



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



Escola Tècnica
Superior d'Enginyeria
Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Un Visor Web de Modelos 3D

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Javier Guillot Sillas

Tutor: Eduardo Vendrell Vidal

2016-2017

Agradecimientos

Agradecer todo el apoyo y ayuda que he recibido por parte de mi familia, amigos y compañeros de carrera, ya que sin su constante apoyo y dedicación no habría sido posible escribir estas palabras.

Agradecer también a la Universidad y el profesorado, por dedicar su tiempo y esfuerzo en compartir sus conocimientos para, no solo formar futuros ingenieros sino, algo más importante, formar personas que pronto representarán el futuro de nuestra sociedad. En especial, agradecer a Eduardo Vendrell su paciencia, dedicación, confianza y tiempo invertido en este proyecto.

Resumen

El trabajo consiste en el desarrollo de un visor de modelos tridimensionales operativo sobre un navegador web. El visualizador mostrará modelos 3D contenidos en una base de datos junto con otra información de referencia. Los modelos deberán estar en formato OBJ o STL. El visor permitirá al usuario examinar el modelo seleccionado desde diferentes puntos de vista, junto con otras utilidades como zoom o herramientas de medición. También se mostrará la información relacionada con el modelo, ya sea formato de texto, o en otros formatos multimedia. El visor debe funcionar sobre los principales navegadores existentes.

Palabras clave: visor, modelos 3d, formatos 3d, web, Unity.

Abstract

The goal of this work is the development of an operating display for three-dimensional models on a web browser. The display will shows 3D models contained in a database along with other reference information. The models must be in OBJ or STL format. The display will allow the user to examine the selected model from different viewpoints, along with other features such as zoom and measurement tools. Information related to the model, either plain text, or other media formats is also displayed. The display should work on major existing web browsers.

Keywords: Visor, 3D Models, 3D Formats, web, unity.

Tabla de contenidos

Introducción	9
Objetivos	11
Realidad Virtual y Realidad Aumentada (RV/RA).....	12
Realidad Aumentada.....	12
Definición y propiedades de la RA.....	13
Tipos de Realidad Aumentada (RA)	15
Aplicaciones de la Realidad Aumentada	21
Realidad Virtual	24
Características de la Realidad Virtual (RV)	25
Tipos de Realidad Virtual	26
Mundos virtuales	27
Diferencia entre RA y RV	28
Aplicaciones de la Realidad Virtual.....	28
Debate ético	32
Motores gráficos	34
Unreal Engine.....	34
CryEngine.....	35
Unity 3D.....	36
Bases de datos	38
SQLite	38
MySQL	38
MongoDB.....	38
Desarrollo e implementación del proyecto	40
Descripción	40
Análisis de las herramientas.....	41
Unity3D.....	41
Visual Studio.....	47
SQLite	48
PHP.....	48
HTML	49
XAMPP	49
Arquitectura software	49
Lógica de la aplicación	49



Capa de persistencia.....	50
Implementación	51
Aplicación Unity	51
Diseño Web	64
Instalación y utilización de XAMPP	68
Conclusiones.....	71
Posibles mejoras, trabajos futuros y aplicaciones	71
Bibliografía.....	73
Enlaces web	75
Anexo I: Manual de usuario	76
Menú Principal	76
Menú Upload	76
Menú Modelos	77
Aplicación	78
Controles.....	79
Imágenes.....	79
Textos	80
Videos	80
Anexo II: Plan de trabajo	81

Tabla de ilustraciones

<i>Ilustración 1: Continuum RV entre realidad y virtualidad</i>	13
<i>Ilustración 2: Aplicación AR Faces</i> (http://play.google.com/store/details?id=com.dacorp.arface).....	14
<i>Ilustración 3: Aplicación Alba Visualizer</i> (http://www.arquimaster.com.ar/web/wp-content/uploads/2014/07/alba_visualizer2.jpg)	15
<i>Ilustración 4: Captura del videojuego Kingdom Hearts III</i> (Cortesía de Disney Interactive).....	35
<i>Ilustración 5: Captura del videojuego Ryse: Son of Rome</i> (Cortesía de Crytek)	36
<i>Ilustración 6: Captura del videojuego Ori and the Blind Forest</i> (Cortesía de Game Informer) ..	37
<i>Ilustración 7: Ventana principal del editor de Unity</i>	42
<i>Ilustración 8: Ventana de juego</i>	43
<i>Ilustración 9: Pestaña desplegada de Assets</i>	44
<i>Ilustración 10: Pestaña desplegada de GameObject</i>	45
<i>Ilustración 11: Propiedades del componente Transform</i>	45
<i>Ilustración 12: Propiedades del componente BoxCollider</i>	45
<i>Ilustración 13: Propiedades del componente Rigidbody</i>	46
<i>Ilustración 14: Propiedades del componente Audio Source</i>	46
<i>Ilustración 15: Propiedades del componente Audio Listener</i>	46
<i>Ilustración 16: Diagrama de flujo de la aplicación</i>	50
<i>Ilustración 17: Capa de persistencia</i>	51
<i>Ilustración 18: Código "Zoom.cs"</i>	52
<i>Ilustración 19: Código "Controles.cs"</i>	53
<i>Ilustración 20: Código "Video.cs"</i>	55
<i>Ilustración 21: Ventana del inspector seleccionando una Movie Texture</i>	56
<i>Ilustración 22: Canvas Video desactivado</i>	56
<i>Ilustración 23: Código "Imagen.cs"</i>	57
<i>Ilustración 24: Ventana del inspector del script "Imagen.cs"</i>	58
<i>Ilustración 25: Botón Anchor Presets</i>	58

<i>Ilustración 26: Ventana del inspector del script "Menus"</i>	59
<i>Ilustración 27: Ventana Jerarquía del proyecto</i>	60
<i>Ilustración 28: SQLite Manager</i>	61
<i>Ilustración 29: Fragmento de código de "DB Manager.cs", método GetModel()</i>	62
<i>Ilustración 30: Fragmento de código de "DB Manager.cs", método ShowButtons()</i>	63
<i>Ilustración 31: Código "DB.cs"</i>	64
<i>Ilustración 32: Diseño PSD de la web</i>	65
<i>Ilustración 33: Código CSS</i>	66
<i>Ilustración 34: Script JavaScript para redimensionar botones</i>	67
<i>Ilustración 35: Panel de control de XAMPP</i>	69
<i>Ilustración 36: Modificar PHP.INI en XAMPP</i>	70
<i>Ilustración 37: Menú Principal</i>	76
<i>Ilustración 38: Jerarquía Uploads</i>	77
<i>Ilustración 39: Menú modelos/Menú Principal</i>	77
<i>Ilustración 40: Ventana de la aplicación</i>	78
<i>Ilustración 41: Aplicación con la opción medir activada</i>	78
<i>Ilustración 42: Galería de imágenes</i>	79
<i>Ilustración 43: Aplicación mostrando texto</i>	80
<i>Ilustración 44: Aplicación mostrando video</i>	80
<i>Ilustración 45: Plan de trabajo</i>	81

Introducción

En este comienzo de siglo, están adquiriendo una fuerte presencia las diferentes tecnologías emergentes (gamificación, computación en la nube, analíticas de aprendizaje, MOOC, “*internet of things*”, entornos personales de aprendizaje, impresoras 3D...) gracias a diversos acontecimientos, desde la reducción de costes de los equipos hasta la fuerte incursión de los dispositivos móviles, que ha fomentado la posibilidad de acceso a la información y a la red independientemente del lugar donde uno se encuentre y ha influido en la deslocalización de las tecnologías. Todo ello está repercutiendo en la aparición de una nueva escenografía tecnológica como nunca antes había ocurrido en la historia de la humanidad.

Y en este contexto, una de esas tecnologías emergentes son la realidad virtual (Virtual Reality, VR) y la realidad aumentada (Augmented Reality, RA), que se están empezando a proliferar en diversos sectores: la publicidad, el ocio, la información turística, el mundo de la moda o la industria.

El sector que más aprovecha estas nuevas tecnologías es el de contenido multimedia, cuya principal finalidad es la de ofrecer entretenimiento. No obstante, existen muchas otras áreas de conocimiento que también se pueden ver favorecidas de las posibilidades que ofrecen estos dispositivos: arqueología, medicina, arquitectura, o incluso el ejército.

En este proyecto, se propone utilizar estas nuevas tecnologías para realizar una aplicación web que permita visualizar e interactuar modelos 3D de una base de datos.

Esta memoria comenzará mostrando los objetivos propuestos en la realización de este proyecto y que se desarrollarán a lo largo de ella.

En el segundo capítulo se abordarán las tecnologías de Realidad Virtual y Realidad Aumentada, los diferentes tipos de RA en función del componente físico utilizado, el componente visual o su funcionalidad, y sus diferentes usos y aplicaciones.

En el tercer capítulo se dedicará al concepto de motor de videojuego, repasando los principales motores actuales y centrándonos en Unity, un motor de videojuego multiplataforma creado por Unity Technologies con el que se ha propuesto desarrollar la aplicación web de visualización de modelos 3D.

Las bases de datos formarán parte de este proyecto, por lo que se analizarán los principales gestores de bases de datos y se mostrarán sus ventajas e inconvenientes.

El quinto capítulo trata sobre la aplicación web objeto de este proyecto. Partiendo de la definición de recurso web, se describirá por completo la aplicación y cada una de sus partes, entrando en detalle en el funcionamiento y organización de la base de datos que hay detrás de la aplicación.

En el penúltimo capítulo del proyecto se ofrecerán al lector las conclusiones de este proyecto: qué se ha realizado, qué se ha quedado sin realizar y qué se mejorará en futuras versiones.

El último capítulo constará de una extensa bibliografía utilizada para la redacción de esta memoria.

Para finalizar, el proyecto consta de dos apéndices. El primer apéndice, es un ejemplo de uso de la aplicación, donde como si de un manual de usuario se tratara, se explica al lector el funcionamiento de la aplicación paso a paso y con referencias visuales. El segundo apéndice es un plan de desarrollo del proyecto, donde a través de un diagrama de Gantt se explica el coste temporal de la realización de este proyecto.

Objetivos

El objetivo principal de este proyecto es desarrollar un producto que permita al usuario acceder a una base de datos online de modelos 3D, seleccionar el modelo que desee visualizar y poder operar con él en el visualizador 3D. Dada esta premisa, se abordan los siguientes problemas: ¿qué acciones puede llevar a cabo el usuario? ¿qué tipo de modelos se deben utilizar? ¿cómo se dispone la información adicional al usuario? Lo principal es que el usuario pueda visualizar el modelo y realizar los movimientos característicos e imprescindibles de un visor 3D, es decir: rotar la cámara alrededor del objeto y realizar zoom.

Se propone que los modelos que pueda leer el visor sean .OBJ o .STL de entre los diferentes tipos de formatos de modelos 3D que podemos encontrar.

Además, el visor tendrá que disponer al usuario de información relacionada con el modelo, ya sea en formato texto como en formato multimedia, por lo que tendremos que plantearnos la forma de poder mostrar dichos elementos dentro de la aplicación y qué elementos puede mostrar: texto, imágenes, video, etc.

Para situarnos en contexto, haremos una amplia introducción teórica a la Realidad Aumentada y Realidad Virtual, dos avances tecnológicos que son la base de este proyecto y que cada vez tienen más calado en la sociedad actual.

En definitiva, el objetivo es crear una aplicación que resulte útil y extrapolable, capaz de innovarse continuamente, añadir nuevas funcionalidades dando total libertad a cualquier desarrollador que quiera continuar con esta labor, siempre adoptando las bases de mejora, con el fin de otorgar un mejor servicio al que cualquier persona sea capaz de acceder y beneficiarse de su contenido.

Realidad Virtual y Realidad Aumentada (RV/RA)

En este apartado se hará una amplia explicación teórica de la Realidad Aumentada y la Realidad Virtual, que servirán como base teórica de este proyecto.

Realidad Aumentada

En una primera aproximación se puede decir, siguiendo a los profesores Julio Cabrera Almenara y Fernando García Jiménez que la Realidad Aumentada (en adelante RA) es *“la combinación de información digital e información física en tiempo real a través de diferentes dispositivos tecnológicos”*.

La RA comenzó siendo considerada una modalidad de la Realidad Virtual (en adelante RV) y desde el primer sistema de RA creado por I Sutherland en 1968 diferentes autores han ido conceptualizando esta tecnología.

El primer autor en utilizar el calificativo de “aumentada” fue Tom Caudell en 1990.

En 1994, Paul Milgram y Fumio Kishino enuncian el continuo virtualidad-realidad en el que estaría incardinada la RA. Establecían un vector de doble dirección entre los entornos real y virtual, situando la realidad aumentada más cercana al plano de lo real que al de lo virtual.

En 1997, Azuma insiste en la diferenciación entre RA y RV, aduciendo que, mientras la inmersión en la RV impide al sujeto ver el mundo real, en RA el usuario ve el mundo real con objetos digitales superpuestos. Para Azuma las características definitorias de la RA son:

- a) combinación de lo real y lo virtual
- b) interacción en tiempo real
- c) ubicación en el mismo sistema de coordenadas 3D que la realidad.

Bimber en 2005 subraya la importancia del mundo real en la RA, ya que se establecen nexos basados en una relación espacial con el objeto virtual, más allá de la mera relación visual propia de dispositivos como los HDM (Head Mounted Display), característica ya puesta de manifiesto por Azuma.

Como ya se vió anteriormente, Milgram y Kishino, en su formulación, definen un *continuum* donde podemos ubicar cualquier fenómeno artificial: el continuo realidad – virtualidad (ilustración 1).

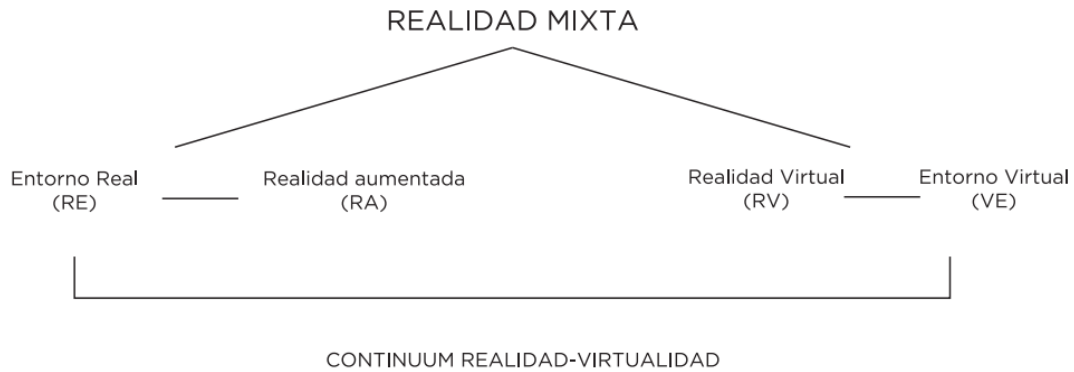


Ilustración 1: Continuum RV entre realidad y virtualidad

Se trata de, en lugar de concebir virtualidad y realidad como conceptos opuestos, concebirlos como los extremos de un continuum. En un extremo, el entorno virtual (VE) el sujeto se ve inmerso en un mundo sintético gracias a una pantalla especial, donde las propiedades de lo percibido (tiempo, gravedad, solidez...) pueden emular o no las propias del mundo físico. En el otro extremo, el entorno real (RE), donde la experiencia no está mediatizada por ninguna pantalla y todo lo percibido se sujeta a las leyes de la física. Pues bien, en un punto entre tales extremos se situaría la RA.

Autores como Hughes, Fuchs y Nannypieri (2011) afirman que la RA es una subclase de la realidad mixta que puede ser vista como un nuevo espacio por asociación entre un entorno virtual y uno real.

Definición y propiedades de la RA

Habiendo llegado a este punto y siguiendo de nuevo a Julio Cabrera Almenara y Fernando García Jiménez, se puede definir la realidad aumentada como un *“tipo de realidad mixta formada por la integración coherente con la realidad física y en tiempo real de una capa de información digital que puede ser diversa (texto, símbolos, audio, vídeo y/o objetos tridimensionales) y con la que es posible la interacción, con el resultado de enriquecer o alterar la información de la realidad física en la que se integra.”*

De esta definición se derivan las siguientes propiedades:

- 1) **Realidad mixta:** Los recursos generados con RA se sitúan en un punto del continuo virtualidad-realidad como una realidad híbrida en la que se mezcla la percepción de lo físico (del mundo real podríamos decir) y de los elementos digitales mezclados. Esta mezcla puede consistir en:
 - a. Una superposición
 - b. Una inclusión
 - c. Una sustitución del entorno circundante o de fondo

- 2) **Integración coherente en tiempo real:** Los elementos visuales digitales se alinean con los elementos activadores de la realidad física (*marcadores o trackables*) guardando una relación espacial con estos, de forma que el marcador y el elemento digital están sujetos a las mismas coordenadas espaciotemporales. Es decir, si el marcador gira o se desplaza, es acompañado por el elemento virtual. Esta propiedad diferencia la RA de otras tecnologías como los fondos digitales (*chroma key*) ya que son meras superposiciones de elementos gráficos digitales sobre el campo visual. Da cuenta de la importancia de esta propiedad el hecho de que, al inventor de la tecnología de tracking visual, Hirozaku Kato, se le conoce como el padre de la RA.
- 3) **Diversidad de la capa de información digital:** Cualquier elemento digital puede formar parte de la capa digital vinculada al marcador. Por tanto, podemos utilizar texto, gráficos, objetos 3D, vídeo, audio, URL...para superponer a la realidad física, de manera que permitan la percepción conjunta de la realidad, sin taparla

Esta necesidad de no tapar completamente la realidad física es la que hace que los códigos QR, que maximizan el navegador y ocultan la realidad que se muestra a través de la cámara del dispositivo, no sean considerados como realidad virtual por muchos autores o que, otros, la clasifiquen en el nivel cero.

- 4) **Posibilidad de interacción:** Sobre los elementos de la capa digital pueden definirse eventos que permitan la interacción con los usuarios o entre los propios objetos. Así, un objeto 3D puede ser rotado, o ampliado, un vídeo puede pararse o reproducirse y pueden incorporarse botones gráficos que desencadenen toda una serie de acciones. Las posibilidades de interacción dependerán del tipo de interacciones que permita el dispositivo de entrada (teclado, pantalla táctica, mouse, comando de voz...)
- 5) **Enriquecimiento o alteración de la información:** Hay que tener en cuenta que podemos desplegar capas de información digital sobre la realidad física que no supongan un enriquecimiento de la información. Estaríamos ante un caso de RA solo desde un punto de vista técnico.

Pongamos un ejemplo: Si un cuadro de Felipe V se vincula a un marcador que contiene un texto con una breve biografía de dicho rey, estaríamos enriqueciendo la información. Si este mismo marcador se vincula a un vídeo con los últimos goles de la selección española de fútbol, no estaríamos ante un enriquecimiento coherente de la realidad.

Esto ha llevado a algunos autores como Nannypieri (2011) a plantearse que pueda existir realidad aumentada sin intención de incremento informativo. Estaríamos ante una RA cuya intencionalidad fuera solo lúdica (Ilustración 2) o meramente instrumental (Ilustración 3).



Ilustración 2: Aplicación AR Faces (<http://play.google.com/store/details?id=com.dacorp.arface>).

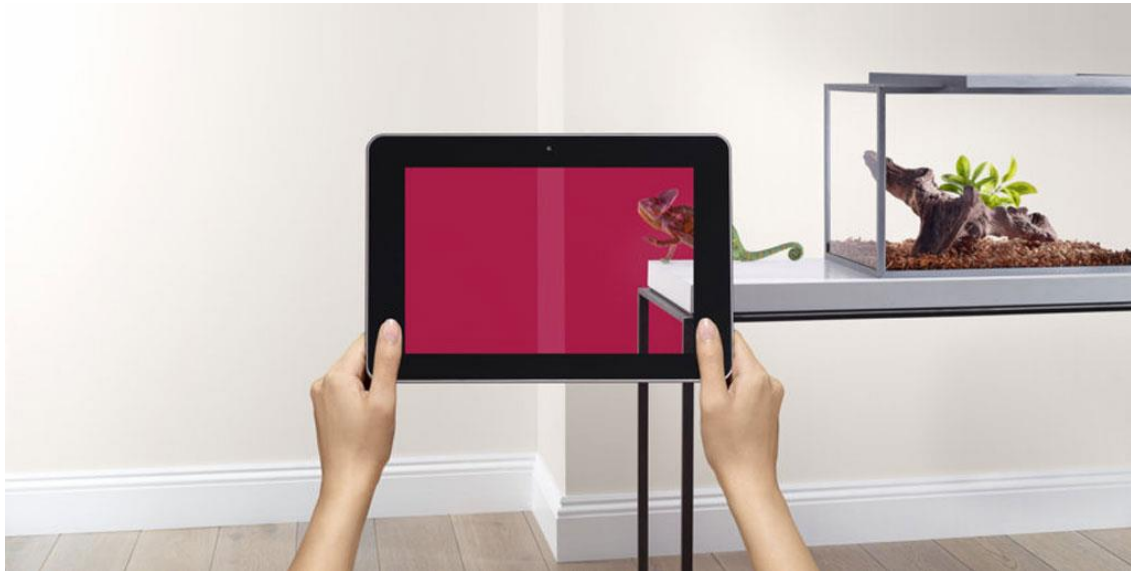


Ilustración 3: Aplicación Alba Visualizer (http://www.arquimaster.com.ar/web/wp-content/uploads/2014/07/alba_visualizer2.jpg)

A la vista de las aplicaciones y recursos que ofrece la red y el mercado de aplicaciones, así como la literatura sobre este tema, podemos observar que el concepto que se suele manejar de RA es flexible, no exigiéndose la presencia simultánea de todas las características. Sin embargo, muchos autores piensan que al menos, en una tecnología determinada, deben encontrarse siempre las tres primeras propiedades (realidad mixta, integración coherente y diversidad de la capa de información digital) y el resto de propiedades (posibilidad de interacción y enriquecimiento de la información) al menos, en potencia.

Tipos de Realidad Aumentada (RA)

Cuando se habla de realidad aumentada, se está hablando de un fenómeno que se construye a partir de tres componentes: la realidad física, la realidad virtual y la programación.

A partir de aquí, se pueden distinguir diferentes tipologías de realidad aumentada. La primera teniendo en cuenta las características del componente físico, la segunda en base a las características del componente virtual y, en tercer lugar, en base a la funcionalidad del entorno RA.

A. Tipos de realidad aumentada según el componente físico.

El componente físico que activa la información digital se denomina también marcador, activador o trackable. Según este, se pueden distinguir cinco niveles de realidad aumentada

- 1) **Nivel 1 de RA. Un patrón artificial en blanco y negro** (podría ser en color, pero la tecnología de Tracking responde a la escala de grises).

En los patrones visuales de RA no hay datos codificados para ser leídos (como en un código QR) sino que, debido a su alto contraste, sirve de referencia geométrica donde anclar el componente virtual. Suelen consistir en marcas cuadradas o circulares según el sistema de tracking empleado.

