Modelado tridimensional y creación de un sistema interactivo de la Escuela de Ingeniería Minera e Industrial de Almadén

Modelado tridimensional y creación de un sistema interactivo de la Escuela de Ingeniería Minera e Industrial de Almadén

Autor: Ángel Luis Hernández Tubío

Director del Proyecto: Javier Alonso Albusac

Especialidad: Grado en Ingeniería Mecánica

Curso: 2013/2014

Agradecimientos a toda las personas que comparten sus conocimientos de Blender en la red y a la gente de foro3d, a Oliver Villar y a Andrew Price. A la Universidad de Castilla la Mancha por permitirnos utilizar el servicio de Supercomputación, sin el cual todavía estaríamos renderizando las escenas.

RESUMEN

En este proyecto, queremos hacer una aproximación a las herramientas y conceptos del **modelado tridimensional**. El modelado 3D es una herramienta muy potente y con un gran número de aplicaciones en distintas disciplinas como arquitectura, ingeniería, medicina, cine, videojuegos, publicidad, etc.

Para ello, vamos a realizar un modelo tridimensional de la EIMIA con dos objetivos principales. Realizar un **video de promoción** donde se muestren las principales instalaciones de la EIMIA y realizar un **sistema interactivo** donde el usuario pueda recorrer libremente las instalaciones mediante una cámara en primera persona.

Utilizaremos el programa de dibujo asistido por computador **Blender**, que es un software de código abierto y gratuito. Este programa, está compuesto por diferentes módulos con los que podremos realizar todas las fases del proyecto.

Para completar el proyecto con éxito, tendremos que completar las siguientes fases:

-**Modelado:** Reconstrucción virtual de la geometría de los edificios e instalaciones que componen la EIMIA mediante el uso de herramientas simples.

-Materializado y texturizado: Mediante el uso de materiales y texturas, reproduciremos el aspecto y las propiedades físicas de los materiales.

-Iluminación: Necesitaremos reproducir la luz ambiental de los espacios para que puedan ser representados correctamente mediante técnicas como el cálculo de rayos.

-**Animación:** Utilizaremos algunas de las técnicas de animación de cámaras y objetos, para generar un video promocional donde mostraremos las instalaciones de la EIMIIA.

-Programación: Mediante el uso del motor de videojuegos de Blender, utilizaremos el editor de lógica y el lenguaje de programación Python, para generar la visita virtual interactiva.

ABSTRACT

In this project, we are going to make an approach to the **three-dimensional modeling** tools and its concepts. 3D modeling is a very powerful tool with a lot of applications in disciplines such as architecture, engineering, medicine, movies, videogames, etc.

To do this, we will make a three-dimensional model of the EIMIA with two main objectives. In one hand, we will make a **promotional video** where we show the EMIA's main facilities. In the other hand, we will programme an **interactive system** where the user can walk freely across the facilities by a first person camera.

We will use **Blender**, a computer aided design software. Blender is open source software and it is free. This program is composed of different modules that can perform the entire project.

To complete the project successfully, we have to complete the following phases:

-Modeling: Virtual reconstruction of the geometry of the builds and facilities using simple tools.

-Materials and Texture mapping: By using materials and textures, we have to reproduce the appearance and physical properties of materials.

