

Índice general

1. Plataforma de Desarrollo	7
1.1. Blender	7
1.2. Simulador Gazebo	8
1.3. JdeRobot	8
1.4. ROS	9
1.4.1. MoveIt!	11
1.4.2. rviz	11
1.4.3. rqt	11
1.5. ARIAC	12
2. Práctica Mónaco	13
2.1. Manejo de Blender	13
2.1.1. Interfaz	13
2.2. Circuito plano	17
2.3. Circuito con elevaciones	22
2.4. Mundos para Gazebo	24
Bibliografía	25

Índice de figuras

2.1.	Interfaz de Blender	14
2.2.	Detalle de Manipuladores de transformaciones en 3D.	14
2.3.	Detalle del cursor de Blender.	15
2.4.	Ventana de propiedades.	16
2.5.	Trazado del circuito de Mónaco usado como plantilla	18
2.6.	Detalle de curva Bezier.	18
2.7.	Trazado del circuito de Mónaco superpuesto a la plantilla.	19
2.8.	Segmento del circuito.	20
2.9.	Circuito (sólo el trazado).	20
2.10.	Circuito plano.	21
2.11.	Diferentes vistas del circuito de F1 de Mónaco plano	22
2.12.	Fondo para el circuito con elevaciones.	23
2.13.	Circuito con elevaciones.	23
2.14.	Diferentes vistas del circuito de F1 de Mónaco con elevaciones	24

Índice de cuadros

Capítulo 1

Plataforma de Desarrollo

1.1. Blender

Blender[1] es un software libre y gratuito de creación en 3D. Está diseñado para realizar tareas como modelado, iluminación, renderizado, animación y creación de gráficos tridimensionales, edición de vídeo y escultura y pintura digital. En Blender, además, se pueden desarrollar videojuegos ya que posee un motor de juegos interno. Actualmente es compatible con todas las versiones de Windows, Mac OS X, GNU/Linux (Incluyendo Android), Solaris, FreeBSD e IRIX. Su interfaz utiliza OpenGL¹ (*Open Graphics Library*) para proporcionar una experiencia consistente y de calidad.

Blender fue liberado al mundo bajo los términos de la Licencia Pública General de GNU v2 (GPL)², y su desarrollo continúa conducido por un equipo de voluntarios procedentes de diversas partes del mundo y liderados por el creador de Blender, Ton Roosendaal.

Se eligió este programa para realizar la edición 3D de mundos para Gazebo frente a alternativas como 3DSMax, Maya o XSI por diversos motivos: es más ligero que sus competidores; posee más herramientas de escultura 3D; la comunidad es muy activa y hay gran cantidad de información, tutoriales y soluciones disponibles; y es de distribución comercial libre y gratuita.

¹<https://www.opengl.org/>

²<https://www.gnu.org/licenses/old-licenses/gpl-2.0.html>

1.2. Simulador Gazebo

Gazebo[2] es un simulador 3D de robots para interiores y exteriores, con un motor de físicas y cinemáticas muy potente. Dispone de un conjunto de plugins que facilita la integración con ROS, lo cual agiliza el desarrollo de código y permite la simulación de algoritmos antes de implementarlos en el robot físico, lo cual se puede lograr sin realizar apenas cambios en el código. Además está mantenido por una comunidad activa y la OSRF³ (*Open Source Robotics Foundation*), la cual también da soporte a ROS, y fué elegido para realizar el DARPA Robotics Challenge⁴ entre 2012 y 2015.

Cabe destacar que Gazebo se compone principalmente de un cliente y un servidor. El servidor es el encargado de realizar los calculos y la generación de los datos de los sensores, y puede ser usado sin necesidad de una interfaz gráfica, por ejemplo en un servidor remoto. El cliente proporciona una interfaz gráfica basada en QT que incluye la visualización de la simulación y una serie de controles de multitud de propiedades. Esta configuración permite lanzar multiples clientes sobre un servidor, consiguiendo multiples interfaces de la misma simulación

1.3. JdeRobot

JdeRobot[3] es una suite de desarrollo de software de robótica, domótica y sistemas de visión computerizados cuya última versión, la 5.5, es la usada en este proyecto y permite la integración con ROS Kinetic. Proporciona un entorno distribuido donde las aplicaciones se forman mediante una colección de componentes asíncronos. Estos componentes utilizan interfaces ICE⁵ para comunicarse, lo que permite lanzarlos desde distintos equipos y que estén escritos en diferentes lenguajes, como C++, Python o Java.

JdeRobot simplifica el acceso a elementos de hardware, siendo tan simple com realizar una llamada a una función. También permite que los sensores o actuadores con los que se comunica sean reales o simulados, conectados mediante la red tanto dentro de la misma máquina como en una red local o de forma remota mediante internet. Actualmente se han desarrollado drivers para una multitud de dispositivos, como por ejemplo sensores RGB como Kinect o cámaras USB o IP, vehículos como Kobuki o Pioneer, drones como el ArDrone de Parrot, etc.

³<http://www.osrfoundation.org/>

⁴<http://www.theroboticschallenge.org/>

⁵<http://www.zeroc.com/>

Es un software libre, licenciado como GPL y LGPL que se sirve de software como Gazebo, ROS, OpenGL, QT... que se usa tanto para docencia como para investigación en la URJC y ha formado parte del *Google Summer of Code 2015*⁶, programa donde Google premia a los estudiantes al completar un proyecto de programación de software libre durante un verano.

1.4. ROS

El Sistema Operativo de Robots, ROS[5] (*Robot Operating System*) es un entorno de trabajo flexible donde desarrollar software para robots. Es una colección de herramientas y librerías que buscan simplificar la tarea de crear comportamientos robustos y complejos en una amplia variedad de plataformas robóticas.

