Escuela Técnica Superior de Ingenieros de Caminos, Canales y Puertos.
UNIVERSIDAD DE CANTABRIA

REALIDAD VIRTUAL EN LA INGENIERÍA CIVIL

VIRTUALIZACIÓN DE UNA OBRA

TRABAJO FINAL DE MASTER

Trabajo realizado por:
Marina Ojanguren Álvarez

Dirigido:
Miguel Cuartas Hernández
Valentín Arroyo Fernández

Titulación:
Máster Universitario en Ingeniería de Caminos, Canales y Puertos

Santander, septiembre de 2016

ÍNDICE

1. RESUMEN

1.1. RESUMEN

La Realidad Virtual es una simulación realista y envolvente de un entorno tridimensional creada por ordenador. Esta tecnología simula la presencia física del usuario en dicho entorno, permitiéndole interactuar con él a través de unas gafas o casco de Realidad Virtual.

La Realidad Virtual ha generado mucho entusiasmo en el campo de los videojuegos, pero sus aplicaciones pueden alcanzar numerosas disciplinas. Cualquier profesión que utilice modelado 3D puede beneficiarse de esta tecnología ya que la sensación de inmersión que genera puede ser muy útil.

Actualmente en el campo de la ingeniería civil se está implementando la gestión de proyectos a través del BIM (Building Information Modeling) que se basa en realizar proyectos mediante un modelo tridimensional digital que abarca la geometría, las relaciones espaciales, la información geográfica, así como las cantidades y las propiedades de sus componentes.

Lo más destacado de estos modelos es su tridimensionalidad y, por tanto, es lógico que la forma de interactuar con ellos sea a través de interfaces 3D, es decir, la Realidad Virtual ofrece cambiar el interfaz con el que acercarse a los modelos BIM para aprovechar todo su potencial. Por ello, la Realidad Virtual se presenta como una tecnología con un gran futuro para el intercambio de información entre los diferentes integrantes de un proyecto y una herramienta de ayuda al diseño.

En este estado de desarrollo se enmarca el presente proyecto, con objeto de explorar las posibles aplicaciones de la Realidad Virtual en la ingeniería civil llevando a cabo un ejemplo práctico de virtualización de una obra.

En la búsqueda de la herramienta que permitiese interactuar con modelos tridimensionales a través de un dispositivo de Realidad Virtual, se ha escogido Unity 3D como plataforma de desarrollo del proyecto, ya que simplifica la tarea de trasladar modelos CAD, BIM o cualquier modelo tridimensional, a un entorno virtual. Y no sólo eso, sino que además Unity 3D permite generar una aplicación para numerosas plataformas objetivo entre las que se encuentran las gafas o cascos de Realidad Virtual más populares (Oculus Rift, HTC Vive, PlayStation VR, Samsung Gear VR).

Por último, el presente proyecto profundiza en el proceso de virtualización de una obra para que todo profesional del sector, principiante en esta nueva tecnología, disponga de los recursos necesarios para crear una aplicación orientada al uso de dispositivos de Realidad Virtual. Entre las características del modelo destaca la posibilidad de importar el terreno real a partir de datos de Instituto Geográfico Nacional, así como la posibilidad de interactuar con los objetos que forman parte de él.

Para la interacción con el entorno, Unity permite programar el comportamiento de los objetos que forman parte del mismo. En el presente proyecto se han creado scripts, en el lenguaje de programación C#, mediante los cuales se añaden instrucciones que definen el comportamiento de los objetos. Tras estudiar diversas opciones se ha optado por desarrollar dos formas intuitivas de interactuar con los objetos en entonos adaptados a la Realidad Virtual, a saber, acercándonos a ellos o mirándolos.

Finalmente, se ha creado una aplicación para PC que soporta el uso de dispositivos de Realidad Virtual, en la que se muestra la obra modelizada a través de un personaje en primera persona que puede moverse alrededor del entorno virtual e interactuar con los diferentes objetos que lo conforman.

1.2. ABSTRACT

Virtual Reality is a realistic and immersive simulation of a three-dimensional environment computer-generated. This technology simulates the user physical presence and it also allows to interact with the environment helped with a HMD (Head Mounted Display).

Virtual Reality has generated a huge excitement in the field of video games, but it could be used in many different disciplines. Any profession that uses 3D modelling can be benefited from this technology thanks to the feeling of immersion that Virtual Reality allows.

Nowadays, in the field of civil engineering it has been implemented the BIM (Building Information Modelling) which is a project management methodology through a digital three-dimensional model that covers geometry, spatial relationships, geographic information, and the quantities and properties of its components.

The highlight of these models is their three-dimensionality and, therefore, it is logical that the way to interact with them should be through 3D interfaces. That is to say, the Virtual Reality offers an interface change to approach the BIM models which permits to exploit its full potential. Therefore, the Virtual Reality presents itself as a technology with great potential for the exchange of information between the different members of a project and as design support tool.

It should be noted that, as a developing technology, there are limitations in its application and the availability of equipment is still insufficient in order to make the Virtual Reality technology affordable to all professionals. In this stage of development, the aim of this project is to explore the possible applications of Virtual Reality in the civil engineering field undertaking a practical example of making a virtualization of a real work.

Searching for a tool which allows to interact with three-dimensional models through a Virtual Reality device is has been chosen Unity 3D development platform because it extraordinarily simplifies the task of import CAD, BIM models or any three-dimensional model to a virtual environment. Not only that, but it also has numerous target platforms among the most popular HMD (Oculus Rift, HTC Vive, PlayStation VR, Samsung Gear VR).

Finally, this project explores the virtualization of a work pursuing the goal of creating a guide for all the professionals who wants to learn about Virtual Reality and the possibilities that it offers. Among the features of the model highlights the possibility to import a real terrain data from the National Geographic Institute and the ability to interact with the objects that are part of the model.

In order to create a way of interaction with the environment, Unity allows to program the behaviour of the objects that are part of it. Scripts has been created, in the programming language C #, using them to add instructions that will define the behaviour of objects. After studying various options, it has been chosen to develop two intuitive ways to interact with objects in Virtual Reality environments, by approaching them or by looking at them.

Finally, it has been created a PC application that supports the use of Virtual Reality devices in which the work is displayed through a character in first person who can move around the virtual environment and interact with different objects.

2. INTRODUCCIÓN

2.1. CONTEXTO

La Realidad Virtual se basa en la creación de un entorno tridimensional modelado digitalmente. Por otra parte, existe la Realidad Aumentada, cuya característica principal es que a un entorno u objeto real se le añade información digital, es decir, la Realidad Aumentada crea una "realidad mixta" entre elementos creados digitalmente y elementos reales.

El nacimiento de la Realidad Virtual ha sido posible gracias a dos grandes avances producidos en el mundo de los teléfonos móviles: el desarrollo de pantallas ligeras, baratas y de alta resolución, así como el de una nueva generación de sensores de movimiento de gran precisión.

Estos avances han logrado que sea posible la creación de las gafas o cascos de Realidad Virtual que han permitido pasar de mostrar los gráficos a través de la pantalla del ordenador a reproducirlos en unas gafas que crean la ilusión de que lo que estás viendo a través de ellas es real. Éstos dispositivos permiten observar desde todas las perspectivas simplemente moviendo la cabeza. Para desplazarse por la escena puede usarse mandos o sensores de movimiento, todo ello logra que la persona se sumerja completamente en el entorno virtual.

La Realidad Virtual ha generado mucho entusiasmo en el campo de los videojuegos, pero sus aplicaciones pueden alcanzar numerosas disciplinas. Cualquier profesión que utilice modelado 3D puede beneficiarse del uso de la Realidad Virtual ya que la sensación de inmersión puede ser muy útil para campos tan diversos como la educación, la ingeniería, el marketing, el entretenimiento o el desarrollo de negocios.

Dentro de la ingeniería existen varios ejemplos donde esta tecnología puede ser aplicada como:

- *La ingeniería mecánica y diseño industrial:* En el diseño asistido por ordenador como AutoCAD o SOLIDWORKS, con la Realidad Virtual, ingenieros y diseñadores pueden experimentar la sensación de tener en sus manos el producto antes de producirlo y simular diferentes escenarios a un precio reducido.

- *La ingeniería civil y arquitectura*: Los arquitectos e ingenieros siempre han construido modelos a escala de sus diseños, tanto para mostrar y vender sus diseños a los clientes como para validar las hipótesis asumidas durante la fase de diseño. Hoy en día, el software de modelado y renderizado es ampliamente utilizado para construir modelos virtuales. La Realidad Virtual facilita la negociación con los clientes o inversores, permitiendo experimentar un diseño antes de construirlo.

- *La simulación de actividades:* otra posible aplicación dentro del mundo de la ingeniería es la simulación de procesos complejos, como un simulador de vuelo para pilotos. La Realidad Virtual permite simular el proceso de vuelo y con ello el personal responsable de la operación puede entrenarse para realizar su tarea de forma barata, simple y segura. Otro ejemplo de aplicación dentro de la ingeniería puede ser la simulación del proceso de mantenimiento de una central nuclear donde el personal tiene una sola oportunidad para desarrollar su actividad y el tiempo de realización de dicha actividad es crucial.

2.2. CÓMO FUNCIONAN LOS DISPOSITIVOS DE REALIDAD VIRTUAL

Un casco de Realidad Virtual, también llamado gafas de Realidad Virtual o HMD (del inglés Head-Mounted Display), es un dispositivo de visualización que permite reproducir imágenes creadas por ordenador sobre una pantalla muy cercana a los ojos.

Debido a su proximidad con los ojos el casco de Realidad Virtual consigue que las imágenes visualizadas resulten mucho mayores que las percibidas por pantallas normales, y permiten incluso englobar todo el campo de visión del usuario. Gracias a que el casco se encuentra sujeto a la cabeza, éste puede seguir los movimientos del usuario, consiguiendo así que se sienta integrado en los ambientes creados por ordenador.

En ocasiones las gafas están acompañadas de guantes con sensores, mandos con control de movimiento o cámaras de posicionamiento que permiten hacer cosas como andar dentro del escenario o tocar los objetos virtuales.

Hoy en día, las gafas de Realidad Virtual no incorporan el procesador, es decir, para disfrutar de la Realidad Virtual se necesitan dos dispositivos: las gafas y el dispositivo que genera el entorno virtual como puede ser un ordenador o una videoconsola. Para el presente proyecto se ha contado con el set de Oculus Rift que cuentan con un casco con dos lentes y unos auriculares, un mando inalámbrico y un rastreador de posición.

El funcionamiento es sencillo, las gafas están formadas por una pantalla y unas lentes. Las lentes amplían el ángulo de visión, generando la sensación de que la pantalla abarca todo el espectro visual del usuario. Por otro lado, el dispositivo informático genera dos imágenes diferentes, una para cada ojo, produciendo un efecto 3D. Esto es posible gracias a la visión estereoscópica, es decir, la percepción de dos imágenes ligeramente diferentes, una en cada ojo, como una sola imagen. Dando lugar a la percepción de profundidad, que permite ver el mundo en tres dimensiones.

Para recrear esto, es necesario generar dos puntos de vista ligeramente diferentes, uno recibido por los ojos izquierdo y otro por el derecho cuando el usuario está utilizando el HMD.

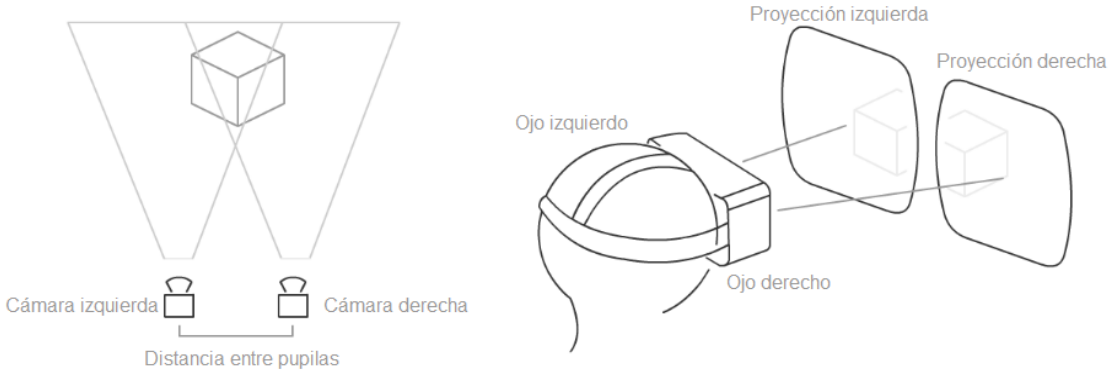

Figura 1. Cómo recrear imágenes estereoscópicas en 3D[1]

[1] (steveluscher, 2005-2016)

2.3. LA INGENIERÍA CIVIL Y LA REALIDAD VIRTUAL

Los modelos relacionados con la construcción necesitan ser capaces de generar cambios en la geometría del proyecto. La integración de dicha geometría junto con el plan del proceso constructivo son la base de los modelos en 4D (3D + tiempo) que junto a la Realidad Virtual permiten una representación muy realista de la obra y su construcción.

Los modelos en 4D tiene numerosas aplicaciones como la mejora de la productividad, el análisis, el diseño y la coordinación del proyecto. En este contexto la Realidad Virtual permitiría a los usuarios observar cada fase del proyecto para analizarla y manipularla tridimensionalmente de forma interactiva y en tiempo real. Por ello, se presenta como una tecnología con un gran potencial para el intercambio de información entre los diferentes integrantes de un proyecto y una herramienta de ayuda al diseño.

El BIM (Building Information Modeling) es una metodología de trabajo para la gestión de proyectos de edificación u obra civil a través de una maqueta digital. Esta maqueta digital conforma el modelo de información del proyecto que abarca la geometría, las relaciones espaciales, la información geográfica, así como las cantidades y las propiedades de sus componentes.

Actualmente, las empresas y profesionales españoles que quieran participar en proyectos de construcción, reforma, instalación, y explotación en lugares como Reino Unido, EE.UU., Emiratos Árabes, China o Australia tienen que implantar el modelo BIM de manera obligatoria para acceder a licitaciones, contratos y colaboraciones. Y no sólo eso, sino que el Ministerio de Fomento ha marcado para el año 2018 como fecha límite para implantar, en las fases de diseño y construcción, el uso de un modelo BIM para las licitaciones de equipamientos e infraestructuras públicas de presupuesto superior a 2 millones de euros.

Por lo que se puede afirmar con seguridad que en un futuro próximo toda obra de envergadura tendrá su modelo BIM. Lo que abre las puertas a un mercado creciente en todo lo relacionado con el modelado tridimensional.

Existen varias finalidades para la creación de un modelo BIM que pueden diferir tanto en su enfoque, como en su alcance, complejidad, nivel de detalle y profundidad de la información incorporada al modelo 3D. Entre dichas finalidades se encuentra la visualización de la obra que se va a construir, que ofrece una ayuda para la toma de decisión en el proceso de diseño mediante la comparación de diferentes alternativas, así como la posibilidad de vender el diseño al cliente o a los ciudadanos, que podría tener un voto sobre el proyecto. [2] A su vez ofrece la integración de diferentes elementos como la estructura, las instalaciones, etc. en un mismo modelo y proporciona información sobre la planificación temporal de los trabajos, o su coste. Por otro lado, ofrece la posibilidad de mostrar visiones parciales del modelo para la formación de los equipos que van a participar en los trabajos constructivos.

Así, lo más destacado de estos modelos es su tridimensionalidad y, por tanto, es lógico que la forma de interactuar con ellos sea a través interfaces 3D. Es decir, la Realidad Virtual ofrece cambiar el interfaz con el que acercarse a los modelos BIM para aprovechar toda su potencialidad.

[2] (GRAPHISOFT, 2016)

3. UNITY 3D

3.1. INTRODUCCIÓN

Trasladar modelos CAD o BIM a un entorno virtual que permitiese el uso de la Realidad Virtual solía precisar de una gran cantidad de tiempo y de amplios conocimientos de programación. La plataforma para la creación de aplicaciones Unity 3D, principalmente enfocado al desarrollo de videojuegos, permite importar modelos BIM a entornos virtuales simplificando mucho el proceso necesario para virtualizar una obra e interactuar con ella en 3D. Además, Unity 3D, permite desarrollar contenidos interactivos en tres dimensiones de manera gratuita y accesible y por ello se ha escogido como plataforma para desarrollar el proyecto.

Existen diferentes modelos de licencia de Unity, la plataforma ofrece una versión de pago (Unity Pro) así como una versión gratuita (Unity Personal). La versión gratuita permite desarrollar y publicar proyectos, con algunas restricciones o funciones no disponibles. En el contexto de investigación y desarrollo para la Ingeniería Civil del presente proyecto la versión gratuita Unity Personal satisface todas las necesidades[3].

A continuación, se indican algunas de las características más destacadas del motor:

- Gran diversidad de plataformas objetivo: Windows, iOS, Android, WebGL, Oculus Rift, HTC Vive, PlayStation VR, Samsung Gear VR, ect.

- Soporte de texturas 3D y mapeado de relieve.

- Tienda con paquetes gratuitos en una amplia gama de categorías, incluyendo modelos 3D, texturas y materiales, sistemas de partículas, música, efectos de sonido, tutoriales, proyectos y paquetes de scripts entre otros.

- El programa ofrece integración nativa con Visual Studio. Unity admite como leguajes de programación el C# y el JavaScript. Además, ofrece todas las prestaciones con path finding y mallas de navegación avanzadas y automatizadas.

3.2. INTRODUCCIÓN AL MANEJO DE UNITY 3D

El presente aparatado tiene como objetivo introducir Unity 3D para poder explicar el alcance de la plataforma. A mismo tiempo ha de servir como base para el ejemplo de virtualización de la obra objeto de este proyecto.

3.2.1. El editor

Cuando se ejecuta el programa, creando un proyecto nuevo, la interfaz aparece dividida en diferentes ventanas, principalmente son:

- *Scene (escena):* en esta ventana es donde se gestiona el proyecto. Permite manipular los elementos para realizar ciertas acciones sobre ellos como puede ser: mover, redimensionar o rotar entre otros.

- Game: muestra el resultado de la ejecución de la escena actual. Para ejecutar nuestro proyecto es necesario pulsar el botón de reproducir situado en la parte superior de la pantalla.

