barra espaciadora.
- **Mirar hacia los lados**: esta acción se realizará con los botones *izquierda* o *derecha,* y también con el mismo movimiento en el *eje X del ratón.*
- **Mirar hacia arriba y abajo**: se llevará a cabo con el movimiento en el eje Y del ratón.
- **Giro de la plataforma cilíndrica**: se utilizará el botón derecho del ratón.
- **Menú Ayuda:** mediante la tecla H del teclado.

Así, se tendría la aplicación del montaje del coche en realidad virtual no inmersiva realizada. No obstante, el entorno se modelará más adelante.

c. Materiales para las piezas

Una vez que se ha realizado el montaje, se procede a cambiar los materiales de todas y cada una de las piezas que forman el coche. El objetivo es dotar de realismo los materiales, ya que los iniciales carecen de esta característica.

Ya que el vehículo se forma con muchos elementos metálicos, se cambiarán las características de los materiales para que parezcan metalizados. Para ello, por ejemplo con el material que forma la carrocería y otras piezas, se varían los valores de *desigualdad* y *metálico,* estableciéndolos en 0.3 y 0.95, respectivamente. Se va a jugar con estos valores para hacer que todos los materiales sean más o menos metálicos dependiendo del elemento en el que se van a aplicar.

Estos materiales no han sido creados, si no que han sido modificados algunos de sus valores, colores, y otras características. Más adelante se explicará cómo crear materiales en Unreal Engine 4. Los materiales tuvieron que ser modificados en todas y cada una de las piezas, tanto en las piezas como en las piezas *referencia,* y también en ambas plantillas para realidad virtual inmersiva y no inmersiva.

Explicar cada material sería muy extenso, por lo que se adjuntan algunas ilustraciones de ciertos materiales como ejemplo:

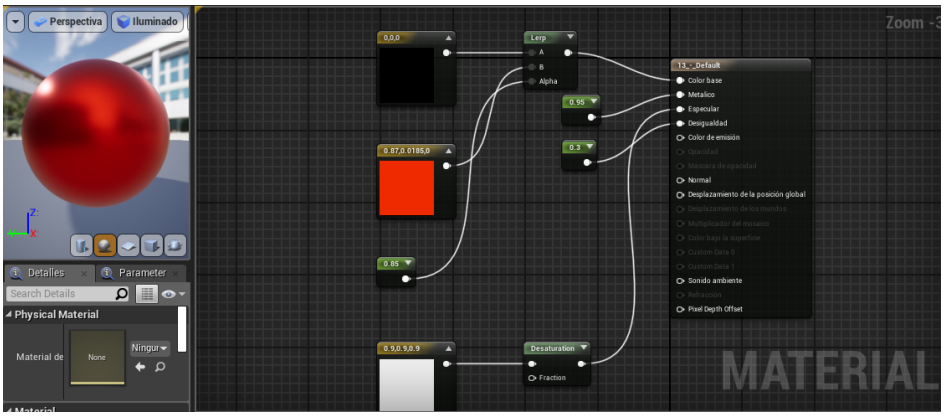


Ilustración 80 - Material Carrocería.