Desde mediados de la década de los 2000 se han realizado esfuerzos para aunar los entornos y herramientas existentes en sistemas de software dinámicos y flexibles para su uso en robots. Estos esfuerzos culminaron, gracias al interés y la ayuda de innumerables desarrolladores, en las ideas que forman el núcleo de ROS y sus paquetes de software fundamentales. Desde entonces se ha expandido su uso bajo la licencia de software libre BSD⁷ y se ha convertido en una plataforma muy usada en la comunidad de investigadores.

Desde el inicio ROS se desarrolló en multitud de instituciones para multitud de robots, permitiendo a una gran variedad de investigaciones tener éxito bajo esta plataforma. Sólo el núcleo de ROS se mantiene en los servidores centrales, cualquier desarrollador es libre de crear, desarrollar y compartir sus propias ideas y proyectos de forma que, si así lo desea, estén disponibles para toda la comunidad desde sus propios repositorios. De esta forma se consigue mantener un ecosistema formado por decenas de miles de usuarios a nivel global, desde proyectos como hobby hasta sistemas industriales automatizados.

ROS se diseñó para ser modular y fragmentado, de modo que los usuarios pueden usar sólo las partes que necesiten. En bajo nivel ofrece una interfaz de comunicación por mensajes que permite ahorrar tiempo manejando los detalles de la comunicación entre nodos mediante un mecanismo anónimo de publicación/subscripción de mensajes estructurados. Este sistema fuerza al usuario a implementar interfaces limpias entre los nodos del sistema, mejorando la encapsulación y promoviendo la reutilización de código.

Adicionalmente ROS proporciona librerías y herramientas para agilizar el trabajo de sus

⁶<https://summerofcode.withgoogle.com/organizations/6493465572540416/>

⁷https://es.wikipedia.org/wiki/Licencia_BSD

usuarios. Dado el carácter colaborativo y comunitario del proyecto, se han unificado una gran variedad de formatos mensajes estándar que cubren la mayoría de las necesidades básicas en robótica, tales como posiciones, transformaciones, vectores, sensores como cámaras o lasers, datos de navegación como caminos o mapas, etc.

Un problema común que aborda ROS es la descripción de un robot de forma que sea comprensible para un ordenador, consiguiendo un Formato Unificado de Descripción del Robot o URDF (*Unified Robot Description Format*). Consiste en un fichero XML en que se describen las propiedades físicas del robot, partiendo del cual el robot se puede utilizar con librerías, simuladores y planificadores de movimientos.

También proporciona herramientas de diagnóstico, estimación, localización, navegación, así como una colección de herramientas gráficas y de línea de comandos para facilitar el desarrollo y la depuración. Las herramientas de línea de comandos permiten la utilización de ROS desde cualquier terminal, incluso con conexión remota. Las herramientas gráficas incluyen rviz y rqt, muy potentes tanto para planificar como para desarrollar proyectos en ROS.

Para entender mejor cómo funciona ROS podemos pensar en un sistema de grafos en el que situamos los siguientes elementos:

- Nodos: Son las partes de código que se ejecutan. Escritos en C++ o Python permiten realizar tareas en el robot, suscribirse y publicar en topics o proporcionar y usar servicios. De esta manera se facilita el diseño modular de los proyectos.
- Topics: Son las vías de comunicación usadas por los nodos. Cada topic utiliza un único tipo de mensaje, de esta manera un nodo puede utilizar varios topics para comunicarse.
- Mensajes: Son los datos estructurados que se envían entre topics. En ROS existen varios tipos definidos para los mensajes más utilizados, pero se pueden definir nuevos tipos de mensajes de acuerdo con las necesidades particulares de cada proyecto
- Servicios: A diferencia de los topics, son vías de comunicación síncronas entre nodos compuestas por dos mensajes: uno de petición y otro de respuesta. De esta forma el nodo que envía la petición espera hasta recibir la respuesta.

1.4.1. MoveIt!

MoveIt![6] es un software de código abierto para ROS (Robot Operating System) que es el estado de la técnica de software para la manipulación móvil. De hecho, podríamos afirmar que se está convirtiendo en un estándar de facto en el campo de la robótica móvil, ya que hoy en día más de 65 robots utilizan este software.

Incluye diversas utilidades que aceleran el trabajo con brazos robóticos, y sigue la filosofía de ROS de reutilización de código. Este software permite llevar a cabo tareas de planificación de trayectorias complejas, percepción 3D, cálculos cinemáticos, control de colisión, control y navegación de forma sencilla, accediendo por la API o mediante las herramientas de la consola.

1.4.2. rviz

Rviz[7] es una herramienta de visualización en 3D llamada que posibilita que prácticamente cualquier plataforma robótica pueda ser representada en imagen 3D, respondiendo en tiempo real a lo que le ocurre en el mundo real. Se puede usar para mostrar lecturas de sensores y obtener información de estado de ROS.

Usado en conjunto con MoveIt! permite mostrar el brazo en su estado actual, la colocación del brazo en una posición objetivo, y la visualización del camino pensado por MoveIt! y del movimiento real del brazo siguiendo dicho camino.

1.4.3. rqt

Rqt[8] es un software de ROS que implementa varias herramientas de GUI (*Graphical User Interface*) en forma de plugins, permitiendo cargarlas unificadas como una ventana en la pantalla facilitando trabajo al usuario. Simplemente con un comando en la consola, *rqt* muestra una ventana donde elegir cualquier plugin disponible en el sistema en ese momento.

Contiene una herramienta que ha resultado muy útil para la realización de este proyecto: *rqt_graph*. Al introducir en la consola *rosrun rqt_graph rqt_graph* crea un grafo dinámico que muestra qué nodos y qué topics están activos en ese momento y cuál es su relación. Al situar el ratón encima de cada elemento marcará con un código de color cuál es el elemento activo, de qué tipo es y cuál es su relación con los demás elementos del grafo.