Llegados a este punto se puede volver sobre el tema del código QR del que ya dijimos que no era realidad aumentada. Pues bien, algunos autores lo clasifican en un hipotético nivel 0 de RA. En un código QR se almacena información que maximiza un navegador.

2) Nivel 2 de RA. Una imagen

Este nivel, también conocido como tracking sin marcas, tiene varias posibilidades. Utilizar una imagen, una imagen extendida (panorámica) o un rostro.

3) Nivel 3 de RA. Una entidad 3D

Como la realidad física tiene tres dimensiones, la entidad virtual que mejor formaría con esta una realidad mixta es un objeto o escenario 3D.

4) Nivel 4 de RA. Un punto del planeta determinado por sus coordenadas GPS (Global Positioning System)

Vinculadas a las coordenadas GPS de un punto del planeta, podemos asociar información diversa sobre dicho punto: si en él hay un edificio de interés o un servicio público, distancia entre ese punto y otro de interés.

5) Nivel 5 de RA. La huella termal

La compañía Metaio presentó una tecnología capaz de vincular entidades virtuales a la huella de calor que los dedos de un usuario dejen sobre una superficie.

Esta tecnología funciona a base de dos cámaras, de las cuales una capta la huella termal. El sistema capta las diferencias de temperatura y registra el rastro de calor que dejan los dedos al tocar una superficie y combina este rastro con una información digital.

Aunque el sistema tiene algunas dificultades prácticas (por ejemplo, que pasa en lugares del planeta donde la temperatura ambiente supera los 37º), desde un punto de vista conceptual añade un nuevo nivel a la RA ya que introduce un nuevo componente de la realidad al que vincular la RA.

B. Tipos de realidad aumentada según el componente virtual.

En primer lugar, hay que aclarar que cuando se habla de componente virtual se hace referencia a objetos digitales perceptibles, no a la programación que afecta a estas entidades digitales para obtener la interacción de los objetos entre si y entre los objetos y el usuario.

Matizado esto, interesa saber que, según la naturaleza del componente virtual se puede hablar de cinco tipos de RA.

1) RA basadas en imagen.

Las imágenes suelen ser mapas de bits como png o jpg. Hay que destacar que si la imagen tiene zonas transparentes resulta más fácil integrarla con la realidad física. Por ello, es más versátil la utilización de formato png que admite el canal con información de opacidad (*canal alpha*) que la utilización de formato jpg.

Un caso especial, sería la utilización de imágenes panorámicas en 360º que dan al sujeto la impresión de estar dentro de un paisaje o estancia. En puridad, no estaríamos ante un caso de RA ya que no se mezcla con la realidad física. Sin embargo, sería un caso de realidad imposible según la clasificación de Hughes, Fuchs y Nannypieri (2011) que veremos dentro del apartado C.

2) RA basadas en 3D.

A su vez se puede dividir este tipo en dos: 3D estática y 3D animada:

a. Estática.

Se trata del formato wavefront (.obj), correspondiente a mallas de superficies poligonales recubiertas con una textura que pueden ser:

- i. *Naturales*: procedentes de objetos reales por medio de escaneado
- ii. *Artificiales*: creadas desde cero, procedentes de CAD o de exportación de softwares específicos
- iii. *Dicom*: como resultado de tecnologías de visualización de imágenes médicas, como por ejemplo una TAC (tomografía axial computarizada).

b. Animada:

Se admiten los formatos fbx y md2.

3) RA basadas en vídeo.

Los formatos más adecuados son mp4 y 3g2. El primero es el más adecuado para vídeos a pantalla completa, aunque, como ocurría con las imágenes panorámicas, al ocultar el escenario real, estamos perjudicando la experiencia de realidad mixta.

En el resto de los casos, es más adecuado el formato 3g2 que admite canal alpha y por tanto es posible hacer transparente el fondo.

Es interesante apuntar que, aunque visionemos una animación 2D que no proceda de la grabación de un suceso real sino que esté generado a través de una imagen sintética, seguiremos considerándolo vídeo, pues desde el punto de vista de su caracterización como objeto digital se trata del mismo tipo de archivo.



4) RA basadas en audio.

En este caso, los formatos más adecuados son los comprimidos como, por ejemplo, el MP3. El audio es útil para incorporar explicaciones con fines turísticos, publicitarios o formativos.

5) RA multimedia.

Se trata de combinar objetos virtuales de diferente formato en un mismo escenario. Por ejemplo, combinar un objeto 3D con un vídeo dotado de transparencia (*video alpha*).

C. Tipos de realidad aumentada según su funcionalidad.

Esta tipología es obra de Hughes, Fuchs y Nannypieri (2011), los cuales manejan un concepto de RA más extenso que el que en este trabajo hemos definido, por lo que, algunas de las subclases que definamos, como la realidad documentada, no tendrían todas las características anteriormente enunciadas.

Comenzando con la taxonomía funcional de Hughes, Fuchs y Nannypieri el entorno de RA puede ir orientado a aumentar la percepción o a crear un entorno artificial.

A) Funcionalidad: percepción aumentada.

Se trata de que la RA constituye una herramienta para la toma de decisiones. Proporciona información que permite una mejor comprensión de la realidad con el fin último de optimizar nuestra acción sobre ella. Esta funcionalidad de aumentar la percepción de la realidad puede dividirse en cinco subfunciones.

1. Realidad documentada y virtualidad documentada.

El primer caso sería una lavadora y su manual de instrucciones y el segundo un cuadro sinóptico de un proceso industrial en el que se incorporan ventanas que muestran partes del proceso en tiempo real, pero sin que constituya un entorno virtual. En esta subfuncionalidad la entidad virtual (EV) y la imagen real (IR) residen en pantallas de diferente naturaleza o ubicación.

2. Realidad con percepción o comprensión aumentadas.

En esta subfuncionalidad la entidad virtual y la imagen real residen en la misma pantalla. Se distinguen dos niveles:

- i. realidad con comprensión aumentada. La comprensión de las imágenes reales se aumenta incrustando información semántica pasiva en el objeto real
- ii. realidad con visibilidad aumentada. Se trata de mejorar la claridad del objeto real destacando sus contornos externos.

3. Asociación perceptual de lo real y lo virtual.

- i. Primero: incrustación de objetos virtuales en imágenes reales (mera superposición)
- ii. Segundo: integración de objetos virtuales en imágenes reales

4. Asociación comportamental de lo real y lo virtual.

Esta funcionalidad define interacciones geométricas y físicas entre objetos reales y virtuales. Como ejemplo una pelota virtual golpeada por lo real.

5. Sustitución de lo real por lo virtual o realidad virtualizada.

Se trata de, conociendo el modelo geométrico que define la escena real que se observa, reemplazarla alternativamente por la imagen sintética del modelo.

B) Funcionalidad: creación de un entorno artificial

En este caso, con la realidad aumentada se producen entornos sin consecuencias prácticas. Tres son las subfuncionalidades que podemos distinguir en este apartado: Imaginar la realidad que podría ser el futuro, imaginar la realidad que pudo ser en el pasado e imaginar una realidad imposible.

1) Imaginar la realidad que podría ser en el futuro asociando lo real con lo virtual.
Se trataría, por ejemplo, de integrar en una habitación real muebles virtuales para ver como quedarían. También puede darse el proceso contrario: integrar en un ambiente virtual elementos reales.

2) Imaginar la realidad que fuera en un pasado asociando lo real con lo virtual.
Estamos hablando de asociar objetos virtuales que representan lo que ya no existe en un entorno real u objetos reales existentes en un entorno que ya no existe (reconstruido digitalmente).

Un ejemplo nos lo ofrece la empresa sevillana Past View que, mediante unas gafas específicas, permite experimentar la Hispalis romana o la Isbilya musulmana.

3) Imaginar una realidad imposible.

En este caso el objetivo no es la comprensión de la realidad ni aumentar su percepción, sino la creación de una realidad mixta imposible con un fin artístico, lúdico o estético.

Otra forma de clasificar la realidad aumentada es la sugerida por Domingo Sánchez Blazquez y M. Carmen Juan Lizandra en su proyecto Desarrollo de una aplicación de realidad aumentada para simulación de moléculas, de la Universitat Politècnica de

València (<http://hdl.handle.net/10251/10240>), que las clasifican según cómo son las interfaces de interacción con la aplicación:

- **Interfaces tangibles basadas en el uso de marcadores.** Ishii et al. definen las interfaces tangibles como aquellas interfaces donde, gracias al emparejado de información digital con entornos y objetos físicos del día a día, aumentan el mundo físico real. En estas interfaces, el usuario manipula un elemento real que tiene instalado un marcador, y los resultados de dichos movimientos son reflejados en los del objeto virtual correspondiente.
- **Interacción basada en movimiento corporal.** Una forma característica de interacción en sistemas de RA radica en la detección y seguimiento del movimiento de algún miembro del cuerpo. Existen diferentes tecnologías para realizar el seguimiento de la posición y orientación de las manos, entre otras, tracking inercial, tracking magnético, sistemas de reconocimiento basados en visión, etc.
- **Interacción basada en dispositivos de bajo coste.** Diversos sistemas de RA hacen uso de dispositivos fácilmente accesibles en el mercado que utilizan como dispositivos de interacción a los que incorporan sistemas de tracking. Ejemplos de éstos citadas en el trabajo referenciado son: “cubo con pegatinas reflectantes de infrarrojos, una pieza de material médico accesible en mercado, usando como métodos de entrada y de salida una pizarra y una PDA diseñados de tal forma, que el sistema de tracking es capaz de detectar su posición y orientación sobre un mapa usando visión por computador, espejo virtual (dispositivo de medicina comercial) con infrarrojos, usando el Wiimote de la consola Wii de Nintendo, o usando teléfonos móviles.”
- **Interacción multimodal.** Se denominan sistemas multimodales a aquellos que procesan métodos de entrada naturales (movimientos corporales, voz, guiños, táctil, gestos de mano) combinados con la salida multimedia del sistema.

También se puede clasificar por cómo se muestran las imágenes:

- **Visores sobre la cabeza:** Se pone un dispositivo sobre la cabeza, Head Mounted Display (HMD), que muestra directamente las imágenes en los ojos a través de él.
- **Visualizador de mano:** El usuario lleva una pantalla pequeña que cabe en la mano y en el que va viendo, como puede ser un teléfono móvil o una videoconsola portátil.
- **Proyector espacial:** Realidad Aumentada Espacial (SAR) hace uso de proyectores digitales para mostrar información gráfica sobre los objetos físicos en lugar de que el usuario use o lleve consigo la pantalla.

Otra catalogación posible que se muestra en el trabajo, atendiendo a distintos criterios:

- **Según el entorno físico en el que se desarrolla la aplicación**, se pueden diferenciar entre sistemas dentro de recintos cerrados («indoors») y sistemas al aire libre o abiertos («outdoors»). La diferencia principal entre aplicaciones para dentro de recintos cerrados y las aplicaciones para el aire libre condiciona muchos aspectos, sobretodo el tipo de dispositivo de registro y el tipo de displays a utilizar, de los sistemas de RA establecidos.
- **Según la movilidad de los dispositivos de registro y/o displays**, se pueden distinguir entre sistemas móviles («mobile») y sistemas espaciales («spatial»). En los sistemas móviles generalmente el usuario lleva consigo los dispositivos de registro, el display e, incluso, el ordenador o PDA que gestiona la aplicación. Estas aplicaciones son usuales, aunque no restrictivas, de espacios abiertos.
- **Según el número de usuarios que simultáneamente pueden interactuar con el sistema**, se pueden distinguir entre sistemas individuales («individual») y colaborativos («collaborative»). El sistema es colaborativo si, como mínimo, existe la posibilidad de que participen dos o más usuarios de forma simultánea en la interacción, mientras que será individual, si tan solo existe la posibilidad de interacción de un único usuario.
- **Según el tipo de colaboración establecida** (para sistemas colaborativos), se pueden distinguir entre sistemas presenciales o cara a cara («face-to-face») y remotos («remote»). En las aplicaciones colaborativas, se distingue si los usuarios colaboran estando físicamente presentes en el mismo entorno, es decir, cara a cara, o por el contrario, se establece la comunicación a través de la red u otro sistema, siendo así la colaboración remota.
- **Según la extensión que abarquen**, se pueden distinguir entre sistemas locales («local») y sistemas ubicuos («ubiquitous»). Los sistemas locales se desarrollan en un ámbito acotado, bien en espacios abiertos o dentro de recintos.

Aplicaciones de la Realidad Aumentada

Son muchos los campos en que puede aplicarse la realidad aumentada y, de hecho, cada día se ven nuevas aplicaciones. Se van a analizar y poner ejemplos de los más importantes:

Educación y pedagogía

La utilización de realidad aumentada en el ámbito de la enseñanza es una de las aplicaciones más importantes y que más literatura y proyectos ha generado.



La utilización en educación de las diferentes tecnologías informáticas emergentes no es nueva. Así, no es de extrañar que se prevea que la penetración de la realidad aumentada en centros educativos y universitarios sea muy importante en un breve plazo.

El profesor Muñoz (Realidad aumentada una oportunidad para la nueva educación Comunicación y Pedagogía 277-278, 6-11) sintetiza las posibilidades que atesora la RA para su aplicación en educación y formación:

- 1) Crear, desarrollar y participar en gymkhanas y/o rutas georeferenciadas sobre cualquier materia del curriculum.
- 2) Generar y disponer de información sobre lugares, edificios, monumentos y otros entornos físicos
- 3) Traducción de textos sin necesidad de estar conectado a Internet
- 4) Crear o trabajar en el aula con libros que contengan objetos 3D
- 5) Aprendizaje basado en el descubrimiento
- 6) Desarrollo de habilidades en formación profesional
- 7) Juegos virtuales educativos, orientados tanto a la formación presencial como en e-learning
- 8) Desarrollar proyectos de ciudad generando informaciones en RA sobre puntos de interés del municipio
- 9) Cualquier otra cosa que se piense y que sea susceptible de utilizar información generada digitalmente

Las experiencias de RA se están desarrollando en cualquier nivel de la educación (primaria, secundaria, bachillerato...) pero, lógicamente, donde están alcanzando un gran nivel de penetración es en la Universidad. A continuación, relacionaremos algunos ejemplos:

- a) **Proyecto Magic Book.** Realizado por el grupo HIT de Nueva Zelanda. En él, a través de un dispositivo visual de mano (algo similar a las gafas 3D que se usan en el cine) se puede leer un libro real pero que incluye escenas de RA.
- b) **Proyecto Mentira.** Desarrollado por la Universidad de Wisconsin-Madison, es un proyecto que consiste en un juego a través del cual se pretende desarrollar las destrezas lingüísticas del castellano. En el juego se plantean una serie de conversaciones entre el jugador y los personajes ficticios con relación a un asesinato. Cada conversación está situada en un lugar y momento de la narrativa del juego, en algún lugar entre la realidad y la ficción.
- c) **Proyecto Care.** Se trata de un proyecto desarrollado por el University College of London (UCL) para crear un manual de prácticas a estudiantes de Enfermería. Está enfocado a mejorar la visualización del cuerpo humano, planificar operaciones y capacitar al personal sanitario en diversos procedimientos.
- d) **Proyecto AR.KEY:** La Universitat de València ha desarrollado un proyecto para mejorar las competencias clave (matemáticas y ciencia y tecnología) de trabajadores no cualificados del sector de la edificación mediante la utilización

de un sistema de formación, secuenciado en módulos, mediante RA, que simula el entorno real de una obra.

Videojuegos:

En 2015, la industria de los videojuegos movió unos 90.000 millones de dólares y, para 2018, podría superar los 113.300 millones, según datos de la firma Newzoo. Por tanto, desde un punto de vista económico, además de, por supuesto, técnico, las aplicaciones de la realidad aumentada en la industria de los videojuegos son muy importantes.

Daremos algunos ejemplos de videojuegos basados en la realidad digital.

- a) **Ingress** desarrollado por Niantic Labs para Nintendo. Existen dos facciones que luchan por el dominio de la tierra: la Resistencia y los Iluminati. El usuario puede pertenecer a uno de los dos bandos y conquistar en su nombre alguno de los puntos de interés (POI) esparcidos por el mapa del mundo, en el que nos desplazaremos gracias al GPS de nuestro móvil.
- b) **Parallel Kingdom AOT**. Uno de los videojuegos de RA con mayor éxito hasta la llegada de Pokemon Go. Se trata de un juego MMO (Mundo Masivo On-Line) de realidad aumentada, donde podemos conseguir premios, intercambiar objetos con otros jugadores, cazar monstruos y conquistar territorios
- c) **Invizimals**. La empresa española Novorama, fue capaz de innovar dentro de los juegos de captura gracias a la RA. En este juego el usuario era capaz de ver a los monstruos a través de la cámara de Playstation Portable (PSP) y capturarlos gracias a un dispositivo físico
- d) **Pokémon Go**. Aunque se ha dejado para el final, se trata del juego más popular en la actualidad, que ha trascendido los círculos de jugadores y ha sido objeto de noticia por las peripecias de sus jugadores llenando en masa Central Park en Nueva York o asaltando comisarías de policía para capturar pokémons. Desarrollado por Nintendo, utiliza el GPS y la cámara del teléfono móvil para que los jugadores puedan capturar los pokémons que ya hicieron furor en los años 90. No obstante no existe unanimidad a la hora de alinear el juego con la RA. Ken Perlin, profesor de informática y director fundador del Media Research Lab de la Universidad de Nueva York, hace una distinción entre simplemente dejar caer personajes digitales en una pantalla basándose en la ubicación de un jugador y la integración de esos personajes en su entorno para que parezcan más reales que virtuales. Prueba de la popularidad conseguida por este videojuego es que un reportaje sobre este juego ocupó la portada y las páginas 2, 3 y 4 del periódico Libération del viernes 26 de agosto de 2016.

Medicina:

Desde no hace tanto tiempo la Realidad Aumentada ha empezado a constituir una herramienta más para los profesionales para llevar a cabo su trabajo, gracias a que la



realidad virtual es combinada eficientemente con la realidad física. En medicina este tipo de tecnología se centra especialmente en áreas para la representación y visualización; concretamente el análisis de imágenes biomédicas, simulación de sistemas fisiológicos o entrenamiento en anatomía.

También permite recabar información del interior de un paciente, sin invadirlo, mediante resonancias magnéticas o realizar una reconstrucción que puede ser superpuesta sobre el cuerpo físico en tiempo real, lo que facilita la labor de los cirujanos.

Otro ejemplo de aplicación es el desarrollo, en la Escuela de Medicina de la Universidad de Washington de unas gafas de realidad aumentada que son capaces de distinguir las células cancerígenas de las sanas.

La Universitat Politècnica de València desarrolló un sistema de RA para tratar la fobia a pequeños animales (insectos, arañas...) y otro para tratar la fobia a las alturas.

Ámbito militar

Fue uno de los primeros campos en utilizar la RA. La imagen de ciencia ficción o de videojuego en que el soldado puede ver, superpuesta a la visión real, información sobre el entorno (ubicación de los compañeros, del enemigo, cómo es el terreno circundante...) está cada vez más cerca.

Dentro de este ámbito, se puede hablar también de los sistemas basados en RA para la navegación de aviones o de barcos. Existen sistemas que “pintan” de rojo los aviones enemigos y de verde los amigos o que proyectan información en la cabina de datos de navegación, meteorología...

Realidad Virtual

Valery Naranjo Ornedo del Departamento de Comunicaciones del Instituto Interuniversitario de Investigación en Bioingeniería y Tecnología Orientada al Ser Humano de la Universidad Politécnica de Valencia en su informe La realidad virtual al servicio del bienestar social define la Realidad Virtual (RV):

“Como la forma más avanzada de relación entre el ordenador y la persona, permitiendo al usuario interactuar con la máquina y sumergirse en un entorno generado artificialmente. Esta tecnología se basa en la generación interactiva multisensorial de estímulos con el objetivo de mantener la sensación completa de inmersión en un mundo real. Se caracteriza por la ilusión de participación en un entorno sintético, más que la observación de éste. El objetivo principal de un sistema de RV es la generación del sentido de presencia en el usuario, es decir, la experiencia subjetiva de estar en un lugar, incluso cuando físicamente se está localizado en otro. El sentido de estar en el entorno virtual en vez del lugar donde físicamente se encuentra el cuerpo físico del usuario se denomina presencia.”

Se pueden ver otras definiciones de RV:

- "Realidad virtual: un sistema de computación usado para crear un mundo artificial en el cual el usuario tiene la impresión de estar y la habilidad de navegar y manipular objetos en él". Manetta C. y R. Blade (1995)
- "La realidad virtual te permite explorar un mundo generado por computadoras a través de tu presencia en él". Hodder y Stoughton(s/a).
- "La realidad virtual es un camino que tienen los humanos para visualizar, manipular e interactuar con computadoras y con información extremadamente compleja". Aukstaklnis (1992)

Se podría definir la Realidad Virtual como un sistema tecnológico que pretende simular las percepciones sensoriales de forma que el usuario las tome como reales. Para ello se define lo virtual como algo que percibimos pero que no se corresponde con la realidad en ese espacio-tiempo (espejismo, grabación virtual, película...). Si se quiere que un usuario perciba algo virtual como real necesitaremos una interfaz que lo simule en tiempo real y le permita interactuar con él a través de múltiples canales sensoriales (visión, audición, tacto, olor, gusto).

El objetivo último de la realidad virtual es crear, almacenar y simular un mundo alternativo, modelar objetos en él, definir relaciones entre ellos y la forma en la que interactúan, para que el usuario pueda más tarde percibirlo.

Características de la Realidad Virtual (RV)

Las características principales, que se ven a continuación, suelen caracterizarse como las tres "i" de la realidad virtual

- **Inmersión.** El usuario pierde contacto con la realidad al percibir únicamente los estímulos del mundo virtual.
- **Interacción.** El usuario interactúa con el mundo virtual a través de dispositivos de entrada, de forma que modifica cosas en él y recibe la respuesta a través de sus sentidos. El objetivo último es la respuesta inmediata del mundo virtual (tiempo virtual = tiempo real). Dentro de esta característica podemos hablar de dos aspectos concretos: la navegabilidad y el posicionamiento del punto de vista del usuario. El primer aspecto es la habilidad del usuario para moverse alrededor del mundo con las restricciones que haya introducido el creador del software (se vuela o no, se puede correr o no...). En cuanto al posicionamiento, el usuario puede verse a sí mismo o ver la realidad desde diferentes puntos de vista.
- **Imaginación.** A través del mundo virtual podemos concebir y percibir realidades que no existen, de manera parecida a como hacemos con la creación artística.



Se pueden agregar otras características como:

- **Tridimensionalidad.** Tiene que ver directamente con la manipulación de los sentidos del usuario, principalmente la visión. Los componentes del mundo real se ofrecen al usuario en las tres dimensiones del mundo real, en cuanto a su ubicación espacial y los sonidos tienen direccionalidad (efecto estereofónico).
- **Tiempo real.** Esta característica quiere decir que la computadora tiene que hacer todos los cálculos necesarios y presentar una nueva imagen lo más rápido posible, lo que requiere grandes requerimientos computacionales.