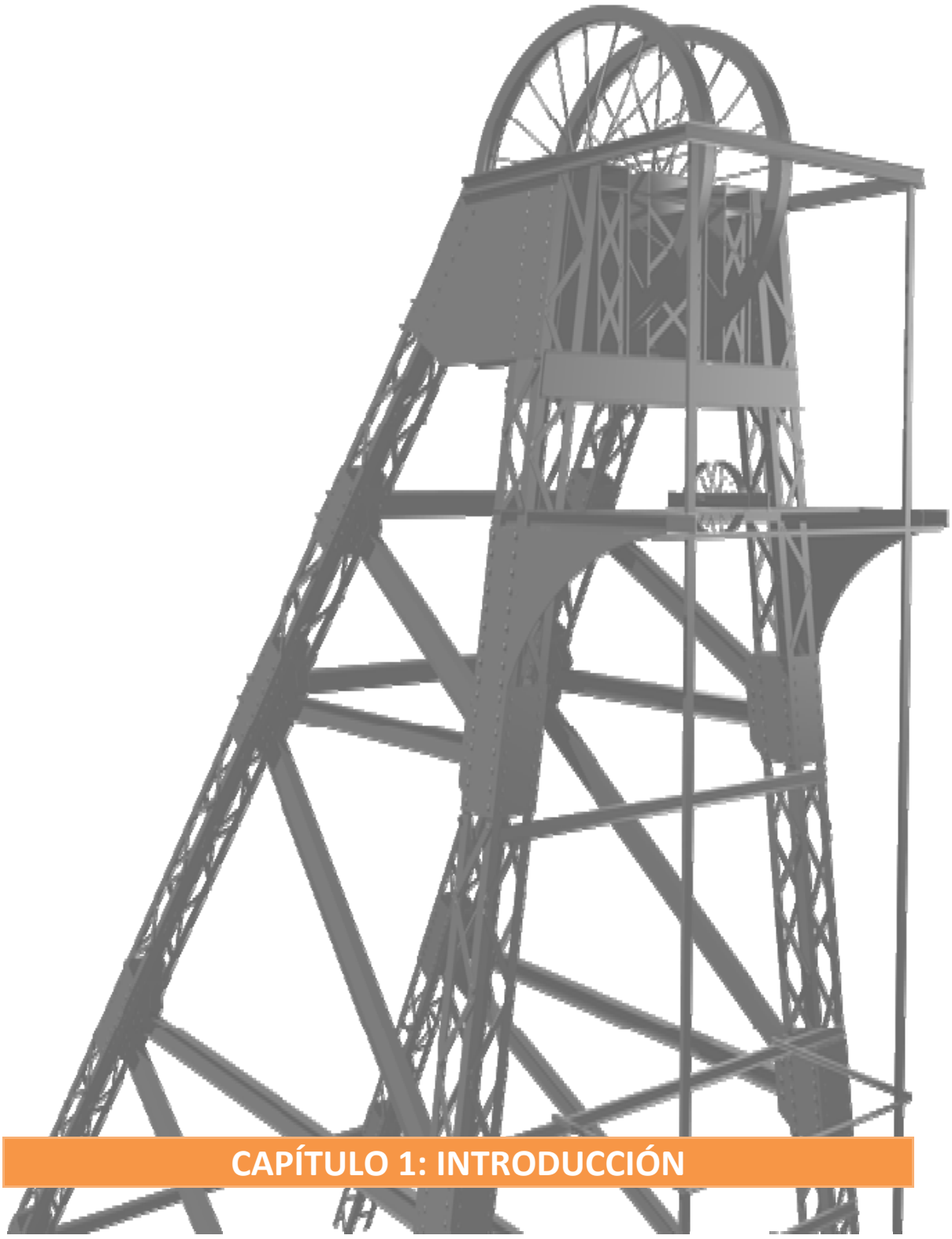
-Lighting: We have to reproduce the spaces' ambient light, so they can be properly represented by some techniques such as ray tracing.

-Animation: We will use some animation techniques with cameras and objects, to make the EIMIA's promotional video.

-Programming: In the Blender Game Engine, we have to use the Logical Editor and Python programming language for make the interactive virtual tour.

ÍNDICE

ILUSTRACIONES



CAPÍTULO 1: INTRODUCCIÓN

CAPÍTULO 1: INTRODUCCIÓN

La **infografía** es la rama de la informática que utilizamos para realizar **representaciones gráficas** mediante computador. Existen muchas formas de tratar estas representaciones, dando cada una de ellas un campo de especialización diferente. Para cada una de estas representaciones, utilizamos un determinado **software CAD** (Dibujo Asistido por Ordenador).

Hoy en día existe una gran variedad de este software, pero básicamente podemos dividirlo en **dibujo 2d** y **modelado 3d**.

Cuando hablamos de **modelado 3d**, estamos hablando de representaciones gráficas realizadas en un "mundo" con un **sistema de tres coordenadas**, donde la posición de cada punto, superficie o sólido, queda representada en relación al origen.