1.5. ARIAC

ARIAC[4] (*Agile Robotics for Industrial Automation Competition*) es una competición pionera cuyo objetivo es probar la agilidad de los sistemas robóticos industriales. Realizada por primera vez en Junio de 2017, nace de un esfuerzo conjunto entre la Conferencia de Automatización en Ciencia e Ingeniería del IEEE o CASE (*Conference on Automation Science and Engineering*) y el NIST (*National Institute of Standards and Technology*). Se sirve de Gazebo como plataforma de simulación y de un conjunto propio de modelos, plugins y scripts para simular un brazo robótico en un entorno dinámico, todo ello elaborado con la ayuda de la OSRF (*Open Source Robotics Foundation*). Tiene el objetivo de aumentar la productividad y la autonomía de los robots industriales, entendiendo como agilidad la cosección de manera automática de identificación de fallos y recuperación de los mismos, automatización para disminuir la reprogramación ante cambios en la producción, interacción con el entorno incluso en áreas no previstas inicialmente, y la capacidad de *plug and play*, es decir, de introducir robots de otros fabricantes sin la necesidad de reprogramarlos.

En este proyecto nos servimos del entorno y del propio brazo robótico para desarrollar nuestros propios plugins para entender y controlar el brazo por medio de ROS.

Capítulo 2

Práctica Mónaco

Una vez explicado el contexto, los objetivos y las herramientas usadas en el proyecto, en este capítulo vamos ver en detalle la construcción de un mundo para JdeRobot, en concreto el circuito de Fórmula 1 de Mónaco.

2.1. Manejo de Blender

En esta sección vamos a explicar cómo se utiliza este potente programa de creación, renderizado y animación de gráficos tridimensionales. El uso de este tipo de programas es, a priori, de los más difíciles, ya que trabajar sobre un mundo tridimensional en una pantalla de ordenador, desplazando un ratón sobre una mesa en dos dimensiones, exige un esfuerzo de abstracción considerable.

Es por ello que en los primeros pasos de esta sección nos dedicaremos a presentar la interfaz del programa y exponer la inmensa cantidad de opciones y herramientas de que dispone, centrándonos en aquellas que han resultado relevantes para la consecución de los objetivos marcados.

2.1.1. Interfaz

Como se verá, la interfaz de Blender no sigue el patrón típico de los programas a los que estamos habituados, como editores de texto y hojas de cálculo o entornos de desarrollo de software, por lo que resulta fácil desorientarse al principio.

La interfaz de usuario de Blender está compuesta por 4 ventanas por defecto como se pueden diferenciar en la imagen (*Figura 2.1*). Cada ventana tiene una cabecera con las

herramientas adecuadas para trabajar sobre dicha ventana y, a su vez, cada herramienta está dotada de sus correspondientes pestañas para una completa edición. Esta disposición facilita y agiliza el uso apropiado del programa. A su vez, cada cabecera de cada ventana tiene el botón Tipo de Editor, mediante el cual se puede cambiar el tipo de ventana que se muestra, por lo que se puede personalizar el aspecto para agilizar el trabajo.

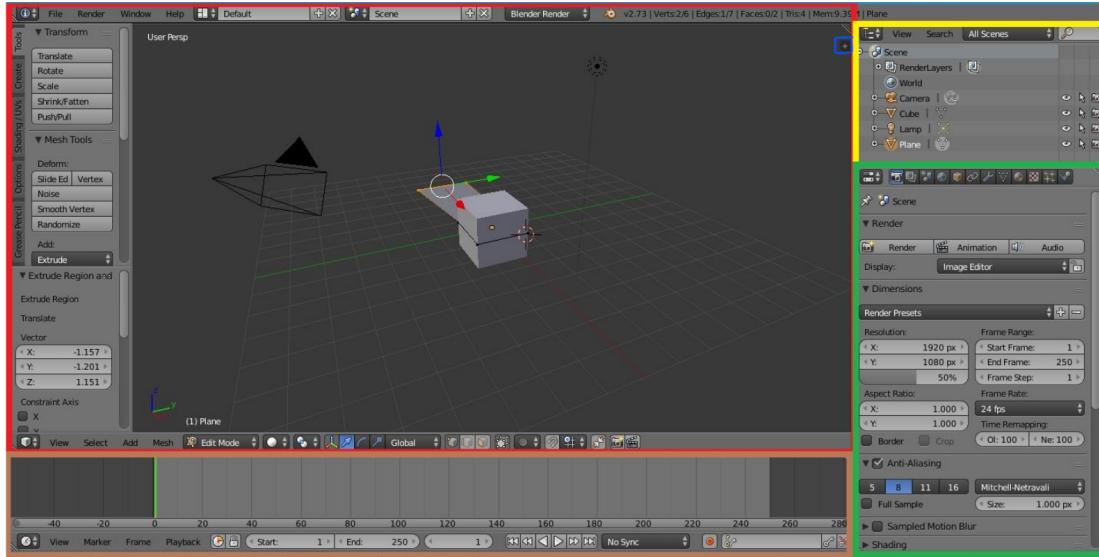


Figura 2.1: Interfaz de Blender.

En primer lugar tenemos la ventana de Vista 3D, bordeada en color Rojo. En esta ventana se visualiza todo el trabajo y los cambios que se realizan con el programa.

En esta ventana podemos observar los siguientes objetos y menús:

- Manipuladores de Transformaciones en 3D (*Figura 2.2*): Muestra de manera visual información de las transformaciones a realizar. Cada objeto puede ser transformado de tres maneras: translación (G), rotación (R) y escalado (S). Utilizando la combinación Ctrl+Space, o bien haciendo clic en el ícono del sistema de coordenadas, se puede mostrar y ocultar el manipulador.

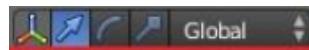


Figura 2.2: Detalle de Manipuladores de transformaciones en 3D.