Tipos de Realidad Virtual

La realidad virtual puede ser inmersiva o no inmersiva:

- La **inmersiva** se basa en la simulación de un ambiente tridimensional el cual el usuario percibe a través de estímulos sensoriales. En ella se utilizan elementos como cascos, guantes, gafas o trajes especiales
- La **no inmersiva** opta por la visualización de los elementos virtuales por una pantalla, dando opción de interactuar con otras personas a través de Internet. Normalmente la tecnología de Realidad Virtual no inmersiva es más barata y más familiar para el usuario.

Otros autores añaden un tipo intermedio: la RV semiinmersiva. En ella, se utilizan cuatro pantallas en forma de cubo (tres en las paredes y una en el piso). El usuario utiliza lentes y un dispositivo de cabeza.

Teniendo en cuenta las características de la RV, se puede desarrollar la siguiente tipología como encontramos en el libro *Educatrónica: innovación en el aprendizaje de las ciencias y la tecnología* de Enrique Ruiz Velasco Sánchez:

- **Cabina de simulación:** el ejemplo más común de este tipo de simulador es la cabina para el entrenamiento de aviadores. Generalmente la cabina recrea el interior del dispositivo o máquina que se desea simular. (un avión, un tanque etcétera), las ventanas de la misma se reemplazan por pantallas de computadoras de alta resolución, además existen bocinas estereofónicas que brindan el sonido ambiental y puede estar colocada fija o sobre ejes móviles. El programa está diseñado para responder en tiempo real a los estímulos que el usuario le envía por medio de los controles dentro de las cabinas.
- **Realidad Proyectada:** en este tipo de realidad virtual una imagen en movimiento del usuario es proyectada junto con otras imágenes en una extensa pantalla donde el usuario puede verse a sí mismo como si estuviese en la escena. En esencia los usuarios se miran ellos mismos como proyectados hacia el mundo virtual. Los usuarios pueden pintar diseños de colores en el aire, o hacer cualquier movimiento que el sistema reacciona en tiempo real. Un ejemplo

actual de este tipo de realidad virtual son los escenarios virtuales que se utilizan en ciertos programas de televisión.

- **Telepresencia:** término creado por Marvin Minsky que significa presencia remota, es un medio que proporciona a la persona la sensación de estar físicamente en otro lugar por medio de una escena creada por computadora. Es una experiencia psicológica que ocurre cuando la tecnología de simulación funciona lo suficientemente bien como para convencer al usuario de que está en un mundo virtual.
- **Realidad Virtual de Escritorio:** Un tablero de realidad virtual es una subinstalación del tradicional sistema de realidad virtual. En lugar de utilizar cascos para mostrar la información visual utiliza un monitor grande de computadora o un sistema de proyección. Algunos sistemas de este tipo permiten al usuario ver una imagen de tres dimensiones en sus monitores, pero utilizando lentes crystal eyes y pantalla de LCD o pantallas de cristal líquido.
- **Ventanas acopladas visualmente:** esta es la clase de sistema de inmersión que se asocia más a menudo con realidad virtual. Este sistema se basa en colocar las muestras directamente en frente del usuario, y conectando los movimientos de la cabeza con la imagen mostrada. Para lograr mayor acople la inmersión se logra con un casco (HMD) estereofónico, que posee sensores de posición y orientación que informan a la máquina la posición del usuario en todo momento, además de indicarle hacia donde está mirando. Un ejemplo de este tipo de realidad virtual son los juegos que hicieron popular esta técnica de computación los últimos años.

Mundos virtuales

Se puede decir que existen tres tipos básicos de mundo virtuales que pueden existir por separados como también mezclados entre sí:

- **Mundo Muerto:** es aquel en el que no hay objetos en movimiento ni partes interactivas, por lo cual sólo se permite su exploración. Suele ser el que vemos en las animaciones tradicionales, en las cuales las imágenes están precalculadas y producen una experiencia pasiva.
- **Mundo Real:** es aquel en el cual los elementos tienen sus atributos reales, de tal manera que, si miramos un reloj, marca la hora. Si pulsamos las teclas de una calculadora, se visualizan las operaciones que esta realiza y así sucesivamente.
- **Mundo Fantástico:** es el que nos permite realizar tareas irreales, como volar o atravesar paredes. Es el típico entorno que visualizamos en los videojuegos, pero también proporcionan situaciones interesantes para aplicaciones serias, como puede ser observar un edificio volando a su alrededor introducirnos dentro de un volcán, o simular la exploración de un planeta.



Diferencia entre RA y RV

Aunque ambas tecnologías son realidad mixta y muchas veces se confunden entre ellas, la principal diferencia es el nivel de inmersión del usuario. En la realidad virtual, el objetivo es crear una realidad sintética, artificial, en la que se sumerge completamente el usuario mientras que en la realidad aumentada se combina el mundo real con el virtual.

Dicho de otro modo, en el continuum realidad-virtualidad, la RV está más cerca del extremo virtual y la RA más cerca del extremo real.

Aplicaciones de la Realidad Virtual

En muchos de los ámbitos en que encontrábamos aplicaciones para la realidad aumentada, se encuentran también aplicaciones para la realidad virtual.

En primer lugar, decir que, en estos mismos momentos, la realidad virtual se está convirtiendo en una “realidad”, aunque parezca un juego de palabras. Las grandes empresas tecnológicas (Facebook, Samsung, Sony o HTC/Valve) han apostado fuerte por el desarrollo de la RV y, en 2016 van a invertir en ello miles de millones de dólares. Se van a ver algunas de las áreas en que se está desarrollando la RV y poner algunos ejemplos.

Videojuegos

Es uno de los sectores a los que más destinada está la Realidad Virtual. Se trata de que el jugador ya no maneje a un personaje, sino que él sea el personaje.

Se puede decir que todo empezó con la consola Nintendo Wii (Nintendo Co. Ltd., Kyoto, Japan), que permite interactuar con el juego realizando los mismos movimientos que si se estuviera en una situación real, por ejemplo, mover el brazo como si se estuviera jugando al tenis.

Más tarde apareció otro dispositivo, el Kinect, de Microsoft (Microsoft Corp., Redmond, Washington) el cual te posibilita manejar el juego con tu propio cuerpo, sin necesidad de ningún otro dispositivo.

Pero la introducción de la realidad virtual en los videojuegos no es solo cosa de grandes empresas, algunos de los mejores dispositivos han sido creados por pequeñas empresas y financiados por Kickstater, como las gafas Oculus Rift o el sensor Razer Hydra.

El desarrollo de los juegos de realidad virtual no es solo usado para el ocio, también pueden ser usados para estimular o rehabilitar al paciente, un proceso que en psicología se llama gamificación.

Aplicaciones militares

El gobierno de los Estados Unidos tiene un acuerdo con INTEL para que desarrolle para ellos una plataforma virtual para el entrenamiento de sus soldados para que sean capaces de tomar decisiones en situaciones reales.

Otro gobierno interesado en aplicar la realidad virtual a sus fuerzas armadas es el noruego. Mediante Oculus Rift, están desarrollando un proyecto para que, mediante la implementación de una serie de cámaras en la parte exterior de los tanques, los soldados que los conducen tengan una visión del exterior de 360°.

El manejo de drones y aviones no tripulados se realiza mediante técnicas de realidad virtual. Es posible bombardear un objeto militar en otro país desde una mesa en una instalación militar.

Medicina

Una de las aplicaciones en la medicina se encuentra en la simulación virtual del cuerpo humano. A partir de imágenes del cuerpo, se puede hacer la recreación en 3D del paciente, cosa que facilita la elaboración de un diagnóstico, o la simulación de operaciones en caso que sea necesario.

Otra aplicación es la utilización de la RV para la formación del personal sanitario. Mediante sistemas de realidad virtual, es posible que los estudiantes realicen operaciones virtuales para adquirir práctica antes de dar el salto a pacientes reales. También se puede aprender anatomía sin necesidad de utilizar cadáveres como hasta ahora.

Existen proyectos para curar el síndrome denominado “dolor del miembro fantasma”. El 70% de los pacientes que tienen algún miembro amputado siguen sintiéndolo porque el cerebro no sabe cómo interpretar la información de las terminales nerviosas del miembro amputado. Pues bien, con realidad virtual se puede enseñar al paciente a controlar estas terminaciones nerviosas.

También puede usarse realidad virtual para el tratamiento de niños autistas. Estos niños, que suelen responder bien a la utilización de la tecnología, aprenden de esta manera más rápido a adquirir habilidades sociales y a reaccionar frente a situaciones en las que puede verse a lo largo de su vida.

Otra aplicación de la RV en el ámbito sanitario es la realización de estudios. En este aspecto, la realidad virtual es tan útil porque con ella se pueden controlar todas las variables del ambiente, lo cual es imposible para las investigaciones y terapias tradicionales. Otra ventaja es que con la realidad virtual puede crearse el mismo ambiente para todos los participantes, de manera que los estudios realizados son altamente replicables. Además, de esta forma, la comparación entre pacientes o entre éstos y los controles, es más fiable pues eso asegura que todos los participantes han pasado por las mismas condiciones.



Psicología

El Virtual TDAH es una herramienta virtual para detectar el Trastorno por Déficit de Atención con Hiperactividad en niños de entre seis y doce años. Se trata de un test en el que los niños no se sienten evaluados y que elimina la subjetividad del padre o educador al observar el comportamiento del niño. TDAH: (<http://phobos.xtec.net/aletosa/virtual>)

José Gutiérrez Maldonado, que aplica la realidad virtual al tratamiento de la fobia a volar enumera, en un artículo aparecido en el periódico El País (Jueves, 8 de diciembre de 2005), las ventajas del uso de realidad virtual para este fin, frente a tratamientos tradicionales: es más económico, permite variar los parámetros (por ejemplo, pasar de un vuelo plácido a uno con turbulencias). Otra ventaja es que, al principio, los pacientes no tienen ganas de huir pues no se sienten en una situación límite. Posteriormente, con el uso del equipo que se utiliza (cascos, guantes, rastreadores de posición electromagnéticos...), las personas suelen vivir la experiencia como si fuera real. (www.ub.es/personal/jgutierrez/jgutierrez.htm)

Turismo

Aunque la realidad virtual no puede sustituir la experiencia de viajar, si puede hacer que el “turista” realice una visita virtual con carácter previo a la contratación del viaje.

También hay hoteles que están realizando guías virtuales de sus instalaciones para que el posible cliente compruebe la calidad de las mismas. Se trata en realidad de sustituir las tradicionales fotografías de habitaciones y otras instalaciones por realidad virtual.

Quizá no se tarde mucho en poder hacer turismo por escenarios que solo se conocen por los libros o las películas. ¿Qué tal viajar a Jurassic Park, a Naboo o dar un paseo a bordo del Halcón Milenario?

Cine

El día 8 de marzo de 2016 abrió en Amsterdam el primer cine de realidad virtual del mundo. Se trata de conseguir lo que, en realidad, no se logró con las gafas 3D: la inmersión del espectador en la historia que cuenta la película.

Simulación para la formación

Ya se ha hablado antes de los simuladores de aviones. Esta tecnología es aplicable a otros tipos de formación. Por ejemplo, simuladores para aprender a conducir automóviles, camiones o maquinaria pesada. Con esta tecnología se consigue imitar el comportamiento de la realidad y las consecuencias que las decisiones del usuario tienen sobre ella, lo que permite que el estudiante adquiera los conocimientos experimentando una realidad que de otra forma sería imposible o impagable.

Simulación de multitudes

La simulación de multitudes consiste en la simulación del comportamiento de grandes cantidades de personas. Sin requerir la presencia de gente, se puede simular el comportamiento de éstas en situaciones que serían complejas como la evacuación de un edificio o los comportamientos en situaciones límite (inundaciones, terremotos, ataques terroristas...)

Visualización arquitectónica.

Cuando alguien compraba su casa no hace tantos años, lo hacía visitando el piso piloto o, en muchos casos, viendo sólo el plano. Posteriormente se empezó a innovar en el sector y aparecieron las imágenes 3D.

Con la realidad virtual, se puede pasear por un edificio que todavía no está construido y comprobar cómo quedará nuestra próxima casa. Incluso se podría modificar la decoración o la disposición de algún elemento.

Parques temáticos

Aunque de momento sólo es un proyecto, “The Void” es el primer paso para introducir la realidad virtual en los parques temáticos.

La tecnología combinará elementos reales (equipamientos, escenarios y lluvia o vientos generados de forma mecánica) con gafas VR y entornos digitales para conseguir el mayor grado de sensación de inmersión

Otra aplicación que demuestra la actualidad de la tecnología virtual es la noticia aparecida en los telediarios recientemente sobre el zoológico de Barcelona. Su alcaldesa, Ada Colau, ha convocado una consulta a diferentes instituciones para decidir qué hacer con el zoológico de Barcelona. Pues bien, una de las sugerencias es realizar un parque zoológico virtual para evitar el sufrimiento de los animales y que el público en general pueda disfrutar de ellos.

Marketing experiencial

Mediante el marketing experiencial se busca crear sentimientos y emociones en los consumidores. Para ello, están empezando a utilizar realidad virtual.

Veremos tres ejemplos:

El primero es Mini Cooper: La campaña, llamada Mini 360º se basó por completo en la realidad virtual. El usuario debía loguearse en el sitio Web y recibía un “wieber” o visualizador para el móvil, con el cual podía vivir una película en primera persona y probar el coche sin necesidad de abandonar el sofá. Los visualizadores se agotaron rápidamente y la campaña fue todo un éxito, habiendo conseguido varios premios.

En segundo lugar, la campaña del fabricante de juguetes Lego. Mediante Lego Augmented Reality Kiosk, al pasar la caja del juguete por dicho dispositivo se forma el producto final.

La campaña fue un éxito total Actualmente se han instalados dichos dispositivos alrededor del mundo y un dato a tener en cuenta es que la cantidad de productos Lego devueltos ha bajado considerablemente, como así también el número de clientes disconformes con la compra, ya que mediante esta innovadora creación los clientes pueden experimentar de antemano el producto que están a punto de comprar.

Para finalizar este apartado, la campaña de NIKE. La empresa norteamericana lanzó la campaña “Nike True City” la cual, mediante una aplicación móvil y unas gafas de realidad virtual, las cuales venían acompañadas de las zapatillas, permitían a los usuarios transportarse a muchísimos sitios alrededor del mundo sin necesidad de poner un pie en un aeropuerto. Esta campaña fue también un éxito.

Reconstrucciones

Se está hablando de reconstrucciones “forenses”. Puede utilizarse en criminología para intentar reconstruir cómo se ha producido un delito.

En arqueología se podrán reconstruir acontecimientos, hechos o épocas pasadas mediante la realidad virtual. Se trataría de un documental inmersivo que nos permitiría trasladarnos a épocas remotas.

Debate ético

Hasta ahora no se ha hablado de los inconvenientes que puede tener la realidad virtual. En los diferentes productos que se están desarrollando, sólo los videojuegos advierten, en algunos casos, de posibles efectos nocivos.

Se puede poner un ejemplo: se utilizó la aplicación de realidad virtual en niños para entrenar sus habilidades en cruzar una calle y resultó ser bastante exitoso. Sin embargo, algunos estudiantes con trastornos del espectro autista después de dicho entrenamiento fueron incapaces de distinguir realidad virtual de la real.

Otro ejemplo sobre mala utilización de la tecnología virtual sería el uso de videojuegos para entrenamiento de terroristas ya que estas prácticas virtuales dotan a simples civiles de técnicas de disparo, visión y planificación estratégica para cometer delitos, lo que es un grave peligro, sobre todo en las circunstancias en que se encuentra la sociedad en los últimos tiempos.

Esto lleva al debate sobre el uso de la tecnología de realidad virtual desde un punto de vista ético, como, de hecho, ha ocurrido siempre ante la implantación de cualquier tipo de tecnología novedosa (por ejemplo, cuando comenzaron a utilizarse las maquinarias de vapor en las industrias).

Tres son, básicamente las visiones sobre la tecnología:

- Carroll W. Pursell dice que la tecnología es un medio y no un fin.
- Melvin Krasberg dice que la tecnología no es buena ni mala, pero tampoco es neutral
- Jacques Ellul dice que no importa cómo se utilice, tiene de por sí consecuencias negativas o positivas.

De estas premisas, aunque parecen contradictorias entre ellas, se puede extraer una conclusión: pese a todas las innegables ventajas de la realidad virtual, también hay que vigilar por si se produce alguna consecuencia negativa de su utilización.



Motores gráficos

Un motor gráfico es una aplicación que surgió con el objetivo de facilitar el desarrollo de software enfocado a videojuegos, pero cada vez más se utilizan para todo tipo de proyectos que requieran algún tipo de programación gráfica, ya que facilitan el trabajo debido a su potente diseñador de escenas tanto 2D como 3D y su capacidad de trabajar mediante interfaz.

Las principales funcionalidades que podemos destacar que aporten los motores gráficos son:

- Motor de renderizado para gráficos.
- Motor de físicas.
- Detección de colisiones.
- Gestión de scripts.
- Animaciones.
- Inteligencia Artificial.
- Control gráfico

A día de hoy existen infinidad de motores, tanto “*open-source*” como comerciales o de pago, desarrollados por empresas para luego ser vendidos a otros estudios de videojuegos para que desarrollen los suyos.

Unreal Engine

Unreal Engine 4 es un motor desarrollado por Epic Games, creadores de sagas como Unreal Tournament y Gears of War, entre otras. Se trata de un motor gráfico para PC y consolas implementado inicialmente en el shooter¹ en primera persona Unreal en 1998, es un motor con multitud de posibilidades que van desde sencillos juegos 2D para teléfonos hasta enormes superproducciones de PC y consola.

Es uno de los motores con más renombre de la actualidad y más utilizado por las *third parties* en la industria del videojuego, además de estar preparado para la llegada de DirectX 12.

El motor Unreal Engine 4 no sólo se ha utilizado en el desarrollo de videojuegos, sino que tiene otras aplicaciones como el desarrollo de simuladores o de previsualizaciones arquitectónicas, la creación de películas de animación, entre otras.

Los videojuegos más importantes desarrollados en este motor son: Unreal Tournament, Batman: Arkham Asylum, BioShock, Deus Ex, Gears of War, Star Wars Republic Commando o Mass Effect. Además de nuevos títulos que apostarán en un futuro por este motor como Kingdom Hearts III, Tekken 7, Fable Legends o, por supuesto, la nueva entrega de la saga Unreal Tournament.

¹ Juego de acción en primera persona que permiten controlar un personaje que dispone de un arma que puede ser disparada a voluntad. Juego de disparos.



Ilustración 4: Captura del videojuego Kingdom Hearts III (Cortesía de Disney Interactive)

En cuanto al precio, Unreal Engine 4 está disponible de forma gratuita para todos los desarrolladores que lo quieran utilizar, al igual que las próximas actualizaciones que se lancen. Tan solo se cobra por su uso al usuario una cierta cantidad de los beneficios del software desarrollado si estos superan una cierta cantidad.

CryEngine

CryEngine es el motor gráfico desarrollado por la empresa alemana Crytek, originalmente como un motor de muestra para Nvidia en 2002, se implementó por primera vez en un videojuego en 2006 con Far Cry, desarrollado también por la misma empresa, creando un referente gráfico durante muchos años.

CryEngine ofrece un paquete de herramientas multiplataforma, compatibles con las consolas de octava generación (PlayStation 4, Wii U y Xbox One), ordenadores Windows y Linux, y dispositivos móviles con sistema iOS y Android.

Es un motor gráfico que presume de una precisa iluminación, físicas realistas y una facilidad para diseñar complejas inteligencias artificiales, además de las ventajas de su diseño multiplataforma.

Son muchos los juegos comerciales que han utilizado este motor gráfico como Crysis, Evolve, Ryse: Son of Rome, The Cursed Forest entre una gran lista de títulos.



Ilustración 5: Captura del videojuego Ryse: Son of Rome (Cortesía de Crytek)

Crytek ofrecía este motor bajo suscripción mensual de 9,90€ al mes, pero con el lanzamiento de la última versión del motor: CryEngine V, la empresa ha cambiado el modelo de negocio por "Pay what you want" o "Paga lo que quieras" en castellano. Los usuarios que decidan hacer una donación podrán destinar hasta un 70% al programa Indie Development por el que Crytek da soporte a prometedores proyectos *indie*.

Con la llegada de CryEngine V también llega el CryEngine Marketplace, un mercado virtual que permite a los desarrolladores acceder a la librería de assets de la propia Crytek así como a cientos de materiales, sonidos y objetos en 3D creados en CryEngine por la propia comunidad y vendedores de confianza.

Unity 3D

Unity 3D es un motor desarrollado por Unity Technologies compatible actualmente con más de veinte plataformas. Éste ofrece una plataforma de desarrollo tanto en 2D como en 3D, y con una amplia compatibilidad en multitud de dispositivos, como Windows, Mac, Linux, Android, iOS, WebGL, WebPlayer, PlayStation 3, PlayStation 4, Wii U, Xbox One, Xbox 360.

El éxito de Unity ha llegado en parte debido al enfoque en las necesidades de los desarrolladores independientes que no pueden crear ni su propio motor del juego ni las herramientas necesarias o adquirir licencias. El enfoque de la compañía es "democratizar el desarrollo de juegos", y hacer el desarrollo de contenidos interactivos en 2D y 3D lo más accesible posible a tantas personas en todo el mundo como sea posible.

Además, se encuentran disponibles una serie de tutoriales (en inglés), aparte de los miles de tutoriales de gente anónima que circulan por internet. La comunidad tan

inmensa que tiene, hace a Unity 3D una opción muy interesante a la hora de decantarse por el desarrollo, no sólo de videojuegos, sino de cualquier tipo de aplicación, puesto que permite ponerse en contacto con desarrolladores más expertos puede ser una buena solución a un problema que no sepamos resolver.

La versión 5 de Unity 3D se ha puesto al día en lo visual, mejorando el desarrollo de interfaces, añadiendo iluminación global en tiempo real, sombreado basado en físicas o la versión 3.3 de la tecnología PhysX de Nvidia.

