UNIVERSIDAD DE CASTILLA-LA MANCHA
ESCUELA SUPERIOR DE INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE GRADO

Aix: Plataforma para la producción 3D en sistemas de escenografía virtual

Alberto Izquierdo Ramírez

Septiembre, 2016

AIX: PLATAFORMA PARA LA PRODUCCIÓN 3D EN SISTEMAS DE ESCENOGRAFÍA VIRTUAL

UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INFORMÁTICA

Departamento de Tecnologías y Sistemas de Información

TECNOLOGÍA ESPECÍFICA DE
INGENIERÍA DEL SOFTWARE

TRABAJO FIN DE GRADO

Aix: Plataforma para la producción 3D en sistemas de escenografía virtual

Autor: Alberto Izquierdo Ramírez

Director: Dr. Carlos González Morcillo

Septiembre, 2016

Alberto Izquierdo Ramírez

Ciudad Real – Spain

E-mail: alberto.izquierdo2@alu.uclm.es
Teléfono: 658 566 626

TRIBUNAL:

Presidente:

Vocal:

Secretario:

FECHA DE DEFENSA:

CALIFICACIÓN:

PRESIDENTE	VOCAL	SECRETARIO
Fdo.:	Fdo.:	Fdo.:

Resumen

En apenas unos 8 años de vida, los cursos masivos online (o MOOC) se han extendido convirtiéndose en uno de los principales medios de enseñanza a distancia. Estos cursos están formados en gran parte por material en formato de vídeo, más concretamente en el formato ”polimedia”, donde aparece la imagen del profesor complementada con diapositivas.

Este formato de composición permite abaratar costes y focalizar la atención del estudiante en el material más relevante. Gracias a la escenografía virtual, ahora es posible realizar producciones audiovisuales que mantengan la atención del alumno con cámaras móviles virtuales que están basadas en una única cámara real, un micrófono y un equipo preparado para el montaje de este tipo de escenarios.

A pesar de ello, existen algunos inconvenientes con los sistemas de escenografía virtual. Uno de ellos es debido a que la mayor parte de las soluciones de calidad profesional son cerradas y la empresa desarrolladora no ofrece un soporte adecuado para la personalización o para la integración de herramientas externas de representación.

El proyecto Aix surge para mitigar estos problemas, ofreciendo una plataforma basada en estándares y software libre para la producción 3D integral en sistemas de escenografía virtual, con un caso de implantación específico en el entorno EasySet de Brainstorm, el cual es utilizado en el Centro de Tecnologías y Contenidos Digitales de la UCLM.

Abstract

In just 8 years, massive open online courses (MOOC) have grown becoming one of the main tools in distance learning. These courses consist mainly of material in video format, specially in the ”Polimedia” format, where a professor appears complemented with slides.

This format allows to reduce the costs and to focus the student's attention on the most relevant material. Thanks to virtual scenery, it is now possible to produce audiovisual content to keep the student's attention using multiple virtual mobile cameras that are based on a single real camera, a microphone and a team prepared to build such scenarios.

However, there are some drawbacks to virtual scenarios. One of them is the fact that most of the best solutions are closed source and the developer company does not provide an appropiate support of customization or to integrate external representation tools.

The Aix project tries to mitigate these problems by offering a platform based on open standards and open source for the integrated 3D production in virtual scenery systems, with a specific implantation case in the EasySet environment of Brainstorm, which is used in the Technology and Digital Contents Center of the UCLM.

Agradecimientos

Gracias a toda mi familia por su apoyo. A mis padres por apoyarme incluso en los momentos en los que parecía que no iba a ser capaz de salir adelante. A mi hermana, cuya capacidad de enfrentarse a lo que aparezca por el camino me ha inspirado tanto a lo largo de mi vida. A mi abuela Amalia, la persona más trabajadora que conozco y sin la cual todos sus nietos no nos habríamos ”estrujado” para sacar las cosas adelante. A la gente de Alcázar por su alegría incluso en los momentos difíciles. A todos los que me dejo y que me han apoyado, gracias, os quiero.

Gracias también a mis amigos, sin ellos estos años no habrían sido igual.

Gracias a los profesores por compartir sus conocimientos y pasión por la informática que tan bien me han sabido transmitir. Más concretamente a mi director de proyecto, Carlos, por haberme transmitido la pasión por los gráficos por computador y darme una meta por la que luchar.

Por último, gracias a todos y cada uno de mis compañeros del CTED, gracias a los que he aprendido tanto profesionalmente como personalmente.

A todos os deseo lo mejor, gracias.

Alberto

A mi familia

Índice general

Índice de cuadros

Índice de figuras

Índice de listados

Listado de acrónimos

GNU GNU is Not Unix

FOV Field of View

CTED Centro de Tecnologías y Contenidos Digitales

BGE Blender Game Engine

MOOC Massive Open Online Course

FPS Frames Per Second/Frames Por Segundo

Capítulo 1

Introducción

1.1 Historia de la escenografía virtual

A pesar de que los primeros trabajos utilizando un chroma se remonta al cine de los años 50, para el primer uso de un escenario en tres dimensiones en la televisión tenemos que avanzar hasta el año 1994 en la empresa Ultimatte en colaboración con IMP en el NAB [DEGC].

En ese mismo año, la empresa valenciana Brainstorm, en colaboración con Antena 3, realiza los primeros experimentos para llevar a cabo este tipo de escenarios en la televisión española. Más tarde, otros canales españoles se unen a este tipo de iniciativas como TVE, Canal 9 y TV3.

Actualmente, todas las cadenas tienen los medios para grabar producciones con escenarios virtuales.

Aunque la primera gran apuesta del uso de escenarios virtuales en la televisión fue en el festival de Eurovisión del año 1996. La idea inicial era utilizar escenarios virtuales durante todo el evento, incluyendo las actuaciones, pero finalmente se descartó porque esto obligaría a los artistas a adaptarse a este tipo de escenarios y podría afectar a la actuación.

Finalmente se decidió usar sólo en la parte de votaciones donde era posible realizar un entrenamiento de la presentadora para su uso. Como puede verse en la figura 1.1, se iban mostrando los resultados en tiempo real en un atril 3D. El resultado del evento fue un éxito a pesar de las dificultades técnicas que implicaba el uso del hardware de entonces, más concretamente de las tarjetas gráficas.

Algunos otros ejemplos fueron las elecciones autonómicas en el año 2009 retransmitidas en el Canal 9, donde se mostraban los resultados de las elecciones en tiempo real en un panel 3D como se puede ver en la figura 1.2.

Otro evento de gran relevancia fue en las elecciones europeas en el año 2009 en el canal BBC en el que los resultados de las votaciones se iban mostrando en un mapa en tres dimensiones y donde el presentador podía moverse sobre él como se muestra en la figura 1.3.

Figura 1.1: Captura del sistema de votaciones del festival de Eurovisión de 1996. Imagen tomada de Youtube

Figura 1.2: Captura del programa especial de elecciones autonómicas en Canal 9 (2009). Imagen tomada de www.revistalatinacs.org

1.2 Importancia de los MOOC

En la actualidad existe un auge en los cursos masivos online o MOOC, así como de los estudios a distancia o la creación de programas con formatos televisivos llevados a cabo por organizaciones como la propia Universidad de Castilla-La Mancha. En tan sólo 8 años de vida este tipo de formato se ha consolidado como uno de los más importantes en la enseñanza

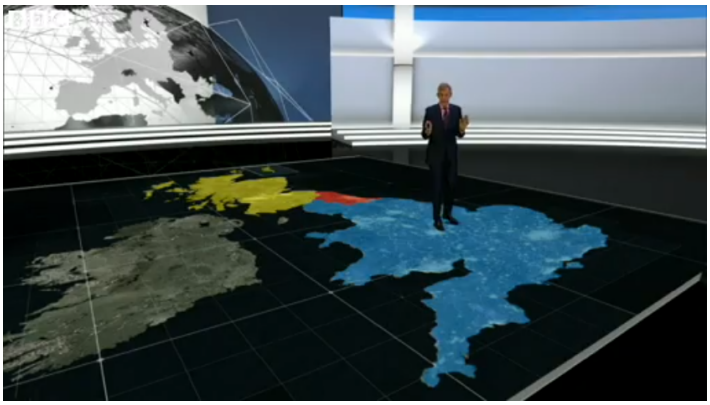

Figura 1.3: Captura del programa de elecciones europeas en BBC (2009). Imagen tomada de www.bbc.com

online pasando de impartirse en una sola universidad para 160.000 alumnos en el año 2011 a impartirse en 570 universidades y 12 instituciones a 35.000.000 de alumnos en 2015 según la página Online Course Report[1]. El crecimiento de los cursos MOOC también se ve reflejado en noticias como la compra de Lynda.com por parte de linkedin[2].

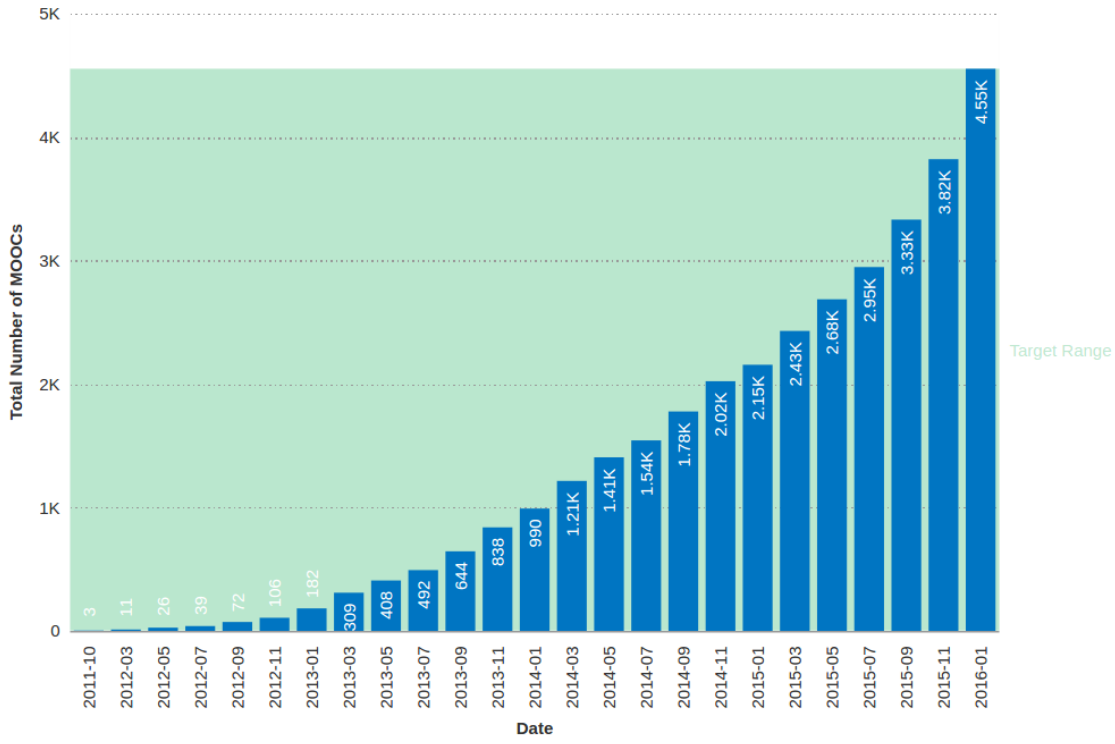

Figura 1.4: Crecimiento de cursos MOOC del año 2011 al 2016. Gráfico de http://www.onlinecoursereport.com

[1] http://www.onlinecoursereport.com

[2] http://www.businessinsider.com/linkedin-buys-lyndacom-for-15-billion-2015-4

En los portales de creación de contenidos para docencia online (como Udemy, Linda.com o la plataforma de MOOC española MiriadaX), exigen una cantidad mínima de vídeos explicativos. Por ejemplo, en el portal Udemy[3] exigen que al menos un 60 %[4] del contenido sea en formato de vídeo. Esto implica una necesidad de generación de contenido en este formato.

En concreto, el tipo de vídeo más utilizado en la docencia es el formato polimedia. Este formato consiste en un vídeo donde aparece un profesor a uno de los lados y en la parte posterior se van mostrando las transparencias con los contenidos sobre los que el profesor está hablando. Podemos ver un ejemplo de este tipo de formato en la figura 1.5.

Figura 1.5: Ejemplo de polimedia publicado en canal de youtube de la UCLM

1.3 Impacto socio-económico

El coste de una producción audiovisual de calidad conlleva una inversión en muchos casos elevada. Además de todos los medios materiales (incluyendo un escenario, cámaras móviles y personal cualificado para utilizar todos los elementos para la producción), existen varias etapas en la generación de contenido:

- Grabación del contenido, tanto audio como vídeo.
- Edición durante la grabación del contenido. Es decir, grabación de las diferentes tomas, ajustes de luz y de audio y demás tareas que facilitarán la tarea de post-producción.

[3] https://www.udemy.com
[4] https://blog.udemy.com/wp-content/uploads/2013/04/Getting-started-guidelines-2.pdf

Esta tarea, junto a la anterior, suele ocupar el doble del tiempo del vídeo final debido a errores cometidos y a tomas que no valen.

- Post-producción del contenido. Este paso incluye tareas como juntar las tomas que han sido buenas sin que se note que hay un corte o eliminación de ruidos del audio. Para esta tarea se requieren unas tres veces el tiempo del vídeo final.
- Publicación del vídeo.

Es decir, para un vídeo de una hora, se necesitarían unas 5 horas para tener un vídeo final. Esto sin contar el tiempo de procesamiento.

Queda clara una necesidad de mecanismos ágiles que requieran una menor cantidad de tiempo por persona para realizar producciones más rápidas de publicar pero manteniendo los indicadores de calidad.

1.4 Entorno

De esta necesidad surgen los sistemas de escenografía virtual[Fos14] que permiten realizar vídeos en entornos 3D[TAMH08][JFH13][DV14] sin necesidad de tener un escenario físico, ni múltiples cámaras móviles. Los componentes necesarios para realizar una producción en este tipo de sistemas son un ciclorama virtual con color chroma (normalmente azul o verde), una cámara y un micrófono, además del ordenador en el que se realizará la producción.

Las ventajas de este tipo de sistema pueden resumirse en cinco:

- Posibilidad de realizar la edición en tiempo real, reduciendo así todo este tiempo de la post-producción. Esto nos permite realizar la grabación y que el contenido esté disponible para su publicación al instante. Además, el profesor puede ver en tiempo real el resultado del vídeo, por lo que el feedback es inmediato.
- Este tipo de sistemas necesita un sólo operador. Además, la formación necesaria para el uso de este tipo de sistemas es mínima, pudiendo ser capaces de utilizar este tipo de sistemas realizando un curso de apenas unas horas.
- No es necesario el movimiento de la cámara física, el sistema proporcionará animaciones de cámaras que permitan el movimiento del punto de vista a través de diversas cámaras virtuales. Esto nos quita la necesidad de utilizar una gran cantidad de espacio, podremos realizar las grabaciones en un espacio muy reducido.
- Integración de elementos 2D y 3D. Esta integración nos posibilita añadir elementos virtuales al escenario. Por ejemplo, podemos añadir elementos 3D animados a la escena o incluso podemos ”introducir” al profesor en un espacio real sin que él esté ahí mandando una cámara y después introduciendo al profesor utilizando un chroma y técnicas de tracking de la cámara.

- En general, este tipo de escenarios reduce el coste por minuto de las producciones audiovisuales.

1.5 Problemática

A pesar de esto, existen algunas dificultades en este tipo de sistemas:

- Suelen contener una serie de escenarios cerrados y privativos en los que apenas se pueden realizar cambios de forma sencilla.
- Existe una dificultad para extender su comportamiento.
- No ofrecen mecanismos para la integración eficaz de software de terceros de rendering 3D.

Y aquí es donde surge Aix, cuyo objetivo principal es el diseño, desarrollo y despliegue de una plataforma basada en estándares y software libre para la producción 3D integral en sistemas de escenografía virtual, con un caso de implantación específico en el entorno EasySet[5] de Brainstorm.

1.6 Estructura del documento

A continuación se detallan los contenidos de los capítulos de este documento.

Capítulo 2 - Objetivos

En este capítulo se expone el objetivo principal del proyecto así como los objetivos funcionales.

Capítulo 3 - Antecedentes

En este capítulo se detallan los requisitos y herramientas que se han necesitado para el desarrollo del proyecto. También se especifican las soluciones que han sido descartadas y los motivos por los que se han descartado.

Capítulo 4 - Método

En este capítulo se indica la metodología de desarrollo utilizada para el proyecto, así como los patrones de diseño utilizados. También se explican las herramientas hardware y software utilizadas durante el desarrollo de este trabajo de fin de grado.

Capítulo 5 - Arquitectura

En este capítulo se explican los sistemas desarrollados desde un punto de vista arquitectónico y las ventajas de haber elegido este diseño.

Capítulo 6 - Resultados

En este capítulo se explican las iteraciones realizadas durante el desarrollo del proyecto y los hitos alcanzados en cada una de ellas.

[5] www.brainstorm.es/products/easyset-3d/

Capítulo 7 - Conclusiones

En este capítulo se indican qué objetivos de los planteados han sido cumplidos. También se presentan algunas líneas de trabajo futuro.

Capítulo 2

Objetivos

En este capítulo se establece tanto el objetivo principal como los objetivos específicos a desarrollar con la realización del proyecto.

2.1 Objetivo general

El objetivo principal de Aix consiste en el diseño, desarrollo y despliegue de una plataforma basada en estándares y software libre para la producción 3D integral en sistemas de escenografía virtual, con un caso de implantación específico en el entorno EasySet de Brainstorm, que es la solución implantada en el Centro de Tecnologías y Contenidos Digitales de la Universidad de Castilla-La Mancha.

2.2 Objetivos funcionales

En base al objetivo principal presentado en el apartado 2.1, se establecen una serie de subobjetivos que se comentan a continuación:

2.2.1 Integración con motores de visualización de Brainstorm.

Los ficheros generados en Aix deberán poder representarse sin necesidad de ningún tipo de edición adicional en motores de Brainstorm. En concreto, los archivos generados por Aix deberán ser representados en EasySet, el motor de escenografía virtual del Centro de Tecnologías y Contenidos Digitales de la UCLM.

2.2.2 Integración con herramientas de edición 3D.

Aix trabajará con al menos una herramienta de edición 3D libre multiplataforma. Se desarrollarán una serie de módulos que den soporte a las necesidades específicas de entornos 3D aplicados a la docencia. En concreto, Aix tendrá que dar soporte para:

- Exportación de geometría 3D. Se dará soporte a la exportación de modelos 3D poligonales.
- Precálculo de iluminación. Aix gestionará coordenadas de textura UV empleadas en la exportación de modelos con iluminación precalculada.
- Exportación de múltiples cámaras virtuales animadas. Aix permitirá la exportación de

animaciones de las cámaras dentro del propio Blender estableciendo la posición inicial y final e indicando el tiempo requerido para realizar esta animación

2.2.3 Soporte para la integración de herramientas adicionales de representación.

Conectado con el punto anterior de exportación de cámaras virtuales, Aix facilitará la integración de herramientas adicionales de representación 3D. En concreto se desarrollará un prototipo demostrador de esta funcionalidad con un sistema para la representación de gráficas 3D en tiempo real. Este prototipo servirá de base para cualquier desarrollo realizado con otros motores 3D y diferentes requisitos de despliegue gráfico.

2.2.4 Multiplataforma.

Aunque los archivos generados en Aix sólo podrán ser ejecutados en el motor de EasySet, las herramientas que forman el presente Trabajo Fin de Grado podrán ejecutarse en sistemas GNU/Linux o Windows. Esto es interesante con vistas a la integración de Aix en futuros desarrollos del Centro de Tecnologías y Contenidos Digitales de la UCLM.

2.3 Objetivos no funcionales:

Además, se especifica un único objetivo no funcional.

2.3.1 Uso de tecnologías y estándares libres.

El sistema será desarrollado utilizando herramientas libres que garanticen que el proyecto podrá utilizarse sin restricciones en el futuro.

Esto se hará en la medida de lo posible dado que el propio programa EasySet no es libre y por tanto no ha sido posible ver su código fuente para facilitar el desarrollo del proyecto. Las consecuencias de ello pueden verse en el capítulo 5.

Capítulo 3

Antecedentes

En este capítulo se hablará de la importancia de los vídeos en la enseñanza. A continuación se hablará de los sistemas de escenografía virtual. Por último se tratarán las soluciones utilizadas para la generación de contenidos en el Centro de Tecnologías y Contenidos Digitales, así como el material necesario y el proceso para generar este tipo de vídeos.

3.1 Importancia de vídeos en la enseñanza

Varios estudios [HJC05][DZ06] han demostrado que el uso de material en forma de vídeo (enseñanza pasiva) acompañado de otros tipos de actividad más activas consiguen unos resultados similares que las clases tradicionales con una sensación de esfuerzo menor por parte del alumno.

Esto hace que el alumno sea capaz de asimilar mayor cantidad de materia en menor tiempo dedicando periodos más largos.

Algunos de los motivos de que esto suceda son los siguientes:

- El uso de varios sentidos combinados, en este caso la vista y el oído, hace que el alumno pueda hacer una representación mental de la materia de forma más fácil que si sólo se percibiera la materia a través de un solo sentido, por ejemplo un texto escrito o apuntes en forma de audio.
- Algunos expertos opinan que los conocimientos son adquiridos de forma más efectiva si se obtienen en el contexto para el que se utilizarán. El formato de vídeo permite poner en contexto al alumno más fácilmente que lo podría hacer un texto escrito. Por ejemplo llevando la explicación al lugar donde se utilizarán estos conocimientos.
- Este formato permite un acceso a cualquier parte del contenido de forma rápida.

Es por esto por lo que la enseñanza en forma de vídeo haya tomado tanta importancia a día de hoy.