[3] (Unity Technologies - Unity Store, 2016)

- *Consol*e: en esta ventana se muestran los mensajes que proporciona el editor, avisa de problemas que puedan estar ocurriendo durante la creación del proyecto.

- *Hierarchy (árbol):* en esta ventana se muestran todos los objetos presentes en la escena. Cuando se selecciona un objeto en el árbol éste aparece marcado en la escena lo que nos permite interactuar con él. Del mismo modo sus componentes aparecerán en la ventana Inspector.

- *Inspector:* es la ventana más importante del editor. En ella se muestran los detalles de todos los componentes del objeto que se tenga seleccionado.

- Project: en esta ventana se los recursos que han sido incluidos en nuestro proyecto, organizados en carpetas.

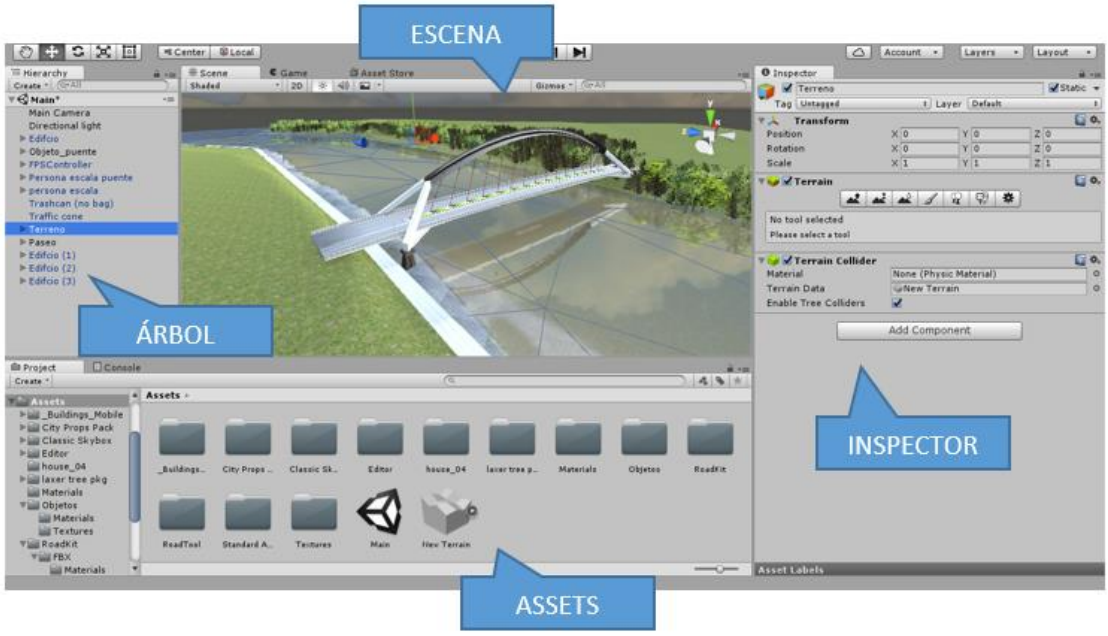

Figura 2. Interfaz Unity 3D

La interfaz del editor de Unity es personalizable y pueden añadirse cuántas ventanas se necesiten simplemente acudiendo a la pestaña en Menú > Window. Así mismo la disposición de las ventanas puede adaptarse a los gustos del usuario.

3.2.1.1. Assets o recursos

El proyecto contiene una serie de assets o recursos, principalmente son:

- *Escenas:* Es el recurso principal, como su nombre indica en cada una de ellas se guarda la escena, es decir, la representación de un modelo, o fase del mismo.

- *Materiales:* Definen el aspecto que se le puede dar a una malla (textura, color, sombreado, etc.).

- *Scripts:* Permiten personalizar el comportamiento de los objetos mediante programación.

- Clips de audio: Música y efectos de sonido.

La carpeta principal del proyecto se llamada Assets y se crea automáticamente al originar un proyecto nuevo. Esa carpeta contendrá tanto las escenas de nuestro proyecto como los recursos que forman parte de él.

En el interfaz del programa se pueden ver los recursos del proyecto en una pestaña llamada “Assets” dentro de la ventana “Project”, en la figura 2 ésta pestaña se localiza en la parte inferior de la pantalla. Es muy importante mantener el sistema de carpetas organizado ya que a medida que se crea el proyecto y se importan recursos puede ser difícil encontrar los elementos que queramos utilizar. Para crear una carpeta basta con situarse en la ventana Assets, hacer clic en el botón derecho y escoger la opción Create< Folder.

3.2.1.2. Árbol

En Unity 3D cada modelo se almacena en un asset de tipo escena. Cada escena tiene un árbol donde aparecen todos los objetos que forman la escena. Cuando se selecciona un objeto en el árbol éste aparece marcado en la escena lo que permite interactuar con él. Del mismo modo, sus componentes aparecerán en la pestaña Inspector.

En el interfaz del programa podemos ver el árbol dentro de la pestaña “Hierarchy”, en la figura 2 de ésta pestaña se localiza en el lado izquierdo de la pantalla.

3.2.1.3. Inspector

Así mismo, si seleccionamos un objeto del árbol podemos observar sus componentes en la pestaña “Inspector”.

Estas componentes son las características del objeto que engloban un sinfín de propiedades que podemos añadir al objeto entre los que destacan:

- Características geométricas: posición, rotación y escala.

- Mallado que conforma y delimita el objeto tridimensionalmente, incluyendo el tipo de material o la textura.

- Características físicas: el tipo de interacción con otros objetos.

- Scripts para controlar el funcionamiento del objeto.

3.2.2. Objetos

Unity 3D permite crear distintos tipos de objetos (Game Objects) predefinidos en el menú GameObject. Estos objetos predefinidos llevan de forma predeterminada una serie de componentes, entre ellos destacan:

- *Objetos 3D*: Estos objetos permiten crear nodos con formas geométricas básicas, es decir, planos, cubos, esferas, cilindros, etc. Incorporan una serie de componentes básicas: posición, orientación, material, malla geométrica y geometría de colisión. En caso de que se quiera añadir al entorno objetos con características geométricas más complejas lo recomendable es acudir a un

programa de modelado tridimensional como puede ser Sketch Up o Blender para posteriormente importar el objeto a Unity.

- *Empty Game Object*: es un objeto vacío que sólo tiene las componentes de posición y orientación. Son muy útiles para agrupar objetos.

- *Cámara*: es un objeto con un componente cámara que sirve para visualizar la escena. Un personaje en primera persona tendrá entre sus componentes una cámara que permitirá observar la escena como si fuera a través de sus ojos.

- *Luz*: es un objeto con un componente Light que se comportará como una fuente de luz para nuestra escena.

- Objetos UI (User Interface): permiten al creador de la escena crear interfaces de usuario. Al mismo tiempo permiten al usuario de la escena o aplicación comunicarse, obtener información a través de dicha interface y modificar el estado del modelo. Un ejemplo podría ser un texto, una imagen o un botón.

3.2.3. Herramientas básicas

Para moverse por la escena 3D e interactuar con los objetos se pueden utilizar los botones de navegación que se muestran en la figura 3.

- Mano: sirve para desplazarse por la escena.
    - +alt. Sirve para mover el punto de vista de la escena (rotación de la vista).
    - +ctrl. sirve para hacer zoom.

- Mover: sirve para mover un objeto sobre alguno de los ejes.

- Rotar: sirve para rotar un objeto sobre alguno de los ejes.

- Escalar: sirve para escalar un objeto sobre alguno de los ejes.

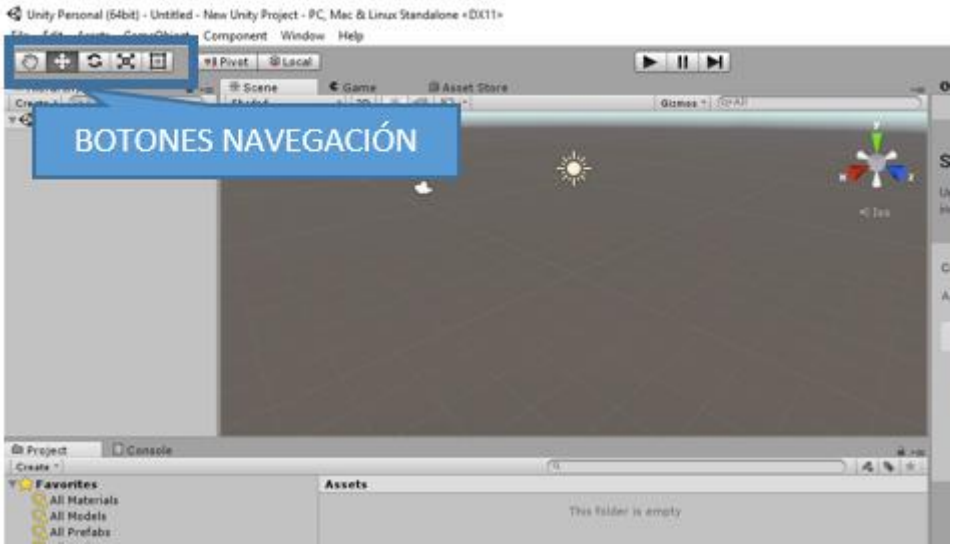

Figura 3. Botones de navegación

3.2.4. Reproducción de la escena

En la parte superior de la pantalla del editor hay tres botones para ejecutar, pausar o avanzar en la escena. Esto permite observar el comportamiento que tendrá finalmente la aplicación que se está editando. Es recomendable ejecutar la escena a medida que se va creando para detectar posibles errores.

Una vez terminada la escena, Unity permite exportarla como una aplicación ejecutable, página web, etc., para que cualquier usuario pueda emplearla.

3.2.5. Los Scripts

Los scripts son un tipo de recurso que permite personalizar el comportamiento de los objetos mediante programación. Se añaden como componente a los objetos. Cuando un Game Object tiene un componente script, este código se ejecuta vinculado al objeto al que se le ha añadido. Más adelante se introducirá la programación en Unity y se crearán varios ejemplos de este tipo de recursos

3.2.6. Paquetes de recursos

Unity incorpora una serie de recursos o “assets” predefinidos. Se puede acceder a ellos para importarlos a la escena a través de la pestaña Menú > Assets > Import Package. Algunos ejemplos de estos recursos son paquetes de vegetación, de agua, de cielos o incluso de personajes.

Por otra parte, Unity tiene una librería de recursos online en la que los desarrolladores ofrecen los recursos que han creado. Esta librería se llama Asset Store y se puede acceder a ella para descargar e importar a la escena los recursos que nos puedan resultar útiles a través de la pestaña Window > Asset Store. Algunos de los recursos ofrecidos por los desarrolladores son de pago, aunque existe una gran cantidad de ellos que son gratuitos.

Ambas fuentes de recursos o “assets” simplifican enormemente la modelización de un entorno virtual. Para el alcance pretendido en el presente proyecto eliminan una gran cantidad de tiempo en la creación de objetos como pueden ser árboles o personajes, que nos son objeto principal de la virtualización de una obra, y que, sin embargo, llevaría mucho tiempo crearlos.

3.3. IMPORTAR MODELOS 3D A UNITY

Una de las grandes ventajas a la hora de virtualizar una obra civil es que Unity soporta la importación de modelos desde las aplicaciones 3D más populares, como por ejemplo Blender o SketchUp así como también desde los programas de modelado BIM como Revit.

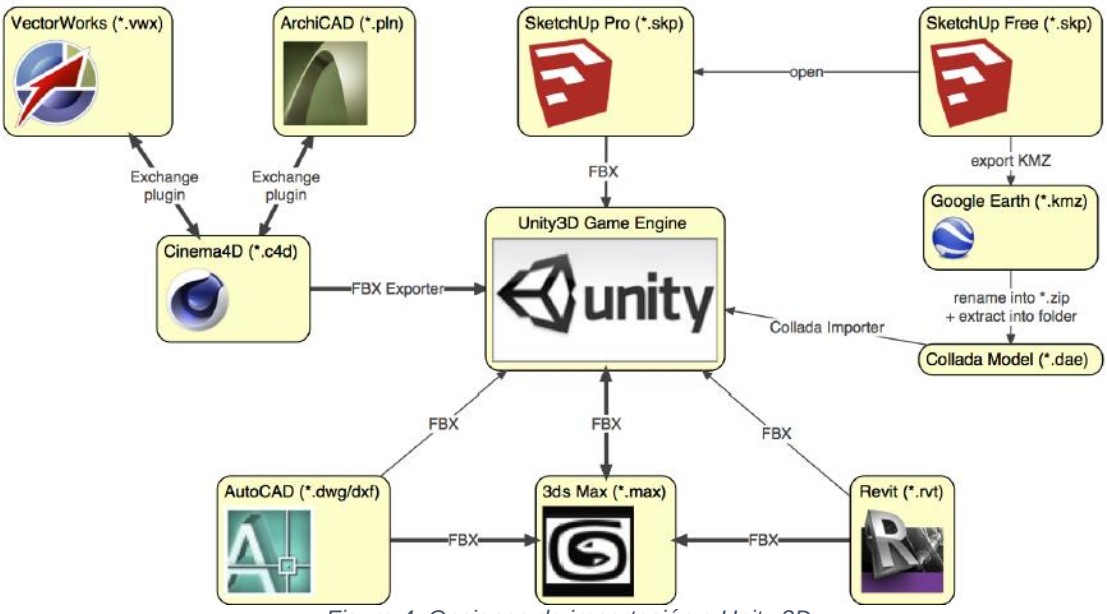

Figura 4. Opciones de importación a Unity 3D

Unity puede leer directamente archivos “.FBX”, “.dae”, “.3DS”, “.dxf” y “.obj”.

Ventajas:

- Solo se exportan los datos que uno necesita.
- Los datos son verificables (se re-importa en el paquete 3D antes que en Unity).
- Generalmente son archivos más pequeños.
- Fomenta un enfoque modular - p. ej. diferentes componentes para tipos de colisión o interactividad.
- Soporta otros paquetes 3D para cuyo formato Unity no tiene soporte directo.

Desventajas:

- Puede ser un flujo de trabajo más lento para la creación de prototipos e iteraciones.
- Es más fácil perder la pista de las versiones entre la fuente (archivo de trabajo) y los datos del juego (por ejemplo, un FBX exportado).

Unity también puede importar mediante conversión archivos, como: Max, Maya, Blender, Cinema4D, Modo, Lightwave &Cheetah3D, etc.

Ventajas:

- Proceso rápido de iteración (al guardar el archivo fuente, Unity re-importa automáticamente).
- Simple de usar al principio.

Desventajas:

- Los archivos pueden llenarse de datos innecesarios.
- Los archivos grandes pueden ralentizar las actualizaciones de Unity.
- Menos validación, por lo tanto, es más difícil solucionar problemas.

3.4. UNITY 3D Y LA REALIDAD VIRTUAL

La Realidad Virtual de Unity (en adelante VR) permite tener como objetivo diferentes dispositivos de Realidad Virtual sin la necesidad de tener plugins externos. Cuando se activa el VR en Unity, se automatizan varios procesos como:

La visualización estereoscópica automática

- Gracias al VR de Unity no se requiere tener dos cámaras para visualizaciones estereoscopias. Las matrices de vista y proyección se ajustan para tener en cuenta el campo de visión.

- Las optimizaciones ocurren automáticamente para que sea menos costoso dibujar frames dos veces (uno para cada ojo).

Headtracked Automático

- Se aplica automáticamente el seguimiento de la cabeza (Head Tracking) y se adapta el campo de visión.

Hay que tener en cuenta que si se quiere modelizar un entorno para el uso de dispositivos de Realidad Virtual la manera con la que se interactúa con el entorno varía.

A continuación, se expondrá el proceso realizado para virtualizar una obra e interactuar con ella en tres dimensiones.

4. VIRTUALIZACIÓN DE UNA OBRA: EJEMPLO PRÁCTICO

4.1. INTRODUCCIÓN

La finalidad del ejemplo práctico es realizar una guía para importar un modelo BIM de un proyecto a un entorno virtual introduciendo la posibilidad de relacionarnos con nuestro modelo en tres dimensiones a través de la Realidad Virtual.

El principal objetivo es demostrar que se trata de un proceso sencillo, al alcance de cualquier profesional del sector, una vez que se ha realizado el modelado tridimensional del proyecto.

Por ello, evitando extendernos en la creación de un modelo tridimensional de una estructura, se ha escogido una obra emblemática de la que existen infinidad de modelos gratuitos en la red.

4.2. LA ESTRUCTRA

Como ejemplo práctico se ha escogido el Puente de la Barqueta, localizado en Sevilla, diseñado por Juan José Arenas de Pablo y Marcos Jesús Pantaleón Prieto y construido entre 1989 y 1992 para permitir el acceso al recinto de la Exposición Universal de 1992.

El puente se compone de un arco de acero de 214 metros cuyos extremos forman un pórtico triangular en cada lado, atirantado por el propio tablero que tiene una longitud de 168 metros cuyos únicos apoyos son cuatro soportes verticales a una distancia de 30 metros sobre las orillas del río Guadalquivir.

4.2.1. 3D Warehouse e importación a Unity desde Sketch Up

Para la búsqueda de un modelo se ha optado por la página web “3D Warehouse[4]”, propiedad de Sketch Up, ya que es la biblioteca digital de modelos tridimensionales más grande del mundo.

Figura 5. Modelo 3D de la estructura. Se ha seleccionado el modelo realizado por Andrés D.L.

Para importar el modelo desde Sketch Up a Unity primero es necesario abrir el modelo con *Google Sketch Up*. Una vez abierto se exporta en formato *.dae* (*Default Settings*).