Durante la creación de un modelo tridimensional, no sólo definimos geometrías, sino que también definiremos texturas, sombras, transparencias, translucidez, reflexiones, iluminación, puntos de vista, movimientos, etc. Toda esa **información** es la que constituye un modelo tridimensional y será transformada en imágenes mediante un complejo sistema de cálculo denominado "**rendering**" o renderización.

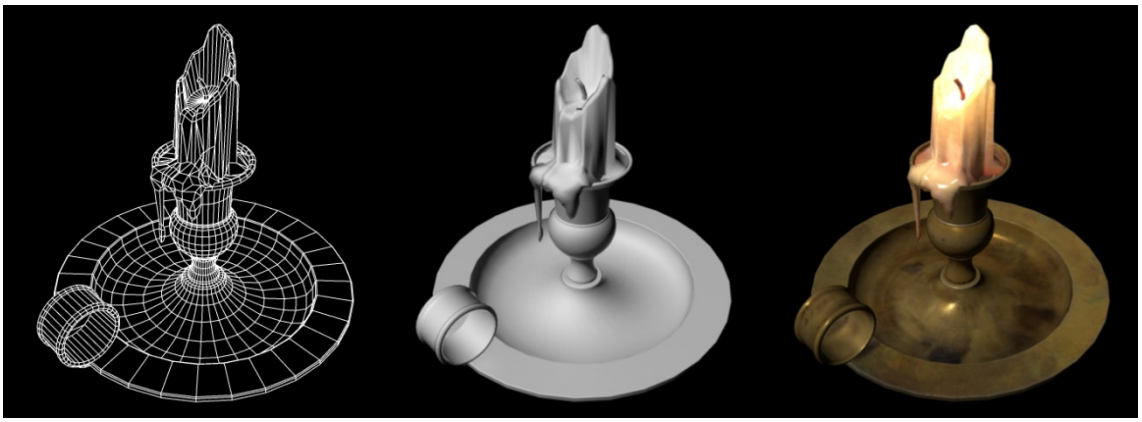


Ilustración 1 - Visualización modelo 3D - Wireframe - Shading - Rendering -

El proceso de la creación de gráficos tridimensionales comienza con un conjunto de **fórmulas matemáticas**, que se convierten en una representación gráfica del modelo 3D gracias a la **interfaz gráfica**. De esta manera, podemos editar y manipular los elementos de una forma **visual**, sin complejas fórmulas matemáticas y sin necesidad de dominar los **lenguajes de programación**.

Hoy día, el uso del modelado tridimensional se ha extendido mucho, y son muchos los **campos de aplicación** del mismo, arquitectura, reconstrucción de patrimonio, ingeniería, interiorismo, diseño industrial, medicina, publicidad, cine, animación, videojuegos… Y prácticamente contamos con un software especializado para cada uno de ellos.

Para el **diseño** en general, se ha convertido en una herramienta indispensable, ya que nos permite realizar pruebas sobre el diseño y los materiales, y además es muy útil para vender una idea aún no construida.