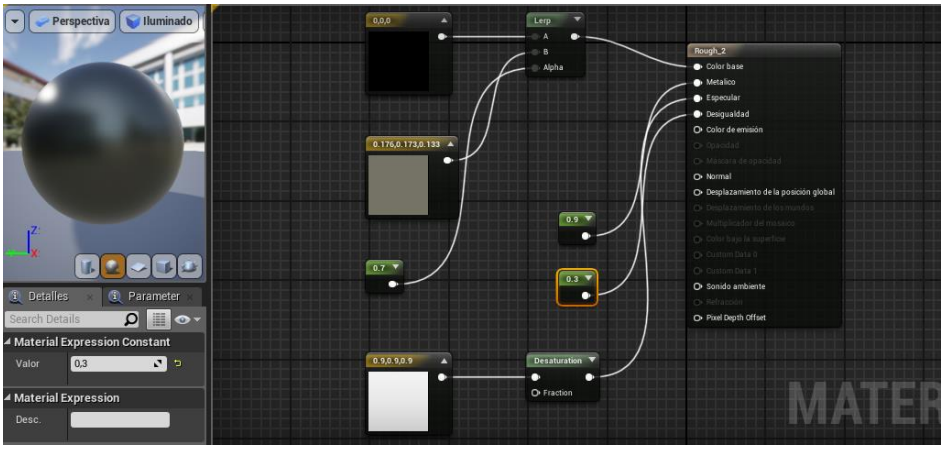


Ilustración 81 - Material Rough_2

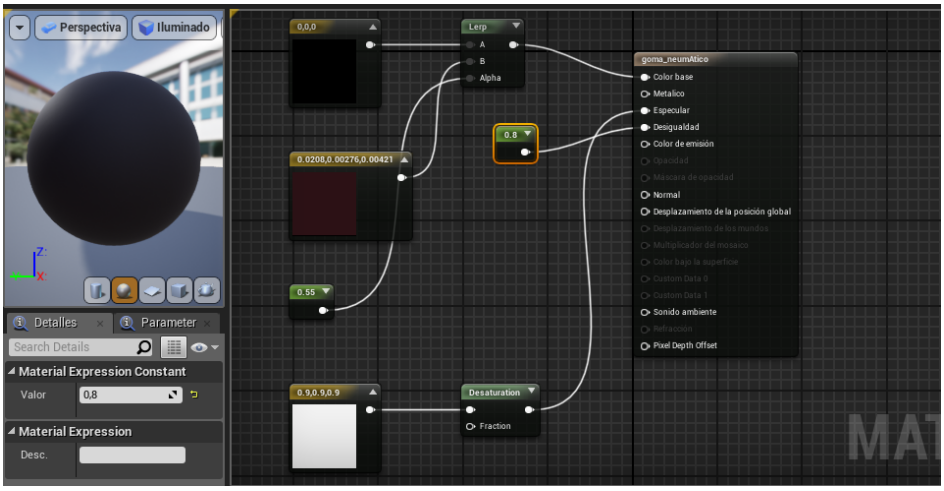


Ilustración 82 - Material Goma_Neumático

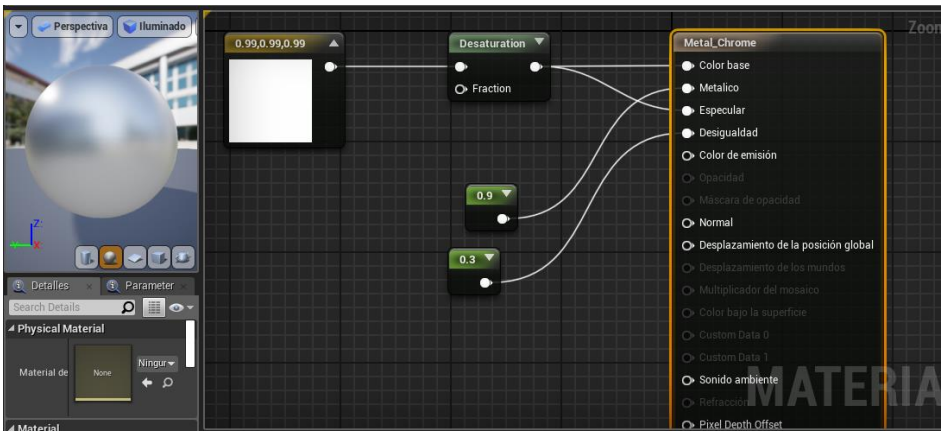


Ilustración 83 - Material Metal_Chrome

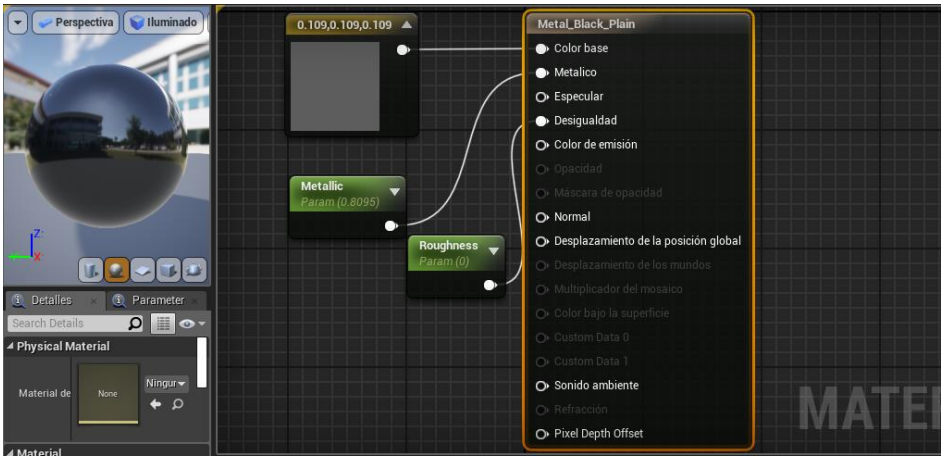


Ilustración 84 - Material MetalBlackPlain

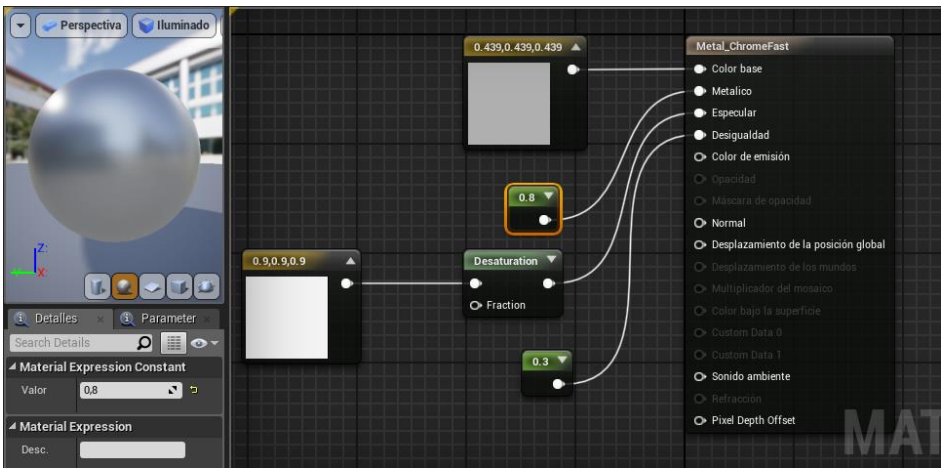


Ilustración 85 - Material Metal ChromeFast

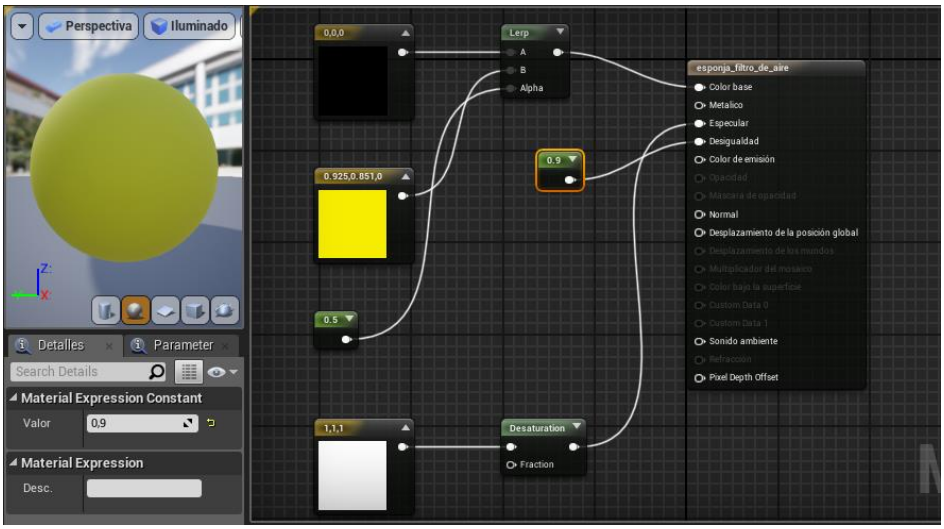


Ilustración 86 - Material Esponja filtro de aire

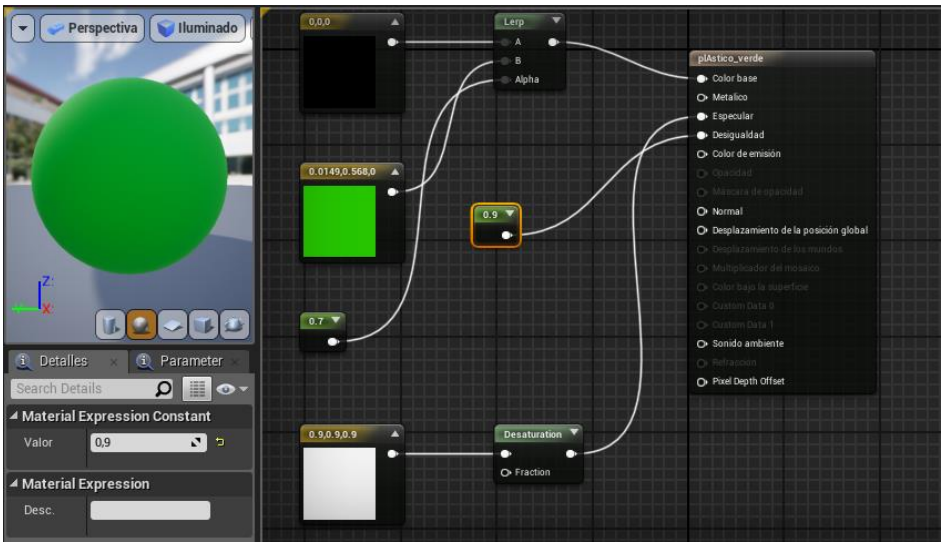


Ilustración 87 - Material Plástico Verde

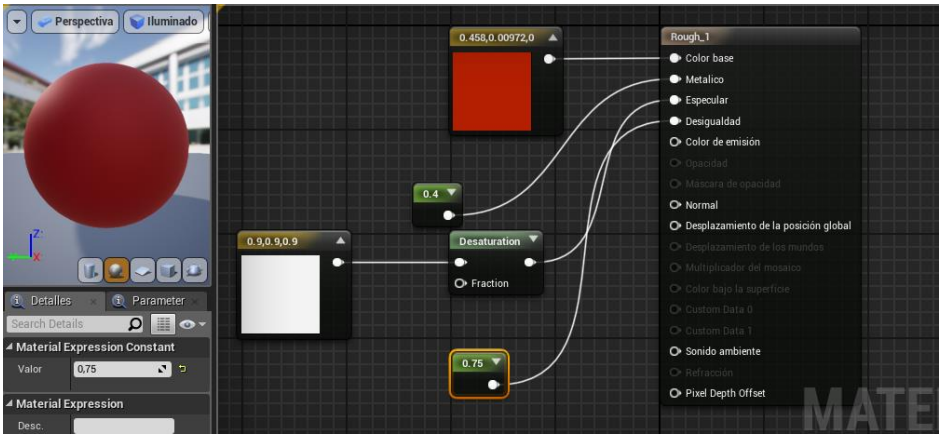


Ilustración 88 - Material Rough1

d. Entorno y materiales empleados

En este punto se comienzan a realizar cambios en el entorno en ambas plantillas. Se puede decir, que el entorno que viene con las plantillas no se aprovecha para nada y que se modela desde cero. Todo el proceso se llevará a cabo en Unreal Engine 4, utilizando 3DS Max para modelar algunos componentes.

El objetivo por realizar es un pequeño taller situado en un garaje de una casa, por lo que se modelan los elementos imaginando la estancia ideal en la que poder montar una maqueta de vehículo por radiocontrol.

Cabe destacar que no fue posible exportar entornos de una plantilla a otra, por lo que el entorno fue creado primero en el archivo de realidad virtual inmersiva, y a continuación se repitió todo el proceso en el de realidad virtual no inmersiva.

Se procede a explicar cada componente a continuación:

- **Paredes, techo y suelo.**

Las cuatro paredes, el techo y el suelo fueron creadas mediante pinceles de caja, dentro de geometría. Se dieron unas dimensiones al azar, creando una habitación en forma prisma rectangular.