Unity es la plataforma más utilizada para el desarrollo de juegos indie que ha brindado grandes juegos como Monument Valley, Ori and the Blind Forest, Grow Home, Cities: Skylines o Pillars of Eternity

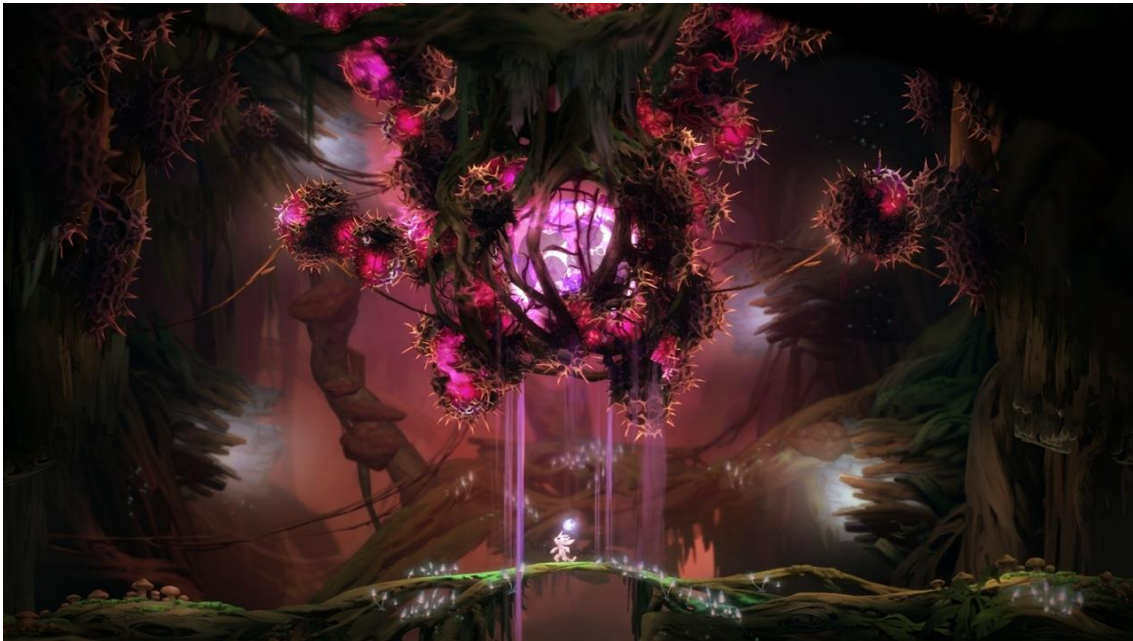


Ilustración 6: Captura del videojuego Ori and the Blind Forest (Cortesía de Game Informer))

Existen actualmente cuatro versiones de Unity distintas. Una versión gratuita llamada Personal, indicada para principiantes y aficionados o realización de trabajos académicos o pequeños proyectos no comercializables o que no perciban más de \$100 mil en fondos o en ingresos brutos anuales viéndose obligados a optar por una licencia de pago en caso de superarse. Esta será la licencia que se utilizará en el proyecto. El resto de versiones de Unity son versiones de pago.

Tanto Unity Personal como las versiones de pago incluyen el entorno de desarrollo, tutoriales, proyectos y contenidos de ejemplo, soporte a través de foros, wiki, API y acceso a todas las actualizaciones.

Bases de datos

Se le llama base de datos a los bancos de información que contienen datos relativos a diversas temáticas y categorizados de distinta manera, pero que comparten entre sí algún tipo de vínculo o relación que busca ordenarlos y clasificarlos en conjunto.

En nuestra aplicación, la base de datos se encargará de gestionar la información de nuestro visor: los modelos que la componen, su ruta y su información. A continuación, se profundizará en una selección de gestores de bases de datos gratuitos que podemos encontrar, comentando sus características y algunas ventajas e inconvenientes.

SQLite

SQLite es una base de datos relacionan compatible con ACID escrita en C, y famosa por operar con un solo fichero por base de datos.

En SQLite la base de datos consta de un solo archivo en la unidad de almacenamiento, lo cual lo hace extremadamente portable. Aunque parezca que utiliza una implementación simple, SQLite utiliza casi todas las características de SQL, siendo muy buena para desarrollar y testear, ya que es muy escalable con la simplicidad de trabajar únicamente con un archivo. Además de ser software libre.

Por otra parte, SQLite no permite la administración de usuarios, por lo que no existirá tampoco la posibilidad de dar privilegios ni soportar concurrencia de usuarios en la escritura. Además, por su diseño, no es posible mejorar el rendimiento de SQLite.

MySQL

MySQL es un sistema de gestión de bases de datos relacional de los más utilizados ya que podemos encontrar de forma completamente gratuita dentro de la distribución de Apache XAMPP aunque está desarrollado bajo licencia dual GPL/Licencia Comercial, por lo que además de la versión distribuida bajo la Licencia pública general de GNU, existen varias versiones Enterprise que incluyen productos o servicios adicionales.

Este gestor multiplataforma, que funciona en la gran mayoría de sistemas operativos, atrajo a desarrolladores web por su simplicidad y, además, cuenta con la ventaja de que todos los elementos de los que pueda carecer, poco a poco son incorporados por desarrolladores tanto internos como de software libre.

MongoDB

MongoDB es un sistema de base de datos NoSQL. En vez de usar tablas y filas como las bases de datos relacionales, se basa en una arquitectura de colecciones y documentos que emularán su comportamiento.

Es un software libre que realiza sus consultas mediante lenguaje JavaScript, que son enviadas y ejecutadas de forma directa en la base de datos. Además, el usuario dispone de una clave para proteger la base de datos de accesos ajenos. Otro punto a destacar es su fácil replicación y su configuración automática.

Entre sus inconvenientes podemos destacar que bloquea la base de datos al realizar una acción de escritura, lo cual reduce la concurrencia. Si no se replica, es una base de datos muy poco fiable ya que un fallo durante la actualización de la tabla implica la pérdida de toda la información y una muy costosa, temporalmente hablando, reparación. Además, presenta problemas de rendimiento cuando el tamaño de los datos supera los 100GB y requiere mucha memoria RAM para la indexación.

MongoDB no garantiza ACID. Esto es el acrónimo de *“Atomicity, Consistency, Isolation and Durability”*, es decir, *“Atomicidad, Consistencia, Aislamiento y Durabilidad”*.

Para que un sistema gestor de bases de datos sea considerado ACID, debe cumplir las cuatro características:

- **Atomicidad:** Propiedad que asegura que las transacciones son completas, es decir, si una operación consiste en una serie de pasos, todos estos ocurren o ninguno de ellos.
- **Consistencia:** Es la propiedad que asegura que no se van a romper las reglas ni directrices de integridad de la base de datos, es decir, que todo lo que se empieza se acaba. Dicho de otro modo, cualquier transacción llevará a la base de datos de un estado válido a otro también válido.
- **Aislamiento:** Propiedad que garantiza que una operación no puede afectar a otras.
- **Durabilidad:** Es la propiedad que asegura que las operaciones realizadas persistirán y los datos sobrevivirán de alguna manera aunque falle el sistema.



Desarrollo e implementación del proyecto

Descripción

El trabajo consiste en el desarrollo de un visor de modelos tridimensionales operativo sobre un navegador web. El visualizador mostrará modelos 3D contenidos en una base de datos junto con otra información de referencia. Para la realización de este proyecto, se utilizará las aplicaciones Unity 3D para el desarrollo de la aplicación web, Visual Studio para la escritura de scripts, SQLite como gestor de bases de datos y programación HTML y PHP para programar la web. Además, se utilizará el servidor XAMPP para realizar las pruebas de la aplicación web en local.

La aplicación se compone de una página web a la que podremos acceder desde cualquier navegador, esta página web contendrá dos enlaces, uno redirigirá al usuario a un formulario web PHP donde podrán subirse nuevos modelos 3D en un archivo comprimido ZIP y se añadirá su registro en la base de datos, el otro enlace redirigirá a la aplicación.

La aplicación constará de un menú principal donde se podrá seleccionar los modelos que se desean cargar en la aplicación. Estos botones se generarán en tiempo de ejecución, es decir, la aplicación conectará con la base de datos, leerá los modelos que la componen y generará un botón por cada modelo encontrado con su respectivo nombre. En caso de no encontrar ningún modelo, la aplicación no contendría ningún botón.

Cuando se seleccione un botón, la aplicación ocultará el menú principal y cargará el modelo en el visor, a partir de entonces el usuario podrá visualizar el modelo, rotarlo, realizar zoom y utilizando los atajos de teclado (Véase Anexo I) o los botones que se encuentran en la parte inferior, se podrá acceder a herramientas o ampliación de información del modelo.

Pulsando el botón herramientas se desplegarán las herramientas, para este proyecto solo se va a desarrollar la herramienta de medir, que permitirá medir el modelo seleccionando dos puntos de este, pero en un futuro se podrán implementar nuevas herramientas como hacer secciones, cambio de luz, etc.

Con el resto de botones, el usuario podrá acceder a la información en formato texto del modelo si tiene, a una galería de imágenes del modelo o que aporten información adicional relacionada con éste y a un reproductor de video que cargará un archivo de video relacionado con el modelo o que, al igual que las imágenes, aporte información representativa. Toda esta información estará referenciada en la base de datos y la aplicación deberá acceder a ella para poder cargar la información en función del modelo seleccionado en el menú principal.

Además, en cualquier momento se podrá volver al menú principal y cargar un nuevo modelo. Para salir de la aplicación bastará con salir del navegador web.

Análisis de las herramientas

Unity3D

Unity, como ya se ha tratado anteriormente, es un motor que ofrece muchas y diversas utilidades que facilitan su uso tanto a programadores como diseñadores o artistas. Estas herramientas van desde editores de terreno o clases ya preparadas hasta *shaders*. A continuación, se mostrarán las herramientas básicas que se utilizarán en este proyecto.

¿Por qué usar Unity?

En el capítulo 3 se trataron los principales motores de videojuego, de entre ellos se ha se elegido el motor Unity 3D para la realización de este proyecto destacando las siguientes razones.

- Unity ofrece la posibilidad de programar con C# y JavaScript, lenguajes más conocidos (frente a LUA de CryEngine) y con gestión dinámica de memoria (frente a C++ de Unreal)
- Aunque se trabaje con la versión gratuita, esta es ampliable mediante plugins, lo que lo convierte en un potente motor suficiente para la mayoría de proyectos.
- Soporte multiplataforma, no solo videoconsolas o PC, sino también Mac, WebGL, iOS, Android, entre otros.
- Unity se está convirtiendo en un estándar del mercado actual, y dado que el objetivo de este proyecto es que siga creciendo en un futuro, convierte a este motor en uno de los principales candidatos.

Interfaz

Unity cuenta con una interfaz muy intuitiva que consta de diversas ventanas integradas. Estas ventanas puedes desplazarse por el interfaz de Unity para amoldarse al gusto del usuario. Las ventanas principales que proporciona Unity por defecto son: la ventana de proyecto, la ventana de la jerarquía, el inspector, la ventana de escena y la ventana de juego.

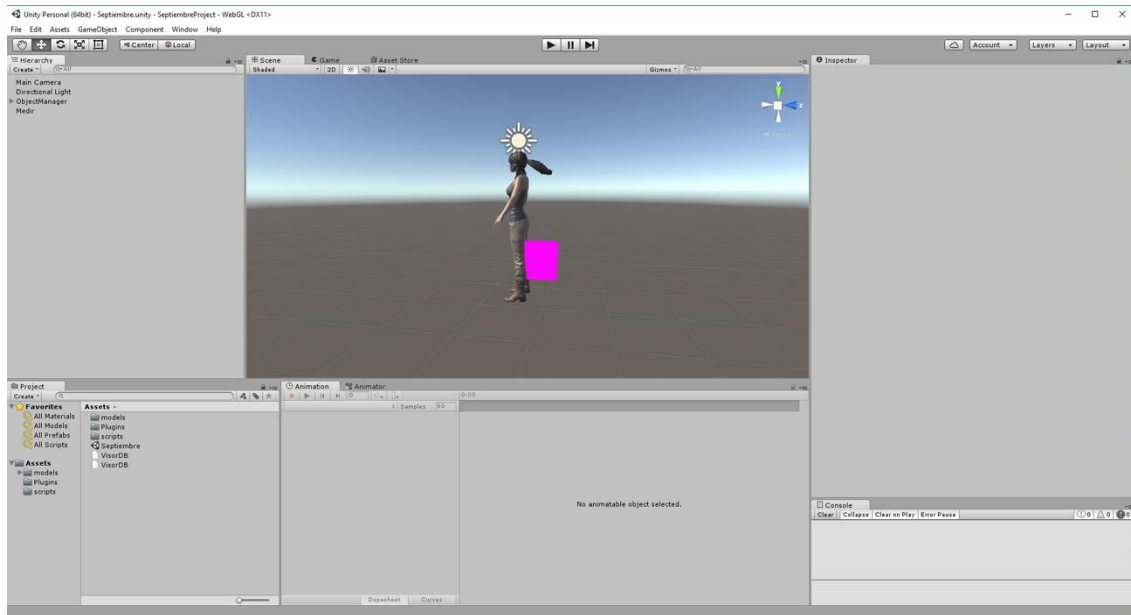


Ilustración 7: Ventana principal del editor de Unity

La **ventana de proyecto** es un explorador de archivos, como se puede observar en la ilustración 7, en el que se puede agregar, eliminar, organizar o modificar elementos (también conocidos como assets) que puede tener nuestro proyecto. Todos los archivos mostrados se corresponden con los archivos que encontramos dentro de la carpeta Assets en el directorio raíz del proyecto y se sincroniza en tiempo real con la carpeta del sistema, eso significa que, si se realiza algún cambio en la carpeta del mismo, al volver a Unity este automáticamente actualizará el proyecto. También se puede arrastrar un objeto dentro de la ventana de proyecto para añadirlo.

En la ventana de proyecto también se encuentra un buscador que permite filtrar o buscar componentes. Además, el usuario puede añadir elementos a la carpeta assets para usarlos en un futuro, ya que, pese a aumentar el tamaño del proyecto, Unity, a la hora de compilar, solo tendrá en cuenta los assets que se utilicen en alguna escena del proyecto, dejando fuera del ejecutable aquellos que no se les de ningún uso.

La **ventana de escena** es la que contiene una representación de aquellos assets instanciados en el proyecto, pudiendo configurar en vista 2D o 3D. En esta ventana el usuario puede hacerse una idea de la composición de todos los elementos, con la posibilidad de editarlos, mover, rotar o escalar. Además, se le pueden añadir funcionalidades como añadir objetos, cambiar propiedades o la posibilidad de dibujar elementos para una mejor organización de la escena o para trazar una trayectoria. Esto último hace referencia al elemento Gizmos.

La **ventana de juego** aparece cuando ejecutamos la aplicación dentro del editor y muestra el resultado que se obtendrá al iniciar la escena actual. Esta acción se realiza dentro del editor, y permite ajustar el juego a distintas resoluciones para probar que se vea bien en los diferentes tipos de resolución de pantalla. Además, permite pausar la ejecución con el botón pause o ejecutar *frame a frame* para una inspección más exhaustiva.

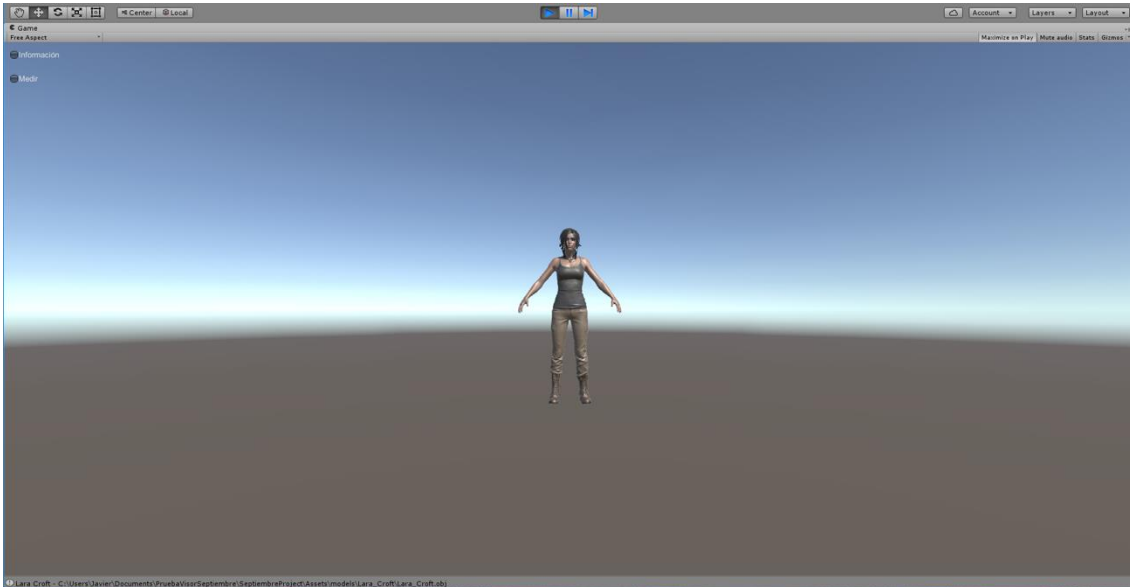


Ilustración 8: Ventana de juego

El **inspector** es probablemente uno de los elementos que hacen de Unity un motor tan intuitivo y manejable. Permite ver las propiedades de cada elemento y configurar la mayoría de ellas desde el propio inspector. La mayor ventaja que ofrece el inspector es la posibilidad de ver las variables de los scripts y permitir cambiarlas en tiempo real y el poder arrastrar elementos a ellas desde el editor.

La **ventana de jerarquía** es la ventana donde se encuentran todos los GameObjects que forman la escena ordenados jerárquicamente, es decir, muestra el vínculo entre un GameObject principal o uno que sea hijo de este.

Además de todas estas ventanas, también posee algunas importantes como la de la consola, que muestra todos los mensajes del sistema: errores, mensajes, etc. y redirige a la línea de código correspondiente, la ventana de animación, que permite crear animaciones mediante la interpolación de valor o el animator, donde se puede crear máquinas de estado (*grafos*) y asignar animaciones a cada una de las transiciones de los mismos. También es posible encontrar alguna ventana personalizada de plugins que se hayan instalado en el editor.

Assets

Es uno de los componentes esenciales de todos los proyectos de Unity. Un asset abarca multitud de elementos: desde modelos 3D que sirven para formar mallas, texturas en formato de imagen hasta archivos de sonido.

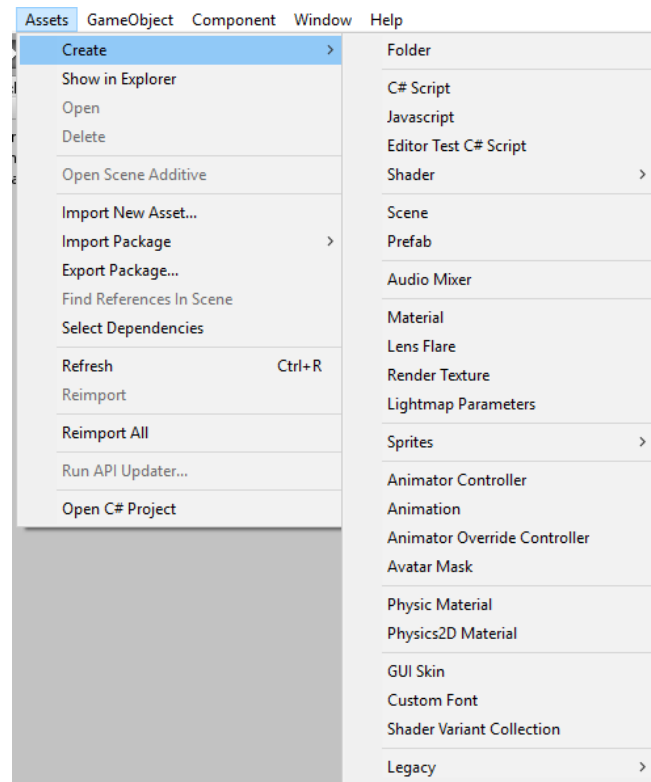


Ilustración 9: Pestaña desplegada de Assets

Todos estos se almacenan en un mismo directorio y se podría decir que son los componentes que van a ser usados para la ejecución de la aplicación. El directorio Assets lo podremos encontrar en la ventana del proyecto.

Scenes

El concepto de Scene de Unity se trata de una composición de assets, scripts y eventos que conformarán los niveles o escenas de la aplicación o videojuego.

Cada escena se maneja de forma individual, lo que quiere decir que no se puede interactuar con más de una escena al mismo tiempo, aunque algunos elementos sí que pueden permanecer entre ellas, que son implementadas en forma de jerarquía.

GameObjects y componentes

Un GameObject es como se denominan los elementos u objetos del juego que componen la escena. Algunos assets pueden hacer dicha función además de los que permite generar Unity. Cada uno de estos objetos están constituidos por una serie de componentes que dotan de comportamiento al objeto. Un GameObject debe contener al menos un componente, el componente Transform, que será explicado a continuación. Para saber si el elemento está activado tiene una variable denominada *enabled* que modula si está activo o no, a la vez que cada componente individualmente dispone de un booleano para indicar si está activo o no, desactivando o activando dicha funcionalidad del objeto.

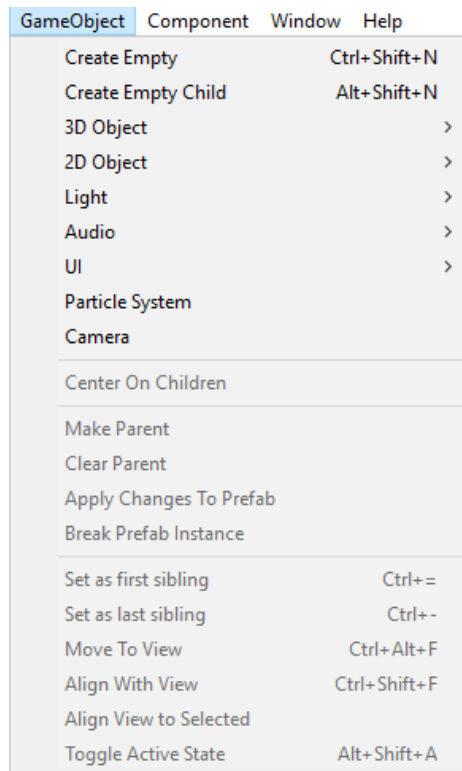


Ilustración 10: Pestaña desplegada de GameObject

Concretamente, un GameObject es un conjunto de componentes instanciado en la escena y puede contener otros GameObjects distintos como hijos, ya que se ordenan de forma jerárquica. Y un componente es básicamente un script que tiene que heredar de la clase *monobehaviour*. Unity ya tiene diversos componentes implementados, pero lo más común es que la inmensa mayoría de los scripts que se utilicen sean propios. Algunos de los más importantes son:

- **Transform:** Componente que contiene la posición, rotación y escala del objeto.

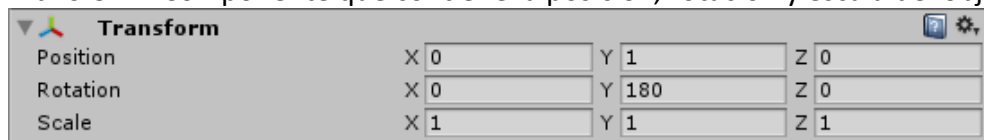


Ilustración 11: Propiedades del componente Transform

- **Collider:** Componente que crea un área de inclusión (caja, esfera, capsula, ...) Esto permite que el objeto pueda colisionar con otros y que se disparen eventos cuando esto suceda.