[4] Disponible a [05/09/2016] https://3dwarehouse.sketchup.com

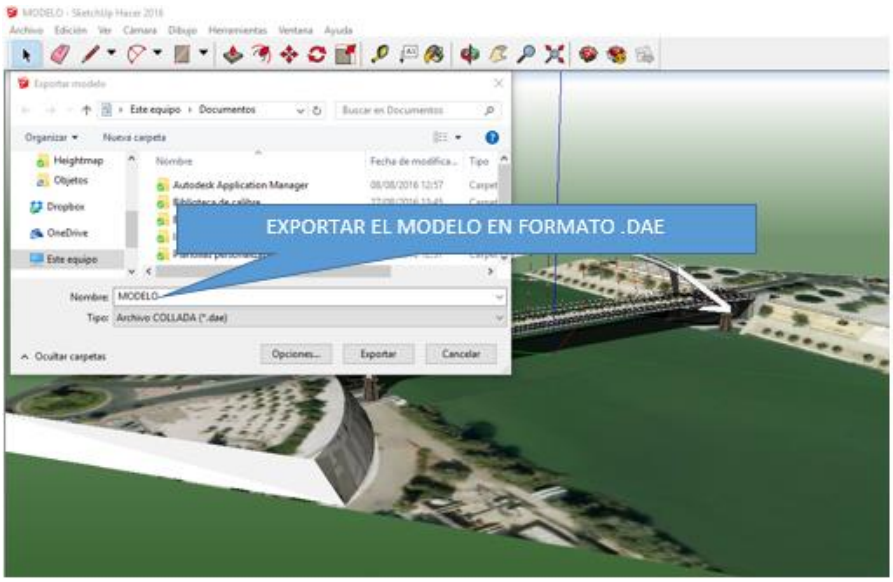

Figura 6. Exportar el modelo en formato .dae en Sketch Up

Tras haberlo exportado abrimos el archivo *.dae* generado con Blender y lo exportamos a formato *.fbx*. Si el modelo se ha realizado con Revit o algún otro programa de BIM o si se ha usado Sketch Up Pro no sería necesario realizar este paso ya que se podría exportar directamente a un formato .fbx sin necesidad de utilizar Blender.

También existe la opción de insertar el modelo de Sketch Up directamente ya que Unity admite ficheros en formato .dae pero se crearán varios objetos vacíos y será más difícil de trabajar con el modelo.

Una vez se tenga el modelo en formato .fbx bien sea porque lo hayamos creado directamente en ese formato o lo hayamos exportado se recomienda crear una carpeta llamada Texturas haciendo clic con el botón izquierdo del ratón dentro de la carpeta Assets acudiendo la pestaña Create > Folder pata tener organizado nuestro modelo. Una vez creada la carpeta se arrastra o importan las texturas del modelo dentro de ella. El proceso es el mismo para el modelo *.fbx*, se crea una carpeta llamada Estructuras dentro de Assets y se arrastra o importa el modelo.

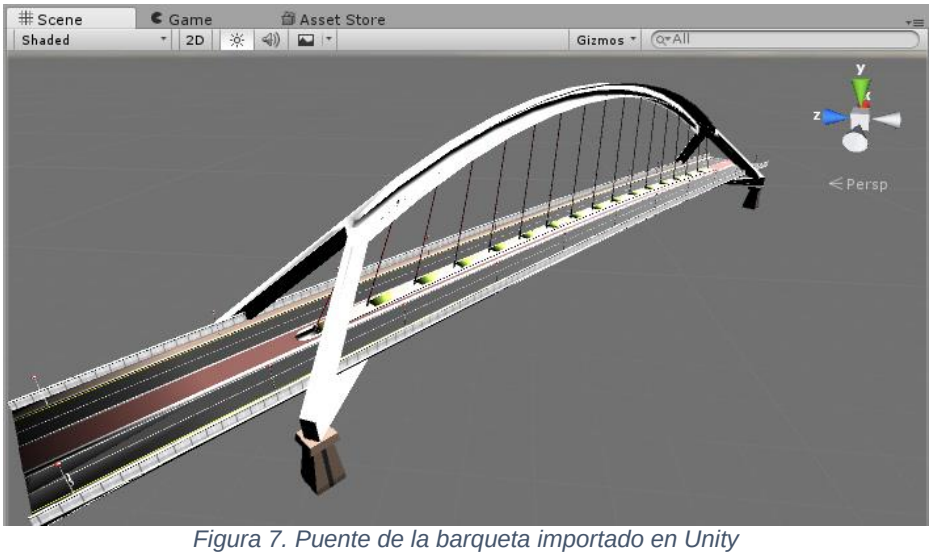

Figura 7. Puente de la barqueta importado en Unity

4.3. CREAR UN TERRENO REAL EN UNITY 3D

Unity es capaz de generar un terreno a partir de un heightmap (en ficheros tipo ".raw") que es un fichero de tipo imagen que representa las elevaciones del terreno en donde cada pixel representa la elevación del terreno en ese punto a través de su color (escala de grises). Un heightmap puede ser de 8-bits y representar altitudes de 0 a 256 o bien de 16-bits y representar alturas con los valores de 0 a 65536.

4.3.1. Obtención de datos para representar un terreno real

Para obtener datos reales que definan el terreno en el territorio español se tiene la información que pone a disposición el Instituto Geográfico Nacional (IGN) con la base de datos *MDT05/MDT05-LIDAR*[5].

Esta base de datos es un modelo digital del terreno (MDT) que contiene la cota de todos los puntos en una cuadrícula de 5 metros de paso de malla. Las coordenadas usadas para referenciar dichos puntos son coordenadas ETRS89. Este MDT se distribuye troceado según la distribución oficial de hojas 1:25.000.

A la hora de descargar el MDT es necesario localizar cuál es la hoja en la que se localiza la obra. Para localizar dicha hoja, se puede utilizar la Búsqueda en visor en la propia página de descargas del IGN que muestra un mapa de España con la distribución en hojas representada como rectángulos superpuestos sobre el mapa.

Para el ejemplo propuesto, Sevilla está ubicada en la hoja denominada 0984. La superficie abarcada por cada una de éstas hojas es muy grande por lo que será necesario definir el área que queremos representar alrededor de la obra. Lo mejor para importar a Unity es escoger un área de tamaño cuadrado, para el ejemplo escogemos un cuadrado de lado 1000 m.

Tabla 1. Delimitación del terreno a modelizar

	COORDENADA ETRS89 (H30) NORTE	
ESQUINA	X (m)	Y (m)
SUPERIOR IZQUIERDA	234492	4144471
INFERIOR DERECHA	235492	4143471

4.3.2. Crear el heightmap

Para generar el heightmap a parir del MDT utilizaremos el programa Heightmap Creator[6]. El programa extrae los datos de la zona de interés (del fichero tipo ".asc") que contiene el *MDT05/MDT05-LIDAR* y los codifica en un fichero de tipo heightmap que puede ser importado en Unity.

Este programa proporciona como dato de salida la altura máxima del terreno creado que será importante recordar para importar el heightmap en Unity 3D.

[5] (Centro Nacional de Información Geográfica - Ministerio de Fomento)
[6] (Cuartas Hernández, 2014-2015)

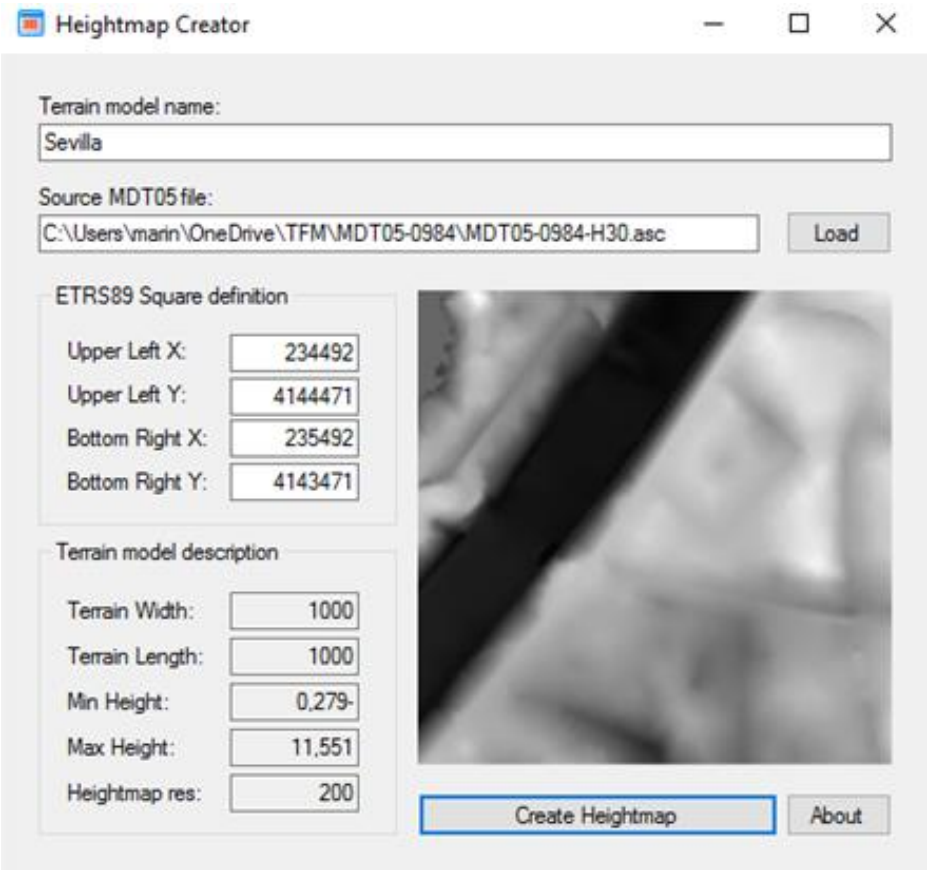

Figura 8. Captura de pantalla del programa HC para crear el Heightmap

4.3.3. Importar el heightmap en Unity3D

Para importar el terreno primero será necesario crear un nuevo proyecto. Una vez ejecutado Unity es necesario seleccionar la opción de crear nuevo proyecto y escoger la ruta del mismo. Se recomienda guardar la escena que se crea automáticamente al originar un nuevo proyecto. Para el ejemplo se le ha dado el nombre de Main.

Una vez creado el proyecto es necesario insertar un objeto de tipo terreno a nuestra escena para poder importar el Heightmap. Para ello se accede al menú en la parte superior de la pantalla GameObject > 3D Object > Terrain. También se puede acceder al menú a través de la ventana Hierarchy > Create > 3D Object > Terrain. Una vez creado se podrá ver en el árbol de la escena y al seleccionarlo se mostrarán sus componentes asociados en el Inspector.

Tras crear el objeto, es necesario configurar su tamaño. Para ello es necesario acceder a la pestaña de configuración a través de un botón en forma de rosca situado como se indica en la figura 9. Las dimensiones deben ajustarse al tamaño que se le ha dado al heightmap, en el ejemplo 1000x1000 y se debe de introducir, también, la altura máxima del terreno. Este dato se proporciona al crear el heightmap.

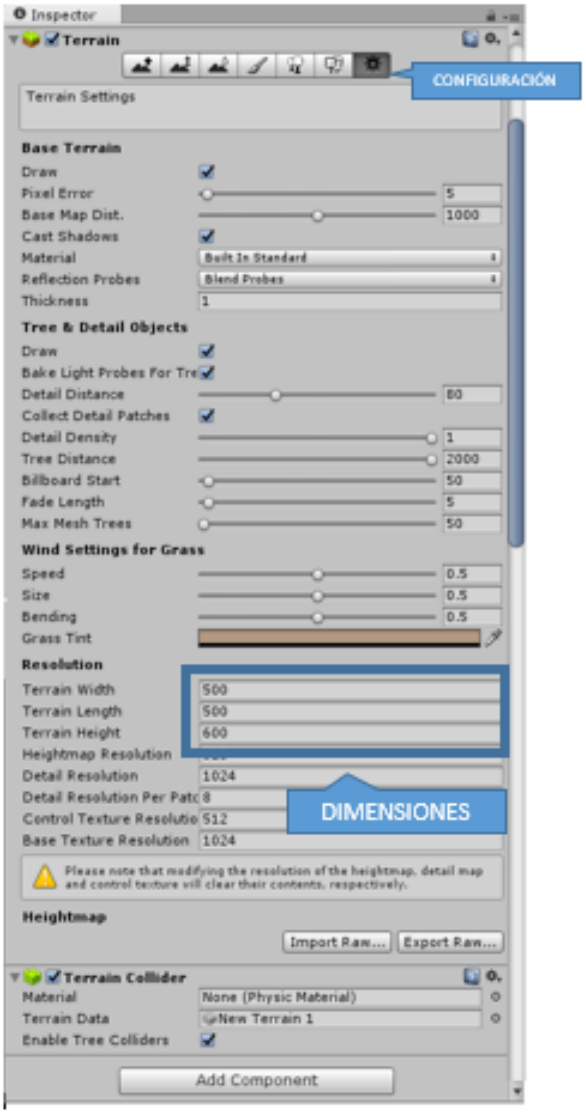

Figura 9. Inspector de un objeto Terrain

Una vez configuradas las dimensiones ya se puede importar el Heightmap a través del botón “Import Raw…” en la misma pestaña. Tras pulsarlo será necesario acceder a la ruta donde se encuentra el heightmap. Una vez seleccionado, aparece una ventana que muestra varios datos referentes al mismo, será necesario comprobar que la profundidad está en Bit16, que las dimensiones (Width y Height) son las correcta y que el Byte Order es Windows.

4.3.4. Referenciar el terreno

Una vez importado el terreno puede ser difícil tener referencias en el mismo para poder ubicar la obra y modelizar el entorno. Por ello, como referencia, podemos crear un plano guía recortando una imagen de un mapa del área con las dimensiones del terreno y añadirla como textura a un objeto plano de las mismas dimensiones como se muestra en la figura 10. Para crear un plano se accede al menú en la parte superior de la pantalla GameObject > 3D Object > Plane. Para añadir una textura al plano creamos una carpeta llamada Textures dentro de la carpeta Assets y arrastramos la imagen dentro de la carpeta. A partir de ese momento la imagen formará parte de los recursos del proyecto y podrá ser utilizada como textura simplemente arrastrando la

imagen dentro del elemento en el que queremos que se inserte. Una vez tengamos el mapa podemos desplazarlo y ubicarlo sobre el terreno para poder ubicar nuestra obra en planta.

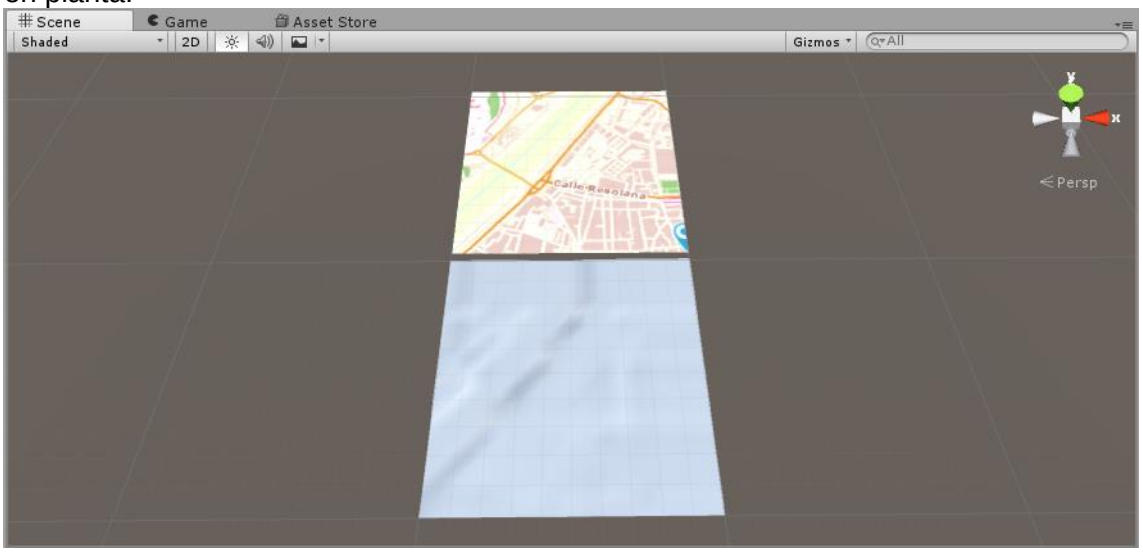

Figura 10. Terreno y plano de referencia

4.3.5. Añadir texturas, árboles y detalles al terreno

Seleccionando el terreno en la pestaña de Hierarchy, aparecerán en la pestaña de Inspector los componentes que tiene asociados.

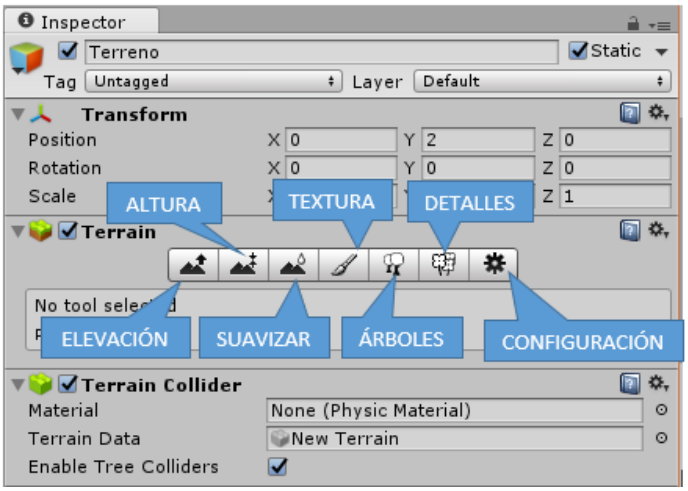

Figura 11. Inspector de un objeto Terrain

Estos componentes son:

- *Elevación del terreno*: permite generar elevaciones o eliminarlas si se mantiene la tecla SHIFT pulsada.

- *Altura del terreno*: permite generar elevaciones con una altura determinada o establecer la altura por defecto del terreno.

- *Suavizar terreno*: permite suavizar los detalles generados con alguno de los procesos anteriores.

- *Textura del terreno*: permite definir y utilizar un conjunto de texturas en el terreno. Para utilizar una textura es necesario añadirla primero.

- *Árboles*: permite incorporar árboles al terreno.

- *Detalles del terreno*: permite definir un conjunto de detalles como césped o rocas sobre el terreno.

- *Configuración del terreno*: permite definir los detalles principales del terreno. Es la pestaña que se ha utilizado para importar el Heightmap y definir la geometría del mismo.