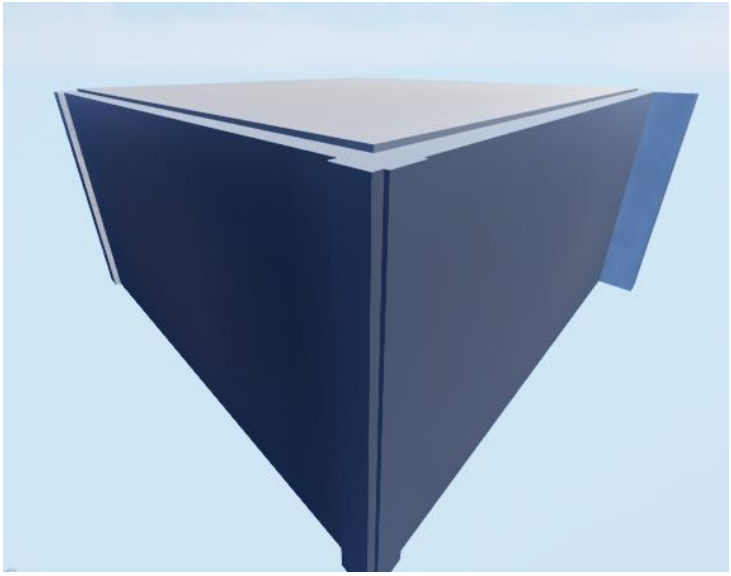


Ilustración 89 - Habitación garaje

En cuanto a los materiales empleados, se descargaron texturas de una página web. Para las paredes y el techo, se opta por un material blanco con rugosidades que simule yeso; para el suelo se aplica una textura de parqué de madera.

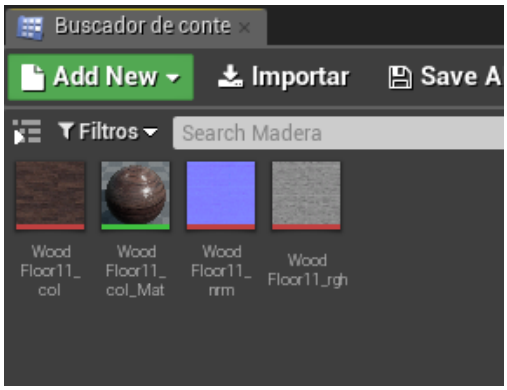


Ilustración 90 - Creación Material.

Para poder crear materiales a partir de estas texturas, primero hay que importar las imágenes que se descargan. A continuación, haciendo clic derecho en la imagen que será el color base se selecciona la opción *Crear Material*. Las otras dos imágenes habrán de ser arrastradas al editor del material, ya que se van a aplicar a diferentes características del material creado.

Una vez que se tienen las tres texturas modificando una característica diferente del material (*Color base*, *Desigualdad* y *Normal*), se les aplica *TextureCoordinate*, que permitirá modificar la repetición del mosaico de la textura.

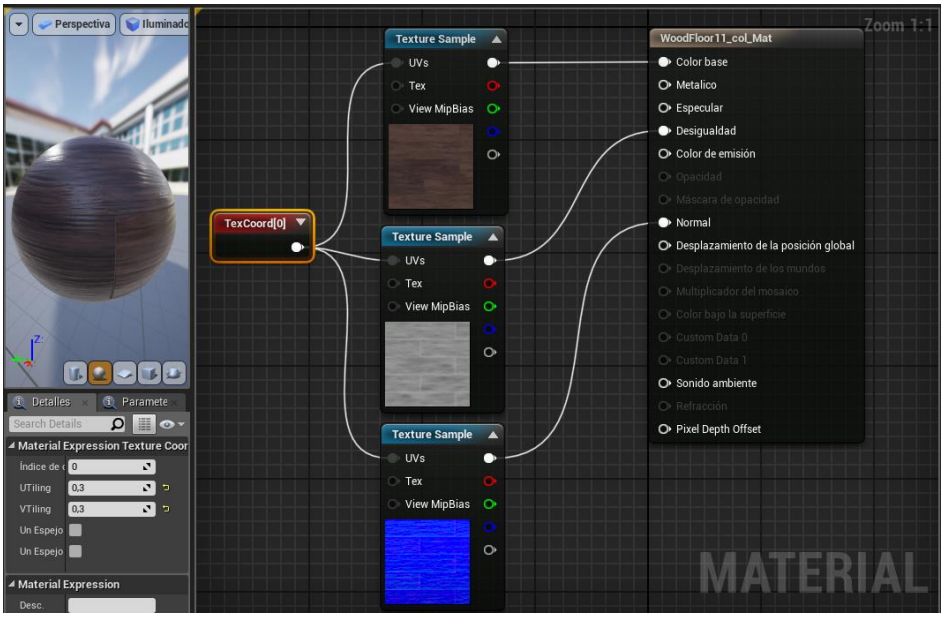


Ilustración 91 - Material Suelo.

Con esto, ya se habría creado el material para el suelo. Se repite el proceso para las paredes y el techo.

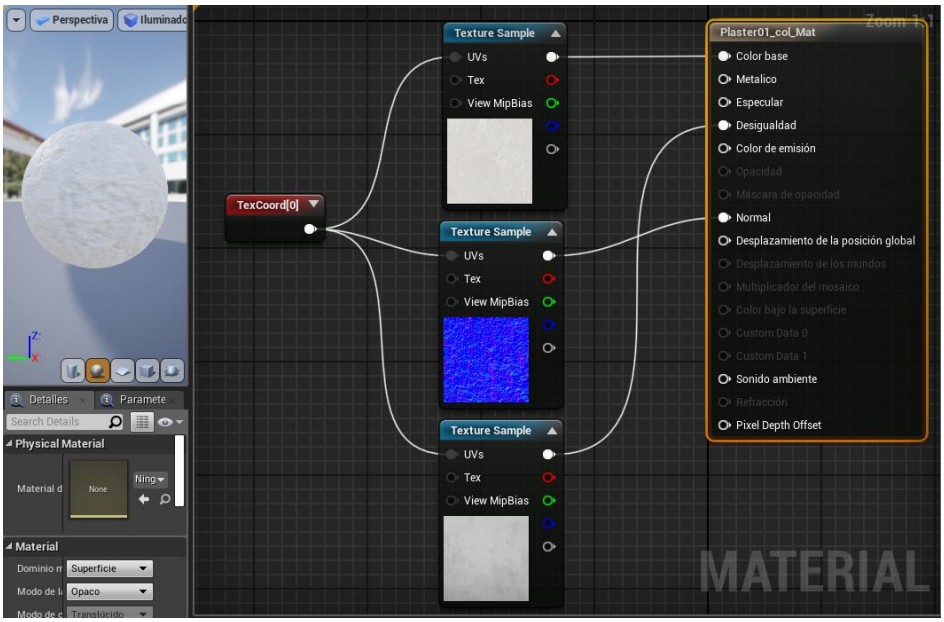


Ilustración 92 - Material Paredes y Techo.

- **Mesas madera**

Para colocar tanto las piezas del coche, como la plataforma en la que se realiza su montaje se emplean mesas de madera. Esta geometría se descarga de una página web, incluyendo sus materiales. Se establecen colisiones para que las piezas no traspasen la mesa, si no que queden colocadas encima de ella.

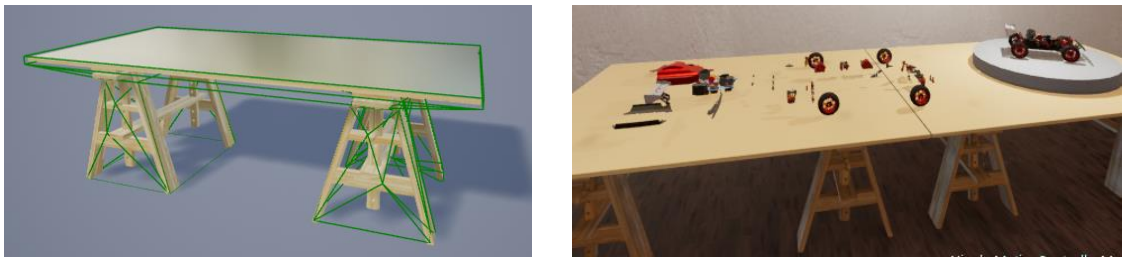


Ilustración 93 - Mesas Madera.