- Cursor 3D (*Figura 2.3*): El cursor 3D es una herramienta muy útil y se utiliza para una variedad de cosas como representar el lugar donde se añadirán nuevos objetos o representar el punto de pivote para una rotación. Sin embargo es muy engorroso a la

hora de trabajar ya que es muy fácil moverlo accidentalmente y al necesitar usarlo se pierde bastante tiempo reubicándolo.



Figura 2.3: Detalle del cursor de Blender.

- Cubo: Al iniciar Blender, por defecto, aparece un cubo situado en el centro de la ventana 3D. Es uno de los elementos básicos que se pueden insertar y la mayoría de las veces basta con este simple cubo para comenzar a trabajar.
- Luz (de tipo lámpara): Al iniciar Blender, por defecto, también aparecerá una Luz de tipo lámpara que estará en algún sitio cerca del centro de la escena. Es posible crear otras fuentes de luz, como un sol en el cenit de la escena o una luz direccional que ilumine todos los objetos. En cualquier caso, aunque en los diferentes modos de edición no sea necesario, a la hora de renderizar la escena para visualizar el resultado del trabajo es necesaria alguna fuente de luz, si no aparecerá una escena llena de siluetas.
- Cámara: Al iniciar Blender, por defecto, aparecerá una cámara que estará en algún sitio por el centro de la ventana 3D y, probablemente, enfocando al cubo. En la figura 2.1 es esa especie de pirámide de vértices negros a la izquierda del cubo. A la hora de renderizar es necesaria una cámara.
- Objeto seleccionado actualmente: Este campo, situado en la parte inferior izquierda de la escena, al lado del eje de coordenadas, muestra el nombre del objeto seleccionado actualmente.
- Modo Edición: Este botón, en forma de desplegable, se encuentra a la izquierda de los botones de manipuladores de transformaciones 3D. Da acceso a un modo de edición para manipular la geometría del objeto. Al activarse aparecen disponibles tres botones de opciones, justo a la derecha de los botones de manipuladores de transformaciones 3D. Nos permiten cambiar entre la selección y edificación de los vértices, de las aristas y de las caras del objeto. La posibilidad de cambiar entre estas opciones de selección se presta especialmente útil a la hora de dar forma a los objetos que componen el circuito.
- Vista de la escena: Este botón, en forma de desplegable, se encuentra a la derecha del botón de Modo Edición. Da acceso a un desplegable que permite elegir la vista

de la escena 3D entre las posibles, como sólida, texturas, materiales, "wireframe", renderizada, etc. Nos permite ver los objetos de la escena con sólo las texturas, sólo los vértices, ya renderizado, etc, para poder trabajar más cómodo en segín que condiciones y ver los resultados de aplicar texturas o del renderizado final.

Para poder realizar con comodidad el trabajo sobre cualquier tipo de elemento existen diferentes vistas disponibles, cada una con un atajo de teclado propio, en concreto del teclado numérico. De esta forma para el "1" el programa proporcionara una vista frontal de la escena; con el número "3" se obtiene una vista derecha; etc... Una de las más cómodas para realizar el trabajo es la que le corresponde al número "7" ya que se trata de una vista cenital, que una vez superpuesto el plano del circuito facilitan la tarea de trazar el recorrido. Por último con la tecla "5" podemos cambiar la vista de Perspectiva a Ortográfica. La vista Perspectiva es la más similar a la realidad, dando profundidad a la escena y sensación de lejanía y proximidad de los objetos. La vista Ortográfica es más artificial, pues es como una proyección de la anterior, pero es útil en determinadas situaciones para no alterar las proporciones o manejar con fluidez objetos que quedarían tapados de otra forma.

Ésta ventana posee un botón de propiedades propio marcado en color Azul en la parte superior derecha, el cual despliega la ventana de propiedades (*Figura 2.4*). Esta ventana muestra las propiedades del objeto seleccionado y resulta muy útil a la hora de añadir nuevos bloques o realizar modificaciones de los ya existentes, pudiendo modificar la localización, rotación, escala, etc...

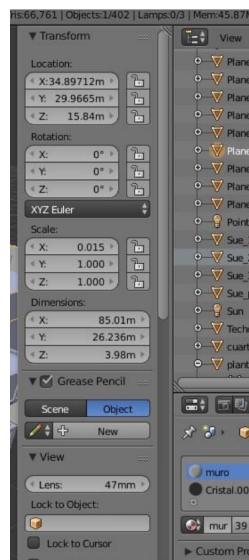


Figura 2.4: Ventana de propiedades.

Debajo de la ventana de vista 3D, bordeada en Marrón, se encuentra por defecto la

ventana de Línea Temporal. Aquí se reflejan cronológicamente los bloques u objetos que se han añadido o modificado y es muy utilizada al trabajar con animaciones. En nuestro caso la hemos sustituido por otra que se adapta mejor a nuestras necesidades.

A continuación, bordeada en Amarillo, se puede identificar la ventana de Objetos y Jerarquías, donde se pueden ver todos los datos que se utilizan en el trabajo. De esta forma se pueden controlar los diferentes bloques que se utilicen, las luces que se añaden a la escena, cámaras y toda clase de elementos disponibles en la escena. En esta ventana se pueden seleccionar directamente los elementos que se deseen independientemente, y realizar acciones sobre ellos como restringir o habilitar la visualización, selección o renderización de dicho elemento. Esto supone una gran ayuda cuando se superponen diferentes elementos en la ventana.