Con estas herramientas y el plano guía como referencia se pueden añadir texturas, vegetación y detalles al terreno. Para las texturas, los árboles y los detalles es necesario importar un recurso dentro de los recursos estándar que incorpora Unity. Para importar el recurso se acude a Assets>Import Package>Environment. Una vez importado, es necesario añadir las texturas y árboles en las pestañas correspondientes.

Para añadir una textura es importante saber que la primera textura seleccionada será la que adquiera todo el terreno por defecto. Para el ejemplo práctico se ha utilizado la textura “GrasHillAlbeldo” que da un aspecto verde a todo el terreno. Para añadir una textura simplemente es necesario seleccionarla en el botón Edit textures>Add texture. Para el ejemplo se han añadido algunas texturas más como una de arena para el fondo del río. Si se quiere pintar sobre el terreno con estas texturas secundarias se puede seleccionar las características del pincel como su tamaño o la opacidad de la textura.

Para añadir árboles el proceso es similar, primero es necesario añadir los elementos en el botón Edit Tree>Add Tree para posteriormente colocarlos sobre el terreno. De esta forma añadimos algunos prefabricados de árbol como: Broadleaf_Desktop, Conifer_Desktop o Palm_Desktop. Para colocarlos sobre el terreno Unity ofrece las propiedades de tamaño de brocha y densidad de árboles.

Por último, es necesario seleccionar los detalles con los que se quiere pintar el terreno pulsando en el botón “Edit Details…” y luego “Add Grass Texture”. Para el ejemplo práctico se ha añadido la textura “GrassFrond01AlbedoAlpha”.

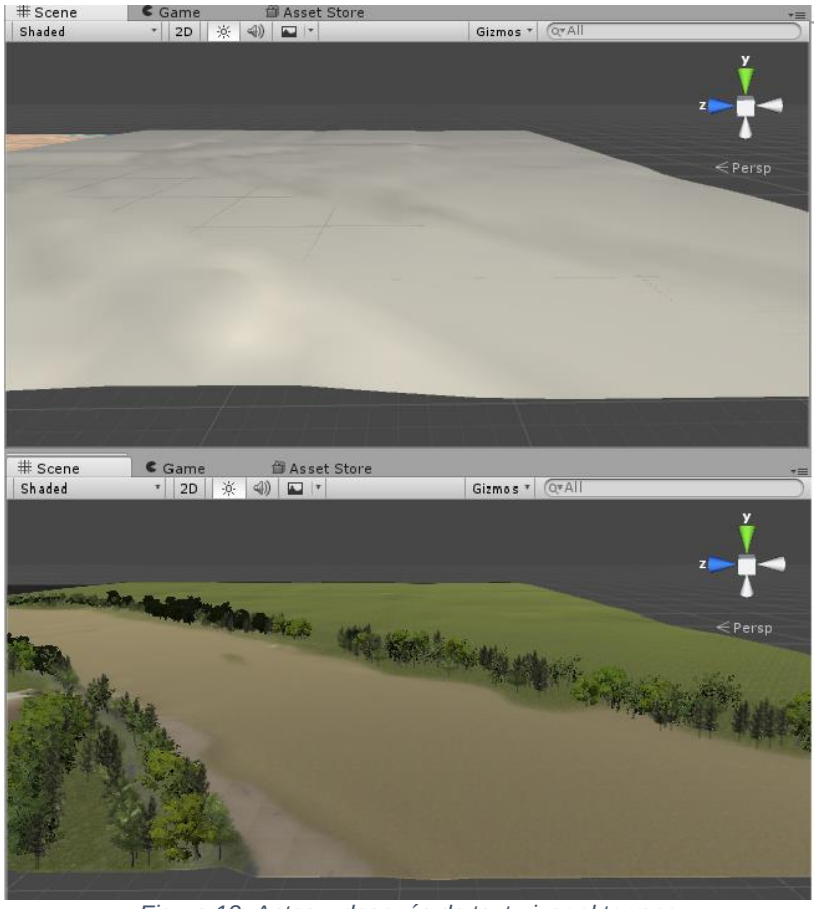

Figura 12. Antes y después de texturizar el terreno

4.4. MODELIZAR EL ENTORNO

4.4.1. Generación del cielo

Por defecto, al crear un nuevo proyecto, Unity ofrece un material para el cielo (Skybox) predefinido. Para dar un aspecto más realista se puede acudir a la tienda de assets (Asset Store) donde se encuentran materiales para el cielo gratuitos de mejor calidad que el predefinido por Unity.

Para descargarlos es necesario acceder a la pestaña Asset Store (ubicada como se indica en la figura 13). En la Asset Store hay cuadro de búsqueda que nos permite buscar un recurso a través de su descripción, en este caso “skybox” y la opción de buscar recursos únicamente gratuitos. En el ejemplo se ha utilizado el asset “Classic Skybox”. Una vez seleccionado es necesario descargar el paquete e importarlo. Al hacerlo lo encontraremos en una carpeta en la pestaña Assets.

Por tanto, el proceso para descargar recursos gratuitos es muy sencillo. A partir de este punto se recomienda el uso de la Asset Store para mejorar el aspecto de la escena. Por ejemplo, el paquete estándar que ofrece Unity para texturizar el terreno puede resultar escaso y se recomienda la búsqueda de paquetes con vegetación para dar más variedad y realismo al entorno.

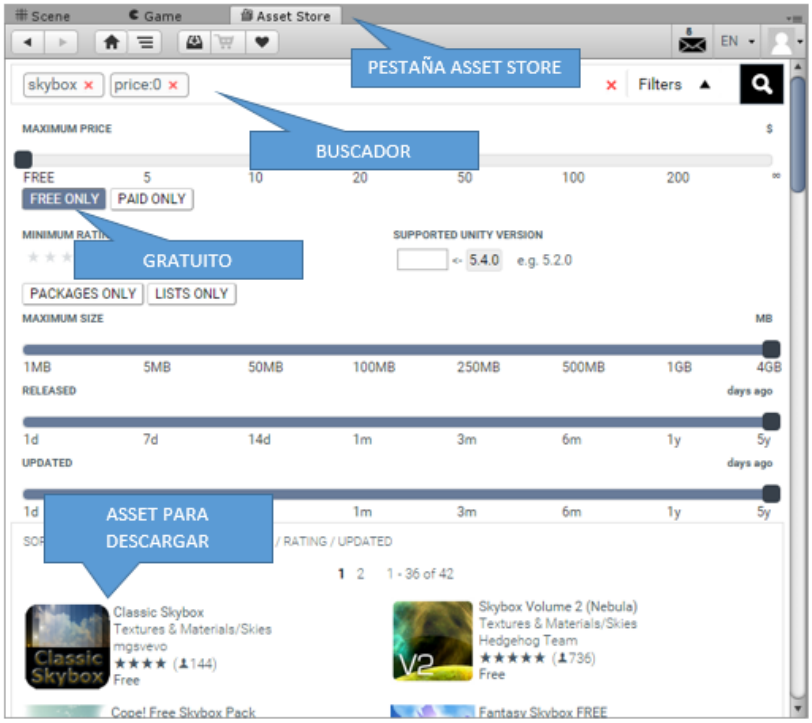

Figura 13. Asset store

Para cambiar el Skybox, es necesario dirigirse al menú Window > Lightning > Scene e incorporar el cielo descargado. El paquete de materiales para cielo importado nos proporciona una amplia lista de cielos. Lo recomendable es probar con diferentes cielos para elegir el que mejor ilumine la escena creada, para el ejemplo práctico se ha elegido el “skybox09”.

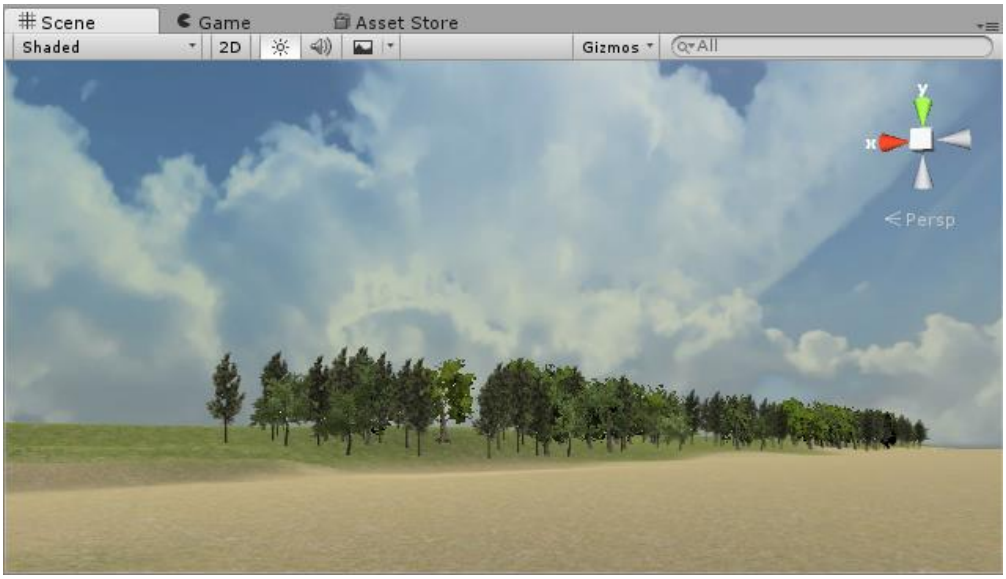

Figura 14. Vista del cielo importado

4.4.2. Generación del agua

Para añadir el agua del río Guadalquivir es necesario importar un asset dentro de los recursos estándar que incorpora Unity. El recurso necesario es el de Environment, que ya habíamos importado para añadir nuestras texturas y árboles al terreno. Por ello, se

encuentra en la carpeta de nuestro proyecto Assets > Standard Assets > Environment > Water > Water > Prefabs > WaterProDaytime. Es muy sencillo ubicarlo en la escena, simplemente es necesario arrastrarlo sobre la misma. Una vez en la escena será necesario ajustar su tamaño y posición para que coincida con la depresión en el terreno formada por el río.

Figura 15. Vista del terreno con texturas, árboles, cielo y agua.

Al ejecutar la aplicación con las Gafas de Realidad Virtual se recomienda cambiar la opción Water Mode (situada en el Inspector del objeto) a “Simple” lo que renderizará el agua sin reflejo y con un aspecto azulado. Esto se debe a que este objeto no está creado para el uso de gafas de Realidad Virtual, si no para mostrarlo en una pantalla. Por lo que la visión estereoscópica del mismo no está ajustada y al observar el agua crea una sensación incómoda al usuario.

4.4.3. Añadir la estructura

Una vez se ha creado el entorno añadir la estructura es un proceso muy simple. Es necesario simplemente localizar dentro de la carpeta “Assets” del proyecto la carpeta que se ha creado con anterioridad llama Estructuras donde se encuentra el puente en formato .fbx. Una vez localizado el puente lo más sencillo es arrastrarlo a la escena y ubicarlo en la misma con ayuda de nuestro mapa.

Así habremos conseguido componer nuestra escena con los elementos principales de la misma como se muestra en la figura 16.

Figura 16. Escena con la estructura.

4.4.4. Añadir alineaciones

Para componer la escena es interesante añadir las alineaciones del paseo que bordea la vereda del río, así como la carretera de acceso al puente. Para ello se recomienda el uso de assets gratuitos disponibles en la Asset Store. Existen dos opciones para componer las alineaciones:

- Paquetes de recursos con objetos prefabricados con forma de trozos de carretera. Puede descargarse paquetes de recursos compuestos por trozos de carretera rectos, con giros a derecha e izquierda, rampas, etc. E ir añadiendo dichos prefabricados a la escena componiendo las alineaciones. Dichos prefabricados vienen con texturas predefinidas que pueden modificarse. Si las alineaciones que se quieren crear no son muy extensas puede ser un recurso válido y fácil de usar.

- Paquetes de recursos con una herramienta para la creación de alineaciones. Por ejemplo, existe el paquete “RoadTool” que permite crear alineaciones simplemente pulsando la combinación de teclas shift + R. Una vez ejecutado es necesario marcar los puntos por donde se quiere pasar la alineación. El único inconveniente es que el terreno sobre el que se quiera dibujar la alineación debe de estar nivelado para poder dibujar la carretera.

Para el ejemplo práctico se ha escogido el segundo recurso previo acondicionado del terreno. Para dicho acondicionado se ha usado la opción “*Altura del terreno”* en el Inspector del objeto tipo terreno marcando una altura fija de 4 m para el paseo en la vereda del río y de 12 m para la carretera de acceso al puente.

Figura 17. Creación de alineaciones gracias al recursos "RoadToll" disponible en la Asset Store de Unity

Para crear la alineación es necesario que esté pulsado el botón “Move” de interacción con la escena (segundo por la izquierda).

Una vez creada la alineación es posible ajusta el ancho y añadirle la textura o material que se desee. Si se desea editar la alineación simplemente se debe seleccionar e objeto creado en el árbol del editor y darle al botón “Modify” del Inspector.

También pueden añadirse nuevos puntos a la alineación una vez seleccionado el botón “Modify”. Estos puntos se añadirán por defecto a continuación del último nodo creado. Si se quiere crear un nodo al comienzo de la alineación habrá que cambiar el parámetro del Inspector “Insert At Point” de -1 a 0.

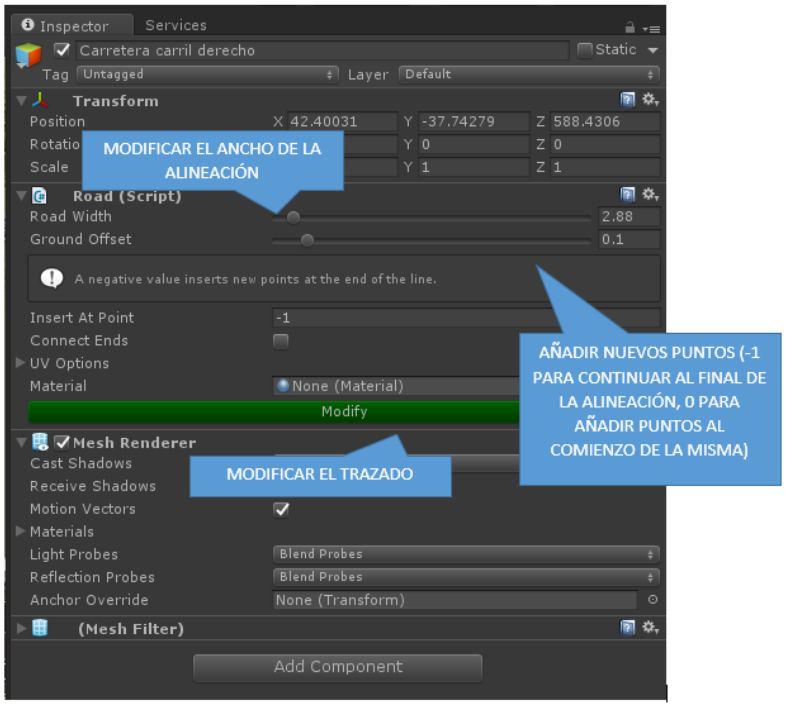

Figura 18. Inspector de los objetos creados con el recurso "RoadTool"

4.4.5. Añadir audio de ambiente

Como el proyecto se encuentra ubicado próximo al río Guadalquivir lo lógico sería añadir un clip de audio que simulase el ruido que hace el río. Para ello existen en Internet bases de datos con sonidos de todo tipo, una vez descargado un clip de audio con un sonido de río se guarda en una carpeta, que se ha creado previamente en la carpeta de Assets, con el nombre de Audios.

Para añadir el audio se selecciona el elemento que emite el sonido, en este caso el río, y se le añade un componente de tipo Audio > Audio Source. Una vez añadido el componente es necesario arrastrar el clip a la ventana Audio Clip. Es importante ejecutar la escena a medida que se va creando la misma para comprobar el funcionamiento de los objetos, por ejemplo, se puede ajustar el volumen del audio para que se convierta en un ruido ambiental.

4.5. INTERACCIÓN CON EL ENTORNO

Una vez creado el entorno el siguiente paso es posibilitar la interacción con el mismo. Aunque a priori pudiera parecer un proceso complejo, Unity lo simplifica al ofrecer un paquete de recursos estándar con personajes. Esto quiere decir que, de una manera no muy compleja, se puede añadir personajes que nos permiten interactuar con la escena.

4.5.1. Personajes

Unity ofrece un paquete de recursos con dos tipos de personajes que se diferencian según el punto de vista:

- *Personaje en tercera persona*: Al ejecutar la escena la cámara se sitúa sobre el personaje de manera que se participa en la escena como observador. La interacción con la escena es posible gracias al control sobre el movimiento del personaje.

- *Personaje en primera persona*: Al ejecutar la escena la cámara simula la visión del personaje. Este tipo de personaje es el apropiado para la Realidad Virtual ya que el objetivo es permitir que el usuario se sumerja en la escena creando la sensación de estar presente en ella.

Para utilizar este recurso es necesario importarlo, lo encontramos en Assets>Import Package>Characters. Una vez importado se habrá creado una carpeta ubicada en nuestro proyecto en Assets>Standard Assets>Characters>FirstPersonCharacter. El objeto prefabricado que añadiremos en la escena estará compuesto por muchos componentes como una cámara, un receptor de audio, o un script que controla el movimiento.