- **Bancos de trabajo**

En 3DS Max se modelan bancos de trabajo de madera y metal para colocar en el garaje. Para ello, aplicando las técnicas aprendidas a lo largo de este proyecto, se obtiene una geometría que es exportada en formato Datasmith para introducirla en UE4, y aplicarle algunos de los materiales de los que ya se dispone. Este es el proceso que se realizará cada vez que se modele una geometría propia en 3DS Max y se importe en UE4.

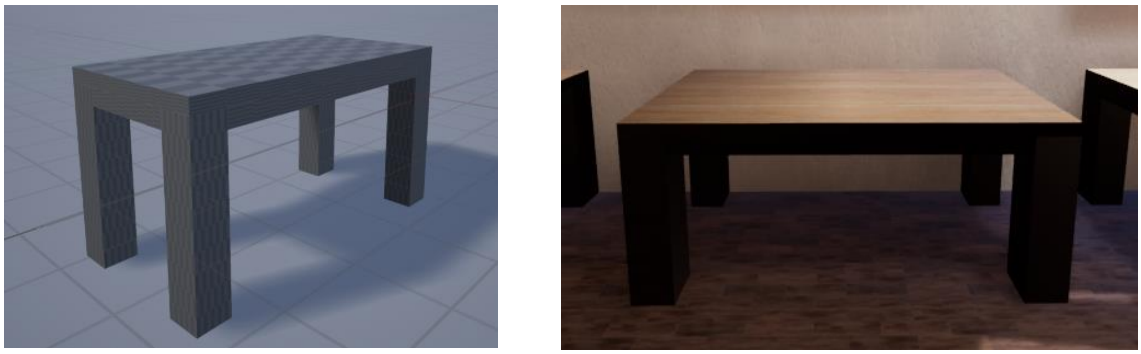


Ilustración 94 - Banco de trabajo.

- **Armarios**

También como geometría propia, es decir, modelada en 3DS Max y no descargada, se modela un armario de madera de pino básico con dos tiradores metálicos.

Ilustración 95 - Armario de Madera.

- **Panel Herramientas y Herramientas**

Para dar un mayor realismo a la habitación, se crea en una de las paredes un panel de herramientas en el que se colocarán algunas herramientas básicas. Algunas de ellas son geometrías propias, otras son descargadas de Internet y adaptadas y modificadas en 3DS Max. Estas herramientas son: martillos, alicates, llaves inglesas, calibres, llaves Allen, limas, destornilladores y un taladro. Los materiales son aplicados en Unreal Engine 4, aprovechando los que ya se han creado o modificado.

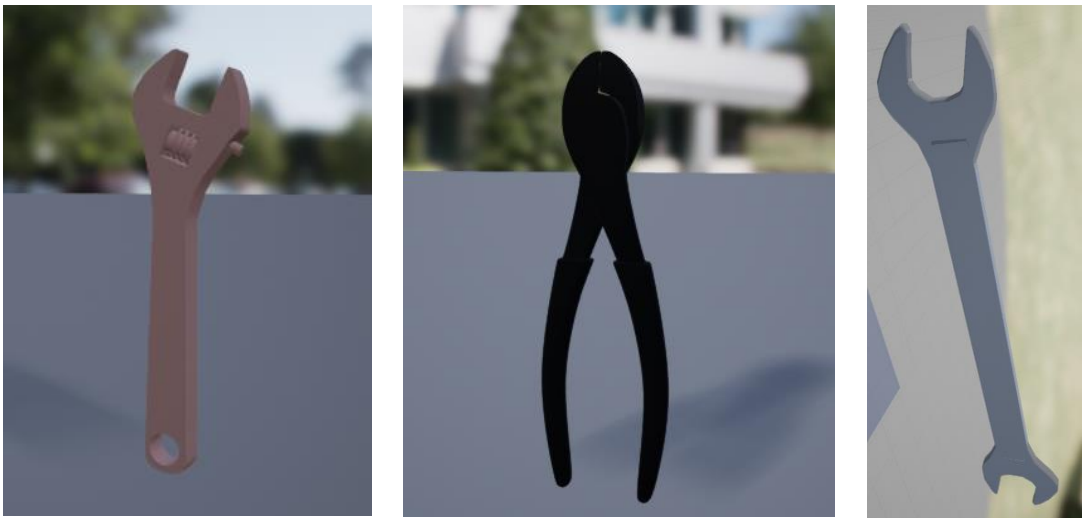


Ilustración 96 - Herramientas.

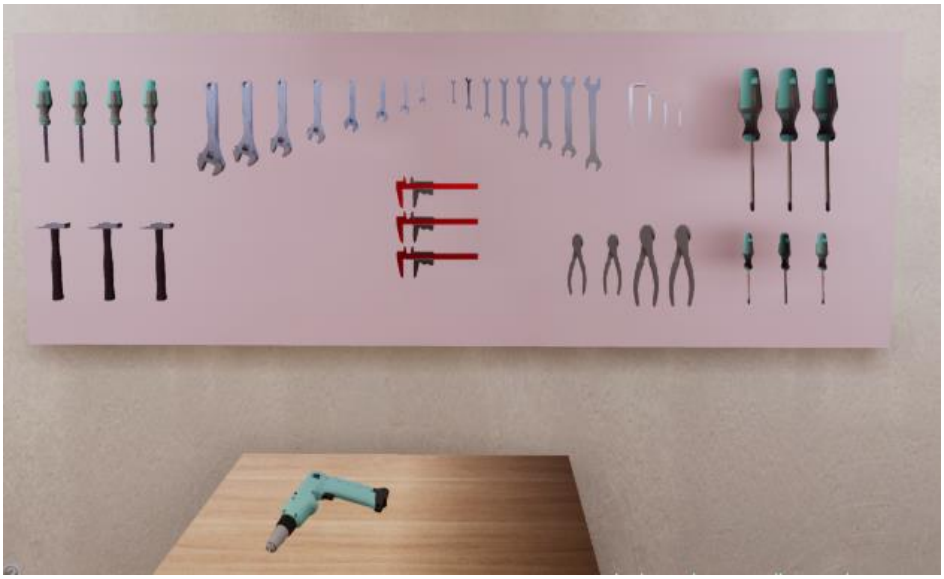


Ilustración 97 - Panel de Herramientas.

- **Ventanas**

Para que pueda entrar iluminación del exterior, se introducen dos ventanas a ambos lados del panel de herramientas. Para ello, primero se creará el hueco en el que se introducirá la geometría de la ventana, colocando una caja de pincel de carácter sustractivo que lo que hace es eliminar material sobre otro, en este caso la pared.

Una vez se tiene el hueco, la geometría de la malla se descarga, se modifica en 3DS Max, y se importa como Datasmith en UE4. Por último se colocan y se aplican los materiales.

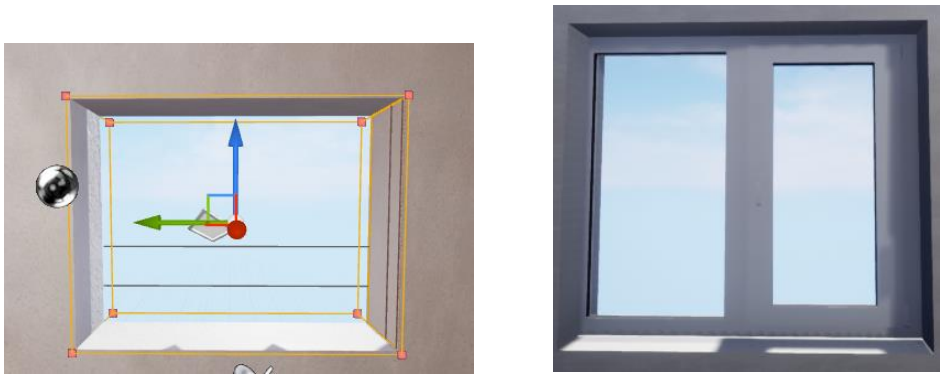


Ilustración 98 - Creación de las ventanas.

- **Puerta Interior**

Para introducir una puerta de interior que simule la vía de conexión entre el garaje y la casa, se realiza el mismo procedimiento que con las ventanas. La caja pincel tendrá otras dimensiones, y la geometría de la puerta también se obtiene de Internet. Como material, se aplica el de Madera de pino, y un dorado metálico para la manilla.