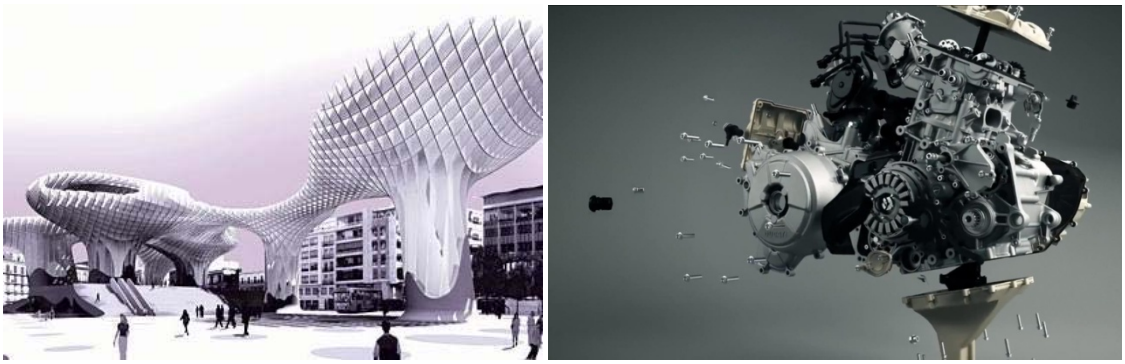


Ilustración 2 – Render en arquitectura e ingeniería

En **arquitectura e ingeniería**, el modelado tridimensional nos permite además **calcular** estructuras o comprobar el correcto funcionamiento del diseño de una máquina. Gracias a los programas de modelado 3D para el cálculo de estructuras, podemos realizar estructuras y formas muy complejas, ya que el programa es capaz de parametrizar cualquier tipo de plano o superficie, e introducirlo en los modelos de cálculo.

Uno de los mejores edificios arquitectónicos de nuestro país, el museo Guggenheim de Bilbao, construido por Frank O. Ghery, fue calculado utilizando el programa de cálculo y modelado 3D Catia. [Caicoya, 1999]

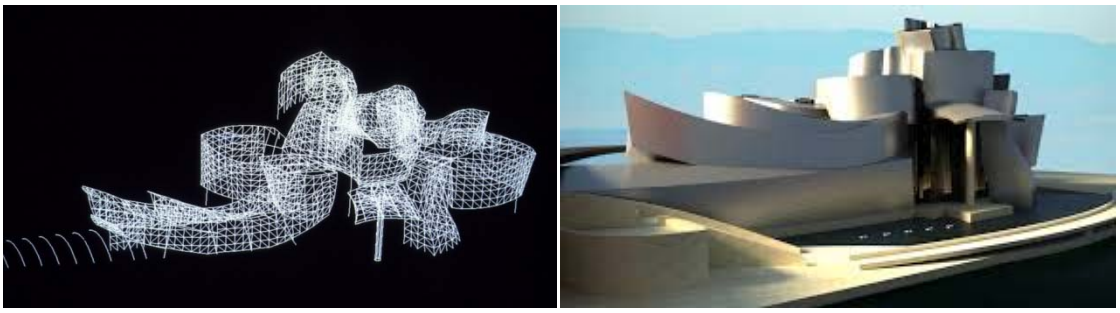


Ilustración 3 – Museo Guggenheim Bilbao

En campos como la **matricería** o en el **diseño industrial,** las nuevas **impresoras 3D**, han ampliado mucho la de aplicación de los modelos tridimensionales, ya que nos permiten la reconstrucción y prefabricación de un **modelo real** partiendo de un modelo 3D. [Armoza, 2013]

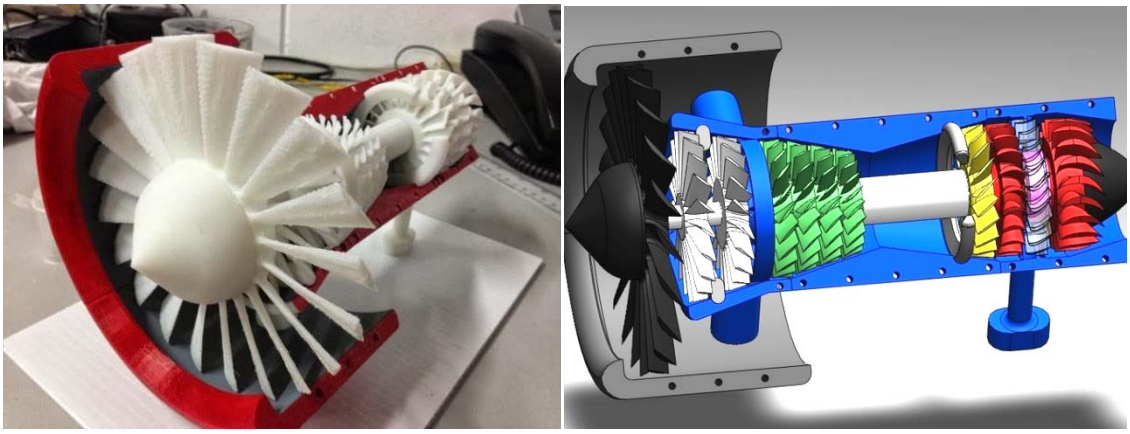


Ilustración 4 – Motor a reacción a escala mediante modelado e impresión 3D.