A pesar de todo y como se ha indicado al principio, es necesario complementar este tipo de material con otros formatos. Algunos estudios demuestran que el material en forma de vídeo suele fallar en la captación de la atención del alumno por tratarse de una forma de enseñanza pasiva y por tanto son necesarias actividades que involucren más al alumno.

3.2 Sistemas de escenografía virtual

En esta sección se hablará de los sistemas de escenografía virtual y las características que debe cumplir para ser consistente.

3.2.1 ¿Qué es la escenografía virtual?

La escenografía virtual consiste en una serie de elementos no reales generados por un ordenador mezclados con elementos reales captados a través de una cámara mostrando un resultado consistente.

Los requisitos que deben cumplir los sistemas de escenografía virtual para ello son:

- La imagen real se incorpora a la virtual a través de un chroma key, eliminando el color elegido a través de un ordenador.
- Las acciones del escenario virtual tienen que producirse en tiempo real.
- Para conseguir una consistencia entre los elementos, los objetos tienen que ser físicamente correctos.

Estas dos requisitos implican una serie de características con los que debe contar un sistema de escenografía virtual. Por un lado la necesidad de un chroma key hace que se necesiten los siguientes elementos:

- Ciclorama con el color elegido. Es decir, una pared pintada del color elegido como chroma key.
- Iluminación uniforme para el éxito del chroma key. Esto implica que el ciclorama tendrá un mismo color por toda su superficie, evitando contrastes sobre la misma. De no ser así, podrían aparecer ruidos en la imagen que eliminan la consistencia de la escena.

Por otro lado, la necesidad de realizar acciones en tiempo real requiere de los siguientes elementos:

- Elementos 3D de baja poligonización o de lo contrario el sistema podría sufrir de bajadas de Frames Per Second/Frames Por Segundo (FPS). A pesar de esto, tampoco se pueden utilizar demasiados pocos polígonos porque le restaría realismo a la escena y se notaría que se trata de un escenario generado con ordenador con más facilidad, hay que conseguir un equilibrio.
- Iluminación precalculada en los elementos del escenario. Si en vez de realizar los cálculos de iluminación en tiempo real se realizan en un procesado previo en un motor realista, se consiguen unos resultados más espectaculares con una mejora en el rendimiento, lo que se traduce en un aumento de FPS.

Por último, cuando hablamos de objetos físicamente correctos nos referimos a que todos los objetos creados a través del ordenador tienen que tener un tamaño acorde al que tienen en el mundo real. El ojo humano es capaz de identificar rápidamente si la escena consta de elementos de tamaño diferente al real, por tanto este aspecto debe ser cuidado.

3.3 Formatos de vídeo utilizados en el CTED de la UCLM

A continuación se explican los formatos utilizados en el CTED, el proceso mediante el que se generan y los puntos a favor y en contra de cada uno de ellos.

3.3.1 Polimedia

Uno de los primeros formatos de vídeos que fueron producidos en el CTED fueron los polimedia.

Como se ha explicado en el capítulo 1, estos vídeos se crean superponiendo el vídeo del profesor a las transparencias que aparecen en el fondo.

Los elementos necesarios para realizar este tipo de producciones son: una cámara para grabar el vídeo del profesor, un ciclorama de un color chroma elegido (en este caso verde), un micrófono para captar el audio, un ordenador en el que se van pasando las transparencias y una capturadora de vídeo en la que se graba el vídeo donde se graba la secuencia de transparencias, aunque esta última puede ser reemplazada por cualquier software de captura de pantalla como OBS [1].

El proceso de producción era en un principio relativamente sencillo, un técnico grababa el vídeo del profesor dando su clase y de las transparencias, además del audio, y posteriormente otra persona se encargaba de eliminar el color del chroma del vídeo del profesor, eliminar los ruidos del audio y juntarlo todo en un sólo vídeo. Lo cierto es que no se encontró ningún software que se adaptara a las necesidades del CTED por lo que fue necesario el desarrollo de una aplicación propia para el montaje de estos vídeos utilizando el sistema de nodos de Blender, que se explicarán en el capítulo 5.

Otro punto en contra era la no inmediatez de la generación del vídeo. No se trataba de un problema de tiempo, sino que si no se cuenta con una muestra de la salida del vídeo, un color de ropa o un peinado no adecuado para su uso con un ciclorama podía llevar a malos resultados y a ruidos en la imagen que no se podían evitar a esas alturas del procesamiento.

Además, a pesar de la gran calidad del contenido generado, el procesado de los vídeos era muy lento incluso en ordenadores de gran potencia, por lo que la generación de un sólo vídeo podía llevar horas o incluso días, lo que hizo que, en contraste con el proceso de grabación, la generación de estos vídeos se alargara de forma exponencial.

Podemos ver un ejemplo del formato polimedia en la figura 3.1

[1] https://obsproject.com/

Figura 3.1: Ejemplo de polimedia publicado en canal de youtube de la UCLM

3.3.2 Grabación de clases y conferencias

Otro de los formatos utilizados en el CTED fue la grabación de clases y conferencias. En ellos, utilizando un equipo MULTICAM [2] se realizaba una producción en tiempo real en la que se hacía un montaje del ponente o profesor con las transparencias.

Así pues, el material necesario para la generación de este formato de vídeo era un equipo MULTICAM con cámaras controladas de forma remota desde este sistema, unos micrófonos y todo el cableado necesario para conectar el equipo al sistema de la sala en la que se graba.

El proceso de generación de contenido es también sencillo, la interfaz de este tipo de sistemas permite que sea controlado por una sola persona utilizando un mando con un joystick para el movimiento de las cámaras y botones para los cambios de toma.

El mayor problema de este tipo de producciones es la necesidad de realizar el montaje del equipo MULTICAM cada vez que había que hacer una grabación en un sitio diferente.

Además, en este formato se pierde la integración que produce el uso de un chroma entre el profesor y las transparencias. También hay que tener en cuenta que se trata de contenidos que se graban de una clase o conferencia normal, por lo que las interrupciones por parte del público o los errores por parte del ponente son algo normal y no se puede repetir una parte porque se haya producido algo inesperado.

[2] http://www.multicam-systems.com/en/

Por tanto, este tipo de formato no puede sustituir a formatos generados específicamente para MOOC en general.

A pesar de esto, las producciones son de gran calidad y la entrega de los contenidos a la persona interesada es casi instantánea ya que, una vez grabado, no suele ser necesario realizar grandes cambios en el vídeo.

En la figura 3.2 podemos ver un ejemplo de este formato.

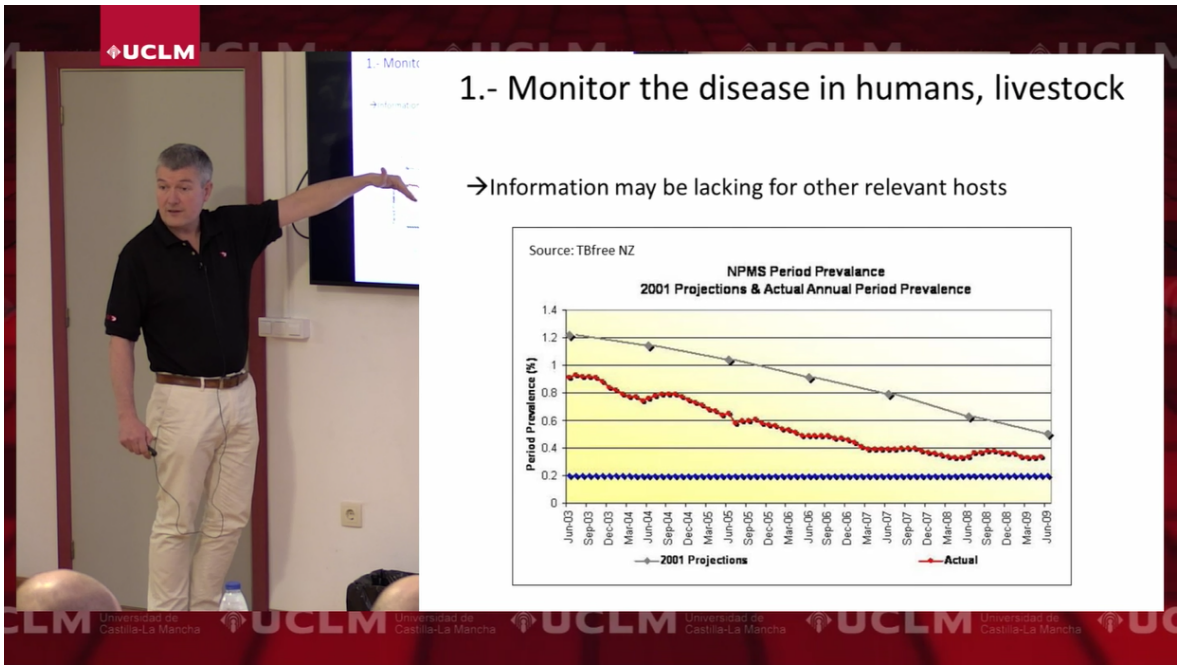

Figura 3.2: Ejemplo de grabación de clase

3.3.3 Vídeos en escenarios 3D utilizando Blender

Este tipo de formato es muy similar al conseguido utilizando EasySet 3D. En él, se muestra al profesor en un escenario 3D, con una pantalla al fondo donde se muestran las transparencias, donde se pueden integrar elementos 3D y es posible realizar animaciones de cámaras. La diferencia con los contenidos realizados en EasySet está en el proceso de producción.

Los materiales necesarios para estos vídeos son los mismos que para los producidos en un vídeo polimedia: una cámara, un micrófono, un ciclorama y un ordenador con capturadora para grabar las transparencias.

El proceso de producción consiste en realizar la grabación de audio y vídeo del profesor y de las transparencias igual que se hacía en el formato polimedia, pero tomando un plano más general del profesor.

Después, una a una, se realizan las animaciones dentro de Blender de las cámaras que se quieran mostrar, por ejemplo al inicio del vídeo se puede realizar un ”vuelo” sobre el

escenario, más tarde se puede hacer un vídeo de medio cuerpo, etc.

Ahora, en el sistema de nodos se seleccionan los vídeos grabados y los frames a procesar en cada una de las animaciones. Una vez seleccionado, se procesan uno a uno cada una de las secciones del vídeo y posteriormente se mezclan con algún programa de edición (por ejemplo Kdenlive[3]).

La calidad de este tipo de vídeos es muy buena, además las herramientas que proporciona Blender hacen que se puedan realizar efectos muy espectaculares, como sombras, reflejos o integración de elementos 3D. El problema es que el tiempo necesario para realizar un vídeo de corta duración es muy grande y además, la persona encargada ha de estar presente y estar atenta durante el mismo, a diferencia del resto de producciones donde se podían procesar los vídeos mientras el encargado podía realizar otras tareas. Para hacernos una idea, para la generación de un vídeo de apenas 2 minutos fueron necesarias unas 4 horas de trabajo continuo.

En resumen, este formato ofrece una calidad excelente, pero sólo es realizable en casos muy excepcionales por la gran cantidad de tiempo que se necesita para su elaboración y no sirve para producciones que se quieran realizar con cierta frecuencia.

En la figura 3.3 podemos ver una catpura de un vídeo generado con esta técnica.

Figura 3.3: Ejemplo de vídeo generado con Blender publicado en canal de youtube de la UCLM

En el capítulo 5 se puede ver una explicación de cómo realizar la composición en Blender

[3] https://kdenlive.org/

para grabar en este tipo de formato.

3.3.4 Solución final: EasySet

En este formato se muestra al profesor totalmente integrado en un escenario 3D, con una pantalla donde mostrar transparencias, que permite animaciones de cámaras e integración de elementos 3D.

El material necesario para este tipo de vídeos es una cámara mediante la cual se captará la imagen del profesor, un micrófono para el audio, un ciclorama, un ordenador con el programa EasySet instalado y un ordenador que capture la salida proporcionada por EasySet.

El proceso es el siguiente: la cámara captura el vídeo del profesor, pero no se graba sino que se manda directamente al ordenador en el que está instalado EasySet. Lo mismo ocurre con las transparencias que se van mostrando en un ordenador que controla el profesor, pero en esta ocasión no es necesario realizar la grabación de las mismas sino que se envía la salida del ordenador a EasySet. Es entonces en el ordenador que tiene instalado EasySet donde se produce la composición directamente sin intermediarios y se manda a otro ordenador que capturará esta salida.

Esto implica que el profesor puede ver en tiempo real tanto las transparencias sobre las que está hablando como lo que se está grabando en formato final, es decir, él/ella en el escenario 3D con las transparencias de fondo. O lo que es lo mismo, tenemos en el momento en el que se graba la salida final de la producción de este tipo de formato sin necesidad de procesamientos previos.

En la figura 3.4 puede verse una captura de un vídeo generado mediante EasySet.

Figura 3.4: Ejemplo de vídeo generado con EasySet

Capítulo 4

Método de trabajo

Este capítulo explica la metodología de desarrollo y las herramientas utilizadas para el desarrollo de Aix.

4.1 Metodología de trabajo

Este proyecto ha sido desarrollado siguiendo un proceso de desarrollo iterativo incremental, basado en hitos y reuniones semanales con el fin de hacer frente a los posibles cambios de requisitos que pudieran surgir a lo largo del desarrollo del proyecto. En concreto se ha utilizado el Proceso Unificado de Desarrollo. Así pues, se ha caracterizado por los siguientes puntos:

- Dirigido por casos de uso. Estudiando las funcionalidades requeridas por el Centro de Tecnologías y Contenidos Digitales y la Universidad de Castilla-La Mancha.
- Centrado en la arquitectura. Se ha dedicado gran cantidad de tiempo a la generación de una esqueleto que permitiera un desarrollo iterativo e incremental y fomentara la escalabilidad para posibles ampliaciones futuras.
- Desarrollado de forma iterativa e incremental.

Se ha seleccionado esta metodología debido a la organización de los proyectos del programa EasySet. En estos, cada característica de los escenarios está contenido en un archivo diferente incluyendo materiales, texturas, cámaras y animaciones entre otros. De este modo el proyecto pudo ir desarrollándose siguiendo los hitos establecidos e indicados en el capítulo 6.

La arquitectura elegida ha sido pues una basada en componentes. A través de ésta fue posible desarrollar y probar las diferentes funcionalidades del proyecto de forma individual sin afectar al resto de las clases.

4.2 Patrones de diseño

El principal patrón de diseño utilizado durante el desarrollo de la aplicación principal es el Patrón Component, descrito en el libro Game Programming Patterns [Nys14].

Este patrón no forma parte del los patrones GOF, pero es considerado de gran utilidad y se

utiliza en proyectos tan importantes como el extendido motor de desarrollo de videojuegos Unity3D[1].

Este patrón consiste en que cualquier entidad en un proyecto es un simple contenedor. Es decir, las entidades no tienen funcionalidad propia, sino que son los componentes que contiene los que le dotan de comportamiento.

En nuestro caso, la clase Exporter sería nuestra entidad y son los componentes (los cuales heredan de la clase PyFile) que están contenidos en la lista "components" los que le dotan de comportamiento procesando cada uno de los archivos de los proyectos EasySet generados. Algunos de estos componentes son Mat, Tex, Geo, etc.

Esto permite que si el día de mañana queremos añadir nuevos archivos al proyecto o queremos cambiar la funcionalidad de alguno de ellos no tenemos que modificar los existentes porque no existe acoplamiento entre ellos, sino que añadiremos los componentes que creamos conveniente o modificaremos sólo los que haga falta.

4.3 Herramientas

En esta sección se indican los lenguajes y las herramientas (tanto hardware como software) que se han utilizado para el desarrollo del proyecto.

4.3.1 Lenguajes

El lenguaje principal utilizado para el proyecto fue Python (versión 2.7)[2][Ram15][Lut13] debido a que los plugins para blender son desarrollados en este lenguaje.

También ha sido utilizado para la aplicación de representación de gráficas en 3D. Ésta ha sido desarrollada utilizando el motor gráfico Blender Game Engine en el cual se pueden identificar comportamientos de los diferentes objetos a través de scripts escritos en este lenguaje.

La generación de la documentación ha sido desarrollada utilizando Latex. Éste lenguaje permite la generación de documentos de alta calidad mediante tags.

4.3.2 Herramientas hardware

El desarrollo del proyecto ha sido realizado principalmente en dos ordenadores:

- Portátil del alumno: en él se realizará la mayor parte de desarrollo del sistema. Sus especificaciones técnicas son:
  - Procesador: Intel(R) Core(TM) i5-3230M @ 2.60 GHz
  - Memoria RAM: 8 GB
  - Disco duro: SSD 256 GB

[1] https://unity3d.com/
[2] www.python.org

- Tarjeta gráfica integrada: Intel HD 4000
- Sistema Operativo: Linux Mint 17.3 / Windows 10

- HP workstation Z440: Ordenador del Centro de Tecnologías y Contenidos Digitales: en él se ejecuta el sistema y se realizan las pruebas necesarias. Sus especificaciones técnicas son:
  - Procesador: Intel(R) Xeon(R) E5-1650 v3 @ 3.50 GHz
  - Memoria RAM: 16 GB
  - Disco duro: HDD 1 TB
  - Tarjeta gráfica: Nvidia Quadro K5000
  - Sistema Operativo: Windows 10.
- Además también se ha utilizado un ordenador en el que se realizan las grabaciones capturando la salida del ordenador que ejecuta EasySet:
  - Procesador: Intel(R) Xeon(R) E5-1410 v2 @ 2.80 GHz
  - Memoria RAM: 8GB
  - Disco duro: HDD 4TB.
  - Tarjeta gráfica: Nvidia NVS 300.
  - Sistema Operativo: Windows 8.1.

El sistema de captura de audio es un Shure BLX88 unido a una mesa de mezclas Presonus StudioLive 16.0.2.

También se cuenta con cámaras Canon XF205 para la captura del vídeo de los ponentes.

4.3.3 Herramientas software

Los medios software que se utilizaron para el desarrollo del sistema fueron:

- Windows 7 y 10[3] y Linux Mint 17.3[4] (GNU/Linux): El proyecto Aix ha sido portado a ambos sistemas, pero las pruebas sólo han podido ser realizadas en Windows 7 (siendo el único sistema operativo soportado por EasySet 3D).
- Vim (versión 7.4)[5]: editor utilizado como IDE para el desarrollo del sistema por ser ligero, libre y versátil.
- Tmux (versión 1.8)[6]: multiplexor de terminales. Junto a Vim permite un workflow fluído gracias a sus atajos de teclado posibilitando trabajar en la aplicación sin levantar las manos del teclado.

[3] https://www.microsoft.com/en-us/windows/
[4] www.linuxmint.com
[5] www.vim.org
[6] https://tmux.github.io/

- EasySet 3D: software creado por Brainstorm[7]. Es un sistema de escenografía virtual, utilizado incluso por cadenas de televisión para realizar sus escenografías. Sobre él se exportarán los escenarios y se realizarán las pruebas correspondientes.
- Redmine: sistema utilizado para la gestión del proyecto. Este sistema proporciona un entorno para la gestión de proyectos normalmente en equipos, aunque en este caso será utilizado por un solo usuario. En ella se indicaron las tareas a realizar y el tiempo dedicado a cada una de ellas.
- Blender (versión 2.77)[ble][Cam]: programa sobre el que se desarrollará el exportador de escenarios, elegido por ser libre y por su soporte al desarrollo de plugins. En él se han realizado los escenarios que posteriormente se exportarían mediante este proyecto. Además, la aplicación de generación de diagramas de barras en 3D también ha sido desarrollada en Blender, más concretamente utilizando el sistema de nodos del Blender Game Engine.
- Git (versión 2.7.2)[8]: software de control de versiones. Utilizado por ser libre y por su integración con la gestión de tareas del propio Redmine. Ha sido utilizado directamente en la línea de órdenes sin utilizar interfaz gráfica.
- Imagemagick (versión 6.9.3-6)[9]: software de procesado de imágenes. Este programa se utiliza para conversión de los previews de las cámaras a formato XPM entendibles por el programa EasySet. Concretamente el procesamiento se lleva a cabo justo después de renderizar la escena desde cada una de las cámaras.

[7] www.brainstorm.es
[8] www.git-scm.com
[9] www.imagemagick.org

Capítulo 5

Arquitectura

En este capítulo se hablará sobre la arquitectura del proyecto Aix. Por un lado se hablará del exportador de escenarios para Blender. Por otro se tratará el soporte para la integración de herramientas adicionales de representación, más concretamente en el ejemplo que se ha desarrollado para representar diagramas en tres dimensiones.

- Exportador de escenarios desde Blender. Es la aplicación más importante del proyecto. Consiste en un plugin para el software de modelado 3D Blender. Éste permitirá la exportación de un proyecto comprensible para el programa EasySet partiendo de cualquier modelo en 3D en formato Blender.
- Visualizador de diagramas de barras en tres dimensiones. Permite la visualización de diagramas de barras en 3D sobre un fondo con chroma. Esto permite la representación de datos interesantes para el ponente de forma atractiva para el alumno o usuario consumidor del curso que se está grabando.

Además también se hará una descripción del proceso de creación de los vídeos generados mediante Blender de los que se habló en el capítulo 3.

5.1 Descripción General

La estructura del proyecto Aix ha sido diseñada en forma de arquitectura por componentes.

Al comienzo del proyecto el único material con el que se contaba fue un proyecto para el programa EasySet con sus diversos archivos de configuración que lo componen e indican las características visuales y de comportamiento dentro de la aplicación.

Debe tenerse en cuenta que la parte más técnica del proyecto EasySet no se encuentra documentada, sólo se contaban con manuales de usuarios y videotutoriales para el funcionamiento de la interfaz de usuario, y por tanto ha sido necesario un gran esfuerzo de tiempo para descifrar la funcionalidad de cada uno de los archivos del proyecto mediante ingeniería inversa del tipo black box (o caja negra).

Para ello, se iban realizando pruebas con cada uno de los archivos hasta que se descurbría la funcionalidad de cada parte.