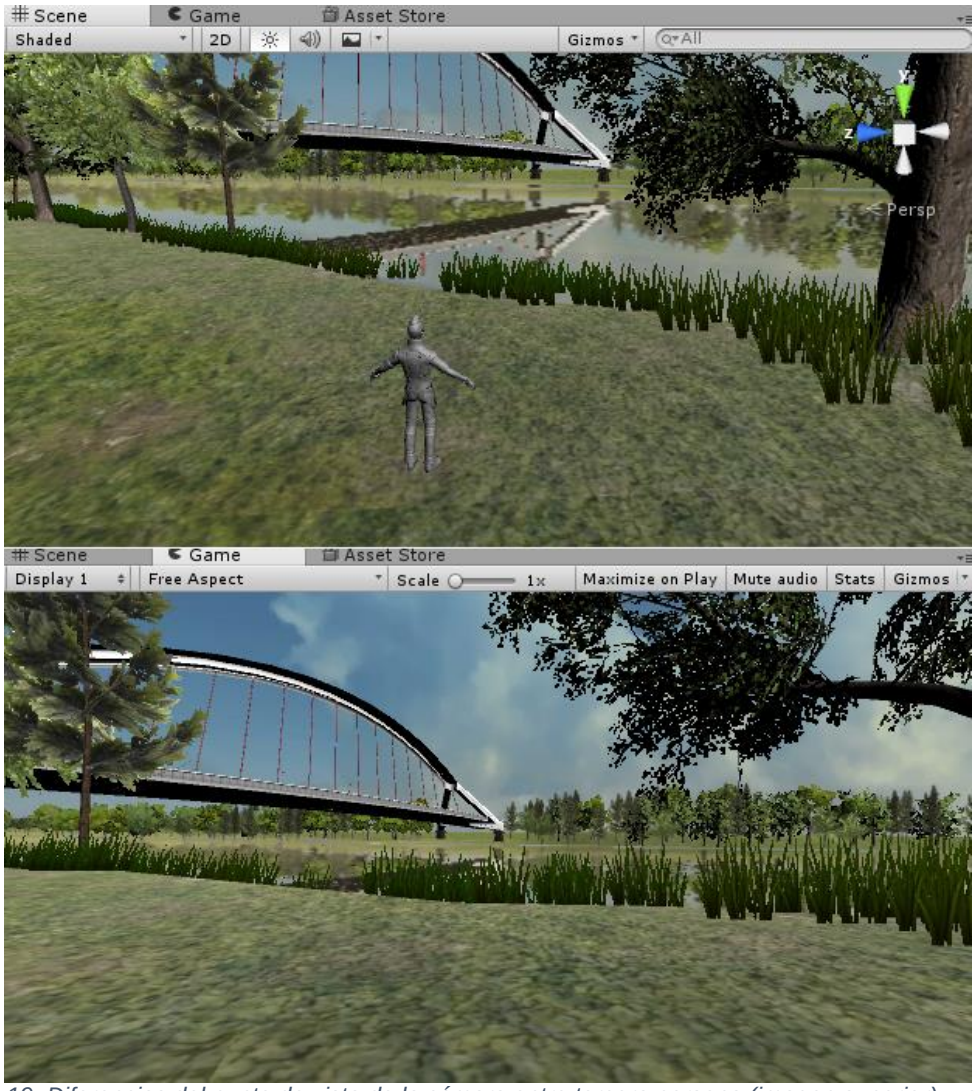

Figura 19. Diferencias del punto de vista de la cámara entre tercera persona (imagen superior) y primera persona (inferior)

Unity ofrece dos controladores en primera persona. Para ajustar la aplicación a el uso de Realidad Virtual se recomienda el uso del Rigid Body FPC ya que no simula los pasos de una persona (lo que puede crear una sensación incómoda para el usuario) y además adapta los ejes de movimiento a la dirección a la que se enfoca lo que es esencial para el usuario de dispositivos de Realidad Virtual.

Para incorporar el personaje principal es necesario añadir el modelo prefabricado Rigid Body FPC a la escena. Para añadir este elemento en la escena bastará con buscarlo en la carpeta StandardAssets >Characters > First PersonCharacter > Prefabs y arrastrarlo a la misma. Por último, será necesario colocarlo en el punto en el que se quiera situar el personaje al comienzo de la aplicación. Siempre que se ejecute el programa se comenzará en dicho punto, a partir del cual el usuario podrá moverse libremente por la escena.

Es importarle dar un nombre al personaje principal ya que éste va a interactuar con otros objetos y será necesario diferenciarlo a la hora de programar el comportamiento de los objetos ante él. Para este ejemplo práctico se ha escogido el nombre “Player”.

Una vez creado el personaje es muy importante ejecutar la aplicación y experimentar la sensación de moverse a través del espacio. Se recomienda ajustar la altura de la

cámara a unos 1.4 metros sobre el suelo que es la altura media de la mirada. Si no se realiza esta operación se podría estar creando una aplicación que provocase malestar sobre el usuario.

A su vez, para combatir la cinetosis, es necesario ajustar la velocidad con la que se desplaza el usuario por la escena. Para ello se acude al Inspector del personaje y se modifica las velocidades en los ajustes de movimiento (Movement Settings). Tras experimentar diferentes velocidades se recomienda ajustar todas las velocidades, como máximo, a dos.

4.5.2. Cómo modificar la configuración para controlar al personaje

Unity utiliza como input el teclado y el ratón para controlar el movimiento del personaje por defecto. Para mover el personaje alrededor de la escena se utilizan las flechas. Para mover la cámara, es decir, enfocar la vista se utiliza el ratón o si se tiene activada la opción de Realidad Virtual simplemente es necesario mover la cabeza.

Para visualizar la obra en tres dimensiones no es necesario levantarse de la silla o alejarse del teclado por lo que la opción para moverse por la escena con las flechas del teclado es suficiente.

El casco de Realidad Virtual con el que se cuenta para ejecutar ese proyecto, el set Oculus Rift, incluye un mando de la Xbox One inalámbrico que pude usarse para desplazarse por la escena, lo que permite alejarse del ordenador o posicionarse levantado mientras se interactúa con la escena. Si se quiere configurar el mando es necesario acudir a Edit > Project Settings > Input.

En el mencionado apartado puede seleccionarse que botones controlan el movimiento del personaje. Hay que prestar especial atención a que plataforma objetivo estará orientada la aplicación que va a crearse ya que la numeración de los ejes es diferente en PC, Mac o Linux.

Una vez creada la aplicación, al ejecutarla, la ventana de inicio de la misma permite cambiar la configuración del input para que una misma aplicación sirva para diferentes controles.

Tabla 2. Controladores para el mando de la Xbox One

MANDO XBOX ONE	WINDOWS
Botón A	joystick button 0
Botón B	joystick button 1
Botón X	joystick button 2
Botón Y	joystick button 3
Botón LB	joystick button 4
Botón RB	joystick button 5
Botón ATRÁS	joystick button 6
Botón START	joystick button 7
Stick izquierdo	joystick button 8
Stick derecho	joystick button 9
Horizontal	X axis

Vertical	Y axis
Giro horizontal	4th Axis
Giro vertical	5th Axis
Flechas - horizontal	7th Axis
Flechas - vertical	8th Axis
Botón LT	9th Axis
Botón RT	10th Axis

Con ello se puede definir el botón o eje que se quiera para cada movimiento.

4.5.3. Geometría y física en Unity 3D

Una vez construido el modelo en Unity con la opción de Realidad Virtual activada la plataforma permite directamente desplazarse por la escena, acercarse a la estructura y verla desde cualquier ángulo.

El siguiente paso es crear una forma de interactuar con los objetos dentro de la escena. Para este capítulo del proyecto será necesario un conocimiento básico en programación. Para el desarrollo del proyecto se ha optado por utilizar el lenguaje de programación C#.

En la Realidad Virtual existen dos formas cómodas y naturales para interactuar con objetos: acercarnos a ellos o mirarlos. Pero antes de avanzar con el desarrollo del ejemplo se van a introducir una serie de componentes de los objetos en Unity que facilitarán la explicación posterior.

Un objeto en Unity se forma a partir de componentes, entre los que destacan:

- *Rigidbody:* Los Rigidbodies permiten a sus GameObjects actuar bajo el control de la física. El Rigidbody puede recibir fuerzas o colisiones con otros objetos para hacer que el objeto asociado se mueva de una manera realista. Cualquier GameObject debe contener un Rigidbody para ser influenciado por gravedad o por fuerzas externas añadidas mediante código

- *Collider:* son componentes que añaden geometría de colisión al objeto, es decir, permiten al objeto al que se le añade reaccionar ante otros objetos que también tengan una componente Collider siempre y cuando alguno de ellos tenga un componente Rigidbody. Es como una malla que se coloca alrededor del objeto y que define su área de interacción. Tiene tres formas primitivas: una esfera, una cápsula y una caja y se muestran en la escena mediante líneas verdes. Para darle formas más complejas se puede usar el componente Mesh Collider que se adaptará a la forma que tenga el objeto. Cuando dos objetos con componentes Collider interactúan ocurre un evento por lo que la manera en que se comportan ambos objetos al interactuar puede ser programada mediante un Script.

  El componente Collider tiene una opción para que el motor de física no tenga en cuenta los choches del colisionador de forma automática y, a cambio, dispare un evento para que pueda tratarse por código (seleccionando la opción “Is Trigger” en el Inspector). Cuando un Collider tiene está opción activada las cosas no chocan contra la malla, sino que pueden pasar a través de ella y esto puede ser detectado en código. Es decir, no sólo podemos usar este componente para la colisión de dos objetos macizos si no que puede utilizarse como un elemento vacío que permita interactuar con la escena. Un ejemplo sería para registrar

cuando un balón cruza una portería. En el plano del larguero podría utilizarse un componente Collider con la opción “Is Trigger” que detectase cuando el balón traspasa la portería sin interferir con la trayectoria del mismo.

Observando los elementos del personaje principal que hemos añadido a la escena podemos comprobar que entre sus componentes se encuentra:

- Un componente Rigidbody que permite simular el movimiento real del personaje, por ejemplo.
- Un Collider en forma de cápsula que limita el área de interacción del personaje.
- Un Script que controla los movimientos del personaje.
- Además, tiene asociado un objeto tipo cámara que sirve de ojos a la hora de visualizar la escena.

4.5.4. Objetos UI (User Interface)

UI se refiere a la interfaz de usuario que se muestra de manera bidimensional en una aplicación y que proporciona información al usuario como pueden ser textos, imágenes o controles tipo botón.

En Unity los objetos tipo UI siempre se disponen sobre un objeto Canvas o lienzo que representan un espacio abstracto sobre el que yace un objeto tipo UI y sobre el que se renderiza. Por lo que cuando se quiera añadir un objeto UI éste deberá ser dependiente de un lienzo.

En las aplicaciones de Realidad Virtual los objetos UI no pueden implementarse de manera tradicional, es decir, como un plano que forma parte de la pantalla sobre el que el usuario puede realizar acciones. Por lo que para utilizar este tipo de elementos en aplicaciones destinadas a la Realidad Virtual será necesario ubicarlos como un objeto más en la escena.

En función de cuándo y dónde se quiera ofrecer la información, estos objetos pueden disponerse de dos maneras distintas:

- En frente de la cámara con independencia de hacia dónde mire el usuario.
- En un emplazamiento concreto del entorno 3D, siendo visibles si el usuario mira en esa dirección.

Para que un lienzo forme parte de la escena es necesario activar la opción “world space” en el Render Mode de su Inspector.

Debido a que éste tipo de objeto es el que va a usarse en los próximos capítulos para interactuar con la obra, se recomienda crear un lienzo por defecto que nos servirá como base. Para ello lo primero que hay que hacer es crear un lienzo a través de Window > GamObject > UI > Canvas. Para el ejemplo práctico se le ha dado el nombre de DefaultCanvas. Una vez creado, se activa la opción “world space” en el Render Mode de su Inspector.

El componente Rect Transform define la malla como si fueran los cuadrados de una libreta. Se usa para colocar los objetos UI sobre él. Por defecto le daremos unas dimensiones de 640 x 480 con una relación de 0.75. Le daremos una escala de (0.00135, 0.00135, 0.00135) que es el tamaño de un pixel en el espacio de la escena. Por último, centraremos el lienzo en el terreno.

Una vez ubicado, se le añadirá un objeto UI tipo imagen blanca que ayudará a visualizar el lienzo cuando se necesite. Para ello se accede a Window > GamObject > UI > Image. Es necesario asegurarse de que la imagen está vinculada al Lienzo. Una vez añadida en el Inspector será necesario modificar el Anchor Presets a strech-strech.

Por último, se le añadirá un objeto UI tipo texto a través de Window > GamObject > UI > Text al que le daremos las siguientes propiedades: Anchor Presets a strech-strech, escala (4,4,4) y alineación centrada.

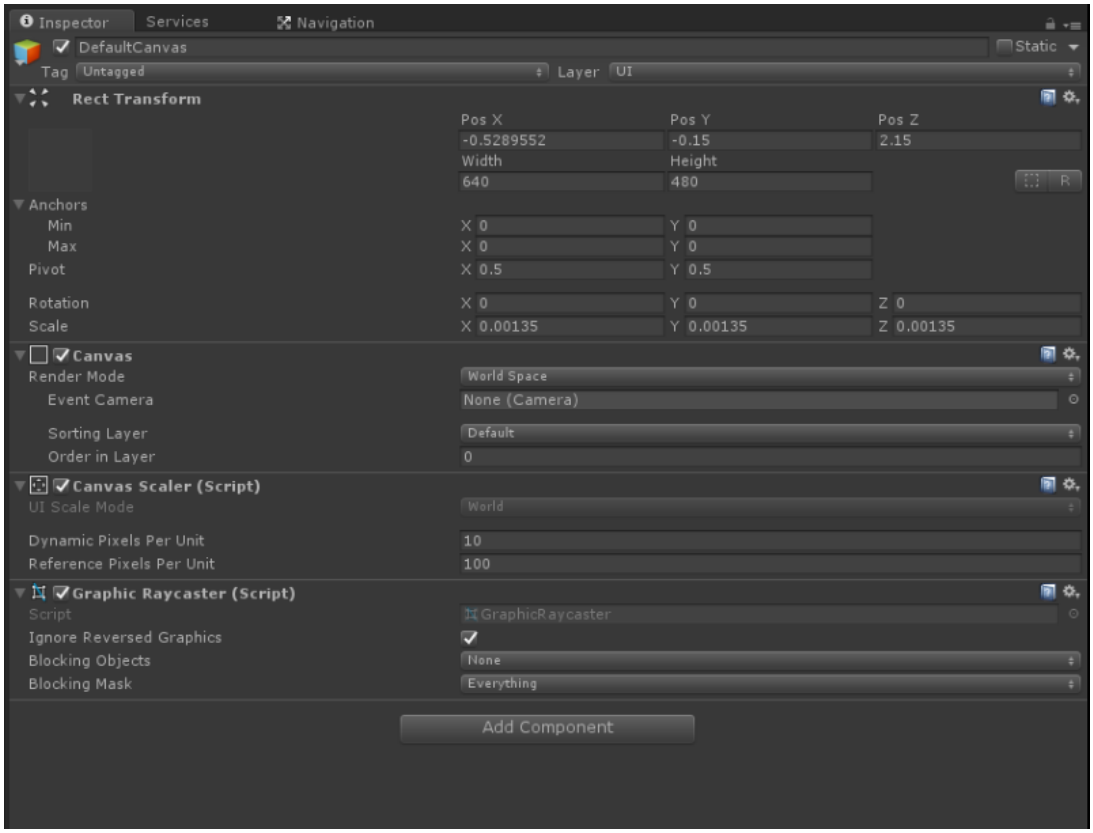

Figura 20. Inspector con valores de referencia para el UI de referencia

Una vez añadido el texto puedo observarse que aparece bastante distorsionado por lo que se recomienda modificar las propiedades del lienzo. Cambiaremos el Dynamic Pixel Per Unit a 10.

Este será el objeto IU que servirá como base para todos los objetos IU que queramos añadir. Es recomendable guardar el objeto en una carpeta creada previamente llamada Prefabs arrastrando el objeto a la carpeta.

4.5.5. Introducción a la programación

Unity es capaz de realizar muchas actividades como administrar los objetos, renderizarlos, animarlos o calcular la física que los afecta. Esto implica que Unity es un programa por sí mismo y que por lo tanto estará construido por código. Este código

interno es accesible a través de la interfaz del editor (API) que se ha estado utilizando. En dicha Interfaz los scripts se muestran como componentes que se pueden configurar.

Como lenguaje de programación Unity se basa en el lenguaje C#, cuando se utiliza un lenguaje de programación es muy importante obedecer la sintaxis del mismo. Si al crear un script Unity detecta algún error lo muestra a través de la ventana Console.

El programa incorpora un editor de código llamado Visual Studio de Microsoft aunque si se cuenta con algún otro editor éste puede ser utilizado.

En un C# script de Unity algunos símbolos y palabras son parte del lenguaje C#, otros son parte de Microsoft .NET Framework y otros son proporcionados por la API de Unity. Cuando se crear un nuevo script en él aparece el siguiente código:

```
using UnityEngine;
using System.Collections;

public class NewBehaviourScript : MonoBehaviour {

	// Use this for initialization
	void Start () {

	}

	// Update is called once per frame
	void Update () {

	}
}
```

Las dos primeras líneas indican que el script necesita otras librerías de código para ejecutarse. La palabra using pertenece al lenguaje C#. La línea using UnityEngine; dice que se estará usando la librería UnityEngine de la interfaz del programa. La línea using System.Collections; permite usar librería de funciones relacionadas con el manejo de colecciones del .Net Framework..

En C# cada línea de código acaba en un punto y coma. La doble línea // permite añadir comentarios ya que todo lo que se encuentre detrás de la doble línea será ignorado al ejecutar el script.

Para facilitar la tarea de programar existen las clases que son como plantillas de código con sus propias propiedades (variables) y comportamiento (funciones). Estas clases derivan del MonoBehaviour. Luego una clase define las variables y las funciones. Una variable contiene datos de un tipo específico. Las funciones permiten implementar la lógica, es decir, permiten definir las instrucciones paso a paso. Dichas funciones pueden recibir argumentos y pueden devolver nuevos valores cuando son ejecutadas.

Cuando se declara un miembro de una clase como público puede ser visto y usado por otro código en otro script, sin embargo, cuando es privado sólo puede ser Dentro de la misma clase donde se define.

Unity permite invocar funciones-mensaje especiales si se las define. Un ejemplo de ello son las funciones Start () y Update () incluidas en el código por defecto de un script en Unity. Delante de las funciones se define el tipo de valor que devolverá. Cuando una función no devuelve un valor incluyen la palabra clave void como tipo de retorno.

La función Start () se ejecuta antes de que comience la escena por lo que se usa para dar instrucciones iniciales. La función Update () es ejecutada al mismo tiempo que la escena. El tiempo es dividido en frames y en cada frame se ejecuta la función.

Una vez se haya escrito o modificado un script, es necesario guardarlo. Una vez guardado se podrá observar en la ventada Console si Unity detecta algún error en él.

Esto es sólo una pequeña introducción a la programación en Unity, en los ejemplos que siguen se explicará el funcionamiento del código creado.