Debajo de ésta, marcada en color Verde, se encuentra la ventana de Propiedades, en la cual se pueden editar las propiedades de los bloques, objetos, materiales, texturas etc... que se utilizan en el trabajo. Esto se consigue mediante los llamados botones de contexto, los cuales muestran un grupo de paneles con opciones diferentes para cada botón. Gracias a esta disposición se esconden multitud de herramientas muy usadas en un espacio reducido y cómodo. Algunas de las tareas que permiten llevar a cabo son asignar el material o la textura deseada a los elementos creados o modificaciones para replicar, deformar, dividir o desdoblar elementos entre muchas otras opciones

2.2. Circuito plano

Una vez presentadas las opciones básicas de Blender y cómo desenvolverse entre ellas con un mínimo de soltura pasaremos a desarrollar el proceso de creación del circuito de Mónaco. En este proyecto comenzamos creando el trazado del circuito sin elevaciones, para coger soltura en el manejo del programa y explorar las diferentes opciones antes de realizar tareas más complejas.

Comenzamos eliminando el cubo que aparece por defecto, desplegando la ventana de propiedades del menú principal y buscando el apartado de "Background Images". Activamos la casilla y buscamos la imagen que deseamos poner de fondo, en este caso la imagen de la figura **??**. Al hacer click en el botón "Add Image" podremos seleccionar la imagen que queremos ver de fondo, así como el eje en el que la queramos ver. Esta herramienta es especialmente útil ya que nos permite ver una superposición de la imagen del trazado al usar la vista cenital, que corresponde con la tecla **"7"** del teclado numérico,

pero en cualquier otro ángulo no se verá, con lo que facilita enormemente la tarea de escalado y modelado del circuito.

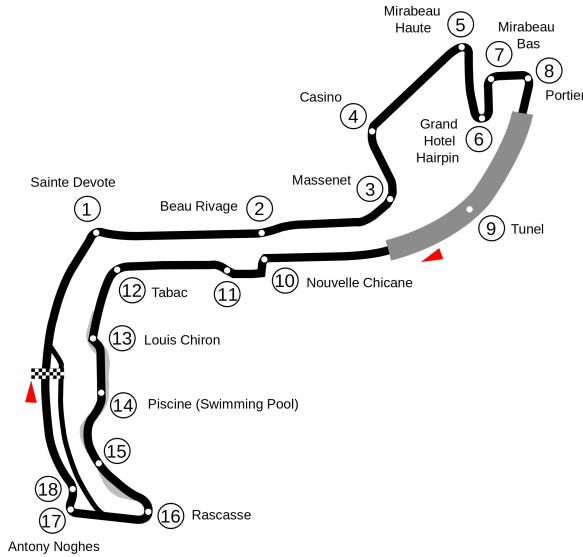


Figura 2.5: Trazado del circuito de Mónaco usado como plantilla

A continuación añadimos una curva "Bezier" (*Figura 2.6*), hacemos esto desde el menú superior en "Add", "Curve", "Bezier". Esto creará un nuevo objeto, que podemos renombrar en la ventana de Objetos para facilitar el acceso y la selección posteriormente, cuando tengamos mas objetos en la escena. Este tipo de curvas, como se puede apreciar en la figura 2.6, son unas curvas com muchos elementos que nos permiten moldearlas a nuestro gusto. En cada extremo del tramo se componen de una recta con tres puntos. el central que establece el inicio del trazo de la curva, y los otros dos que establecen el giro de la curva y lo pronunciado que és. Cuanto más cerca estén del punto central más cerrado será el ángulo de giro, y cuanto más alejados más suave la curva descrita.

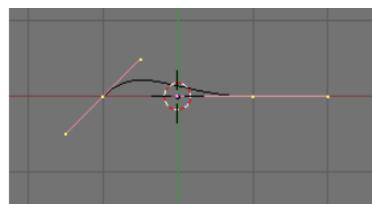


Figura 2.6: Detalle de curva Bezier.

Al hacer click con el botón izquierdo del ratón mientras mantenemos la tecla Control pulsada añadimos un nuevo punto a la curva ya existente. De esta forma podemos ir

añadiendo puntos y deformando la curva hasta que se superponga con la imagen del trazado. Así logramos una curva cerrada correspondiente al trazado sobre la cual situar los elementos que compondrán la carretera, como podemos apreciar en la Figura 2.7 (*La línea naranja es la curva Bezier que será el trazado*).

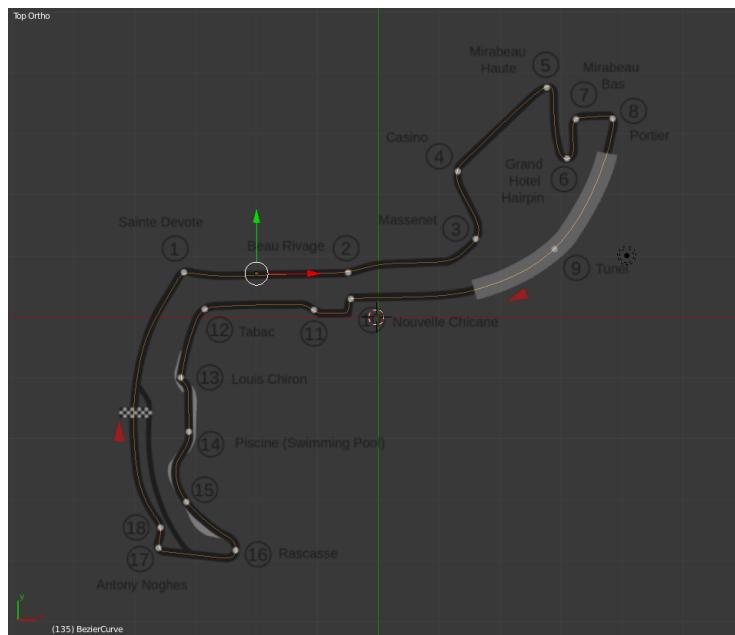


Figura 2.7: Trazado del circuito de Mónaco superpuesto a la plantilla.