Una vez descubiertos los objetivos que cumple cada uno de los archivos, se desarrolló un componente distinto para cada tipo de archivo del proyecto EasySet.

De este modo se permitió realizar el desarrollo de los archivos de forma incremental. En la figura 5.1 se muestran los diferentes archivos que incluye un proyecto básico de EasySet.

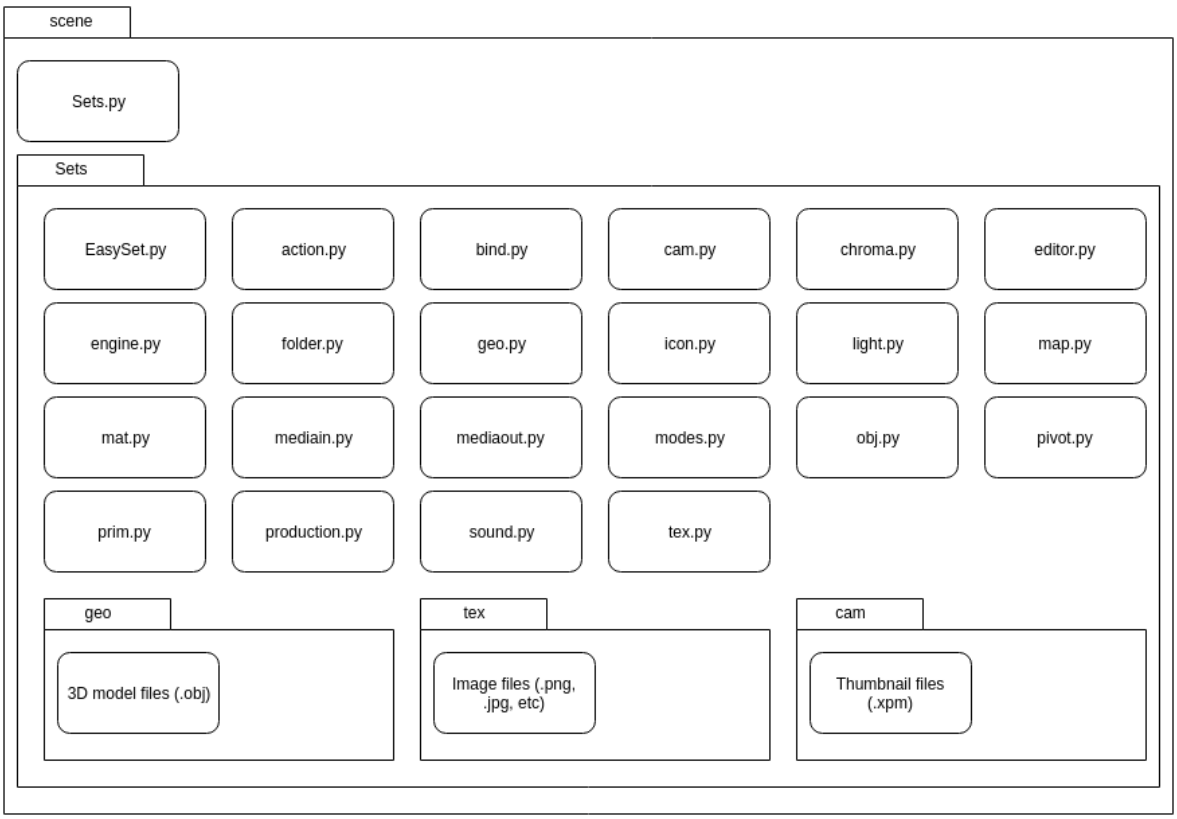

Figura 5.1: Archivos contenidos en un proyecto EasySet.

5.2 Aix

Después del estudio previo de las características con las que debía contar un escenario para su uso dentro de EasySet, se identificaron tres elementos principales: modelos 3D, cámaras y luces. A continuación se listan las características de estos:

- Modelos 3D: los escenarios exportados deben tener al menos un modelo 3D que mostrar en pantalla, este modelo además debe tener una serie de características propias de los modelos 3D, materiales y texturas:
    - Materiales: los modelos exportados deben tener uno o varios materiales asignados a grupos de vértices del modelo. Estos materiales deben tener a su vez varias características que lo definen:
        - Nombre: identificador único que representa al material.

    - Color definido mediante el formato RGBA (red, green, blue, alpha). Es de especial importancia el último parámetro ya que nos servirá para representar el reflejo del suelo.
    - Sombreado especular: este valor indica la cantidad de brillo que se produce al interactuar el objeto con una luz en la escena.
    - Sombreado difuso: indican la rugosidad del material. A mayor valor, más pulida parece la superficie del objeto.
    - Sin sombra: esta propiedad permite que un objeto no se vea alterado por las luces que se encuentran en la escena. Esta propiedad es especialmente importante para el proyecto ya que la mayor parte de objetos exportados desde blender tendrán un preprocesado de luz de modo que la textura guarda los valores de interacción del objeto con la luz y por tanto se eliminará mucho tiempo de cómputo para representar estos cálculos.
    - UV mapping: imagen utilizada sobre ese material. Además también será necesario guardar las coordenadas UV para aplicar esa imagen sobre el material.
  - Texturas: los modelos pueden tener una o varias texturas asignadas a los materiales aplicados. Estas texturas deben tener las siguientes propiedades:
    - Nombre: identificador único que representa la textura.
    - Ruta hacia la imagen: directorio donde se encuentra la imagen de esta textura.
    - Tamaño de la imagen en pixels.
- Cámaras: es una de las partes más importantes de los proyectos EasySet. Será necesario definir la posición, dirección, campo de visión (FOV) y el tiempo de animación de cada una de las cámaras que queramos definir dentro de la escena.
- Luces: deben definirse las luces aplicadas a la escena.

5.2.1 Definición de las propiedades de objetos del escenario

A continuación se desarrollará la lista anterior para explicar los conceptos relativos a los gráficos por computador y su aplicación en los proyectos EasySet.

Color de los materiales

Esta propiedad no tiene mayor complicación, se trata de la representación del color de un material en formato RGBA.

El único problema que hubo es que Blender representa este formato con valores entre 0 y 1 al igual que EasySet, pero no ocurre lo mismo con los archivos de tipo MTL, de donde se sacan los valores, que guardan valores de 0 a 0.8 para los colores y de 0 a 1 para el alpha. Por esta razón hubo que recalcular estos valores previamente antes de guardarlo en el archivo de

materiales.

Así pues, un material de color blanco totalmente opaco en Blender y EasySet se representa mediante los valores (1,1,1,1) mientras que en en un archivo MTL se hace con los valores (0.8,0.8,0.8,1.0).

Sombreado difuso

Este parámetro indica la cantidad de luz que se refleja cuando ésta interactúa con el objeto. A mayor valor, mayor cantidad de luz reflejará el objeto y con valores cercanos a cero, apenas se reflejará luz.

Podemos ver un ejemplo de este comportamiento dentro de Blender en la figura 5.2.

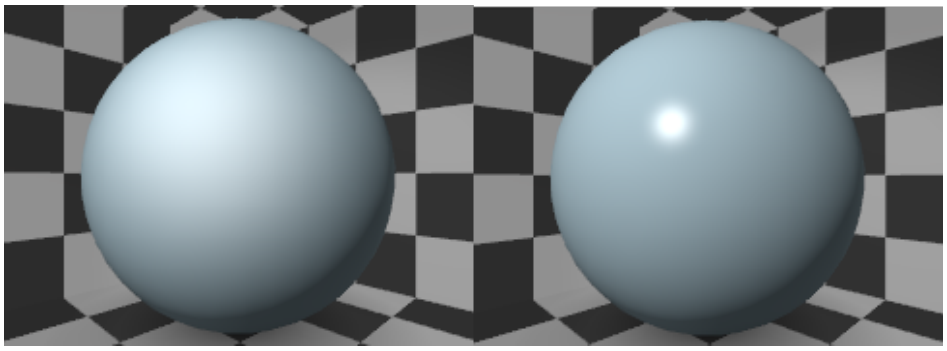

Figura 5.2: Materiales de diferente sombreado difuso

Como podemos ver, la esfera de la izquierda refleja la luz en múltiples direcciones y parece que la luz se refleja toda la superficie

Por otro lado, en la esfera de la derecha, la reflexión de la luz se concentra en una parte muy pequeña de la superficie debido a la poca rugosidad del material.

Podemos ver cómo se simula el sombreado difuso en gráficos por computador en la figura 5.3.

Es decir, la cantidad de luz representada en un punto depende únicamente del ángulo que forman la dirección de incidencia de los rayos de luz (l), con la normal de la superficie (n) para la que se quiere calcular.

En este caso no hizo falta realizar ningún tipo de cálculo a la hora de exportar estos valores a EasySet ya que tanto Blender, como EasySet como MTL representan este parámetro con valores de 0 a 1.

Sombreado especular

Este parámetro indica la rugosidad de un material con respecto a su comportamiento frente a la luz.

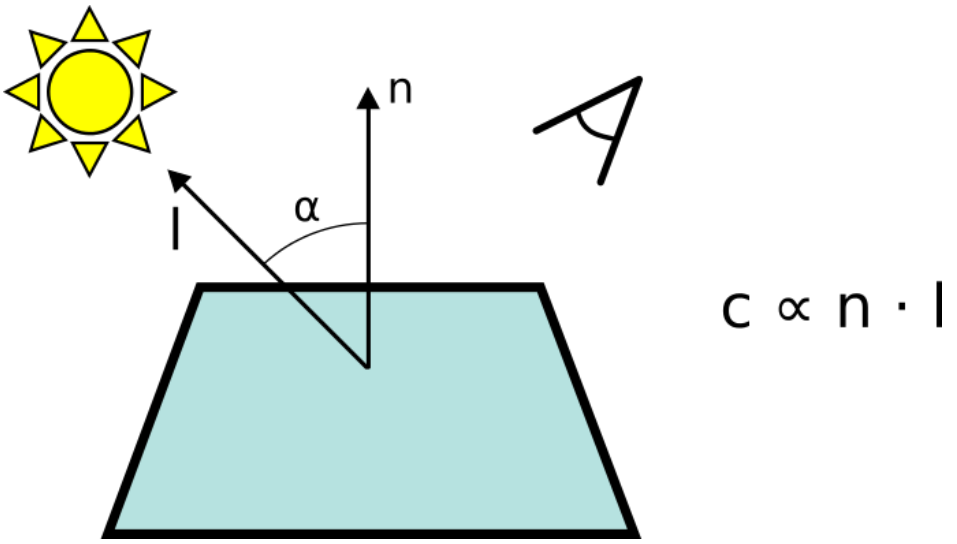

Figura 5.3: Simulación del sombreado difuso

En la figura 5.4 se muestra el comportamiento de la luz ante un material muy rugoso con uno con menor rugosidad.

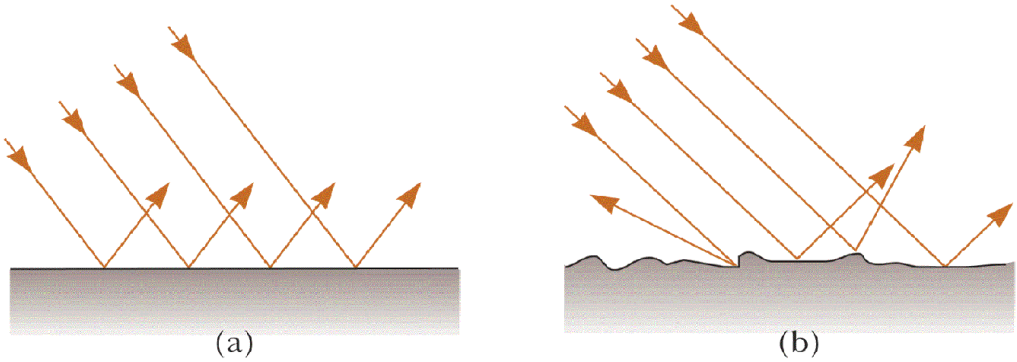

Figura 5.4: Comportamiento de la luz ante un material poco rugoso y muy rugoso. Imagen obtenida de http://neilatkin.com/

Como puede verse, un material liso (a) refleja los rayos de luz que vienen de forma paralela en la misma dirección debido a la forma de la superficie.

Por otro lado, un material rugoso (b) al tener una superficie uniforme hace que los rayos de luz se reflejen en direcciones diferentes aunque provengan que la misma dirección.

En la figura 5.5 podemos ver un ejemplo de distintos valores de este parámetro para un mismo objeto.

La imagen más a la izquierda muestra una esfera con un valor de sombreado especular

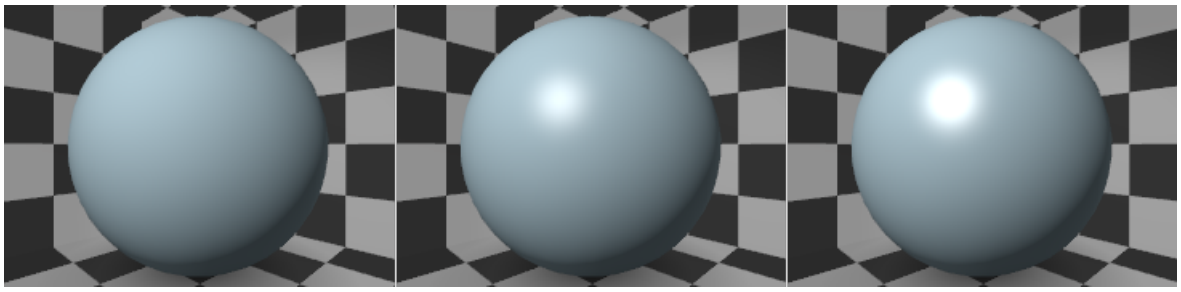

Figura 5.5: Diferentes valores de sombreado especular en Blender

igual a cero. Como podemos ver, no refleja luz.

La imagen del centro tiene un valor de sombreado especular de 0.5 y se puede ver cómo refleja algo de luz.

Por último la imagen de la derecha tiene un sombreado especular de valor 1, por lo que la luz reflejada es mayor.

Podemos ver cómo se simula el sombreado especular en gráficos por computador en la figura 5.6.

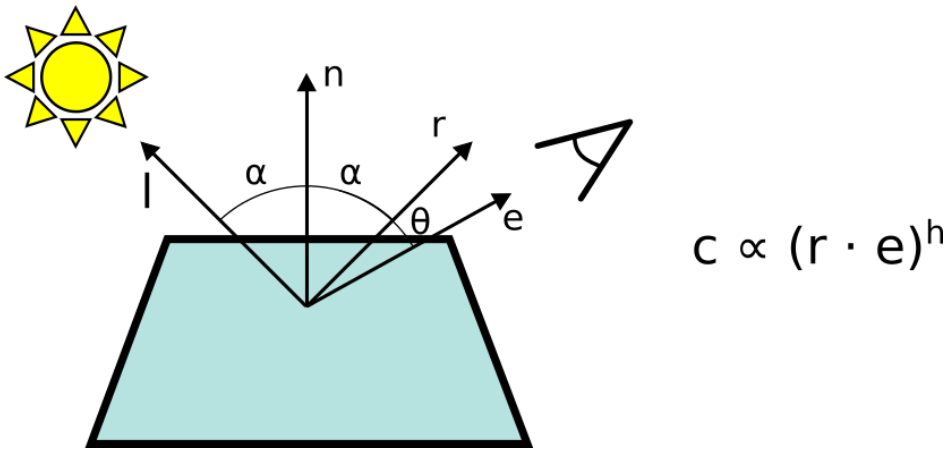

Figura 5.6: Simulación del sombreado especular

En este caso, la cantidad de luz reflejada depende de cuatro parámetros. Por un lado el ángulo de incidencia de la luz (l), por otro la normal de la superficie sobre la que se refleja (n), y por último, el punto de vista del observador (e).

En este caso, como en el anterior, no hubo que realizar ningún cálculo antes de guardar el valor de este parámetro ya que tanto Blender, como EasySet y MTL utilizan valores de 0 a 1.

Sin sombra

El parámetro ”sin sombra” (más comunmente llamado ”shadeless”) nos indica si un objeto reaccionará ante las luces de la escena.

Esto quiere decir que un objeto en el que este parámetro esté activado será dibujado de la misma manera en pantalla tanto con poca como con mucha luz, o incluso sin luz ninguna.

Este parámetro es particularmente importante durante el desarrollo del proyecto, ya que, gracias al precálculo de la iluminación que nos permite realizar los cálculos relacionados con la iluminación antes, podemos evitar todos los cálculos durante la representación de la escena dentro de EasySet.

En la figura 5.7 podemos ver una representación de una esfera con el parámetro shadeless activado. Al no tener aplicada ninguna textura, el objeto se muestra como una figura plana.

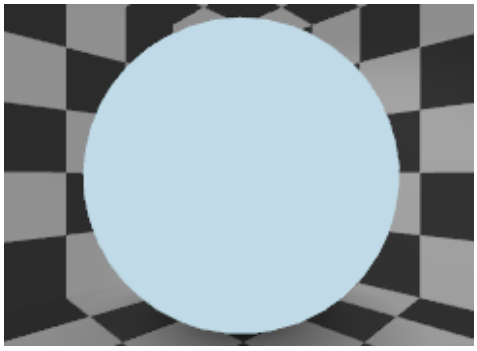

Figura 5.7: Objeto con el parámetro shadeless activado

Dentro de EasySet se indica si un material es de tipo shadeless o no es mediante el parámetro ”MAT_SHADE_MODEL” del archivo ”mat.py”, si lo es se indicará mediante el valor ”on”, de lo contrario se usará el valor ”off”.

UV mapping

El mapeado UV o UV mapping es la técnica mediante la cual podemos dibujar una imagen sobre el modelo 3D asignando a cada una de las caras de ese modelo una sección de una imagen.

El proceso de asignación de las secciones a cada una de las caras es el siguiente: Por cada vértice del modelo 3D se le adjudica una coordenada UV a cada una de las caras adyacentes a él con un valor entre 0 a 1. Este valor corresponde a una coordenada en un mapa que se utilizará posteriormente con la imagen seleccionada.

El valor de la coordenadas se representa de 0 a 1 porque si se diera la coordenada utilizando los pixels de la imagen que se va a utilizar sólo serviría para esa sola imagen, en cambio si

se hace de 0 a 1 podemos cambiar esa misma imagen a una de mayor o menor resolución o incluso de diferente proporción sin tener que alterar este mapeado UV original.

En la figura 5.8 podemos ver un ejemplo de este tipo de mapeado y cómo las caras del cubo toman la sección de la textura que se le ha asignado.

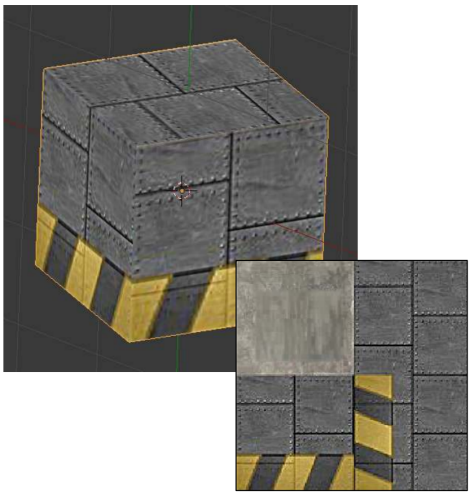

Figura 5.8: Ejemplo de mapeado UV

En el listado 5.1 podemos ver un ejemplo de mapeado UV. En él se muestra un trozo del mapeado UV de un simple cubo.

```
vt 0.0000 0.6667
vt 0.3333 0.6667
vt 0.3333 1.0000
vt 0.0000 1.0000
...
```

Listado 5.1: Mapeado UV de un cubo

Para el primer vértice y la primera cara se le da un valor de coordenadas (0.0, 0.6667) y lo mismo ocurre por cada uno de los vértices y caras del cubo.

Este parámetro fue fácil de incluir en EasySet, ya que los archivos de tipo OBJ ya lo incluyen en su información, por lo que sólo se tuvo que importar esos datos.

Campo de visión

El campo de visión (FOV) de una cámara es la parte del mundo que es visible desde esa cámara desde una posición y dirección.

Más concretamente, en Blender, el campo de visión define el ángulo que una cámara puede ver delante de ella.

En la figura 5.9 puede verse una representación gráfica de lo que es el campo de visión.

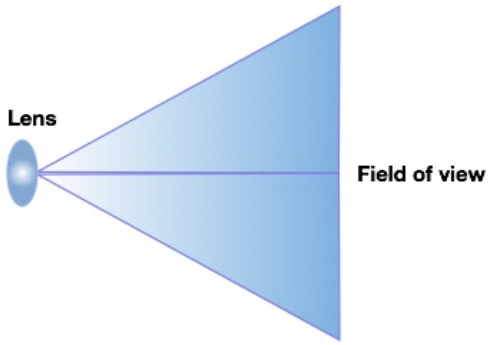

Figura 5.9: Representación gráfica del FOV. Obtenida de https://kintronics.com

Podemos ver un ejemplo de lo que ocurre cuando cambiamos el FOV dentro de blender en la figura 5.10.

Figura 5.10: Diferentes FOV dentro de Blender

La imagen de la izquierda muestra un campo de visión de 90º mucho mayor que la de la derecha, que tiene 15º, es por ello por lo que la cámara ha tenido que alejarse de la escena para captar el mismo trozo de escenario que la de la derecha.

Por el contrario, la imagen con mayor campo de visión muestra cierta distorsión en la escena, lo cual implica que tenemos que encontrar un equilibrio entre mostrar todo lo que queremos en la escena sin distorsionar la imagen final.

Este parámetro se incluyó en el archivo ”cam.py”. Concretamente la variable ”CAM_FOVH” es la que indica del campo de visión de cada una de las cámaras. Para añadirlo, se recorrieron mediante un bucle cada una de las cámaras y se obtuvo su valor dentro de Blender. Posteriormente se escribió en la plantilla correspondiente (cam.py) este valor.