4.5.6. Programar un objeto para que aparezca cuando el usuario se aproxima: carteles interactivos

Una de las formas intuitivas con las que interactuar con un objeto en la Realidad Virtual es acercándonos a él. En este ejemplo práctico se va a realizar un código para que varios objetos tipo texto aparezcan dinámicamente a medida que el usuario se desplaza por la escena.

Primero es necesario colocar un Empty Game Object con un componente Collider en las zonas dónde se quiere que aparezcan los textos. Marcamos la opción “Is Trigger” para que el colisionador desencadene el evento cuando nos acerquemos a él. Una vez se ha colocado el Game Object con su componente Collider le daremos el nombre de Activador.

Figura 21. Empty Game Object al que se le ha añadido un componente collider esférico.

El siguiente paso es crear el cartel, para ello se utilizará el lienzo que se ha construido en el apartado anterior ubicado en Assets>Prefabs>DefaultCanvas. Para incorporarlo a la escena simplemente es necesario arrastrarlo hasta la ubicación deseada (detrás del activador).

Para el ejemplo se ha escrito el siguiente mensaje de bienvenida “*BIENVENIDO AL PASEO INTREACTIVO POR EL PUENTE DE LA BARQUETA”* y se le ha añadido otro componente tipo texto para añadir en un tamaño de letra más pequeño*: “Puedes moverte libremente por la escena. Encontrarás carteles informativos y alguna sorpresa escondida ¡Diviértete!*”.

Figura 22. Cartel de bienvenida

Una vez creados los dos elementos es necesario programar un Script asociado al Activador para que cuando el personaje principal entre en contacto con el Collider se active el cartel y que cuando salga se desactive.

Unity pone a disposición del usuario, a través de su página web, una gran base de ejemplos prácticos y tutoriales que sirven como herramienta de aprendizaje. Así mismo, en la esquina superior derecha de cada componente hay un icono de un libro con una interrogación que nos lleva a la ayuda online de Unity dónde nos explican todas las propiedades del componente. Esto sirve de gran ayuda a la hora de programar Scripts para el funcionamiento de los objetos porque sabremos que propiedades se vinculan a cada componente del objeto.

Para crear el script primero es recomendable crear una carpeta y nombrarla Scripts. Dentro de la carpeta haciendo clic sobre el botón derecho se selecciona la opción Create> C# Script y se le da un nombre al fichero. En el ejemplo TextoDescriptivo. Una vez creado se hace doble clic en él para abrir Visual Studio de Microsoft de Unity dónde se podrá escribir el siguiente código:

```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;


public class ActivarTexto : MonoBehaviour

{
    public GameObject TextoDescriptivo;

    void Start()
    {
```

```
    TextoDescriptivo.SetActive(false);
    }


    void OnTriggerEnter(Collider other)
    {
        if (other.gameObject.name == "Player")
        {
            TextoDescriptivo.SetActive(true);
        }
    }

    void OnTriggerExit(Collider other)
    {

         if (other.gameObject.name == "Player")
        {
            TextoDescriptivo.SetActive(false);
        }
    }
}
```

El funcionamiento del script es el siguiente:

- Primero se hace pública la variable de tipo TextoDescriptivo que será aquel objeto que aparecerá al acercarnos. De esta forma será visible en el Inspector de Unity y podremos asignarle un objeto presente en la escena.

- Luego, antes de iniciar la escena, se asegura que dicho objeto está desactivado.

- Al iniciar la escena se utiliza una función que detecta cuando un objeto ha entrado en el área encerrada por el Collider del objeto al que se le añade el script y además se le añade la condición de que si el objeto que entra en contacto es el jugador, que active el objeto al que hemos llamado TextoDescriptivo.

- Por último, se utiliza una función que detecta cuando un objeto ha salido del área definida por el Collider al que se le añade el script. Incluyendo la condición de que si el objeto que sale del mismo es el jugador, que desactive el objeto al que hemos llamado TextoDescriptivo.

Como podemos ver se han utilizado las siguientes funciones:

- *OnTriggerEnter:* reconoce cuando otro objeto con un elemento Collider (y un Rigid Body) entra en la malla del Collider con la opción “Is Trigger” activada al que se le añade el Script.

- *OnTriggerExit:* reconoce cuando otro objeto con un elemento Collider (y un RigidBody) sale del interior de la malla del Collider Trigger al que se le añade el Script.

- *gameObject.SetActive(true/false):* activa si (true) o desactiva (false) al objeto que invoca.

Es importarte que se invoque al personaje en primera persona por el nombre que se le haya dado. En el ejemplo se ha usado el nombre de “Player”. Como puede verse en el Script, condicionamos el comportamiento del cartel a la entrada del objeto “Player”.

Una vez creado el Script se le añade al Activador. Para añadir el Script basta con arrastrar el fichero al objeto dentro del árbol del proyecto o añadirlo como componente en el Inspector (una vez seleccionado el objeto).

Cuando se haya añadido el Script en la pestaña del Inspector, éste preguntará por el objeto al que se le quiere llamar “Texto Descriptivo” y que será el que aparezca y desaparezca cuando se entre en contacto con el Collider. Por lo que se selecciona el Lienzo que contiene el texto y se arrastra a la ventana como se muestra en la siguiente figura.

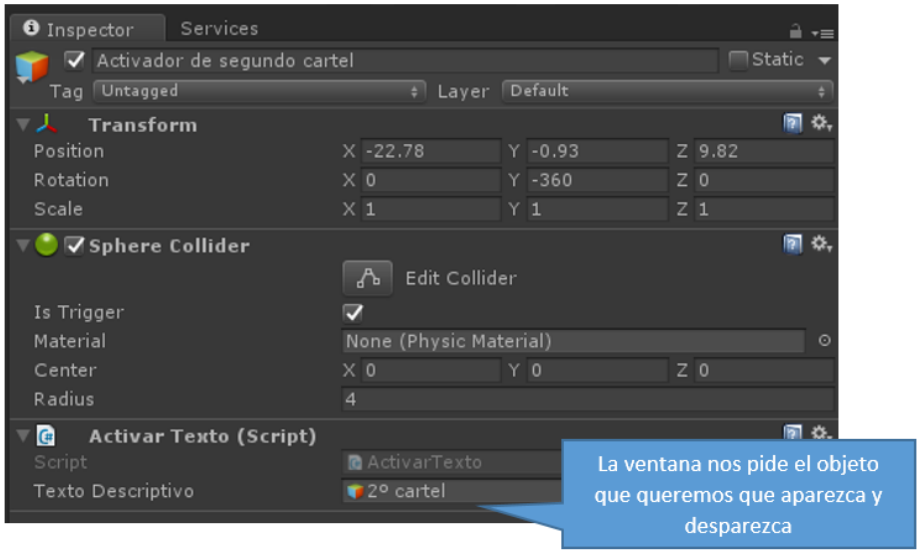

Figura 23. Script añadido como componente al objeto Collider y ventana que nos pide el objeto al que se invoca en el Script

Se recomienda ejecutar la escena para comprobar el funcionamiento del Script. Si éste no funcionase habrá que revisar que el nombre del personaje principal de la escena coincide con el invocado en el Script y que éste contiene un componente Rigid Body, otro posible fallo es no haber activado la función “Is Trigger” en el objeto con el componente Collider que funcionará como activador.

Para que la experiencia de usuario sea más cómoda se recomienda que el cartel siempre se oriente a la cámara para ello se puede escribir el siguiente script:

```
using UnityEngine;
using UnityEngine.UI;
using System.Collections;

public class LookMoveTo : MonoBehaviour {

    public GameObject terreno;
    public Transform cartel;

        void Start ()
{

        }
```

```
    void Update () {

Transform camera = Camera.main.transform;
Ray ray;
RaycastHit[] hits;
GameObject hitObject;

ray = new Ray(camera.position, camera.rotation * Vector3.forward);
hits = Physics.RaycastAll(ray);

for (int i=0; i< hits.Length; i++)
{
   RaycastHit hit = hits[i];
   hitObject = hit.collider.gameObject;

   if (hitObject == terreno)
   {
         cartel.LookAt(camera.position);
         cartel.Rotate(0.0f, 180.0f, 0.0f);
   }
   }
}
   }
```

El funcionamiento del script es el siguiente:

- Primero se hacen públicas las variables terreno y el cartel para que se puedan escoger a través del Inspector.

- Luego se crean unas variables para referirse a la cámara, al rayo de visión central, y al objeto sobre el que colisionará así como la propia colisión. Tras lo que se define el rayo en función de la posición de la cámara y la colisión de éste con cualquier elemento.

- Lo siguiente es definir el comportamiento de los objetos. Se utiliza un ciclo for para proyectar el rayo y registrar si colisiona con un objeto.

- Por último, se define el comportamiento en caso de que el rayo colisione con el cartel. Cuando el rayo colisione con el terreno, es decir, cuando no se esté mirando al cartel, éste se orientará hacia la cámara a través del comando “LookAt”. El resultado de dicho comando es que el cartel estará enfocado hacia la dirección de la mirada cuando el usuario vuelva a mirarlo por lo que es necesario rotarlo 180 grados alrededor del eje y.

Este script es muy interesante ya que introduce la forma de interactuar con los objetos al mirarlos.

Tras crear el script será necesario añadirlo como componente al personaje en primera persona y arrastrar hasta la ventana “Terreno” el objeto terreno y a la ventana “Cartel” el cartel que queremos que se oriente en el Inspector. Si queremos que más de un objeto se oriente, por ejemplo, todos los carteles será necesario añadirlos en el script de la siguiente manera:

```
using UnityEngine;
using UnityEngine.UI;
using System.Collections;

public class LookMoveTo : MonoBehaviour {

    public GameObject terreno;
    public Transform cartel1;
    public Transform cartel2;
    public Transform cartel3;

    void Start ()
    {

        }

        void Update () {

      Transform camera = Camera.main.transform;
      Ray ray;
      RaycastHit[] hits;
      GameObject hitObject;

      ray = new Ray(camera.position, camera.rotation * Vector3.forward);
      hits = Physics.RaycastAll(ray);

      for (int i=0; i< hits.Length; i++)
      {
         RaycastHit hit = hits[i];
         hitObject = hit.collider.gameObject;

         if (hitObject == terreno)
         {
                cartel1.LookAt(camera.position);
                cartel1.Rotate(0.0f, 180.0f, 0.0f);
                cartel2.LookAt(camera.position);
                cartel2.Rotate(0.0f, 180.0f, 0.0f);
                cartel3.LookAt(camera.position);
                cartel3.Rotate(0.0f, 180.0f, 0.0f);
         }
      }
     }
    }
```

Así podremos incluir todos los carteles necesarios y orientarlos hacia el usuario. Para el ejemplo se han añadido otros dos carteles, uno con la historia del puente y otro para informar del siguiente apartado.

Por último, se ha incluido un cartel con el título del proyecto que se activará una vez que el usuario se posicione encima del puente. Éste aparecerá en el horizonte y será de gran tamaño.

4.5.7. El cambio del puente

En el presente apartado se propone realizar un cambio en el modelo del puente. Una vez que el personaje haya dejado el puente a sus espaldas, al girarse, podrá ver un modelo simplificado de cómo viajan las fuerzas en el puente.

Para ello primero es necesario construir un modelo tridimensional con el esquema de fuerzas. Para el ejemplo práctico se ha modificado la geometría original del puente en Sketch Up. Una vez construido el modelo e importado a Unity siguiendo los pasos del apartado 4.2.1 se emplazará en el mismo lugar que el puente original.

Lo siguiente será crear un Objeto con un elemento Collider con la opción “Is Trigger” seleccionada y colocarlo como Activador en el punto en el que se quiera cambiar un puente por otro. Se le dará una dimensión importante para que el personaje pueda ver el esquema del puente al mismo tiempo que se desplaza por la escena.

Una vez creado el Activador es necesario adjuntarle un script que detecte la entrada del personaje, y que cuando esto ocurra active el modelo del esquema del puente y desactive el modelo del puente real.

Se irá a la carpeta Scripts dentro de la carpeta Assets y calcando sobre el botón derecho se seleccionará la opción Create> C# Script y se le dará un nombre al fichero. En el ejemplo CambioPuente. Una vez creado se hace doble clic en él para abrirlo dónde se podrá escribir el siguiente código:

```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;


public class CambioPuente : MonoBehaviour

{
    public GameObject PuenteReal;
    public GameObject EsquemaPuente;

    void Start()
    {
        EsquemaPuente.SetActive(false);
    }


    void OnTriggerEnter(Collider other)
    {
        if (other.gameObject.name == "Player")
        {
            PuenteReal.SetActive(false);
            EsquemaPuente.SetActive(true);
        }
    }

    void OnTriggerExit(Collider other)
    {

        if (other.gameObject.name == "Player")
```

```
    {
        PuenteReal.SetActive(true);
        EsquemaPuente.SetActive(false);
    }
  }
}
```

El script es parecido al de los carteles interactivos y funciona de la siguiente manera:

- Primero se hacen públicos los objetos PuenteReal y EsquemaPuente para que se puedan escoger a través del Inspector.

- Luego, antes de iniciar la escena, se asegura que el objeto EsquemaPuente está desactivado.

- Al iniciar la escena se utiliza una función que detecta cuando un objeto ha entrado en el área encerrada por el Collider del objeto al que se le añade el script y además se le añade la condición de que si el objeto que entra en contacto es el jugador, que active el objeto al que hemos llamado PuenteReal.

- Por último, se utiliza una función que detecta cuando un objeto ha salido del área definida por el Collider al que se le añade el script. Incluyendo la condición de que si el objeto que sale del mismo es el jugador, que desactive el objeto al que hemos llamado PuenteReal.

Una vez creado el Script, será necesario añadirlo como componente al objeto Collider y arrastrar dentro del Inspector el modelo del puente en la ventana PuenteReal y modelo del esquema del puente en la ventana EsquemaPuente como se indica en la siguiente figura.

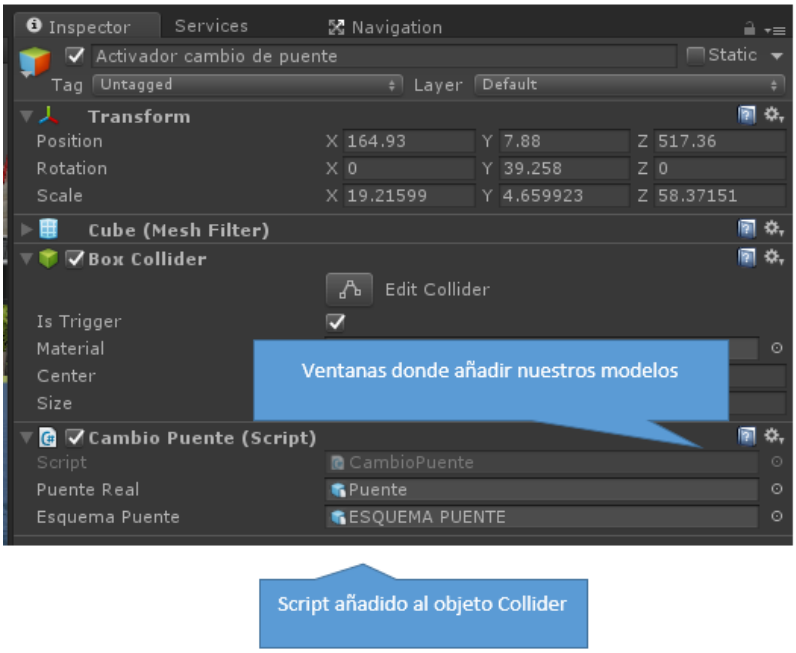

Figura 24. Componente Script añadida al objeto Collider

Una vez terminado el proceso, al ejecutar la escena, y tras haber desplazado hasta el otro lado del puente ya no se observará el puente real si no un esquema del mismo.

Figura 25. Esquema del puente visto desde el otro lado del paseo

4.5.8. Botones

En el presente apartado se introducirá el uso de los objetos tipo botón los cuales permiten interactuar con la escena.

4.5.8.1. Creando el panel con los botones

El botón en Unity viene integrado como un elemento UI (User Interface) por lo que para insertar un botón en la escena será necesario tener un lienzo. Una vez creado el lienzo a través de la pestaña Create > IU > Canvas se añade un botón asociado al lienzo a través de Create > IU > Button. Para ello puede utilizarse como lienzo el prefabricado generado en apartados anteriores. Para desplazar los botones hasta el comienzo del puente basta con mover el lienzo ya que se sitúan sobre él, lo mismo pasa con las dimensiones. Es importante recordar que si se quiere que el botón aparezca como un objeto físico en la escena, es necesario seleccionar la opción World Space en el Render Mode del lienzo.

El objeto botón en Unity está diseñado para iniciar una acción cuando el usuario hace clic y lo suelta. Si el mouse se mueve del control del botón antes de que el clic se haya soltado, la acción no toma lugar.

En el Inspector una vez seleccionado el botón se observa el siguiente componente:

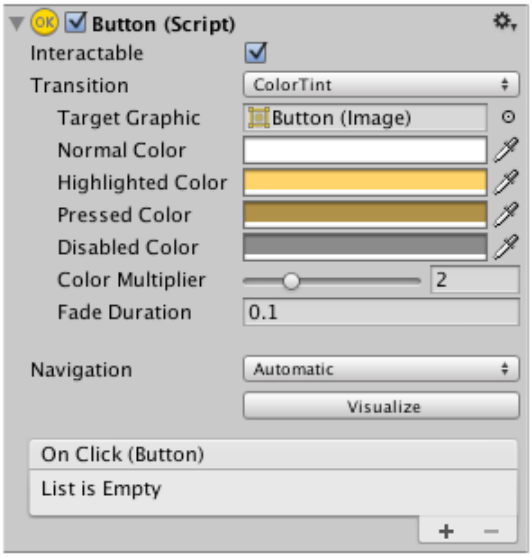

Figura 26. Script asociado al objeto button

Compuesto de de los siguientes atributos:

- *Interactable*: Esta opción sirve para definir si el botón va a ser interactivo o no.

- *Transition*: Determina la manera en la que el control responde visualmente a las acciones del usuario. Hay varias opciones de transición dependiendo en qué estado se encuentre el botón. Los diferentes estados son: normal, highlighted, pressed y disabled. Ofrece cuatro opciones:

    - *None*: Esta opción sirve para que no se realice ninguna transición, es decir, que no se diferencie el estado del botón.