A continuación añadimos un plano haciendo "Add", "Mesh", "Plane", lo cual generará un cuadrado plano en el centro de la escena. A continuación seleccionamos una arista u la extruimos (*estiramos*) dos veces, una más corta y una más larga. hacemos lo mismo con la arista opuesta del cuadrado. Es importante realizar este paso ayudándonos de las flechas de colores (*verde, rojo y azul*) que aparecen al seleccionar un objeto para mantener el conjunto en el mismo plano. Una vez realizado, extruimos los dos rectángulos más pequeños hacia arriba, consiguiendo crear la carretera, las vallas y una acera para el circuito, aunque de momento sólo aparecen en gris.

Para ayudarnos en la labor de texturizar objetos, editamos el tipo de ventana que es la de la parte inferior, el gráfico temporal, y lo sustituimos por un Editor de Imagen/UV, el cual posee herramientas avanzadas de manejo de imágenes para texturizado de superficies. De esta manera vamos cara por cara de nuestro objeto seleccionándola en la vista 3D, añadiendo una textura ¹ en el Editor UV, y observando los resultados en la vista 3D cambiando el modo de visualización a texturizado. La ventana del Editor de imagen nos

¹Las imágenes usadas como texturas y materiales en este proyecto han sido obtenidas de manera gratuita de la página textures.com[9]

sirve para redimensionar las texturas, girarlas, pintar encima de ellas, establecer qué zonas aparecen en la superficie seleccionada, y multitud de herramientas que hacen mucho más sencillo la edición de texturas. Una vez finalizado el paso de texturizar las caras del objeto obtenemos el segmento de la figura 2.8.

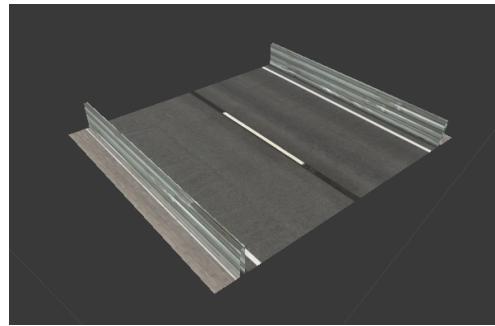


Figura 2.8: Segmento del circuito.

Es muy importante crear un material para cada textura y asignarlo a las caras correspondientes, de otra manera no se aplicará la textura al renderizar y será como haber trabajado en vano. Esto se consigue mediante la ventana de propiedades y haciendo click en la pestaña de materiales. Es necesario crear un material nuevo, añadir como fuente la misma imagen que se ha usado como textura anteriormente, renombrar para no confundirlos más adelante, y asignarlo a la superficie correspondiente. Además, en la pestaña texturas, debemos repetir los pasos para crear la textura de la misma forma que el material. Necesitaremos además asignar cada textura al material correspondiente y cambiar el tipo de mapping a UV. Es una tarea laboriosa pero muy importante, ya que de no realizarse correctamente las texturas podrían no aplicarse y el objeto aparecería gris.

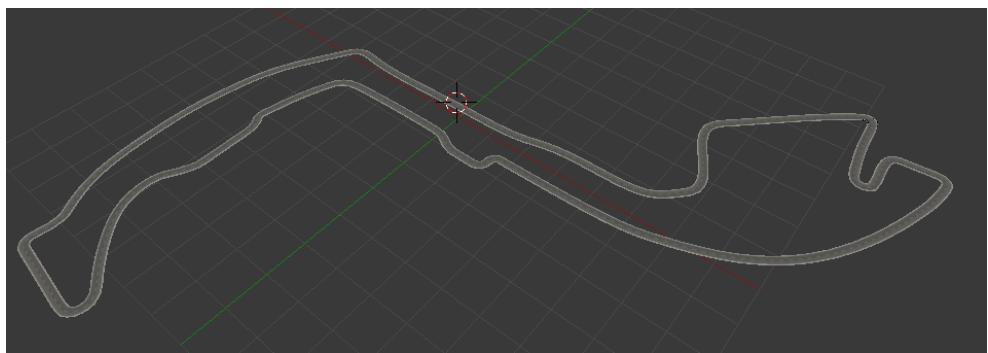


Figura 2.9: Circuito (sólo el trazado).

Una vez realizados todos estos pasos seleccionamos el segmento del trazado creado y vamos a la ventana de propiedades y hacemos click en la pestaña de modificadores. Añadimos un nuevo modificador, un Array. Con esto conseguimos crear multitud de objetos



Figura 2.10: Circuito plano.

iguales conectados entre sí, como si de un circuito de slot se tratase. Despues añadimos otro modificador, una curva en este caso, y elegimos como elemento modificante la curva Bezier que hemos creado anteriormente. De esta manera, jugando con el número de copias que realiza el array, conseguimos que el segmento creado se repita y deforme siguiendo el trazado de la curva, adquiriendo la forma del circuito deseado. Este método requiere de muchos retoques manuales, ya que en los vértices de las curvas cerradas el programa no puede calcular bien y solapa y deforma de manera incorrecta los vértices del segmento. Una vez realizados todos los retoques, así como la unión del primer y último segmento, obtenemos el objeto de la figura 2.9.

Una vez obtenido el trazado necesitamos añadir un plano que serfirá de fondo para el circuito. Para ello añadimos un nuevo objeto plano a la escena, lo aumentamos de tamaño hasta que se pueda situar el circuito en su interior y lo subdividimos en cuadrículas. Esto lo hacemos así para poder asignar texturas a cada cuadrícula independientemente y simular de manera más realista que este circuito se sitúa en un puerto, estando a la orilla del mar casi la mitad de su recorrido. Una vez realizada la división deformamos algunos vértices escondiéndolos debajo del trazado, necesitando así menos divisiones y aligerando la carga de procesamiento del circuito. Una vez asignadas todas las texturas, repitiendo los pasos descritos para texturizar el segmento del circuito, obtenemos el circuito de Mónaco2.10, el cual podemos ver en detalle en la Figura 2.11.