Animación de las cámaras

En la animación tradicional, éstas se definen mediante una serie de frames clave o keyframes a través de los cuales se definen las poses por las que los vértices de un modelo en 3D

tenía que pasar antes de terminar la animación.

En este caso, EasySet permite animaciones de cámara sencillas en las que se toma como punto de partida la posición en ese momento de la misma y como posición final la posición elegida como objetivo.

Estas simples animaciones proporcionan una sensación de movimiento y de realismo que producen una sensación de inmersión mayor.

La forma de incluir este tipo de animaciones ha sido a través del archivo ”EasySet.py”. Cuando se ejecuta el procesado del escenario se recorren una a unas las cámaras y se comprueba si se existe un valor de tiempo de animación, y si no, se le da un valor de 1 por defecto, por último se toma la plantilla de EasySet y se sustituye el valor donde procede.

5.2.2 Archivos de proyecto EasySet

Una vez sabemos las características con las que debe contar nuestro escenario, se pudo proceder a investigar los archivos que nos harán falta.

La conclusión fue que los archivos para los que había que crear plantillas eran 6: cam.py, EasySet.py, geo.py, light.py, mat.py y tex.py, el resto de archivos podían ser reutilizados de un proyecto a otro sin realizar modificaciones.

A continuación se explica la funcionalidad cada uno de ellos:

- cam.py: Este archivo contiene la información relativa a las cámaras. En él se especifican la posición, orientación y campo de visión (FOV).
- EasySet.py: Es el archivo principal del proyecto. Aquí se indican diversas propiedades del proyecto, pero las que han sido necesarias para el desarrollo del proyecto han sido todas las relacionadas con las cámaras, incluyendo una preview de cada una de ellas en formato XPM o el tiempo de animación de las mismas.
- geo.py: Contiene la información con los archivos de modelos 3D generados por blender y referencia la ruta que los contiene.
- light.py: Este archivo incluye la información de cada una de las luces incluídas en la escena.
- mat.py: Aquí se especifican los materiales utilizados en la escena. Las características que incluye son el nombre, el color, la intensidad del color especular y de reflexión, si es transparente, si es shadeless, y la imagen que utiliza en el mapeado UV si es que lo tiene.
- tex.py: Este archivo indica las imágenes que se utilizan en el mapeado UV de la escena. Las características que se especifican son el nombre, el path de la imagen y el tamaño de la imagen en pixels.

5.2.3 Diseño elegido

Como se ha mencionado previamente, el diseño elegido para este proyecto fue uno basado en el patrón component. En la figura 5.11 se muestra el diagrama de clases que sigue este patrón.

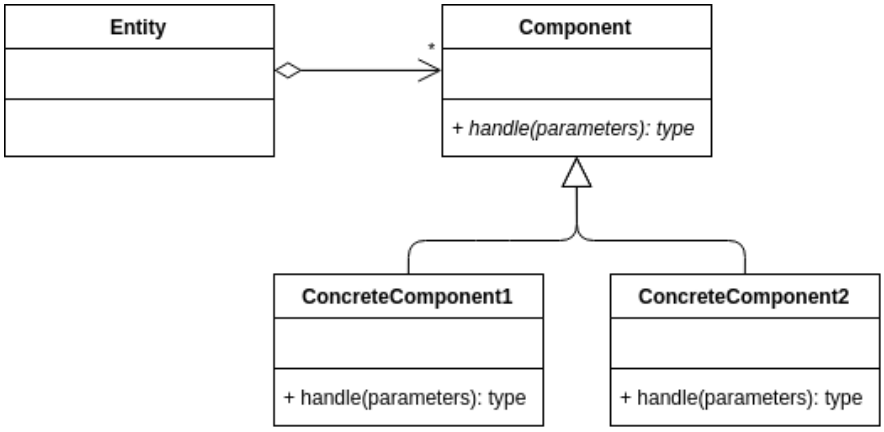

Figura 5.11: Diagrama del patrón componente

Una vez identificadas las propiedades que se tenían que añadir y los archivos que había que rellenar, se procedió a crear los diferentes componentes que harían falta.

Como se ha explicado en el capítulo 4, el funcionamiento de este patrón es bastante sencillo. Por cada funcionalidad diferente que queramos que tenga una entidad, creamos un componente que heredará de una clase que tendrá una función que tendrán que implementar cada uno de los componentes. Cuando queramos procesar los archivos, recorremos todos los componentes y llamamos al método de procesamiento de cada uno de ellos.

A continuación se incluye el código de la clase PyFile de la que heredarán el resto de componentes en el listado 5.2.

Como podemos ver, la clase consta de la función abstracta ”process” que heredarán el resto de componentes, y de tres atributos; file_name, _template y _destiny.

El atributo file_name indica el nombre del archivo que se utilizará en el archivo de destino.

El atributo _template indica la plantilla que utilizará esa clase, es decir, el archivo que copiará o que rellenará con los datos obtenidos en el procesamiento.

El atributo _destiny indica la ruta completa del archivo de salida.

Llegados a este punto se dieron lugar dos tipos de funciones de procesamiento de archivos. Por un lado, teníamos las funciones de copia. Este es el caso en el que los archivos sólo tienen que ser copiados a la carpeta del proyecto de EasySet debido a que siempre tienen el mismo contenido para todos los proyectos EasySet. En el listado 5.3 podemos ver cómo se realiza

```
import os, abc

class PyFile(object):
    __metaclass__ = abc.ABCMeta

    def __init__(self, path, file_name):
        self._file_name = file_name
        self._template = os.path.join(os.path.abspath(os.path.dirname(__file__))
        , 'templates', file_name)
        self._destiny = os.path.join(path, file_name)

    @abc.abstractmethod
    def process(self):
        """ Abstract funcion"""
        return

    @property
    def template(self):
        return self._template

    @template.setter
    def template(self, val):
        self._template = val

    @property
    def destiny(self):
        return self._destiny

    @destiny.setter
    def destiny(self, val):
        self._destiny = val
```

Listado 5.2: Código de la clase PyFile

el procesamiento en la clase Action. Como se puede ver, cogemos la plantilla y la copiamos en el destino especificado en el constructor.

```
    def process(self):
        shutil.copy(self.template, self.destiny)
```

Listado 5.3: Código de la clase Action de la función de copia

Por otro lado, tenemos las funciones de sustitución. En ellas, se toman las plantillas, se realizan los cálculos necesarios y se obtienen los parámetros que hagan falta de la escena de Blender y rellenamos la plantilla con los diferentes valores. Para conseguir este fin, se ha utilizado la función ”substitute” contenida en la librería ”string” de python. Gracias a ella, se puede sustituir un fragmento colocando una palabra clave precedida del símbolo $ con el valor que hiciera falta. En el listado 5.4 podemos ver un ejemplo de este tipo de función, en concreto el de la clase ”Mat” que se encargará de la exportación de los datos relacionados con los materiales.

Como se ha indicado, la clase ”Exporter” será la que contenga esta serie de componentes y de llamarlos cuando sea necesario realizar el procesamiento. La función ”create_files” se llama a procesar todos los componentes como se muestra en el listado 5.5. En ella se inicializan los diferentes componentes y se ejecutan llamando a la función ”process” de cada uno de ellos.

El diagrama de clases de la aplicación principal queda como se muestra en la figura 5.12.

5.3 Ejemplo de soporte para la integración de herramientas adicionales de representación: representación de diagramas en 3D

Por otro lado, la aplicación de representación de diagramas de barras en 3D es más sencilla. Se trata de una aplicación creada con Blender Game Engine que hace aparecer y desaparecer un diagrama de barras sobre un chroma verde teniendo como entrada un archivo de configuración.

Esta aplicación funciona de la siguiente forma:

Al iniciarse, el programa lee el archivo de configuración donde se indican varios parámetros necesarios para la ejecución del proyecto:

- En la primera línea se indican los siguientes parámetros:
  - Horizontal/Vertical: nos indica si las barras del diagrama se muestran de forma horizontal o vertical.
  - Tamaño total del diagrama en el eje X.
  - Tamaño total del diagrama en el eje Y.
- En la segunda línea se indica la posición de la cámara dentro de la escena con el fin de hacerla coincidir con la de la cámara de EasySet.
- A partir de la tercera línea se indican las características de las barras que se van a mostrar, una por línea, éstas incluyen:
  - Imagen que se va a utilizar como título de la barra. Nos permite indicar qué representa el tamaño de esta barra.
  - Valor: indica el tamaño de la barra. No hay restricción con respecto a este valor, ya que los tamaños de todas las barras son relativos al mayor valor de entre todas.
  - Color (opcional): la aplicación asigna por defecto diferentes colores a todas las barras, pero es posible indicar el color de cada una de ellas en formato RGBA.

En el listado 5.6 podemos ver un ejemplo de archivo de configuración para esta aplicación.

El programa se encargará de leer una a una las líneas, dividirlas utilizando el carácter ":" y de asignar las propiedades a la cámara o a las barras del diagrama a través de propiedades internas de BGE del objeto. Por ejemplo, las barras guardan su tamaño máximo en una variable llamada "total_height" y será la altura máxima que ésta alcanzará cuando se actualice la escena, mientras que la posición de la cámara se cambia sin modificar ninguna variable interna de la misma.

La decisión de usar un formato propio en vez de usar algún estándar, como pueden ser XML o JSON, no fue tomada a la ligera. Es cierto que existen librerías para python que permiten leer y escribir los valores en estos formatos de forma muy simple, pero el número de parámetros que se querían guardar en estos archivos era muy pequeño y se pensó que era aún más sencillo realizar una lectura a mano de unas cuantas líneas separándolas utilizando algún carácter.

Una vez cargados los parámetros, la aplicación se inicia mostrando únicamente un fondo verde que, combinado con el chroma de dentro de EasySet, muestra el escenario como si no hubiera nada en esa posición.

Cada frame, la aplicación se actualiza esperando eventos de usuario. En este caso la tecla J hará que se muestren los títulos y las barras (con un tamaño cercano a cero) y poco a poco irán creciendo hasta alcanzar el tamaño final. Una vez las barras han alcanzado su tamaño máximo, la aplicación espera de nuevo a la interacción de usuario, cuando pulse la tecla K, las barras harán la animación contraria y los títulos desaparecerán.

La forma de captura de eventos ha sido a través del sistema de nodos que proporciona Blender Game Engine, lo cual nos evita utilizar alguna librería externa como "curses". Este sistema permite realizar gran parte de la lógica de la aplicación de forma visual uniendo nodos. Podemos ver cómo quedó la lógica de la aplicación en la figura 5.13. En este caso, la lógica del sistema se centralizó en la cámara de la escena, ya que es el único elemento que siempre está presente cada vez que se ejecuta la aplicación. Puede observarse que, además de la captura del teclado, también se actualiza la escena cada frame utilizando este sistema de nodos.

En la figura 5.14 se muestra la lógica de la aplicación de diagrama de barras en forma de diagrama de estados.

5.4 Vídeos utilizando Blender

En esta última sección del capítulo se explicará cómo realizar los vídeos en un escenario 3D utilizando Blender como alternativa al uso del programa EasySet.

Los pasos a seguir para realizar este tipo de formato son los siguientes:

- Añadimos al profesor sobre el escenario.

- Añadimos el reflejo del profesor sobre la superficie que queramos.
- Añadimos las transparencias en la superficie que queramos (una pantalla).
- Añadimos el reflejo de las transparencias sobre la superficie del suelo.
- Añadimos render layers.
- Creamos los nodos necesarios para realizar la composición.

A continuación se entra en detalle en cada uno de ellos.

5.4.1 Profesor

Una vez tenemos el escenario tenemos que añadir el vídeo del profesor con el chroma verde. Para ello creamos un plano sobre el que irá el vídeo, por tanto tendrá que tener un tamaño un poco más grande del que ocuparía el profesor sobre el escenario (dado que el profesor podría mover los brazos y salirse de él). Además es importante aplicar rotación y escalado para que no haya errores en el mapeado UV, para ello, con el plano seleccionado, pulsamos "control + A" en el modo objeto y seleccionamos la opción "Rotation & Scale".

A continuación moveremos este plano a otra capa diferente pulsando la tecla M con el plano seleccionado. También será necesario hacer el unwrap sobre el que dibujaremos el vídeo del profesor. Para ello primero tenemos que añadir el vídeo del chroma seleccionando el "UV/Map Editor" y haciendo click en "Open image", ahora navegamos hasta seleccionar el vídeo. Después seleccionamos el plano, pasamos al modo edición pulsando el tabulador y seleccionamos todos los vértices con la tecla A y después la tecla U y seleccionamos la opción Unwrap. Ahora tendremos que ajustar el mapa UV sobre el profesor como se muestra en la figura 5.15.

5.4.2 Reflejo

Si queremos que haya reflejo del profesor sobre alguna superficie tenemos que seguir los siguientes pasos.

Lo primero, cambiamos el centro del plano para que esté en el centro de la arista inferior, esto simplificará los pasos siguientes. Ahora duplicamos el plano sobre el que se proyectará el profesor y lo escalaremos con un valor -1 sobre el eje Z. Como podemos ver, ahora aparece justo debajo. Además, hay que mover este nuevo plano a un layer diferente a los anteriores pulsando la tecla M y seleccionando otra.

Ahora tendremos que preparar la zona sobre la que se reflejará el profesor, para ello creamos un plano sobre la superficie en la que se hará el efecto espejo, en nuestro caso la tarima y hacemos unwrap igual que antes, seleccionamos el plano, vamos al modo edición con el tabulador, seleccionamos todos los vértices, pulsamos la tecla U y seleccionamos "Unwrap". En este caso no tenemos que modificar el mapeado UV ya que "sólo" nos servirá para decir

sobre qué zona queremos realizar el reflejo. También deberemos mover el objeto a una capa diferente. La escena queda como se muestra en la figura 5.16.

Ya tenemos todo listo para proyectar el profesor y el reflejo sobre el escenario, ahora procedemos con la parte de las transparencias.

5.4.3 Pantalla con las transparencias/vídeo

Añadimos un plano en la superficie sobre la que queramos mostrar las transparencias y aplicamos rotación y escalado, hacemos unwrap y movemos a otra capa como en el anterior.

5.4.4 Reflejo de la pantalla

Ahora duplicamos este plano, aplicamos un escalado de -1 sobre el eje Z y movemos a una capa diferente. Ahora tendremos que mover el plano en el eje Z para que quede donde estaría el reflejo de la pantalla en la superficie del suelo.

Por último, es necesario aplicar un material invisible a todos los planos sobre los que sólo se reproducirán vídeos y los que se realizará el reflejo, para ello creamos un material, vamos a la parte de ”Transparency”, marcamos ”Raytrace” y ponemos el valor ”Alpha” a 0. La escena queda como se muestra en la figura 5.17.

5.4.5 Render layers

Ahora tenemos que crear un Render layer (o capa de renderizado) por cada plano que hayamos añadido a la escena.

En el render layer original podemos dejar los parámetros por defecto.

Ahora, por cada plano creado (incluído el del reflejo), creamos un render layer y en la parte de ”Layer” y ”Mask Layer” tendremos que seleccionar la capa en la que aparece el plano que corresponda a este render layer. Además, en la sección ”Passes”, hay que marcar la opción ”UV”, el resto se pueden dejar por defecto.

Una vez todos los render layers están listos, procedemos a la composición de nodos.

5.4.6 Composición de nodos

La idea es sencilla: por cada render layer que contenga un plano se dibuja el vídeo correspondiente sobre el mapa UV de ese render layer y lo mezclamos con la escena original. En la figura 5.18 se muestra cómo se realiza este proceso. El resto de pasos consistirá en añadir nuevos nodos intermedios.

La cosa se complica cuando queremos realizar algún tipo de procesado anterior a esto, por ejemplo, en el vídeo del profesor es necesario quitar el chroma verde. Para ello añadimos un nodo de tipo ”Keying” entre el nodo ”Image” y el ”Map UV” y ajustamos los parámetros necesarios.

O por ejemplo para el caso del reflejo del profesor tenemos que realizar la mezcla sólo en la zona del suelo sobre la que queremos hacer el efecto espejo. Para ello en el mezclado (nodo Mix) tenemos que añadir el factor del mapa UV del suelo. Además sería conveniente realizar un Blur para que parezca que la superficie no está totalmente pulida.

En la figura 5.19 se muestra la composición final con todos los nodos necesarios para renderizar la escena final.

Definición de nodos y sus parámetros.

Parámetros comunes:

- Image: Imagen que se toma como entrada-salida de los nodos.

Image: Nodo de entrada para archivos de imagen o vídeo.

- File: Archivo con la imagen o vídeo.
- Source: Tipo de imagen. Puede ser una generada, vídeo, una secuencia de imágenes o una sola imagen. En este caso es un vídeo.
- Frames: Número de frames que se usarán para el renderizado, se han seleccionado el total de frames porque el vídeo ya estaba preparado para ser usado al completo.
- Start Frame: Frame inicial del vídeo.
- Offset: Frame desde el que nos interesa renderizar.
- Cyclic: Indica si el vídeo se volverá a reproducir cuando acabe.
- *Auto-Refresh: Actualiza la imagen si se producen cambios.

Render Layers: Nodo de entrada, permite usar los elementos de una capa determinada.

- UV: A través de este parámetro indicaremos dónde queremos renderizar cierta imagen a través del mapa UV que haya en ese render layer. Por ejemplo, para dibujar el profesor añadimos un plano al cual le hacemos unwrap para después dibujar sobre él.
- Scene: Escena que queremos renderizar.
- Layer: Aquí se especifica la capa que queremos seleccionar.

Keying: permite eliminar un determinado color de una imagen.

- Pre Blur: Indica el valor de blur que se realizará sobre el color elegido antes de realizar los cálculos. Se utiliza en imágenes con granularidad alta en la parte del chroma.
- Despill Factor/Despill Balance: Indica el borde que se eliminará de la imagen final.
- Edge Kernel Radius: Indica el radio del círculo el cual indica si un píxel forma parte del borde.

- Edge Kernel Tolerance: Indica el nivel de tolerancia para saber si un píxel forma parte del borde.
- Clip Black/Clip White: Indica el nivel de contraste a la hora de decir si un píxel forma parte del chroma o de la parte que queremos usar.
- Dilate/Erode: Aumenta o disminuye la zona que no forma parte del chroma.
- Feather Fallof/Feather Distance: Igual que dilate/erode, pero permite más precisión a través de diferentes algoritmos.
- Post Blur: Blur realizado después de eliminar el chroma para evitar picos en los bordes. Key Color: Color utilizado en el chroma.

Transform: permite realizar una serie de transformaciones básicas.

- Method to use to filter transform: Puede tener tres valores, Nearest, Bilinear y Bicubic.
- X: Traslación en el eje X que queremos realizar.
- Y: Traslación en el eje Y que queremos realizar.
- Angle: Rotación que queremos aplicar.
- Scale: Escalado que queremos aplicar.

Crop: Nodo de transformación permite recortar una imagen.

- Crop Image Size: Cuando se habilita, la imagen se recorta a la región especificada, si no, la imagen mantiene su tamaño original
- Relative: Cuando se habilita, los valores aparecen en forma de porcentaje, si no, aparecen por píxeles.
- Left/Right/Up/Down: Parte de la imagen que se va a recortar a los lados de la imagen.

Map UV: Nodo de transformación que permite componer una imagen sobre un mapa UV.

- Alpha: Valor que indica la transparencia que se le da a la imagen renderizada.
- UV: Parámetro que indica sobre qué mapa UV se va a renderizar la imagen.

Blur: Nodo de transformación que permite dar un efecto blur sobre una imagen.

- Filter Type: Algoritmo que se va a utilizar para realizar el blur. Puede ser Gaussian, Fast Gaussian, Flat, Tent, Quadratic, Cubic, Catrom, Mitch.
- Variable Size: Permite hacer blur per-píxel.
- Bokeh: Habilita blur circular.
- Gamma: Aplicar blur con valores gamma corregidos.