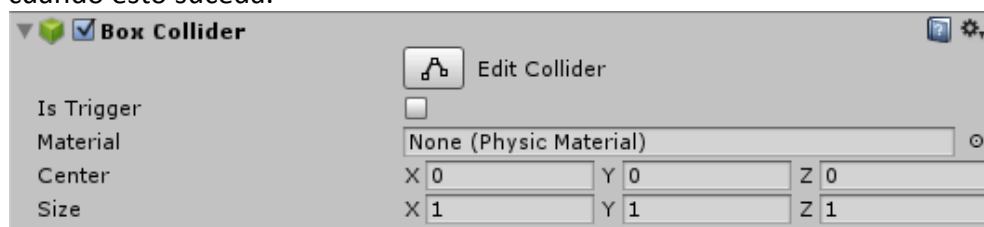


Ilustración 12: Propiedades del componente BoxCollider

- **Rigidbody:** Componente que permite al GameObject interactuar con las físicas, ya que este componente hará que la gravedad le afecte y le permite añadir fuerzas con un tipo, sentido y magnitud.

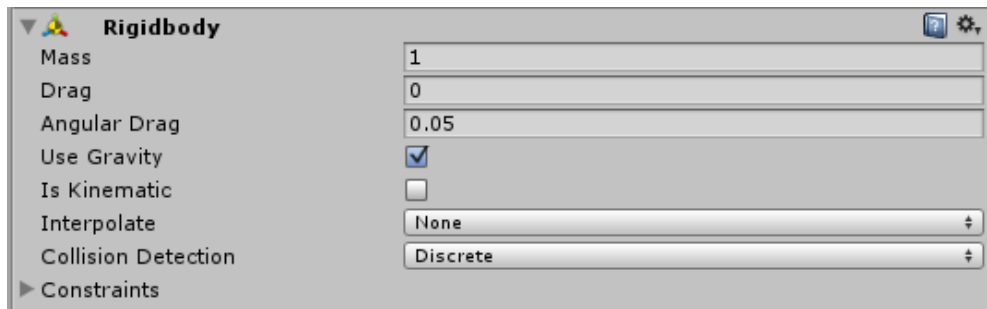


Ilustración 13: Propiedades del componente Rigidbody

- **AudioSource:** Componente que permite al GameObject emitir sonidos tanto en 3D (con atenuación del sonido en función de la distancia) o en 2D.

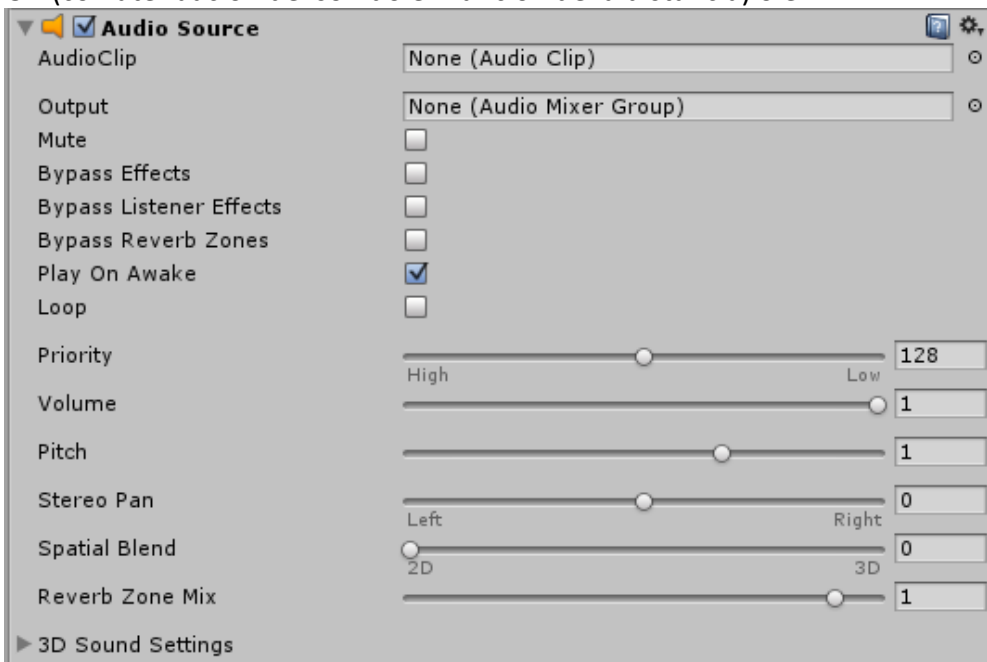


Ilustración 14: Propiedades del componente Audio Source

- **AudioListener:** Componente complementario del AudioSource, ya que recibirá los sonidos que estos emitan.



Ilustración 15: Propiedades del componente Audio Listener

Prefabs

Un prefab, contracción de prefabricado, es una copia de un GameObject convertida en un recurso utilizable. Se muestran en el directorio del proyecto y está serializado en el disco como un archivo. Junto con el prefab también se guarda la jerarquía que depende de este GameObject, de este modo cuando se instancia también crearemos a todos sus hijos. Gracias a todo esto se pueden crear objetos complejos de una manera más rápida y sencilla e incluso cargarlos en tiempo de ejecución, además de poder modificar los componentes y variables de un prefab cambiando automáticamente en todas sus instancias.

Scripts

Los scripts o secuencias de comandos son los componentes encargados de dotar de funcionalidad al proyecto ya que se encargan de modificar los valores y propiedades de cualquier objeto, convirtiéndose en un componente imprescindible de cualquier aplicación en Unity.

Para la escritura de estos scripts, se puede utilizar el editor que trae Unity por defecto, *Monodevelop*, aunque en este proyecto y desde la versión 5 es recomendable utilizar Visual Studio. Al guardar o modificar las secuencias en el editor de scripts, estos se actualizan inmediatamente en el editor de Unity.

Para realizar scripting es importante conocer la clase *MonoBehaviour*, que traen los scripts por defecto. Es la clase principal que cualquier script que desee formar parte de un *GameObject* debe de heredar. Esta clase aporta algunos métodos importantes en los que cabe destacar:

- **Start:** Se llama al inicio de la creación del *GameObject* que lo contine. Este método se utiliza principalmente para asignar valores por defecto al *Script*.
- **Awake:** Se ejecuta justo antes que el método *Start*. Esta inicialización en dos tramos de tiempo diferente puede resultar útil para algunos scripts en los que se necesitan que otros scripts ya tengan algunos valores calculados.
- **Update:** Este método se ejecuta *frame a frame*. Es útil para realizar control del input o de movimientos de personajes en videojuegos.

Además de otros métodos también destacables, sobretodo en el ámbito de los videojuegos, como *LateUpdate* o *FixedUpdate*.

Visual Studio

Microsoft Visual Studio es un entorno de desarrollo integrado para sistemas operativos Windows, capaz de soportar bastantes tipos de lenguajes de programación. Además, desde la versión 5 de Unity se incluye este entorno con el motor, al igual que el editor propio de Unity *MonoDevelop*, pero se utilizará Visual Studio para el desarrollo de este proyecto ya que es un software mucho más potente, completo, más utilizado y generalizado que *MonoDevelop*.

A la hora de programar los scripts de este proyecto, Unity interactúa con este programa para la coordinación del código de los scripts con el proyecto sobre el que se está trabajando.

En este proyecto, como ya se ha comentado, se utilizará Visual Studio para escribir el código de los scripts que se crearán para esta aplicación, aprovechando su vinculación con Unity.



SQLite

Se trata de un sistema de Gestión de bases de datos relacional contenido en una biblioteca escrita en C y no necesita un motor de Base de datos a diferencia de otros como MySQL, PostgreSQL, Oracle, etc.) que lo hacen totalmente independiente y que pasa a integrarse en la aplicación. Además, todo el conjunto de la base de datos (definiciones, tablas, índices...) son guardados como un solo fichero estándar en la máquina local. Además, permite gestionar hasta 2TB de información, lo cual lo hacen una opción muy interesante.

En este proyecto utilizaremos SQLite para crear y gestionar nuestra base de datos.

¿Por qué usar SQLite?

- **Tamaño:** tiene una pequeña memoria y una única biblioteca para acceder a bases de datos.
- **Rendimiento:** realiza operaciones eficientes y es más rápido que MySQL y PostgreSQL.
- **Portabilidad:** funciona en casi todas las plataformas y sus bases de datos pueden ser transportadas sin ninguna configuración o administración.
- **Estabilidad:** SQLite es compatible con ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad)
- **Precio:** SQLite es una aplicación libre y gratuita

PHP

PHP (acrónimo recursivo de PHP: Hypertext Preprocessor) es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML.

En lugar de usar muchos comandos para mostrar HTML (como en C o en Perl), las páginas de PHP contienen HTML con código incrustado que hace "algo". El código de PHP está encerrado entre las etiquetas especiales de comienzo y final `<?php` y `?>` que permiten entrar y salir del "modo PHP".

Lo que distingue a PHP de otros lenguajes como JavaScript desde el lado del cliente, es que el código es ejecutado en el servidor, generando HTML y enviándolo al cliente. El cliente recibirá el resultado de ejecutar el script, que no sabrá el código subyacente que era. El hecho de que el servidor web pueda ser configurado para que procese todos los ficheros HTML con PHP, hace que no haya manera de que los usuarios puedan saber el código que hay tras él.

Lo mejor de utilizar PHP es su extrema simplicidad para el principiante, pero a su vez ofrece muchas características avanzadas para los programadores profesionales. Aunque el desarrollo de PHP está centrado en la programación de scripts del lado del servidor, se puede utilizar para muchos otros fines.

En este proyecto se utilizará PHP para desarrollar un script de subida de archivos, que los descomprima en el servidor y además añada su información a una base de datos.

HTML

HTML es el lenguaje que se emplea para el desarrollo de páginas de internet. Está compuesto por una serie de etiquetas que el navegador interpreta y da forma en la pantalla. HTML dispone de etiquetas para imágenes, hipervínculos que nos permiten dirigirnos a otras páginas, saltos de línea, listas, tablas, etc. Se podría decir que HTML sirve para crear páginas web, darles estructura y contenido.

En este proyecto se utilizará HTML para crear y diseñar la web principal que albergará la aplicación web desarrollada en Unity.

XAMPP


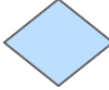



XAMPP es un servidor independiente basado en software libre de apache que incluye MySQL, PHP y otras herramientas para el desarrollo de aplicaciones web, es decir, podemos disponer de un servidor propio o simplemente puede usarse para hacer pruebas de nuestras páginas web. Es un paquete gratuito y fácil de instalar, tan sencillo como descargar el archivo y extraerlo. Una de sus ventajas es ser multiplataforma, ya que existen versiones para Windows, MacOS y Linux.

En este proyecto se utilizará XAMPP para la realización de pruebas de la aplicación web en un entorno web local.

Arquitectura software

Lógica de la aplicación

A continuación, se explica el comportamiento de la aplicación mediante un diagrama de flujo (ver Ilustración 16), mostrando las acciones que se pueden realizar en la misma. La leyenda de las figuras que se utilizan en el diagrama son las siguientes:

	Este icono representa las distintas acciones a realizar en la pantalla.
	Este icono representa una decisión en función de la cual el sistema determinará el camino a seguir.
	Este icono representa una pantalla o vista dentro de la aplicación.
	Este icono representa una pantalla o vista dentro de la aplicación que varía en función de su contenido.
	Este icono representa el inicio o final de la ejecución del programa.

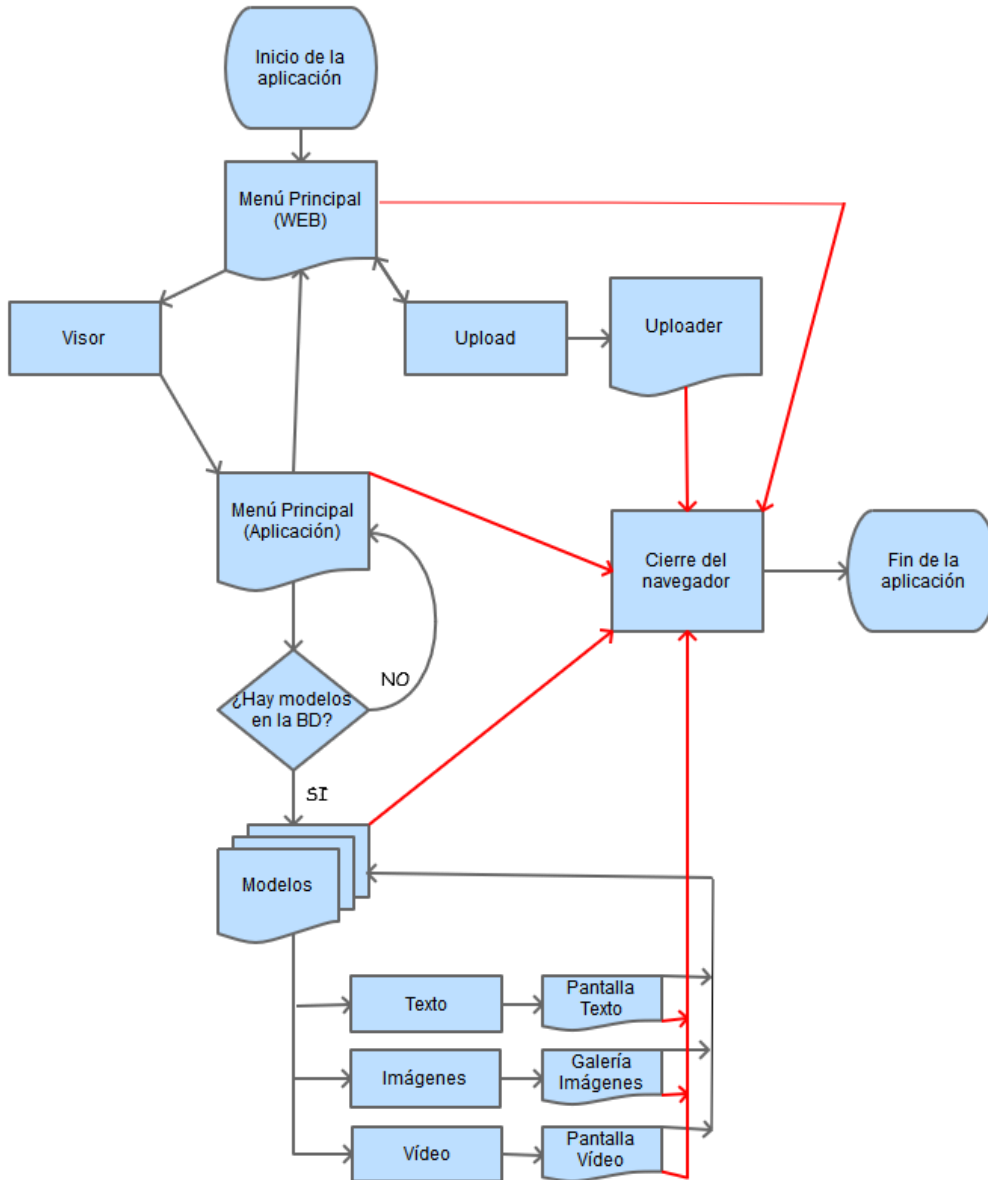


Ilustración 16: Diagrama de flujo de la aplicación

Se puede notar que la aplicación es bastante sencilla y básica, y casi todos los elementos giran en torno a la ventana de modelos.

Capa de persistencia

La base de datos que se utiliza en el servidor será muy básica con el fin de no crear una aplicación demasiado compleja, aunque podrá ampliarse en un futuro. La base de datos guardará la ruta de los modelos 3D, o de sus piezas, de nuestro visor junto a su nombre y la información importante que esta posea. Las tablas se muestran en la siguiente ilustración.

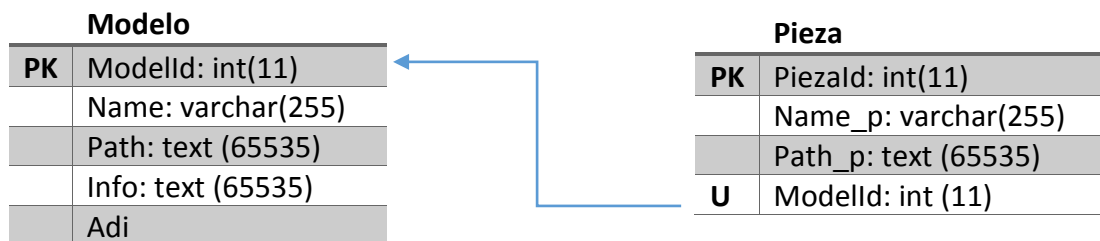


Ilustración 17: Capa de persistencia

Implementación

Aplicación Unity

Para comenzar con el desarrollo de esta aplicación, se abrirá el editor de Unity 3D y se creará un proyecto nuevo. El editor de Unity se abrirá y nos encontraremos con una escena vacía, una luz direccional y una cámara principal.

Lo primero que se deberá hacer es diseñar las funcionalidades del visor, para ello se creará un GameObject vacío situado en el centro de la cámara el cual será denominado ObjectManager. Este será el lugar donde se abrirán los modelos 3D seleccionados de la base de datos. Para poder trabajar hasta que la base de datos sea vinculada a la aplicación, se cargará un modelo manualmente. Para ello, se creará una nueva carpeta llamada “models” en el directorio “Assets” donde se guardaran todos los componentes de un modelo (la malla², la textura³, etc). Para trabajar más cómodamente en un futuro, se deberá trabajar con Prefabs de los modelos, es decir, trabajar tan solo con un GameObject donde ya tiene sus propiedades y texturas asignadas. Una vez cargado un modelo, se le asignará como hijo a ObjectManager, arrastrándolo a la ventana Jerarquía sobre él.

Ahora se comenzará a desarrollar los scripts, por lo que dentro de la carpeta Assets se creará una carpeta Scripts para contenerlos todos ahí y mantener ordenado el workspace⁴ del proyecto.

La primera versión que se hizo del visor, la cámara rotaba sobre el objeto, pero en las pruebas se descubrió que, al variar el tamaño de los modelos, esta llegaba a descentrarse y realizaba movimientos extraños, por lo que esa opción se desechó y se pensó en una nueva alternativa.

En esta opción alternativa, que es la que tiene implementada el visor, el único movimiento que realiza la cámara es el zoom. Para esto, se creará un nuevo archivo C# que será nombrado “Zoom.cs”.

² Redes de triángulos interconectados que representan el esqueleto o la forma del modelo 3D

³ Archivo de imagen que se sitúa sobre la malla, dotándole de visibilidad y aspecto al modelo 3D

⁴ Espacio de trabajo

```

using UnityEngine;
using System.Collections;

public class Zoom : MonoBehaviour
{
    float minFov = 15f;
    float maxFov = 90f;
    float sensitivity = 10f;

    // Use this for initialization
    // void Start () {

    //}

    // Update is called once per frame
    void Update()
    {
        float fov = Camera.main.fieldOfView;
        fov -= Input.GetAxis("Mouse ScrollWheel") * sensitivity;
        fov = Mathf.Clamp(fov, minFov, maxFov);
        Camera.main.fieldOfView = fov;
    }
}

```

Ilustración 18: Código "Zoom.cs"

Con este script se consigue controlar el zoom de la cámara mediante la rueda del ratón. Este script se le asignará como componente a la cámara situada en la escena. Para ello, se seleccionará la cámara en la ventana de Jerarquía y una vez se desplieguen todos sus componentes en la ventana del Inspector se podrán realizar dos acciones: arrastrar el script desde la ventana del proyecto al inspector o seleccionar el botón "Add Component" del inspector y seleccionar el script en el menú desplegable que aparece. En caso de no aparecer, este menú contiene un buscador donde se puede escribir el nombre del script para seleccionarlo.

La rotación se le asignará al modelo con el script que denominaremos "Controles.cs", además de una función para reiniciar la posición del modelo. Como los modelos se añadirán en tiempo de ejecución, el Script será colocado como un componente del GameObject "ObjectManager", que será donde cargará el modelo como hijo de este. Además, utilizaremos este GameObject para añadir el resto de scripts de gestión como "DB Manager.cs" y "Menus.cs", que explicaremos más adelante.

```

using UnityEngine;
using System.Collections;

public class Controles : MonoBehaviour {
    private Vector3 initialPosition;
    private Quaternion initialRotation;
    public float horizontalSpeed = 2.0F;
    public float verticalSpeed = 2.0F;

    void Start()
    {
        initialPosition = transform.position;
        initialRotation = transform.rotation;
    }

    void Update()
    {
        if(Input.GetMouseButton(0))
        {
            float h = horizontalSpeed * Input.GetAxis("Mouse X");
            float v = verticalSpeed * Input.GetAxis("Mouse Y");
            transform.Rotate(v, h, 0);
        }

        if (Input.GetKey("r"))
        {
            transform.rotation = initialRotation;
            transform.position = initialPosition;
        }
    }
}

```

Ilustración 19: Código "Controles.cs"

Para que el visor pueda medir, se creará un nuevo `GameObject` vacío, denominado "Medir". Como se necesita poder marcar visualmente los puntos que se van a medir, se le añadirá el componente *Line Renderer*. El *Line Renderer* (o renderizador de línea) toma un fragmento de dos o más puntos en espacio 3D y dibuja una línea recta entre cada uno, por lo que facilita la representación gráfica que se busca a la hora de medir. Para controlar este componente, se creará un script llamado "Measoure.cs" que también hemos añadido al `GameObject` Medir.

En este código, en el método *OnGui()* se creará una *checkbox* que activará o desactivará el *Line Renderer*, a la vez que abrirá un cuadro donde se podrán ver las medidas entre los dos puntos seleccionados. En el resto de funciones, se inicializa el *Line Renderer*, se controla que haya únicamente dos puntos (ya que en un *Line Renderer* se pueden tener infinitud de puntos, lo cual impediría realizar cálculos de medición entre dos puntos) y se mide la distancia entre ellos, que será la que se muestre en el cuadro. Las distancias en Unity, por defecto, vienen en metros, por lo que habrá que preocuparse de que el modelo este escalado correctamente a la hora de importarlo a la aplicación.

Desde Unity 5, el motor dispone de un nuevo sistema de interfaz que deja obsoleto las llamadas *OnGui()* (aunque se haya utilizado antes), que es el denominado UI (Unity

Interface). Por ello, se van a aprovechar sus funcionalidades para crear las interfaces relacionadas con la información del modelo, es decir, que muestren textos, imágenes y videos si el modelo dispone de ellos.

El primero que se va a desarrollar es el video, el cual no estará disponible actualmente en la aplicación debido a que Unity todavía no permite trabajar con *MovieTexture* en entornos de red (Visor Web, WebGL, Android y Iphone), por lo que se dejará el código desarrollado para una futura inserción cuando el editor lo permita.

Lo primero que se hará es crear un Canvas. El Canvas es el área donde todos los elementos UI deben estar. El Canvas es un Game Object con un componente Canvas en él, y todos los elementos UI deben ser hijos de dicho Canvas. Para crear un Canvas hay que acceder al menú GameObject > UI > Canvas. Aparecerá en la ventana de Jerarquía y se renombrará por "Video". Se pulsará clic derecho a "Video" y se seleccionará UI > *RawImage* para crear este objeto como un hijo de "Video". Esta *RawImage* será el lugar donde se reproducirá el video, por ello, seleccionaremos la "Rect Tool" de la barra de herramientas (última herramienta de la barra comenzando de izquierda a derecha), ya que es la herramienta para mover, cambiar el tamaño y rotar los elementos de la UI utilizando la ventana de escena. Por comodidad, se pondrá esta ventana en modo 2D y se editará el objeto *RawImage* manualmente desde la escena o utilizando el componente *Rect Transform*, que sería el equivalente al componente *Transform* de los GameObjects, para que ocupe toda la pantalla y consiga que así el video se reproduzca a pantalla completa, aunque se podría decidir cualquier otro tamaño o posición.