Este proceso es muy útil en estaciones espaciales, donde enviar piezas de repuesto supone un gran problema. De esta forma, si se necesita sustituir alguna pieza, se fabrica directamente mediante un modelo 3D y la impresora.

También se utilizan para la fabricación de **prótesis médicas**, donde son ideales para adaptar cada pieza a las necesidades del cliente. El diseño por computador permite modelar las diferentes **características** y **propiedades de los tejidos**. Los datos se obtienen mediante el **procesamiento digital** de las imágenes médicas y muestran la **forma y dimensión** de diferentes partes del cuerpo humano a reconstruir o sustituir, con una exactitud enorme. [Santos, 2013]

Ilustración 5 – Prótesis médicas - Audífono / Implante óseo

En cuanto a la industria del **cine** y de los **videojuegos**, las nuevas técnicas de renderización y animación nos permiten obtener unos gráficos cada día **más realistas.** Los software de modelado 3D, incluyen **motores de videojuegos** que permiten a usuarios no especializados desarrollar pequeños proyectos. [Hernán, 2009]

También en **publicidad,** cada día se utilizan más los **modelos virtuales,** realistas o animados, donde se pueden controlar todos los parámetros de presentación del producto.

Ilustración 6 – Render en publicidad y cine

Por lo tanto, estamos delante de una herramienta **muy potente** y cada día más **accesible** a todos gracias a los proyectos de **software libre**, que permiten a cualquier usuario iniciarse en el modelado tridimensional sin tener que desembolsar una gran cantidad de dinero en licencias.

A lo largo de este trabajo, veremos las herramientas y el proceso de realización de un modelo tridimensional. Para ello, recrearemos un **modelo virtual de la Escuela de Ingeniería Minera e Industrial de Almadén** con el software **Blender 2.68a**.

Este modelo, será utilizado para generar un **video de presentación**, para dar a **conocer las instalaciones** a los nuevos alumnos y poder utilizarlo en actividades de **promoción de la Escuela**. Gracias a la animación del modelo, podremos mostrar las instalaciones de una **forma original** y que nos permitirá utilizar una serie de recursos que no podríamos realizar en un video real de las mismas.

Además, mediante los **motores de videojuegos**, desarrollaremos un **entorno interactivo** a través del cual el usuario podrá moverse libremente por las instalaciones, tanto interior como exteriormente.

CAPÍTULO 2: OBJETIVOS

CAPÍTULO 2: OBJETIVOS

Como punto de partida, tenemos **tres objetivos** principales:

- **Modelado tridimensional**

Realizaremos un modelo tridimensional de la Escuela de Ingeniería de Almadén, siguiendo las siguientes fases:

  - **Modelado:** Comenzaremos realizando una análisis de la **planimetría** existente de la Escuela, y partiendo de las plantas 2D en formato CAD, generaremos un **modelo tridimensional** [ver 4.2] que refleje la geometría de cada uno de los edificios, tanto en el exterior como en el interior.
  - **Iluminación:** Tendremos que estudiar la correcta iluminación del modelo, para que pueda ser representado mediante el **cálculo de rayos** del programa y nos permita obtener un resultado **realista**. [ver 4.5]
  - **Texturizado:** Mediante el uso de **imágenes** y la correcta configuración de los **materiales** (transparencia, reflexión, refracción, etc.) trataremos de conseguir que la geometría adquiera un aspecto material muy parecido al que presenta realmente. Para ello nos ayudaremos de fotografías reales de los diferentes materiales, y mediante las **técnicas de texturizado** [ver 4.4.3], aplicarlos sobre los diferentes objetos.