- Relative: Cuando se habilita, los valores aparecen en forma de porcentaje.
- X: Valor del blur en el eje X.
- Y: Valor del blur en el eje Y.
- Extend Bounds: Extiende la imagen de entrada para rellenar la imagen con el blur aplicado.

```
def process(self):
    string = '# -*- coding: utf-8 -*-\nitemnew("mat", "<>EasySet")\n\n'
    original = ''
    # Read the template and save it in the temporal variable "templ"
    templ = None
    with open(self.template, 'r') as filetemp:
        templ = Template(filetemp.read())
    with open(os.path.join(os.path.dirname(self.destiny),
                           'geo', 'scene.mtl'), 'r') as file:
        original = file.read()
        array_original = original.split('newmtl ')
        for i in range(1, len(array_original)):
            has_texture = False
            #line by line, look for some special words
            name = x = y = z = alpha = ref = shin = color = traenable = shadeless = ''
            lines = array_original[i].split('\n')
            name = lines[0]
            for j in range(1, len(lines)):
                # check the first word of the line
                line_array = lines[j].split(' ')
                first_word = line_array[0]
                if first_word == 'Ns':
                    shin = line_array[1]
                elif first_word == 'Ks':
                    ref = line_array[1]
                # if the alpha is lower than one, change the value of the alpha and add a new
                    line
                elif first_word == 'd':
                    alpha = line_array[1]
                    if float(alpha) < 1:
                        traenable = '\n        "MAT_TRAENABLE", True,'
                elif first_word == 'Kd':
                    colors = [float(line_array[1]), float(line_array[2]), float(line_array[3])]
                    i = 0
                    while i < 3:
                        colors[i] /= 0.8
                        i += 1
                    color = str(colors[0]) + ', ' + str(colors[1]) + ', ' + str(colors[2])
                elif first_word == 'map_Kd':
                    has_texture = True
                    tex = line_array[1]
                    for x in range(2, len(line_array)):
                        tex += ' ' + line_array[x]
                    tex = tex.split('\\')[-1]
                    tex = tex.split('/')[-1]
                elif first_word == 'illum':
                    if line_array[1] == '0':
                        shadeless = '"off"'
                    else:
                        shadeless = '"on"'
            if has_texture == False:
                tex = ''
            if alpha == '':
                alpha = 1
            string += templ.substitute(name=name, color=color, alpha=alpha, ref=ref, shin=shin,
                traenable=traenable, shadeless=shadeless, tex=tex)
            string += '\n\n'
    with open(self.destiny, 'w') as wfile:
        wfile.write(string)
```

Listado 5.4: Código de la clase Mat de la función de sustitución

```
  def create_files (filepath):
    path = filepath
    scene = os.path.join(path, 'scene')
    pyClases.SceneMain(scene).process()
    scene = os.path.join(scene, 'scene', 'Sets');
    pyClases.Scene(scene).process()
    scene = os.path.join(scene, 'Sets');
    classes = [pyClases.Action(scene), pyClases.Bind(scene), pyClases.Cam(scene), pyClases.Chroma(
         scene), pyClases.EasySet(scene), pyClases.Editor(scene), pyClases.Engine(scene), pyClases.
         Folder(scene), pyClases.Geo(scene), pyClases.Icon(scene), pyClases.Light(scene), pyClases.
         Map(scene), pyClases.Mat(scene), pyClases.Mediain(scene), pyClases.Mediaout(scene), pyClases
         .Modes(scene), pyClases.Obj(scene), pyClases.Pivot(scene), pyClases.Prim(scene), pyClases.
         Production(scene), pyClases.Sound(scene), pyClases.Tex(scene)]
    for pyClass in classes:
        pyClass.process()
```

Listado 5.5: Código de llamada al procesamiento de los componentes del sistema

```
vertical:10:5
10:-40:4
logoA.png:202:0.053:0.059:0.8:1
logoB.png:143:0.8:0.085:0.068:1
logoC.png:189:0.061:0.668:0.8:1
```

Listado 5.6: Ejemplo de archivo de configuración de aplicación de diagramas 3D

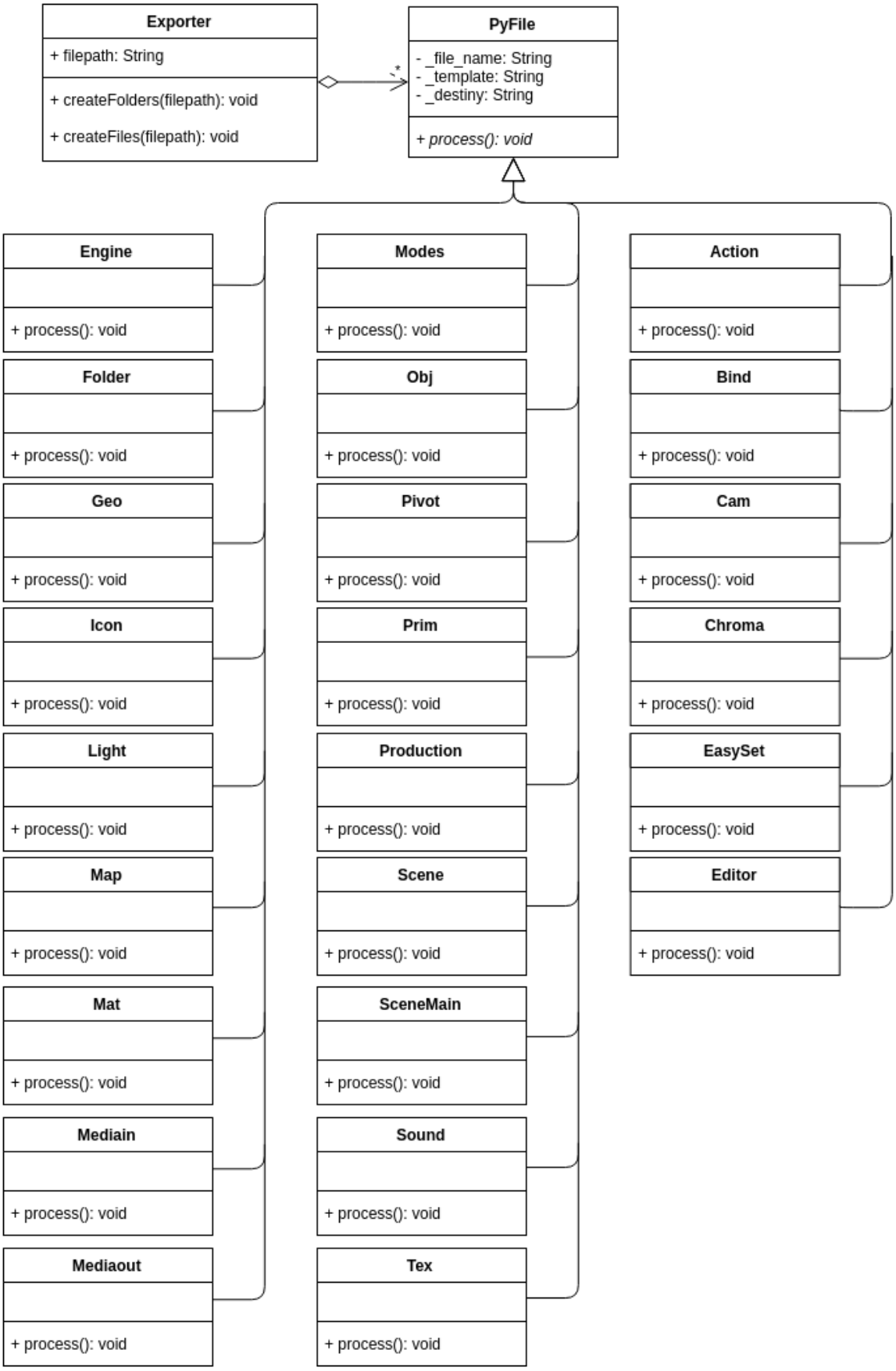

Figura 5.12: Diagrama de clases de la aplicación principal

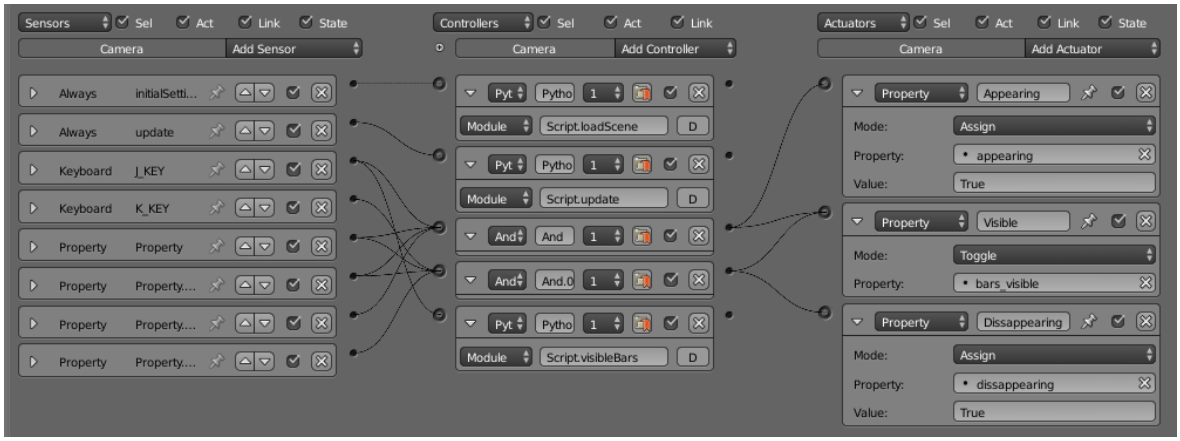

Figura 5.13: Vista general del sistema de nodos dentro de BGE

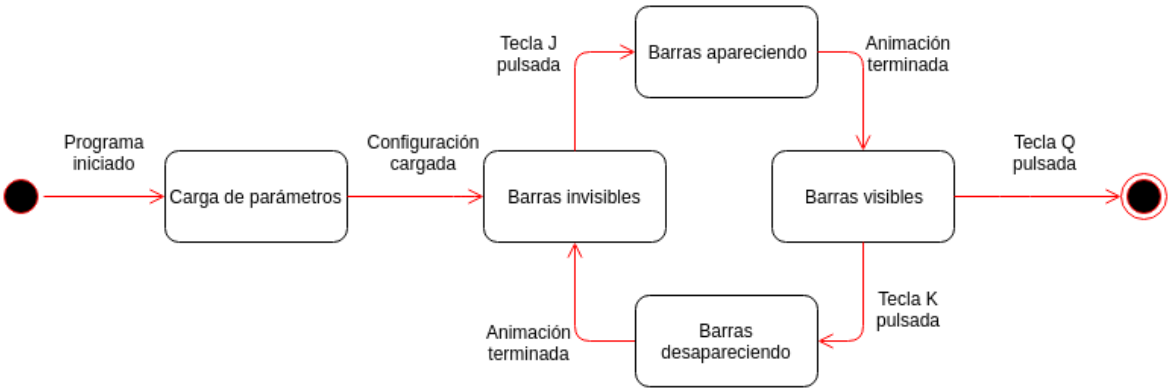

Figura 5.14: Diagrama de estados de la aplicación de diagrama de barras

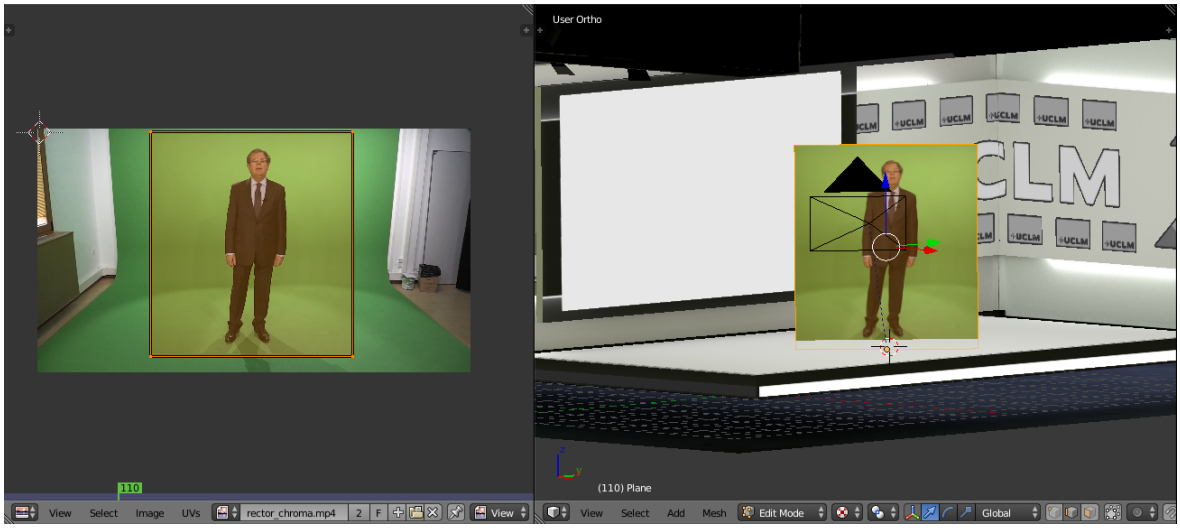

Figura 5.15: Imagen del plano del profesor dentro de Blender

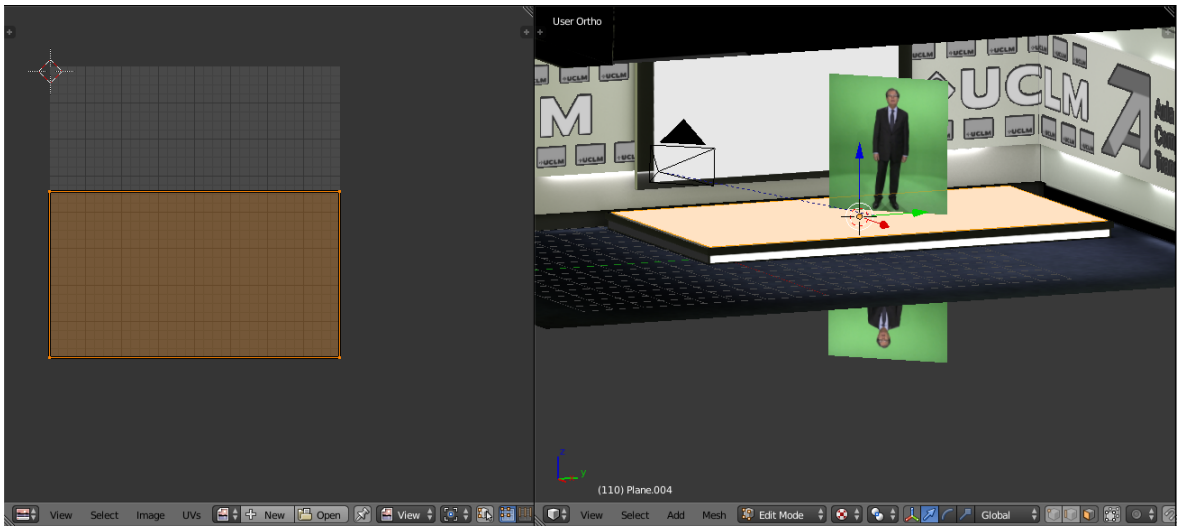

Figura 5.16: Imagen del plano del reflejo del profesor dentro de Blender

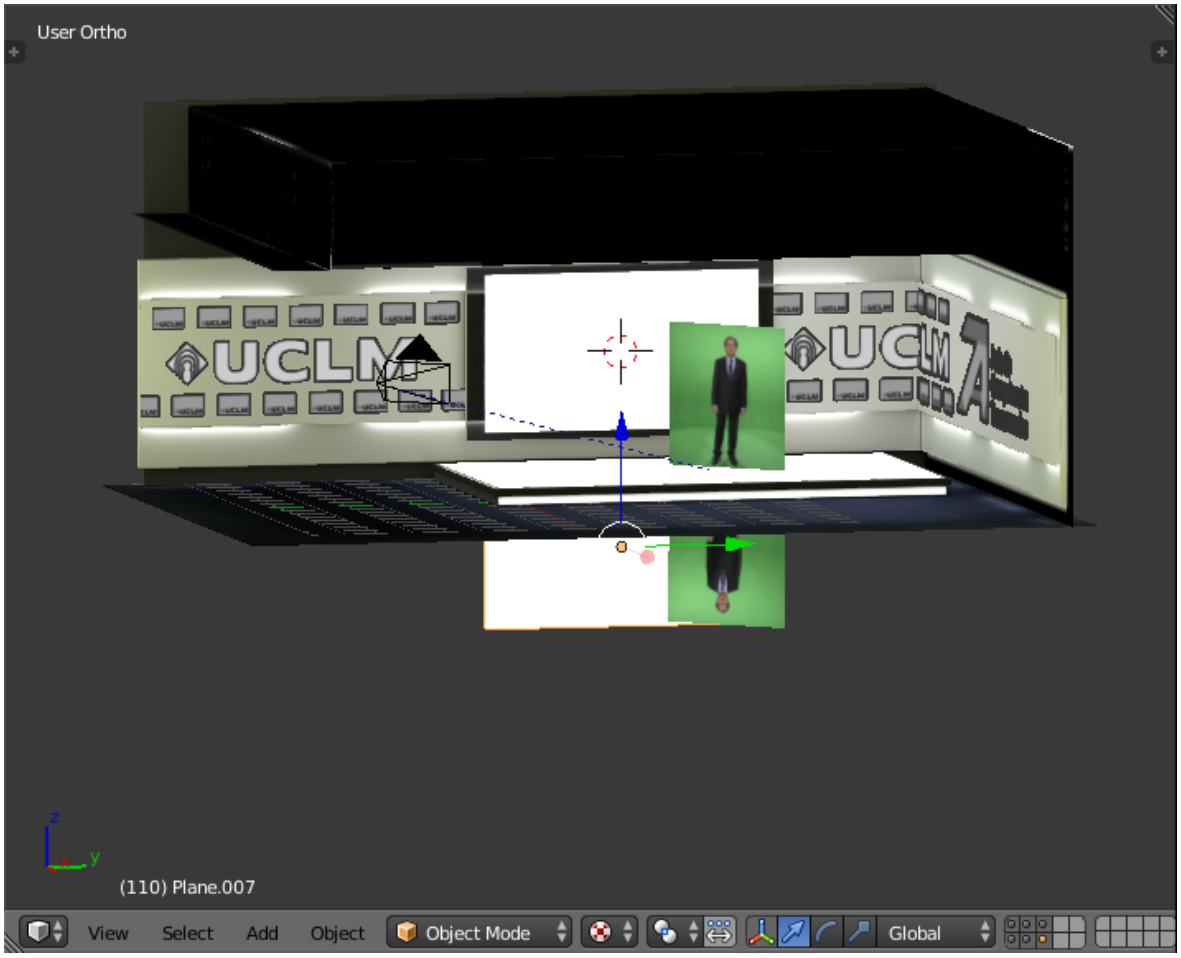

Figura 5.17: Imagen del plano del reflejo de las transparencias dentro de Blender

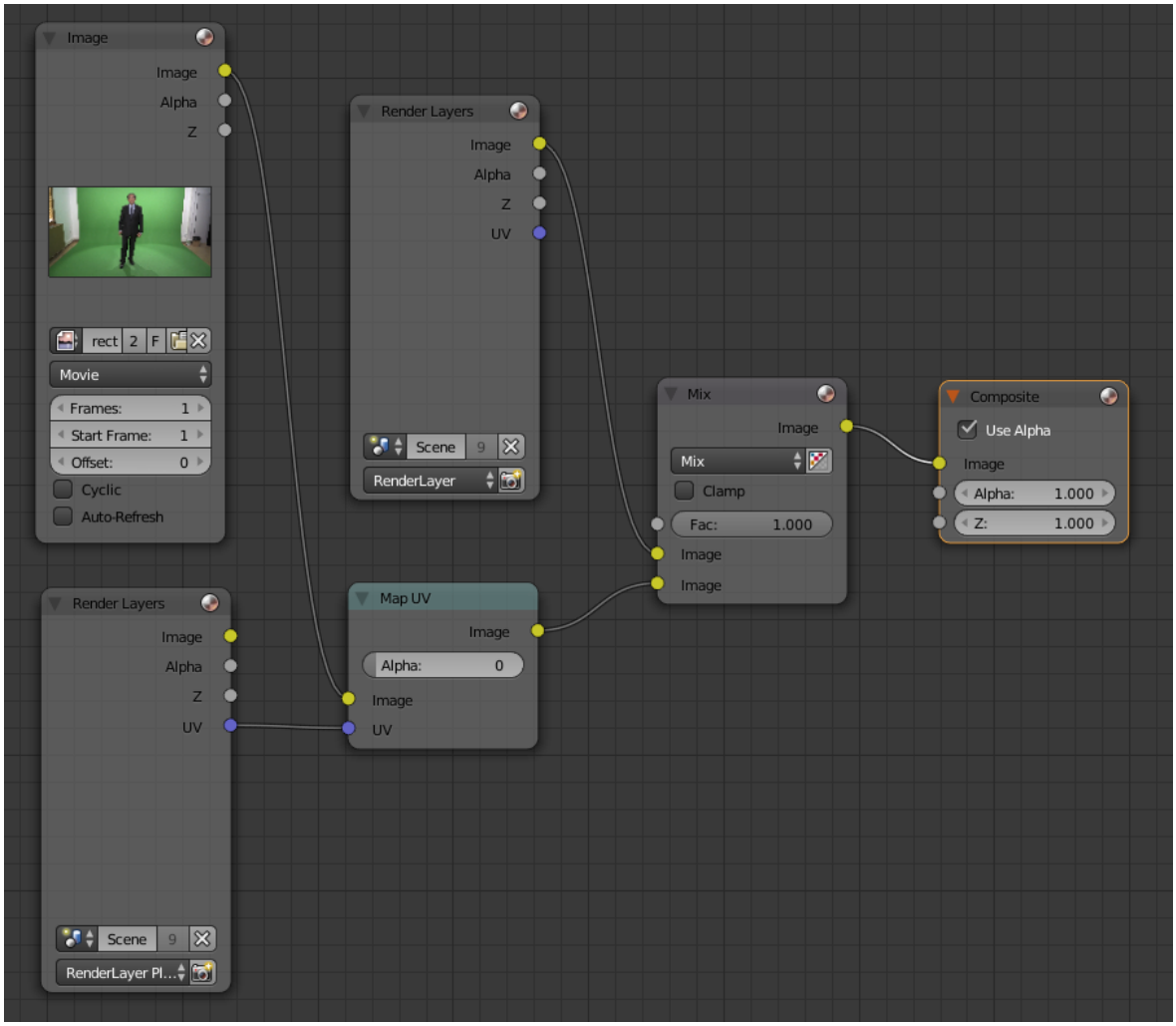

Figura 5.18: Composición de nodos para mostrar al profesor sobre el escenario sin eliminar el chroma

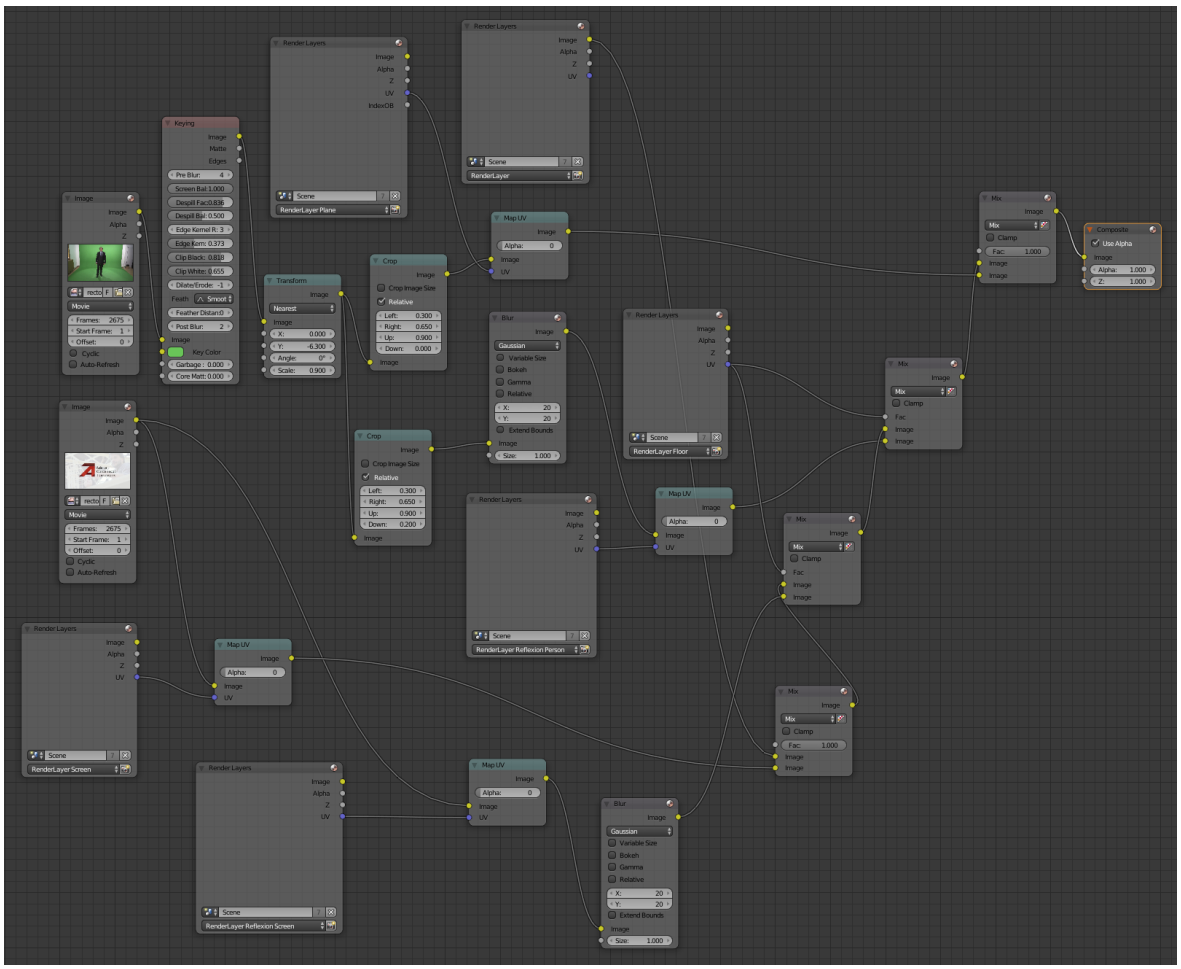

Figura 5.19: Composición del sistema de nodos para crear escenarios con reflejo en Blender

Capítulo 6

Resultados

En este capítulo se indican las iteraciones que se han llevado a cabo para el desarrollo del proyecto.