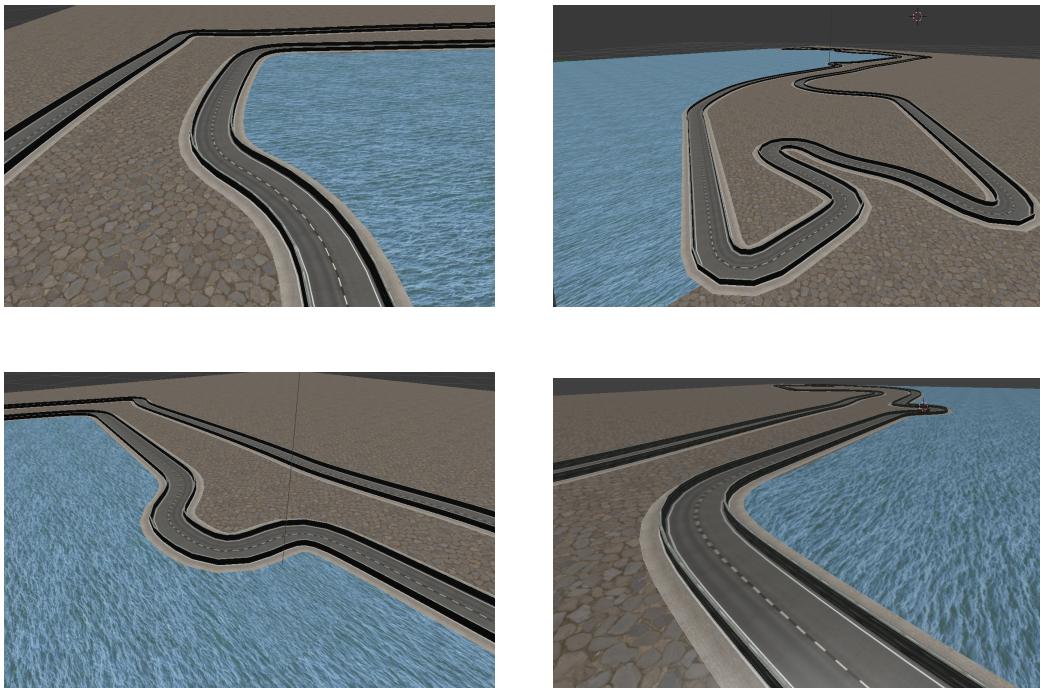


Figura 2.11: Diferentes vistas del circuito de F1 de Mónaco plano

2.3. Circuito con elevaciones

Una vez modelado el circuito plano pasamos a modelarlo ahora con elevaciones. Dada la dificultad de simplemente añadirlas al circuito ya creado, partimos de cero en la creación de este nuevo escenario. Puesto que ahora es mucho más relevante el fondo comenzamos por esa parte. Eliminamos el cubo por defecto y añadimos un nuevo objeto, ésta vez una "Mesh", "Grid", que llamaremos rejilla. Es diferente del plano por que ya está subdividido, como una rejilla. Al crearlo, en la parte inferior izquierda de la ventana de vista 3D, aparecen una serie de propiedades únicas de este objeto que modificamos a nuestro gusto, en este caso el numero de subdivisiones de la rejilla y el tamaño de ésta.

Haciendo click en el botón de Modo Edición accedemos a otro modo, en este caso de Escultura. Una vez en este modo, en la parte superior izquierda de la ventana de vista 3D se despliega una multitud de herramientas y opciones para esculpir los objetos de la escena. Como lo que nos interesa es elevar la rejilla de forma que simule las alturas y elevaciones del circuito real de Mónaco, accedemos a la opción de bloqueo y seleccionamos los ejes x e y, con lo que sólo modificaremos la altura del plano sobre el que proyectaremos el trazado. Modificando las opciones del pincel hasta dejarlo a nuestro gusto comenzamos a esculpir la rejilla. Usando la misma imagen de antes como plantilla de fondo y fotos reales del circuito esculpimos un prototipo de lo que serán las elevaciones, que más tarde retocaremos para

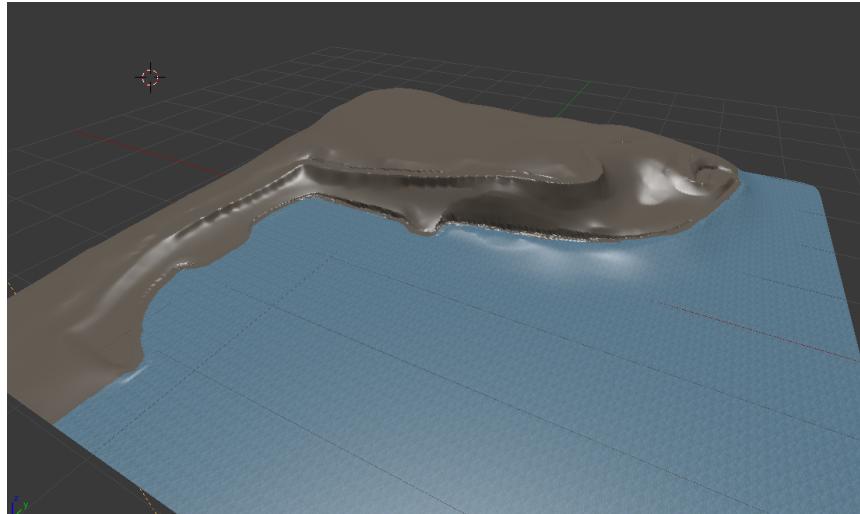


Figura 2.12: Fondo para el circuito con elevaciones.

ajustarnos mejor a las peculiaridades del trazado. También nos serviremos de otro tipo de pincel para alisar las rugosidades creadas al modelar de forma mas basta y así suavizar el terreno para acomodar mejor al circuito.