Una vez se tiene el objeto *RawImage* colocado correctamente, se pasará a escribir el script que reproducirá el video. Se creará un nuevo script en C# llamado "Video.cs":

```

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

[RequireComponent(typeof(AudioSource))]

public class Video : MonoBehaviour {

    bool pausa;

    public MovieTexture movie;
    private AudioSource audio;

    void Start () {
        GetComponent<RawImage>().texture = movie as MovieTexture;
        audio = GetComponent<AudioSource>();
        audio.clip = movie.audioClip;
        movie.Play();
        audio.Play();
    }

    void Update () {
        if (Input.GetKeyDown(KeyCode.Space)&& movie.isPlaying)
        {
            movie.Pause();
            pausa = true;
        }
        else if (Input.GetKeyDown(KeyCode.Space) && !movie.isPlaying)
        {
            movie.Play();
            pausa = false;
        }
        if (!movie.isPlaying && !pausa)
            movie.Stop();
    }
}

```

Ilustración 20: Código "Video.cs"

El código carga el video como una textura para *RawImage* (de ahí que el tamaño del video dependiera del tamaño del objeto). Además, se debe de añadir un componente *AudioSource* que se inicie a la vez que el video para que se reproduzca el sonido de este, de otra forma, el video se reproduciría muteado, ya que Unity no trabaja los videos con sonido. Habrá que asegurarse de que la cámara tiene el componente *AudioListener* activado para poder escuchar el sonido del video. En la método *Update()* se añaden las opciones de reproducción, es decir, que con el botón Espacio se pueda pausar o reproducir el video, y que una vez acabe el video este se detenga, consiguiendo que si se volviera a entrar al video se pueda reproducir otra vez, si no este se quedaría pausado en el final infinitamente.

Se añadirá este Script como componente de la *RawImage* y en la ventana del inspector aparecerá que la variable *Movie* está en "None (Movie Texture)", por lo que se deberá arrastrar el fichero de video a ese cuadro o seleccionarlo pulsando el circulo que se encuentra a su derecha y que abrirá una ventana del navegador con todas las *Movie Texture* que encuentre en el proyecto (*ilustración 21*).

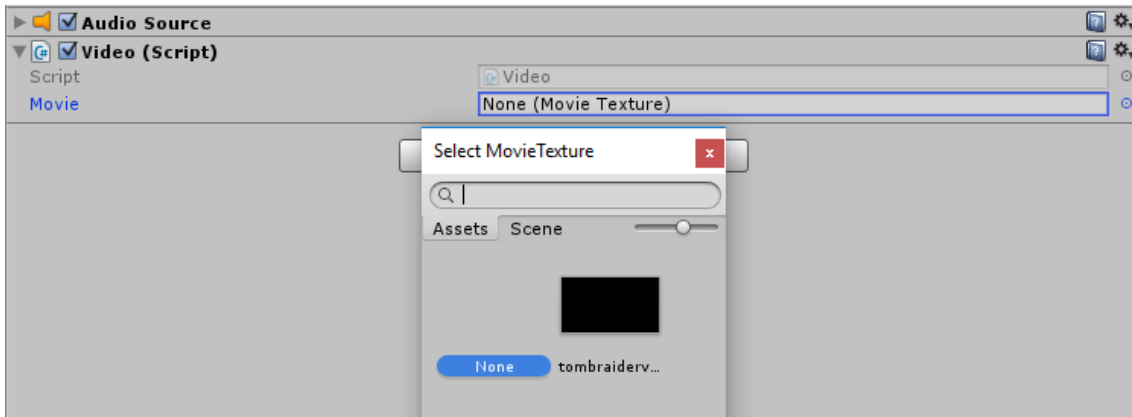


Ilustración 21: Ventana del inspector seleccionando una Movie Texture

Como la textura del video bloquea toda la imagen tanto en la ventana de escena como en la de juego, se deberá crear un script que controle la aparición y desaparición de esta textura, pero se tratará esto más adelante. Ahora se seleccionará el Canvas "Video" en la ventana Jerarquía y se desactivará quitando el tic del *checkbox* que aparece a la izquierda de su nombre en la ventana inspector.

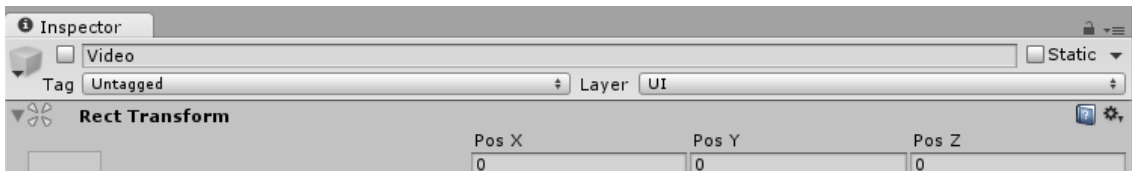


Ilustración 22: Canvas Video desactivado

El procedimiento para mostrar las imágenes, textos e incluso el menú principal son muy parecidos, por lo que se profundizará menos en la creación de la UI.

Para mostrar imágenes, lo que se hará es crear una pequeña galería donde se muestren todas las imágenes que se encuentren en una carpeta. Para ello, se creará un nuevo Canvas que se llamará "Imágenes" y se creará un *Panel* como hijo de este Canvas. Se utilizará el *Panel* para poder tener un mejor control de la posición de las imágenes y los botones. Se amplía el *Panel* al tamaño total de la ventana pulsando en el cuadrado de *Rect Transform* de la ventana del inspector, conocido como *Anchor Presets*, y en el menú desplegable que aparecerá se pulsará *Shift + Alt* y se seleccionará la opción de la esquina inferior derecha representada por un cuadro con flechas azules que hará que se expanda a todo el ancho de pantalla (*Stretch x Stretch*). Haciendo clic derecho en el panel y el menú UI se seleccionarán dos *Image* y dos *Button*. Una imagen será renombrada como "fondo" y se hará lo mismo que se ha hecho con el *Panel*, esto evitará que las imágenes que se visualicen se confundan con el fondo. Se colocarán los botones correctamente y se cambiará su hijo *Text* al igual que el nombre del botón por la acción que va a realizar, en este caso *Next* y *Previous*. La imagen que nos queda será donde aparecerán las imágenes, para ello, se creará un script que se llamará "Imagen.cs" con el siguiente código:


```

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

public class Imagen : MonoBehaviour {

    public Image MostrarImagen;
    public Sprite[] galeria;
    public Button next;
    public Button prev;
    public int i = 0;

    void Awake()
    {
        galeria = Resources.LoadAll<Sprite>("Lara Croft/Info/img/");
        //galeria = Resources.LoadAll<Sprite>(tmpDatos.Name + "/Info/img/");
    }

    void Start ()
    {
        MostrarImagen = GameObject.Find("Image").GetComponent<Image>();
    }

    // Update is called once per frame
    void Update ()
    {
        MostrarImagen.sprite = galeria[i];
    }

    public void BNext()
    {
        if (i + 1 < galeria.Length)
        {
            i++;
        }
    }

    public void BPrev()
    {
        if (i - 1 >= 0)
        {
            i--;
        }
    }
}

```

Ilustración 23: Código "Imagen.cs"

En este código se cargan todos los recursos de formato que se encuentran en la carpeta indicada en un array⁵ denominado "galería". Esto obliga a conseguir que todas las imágenes sean de tipo *Sprite*, ya que Unity no permite cargar imágenes directamente como .jpg o .png, tan solo como *Textura* o como *Sprite* (Permite más opciones pero que no mostrarían la imagen, como por ejemplo, *Normal Map*), y cargándolas como textura se perdería la resolución y proporción real de la imagen. Para convertir las imágenes en *Sprite* solo se debe seleccionar la imagen en la ventana de proyecto y, una vez

⁵ Tipo de dato estructurado que permite almacenar un conjunto de datos homogéneo, es decir, todos ellos del mismo tipo y relacionados.

seleccionada, en la ventana del inspector cambiar la opción Texture Type de *Texture* a *Sprite* (2D and UI). Una vez cargados en el método *Awake()*, en el método *Start()* se instancia donde se mostraran las imágenes, que será el segundo objeto *Image*, el mismo que contendrá este script. En el método *Update()* se encarga de mostrar la imagen cargándola en el UI *Image* seleccionado en el método *Start()* y con las funciones *BPrev* y *BNext*, se avanzará o volverá en la galería, siempre comprobando no salirse de los índices del array.

Se colocará el script en el UI *Image* que queda sin renombrar y se añadirá en el inspector las referencias indicadas. En Mostrar Imagen se arrastrará el propio UI Image, ya que es aquí donde se quieren mostrar las imágenes, y en las referencias *Next* y *Prev* se añadirán los dos botones que se han creado en su lugar indicado (Ilustración 24).

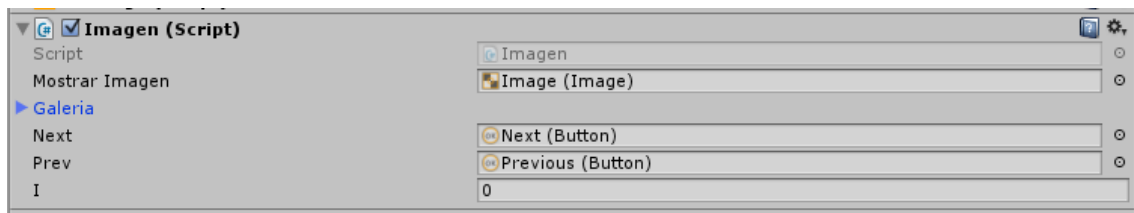


Ilustración 24: Ventana del inspector del script "Imagen.cs"

Como sucede lo mismo que con la pantalla de video, se desactivará el Canvas "Imágenes" hasta que se desarrolle un gestor de pantallas y se comenzará a crear la interfaz para los textos de información.

Para la interfaz de texto hay que enfrentarse a un nuevo problema, no se conoce la longitud del texto y puede que este no entre en la pantalla, por lo que la solución propuesta es añadir una *ScrollBar* para poder desplazarse por el texto en caso de que este sea de un gran tamaño.

Se va a crear un Canvas que se nombrará "Texto", seguido de un *Panel*. El *Panel*, al igual que se ha hecho antes, se va a expandir a todo el ancho de pantalla seleccionando el Anchor Presets de *Rect Transform* y seleccionando con *Shift + Alt* la opción *Stretch x Stretch*. (Ilustración 25).

Seguidamente se crea una *Image* como hijo del *Panel*, a este *Image*, pulsando clic derecho, se seleccionará "Create Empty", creando un *GameObject* vacío como hijo de *Image* y a este *GameObject* se le crea un hijo de tipo UI Text. En resumen, la Jerarquía de este canvas será: *Texto(Canvas) > Panel > Image > GameObject > Text*.

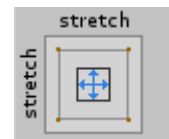


Ilustración 25: Botón Anchor Presets

Se colocarán todos los elementos con la posición de anclaje mencionada en el párrafo anterior y se comenzará a situar los elementos. Se colocará el objeto Text en la posición deseada y en la ventana del inspector se podrá seleccionar la fuente, la alineación que se desee que tengan los textos y más características sobre el texto. Además, se le añadirá el componente *Content Size*⁶, y se cambiará la opción *Horizontal Fit* por *Unconstrained*.

⁶ Tamaño del contenido

En el objeto *Image*, se le añadirán los componentes *Mask* y *Scroll Rect*, para poder conseguir que, en caso de encontrarse con un texto que se quede fuera de pantalla se pueda hacer *Scroll*⁷. En la ventana del inspector se arrastrará el texto a la variable *Content* y se deshabilitará el *Scroll Horizontal*, ya que antes se ha bloqueado el texto para que se quede dentro de la máscara y no se expanda horizontalmente sino verticalmente. Se pueden cambiar diversos parámetros e incluso añadir una barra de *Scroll gráfica*, pero en esta aplicación se utilizará únicamente el ratón para realizar el *Scroll*. Una vez ya se tienen desplegados todos los elementos gráficos toca programar el script.

Se creará un nuevo script llamado "*Textos.cs*", el cual tan solo leerá de un archivo TXT un texto que cargará en un *TextAsset* denominado txt a través de la función *Resources.Load* y este texto se mandará a la variable "*content*" de tipo *Text* que se ha creado al iniciar el script con "*content.text = txt.text;*". Esta variable tendrá que dársele un valor en el inspector, por lo que, se añadirá el script al *Panel* del *Canvas* "*Texto*" y se arrastrará el objeto *Text* al interior de "*content*".

Una vez se tiene ya toda la funcionalidad del visor desarrollada, se va a crear un script que gestione esta información. Por lo que, en la carpeta script, se añadirá uno nuevo llamado "*Menus.cs*". Este script se encargará de mostrar u ocultar los diferentes menús que han sido diseñados, es decir, se tiene que automatizar el proceso que se ha ido haciendo hasta ahora de ocultar los menús, por lo que se van a crear las diferentes variables de tipo *Transform* y asignarles una tecla. Si al pulsarse la tecla están activadas, se desactivan y viceversa. Se añadirá este script al *ObjectManager* y en la ventana del inspector se arrastrará los diferentes *Canvas* que han sido creados a su respectiva variable.

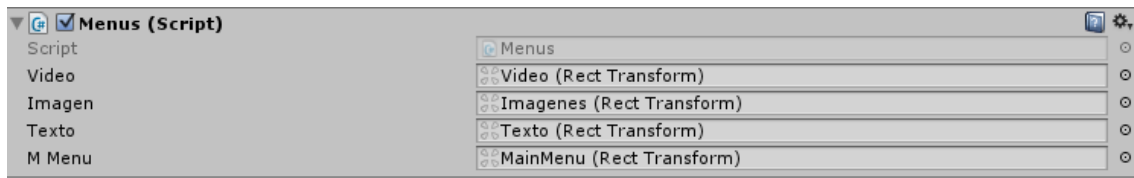


Ilustración 26: Ventana del inspector del script "*Menus*"

Para una correcta visualización de la aplicación se debería tener en cuenta que todos los *Canvas* tengan desactivada su visualización cuando se inicie la aplicación excepto el Menú Principal, que es lo que se va a desarrollar ahora.

Para crear el menú principal, se creará un nuevo *Canvas* llamado "*MainMenu*", después se añadirá un objeto *Image* que será el fondo del menú principal y se le pondrá ese nombre. Se añadirán dos nuevas *Images* como hijas de "*Fondo*", una será el logo de la aplicación, por lo que en su componente *Image* se seleccionará el logo en *Source Image*, se colocará en el centro superior del fondo y se renombrará como Logo. La segunda *Image* se deshabilitará el componente *Image* para que sea invisible y se le añadirá el componente *Grid Layout Group*. Además, se creará un Botón como hijo. Por tanto, a

⁷ Movimiento en 2D de los contenidos que conforman el escenario de un videojuego o la ventana que se muestra en una aplicación informática

estas alturas ya tendremos toda la Jerarquía del proyecto desplegada y sería algo como la siguiente imagen: (Ilustración 27).

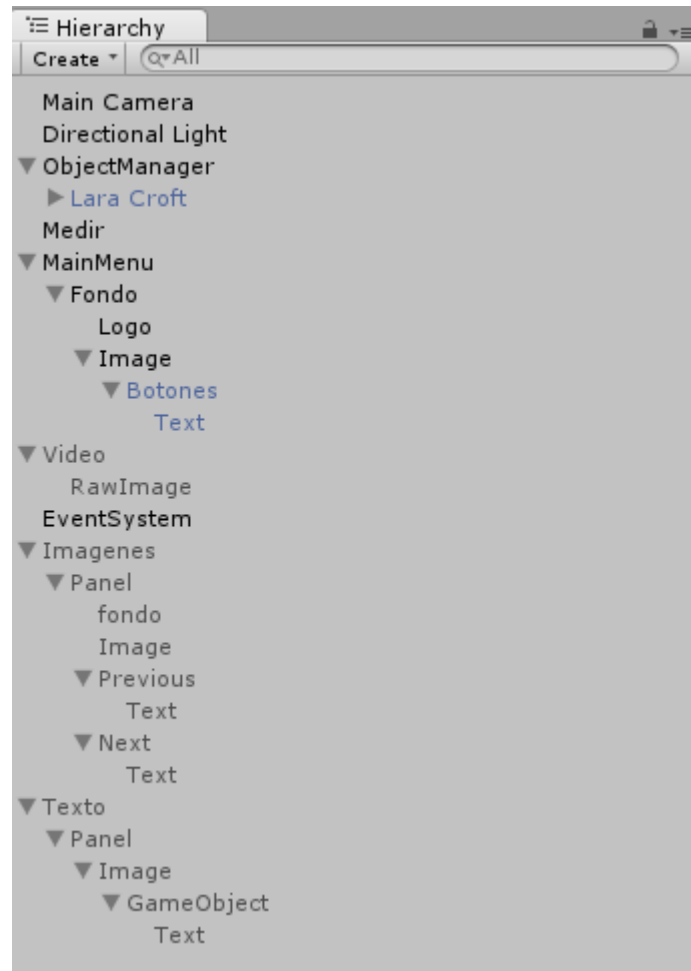


Ilustración 27: Ventana Jerarquía del proyecto

El problema que se encuentra en el menú principal de la aplicación es que los botones se tienen que generar en tiempo de ejecución, ya que tienen que leer de la base de datos, así que se va a proceder a gestionar la base datos.

Primero, se deberá crear la base de datos, por lo que se dejará Unity un momento de lado y se trabajará con SQLite. Para gestionar la base de datos, se utilizará una extensión gratuita de Mozilla Firefox que se llama SQLite Manager. Una vez instalada, se ejecutará la extensión para comenzar con el proceso de creación de la base de datos de la aplicación.

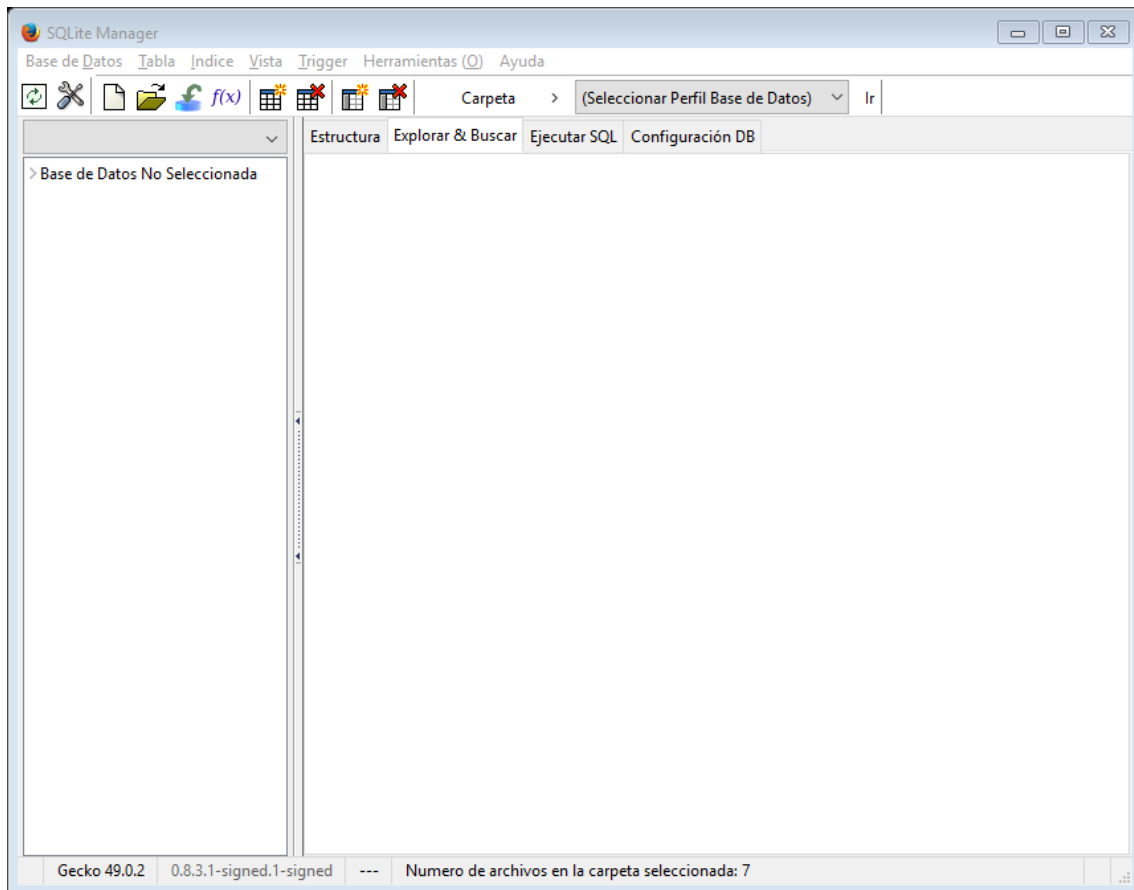


Ilustración 28: SQLite Manager

Se seleccionará “*Base de Datos > Nueva Base de Datos*”. Aparecerá una ventana emergente para poner un nombre a la base de datos, en este caso “*VisorDB*”. En este momento se tendrá una base de datos creada pero vacía, así que se crearán las tablas necesarias. Se seleccionará “*Tabla > Crear Tabla*” y se crearán las tablas Modelo y Piezas tal como se indica en el apartado persistencia (Ilustración 17). Una vez creadas las tablas, se pueden insertar registros nuevos desde la aplicación o desde el script PHP que se ha desarrollado (Véase siguiente apartado). Para poder trabajar en Unity, se añadirá por lo menos un registro por el medio que se prefiera y se regresará al editor de Unity.

Para poder trabajar con SQLite se deben añadir ciertos plugins a Unity. Para ello, se crea una nueva carpeta en la carpeta de Assets con el nombre de “*Plugins*”. En su interior se copiará el plugin “*sqlite3.dll*”. Este plugin está disponible en la web oficial de sqlite (<http://www.sqlite.org/download.html>), donde se deberá descargar el necesario para la arquitectura de nuestro PC, ya sea de 32bits o de 64bits. En caso de que el plugin se denomine de otra forma, como por ejemplo, “*sqlite3_64.dll*” se tendrá que renombrar a “*sqlite3.dll*”. Además, será necesario dos archivos .dll más, que habrá que modificar cada vez que se cambie de equipo al trabajar con Unity. Estos dos .dll son “*System.Data.dll*” y “*Mono.Data.Sqlite.dll*”, que por defecto se encuentran en “*C:\Archivos de programa\Unity\Editor\Data\Mono\lib\mono\2.0*” y solo se tendrán que copiar en la carpeta Plugins.