Iteración	Fecha de entrega
1	10-01-2016
2	25-01-2016
3	15-02-2016
4	02-03-2016
5	09-03-2016
6	28-03-2016
7	17-04-2016
8	15-05-2016
9	20-05-2016
10	30-05-2016

Cuadro 6.1: Fechas de entrega según iteración

6.1 Inicio

En esta fase se obtuvo una visión general del sistema, además se obtuvieron los requisitos del sistema de los cuales se pudieron deducir los objetivos (tanto principal como secundarios mostrados en las tablas 6.2 y 6.3) del proyecto.

También se realizaron avances en la obtención de información de los proyectos EasySet a través de ingeniería inversa.

6.1.1 Iteración 1

En la primera iteración se centraron los esfuerzos en la obtención de una definición de requisitos.

- **Requisitos:** Tras una serie de reuniones con las personas interesadas en el desarrollo se elaboró una lista con los requisitos que debía cumplir el proyecto en las tablas 6.2 y 6.3 se muestran los requisitos capturados.

Requisito funcional numerado	Descripción
RF.1	El sistema debe permitir la exportación de escenarios al formato EasySet.
RF.2	El usuario podrá realizar un procesado de la iluminación de los elementos del escenario antes de realizar la exportación
RF.3	El usuario podrá colocar las cámaras finales para ser exportadas al escenario final
RF.4	El usuario podrá indicar animaciones de estas cámaras antes de su exportación

Cuadro 6.2: Requisitos funcionales del sistema

Requisito no funcional numerado	Tipo	Descripción
RNF.1	Tecnologías empleadas	El sistema exportará proyectos entendibles por el programa EasySet sin necesidad de conocimiento de este tipo de archivos por parte del usuario
RNF.2	Tecnologías empleadas	El sistema deberá implementarse como un plugin de blender
RNF.3	Rendimiento	El escenario exportado deberá sobrepasar una tasa de frames preestablecido dentro del sistema EasySet
RNF.4	Estándares	El sistema hará uso de tecnologías y estándares libres en la medida de lo posible
RNF.5	Tecnologías empleadas	El sistema debe ser compatible con los sistemas GNU/Linux y Windows

Cuadro 6.3: Requisitos no funcionales del sistema

- **Análisis:** Una vez obtenidos los requisitos, se definieron los objetivos básicos que el proyecto debía cumplir.

El resultado de esta iteración fue una visión global del sistema y el descubrimiento de una necesidad de uso de ingeniería inversa para comprender el funcionamiento de los archivos de los proyectos creados por EasySet.

6.1.2 Iteración 2

En esta segunda iteración, se trató de obtener una visión general de la estructura de archivos contenidos en un proyecto EasySet, así como la interacción entre ellos.

Además se obtuvieron nuevos requisitos que no habían sido descubiertos previamente.

- **Requisitos:** Tras una serie de nuevas reuniones con los stakeholders, se obtuvieron nuevos requisitos. Concretamente la necesidad del desarrollo de un nuevo sistema que generase diagramas de barras en 3D utilizando un chroma que sería utilizado junto con la aplicación principal. Además los requisitos obtenidos en la iteración anterior fueron refinados.
- **Análisis:** Una vez se tuvo acceso a un proyecto de EasySet, comenzó la tarea de ingeniería inversa con la que se obtendrían las funcionalidades de cada archivo del proyecto. Más concretamente se obtuvo una idea general de cómo los archivos interaccionan unos con otros en este tipo de proyectos. También se comenzó con el análisis de la aplicación secundaría de diagramas en 3D obteniendo los objetivos referentes al mismo y con el estudio de aplicaciones creadas con Blender Game Engine.

Al finalizar la iteración se pudo observar que el sistema de archivos seguía una estructura parecida al formato XML en la que los archivos en las carpetas superiores del proyecto hacen referencia a los archivos de carpetas de niveles inferiores.

6.2 Elaboración

En esta fase se creó la arquitectura del sistema y además se comenzó con el desarrollo del proyecto, definiendo la estructura básica de plantillas que más tarde, en la fase de construcción se completaría.

6.2.1 Iteración 3

En la tercera iteración se continuó con la captación de requisitos. Además se comenzó con un análisis más intensivos de los archivos de proyectos de EasySet.

- **Requisitos:** En esta fase se continuó con el refinamiento de requisitos de la aplicación principal. Además se especificaron nuevos requisitos de la aplicación de diagramas de barras en 3D.

- **Análisis:** En esta fase se produce un análisis más intensivo de ambas aplicaciones. Por una parte, se elicitan los objetivos con los stakeholders de modo que cumplan las necesidades del cliente de la aplicación principal. Por otro lado se redefinen los objetivos de la aplicación de barras en 3D teniendo en cuenta los nuevos requisitos. En esta iteración se decidió usar Blender Game Engine para esta aplicación secundaria gracias a su sencillez y a la familiaridad con el mismo. Además se continuó la labor de ingeniería inversa de forma más precisa de los archivos de proyecto de EasySet. Más concretamente se comenzó el estudio de los archivos que definen los modelos 3D, los materiales y las texturas de los modelos 3D.

Al final de esta iteración se tenían los objetivos de ambas aplicaciones bien definidos listos para comenzar con el diseño de la aplicación.

6.2.2 Iteración 4

En esta iteración se comenzó con la tarea de diseño de las dos aplicaciones.

- **Análisis:** En esta fase se continuó la labor de ingeniería inversa. Una vez se tenían claras las funcionalidades de los archivos de materiales y texturas, se dio paso al estudio de los archivos de las cámaras y el archivo principal ”EasySet.py”".
- **Diseño:** Una vez bien definidos los objetivos del proyecto se continuó con la labora de diseño. En la aplicación principal se decidió dar la responsabilidad de rellenar cada uno de los archivos del proyecto a una clase distinta. De este modo, si se necesitaba modificar alguna de las clases no afectaría al resto. En la aplicación de diagramas se decidió usar una estructura simple en la que el motor se inicializase en una función y se actualizase cada frame en otra.

Al final de esta iteración se tenían unas nociones básicas sobre la arquitectura de la aplicación.

6.2.3 Iteración 5

En esta iteración se empezó con la implementación de la aplicación principal mientras se continuaba con el diseño de ambas aplicaciones.

- **Diseño:** En esta fase comenzó el estudio de la creación de plugins para Blender. Más concretamente se estudió qué clases había que heredar y las funciones que había que crear para que la aplicación se registrase correctamente como plugin en el entorno Blender.
- **Implementación:** Una vez obtenido el diseño, se empezó a implementar la clase principal del proyecto. También se comenzó con la implementación básica de la aplicación de diagramas.

Al final de esta iteración se tenía el diseño de la aplicación listo para ser implementado. Además se tenía implementada la clase principal del proyecto que contendría la lista de otras clases las cuales iban a procesar los diferentes archivos.

6.3 Construcción

En esta fase se terminó la implementación de los diversos componentes del sistema, así como el sistema de plantillas necesario para el procesamiento de proyectos de EasySet.

También se completó la implementación del sistema de diagramas de barras en 3D.

6.3.1 Iteración 6

En esta iteración se implementaron las clases encargadas de rellenar los archivos EasySet y se completaron las plantillas necesarias para esta tarea.

- **Diseño:** Una vez definido el diseño de la aplicación, se decidió utilizar el patrón State en el proyecto, ya que las clases de procesamiento simplemente tenían que procesar su archivo y continuar la ejecución, por lo que hacer que todas las clases heredaran de una común y realizaran el procesamiento llamando a una función común tenía sentido.
- **Implementación:** En esta fase se implementaron las clases que heredarían de la clase "PyFile.py"siguiendo el patrón State. Éstas se encargan del procesamiento de cada uno de los archivos de los proyectos EasySet. También se implementaron las plantillas utilizadas para este procesamiento. Llegado a este punto nos encontramos con dos tipos de clases y plantillas; las que son necesarias dentro del proyecto pero no necesitan procesamiento y por tanto son copiadas tal cual y las que sí que necesitan un procesamiento previo (por ejemplo sustituyendo alguno de sus valores o duplicando alguna de sus partes).

Al final de la iteración se tenía la aplicación principal completamente implementada, quedaba por tanto implementar la aplicación de diagramas.

6.3.2 Iteración 7

En esta última iteración de la fase de elaboración se impementó el sistema secundario de diagramas.

- **Implementación:** En esta fase se implementaron la funcion de creación de diagramas en la que se lee del archivo de configuración para establecer los parámetros iniciales y la función de actualización de los mismos en la que se hacía crecer o disminuir el tamaño de las barras según el estado en el que nos encontráramos. Además, se utilizó el sistema de bloques de blender para las partes de interfaz de usuario.

Al final de esta iteración teníamos la aplicación de diagramas completamente implementada.

6.4 Transición

Con ambos sistemas implementados completamente, se pasó a la fase de transición, en la que se realizaron pruebas de funcionamiento individuales de ambos sistemas, dando paso más tarde a las pruebas de integración entre ambos sistemas.

6.4.1 Iteración 8

En esta iteración se realizaron las pruebas de exportación de la aplicación principal.

- **Pruebas:** Una vez implementada la aplicación principal se pasó a las pruebas de exportación de escenarios al programa EasySet. Se comprobó que cada uno de los objetivos propuestos estuviera cubierto en el escenario creado.

Al final de esta iteración teníamos la aplicación primaria probada y lista para las pruebas de integración.

6.4.2 Iteración 9

En esta iteración se realizaron las pruebas de la aplicación de diagramas de barras en 3D.

- **Pruebas:** En este caso se comprobó que la aplicación funcionaba correctamente y que los parámetros introducidos en el archivo de configuracion se tomaban de forma correcta.

Al final de esta iteración teníamos la aplicación de diagramas probada y lista para las pruebas de integración.

6.4.3 Iteración 10

Una vez realizadas las pruebas se realizaron los ajustes necesarios para pasar las pruebas llevadas a cabo y se pasó a las pruebas de integración de las dos aplicaciones.

- **Implementación:** Una vez realizadas las pruebas de las dos aplicaciones en las últimas dos iteraciones, se realizó la implementación de los errores que se habían encontrado.
- **Pruebas:** Finalmente se realizaron las pruebas que aseguraban que los pequeños errores que se habían observador habían sido arreglados. Cuando se terminaron las pruebas por separado, se realizaron pruebas de integración entre las dos aplicaciones.

Una vez terminada esta última iteración se obtuvo la primera versión funcional del sistema.

6.5 Costes

En esta sección se indicarán los costes temporales y económicos.

6.5.1 Costes temporales

El desarrollo del proyecto comenzó el 10 de Enero de 2016 y terminó el 30 de Julio de 2016 dedicandóle una media de 5 días a la semana y unas 4 horas al día.

6.5.2 Costes económicos

El suelo medio de un ingeniero informático en España es de unos 25-35€ por hora. Lo que nos hace un total de alrededor de unos 19.500€ en total.

Además, para el desarrollo del proyecto se ha utilizado un ordenador HP workstation Z440 en el que se ha realizado gran parte del desarrollo y en el que se han realizado las pruebas. Está valorado en unos 1800€.

El resto de componentes hardware utilizados por EasySet no se han tenido en cuenta para el presupuesto del proyecto ya que formaban parte del material utilizado en el CTED y fueron adquiridos antes del desarrollo del mismo.

Capítulo 7

Conclusiones

En este capítulo se describirán los objetivos alcanzados con el desarrollo del proyecto uno a uno, indicando su nivel de completitud. Además se indicará el posible trabajo futuro del proyecto y las conclusiones personales.

7.1 Objetivos alcanzados

El objetivo general del proyecto ha sido cubierto de forma satisfactoria. La aplicación principal de exportación de escenarios para el programa EasySet implantado en el CTED ha sido realizado con éxito.

A continuación se analizará la completitud de los objetivos específicos del proyecto indicando cómo se alcanzaron.

7.1.1 Integración con motores de visualización de Brainstorm

Como en todo proyecto de desarrollo software, han surgido ciertos problemas. En este caso, el problema principal fue el desconocimiento del programa EasySet y su sistema de archivos ya que no se disponía de una documentación a la que consultar, por lo que gran parte del tiempo de desarrollo fue dedicado a la investigación y análisis del formato y funcionamiento de estos.

Este objetivo específico ha sido completado de forma satisfactoria a través de un estudio intensivo de los archivos que forman un proyecto EasySet y de un sistema de plantillas mediante el cual se iban rellenando los archivos necesarios para el funcionamiento del mismo.

7.1.2 Integración con herramientas de edición 3D

Este objetivo específico ha sido cubierto de forma satisfactoria casi desde el principio de la fase de construcción ya que fue uno de los primeros objetivos en ser abordado.

Dada la necesidad de una herramienta de edición 3D multiplataforma y libre, la elección estaba clara, Blender fue el software elegido.

Esto acabó siendo un acierto ya que, gracias al sistema de desarrollo e instalación de plugins de Blender y una buena documentación apoyada por una gran comunidad, hizo que

la integración en el sistema en este entorno no fuera un problema.

A continuación se detalla cómo se cumplieron las características a las que tenía que dar soporte el sistema:

Exportación de geometría 3D

Una de las primeras funcionalidades de la aplicación fue la exportación del modelo 3D sin más información que vértices y caras en un formato comprensible por el programa EasySet.

En este caso, tras la observación de algunos ejemplos de muestra en el que se utilizaba, se optó por utilizar el formato OBJ ampliamente extendido en el mundo de gráficos por computador. Por tanto, uno de los primeros pasos realizados a la hora de exportar un escenario es guardar el escenario en este tipo de formato.

Precálculo de iluminación

Gracias a la exportación del archivo en formato OBJ, como se ha indicado en el apartado anterior, también se puede obtener los datos sobre materiales en un formato estándar, MTL. A través del mismo, se pudieron obtener datos de los materiales y por tanto de las texturas y poder rellenar los datos de los archivos de EasySet según éste.

Una vez hecho esto, sólo había que exportar las imágenes utilizadas en el escenario y hacer que el path correspondiera con su ubicación especificado en los archivos de texturas e indicar IDs que identificaran cada una de las texturas de forma individual.

Exportación de múltiples cámaras virtuales animadas

Esta característica ha sido cubierta obteniendo la información de las diferentes cámaras de una escena en blender. Este proceso incluye añadir las cámaras que queramos obtener dentro del escenario en EasySet dándole las propiedades que queramos incluyendo posición, dirección, FOV y vista previa.

Este último parámetro fue el que causó más problemas pero pudieron ser resueltos de la siguiente forma: para empezar renderizamos la escena a una baja resolución (216x120) en Blender y exportamos la imagen a una carpeta temporal. Después obtenemos la imagen en un formato comprensible por el programa EasySet, en nuestro caso XPM, utilizando el conversor Imagemagick a través de la línea de órdenes. Por último, se incrustaron las líneas necesarias de este archivo en el archivo ”EasySet.py”.

Por último, las animaciones de las cámaras pudieron ser añadidas casi al final del desarrollo del proyecto. Para ello se tuvo que utilizar las propiedades custom de Blender. A las cámaras que se requiera, se añade una propiedad con nombre ”transition” dándole como valor el tiempo de animación de la anterior cámara a ésta.

7.1.3 Soporte para la integración de herramientas adicionales de representación

Este objetivo específico desembocó en el desarrollo del proyecto secundario de diagramas de barras en 3D. A través de éste, se muestran una serie de diagramas animadas con un chroma verde que posteriormente será eliminado a través de las herramientas proporcionadas por EasySet.

Por tanto, este objetivo fue completado con éxito.

7.1.4 Multiplataforma

Ambos proyectos han sido desarrollados con el fin de ser ejecutados en sistemas GNU/Linux y Windows y posteriormente probados para su correcto funcionamiento.

No ha sido posible realizar pruebas en sistemas Mac debido a la falta de disponibilidad de una de estas máquinas.

Aún así, los cambios para el funcionamiento del sistema en una de estas máquinas deberían ser mínimos gracias a la utilización de librerías estándares y herramientas externas multiplataforma.

7.1.5 Uso de tecnologías y estándares libres

Este proyecto ha sido desarrollado utilizando únicamente herramientas y librerías con licencias de código abierto por lo que este objetivo ha sido completado de manera satisfactoria.

A pesar de todo, la salida generada a través de ambas aplicaciones será utilizada por un sistema de código privativo.

7.2 Trabajo futuro

Aún habiendo cumplido los objetivos planteados para este proyecto, aún quedan algunas posibles mejoras que terminarían de pulirlo:

Animaciones dentro del escenario:

A pesar de haber realizado las animaciones en la aplicación secundaria y las de las cámaras, no ha sido posible exportar animaciones de los elementos dentro de los escenarios. Se produjo un estudio de este tipo de animaciones y su exportación en Blender, pero al final tuvieron que quedarse fuera por falta de tiempo.

Utilización de un motor gráfico más realista en la aplicación secundaria:

Una de las principales razones por las que se eligió Blender Game Engine es por la familiaridad con el mismo y su facilidad de uso a través del sistema de nodos. Aún así, algunas de sus limitaciones han dificultado el desarrollo de la misma, como no permitir la posibilidad de realizar duplicados de un mismo objeto (con materiales

y características propias) y la más importante, su bajo realismo gráfico. Queda por tanto como trabajo futuro realizar este proyecto en un motor de mejor calidad gráfica acorde al realismo de los escenarios exportados a través de la aplicación principal. Para ello se ha pensado en Unreal Engine 4 por su calidad gráfica, su accesibilidad al código, su simplicidad de uso a través del sistema de ”blueprints” y su asentamiento en la industria del videojuego. Además, también sería conveniente crear un programa con interfaz gráfica con el fin de hacer más llevadera la fase de configuración de los diagramas en vez de modificar el archivo de configuración a mano.

Conexión a través de red entre EasySet y aplicación secundaria:

Uno de las posibles mejoras del sistema sería posibilitar la sincronización entre el programa EasySet y la aplicación de diagramas. Para ello, EasySet tendría que enviar de algún modo (por ejemplo a través de sockets) la posición y la orientación de la cámara de forma que la cámara en la aplicación se pudiera actualizar y poder realizar animaciones de cámara mientras los diagramas están presentes en pantalla y así se evitaría mostrar el diagrama en la cámara que no fuera válida. Esta tarea sólo sería posible si la empresa que desarrolla EasySet, Brainstorm, diera acceso a la API para desarrolladores del mismo.

7.3 Conclusiones personales

El desarrollo de este proyecto me ha permitido aplicar una gran cantidad de conocimientos adquiridos durante los años que he cursado la carrera, en concreto los de mi intensificación, Ingeniería del Software, además de los cursos realizados también en la escuela, como el Curso de Experto de Desarrollo de Videojuegos o el curso de enseñanzas propias de OpenFL.

Por un lado cursar la intensificación de Ingeniería del Software me ha permitido obtener muchos conocimientos sobre la parte más formal del desarrollo de software, como la calidad del software, la captación de requisitos, etc. Por otro, los cursos realizados me han permitido obtener conocimientos más técnicos que no podría haber obtenido de otra manera, aplicando además los principios de Ingeniería del Software en los proyectos creados.