Ilustración 99 - Puerta de interior.

- **Puerta Garaje**

La puerta de garaje se realiza mediante una caja rectangular y un marco alrededor, que sobresalgan de la pared unos centímetros. Mediante la creación, a partir de unas texturas, de un material metalizado, se consigue en apariencia una puerta gris metálica.

Ilustración 100 - Puerta Garaje.

- **Lámparas**

El modelado de las lámparas que se sitúan en el techo se realiza en 3DS Max mediante la superposición de cilindros, que simulen el exterior de la lámpara y el cristal que protegerá la bombilla. Los materiales serán un metal blanco y el material cristal.

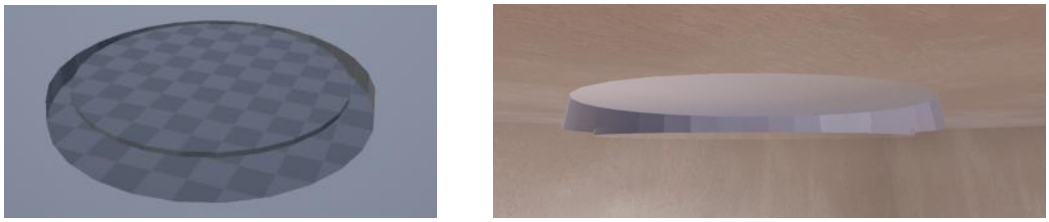


Ilustración 101 - Lámparas.

De este modo, y aún sin construir la iluminación, la habitación quedaría de la siguiente manera:

Ilustración 102 - Garaje.

Ilustración 103 - Garaje 2.

Como dato, cualquier archivo Datasmith importado a UE4 habrá de ser convertido en malla estática para poder trabajar de una mejor forma con él. Esto se realiza aplicando botón derecho sobre cualquier actor, en la opción *Convert to a Static Mesh*.

e. Iluminación

Se realiza la iluminación del entorno partiendo desde cero, y desde el exterior hacia el interior. Con esto se quiere decir que la iluminación por defecto de ambas plantillas es eliminada, y que se comienza por el exterior de la habitación y se termina en el interior de esta.

Se introduce un *DirectionalLight*, que va a simbolizar la luz solar. Se le aplican los giros necesarios para establecer así la dirección que tendrán los rayos solares, y se coloca en un sitio cualquiera del entorno, preferiblemente fuera de la habitación. Mediante *AtmosphericFog* se dota al entorno de una sensación realista de atmósfera, densidad de aire y luz. A continuación y para aumentar más aún el realismo, se introduce *SkyLight*, simbolizando la luz del cielo.

Para que todos los materiales existentes, suelo, mobiliario, paredes, etc, simulen reflejos naturales hay que aplicar una serie de filtros específicos. Estos son: *SphereReflectionCapture*, que se colocará en el centro de la habitación para que los fotones se reflejen desde aquí; *BoxReflectionCapture*, que es un componente que habrá que colocar en ventanas y puertas que posean cristales por los que pueda entrar luz exterior; *LightmassImportanceMass*, que indica el área que va a ser iluminada y por

último *LightMassPortal*, que lo que hace es ayudar al motor de juego a saber por dónde viene la luz exterior, por lo que se colocará uno en cada ventana.

En cuanto a la iluminación interior, se emplean 4 puntos de luz que se colocan en el techo, en cada una de las lámparas previamente colocadas, con una intensidad lumínica de 20 cd. Los dos que se encuentran más cercanos a las mesas, serán de tipo estacionario, mientras que los dos restantes serán estáticos. No se pueden colocar más de 4 puntos de luz de carácter estacionario.

Un aspecto realmente importante en la iluminación es la complejidad de la luz. Esta característica puede verse cambiando el modo en la opción Iluminado. De esta forma, se podrá ver un gráfico de colores que muestra lo fácil o difícil que va a ser la construcción de la luz. Por ello, es esencial que los colores sean azules, verdes o incluso amarillos.

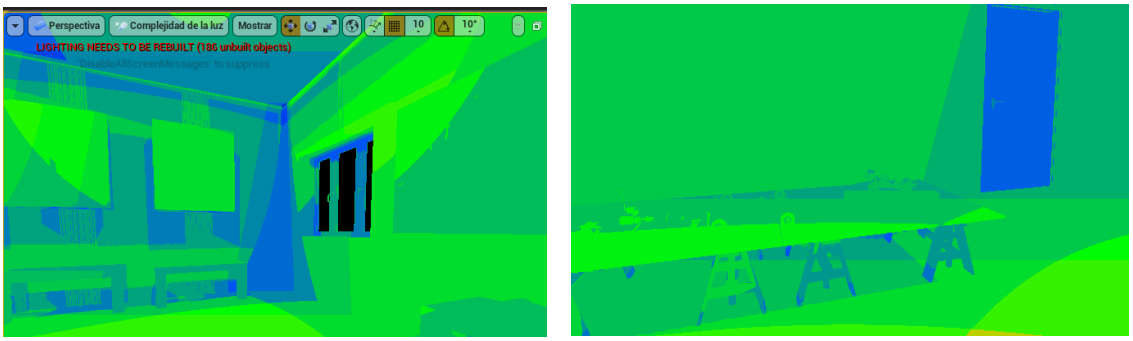


Ilustración 104 - Complejidad de la luz.

Cuando todas las características han sido estudiadas y la iluminación es correcta, se procede a la construcción. Para ello, primero se decide el nivel de construcción: cuanto más alto, mejor calidad; y después se procede a la construcción en *Opciones de diseño*. Unos minutos después, se tendrá el resultado.

f. Sonido

Se introducen unos sonidos como ejemplo, para que se vea lo complejo y completo que es este motor. Todos los sonidos son descargados e importados en formato *.wav*.

El primero es un sonido ambiente que se colocará en un punto cualquiera del entorno. El sonido se reproducirá a lo largo del desarrollo de la aplicación.

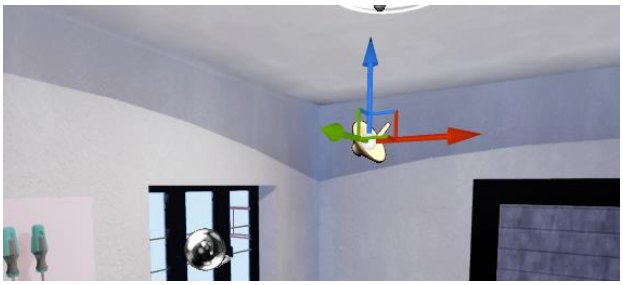


Ilustración 105 - Colocación sonido ambiente

El segundo es un sonido que simboliza que cada pieza ha sido colocada correctamente. Para ello, y justo después de haber ajustado la visibilidad de las piezas, se introduce *PlaySound2D*, que permite reproducir cualquier sonido seleccionado previamente importado. Tras varios intentos viendo que el sonido se repetía en bucle y de manera incorrecta, se coloca *DoOnce*, que disparará la ejecución una única vez.

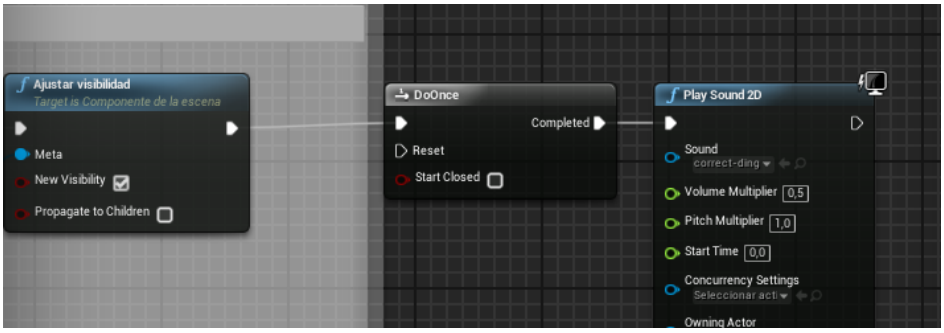


Ilustración 106 - Sonido Ding.