Ilustración 7 – Fachada principal EIMIA

- **Visita virtual**

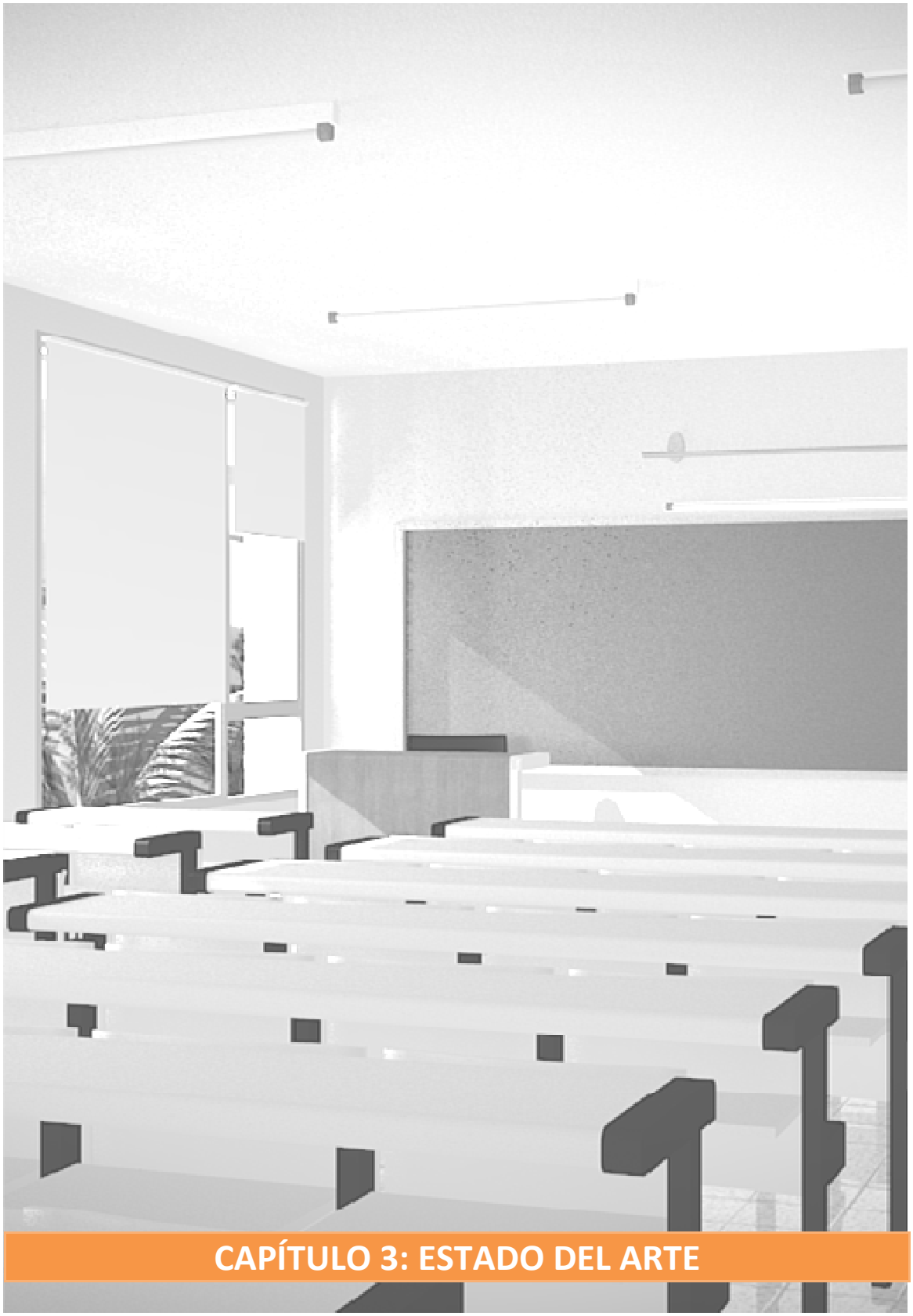
Utilizando las posibilidades de representación y **animación** de Blender, realizaremos un video recorriendo el interior y el exterior de las principales instalaciones de la EIMIA. La ventaja de realizar este video mediante modelado 3D, es que tenemos la posibilidad de mostrar la Escuela de formas que serían imposibles mediante un video real. Por ejemplo, podemos enseñar las instalaciones desde una vista pájaro, sin necesidad de contratar un helicóptero, o podemos modificar la realidad, aislando edificios o haciendo desaparecer las fachadas para mostrar las instalaciones interiores.