Además, la posibilidad de haber formado parte del equipo del CTED y crear otros proyectos como Arges[1] con compañeros de la carrera de otras intensificaciones, con mayor y menor experiencia, me ha permitido obtener una experiencia real de trabajo en equipo, en la que el trabajo en equipo en los diferentes proyectos desarrollados ha sido crucial para el éxito de los mismos. Ha sido, lo que yo considero, una experiencia parecida a lo que encontraré en un trabajo futuro.

[1] http://blog.uclm.es/cted/proyectos/arges/

ANEXOS

Anexo A

Manual aplicación principal de exportación de escenarios para EasySet

A.1 Instalación

Una vez tenemos el archivo ZIP que contiene la aplicación, el sistema se instala como un plugin más de Blender.

Para ello, dentro de Blender, vamos a ”File” ”User preferences” vamos a la pestaña ”Add-ons” y hacemos click sobre ”Install from file...” Se abrirá un explorador de archivos en el que seleccionaremos el archivo ZIP proporcionado y aceptamos.

Además, si estamos en un sistema GNU/Linux nos hará falta instalar imagemagick desde el gestor de paquetes. Por ejemplo, en las distribuciones basadas en Debian, ejecutaremos la orden:

```
sudo apt-get install imagemagick
```

Listado A.1: Instalación de Imagemagick

En sistemas Windows no hará falta cumplir ninguna dependencia, ya que se incluye un binario que se utilizará de forma estática.

A.2 Uso de la aplicación

Aunque no es necesario, es recomendable realizar una precálculo de la iluminación como se explica en el Anexo C con el fin de conseguir unos resultados de calidad.

Una vez tenemos el escenario procesado, colocaremos las cámaras donde queramos que aparezcan en el escenario. Para ello, en la ventana de ”3D View”, pulsamos la combinación de teclas ”Mayusc + A” y seleccionamos ”Camera”. Una vez insertada en la escena, la colocamos en la posición que queramos mediante transformaciones básicas (Traslación y rotación). En la figura A.1 podemos ver una escena en blender con cámaras.

Además, si queremos que estas cámaras estén animadas, tenemos que añadir el tiempo de animación de esa cámara, aunque no es indispensable añadir animaciones para cada una de ellas, depende de las transiciones que queramos hacer en el montaje. Para añadir esta propiedad, seleccionamos la cámara y vamos a las propiedades del objeto, concretamente a

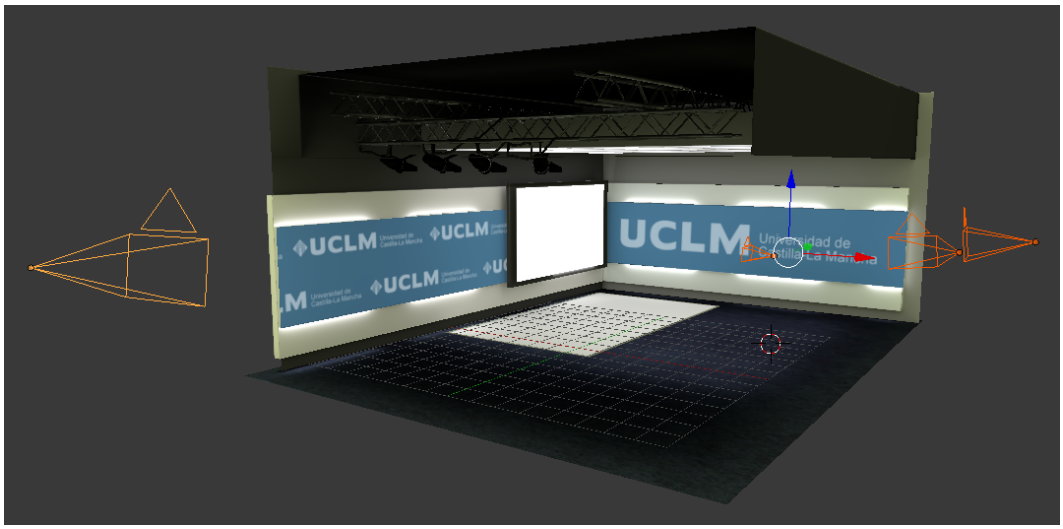

Figura A.1: Escena en Blender con cámaras añadidas

la pestaña ”Object”. En la parte inferior tenemos la parte de ”Custom Properties” y hacemos click en ”Add”. Una vez añadida, hacemos click en ”Edit” y cambiamos su nombre a ”transition” (Property Name) y le damos el valor de tiempo que queramos (Property Value) como se puede ver en la figura A.2.

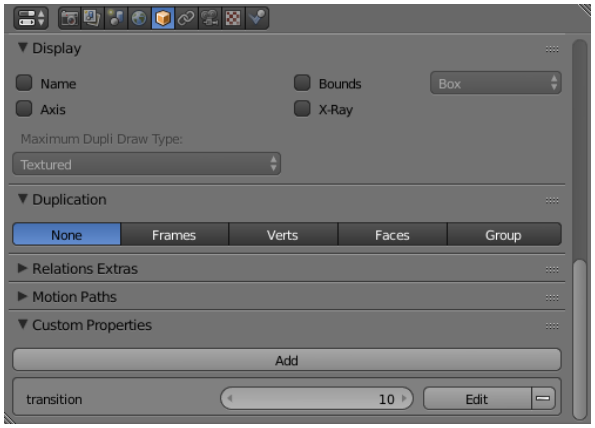

Figura A.2: Propiedad transition añadida a una cámara para activar las animaciones dentro de EasySet

Una vez tenemos colocadas las cámaras y hemos indicado su tiempo de transición pasamos a exportar la escena. Al importar el plugin se creó una opción en el menú de exportación de forma automática que será el que utilizaremos. Para ello vamos a ”File”, ”Export” y ”EasySet 3D(.py)” como se muestra en la figura A.3. Aparecerá un explorador de archivos, en ella seleccionamos la localización en la que queremos exportar el proyecto y el nombre de la carpeta que lo contendrá y pulsamos el botón ”Export EasySet Data”. Se generarán todos los archivos necesarios para la importación dentro de EasySet.

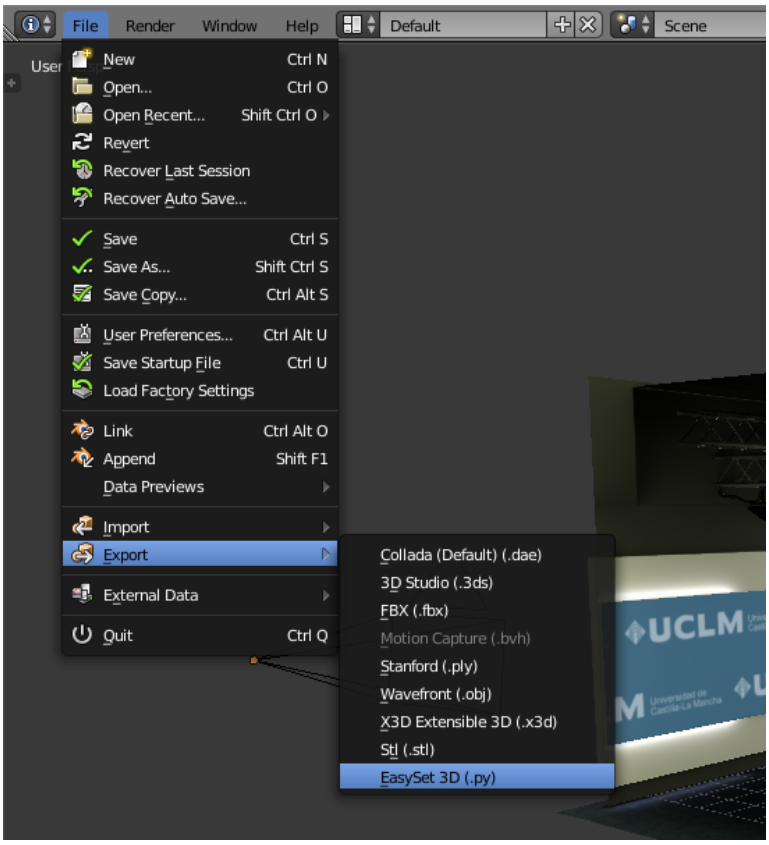

Figura A.3: Menú para exportar el escenario

Una vez generado el proyecto, para usarlo en EasySet pulsamos en ”File”, ”Load” y seleccionamos el archivo ”scenemain.py”.

El resultado se muestra en la figura A.4.

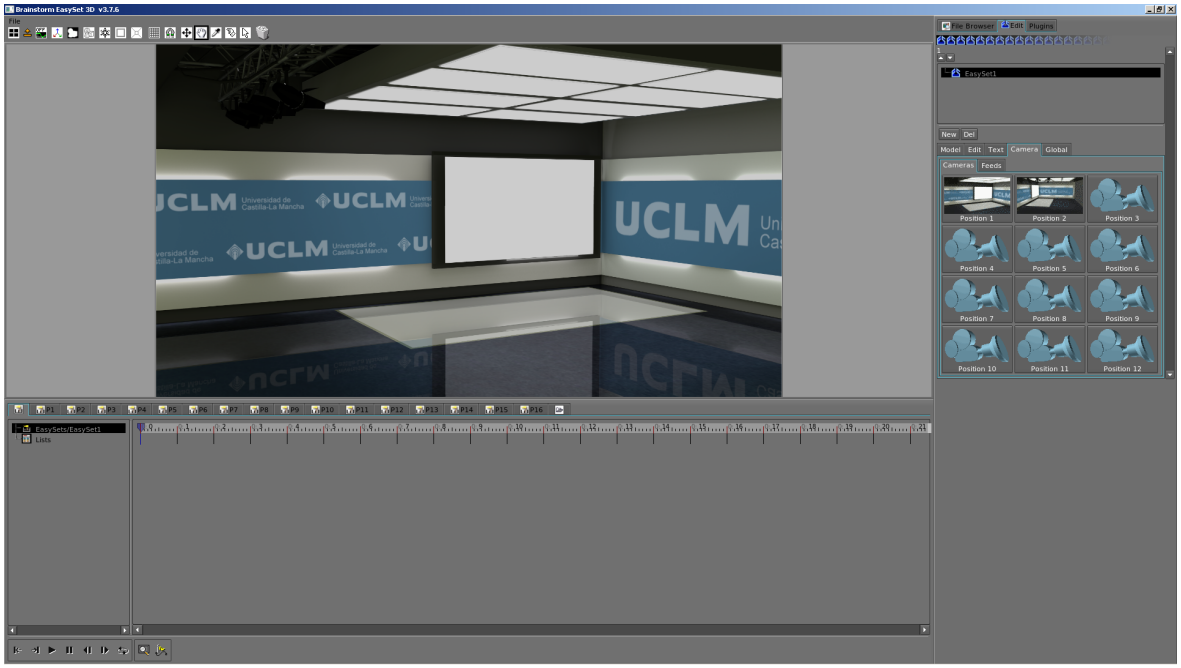

Figura A.4: Captura realizada dentro de EasySet con un escenario exportado desde Blender utilizando Aix

Anexo B

Manual aplicación diagramas de barras en 3D

B.1 Instalación

En este caso no hace falta realizar ningún tipo de instalación, ya que se trata de un ejecutable autocontenido sin ninguna dependencia. Para utilizarla descomprimimos la carpeta proporcionada y pasamos a su configuración.

B.2 Configuración

La forma de obtener los datos necesarios para la muestra de los diagramas es a través de un archivo de configuración en el que pondremos los datos que nos hagan falta.

Se trata del archivo ”config.txt”, a continuación se indican los parámetros necesarios para el correcto funcionamiento de la aplicación.

El formato de este archivo es una serie de valores separados por ”:”.

La primera línea contendrá las propiedades generales del diagrama. El primer parámetro indica si las barras del diagrama estarán colocadas de forma horizontal o vertical mediante los valores ”vertical” u ”horizontal”. A continuación (y como se ha indicado antes, separado por ”:”) se indican el ancho y alto máximos del diagrama. No debemos preocuparnos de la altura o anchura de las barras según su valor, ya que la de mayor valor tendrá el tamaño máximo establecido en alguno de estos parámetros (según sea horizontal o vertical) y el resto de barras tendrán un tamaño relativo a la más grande.

La segunda línea nos indicará la posición de la cámara en la escena en formato ”X:Y:Z”.

A partir de la tercera línea se indican los valores que queremos representar a través de este diagrama. Por cada nueva barra en el diagrama, tendremos que añadir una nueva línea con sus parámetros correspondientes. Cada línea tiene los siguientes parámetros:

- El primer valor que tenemos que poner es la imagen que se utilizará como título para esa barra, es importante hacer que el fondo de la imagen sea transparente para que aparezca bien en el escenario.
- A continuación se indica el valor que queremos representar en la barra actual. Como se ha indicado antes, no es necesario realizar un escalado de los valores ya que los tamaños serán relativos al máximo.

- El siguiente parámetro es opcional e indica el color del que queramos que sea la barra en formato RGBA (separados por ":"). Esto será útil si queremos utilizar un color representativo para cada una de las barras (por ejemplo para diferentes países).

Actualmente se pueden representar un máximo de 5 valores diferentes en el diagrama.

A continuación, en el listado B.1 podemos ver un ejemplo de archivo de configuración que es el que trae el programa como ejemplo.

```
vertical:10:5
10:-40:4
texture1.png:0.3
texture2.png:0.6:0.8:0.085:0.068:1
texture3.png:0.1
texture4.png:0.2
```

Listado B.1: Ejemplo de archivo de configuración en aplicación de diagramas

Además, para su correcto funcionamiento es necesario realizar una serie de configuraciones dentro de EasySet.

Para empezar, añadimos un nuevo input a la escena e indicamos la entrada por la que se capta la salida del portátil. Después lo dejamos activo sólo en la cámara para la cámara para la que hemos configurado la aplicación (la que está en la misma posición y orientación), de este modo sólo se mostrará el diagrama en la cámara correcta.

También es necesario indicar que se va a utilizar un chroma con el fin de no mostrar el fondo verde, es posible que tengamos que realizar unas pequeñas modificaciones en éste para que el diagrama se muestre de forma correcta.

B.3 Uso

Una vez tenemos el archivo de configuración listo y en la carpeta del programa y hemos configurado el input dentro de EasySet como se indica en el anexo **??**, sólo tenemos que lanzar el ejecutable.

Cuando necesitemos mostrar el diagrama, cambiaremos a la cámara para la que hemos configurado el diagrama y pulsaremos el botón "J". Éste aparecerá haciendo una animación. Cuando hayamos terminado de utilizarlo, pulsaremos la tecla "K" para hacerlo desaparecer con otra animación. Para salir del programa pulsaremos la tecla "Q".

En la figura B.1 se muestra lo que se verá cuando se ejecute el programa que servirá como entrada para EasySet el cual se encargará de eliminar el chroma y realizar la composición.

Y por último en la figura B.2 se muestra un ejemplo de un diagrama integrado dentro de EasySet con el chroma eliminado.

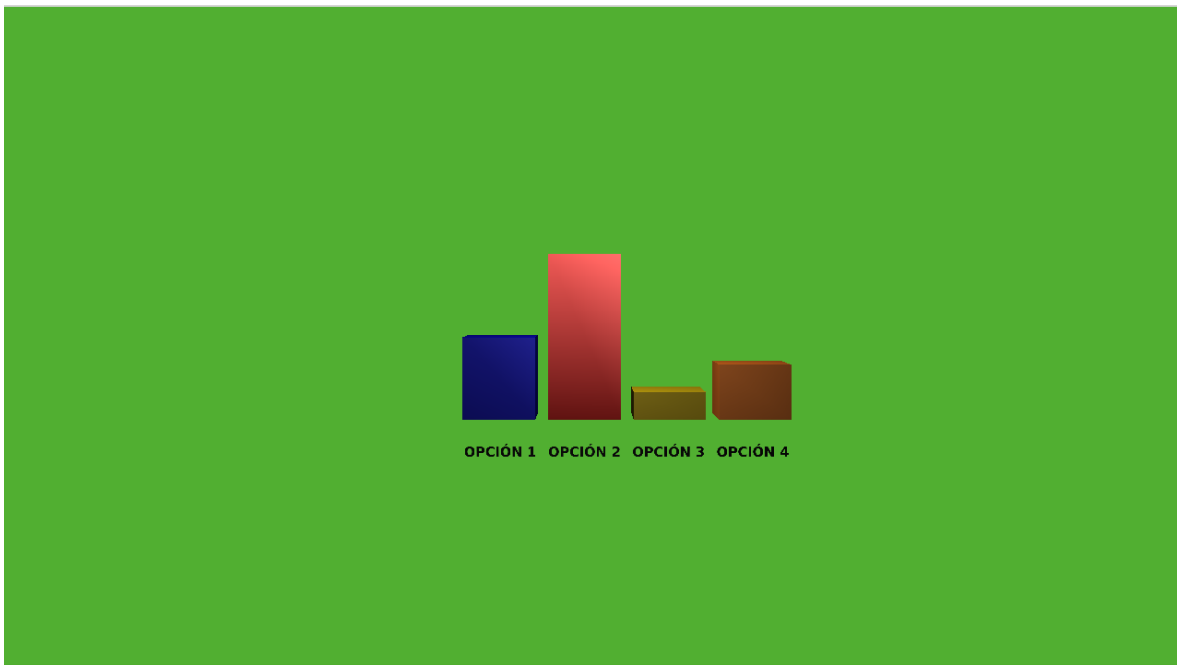

Figura B.1: Captura del programa en ejecución mostrando las barras en 3D

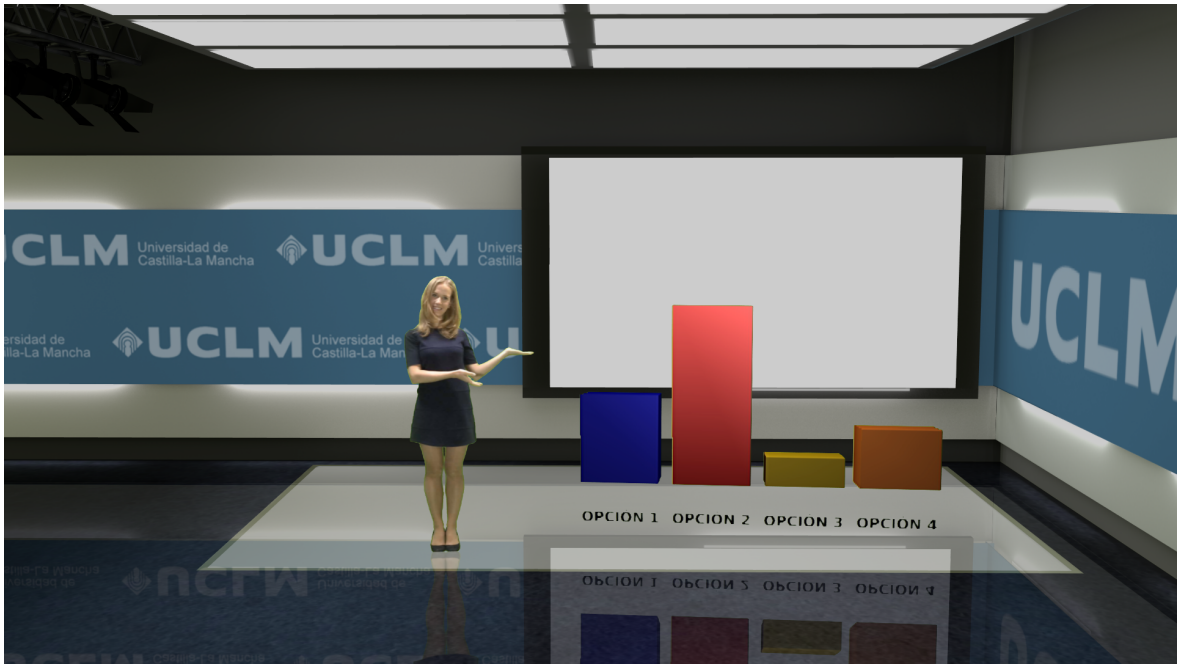

Figura B.2: Captura de EasySet mostrando un diagrama de barras en 3D

Anexo C

Anexo: Manual de bakeado de escenarios

C.1 Preparación de la escena

Una vez tenemos el escenario modelado en Blender, tenemos que cambiar el tipo de renderizado de ”Blender render” a ”Cycles render” en la parte superior de la ventana (figura C.1), ya que ”Cycles” es el motor de renderizado realista que utiliza Blender.

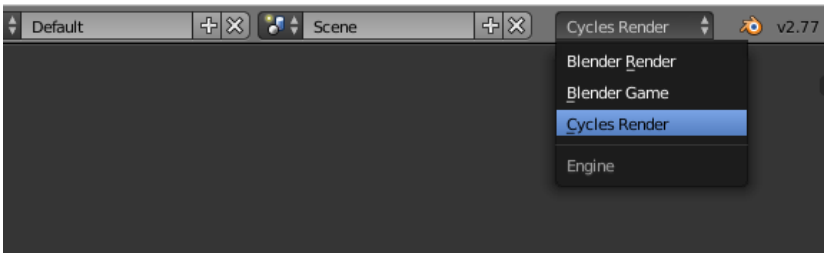

Figura C.1: Opción cyles en Blender

Una vez hecho esto, tenemos que aplicar los materiales que queramos utilizar para cada elemento del escenario. Para ello, es recomendable utilizar el sistema de nodos que soporta Blender y que podemos activar en la pestaña "Materiales"del objetoz después añadimos los nodos que queramos para ese material en la ventana de nodos.

También es necesario realizar un mapeado UV para cada uno de los elementos del escenario con el fin de poder aplicar una textura más adelante. Para ello, en el modo ”edición” (lo seleccionamos pulsando el tabulador) seleccionamos el objeto y pulsamos la tecla ”U” y seleccionamos la opción ”smart UV project” y le damos un valor distinto a 0 a la opción ”Island Margin” con el fin de dejar espacio entre las diferentes caras en la textura y que no haya interferencias entre ellas.