2. Conducción del vehículo

En esta segunda parte del proyecto que se desarrolla se va a permitir la conducción del vehículo montado. No se van a tener dos proyectos diferentes, si no que en el mismo se posibilitará la conducción en realidad virtual inmersiva y no inmersiva, dependiendo del dispositivo que utilice el jugador.

Unreal Engine 4 dispone de una plantilla de conducción de un vehículo. Por tanto, el trabajo en esta parte consistirá en adaptar el vehículo radiocontrolado para poder manejarlo en la plantilla, así como modificar el entorno.

Ilustración 107 - Plantilla Vehículo

a. Importación del vehículo

Lo primero que se hace es una simplificación del vehículo, es decir, se eliminan algunas piezas y conjuntos que no se van a ver en la conducción para simplificar su procesado. A continuación, hay que realizar es un *rig* (aparejo) en 3DS Max. Esto significa que, el coche va a tener diferentes partes cada una con un movimiento independiente, como son las ruedas, que giran en torno al eje, y el vehículo entero, que tendrá que desplazarse con las ruedas.

Para ello, se crea una estructura de huesos (*bones*), de forma que cada rueda tenga asociado uno, y que los cuatro estén unidos a un hueso principal o *root*. Por tanto, se crea primero el hueso *root* en el menú *Crear>System>Bones*. Este hueso será copiado cuatro veces más, uno por cada rueda, y cada uno nombrado según corresponda. Es importante la ubicación de los huesos correspondientes a cada rueda, ya que habrá que colocarlos en cada eje y no en las ruedas. Por otro lado, el *root* será colocado en un punto aproximadamente equidistante a los cuatro, sin importar su ubicación.

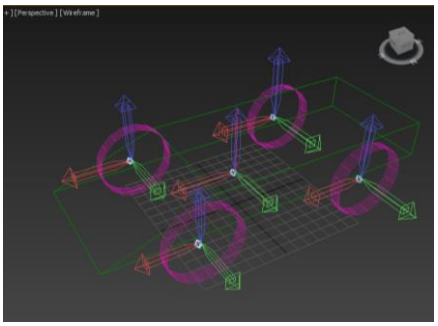


Ilustración 108 - Posición Ejes en huesos.

También será esencial la ubicación de los ejes de los huesos. Al colocar el primer hueso, los ejes quedan al azar. Para hacer que los ejes tengan la orientación correcta, se crea una geometría simple, como por ejemplo un cubo. El cubo se colocará con los ejes en la posición que se desea para los huesos, se seleccionará el hueso y se empleará *Align* (Alinear). De esta forma, el hueso tendrá la orientación correcta, se eliminará la geometría creada y se harán cuatro copias de los huesos, para las rueda.

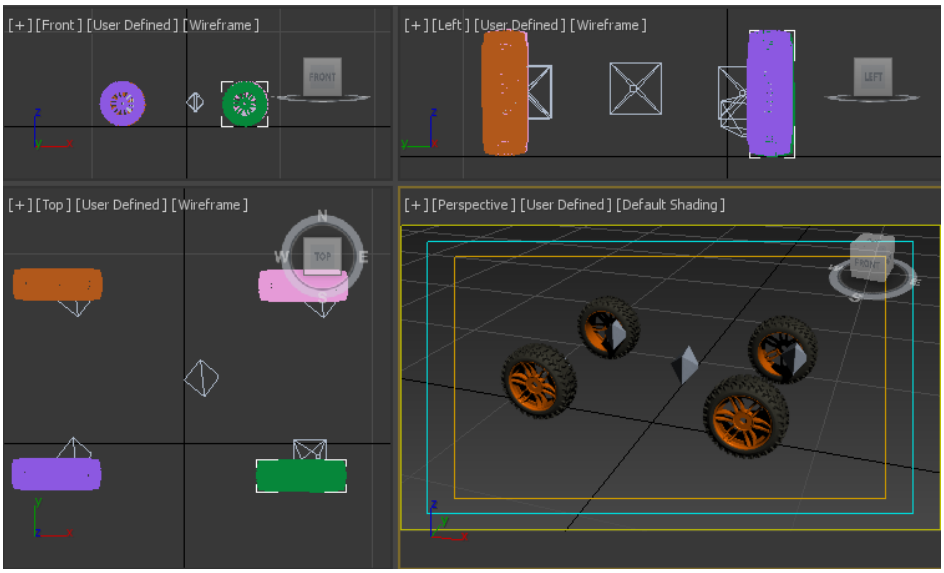


Ilustración 109 - Ubicación de los huesos.

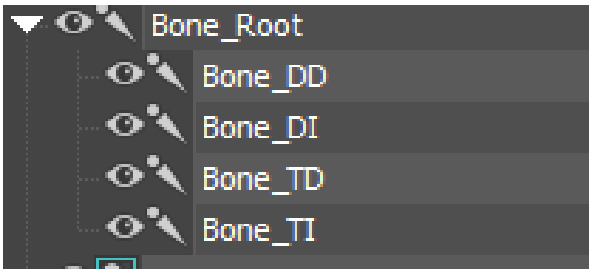


Ilustración 110 - Jerarquía de huesos.

Los cuatro huesos de cada rueda tienen que ser unidos al hueso *root*, estableciendo una jerarquía. De este modo, cualquier desplazamiento, rotación, o escalado en el *root* hará que los otros cuatro se vean afectados por igual. Para ello, se seleccionan los cuatro huesos de las ruedas, y mediante *Select and link* se arrastra hasta el hueso *root*, estableciendo el árbol que se muestra en la ilustración.

Como se ha comentado antes, cada rueda ha de tener un movimiento propio y éste estará ligado a cada hueso. Para lograr estos movimientos, se selecciona cada rueda, y convirtiéndola previamente en una polilínea editable (Editable Poly), con el modificador Skin se añadirá el hueso que corresponda a cada rueda. Por último, se debe hacer lo mismo seleccionando todo el vehículo y aplicándole el mismo modificador para asociarle el hueso *root*. Esto permitirá que todas las piezas en bloque se muevan a la vez.

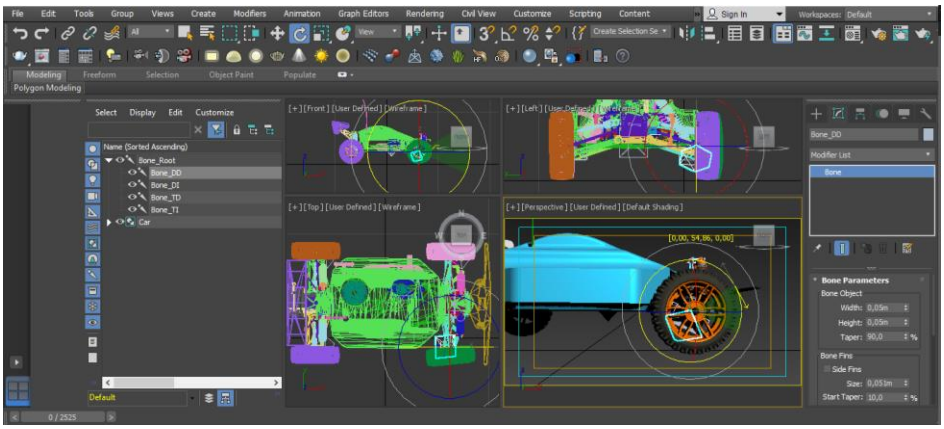


Ilustración 111 - Giro solidario hueso y rueda.

Este archivo es exportado en formato *.fbx* e importado en Unreal Engine 4. Es muy importante realizar una correcta importación para que el vehículo pueda desplazarse por el entorno.

Los activos que habrá que crear serán, una malla esquelética, un activo de física, un Blueprint de animación, un modelo de vehículo, dos Blueprint de ruedas y una activo de datos.

Con el **activo de datos** se establece la fricción de los neumáticos, en un valor entre 1 y 2. Se crea en *Miscellaneous> Data Asset> Tire Type*.