A continuación creamos de nuevo, siguiendo los mismos pasos, la curva Bezier que servirá de trazado. Creamos ahora un plano, que pintamos de un color llamativo, y extendemos a lo largo de la curva bezier, consiguiendo un trazado de referencia para acabar de dar los últimos detalles a la rejilla. Tanto a la curva como a este plano le aplicamos como modificador la rejilla, consiguiendo darles altura y pudiendo ver el recorrido real del circuito con las elevaciones modeladas. Ahora modelamos la rejilla teniendo en cuenta el recorrido del circuito y su situación a orillas del mar, por lo que tenemos mucho cuidado de mantener plana toda la parte baja y del mar y aplicamos las elevaciones en las demás partes. Una vez modelado texturizamos de forma análoga a como hicimos anteriormente. Una vez modelado y texturizado el plano, eliminamos el trazado de referencia y obtenemos el suelo sobre el que descansará el circuito (*Figura 2.12*).

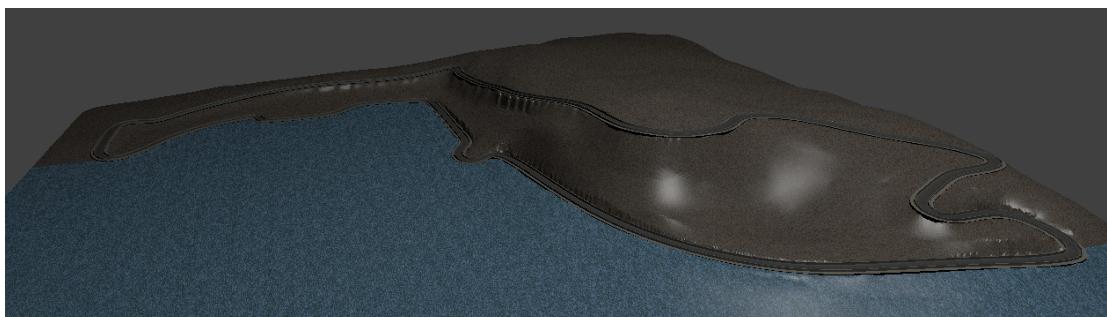


Figura 2.13: Circuito con elevaciones.

Ahora repetiremos todos los pasos de creación del trazado (creamos el segmento, lo texturizamos, lo extendemos a lo largo de la curva) y además le aplicamos la rejilla como modificador. Si hemos realizado bien el paso anterior con el trazado de referencia, el trazado final debería descansar sobre la caja realizada y acoplarse perfectamente al terreno, de no ser así retocamos nuevamente la rejilla con las herramientas de escultura.

Una vez completado este paso podemos dar por finalizado el circuito (*Figura 2.13*). En la Figura 2.14 se pueden ver detalles del circuito donde se aprecian mejor las elevaciones del terreno.

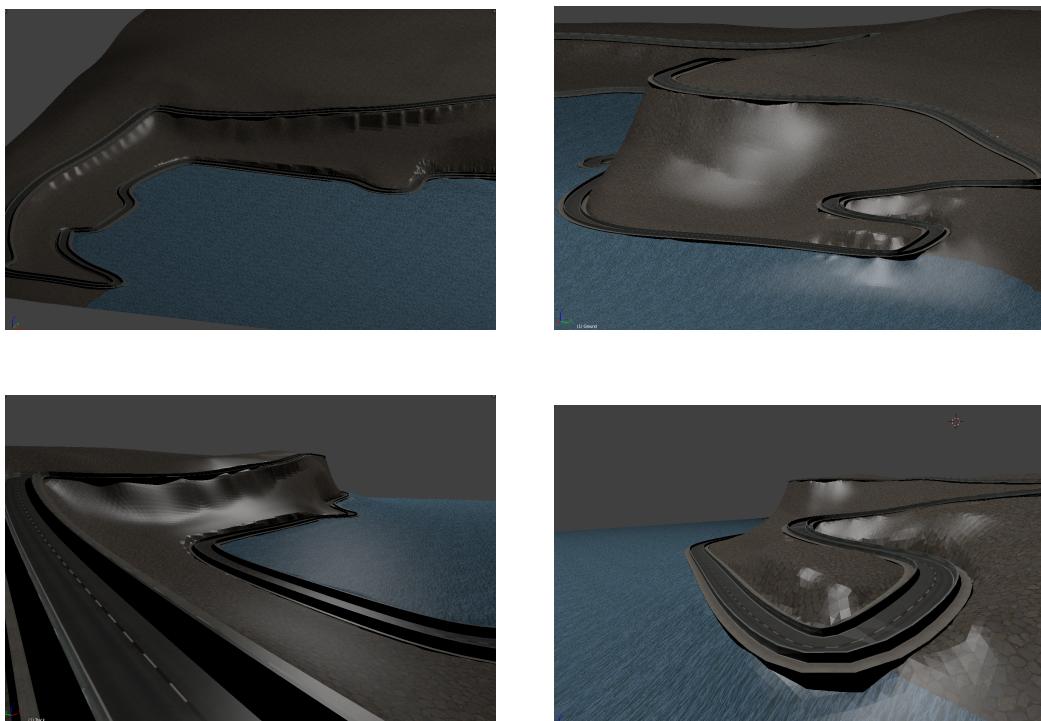


Figura 2.14: Diferentes vistas del circuito de F1 de Mónaco con elevaciones

2.4. Mundos para Gazebo

Bibliografía

- [1] Página oficial de Blender <https://www.blender.org/>
- [2] Página oficial de Gazebo <http://gazebosim.org/>
- [3] Página oficial de JdeRobot http://jderobot.org/Main_Page
- [4] Página oficial de ARIAC <http://gazebosim.org/ariac>
- [5] Página oficial de ROS <http://www.ros.org/>
- [6] Página oficial de MoveIt <http://moveit.ros.org/>
- [7] Página oficial de rViz <http://wiki.ros.org/rviz>
- [8] Página oficial de rqt <http://wiki.ros.org/rqt>
- [9] Página de texturas <https://www.textures.com/>