Se creará un nuevo script que se encargará de gestionar la Base de Datos que se llamará “*DB Manager.cs*”. En este script, en su método Start(), se creará la conexión con la base

de datos a través de la siguiente línea de código : `connectionString = "URI=file:" + Application.dataPath + "/VisorDB.sqlite"`; Se utilizará `Application.dataPath` para que se pueda acceder a la base de datos sin importar donde se encuentre la aplicación, ya que se referencia al directorio de la aplicación. En el método `Start()` también se encontrará una llamada al método `ShowButtons()` que se desarrollará en este mismo script.

Se creará un método `GetModel()` que será el encargado de acceder a la base de datos a la que se ha conectado en el método `Start()` y, mediante una sentencia SQL, se devolverá la información deseada de la base de datos. Esta información se guardará como una lista de tipo `DB`, que es una clase que se creará seguidamente en otro script.

```
// GetModel is called for read the DB
private void GetModel()
{
    datos.Clear();

    using (IDbConnection dbConnection = new SqliteConnection(connectionString))
    {
        dbConnection.Open();

        using (IDbCommand dbCmd = dbConnection.CreateCommand())
        {
            string sqlQuery = "SELECT * FROM Modelo";

            dbCmd.CommandText = sqlQuery;

            using (IDataReader reader = dbCmd.ExecuteReader())
            {
                while (reader.Read())
                {
                    // Debug.Log(reader.GetString(1) + " - " + reader.GetString(2));
                    datos.Add(new DB(reader.GetInt32(0), reader.GetString(1), reader.GetString(2), reader.GetString(3)));
                }

                dbConnection.Close();
                reader.Close();
            }
        }
    }
}
```

Ilustración 29: Fragmento de código de "DB Manager.cs", método GetModel()

El siguiente método se encargará de generar los botones del menú principal en tiempo de ejecución y en función de los modelos que encontrarán en la base de datos de la aplicación. Para ello, se llamará a `GetModel()` y se irán instanciando un prefab con los botones y la información dada, se le dará al botón el nombre del modelo y se le añadirá la funcionalidad al hacer clic en él, llamando al método `LoadModel()`.

```

private void ShowButtons()
{
    GetModel();
    for (int i = 0; i < datos.Count; i++)
    {
        GameObject tmpObjec = Instantiate(botonPrefab);
        DB tmpDatos = datos[i];

        //tmpObjec.GetComponent<MainMenuScript>().SetButton(tmpDatos.Name);

        tmpObjec.transform.SetParent(botonParent);

        tmpObjec.GetComponent<RectTransform>().localScale = new Vector3(1,1,1);
        tmpObjec.GetComponentInChildren<Text>().text = tmpDatos.Name;
        Button b = tmpObjec.GetComponent<Button>();
        b.onClick.AddListener(() => LoadModel(tmpDatos.Name));
    }
}

```

Ilustración 30: Fragmento de código de "DB Manager.cs", método ShowButtons()

Al método LoadModel() se le pasa un string con el nombre del modelo y este se encarga de instanciarlo en la posición dada. Además, como a este método será llamado al pulsar un botón del menú principal, se le añadirá una línea de código para ocultarlo una vez pulsado el botón.

Se colocará este Script como componente de "ObjectManager" y se le asignará los elementos necesarios a las variables. En "M Menu" se seleccionará el Canvas de MainMenu como se hizo en el Script "Menus", ya que es la variable que permitirá ocultarlo una vez pulsado el botón. En "Object Manager Pos" se arrastrará el GameObject "GameManager", que será la posición donde se cargarán los modelos. A continuación, a "Botón Prefab" se necesitará añadir un prefab para generar los botones del menú principal. Para crear un prefab se utilizará el botón que creamos en el "Main Menu", para ello se arrastrará desde la ventana de jerarquía hasta la ventana de proyecto y automáticamente aparecerá en esta ventana. Se arrastrará el prefab a la variable del DB Manager y se ocultará de la ventana de Jerarquía como se hizo anteriormente con los Canvas. Por último, en "Botón Parent" se arrastrará la Image del "Main Menu", ya que es el elemento del cual dependerán los botones.

Por último, para que este Script funcione se ha creado antes una lista de clase DB pero que no se ha definido dicha clase. Para ello, se creará un Script llamado "DB.cs" con el siguiente código:

```

using UnityEngine;
using System.Collections;

class DB
{
    public string Name { get; set; }

    public string Path { get; set; }

    public string Info { get; set; }

    public int ID { get; set; }

    public DB(int id, string name, string path, string info)
    {
        this.Name = name;
        this.Path = path;
        this.ID = id;
        this.Info = info;
    }
}

```

Ilustración 31: Código "DB.cs"

Cabe matizar, que al contrario que el resto de clases, esta no deriva de *MonoBehaviour*, y simplemente se encarga de crear objetos con la información de la base de datos, permitiendonos trabajar con ellos.

Por último, se añadirán cuatro botones en la parte inferior de la pantalla que permitirán acceder a los diversos menús de forma gráfica sin utilizar el teclado. Por ello, se creará un nuevo Canvas que se nombrará "Botones". En él, se crearán y se situarán los respectivos botones, con un punto de anclaje a la parte inferior de la pantalla para que estos se mantengan siempre en la parte baja sea cual sea la resolución de la pantalla. Se creará un script para controlar los botones, donde se creará un método por cada botón, el interior de cada método realizará la misma función que en el script "Menus.cs". Es decir, se comprobará si el menú que abre esta desactivado, de ser así lo activa o en caso contrario, lo desactiva.

Este script se añadirá al Canvas *Botones*, y además, se accederá al inspector de cada botón y se le añadirá una función *AddListener*. Se arrastrará el canvas *Botones* y en función buscaremos el método del botón correspondiente.

Diseño Web

Esta aplicación es una aplicación web, por ello se ha diseñado una página web para acceder a ella y, además, tener una página de subida de archivos para poder incorporar nuevos modelos a la base de datos de la aplicación.

La pantalla principal es una pantalla sencilla e intuitiva, a la vez que bastante atractiva visualmente con el fin de llamar la atención al usuario y, por otro lado, poder acceder fácilmente a los servicios que ofrece la web. Para ello, se ha diseñado primero en

Photoshop, con el fin de desarrollar una guía que sirva para su posterior diseño web a través de una Hoja de Estilo en Cascada(CSS) y su código HTML.



Ilustración 32: Diseño PSD de la web

Además, el diseño de esta web es el que se denomina “*Diseño Responsive*”. Se trata de redimensionar y colocar los elementos de la web de forma que se adapten al ancho de cada dispositivo permitiendo una correcta visualización y una mejor experiencia de usuario. Es decir, se proporciona a todos los usuarios de esta web los mismos contenidos y una experiencia de usuario lo más similar posible independientemente de la pantalla o dispositivo en la que estemos accediendo, algo muy importante hoy en día debido a la variedad de dispositivos con los que la gente accede a internet, especialmente la gran proliferación de smartphones y tablets.

Para conseguir que nuestra web fuera una “*Responsive Web*” se ha utilizado el siguiente código CSS:

```
1 @charset "utf-8";
2 /* CSS Document */
3
4 * {
5     padding: 0;
6     margin: 0;
7 }
8 .fit { /* tamaño de imagenes relativas */
9     max-width: 100%;
10    max-height: 100%;
11 }
12 .center {
13     display: block;
14     margin: auto;
15 }
16
17 div.footer { position: absolute; right: 0; bottom: 0; left: 0; padding: 1rem;
18 background-color: #efefef; text-align: center;}
19 #main-header {
20     color: white;
21     height: 180px;
22 }
23 #main-header a {
24     color: white;
25 }
26
27 #footer {
28     position: fixed;
29     bottom: 0;
30     left: 0;
31     height: 70px;
32     background-color: white;
33     width: 100%;
34 }
```

Ilustración 33: Código CSS

Y mediante un pequeño script en JavaScript, utilizando la biblioteca jQuery, se ha conseguido que los botones reajusten su tamaño y no se queden nunca fuera de la pantalla.

```

<script src="http://code.jquery.com/jquery-latest.js"></script>
<script type="text/javascript" language="JavaScript">
  function set_body_height() { // set body height = altura de la ventana
    $('body').height($(window).height());
  }
  $(document).ready(function() {
    $(window).bind('resize', set_body_height);
    set_body_height();
  });
</script>

```

Ilustración 34: Script JavaScript para redimensionar botones

Para acceder al visor, será tan sencillo como pulsar el botón “Visor3D MODELOS”, que traslada al usuario a una segunda web donde está alojada la aplicación Unity y donde se podrán seleccionar y visionar los diferentes modelos que se encuentren en la base de datos.

Por otro lado, se encuentra el botón “SUBIR archivos .zip”, este botón enlaza a una segunda página donde se podrán subir nuevos modelos a la base de datos. Esta segunda página, sobria y sencilla para no añadir complejidad a su desarrollo, consta de un formulario PHP en el que los usuarios pueden subir nuevos modelos 3D a la aplicación. Para ello, deberán subir un archivo comprimido ZIP con el modelo 3D en el formato requerido y añadir el nombre y la información del modelo que se añadirá a la base de datos una vez subido, junto con su ruta y un identificador único. Se recomienda que los archivos comprimidos sigan la jerarquía explicada en el Anexo I. (Ilustración 38).

Para la realización de esta página se ha desarrollado un script en PHP que se puede dividir en tres partes: el *Uploader*, el descompresor y la conexión a la base de datos.

El *Uploader* o cargador es la parte del script que se va a encargar de subir nuestro modelo y su información a la red. Está compuesto por dos partes escritas en dos lenguajes diferentes, HTML y PHP. La parte HTML será la parte que verá el usuario al entrar a dicha web, y está compuesta de dos casillas de texto donde escribir el nombre y la información del modelo, y un botón de selección de archivo para seleccionar el archivo zip con el modelo preparado.

El script PHP se encargará de subir el archivo y su información utilizando el método post. La información de texto se guardará en las variables correspondiente, pero ahora se abordará la información en el archivo.

Se recogerán los datos necesarios del archivo como su nombre y extensión, para comprobar que esta sea .zip y no cualquier otra. Si el archivo reúne las características adecuadas, se podrá subir al servidor, sino dará error. Para subir se utiliza la función *move_uploaded_file()*, que recibirá el nombre del archivo temporal que se desea subir y el nombre del archivo que se desea dar.

Cuando se sube el archivo, el servidor lo copia en una carpeta temporal para que sea el programador quien elija su posición definitiva donde se quiere que este lo guarde, si no se copiara en ningún lado, el archivo sería eliminado al finalizar la ejecución de la página.

Por ello, con la función `move_uploaded_file()` se moverá el archivo a la posición definitiva. Por un lado, recibirá el nombre temporal y, por otro lado, el nombre definitivo y la ruta para llegar al directorio donde se quiere guardar. Si el archivo no es subido con éxito, saltará un error. En caso de éxito, se accederá al fragmento de código del descompresor.

El descompresor es muy sencillo, se creará una nueva variable de tipo `ZipArchive`, la cual abrirá el archivo recientemente subido mediante el método `open()`, lo extraerá en la ubicación deseada mediante el método `extractTo()` y finalmente se cerrará el archivo utilizando la función `close()` y se notificará del éxito de la extracción. En caso de no haber podido abrir el archivo, el script notificará el error.

La conexión con la base de datos SQLite se realiza en dos fases.

- La primera fase es la propia conexión que se realiza creando una clase llamada `MyDB` que extiende de `SQLite3`, donde se le pasa la ruta de nuestra base de datos para conectarnos a ella.
- El siguiente paso es la inserción, donde una vez enviado el formulario (es decir, pulsado el botón “`submit`” para que se realice el POST) se recoge la información recibida por el formulario, cada una en una variable y se añade mediante una instrucción SQL. Esta se ejecuta, notificando éxito o fracaso, y se cierra la conexión con la base de datos.

Cabe mencionar que se tendrá que tener la jerarquía de carpetas correctamente situadas en nuestro servidor o el Uploader no funcionará correctamente. Es decir, si el Uploader sube un archivo a “`upload/`” pero dicha carpeta no existe, el archivo nunca podría situarse ahí y, por tanto, no sería subido al servidor.

Instalación y utilización de XAMPP

Para poder subir la aplicación a la web antes se necesita probarla en local. Para ello, se instalará en el ordenador el servidor XAMPP. Lo primero que se necesita es descargar el instalador que se encuentra en su web oficial correspondiente al sistema operativo que se esté utilizando. La instalación es tan sencilla como seguir las indicaciones del *launcher*, pero comprobando que se instalan los componentes necesarios para probar la aplicación, como en este caso, PHP, phpMyAdmin, etc.

La carpeta de instalación predeterminada es `C:\xampp`, aunque puede cambiarse durante la instalación. Allí se copiarán los archivos de nuestro proyecto para poder visualizarlos en `http://localhost`.

Para iniciar el panel de control se accederá al mediante el menú de inicio: “*Todos los programas > XAMPP > XAMPP Control Panel*” o en el icono del área de notificación en caso de que ya estuviera iniciado.

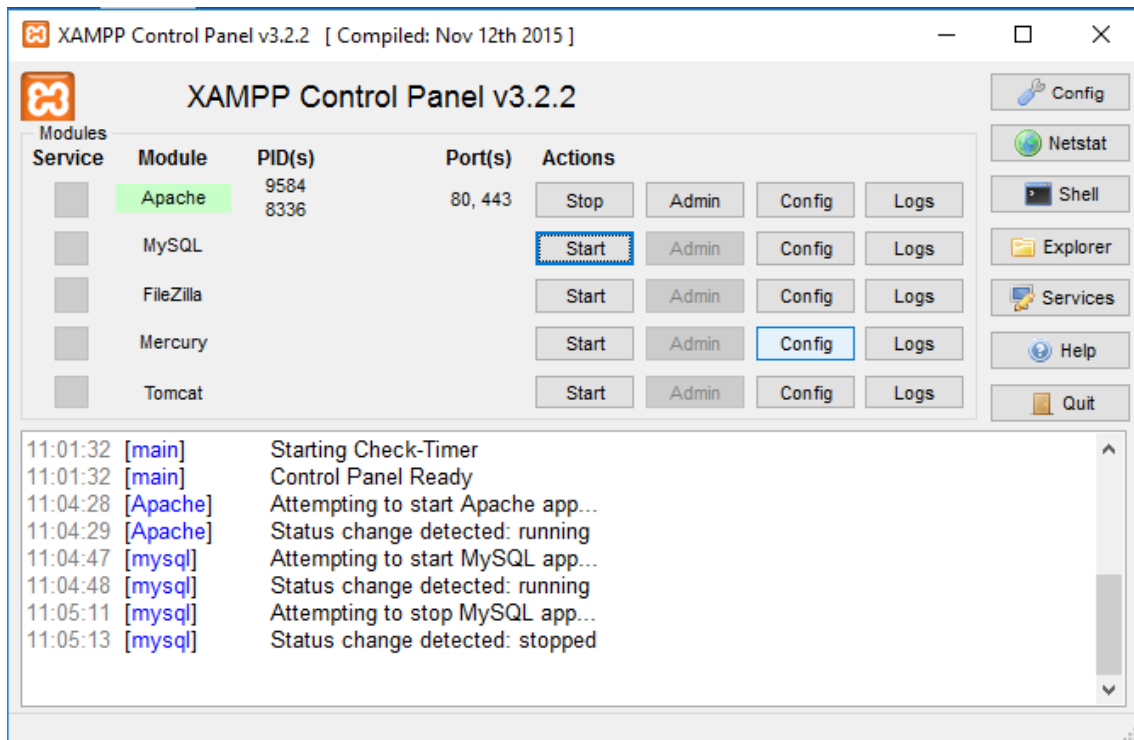


Ilustración 35: Panel de control de XAMPP

El panel de control de XAMPP está dividido en tres zonas:

- La zona de módulos, que indica para cada módulo si está instalado, su nombre, el identificador, el puerto que utiliza y además los botones para iniciar y detener los procesos, administrarlos, editar los archivos de configuración y abrir sus logs o registros de actividad.
- La zona de consola o de notificación, donde XAMPP informa de las acciones realizadas y su estado (Éxito o fracaso)
- Y la zona de utilidades que permite acceder rápidamente a ellas.

Para arrancar el servidor local se deberá empezar poniendo en marcha Apache mediante el botón “Start” correspondiente. Al arrancarse por primera vez en Windows, el cortafuegos pedirá confirmación al usuario ya que Apache abre puertos en el ordenador. El panel de control mostrará el nombre del módulo con fondo verde si el arranque de Apache tiene éxito, el botón “Start” pasará a ser “Stop” y en la zona de notificación se verá el resultado de las operaciones realizadas. Se iniciarán del mismo modo el módulo MySQL y FileZilla.

Para poder utilizar esta aplicación, se necesitará que nuestro servidor tenga habilitado las bases de datos SQLite en PHP, para ello se tendrá que modificar el fichero “php.ini”. Para acceder a este archivo, se deberá pulsar sobre el botón Config que corresponde a Apache y seleccionar PHP (php.ini).

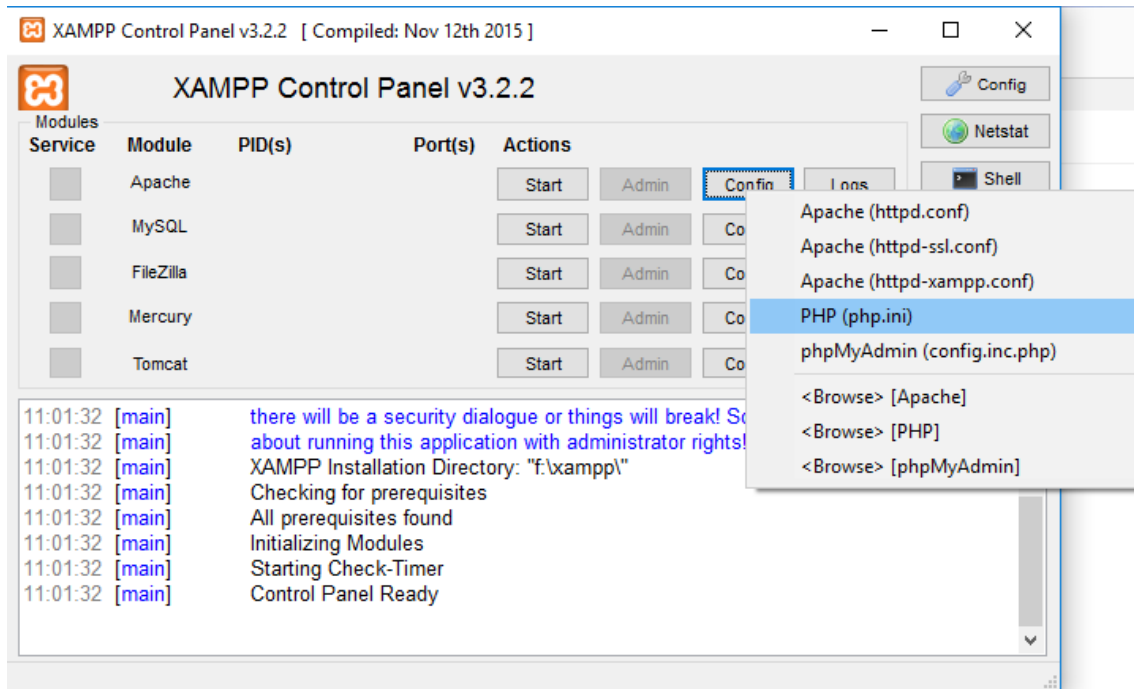


Ilustración 36: Modificar PHP.INI en XAMPP

Por precaución, antes de modificar cualquier valor del archivo de configuración, se recomienda hacer una copia de seguridad de este.

Se abrirá el fichero en el bloc de notas o procesador de texto que tenga el usuario por defecto. Ahora solo se tendrá que buscar la línea `“;extension=php_sqlite3.dll”` y descomentar esta, es decir, eliminar el punto y coma del principio. Se guardará el archivo y reiniciará el servidor Apache en caso de que ya estuviera arrancado para que los cambios sean válidos.

Se accederá desde cualquier navegador a localhost para comprobar el correcto funcionamiento de la web y poder corregir errores. Una vez todo sea correcto, se podrá utilizar FileZilla para subir los archivos al servidor web (no el servidor local de Apache) para que nuestra web ya sea accesible para todo el mundo.

Conclusiones

Tras la realización de este proyecto y su proceso de investigación y desarrollo, se ha conseguido desarrollar una aplicación que cumple con todos los objetivos propuestos en el comienzo de este documento siempre que el software actual lo ha permitido. Además, mientras se llevaba a cabo este proyecto han surgido diferentes cuestiones que se desarrollarán en este apartado, aportando varias soluciones para llevarlas a cabo en un futuro. En resumen, se ha logrado crear una aplicación base en la que, con tiempo y esperando nuevas actualizaciones en la API de Unity, nos permitan ir expandiendo y mejorando esta aplicación en diferentes proyectos.

Por una parte, en lo que atiende a la parte personal, con este proyecto ha servido para utilizar muchas de las características que aún no había utilizado de Unity, ya que, pese a que ya había utilizado este motor, había utilizado una versión anterior al 5, con el cambio que esto supone sobretodo en el tema de interfaces y su nuevo sistema UI. Sin duda, en el mercado laboral actual, con la proliferación de la RA y RV y el mundo del videojuego independiente, es indispensable tener cierto nivel de programación en motores de videojuego como Unity.

Por otra parte, me he introducido y mejorado mis conocimientos en programación web y C# respectivamente, y creo que me resultarán muy útiles en el futuro, además de aprender a gestionar bases de datos desde aplicaciones externas, algo que no había hecho hasta ahora. También destacar el hecho de enfrentarme al desarrollo de una aplicación web, ya que, al haberme especializado en computación durante la carrera, me ha supuesto un auténtico desafío.

Posibles mejoras, trabajos futuros y aplicaciones

Aunque se ha conseguido una aplicación funcional, las sensaciones con Unity no han sido tan satisfactorias como se esperaba debido a las limitaciones que el propio programa ha dado, que se enumerarán a continuación y se sugerirán diversas soluciones cuando las haya, además, se propondrán diversas ideas surgidas durante el desarrollo del proyecto para la realización de trabajos futuros respecto a este trabajo.

- **Modelos .STL:** Unity solo puede leer archivos .FBX, .DAE(Collada), .3DS, .DXF y .OBJ. Por lo que nuestra aplicación solo puede leer formatos de archivo 3d exportados, es decir: .FBX y .OBJ, pero no pudiendo trabajar con .STL. Como posible solución, en la Asset Store de Unity encontramos diferentes Plugins de pago que nos permiten cargar archivos STL en tiempo de ejecución.
- **Mostrar videos:** Como se puede leer en la memoria, la visualización de videos ha sido desarrollada completamente, pero debido a que Chrome y nuevos navegadores han dejado de dar soporte al Unity Web Player, lo que obligada a exportar la aplicación en WebGL, lo cual no es compatible actualmente con las MovieTexture, impidiendo exportar el archivo. Se ha intentado utilizar los



diferentes plugins gratuitos que se han encontrado en la Asset Store de Unity como WebGLMovieTexture, pero Unity no ha conseguido reproducir el video, por lo que se optó por suprimirlo, dejando el código escrito para futuras actualizaciones de la aplicación, a la espera de que Unity le de compatibilidad en un futuro.