Además todos los elementos deben haber aplicado su escala y rotación con el fin de que se dibuje bien la textura, esto lo conseguimos seleccionando todos los elementos del escenario y pulsando ”Control + A” y haciendo click en ”Rotation & Scale”.

Una vez tenemos el escenario listo podemos empezar con la preparación del bakeado.

C.2 Preparación del bakeado

Para empezar tenemos que instalar el plugin ”UV: Texture Atlas”. Para ello vamos a ”File”, ”User preferences” y a la pestaña ”Add-ons”. Poniendo ”Atlas” en la barra de búsqueda aparecerá como primer resultado. Una vez realizado este paso podemos ver que ha aparecido una nueva opción en la pestaña de ”Render” como se muestra en la figura C.2.

Figura C.2: Pestaña render en Blender

Abrimos este desplegable y seleccionamos todos los elementos del escenario sobre los que queramos realizar el bakeado (pueden existir elementos que queramos dejarlos con su material original o ya tengan una textura predefinida), pulsamos el botón ”+” para añadir un nuevo Texture Atlas como se muestra en la figura C.3 y le asignamos un nombre, en nuestro caso usamos ”TextureAtlas”.

Seleccionamos la resolución que creamos conveniente, en nuestro caso 4096x4096 y hacemos click en “Add selected” y en “Auto unwrap”, esto hace que el mapa UV de los elementos se proyecte sobre la imagen que contendrá todos los elementos.

Ahora por cada elemento del escenario tenemos que hacer una copia del material o los materiales que tenga. Para ello seleccionamos el objeto, hacemos click en la pestaña de materiales, seleccionamos uno a uno los materiales y hacemos click sobre el botón ”+” que aparece al lado de su nombre (no confundir el con el que aparece a la derecha de la lista de materiales).

Aparecerá un nuevo material con el anterior nombre y la extensión ”.00X” (siendo X un número).

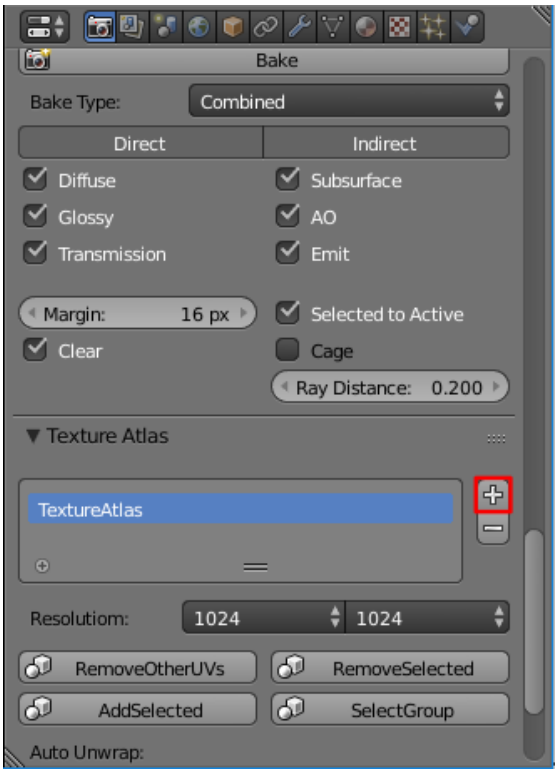

Figura C.3: Ventana de Texture Atlas en Blender

C.3 Añadido de textura

Ahora tenemos que seleccionar la textura sobre se realizará el bakeado. Si abrimos un nuevo área y nos vamos a ”UV/Image Editor”, veremos que blender ha creado una nueva textura con letras, números y colores, el Atlas. Ésta es la imagen sobre la que se bakearán los materiales y texturas del escenario.

Ahora tenemos que decirle al material la textura sobre la que queremos bakear, el Atlas que se ha creado previamente. Para ello nos vamos a la ventana ”Node editor” y seleccionar el primer material del primer elemento. Ahora añadimos una textura haciendo click en ”Add”, ”Texture” y ”Image Texture” y seleccionamos el Atlas ”Texture Atlas”. Es importante no unir este nodo con ningún otro, ya que sólo se utilizará para bakear su material. En la figura C.4 se muestra cómo queda en el diagrama de nodos. Ahora podemos copiar y pegar este nodo ”Image Texture” dentro de cada material de todos los elementos de la escena.

Después tenemos que juntar todos los elementos del escenario en uno solo. Esto lo hacemos seleccionando todos los elementos y yendo a la pestaña ”Render”, al apartado ”Texture Atlas” y haciendo click sobre el botón ”Start Manual Unwrap”.

Podemos ver que blender ha dejado de mostrar y ha deshabilitado el renderizado de los elementos individuales para dejar activado el objeto unido. Ahora tenemos que habilitar el renderizado de todos los elementos de la escena y además, por comodidad, desactivamos la

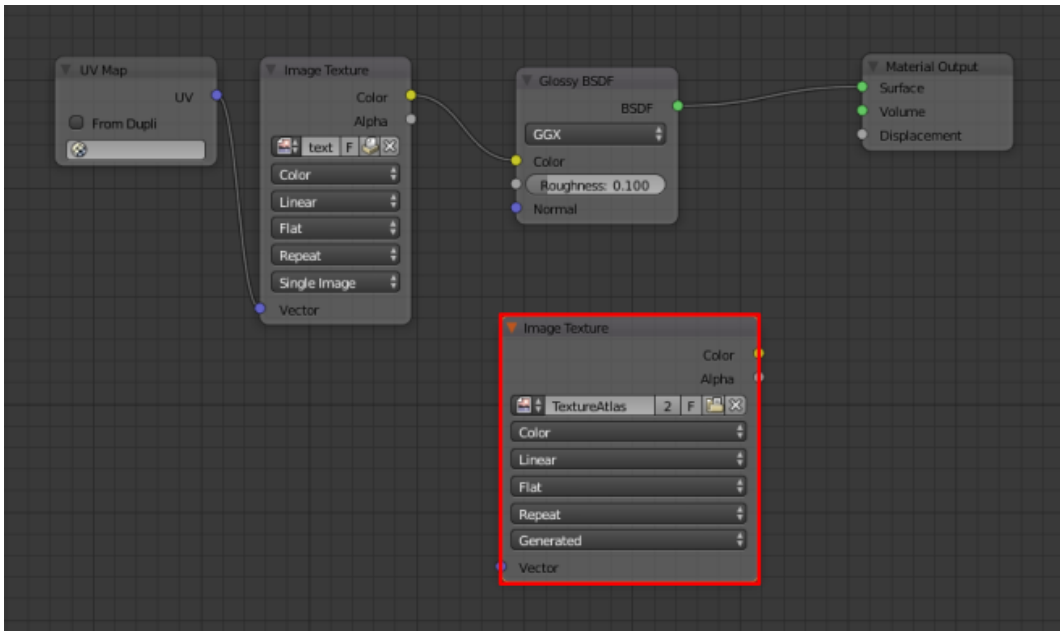

Figura C.4: Ventana de nodos tras añadir la textura

visualización del objeto unido y activamos el resto.

A continuación, por cada elemento de los iniciales (todos menos el unido) tenemos que volver a seleccionar los materiales originales, es decir cambiar de ”Material.001” a ”Material” y en la pestaña ”Object Data” eliminar el mapa UV de ”TextureAtlas” haciendo click sobre el botón a la derecha de la lista.

Nos aseguramos que en la pestaña ”Render”, en el apartado ”Bake” está activada la opción ”Selected to active” y ponemos el valor ”Ray Distance” a 0.200 y en ”Sampling” ponemos el número de samples alrededor de unos 100 para obtener una buena calidad del renderizado de luces.

C.4 Bakeado

Ahora seleccionamos todos los elementos de la escena, pero seleccionando como último elemento el objeto unido con el fin de que se dibuje todos los materiales, texturas, luces y sombras de los elementos por separado sobre el texture atlas del elemento unido. Hacemos click en el botón ”Bake” en la pestaña ”Render”. Este proceso llevará mucho tiempo incluso en equipos potentes.

C.5 Guardado

Una vez generada la imagen es muy importante hacer click en ”Image”, ”Save as image”, ya que si cerramos blender sin guardarla, la imagen se perderá y deberemos repetir el paso anterior.

En la figura C.5 se muestra el resultado obtenido en el escenario generado para la UCLM.

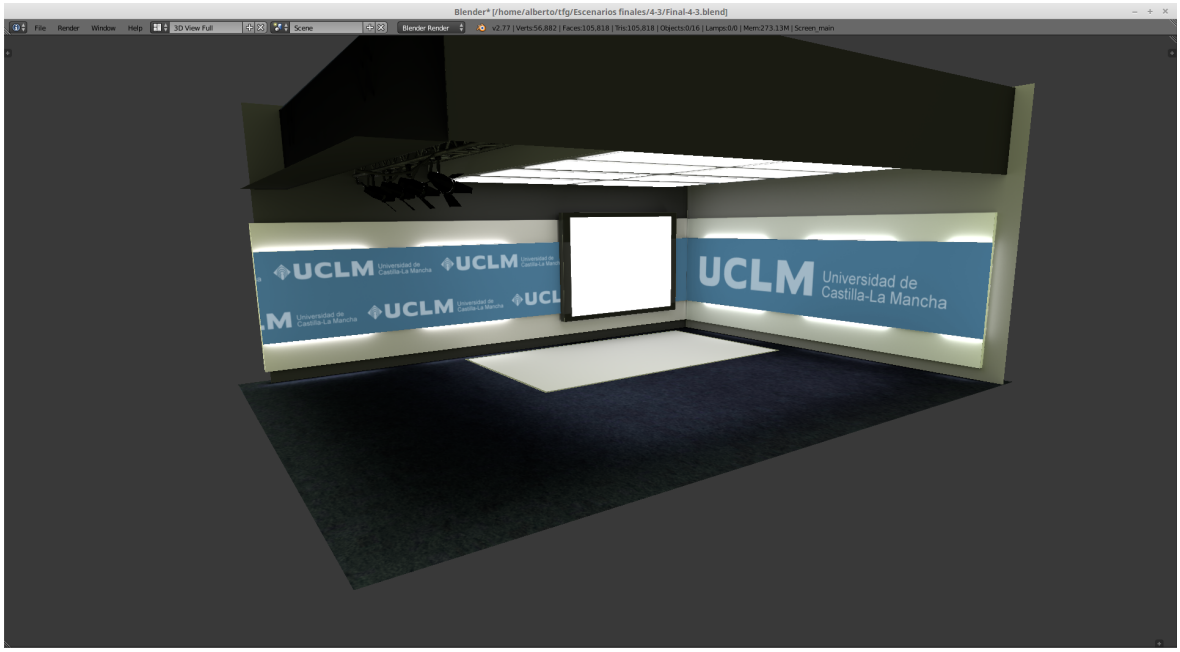

Figura C.5: Resultado del escenario tras aplicar el procesado de la iluminación

Anexo D

Manual de uso de EasySet

D.1 Configuración de feeds

En esta sección se hará un breve manual sobre cómo añadir y configurar los feeds necesarios para realizar grabaciones dentro de EasySet.

Una vez hemos exportado el escenario como se indica en el anexo A es necesario indicar los flujos de entrada que tomará EasySet y configurarlos correctamente. Para ello nos vamos a la pestaña ”Edit”, ”Camera” y ”Feed”.

Aquí nos encontramos una lista de los posibles feeds que podemos utilizar dentro de EasySet como se muestra en la figura D.1. Cada uno de estos feeds puede ser un flujo de vídeo y audio tomada de una cámara, la salida de un ordenador o incluso un vídeo que se quiera reproducir en un momento determinado.

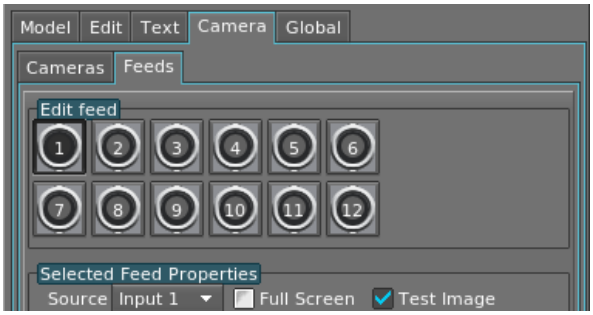

Figura D.1: Lista de feeds en EasySet

En el caso de querer mostrar el vídeo de una cámara, seleccionamos como entrada (Source) el input de la cámara que queramos mostrar, en nuestro caso el input 1 y marcamos las casillas ”Visible to all cameras” y ”Chromakey”, con el fin de mostar al profesor independientemente de la cámara que se esté mostrando y para eliminar el color elegido como chroma key.

Ahora debemos realizar las transformaciones necesarias para mostrar la entrada. Para ello, lo primero que haremos será ”recortar” la imagen para eliminar los bordes que no tengan chroma, para ello nos vamos a la pestaña ”Crop” y variamos los valores hasta mostrar únicamente al profesor con la pantalla verde de fondo como se muestra en la figura D.2.

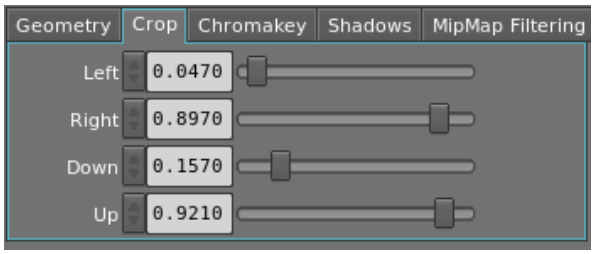

Figura D.2: Ventana de crop en EasySet

Ahora queremos modificar la posición y el tamaño con el fin de que encaje en la escena. Para ello, vamos a la pestaña ”Geometry” en la que aparecerá la ventana que se muestra en la figura D.3 y seleccionamos los valores de posición que creamos convenientes en las variables ”x” e ”y”. La posición Z se cambiará utilizando el parámetro ”Base Z”. Utilizamos esta variable y no la ”z” debido a que si utilizamos esta última y después cambiamos el tamaño, haremos que los pies de la persona queden en el aire o se metan dentro del suelo. Además podemos cambiar el tamaño de la entrada utilizando la variable ”Size”.

Figura D.3: Ventana de geometry en EasySet

Por último, tenemos que eliminar el chroma de la imagen. Para ello vamos a la pestaña ”Chromakey” en la que aparecerá la ventana que se muestra en la figura D.4 y ajustamos los valores hasta que se elimine el color de fondo.

En el caso de querer añadir un diagrama en 3D, el proceso es igual pero seleccionando como entrada el input 2 en la lista de feeds.

D.2 Grabación de vídeos

Una vez tenemos listo tanto el escenario como los inputs podemos empezar el proceso de grabación. Para ello nos vamos a la ventana de producción pulsando el tabulador o el icono más a la izquierda en la barra de herramientas.

Aparecerá una ventana como la que se muestra en la figura D.5. El uso es bastante sencillo. Para cambiar de una cámara a otra sin transición la seleccionamos en la fila inferior y pulsamos el botón ”CUT”. Si por el contrario queremos realizar un cambio con animación seleccionamos también la cámara de la fila inferior y pulsamos el botón ”TAKE” y veremos como EasySet realiza la animación que indicamos en el exportador desde la cámara

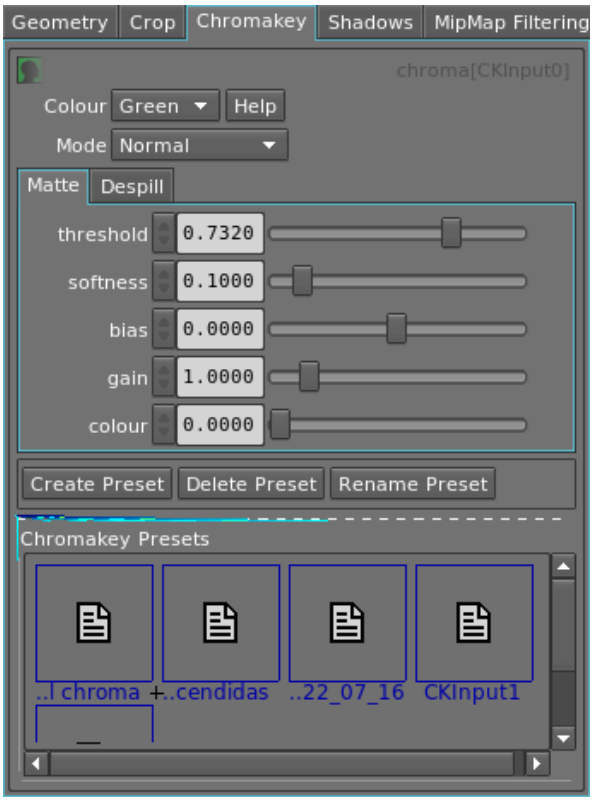

Figura D.4: Ventana de chroma key en EasySet

actual.

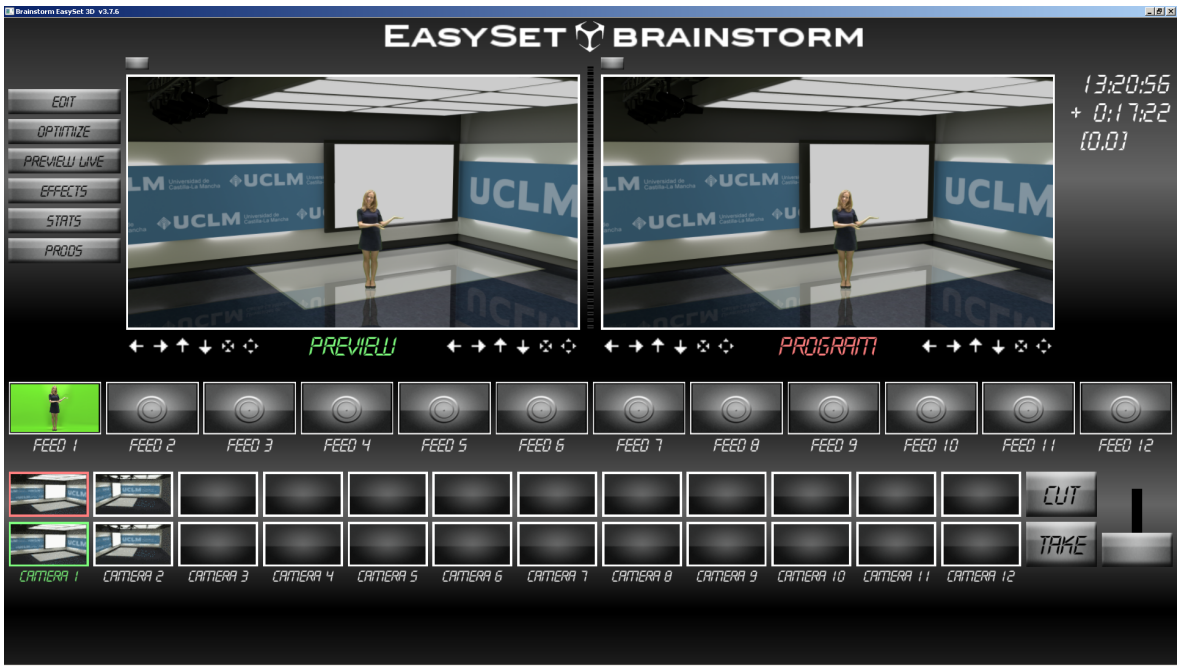

Figura D.5: Ventana de producción en EasySet

Anexo E

GNU Free Documentation License

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

The “publisher” means any person or entity that distributes copies of the Document to the public.

A section “Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve the Title” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version. 5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements”.

5. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

6. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

7. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

8. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

9. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

10. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

Referencias

[ble] Blender scripts cookbook. Disponible en http://wiki.blender.org/index.php/Dev:Py/Scripts/Cookbook. [Consulta: mar. 2016].

[Cam] Campbell. Blender api introduction. Disponible en https://www.blender.org/api/blender_python_api_2_72_release/info_quickstart.html. [Consulta: mar. 2016].

[DEGC] Dr. Cesáreo Fernández-Fernández Dr. Esteban Galán-Cubillo. La escenografía virtual en la retransmisión de grandes eventos.

[DV14] D. Villa F. Jurado F. Moya J.A. Albusac C. Martín S. Pérez F.J. Villanueva C. Mora J.J. Castro M.A. Redondo L. Jiménez J. López M. García M. Palomo G. Simmross J.L. González D. Vallejo, C. González. *Desarrollo de videojuegos, un enfoque práctico*. Edlibrix, 2014.

[DZ06] Robert O. Briggs Jay F. Nunamaker Jr. Dongsong Zhang, Lina Zhou. Instructional video in e-learning: Assessing the impact of interactive video on learning effectiveness. *Information and Managerment*, 43:15–27, 2006.

[Fos14] J. Foster. *The Green Screen Handbook: Real-World Production Techniques*. Taylor & Francis, 2014.

[HJC05] Scott D. Johnson Hee Jun Choi. The effect of context-based video instruction on learning and motivation in online courses. *The american journal of online education*, 19(4):215–227, 2005.

[JFH13] Morgan McGuire David F. Sklar James D. Foley Steven K. Feiner Kurt Akeley John F. Hughes, Andries van Dam. *Computer Graphics: Principles and Practice (3rd Edition)*. Addison-Wesley Professional, 2013.

[Lut13] Mark Lutz. *Learning Python*. O'Reilly Media, 2013.

[Nys14] R. Nystrom. *Game Programming Patterns*. Genever | Benning, 2014.

[Ram15] Luciano Ramalho. *Fluent Python*. O'Reilly Media, 2015.

[TAMH08] E. Haines T. Akenine-Moller and N. Hoffman. *Real-Time Rendering (3rd Edition)*. A K Peters/CRC Press, 2008.

Este documento fue editado y tipografiado con LaTeX empleando
la clase **esi-tfg** (versión 0.20160827) que se puede encontrar en:
https://bitbucket.org/arco_group/esi-tfg