El resultado será un **video presentación**, que será utilizado en la página web para dar a conocer las instalaciones y **promocionar** la Escuela. Para ello tendremos que seguir las siguientes fase, una vez terminado el modelo:

- **Animación:** Utilizando las herramientas de animación de Blender, configuraremos la escena a lo largo del **timeline** [ver 4.7]. De esta forma, configuraremos **movimientos** de cámara, objetos, orden de aparición, etc. dando forma a nuestra animación.
- **Renderización:** Una vez configuradas todas las acciones que ocurrirán a lo largo de la animación, procederemos a la **generación de imágenes**. Estas imágenes son el resultado de una serie de cálculos realizados por el programa y el resultado dependerá del punto de vista, la iluminación y los materiales utilizados en la escena. [ver 3.6.8]
- **Clips de video:** Tenemos que tener en cuenta que el video lo generaremos a 24 fps (frames por segundo), por lo que por cada segundo de video tendremos 24 imágenes estáticas. Utilizando el editor de video de Blender, iremos **juntando las imágenes obtenidas** en el apartado anterior para generar pequeños clips de video. [ver 4.9]
- **Edición de video:** Una vez terminados todos los clips de video, tendremos que montarlos y añadir transiciones entre ellos, dando forma al **video definitivo**. [ver 4.9]

- **Sistema interactivo**

Desarrollaremos un sistema **interactivo** utilizando el **motor de videojuegos** de Blender [ver 4.10]. La idea es que el usuario pueda **recorrer libremente** y de forma interactiva el modelo virtual de la Escuela [ver 3.1], como si fuese un videojuego en **primera persona**. Mediante la **programación** de una cámara, podremos movernos libremente tanto por el exterior como por el interior de las instalaciones, recorriendo sus plantas y visitando las diferentes habitaciones. De esta forma los nuevos alumnos podrán conocer y visitar las instalaciones de la misma sin necesidad de desplazarse.



CAPÍTULO 3: ESTADO DEL ARTE

CAPÍTULO 3: ESTADO DEL ARTE

3.1 VISITAS VIRTUALES

Las visitas virtuales, nos permiten conocer y visitar lugares libremente desde nuestra casa, a través de la pantalla de nuestro ordenador. Se trata de un recurso con muchísimas posibilidades y cuyo uso cada vez está más extendido en sectores muy diferentes. Podemos clasificarlas en tres grupos: visitas fotográficas, visitas virtuales y visitas interactivas.

- **Visitas fotográficas**: Se trata de recorridos fotográficos o fotografías 180^{o}x360^{o}. El punto del observador es fijo, y únicamente nos permite rotar la cámara. La principal ventaja es que al ser una fotografía, lo que estamos viendo es real, pero la sensación de "visita" es muy limitada, ya que los puntos de observación viene dados y son inamovibles.

- **Recorrido virtuales:** Son videos a partir de modelos tridimensionales generados por ordenador. Normalmente realizan un recorrido exterior o interior de las instalaciones del edificio. La interactividad con el usuario suele ser nula, ya que simplemente es un espectador.

- **Visitas interactivas:** Nos permiten recorrer Modelos tridimensionales generados por ordenador, generando una libertad total de movimiento por el mismo. Este sistema es mucho más interactivo con el usuario y tiene muchas más posibilidades. En su contra, que estamos viendo un modelo reconstruido que no es real.

Las posibilidades de uso son ilimitadas, podemos encontrar desde un promotor que realiza una visita virtual de unas **viviendas** aún no construidas, a **universidades**, **hospitales** o **museos** que dan a conocer sus instalaciones a través de las mismas.

Uno de los usos que más repercusión ha tenido, es utilizar las visitas virtuales para dar a conocer el **patrimonio histórico** de las ciudades. Gracias a las visitas virtuales podemos conocer mejor algunos edificios históricos sin necesidad de viajar. Pero las visitas virtuales mediante modelos tridimensionales en este campo ofrecen muchas más posibilidades. Por ejemplo, podemos reconstruir yacimientos arqueológicos, hacer reconstrucciones del edificio en su contexto histórico, permitir visitas mientras el monumento está siendo restaurado, etc.