    - *Color Tint*: Cambia el color del botón dependiendo de en qué estado se encuenrte. Es posible seleccionar el color para cada estado individual. También es posible configurar la Fade Duration (Duración de desvanecimiento) entre diferentes estados. Cuanto mayor sea el número, más lento será el desvanecimiento entre los colores.

    - *Sprite Swap*: Permite que diferentes sprites se muestren dependiendo de en qué estado se encuentre el botón. Los sprites pueden ser personalizados.

    - Animation: Permite crear animaciones dependiendo del estado del botón, para utilizar esta opción es necesario crear un componente animator.

- *Navigation*: Propiedades que determinan la secuencia de controles.

Para el ejemplo práctico se ha elegido la transición basada en una animación ya que servirá como base para introducirlas. Las animaciones son muy interesantes ya que permiten crear un recorrido predefinido dentro de la escena, es decir, si se quiere que alguien pueda visualizar el resultado a través de las gafas de Realidad Virtual pero que no tenga la capacidad de moverse libremente se puede crear un recorrido predefinido por la escena.

Para crear una animación, que sirva como transición en los botones, se selecciona la opción Animation. Una vez seleccionada Unity proporciona la opción de autogenerar una animación. Si se pincha en la opción, el programa pide una ruta dónde se guardará

la animación. Se recomienda crea una carpeta llamada Animations dentro de la carpeta Assets. Una vez seleccionada la ruta es necesario darle alguna característica a la animación. El primer paso consiste en abrir la ventana Animation en Window>Animation. Una vez abierta y con el botón seleccionado en el árbol se puede seleccionar el estado del botón para el que se quiera crear la animación como se muestra en la siguiente figura:

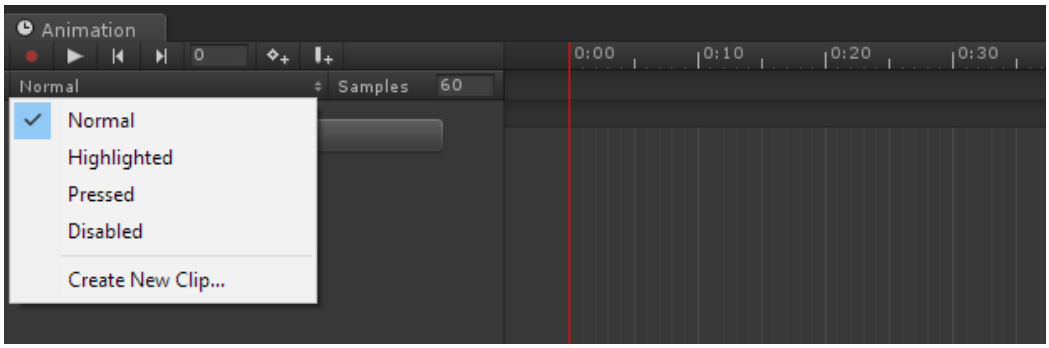

Figura 27. Ventana Animation y fases de la transición

Para el estado Normal no se creará una transición, pero sí para el estado Highlighted que se activará cuando el puntero se encuentre sobre el botón. Para crear la transición simplemente es necesario hacer clic sobre el botón record (el punto rojo de la izquierda) y cambiar en el Inspector alguna propiedad del botón. Para el ejemplo se ha cambiado el ancho y la altura para que cuando el puntero esté sobre el botón éste se haga más grande. Para la transición Pressed se ha optado por hacerlo aún un poco más grande. De esta manera cuando el puntero se encuentre sobre el botón éste se hará más grande y cuando se pulse será un poco más grande aún.

Una vez creada la transición se cierra la ventana Animation y se puede comenzar a definir qué acción realizará el botón al ser pulsado. El objeto botón que aporta Unity tiene un solo evento llamado “On Click” que responde cuando el usuario completa un clic.

Para el ejemplo se ha utilizado un panel con dos botones, uno de ellos transforma el material de los tirantes del puente para que éstos sean verdes y otro devuelve el color rojo inicial a los mismos.

Figura 28. Tablero con botones creado para el ejemplo práctico

A partir de este ejemplo se sentará la base para la interacción con los objetos de la escena a través de la mirada.

Pare construir el panel con los botones se ha utilizado el lienzo creado en el apartado 4.5.4. Como ubicación se ha escogido la entrada del puente por ser un lugar desde dónde se apreciará el cambio de color de los tirantes. Una vez colocado el lienzo se le añaden dos objetos tipo botón a través de Create > IU > Button asegurándose de que éste objeto es dependiente del lienzo. Es importante darle una escala adecuada tanto al lienzo como a los botones. En caso de que se pierda la perspectiva se recomienza ejecutar la escena, colocarse las gafas e ir ajustando el tamaño hasta quedar satisfecho.

Para el ejemplo práctico, como se puede ver en la figura de arriba, se ha añadido al lienzo dos botones, un título y un texto. Los objetos tipo botón también tienen asociado un texto que por defecto tiene la palabra Button escrita. En el ejemplo se han adaptado los botones tanto en color como en texto para aclarar al usuario que ocurrirá cuándo pulse los botones y se les ha añadido una animación para los diferentes estados.

4.5.8.2. Activando los botones con la mirada

Una vez creado los botones y las animaciones para los diferentes estados del mismo, es necesario lograr que dichos botones se activen cuando se les mira. Una forma de conseguir esto consiste en superponer sobre los botones dos objetos con un Collider que serán interceptados por el rayo coincidente con el centro de la visión.

Como se ha escogido una forma rectangular para los botones del ejemplo, se añadirán dos objetos tipo cubo a los que se les quitará el mallado para dejarlos únicamente con la componente Collider.

Para que el rayo pueda distinguir estos objetos del resto en la escena, se creará una nueva capa donde se colocarán únicamente estos dos objetos. Para ello es necesario, una vez seleccionado los cubos, ir a su Inspector y en la parte superior del mismo, donde nos deja seleccionar el Tag, añadir uno nuevo llamado Button seleccionando la opción “Add Tag…” como se muestra en la siguiente figura.

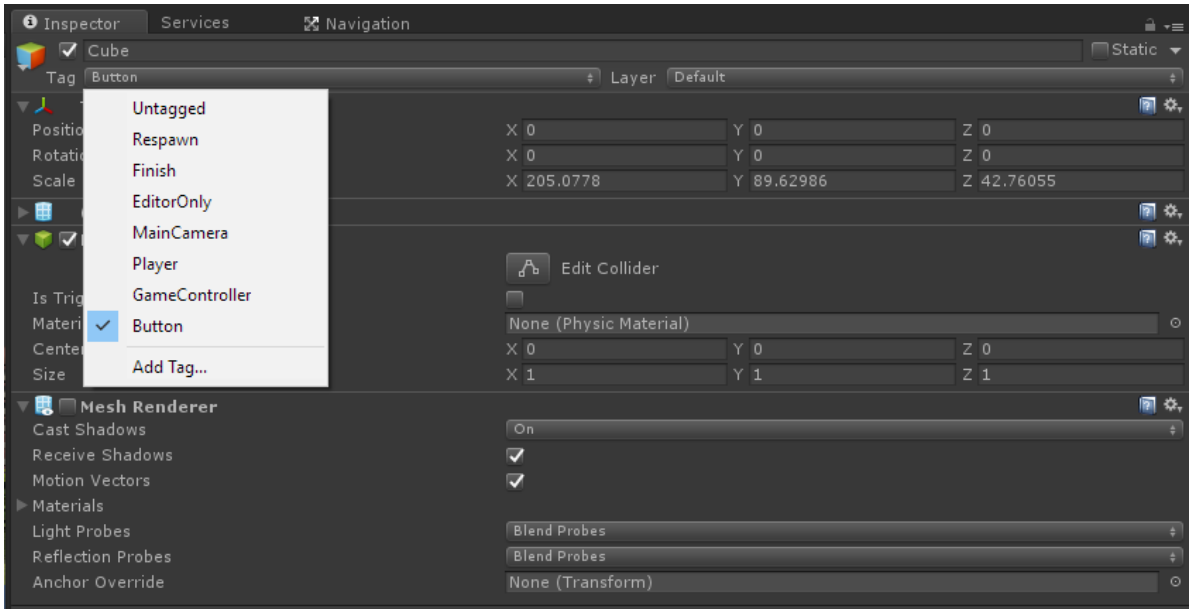

Figura 29. Cómo añadir una nueva capa para que el rayo de visión la distinga del resto

Una vez añadido simplemente es necesario ubicar ambos objetos en esa nueva capa creada.

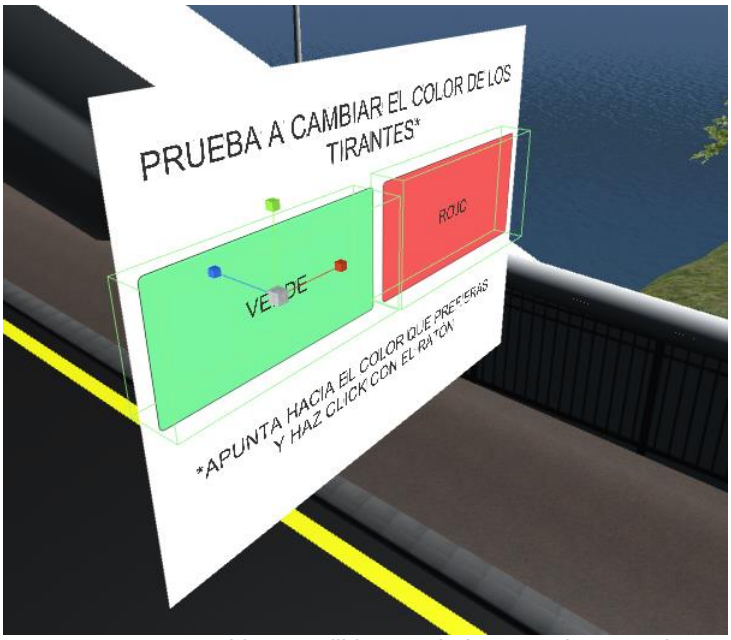

Figura 30. Botones con sus Objetos Collider asociados para detectar el rayo de visión

Una vez montado el panel con los botones animados y los Collider cúbicos en la capa Button es necesario generar un script que marque los botones cuando el centro de visión se posa sobre ellos. Para el ejemplo práctico se programado el siguiente código al que se le ha llamado ButtonExecute:

```
using UnityEngine;
using UnityEngine.EventSystems;
using System.Collections;


public class ButtonExecute : MonoBehaviour
{

    private GameObject currentButton;


    void Update()
    {

        Transform camera = Camera.main.transform;
        Ray ray = new Ray(camera.position, camera.rotation * Vector3.forward);
        RaycastHit hit;
        GameObject hitButton = null;
        PointerEventData data = new PointerEventData(EventSystem.current);

        if (Physics.Raycast(ray, out hit))
        {
            if (hit.transform.gameObject.tag == "Button")
            {
                hitButton = hit.transform.parent.gameObject;
            }
        }
```

```
        if (currentButton != hitButton)
        {
            if (currentButton != null)
            { //deseleccionar/unhighlight
                ExecuteEvents.Execute<IPointerExitHandler>(currentButton, data,
ExecuteEvents.pointerExitHandler);
            }
            currentButton = hitButton;
            if (currentButton != null)
            {//seleccionar/highlight
                ExecuteEvents.Execute<IPointerEnterHandler>(currentButton, data,
ExecuteEvents.pointerEnterHandler);
            }
        }
    }
}
```

El funcionamiento del script es el siguiente:

- Primero se declara privada la variable tipo objeto referente al botón.

- Luego se crean unas variables para referirse a la cámara, al rayo de visión central, y al objeto sobre el que colisionará así como la propia colisión. Tras lo que se define el rayo en función de la posición de la cámara y la colisión de éste con un cualquier elemento.

- Finalmente se define el comportamiento del botón en caso de que el rayo colisione con un objeto ubicado en la capa Button. En cada frame (cada unidad de tiempo) se capta el rayo que parte del centro de visión de la cámara y se comprueba si ha encontrado un objeto de la capa Button. En caso de que lo haga se marca al padre del elemento (el botón en sí) como seleccionado (highlight).

- De la misma forma detecta si el rayo ha salido del objeto para deseleccionarlo (unhighlight).

4.5.8.3. Seleccionando los botones al activar una tecla.

Se ha conseguido que se marque el botón cuando se enfoca hacia él. Para que el panel formado por botones sea funcional debería de activarse cuando se presiona una tecla del ratón, teclado o mando. Para ello se creará una clase genérica llamada Clicker que comprobará si el usuario ha presionado alguna tecla. Esto se puede lograr con el siguiente código:

```
using UnityEngine;
using System.Collections;

public class Clicker
{
    public bool clicked ()
    {
        return Input.anyKeyDown;
    }
}
```

Si, por el contrario, se quiere que se active únicamente cuando se pulsa una tecla en concreto, simplemente será necesario especificar qué tecla en el script.

Una vez creada ésta clase, necesitamos añadir las siguientes líneas al script ButtonExecute:

```
using UnityEngine;
using UnityEngine.EventSystems;
using System.Collections;


public class ButtonExecute : MonoBehaviour
{

    private GameObject currentButton;
     private Clicker clicker = new Clicker();


    void Update()
    {

        Transform camera = Camera.main.transform;
        Ray ray = new Ray(camera.position, camera.rotation * Vector3.forward);
        RaycastHit hit;
        GameObject hitButton = null;
        PointerEventData data = new PointerEventData(EventSystem.current);

        if (Physics.Raycast(ray, out hit))
        {
            if (hit.transform.gameObject.tag == "Button")
            {
                hitButton = hit.transform.parent.gameObject;
            }
        }

        if (currentButton != hitButton)
        {
            if (currentButton != null)
            { //unhighlight
                ExecuteEvents.Execute<IPointerExitHandler>(currentButton, data,
ExecuteEvents.pointerExitHandler);
            }
            currentButton = hitButton;
            if (currentButton != null)
            {//highlight
                ExecuteEvents.Execute<IPointerEnterHandler>(currentButton, data,
ExecuteEvents.pointerEnterHandler);
            }
        }
        if (currentButton != null)
        {
            if (clicker.clicked())
            {
                ExecuteEvents.Execute<IPointerClickHandler>(currentButton, data,
ExecuteEvents.pointerClickHandler);
```

```
        }
    }
  }
}
```

Esto logra que si el botón está seleccionado y el usuario presiona alguna tecla se presiona el botón.

Por último, es necesario definir qué pasará cuando el botón sea presionado. El objeto UI Button de Unity ofrece una gran cantidad de opciones en función del tipo de objeto sobre el que se quiera actuar. Para el ejemplo se ha decidido cambiar el material de un objeto para que el usuario pueda observar un cambio en el color de los tirantes.

Para añadirle esta opción es necesario seleccionar el botón y acudir a su Inspector. En él encontraremos una sección llamada "On Click ()" que permite añadir órdenes para cuando se presione el botón. Acudiendo al botón "+" añadimos el objeto sobre el que queremos actuar, en este caso el grupo de objetos que forman los tirantes de puente. Unity ofrece una lista de acciones que se pueden hacer sobre el mismo como se muestra en la siguiente figura:

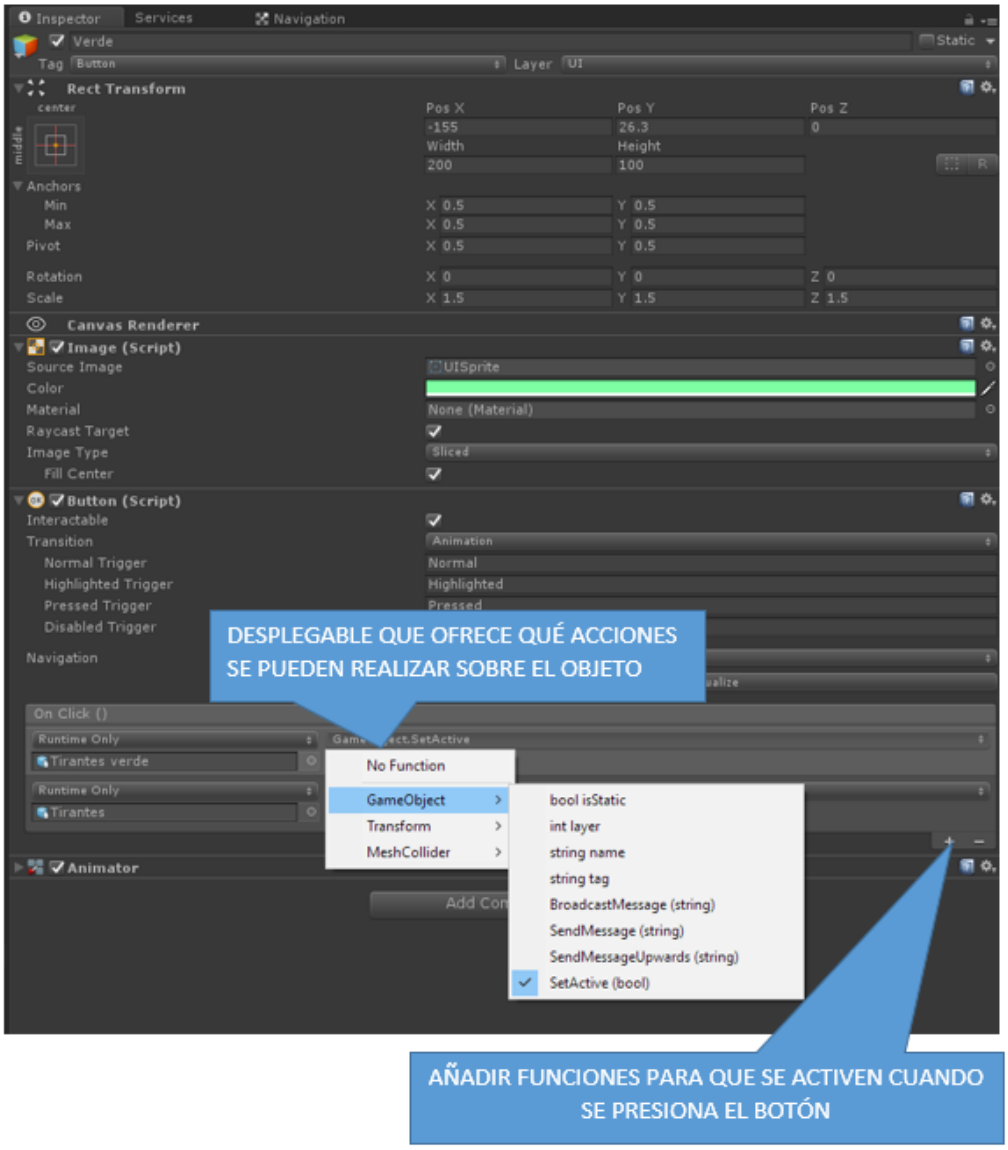

Figura 31. Inspector de un objeto UI Button y acciones permitidas por la clase On Click ()

Entre ellas podemos elegir, por ejemplo, MeshCollider > Physicmaterial y escoger una propiedad física a añadir sobre el material del objeto cuando se presione el botón.