Los Blueprint de ruedas se realizan porque las ruedas delanteras se ven afectadas por la dirección, mientras que las traseras no. Para crearlos, se selecciona *Vehicle Wheel* en la clase de Blueprint, y se editan algunos valores de la siguiente manera:

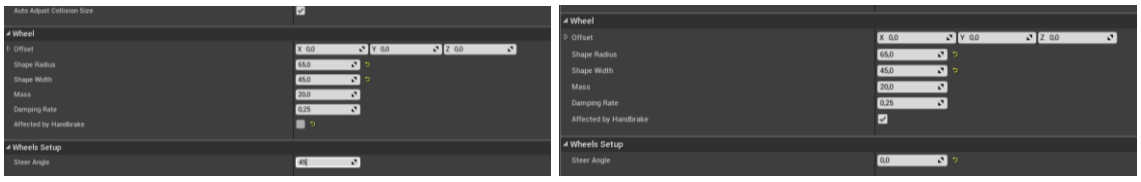


Ilustración 112 - BP Ruedas Delanteras VS. Traseras.

Los valores que se modifican son radio de forma, ancho de forma, afectado por el freno de mano, y ángulo de dirección.

El Blueprint de animación proporcionará animaciones especiales al vehículo. Estas animaciones se pueden controlar fácilmente con el nodo *Wheel Handler*. Para crear este Blueprint, a partir de agregar nuevo se seleccionará, sobre *Animación*, *Animation Bluprint.* Una vez abierto, en configuración de clase se seleccionará *VehicleAnimInstance*, ya que de no hacerlo no se podrá acceder a otros nodos. Ya en el gráfico del evento, se emplearán los nodos *Mesh Space Ref Pose* y el mencionado *Wheel Handler*.

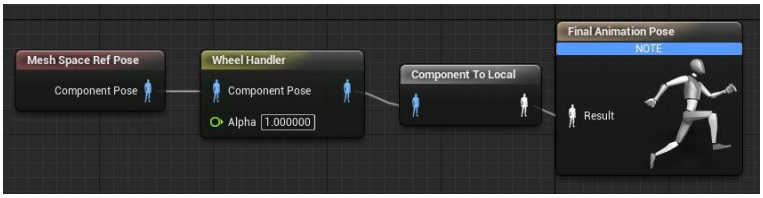


Ilustración 113 - Blueprint de animación.

El Blueprint del vehículo proporcionará al coche la funcionalidad que se busca. Se crea eligiendo la clase *WheeledVehicle* dentro de las clases de Blueprints, y lo primero que se hará será asociarle una malla esquelética. Esta malla esquelética no es otra que el archivo exportado de 3DS Max con la estructura de huesos, y que será importada a Unreal Engine 4. En la pestaña *Animation*, se seleccionará el Blueprint de animación creado anteriormente. A la malla se le asocia un brazo extensible y una cámara, que proporcionarán la vista de la conducción.

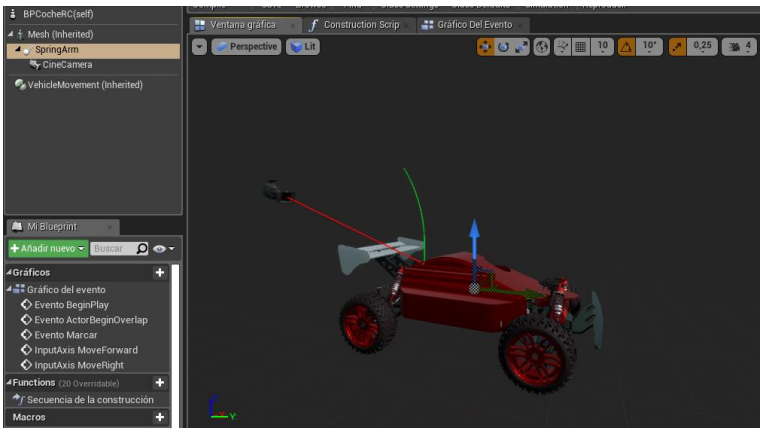
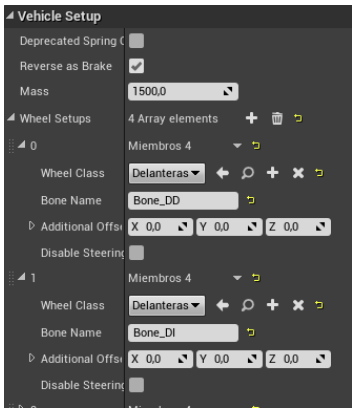


Ilustración 114 - Vehículo y cámara (BP del vehículo).



En ajustes del vehículo (*Vehicle Setup*) en la pestaña *Wheel Setups* aparecen 4 subpestañas en las que habrá que introducir los nombres de los huesos de cada rueda, así como si son ruedas delanteras o traseras.

En caso de que se necesitasen más número de ruedas, podrían añadirse pulsando sobre el icono "+".

Ilustración 115 - Vehicle Setups

El archivo de físicas que se genera automáticamente al importar una malla esquelética. En él, habrá que crear unos cuerpos para la malla, asociados a los huesos que la conforman. Para ello, se seleccionará cada hueso y en la pestaña *Body Creation* se seleccionará el tipo de cuerpo que se va a crear en ese hueso. Para el root será una caja, y para los huesos de las ruedas serán esferas.

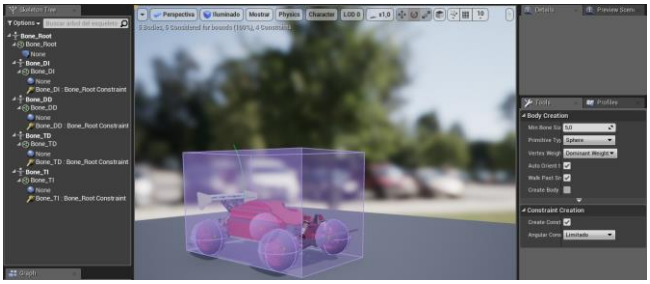


Ilustración 116 - Físicas de la malla esquelética.

El vehículo es importado sin materiales, por lo que se emplearán los de los otros proyectos, habiendo que migrar la carpeta que contenía las piezas a este proyecto para poder disponer de ellos. Se introducen uno a uno en la malla esquelética.

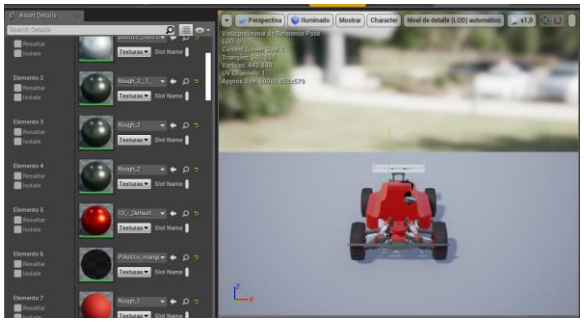


Ilustración 117 - Materiales del vehículo en malla esquelética.

Quedarían por describir los **controles**, que van a ser básicamente movimiento hacia adelante, y movimiento a la izquierda. En *Ajustes del proyecto* > *motor* > *entradas*, se generan asignaciones del eje, que van a ser movimiento hacia adelante (*MoveForward*) y movimiento hacia la derecha (*Move Right*). Los movimientos hacia atrás y hacia la izquierda se realizan poniendo una escala negativa. También se podrá mirar hacia cualquier dirección con el ratón en caso de usar VR no inmersiva mediante *LookUp* y *LookRight*.

- **Adelante y atrás:** en RV inmersiva se realiza con los gatillos (Trigger) derecho e izquierdo, respectivamente. En RV no inmersiva se realiza con *W* y ↑, y con *S* y ↓, respectivamente.
- **Derecha e izquierda:** en RV inmersiva se realizan con el panel táctil. En no inmersiva, con *D* y →, y con *A* y ←, respectivamente.

Ilustración 118 - Controles conducción.

b. Entorno para la conducción y materiales empleados

Se realiza un entorno para la conducción como si fuera un terreno al aire libre en el que se encuentran diversos elementos. Estos elementos son: una pista de asfalto, un terreno de tierra rugoso, un conjunto de rampas metálicas, y un circuito con conos. Para delimitar el entorno, se crean paredes a diferentes alturas. A continuación se hace una descripción breve de los elementos:

- **Paredes y suelo**

Como se ha dicho, se colocan paredes para delimitar el terreno a lo largo del perímetro del suelo. Para estas paredes se emplea un material que simule paredes de ladrillos, por lo que se sigue el procedimiento visto anteriormente para la creación de materiales. La textura de ladrillos se descarga de Internet, creando el material y aplicándole el nodo *Texture Coordinates* a una escala de 20.