- **SQLite:** se eligió esta base de datos debido a su comodidad y pensando en una futura implementación en sistemas móviles como Android o iOS, pero tras la realización de toda la aplicación con este formato, por algún motivo al estar el archivo en red (tanto local como internet) la base de datos no conecta mientras que dentro de la aplicación si, por lo que, como primer paso a mejorar, sería cambiar la base de datos SQLite por una MySQL. Debido a este motivo, no se ha podido realizar la muestra de piezas o fragmentos del modelo.
- **Automatización:** Debido a la forma de trabajar de Unity, no se puede automatizar el proceso de subida de archivos, es decir, pese a que los archivos se suben y se exportan en la carpeta indicada, añadiendo su registro en la base de datos, la aplicación necesitará volver a exportarse y hacer algunas correcciones antes de que aparezcan nuevos modelos, ya que, por ejemplo, para cargar el modelo se necesita crear un prefab de él.

En cuanto a posibles aplicaciones, este visor puede tener muchas aplicaciones en diferentes ámbitos.

- **Videojuegos:** Esta aplicación se puede gastar en el mundo de los videojuegos: desde incrustar el visor como parte del videojuego para visualizar personajes, etc... hasta la utilización del visor para el trabajo de los artistas que trabajan en el desarrollo de este, para trabajar con ellos, ver cambios, mostrar los modelos a los GameDesigner y/o directores y productores del proyecto entre otras opciones.
- **Arqueología y museos:** Al igual que el museo Smithsonian, se puede utilizar el visor para mostrar modelos de piezas arqueológicas escaneadas 3D y poder trabajar con ellas. Del mismo modo, este visor puede mejorar la forma de trabajo de muchos historiadores y arqueólogos, ya que, si surge la necesidad de estudiar una pieza, con la utilización de este visor se omite la necesidad de desplazarte hasta el lugar donde se conserva dicha pieza, ya que puede trabajar con ella a través de cualquier ordenador, evitando así el desplazamiento y el posible deterioro y exposición al que se sometería la pieza.
- **Impresión 3D:** Se puede utilizar este visor para visualizar modelos 3D antes de mandarlos a imprimir.

Bibliografía

Bimber O., Raskar R.; (2005) Spatial augmented reality: merging real and virtual worlds. Pages 71-92

Blackman S.;(2011). Beginning 3D Game Development with Unity: The World's Most Widely Used Multiplatform Game Engine. New York, USA: Apress. Capítulos: 2,5-9.

Cabero, J.; García, F.; (2016) Realidad Aumentada: Tecnología para la formación. Capítulos: 1-2,5,7-9, Sevilla, España: Editorial Síntesis.

Esteve Guinot, A. (2014). Scares for sale: diseño y desarrollo de un videojuego en Unity 3D. Arquitectura de GUI. Páginas: 16-23. <http://hdl.handle.net/10251/48367>.

Feuerstein, M.; Navab, N.; and Bichlmeier, C.; (2007) Laparoscopic virtual mirror: New interaction paradigm for monitor based augmented reality. In Virtual Reality Conf, page 43–50. IEEE Press

Fombona, J., Pascual, M. J. y Madeira, M. F. (2012). Realidad aumentada, una evolución de las aplicaciones de los dispositivos móviles. Pixel-Bit. Revista de Medios y Educación, 41, 197-210.

García Jiménez, F. (2014). Fundamentos psicológicos de la efectividad de la realidad aumentada. Comunicación y Pedagogía, 277-278, 91-96.

Henrysson, A.; Billinghurst, M.; and Ollila M.; (2006) Ar tennis. In SIGGRAPH '06: ACM SIGGRAPH 2006 Sketches, page 13, New York, NY, USA. ACM

Hiroshi, I.; Brygg, U.; Tangible bits: towards seamless interfaces between people, bits and atoms. In CHI '97: Proceedings of the SIGCHI conference on Human factors in computing systems, pages 234–241, ew York, NY, USA, 1997. ACM.

inScope Studios. (2015, 08, 30) Unity tutorial: High score with SQLite. Recuperado de: https://www.youtube.com/watch?v=laspFwXGprg&list=PLXuZVK_0K_7NmsYfe2BTOk_IamWC2kU3&index=1 (Consulta: 19/11/16)

Looser, J.; Billinghurst, M.; Grasset, R.; and Cockburn, A.; An evaluation of virtual lenses for object selection in augmented reality. In GRAPHITE '07: Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia, pages 203–210, New York, NY, USA, 2007. ACM.

Marcos Miñarro, P. (2016). Desarrollo de una aplicación de Realidad Virtual. Páginas: 24-32. <http://hdl.handle.net/10251/69203>.

Martens, J.B; Qi, W.; Aliakseyeu, D.; J. F. Kok, A.; and van Liere, R.; (2004) Experiencing 3d interactions in virtual reality and augmented reality. In EUSAI '04: Proceedings of the

2nd European Union symposium on Ambient intelligence, pages 25–28, New York, NY, USA. ACM.

Martínez Abril, J. (2016). Desarrollo de un videojuego de plataformas en C# sobre el motor Unity. Páginas: 43-51. <http://hdl.handle.net/10251/71037>.

Muñoz, J.;(2014) Realidad aumentada una oportunidad para la nueva educación Comunicación y Pedagogía,277-278, 6-11.

Mullen, T. (2012). Realidad aumentada. Crea tus propias aplicaciones. Madrid: Anaya.

Naranjo Ornedo, V. (2009). La realidad virtual al servicio del bienestar social <http://catttelefonica.webs.upv.es/documents/Informe Realidad Virtual.pdf>

Oviatt S.; (1999) Ten myths of multimodal interaction. Commun. ACM, 42 (11):74–81.

Portalés, C.; (2008). Tesis Doctoral: Entornos Multimedia de Realidad Aumentada en el Campo del Arte. Doctorado: Artes Visuales e intermedia. Páginas:67- 69. Sección 1.2.2.

Reitmayr, G.; Eade, E.; and Drummond T.; (2005) Localisation and interaction for augmented maps. In ISMAR '05: Proceedings of the Fourth IEEE and ACM International Symposium on Mixed and Augmented Reality, pages 120–129, Washington, DC, USA,. IEEE Computer Society.

Rohs M.; (2006) Marker-based embodied interaction for handheld augmented reality games. Journal of Virtual Reality and Broadcasting, 4 (5).

Ruiz, D. (2011). La Realidad Aumentada y su dimensión en el arte: la obra aumentada. Arte y Políticas de Identidad, 5, 129-144.

Ruiz-Velasco Sánchez, E.; (2007). Educatrónica: innovación en el aprendizaje de las ciencias y la tecnología. México: Ediciones Díaz de Santos. <https://books.google.es/books?isbn=8479788224>

Sánchez Blázquez, D. (2010). Trabajo de Fin de Grado: Desarrollo de una aplicación de Realidad Aumentada para simulación de moléculas. Páginas: 12 – 14. <http://hdl.handle.net/10251/10240>.

Sánchez Molina, S. (2015). MusGuide. Páginas 43-47. <http://hdl.handle.net/2117/82916>

Straier W Fischer J, Bartz D. Intuitive and lightweight user interaction for medical augmented reality. In Proceedings of vision, modeling, and visualization, pages 375–82, 2005.

Enlaces web

API de Unity [Internet] Disponible en: <https://docs.unity3d.com/Manual/index.html>

CryEngine. [Internet] Disponible en: <https://www.cryengine.com>

MongoDB. [Internet] Disponible en: <https://www.mongodb.com/es>

PHP. [Internet] Disponible en: <http://php.net/manual/es/intro-what-is.php>

SQLite. [Internet] Disponible en: <https://sqlite.org>

System properties comparison MongoDB vs. SQLite. [Internet] Disponible en <http://db-engines.com/en/system/MongoDB%3BSQLite> (Consulta: 19/11/16)

Tienda de Unity. [Internet] Disponible en: <https://www.assetstore.unity3d.com/es/>

Unity, Source2, UnrealEngine4 or CryEngine – Which game engine should i choose? [Internet] Disponible en: <http://blog.digitaltutors.com/unity-udk-cryengine-game-engine-choose/> (Consulta: 19/11/16)

Unity3D. [Internet] Disponible en <http://unity3d.com/es>

Unreal Engine 4. [Internet] Disponible en: <https://www.unrealengine.com/>

XAMPP. [Internet] Disponible en <https://www.apachefriends.org/es/>

Anexo I: Manual de usuario

Menú Principal



Ilustración 37: Menú Principal

Esta es la pantalla principal de la aplicación, donde encontramos dos botones que nos enlazan a la aplicación (el botón de la izquierda) o a la página de subida de modelos (botón derecho). Para ello, simplemente debemos clicar con el ratón en la opción deseada.

Menú Upload

En esta pantalla secundaria encontramos un formulario que nos facilitará la incorporación de nuevos modelos a la base de datos de la aplicación. Arriba del formulario encontramos información sobre la conexión con la base de datos, lo cual nos informará si la conexión ha sido correcta; al igual que debajo del formulario sabremos si la subida ha sido satisfactoria o no. Por cuestiones de automatización, se recomienda subir los archivos comprimidos en ZIP con la siguiente jerarquía.

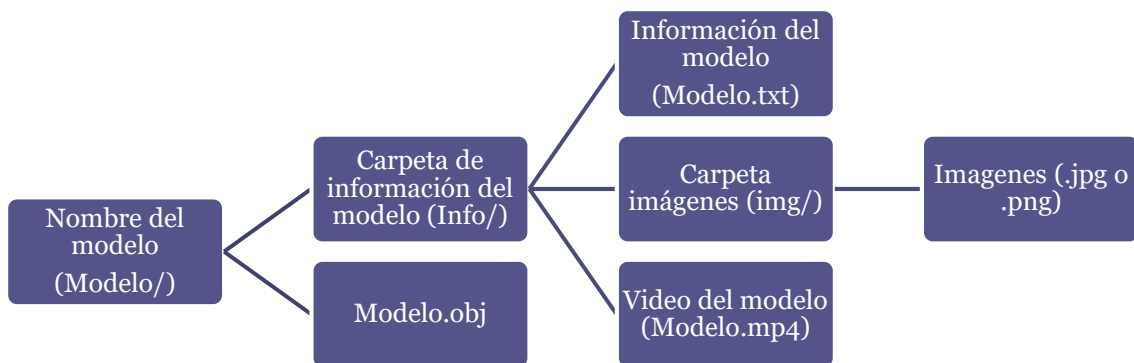


Ilustración 38: Jerarquía Uploads

Menú Modelos



Ilustración 39: Menú modelos/Menú Principal

Realmente es el menú principal de la aplicación Unity. Aquí encontramos todos los modelos que el visor ha encontrado en la base de datos, el usuario deberá seleccionar uno para poder visualizarlo.

Aplicación

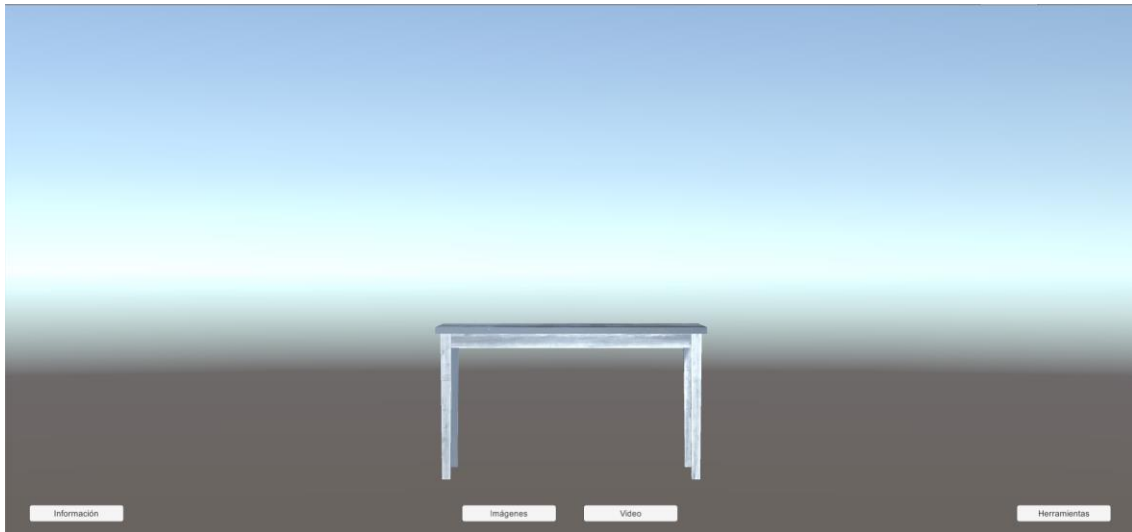


Ilustración 40: Ventana de la aplicación

Tendremos el modelo con el cual podremos interactuar utilizando los controles explicados en la siguiente tabla. Podemos rotar, hacer zoom, visualizar información, etc. Para acceder, primero hay que pulsar sobre el botón herramientas, que desplegará esta sección. Para medir, debemos clicar en el *checkbox* "Medir" que encontramos en la parte superior izquierda de la pantalla. Para dejar de medir debemos desactivar el *checkbox* clicando nuevamente en él.

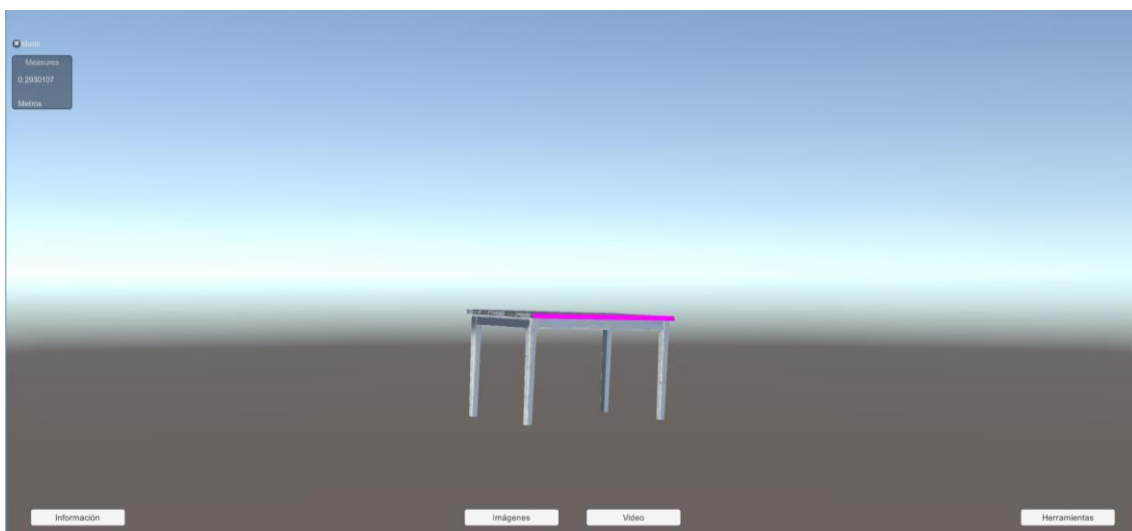


Ilustración 41: Aplicación con la opción medir activada

Controles

Tecla	Función
I	Accederemos o saldremos de la galería de imágenes del modelo. (Si contiene imágenes)
T	Accederemos o saldremos de la información de texto del modelo. (Si contiene información)
V	Accederemos o saldremos del video del modelo. (Si contiene video)
Espacio	Con la ventana de video abierta, nos servirá para pausar o reproducir el video.
R	Resetea el modelo a su posición inicial
Escape	Regresamos al menú principal de la aplicación (Menú modelos)
Rueda del ratón	Zoom
Clic izquierdo (mantener) + Mover ratón	Rotar
Clic izquierdo (medir activado)	Selecciona los puntos para medir el objeto
Clic izquierdo (Menú)	Selecciona modelo pulsando el botón correspondiente.

Imágenes

A esta ventana accederemos pulsando la tecla “I” en el teclado o el botón Imágenes. Nos muestra una galería de imágenes, y para desplazarnos por ella solo tendremos que pulsar los botones que encontramos bajo la imagen. Si queremos salir de la ventana basta con pulsar nuevamente la tecla “I” o el botón Imágenes.

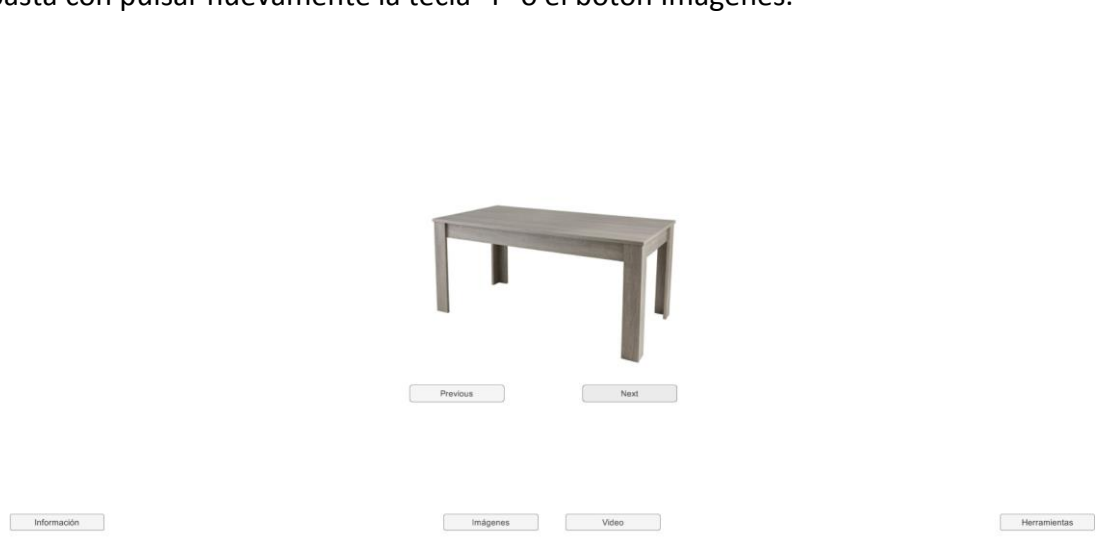


Ilustración 42: Galería de imágenes

Textos

A la ventana de textos accederemos pulsando la tecla “T” o el botón Información. Si el texto es muy largo y no cabe en pantalla podemos desplazarnos por el haciendo clic con el ratón y arrastrándolo. Para cerrar la ventana simplemente deberemos volver a pulsar la tecla “T” o el botón Información.



Ilustración 43: Aplicación mostrando texto

Videos

A la ventana de video se accede mediante la tecla “V” o el botón Video. El video comenzará automáticamente, pero podemos pausarlo con la tecla “Espacio”. Para volver a reproducirlo bastará con pulsar otra vez la tecla “Espacio”. Si salimos de la ventana de video pulsando la tecla “V” o el botón Video antes de que finalice este seguirá con la reproducción en segundo plano (No se oirá ni verá), pero al volver a la ventana video esté habrá avanzado, por lo que se recomienda pausar antes de salir, o esperar a que acabe para volverlo a reproducir.

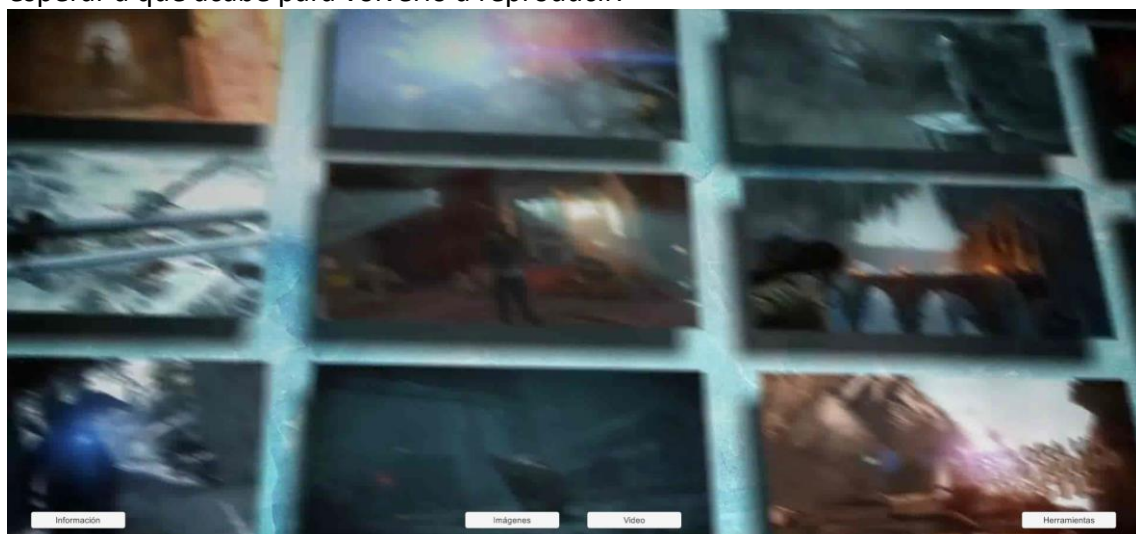


Ilustración 44: Aplicación mostrando video

Anexo II: Plan de trabajo

Como todo proyecto de ingeniería, el desarrollo de esta aplicación debe ser una actividad que disponga de un plan de trabajo. Por eso, a continuación se presenta un plan de trabajo propuesto al inicio del proyecto.

Mes	Septiembre					Octubre					Noviembre			
Semana	01-04	05-11	12-18	19-25	25-02	03-09	10-16	17-23	24-30	31-6	7-13	14-20	21-27	27-30
Búsqueda de información y documentación	■	■	■	■	■									
Estudio y diseño de la aplicación				■	■	■								
Implementación y desarrollo de las funcionalidades de la aplicación						■	■	■						
Implementación y desarrollo de la base de datos de la aplicación								■	■	■				
Implementación y desarrollo de la parte web de la aplicación						■				■				
Escritura de la memoria				■	■					■	■	■		
Corrección de errores											■	■	■	

Ilustración 45: Plan de trabajo

Con este plan se proponen las fases necesarias para poder terminar con éxito el desarrollo de este proyecto. Comenzando el proyecto en septiembre, se propone dedicar ese primer mes para realizar todo el trabajo de investigación y documentación, tanto para la escritura de la parte teórica de la memoria como asimilación de nuevos conocimientos necesarios para el desarrollo de ésta. A mitad del primer mes, una vez recabada la información, se propone comenzar la escritura de los primeros capítulos de la memoria y comenzar con el estudio y el diseño de la aplicación.

En el comienzo de octubre, una vez que ya se ha estipulado como diseñar la aplicación y recabado toda la información, se propone comenzar con la implementación de las funcionalidades de la aplicación en Unity (*Movimientos, Menús*), a la vez que el desarrollo de la parte web que no depende de la base de datos (*Página principal, Formulario de subida sin conexión a la base de datos*). Una vez esté desarrollado esto, se comenzará con la creación de la base de datos y su conexión a la aplicación Unity y, finalmente, al Uploader. Finalmente, se realizarán las últimas pruebas y se finalizará la memoria.

El último mes se dedicará principalmente a la corrección de errores y dejar finalizado este proyecto.