Para el propósito elegido (cambiar el color de los tirantes) una opción sencilla es duplicar el grupo de objetos formado por todos los tirantes y simplemente renombrarlo como “tirantes verdes” y cambiarles el material. Una vez duplicado simplemente ordenaremos a través de los botones que se active o se desactive un grupo.

Para el primer botón, el verde, se activarán el grupo de tirantes verdes y se desactivará el grupo de tirantes rojos. Par el segundo botón lo contrario.

Tras estas operaciones se dispone de un tablero con botones con los que se puede interactuar a través de la mirada. Es momento para ejecutar la aplicación y explorar el funcionamiento del mismo y las sensaciones que despierta.

4.5.8.4. Añadir un puntero en el centro de la visión

Al ejecutar la escena se ha comprobado que resulta complicado saber cuál es el centro de la visión para poder apuntar con exactitud hacia los botones.

Una solución muy simple a este problema es añadir, como objeto dependiente de la cámara, un lienzo con una imagen centrada de tamaño muy pequeño y situar el lienzo en la posición (0,0,2) es decir en el centro de la visión alejado dos metros de los ojos sobre el eje z. Esto permite disponer de un puntero en el centro de la visión que facilita mucho la tarea de apuntar a los botones.

Otra manera sencilla de disponer de un puntero es añadir a la cámara un objeto UI tipo Cuad, que no es más que un objeto bidimensional, y darle una escala muy reducida, del orden de 0.01, para posteriormente posicionarlo, al igual que el lienzo, en la posición (0,0,2).

Como éste puntero es útil exclusivamente cuando se entra en contacto con el tablero que contiene los botones, se puede añadir una Activador, al igual que para los carteles interactivos, que active el puntero cuando nos acercamos a los botones. Para ello añadimos un objeto con un Collider, con la opción “Is Trigger” activada, alrededor del cartel con los botones, le añadimos el script ActivarTexto y arrastramos el puntero a la ventana que pregunta por el objeto que se quiere activar al entrar en contacto. Con ello se habrá conseguido que el puntero se active cuando se entra en contacto con el Collider.

De la misma forma se puede añadir otro script del mismo tipo al Activador para que active el tablero con los botones, de manera que cuando nos acercamos al puente éste se active junto con el puntero.

Con ello, habremos logrado crear un cartel con dos botones que se activan al mirarlos y que se seleccionan al pulsar una tecla. Dicho cartel aparecerá cuándo el usuario se acerque al puente junto con un puntero que facilite la tarea de apuntar a los botones. Cuando el botón verde sea seleccionado los tirantes del puente cambiarán a ese color, por el contrario, cuando el botón rojo sea seleccionado estos volverán a su estado inicial.

4.6. CREAR UN PASEO INTERACTVO POR LA ESCENA

Por último, se creará un paseo predefinido por la escena. Esto permitirá al usuario controlar la visión del personaje, pero no el movimiento del mismo. Para ello es necesario crear una animación, del mismo modo que con los botones, pero esta vez sobre el personaje principal.

El primer paso es seleccionar el personaje en el árbol del editor y acudir a la ventana Animation. Debido a que sobre el personaje aún no se ha creado una animación, Unity ofrecerá crear una nueva a través del botón “Create”, que aparecerá en el centro de la ventana Animation. Una vez creada la animación podemos elegir las propiedades del objeto sobre las que se quiere crear una animación, para el ejemplo queremos actuar sobre la posición del personaje luego será la propiedad Transform > Position.

En la ventana Animation se crearán en eventos o “keys” sobre la línea de tiempo. La línea de tiempo permite establecer la temporalidad de la animación y se mide en frames y en segundos. La línea vertical roja permite seleccionar el frame que se quiere modificar. Al lado de la línea de tiempo hay dos botones, el de grabar y el de reproducir. Cuando el botón de grabar está seleccionado los cambios hechos sobre el objeto deseado en la escena serán agregados automáticamente a la animación en el frame seleccionado. El botón reproducir permite pre-visualizar la animación. La propiedad “samples” permite definir cuántos segundos equivalen a un frame de la animación. Reducir éste número hace que la animación sea más lenta. Para no generar una sensación de malestar en el usuario reduciremos éste número a seis.

Para crear la animación será necesario escoger el frame en el que queremos que nuestro personaje alcance la nueva posición y mover el personaje principal a dicha posición con el botón grabar seleccionado. Al realizar esta operación se creará un evento o “key” en la animación que registrará la posición a alcanzar en dicho frame.

Figura 32. Ventana para crear una animación en Unity

Para crear el recorrido crearán varios “keys” en los puntos por lo que pasará el personaje principal. Es importante, una vez creado el recorrido, ejecutar la escena y comprobar la velocidad con la que se mueve el personaje. Si ésta fuese excesiva será necesario separar las “keys” para que el personaje se mueva más lento.

Una vez creada la animación cuando se ejecute la aplicación el personaje se moverá sin que el usuario realice ninguna acción. Cuando la animación esté activa, el usuario

tendrá control sobre la visión, es decir, tendrá la libertad de mirar para cualquier dirección, pero no sobre el movimiento.

Esta opción puede ser interesante a la hora de crear una visita virtual a una obra para, por ejemplo, el cliente al que se le quiere vender la obra o para un alumno que quiera entender el funcionamiento de una estructura. Además, al crear un paseo el diseñador puede asegurarse de que el usuario encontrará todos los carteles y objetos interactivos de la escena.

4.7. LA REALIDAD VIRTUAL Y LA CINETOSIS

La cinetosis es la sensación de mareo provocada por el movimiento. La Realidad Virtual puede llegar a producir éste efecto sobre el usuario por lo que es muy importante tenerlo en cuenta a la hora de diseñar los modelos y las aplicaciones orientada al uso de esta tecnología.

Éste fenómeno se debe a que, cuando una persona mueve la cabeza, su cerebro espera que el entorno cambie exactamente en sincronía, por lo que, si se genera algún retraso perceptible, éste puede llegar a crear una sensación de incomodidad en el usuario. La latencia será menor cuánto mejor sea la velocidad de renderizado por cada frame. Para el uso de dispositivos de Realidad Virtual se recomienda una alta velocidad de frame por segundo. En general, éste problema incumbe tanto a la calidad de los procesadores como a la de los dispositivos de Realidad Virtual. En este campo ha habido grandes avances y se espera aún mayores progresos en el futuro.

A mismo tiempo, los desarrolladores de aplicaciones que funcionen a través de dispositivos de Realidad Virtual, tendrán que tener en cuenta este problema de latencia y otras causas de cinetosis, ya que no sólo ésta sensación se produce por culpa de la tecnología.

A la hora de desarrollar una aplicación que mejore la experiencia del usuario y su seguridad se deberá de tener en cuenta que:

- La velocidad con la que se mueve el personaje por la escena debe de ser moderada.

- La sensación de mareo disminuye si el usuario mira hacia adelante.

- La velocidad de giro de la cabeza debe de ser modera ya que el problema de latencia aumenta cuánto más rápido sea el cambio de posición de la cámara.

- Es necesario añadir al entorno referencias visuales que ayuden al usuario a estar orientado tales como horizontes u objetos cercanos.

- Por último, es esencial prestar especial atención al renderizado. Como se ha visto durante el desarrollo del proyecto, algunos objetos, texturas o reflejos no están adaptados al uso de dispositivos de Realidad Virtual, por lo que es fundamental eliminarlos si se quiere ofrecer una buena experiencia al usuario.

4.8. CONSTRUYENDO LA APLICACIÓN

Una vez terminado el modelo, Unity permite crear una aplicación para diferentes plataformas a través de File >Building Settings. La ventana de Build Settings permite escoger la plataforma de destino, ajustar los parámetros para la construcción de la misma y empezar el proceso de construcción.

Una vez se haya escogido la plataforma objetivo y los ajustes, simplemente es necesario añadir la escena o escenas que formarán parte de la aplicación a través del botón “Add Open Scenes” y seleccionar el botón “Build” para crear aplicación.

4.8.1. Creando una aplicación para PC

Una vez se haya creado el modelo con todas las características deseadas, es hora de generar la aplicación.

Para ello, es necesario escoger la plataforma objetivo de la aplicación. Para el ejemplo se ha escogido PC por lo que se escogerá en la lista “Platform” PC, Max & Linux tras lo que aparecerá en la pestaña “Target Platform” Window x 86_64.

Además, Unity ofrece las siguientes opciones:

- *Development Build: que* permite al desarrollador trabajar sobre la aplicación mientras ésta se construye.

- Autoconnect Profiler: cuando la opción Development Build sea seleccionada, permite al profiler estar conectado a la construcción.

- Script Debugging: si la opción Development Build se selecciona, el código script puede ser depurado.

Antes de generar la aplicación será necesario habilitar VR en Unity, si no se ha hecho antes, para que la aplicación admita el uso del dispositivo de Realidad Virtual que se pretenda utilizar con la aplicación. En el caso del presente proyecto el set Oculus Rift.

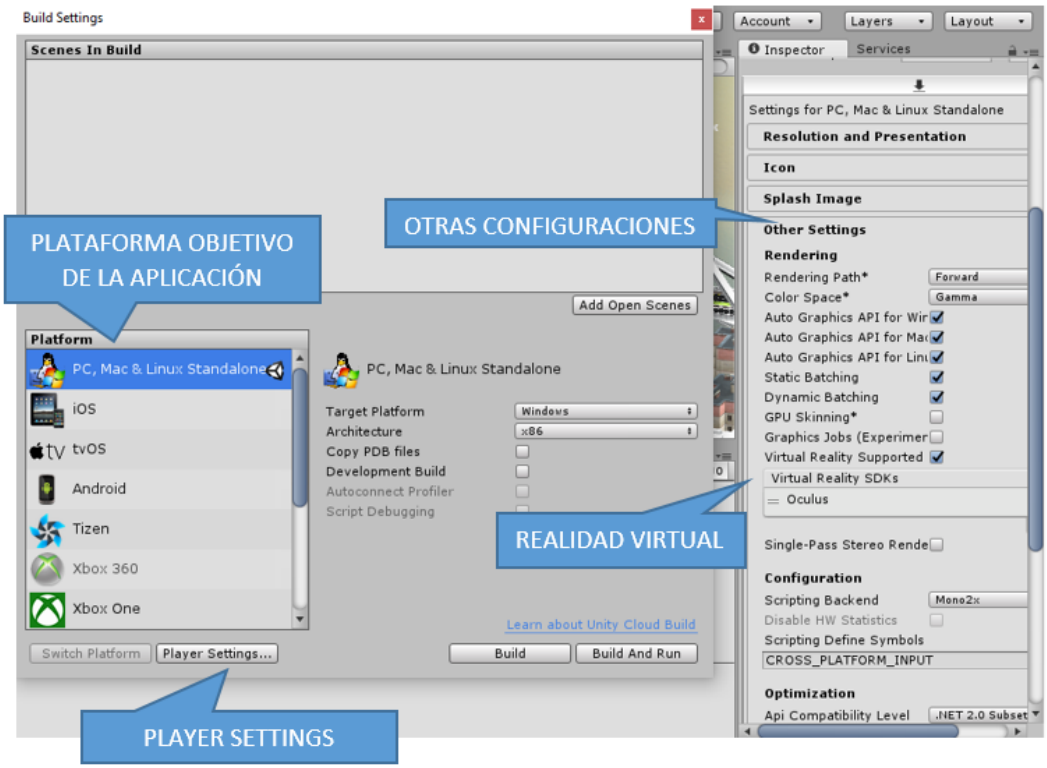

Figura 33. Ventana Build Settings

Para lograrlo es necesario abrir la ventana de Player Settings ubicada en File < Building Settings y seleccionar el botón Player Settings como se observa en la figura superior. Al seleccionarlo, en la pestaña del Inspector se muestra una lista de características entre las que se encuentra una opción de “Virtual Reality Supported”. Una vez activada, el programa preguntará por el dispositivo de Realidad Virtual que será utilizado.

4.8.2. Creando una aplicación para web

La versión más reciente de Unity no permite generar un aplicaicón web que soporte el uso de dispositivos de Realidad Virtual pero se puede montar una aplicación destinada a mostrar la escena a través de la pantalla del ordenador en un navegador Web.

Para ello es necesario escoger como plataforma de trabajo WebGL. Si se acude a las Player Settings, Unity permite personalizar la pantalla de inicio de la aplicación. Para el presente proyecto se ha escogido como nombre de la compañía “E.T.S. Caminos, Canales y Puertos. Universidad de Cantabria” y como título de la aplicación “Virtualización del Puente de la Barqueta para Realidad Virtual”, además se ha añadido como icono el logo de la Escuela.

La aplicación WebGL está disponible en la página web: cuartas.es/tfm-moa

5. CONCLUSIONES

La Realidad Virtual ofrece, sin duda, una infinidad de ventajas a la hora de aproximarnos a una estructura entre las que destacan:

- Ayuda al diseño y toma de decisiones en la fase de proyecto y construcción de una obra.

- Venta del diseño al cliente o a los ciudadanos, que podrían tener un voto sobre el proyecto.

- Herramienta didáctica a la hora de entender el funcionamiento de cualquier estructura compleja. Puede ofrecerse una visión global de la estructura y su funcionamiento. En el ejemplo práctico llevado a cabo en el proyecto, se ha representado tanto la estructura real como un esquema tridimensional del funcionamiento del puente. Incluso puede ofrecer una visión práctica de cómo construir una estructura.

- Herramienta turística o histórico-didáctica para ofrecer un recorrido interactivo por las grandes infraestructuras del mundo.

El objetivo principal del proyecto consiste en explorar las alternativas para virtualizar una obra e interactuar con ella en tres dimensiones, analizando el alcance y posibilidades que puede ofrecer dicho proceso para los Ingenieros Civiles.

La conclusión más destacada es que el motor de renderizado en tiempo real multiplataforma Unity 3D, es una herramienta muy útil para importar modelos tridimensionales a un entorno virtual y para interactuar con dicho entorno a través de la Realidad Virtual.

Durante el desarrollo del proyecto, se ha podido comprobar que el proceso para importar un modelo tridimensional a un entorno que nos permita visualizar dicho modelo a través de la Realidad Virtual es relativamente sencillo. A su vez, se ha investigado la forma en la que incorporar un controlador que pueda moverse por la escena sin crear una sensación de malestar al usuario. Por otro lado, se han desarrollado mecanismos para interactuar con el modelo, lo que ha resultado ser un proceso más complejo que requiere, además, de conocimientos en programación.

El presente proyecto sirve, por tanto, como base para cualquier persona que quiera introducirse en la Realidad Virtual, ofreciendo un guía para crear aplicaciones que incluyan terreno y estructuras reales y diversas formas de interactuar con los objetos que forman parte del entorno virtual creado.

Por último, cabe destacar que las empresas que se dedican a la fabricación de dispositivos de Realidad Virtual, como Oculus Rift, prevén un aumento significativo en el número de clientes no asociados a los videojuegos, es decir, otros sectores están mostrando interés por esta tecnología. Dentro del campo de la Ingeniería, no sería extraño esperar que los grandes desarrolladores de software de modelado 3D, como Autodesk o Solidworks, desarrollen en un futuro aplicaciones que permitan el manejo de sus modelos en tres dimensiones, evitando así el proceso necesario para importar sus modelos a otras plataformas con el objetivo de interactuar con ellos en tres dimensiones.

6. BIBLIOGRAFÍA

BIBLIOGRAFÍA

Bradley Austin Davis, Karen Bryla, Phillips Alexander Benton. 2015. *Oculus Rift in Action.* Shelter island : Manning Publications Co., 2015. ISBN 9781617292194.

Centro Nacional de Información Geográfica - Ministerio de Fomento. Centro de Descargas. [Online] [Cited: 27 08 2016.] http://centrodedescargas.cnig.es/CentroDescargas/catalogo.do.

Cuartas Hernández, Miguel. 2014-2015. Unity 3D. *Crar un terreno real en Unity 3D.* [Online] 2014-2015. [Cited: 07 08 2016.] http://cuartas.es/unity/practica4.pdf.

GRAPHISOFT. 2016. https://www.graphisoft.es/. *Acerca de BIM.* [Online] GRAPHISOFT SE., 2016. [Cited: 08 08 2016.] https://www.graphisoft.es/archicad/open_bim/about_bim/.

Hed Tracking for the Oculus Rift. **Steven M. La Valle, Anna Yershova, Max Katsev, Michael Antonov. 2014.** Hong Kong : IEEE International Conference on Robotics & Automation (ICRA), 2014.

Linowes, Jonathan. 2015. *Unity Virtual Reality Projects.* Birmingham : Packt Publishing Ldt., 2015. ISBN 978-1-78398-855-6.

steveluscher. 2005-2016. The history of VR. *Mozzilla Developer Network.* [Online] 21 06 2005-2016. [Cited: 05 09 2016.] https://developer.mozilla.org/en-US/docs/Web/API/WebVR_API/WebVR_concepts.

Unity Technologies - Unity Store. 2016. Unity Store. *https://store.unity.com/es/.* [Online] 2016. [Cited: 2016 07 05.]

Unity Technologies. 2016. Learn with Unity. *https://unity3d.com/es/learn* [Online] 2016. [Cited: 2016 julio]