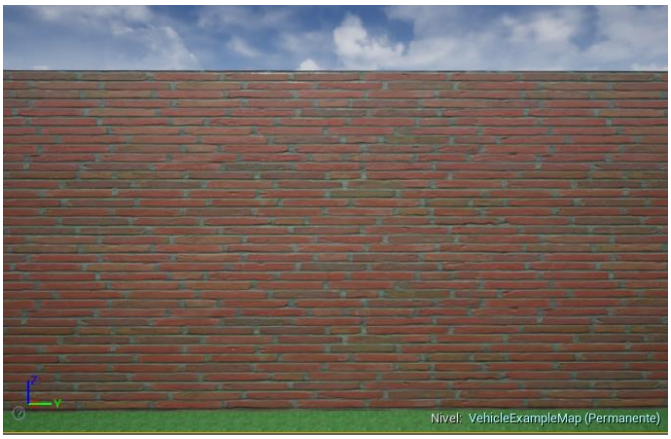


Ilustración 119 - Pared de ladrillo.

Para el suelo se decide colocar un material que simule el césped. Esta textura también es descargada e importada, y el material creado desde cero y modificado.

Ilustración 120 - Suelo césped.

- **Pista de asfalto**

La pista de asfalto fue creada mediante la unión de diversas rampas con distintos desniveles en 3DS Max, utilizando las técnicas aprendidas a lo largo de este proyecto. El archivo es exportado en formato Datasmith e importado a UE4, para aplicarle un material asfáltico cuya textura también fue descargada.

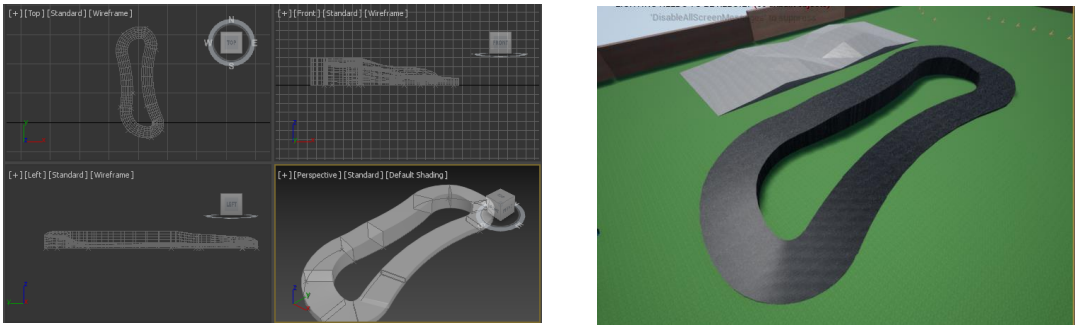


Ilustración 121 - Pista en 3DS Max y en UE4.

- **Terreno rugoso**

El terreno rugoso se encontraba en la plantilla, por lo que sólo se ha recolocado y aplicado un material creado con unas texturas descargadas.

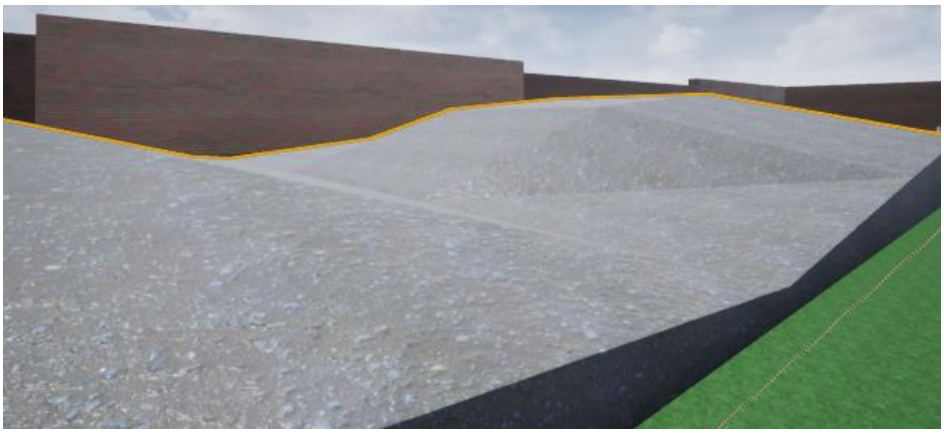


Ilustración 122 - Terreno rugoso.

- **Rampas metálicas**

Estas rampas también se encontraban en la plantilla y se les aplica un material metálico de los que se importaron del otro proyecto.

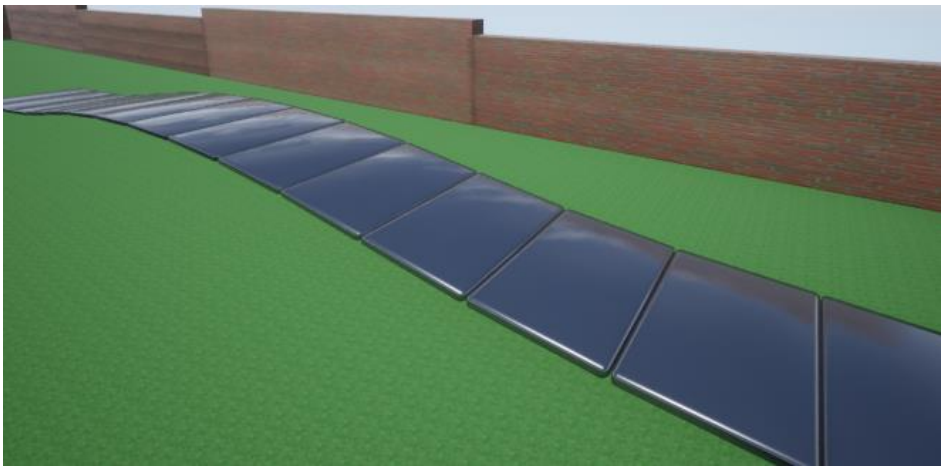


Ilustración 123 - Rampas metálicas.

- **Circuito de conos**

El circuito de conos es creado para que el jugador pueda hacer eslalon entre ellos. Fue esencial descargar la geometría del cono y sus materiales, por lo que sólo fue necesario importar los archivos, y colocarlos para crear el circuito.

Ilustración 124 - Circuito de conos.

c. Iluminación

La iluminación de este entorno ya es proporcionada por la plantilla. Se modifican algunos valores de los modificadores introducidos pero básicamente es la misma iluminación exterior que se explicó anteriormente. En este caso no se emplea iluminación para el interior, y se construye directamente la iluminación de la escena.

3. Referencias

[1] docs.unrealengine.com/en-US/Engine/Physics/Vehicles/VehicleUserGuide

Conclusiones

En general los resultados que se han obtenido han cumplido los objetivos que se plantearon al principio de este proyecto.

Como el proyecto ha constado de dos partes, se decidió dar algo más de importancia al montaje ya que es la parte más visual e interactiva. Se ha tratado de realizar simulaciones muy cercanas a la realidad, consiguiéndose unos entornos muy cálidos y movimientos y animaciones realistas. Además, los efectos de iluminación y sonido han formado una parte muy importante a pesar de que en un principio no se había planeado incluir algunos de ellos.

El desarrollo de este trabajo ha llevado mucho tiempo de aprendizaje inicial. Los conocimientos en los programas 3DS Max y Unreal Engine 4 eran nulos, por lo que al tratarse de programas nuevos se ha tenido que partir desde cero en ambos para ir exponencialmente ampliando los conceptos y adaptando las ideas que iban surgiendo al proyecto mediante estos programas. Este aprendizaje ha sido continuo desde que se adjudicó el proyecto hasta que se ha llegado a este punto, y por supuesto, podrían realizarse y ampliarse muchos más conceptos.

Tanto los entornos tridimensionales como la realidad virtual ofrecen un gran número de posibilidades. A lo largo de esta memoria se han llevado a cabo un gran número de tareas e ideas cumplimentándolas con los programas, por lo que se han obtenido bastantes conocimientos y alcanzado y superado prácticamente todos los objetivos marcados inicialmente.