3.1.1 Visitas fotográficas

Es el modelo de visita virtual más extendido en la red. Normalmente utiliza un sistema de interfaz basado en Flash o similar. Desde esta interfaz accedemos a las **plantas** del lugar, o la ortofoto, donde vienen indicados los **diferentes puntos** desde donde están tomadas las **fotografías**. El usuario elige el lugar desde donde quiere ver el lugar y entra en la fotografía **180º x 360º**. Mediante el uso del ratón, podemos rotar la cámara.

Este sistema es de los más utilizados en universidades, museos, conjuntos arqueológicos, etc. Pero quizá el más conocido y utilizado, sea el Street View de Google, cuyo sistema es muy similar al utilizado en estas visitas.

- **Machu Picchu, Perú.**

http://panoramas.pe/machupicchu100.html

Ilustración 8 – Visita virtual Machu Picchu

- **Universidad de Jaén.**

http://www.ujaen.es/visita-virtual-ujaen/index.html

Ilustración 9 – Visita virtual Universidad de Jaén

3.1.2 Recorridos virtuales

Como comentábamos anteriormente, este sistema consiste en realizar un **video del lugar**, pero mediante su recreación tridimensional. Suele ser más utilizado para presentar edificios aún no construidos, en fase de **proyecto**, o realizar **recreaciones** y reconstrucciones de edificios históricos. Aunque también se utilizan en edificios construidos, ya que las **posibilidades de animación** de los modelos tridimensionales, nos permiten representarlos de muchas formas originales y diferentes a lo que podríamos mostrar en un video real.

- **Edificio Granato, viviendas en Lima, Perú.**

Es el video promocional de una promoción de viviendas. El video comienza en el exterior, y la cámara realiza un recorrido hacia las zonas comunes del interior. A partir de ahí nos muestra diferentes habitaciones de las viviendas.

https://www.youtube.com/watch?v=Mw-voVnKtcA&list=TLVocc-pTx4442fPAdNqlFZEu_UXyex99w

Ilustración 10 – Visita virtual edificio de viviendas

El modelo tridimensional está realizado por una empresa de marketing inmobiliario llamada 3DProyecta. La calidad del video es muy buena, y el realismo de las imágenes está bastante conseguido. El modelo, tanto exterior como interiormente, está muy cuidado y utiliza una gran variedad de mobiliario que ayuda mucho a la hora de dar credibilidad a la imagen. Los materiales y la iluminación de las escenas también están muy bien conseguidos. Quizá el video es demasiado lineal, pasando de unas estancias a otras y no utiliza muchos del recurso que nos permiten las animaciones 3D. Por ejemplo, podrían haber aislado una planta, de manera que se viese la distribución general de la planta. Aún así, es un recorrido virtual de una calidad muy alta.

- **Escuela Superior de Informática. Ciudad Real**

Video de presentación de la Escuela Superior de Informática. Durante el video se recorre las instalaciones de la Escuela, además ofrece una completa información sobre la labor de investigación y desarrollo llevada a cabo en los laboratorios.

http://www.inf-cr.uclm.es/virtual/

Ilustración 11 – Visita virtual Escuela Superior de Informática. Ciudad Real

El modelo tridimensional, está bastante completo, con una gran cantidad de detalles que consiguen reproducir a la perfección las diferentes instalaciones de la Escuela. También tiene unos materiales y una iluminación muy bien conseguida. Una de las cosas a destacar de esta visita, es la animación. Se han utilizado las técnicas de animación para dar vida a un robot, que es el que nos guía durante la visita, haciendo mucho más ameno y divertido el video.

- **Hospital General. Ciudad Real**

Video donde se muestran las instalaciones del Hospital General de Ciudad Real. En este caso se trata de una visita exterior. Se utilizan diferentes técnicas de animación para resaltar las diferentes instalaciones interiores, como marcar los volúmenes de las diferentes zonas, casi como si estuviésemos radiografiando el edificio. También utiliza otros recursos muy interesantes, como insertar parte virtual en una escena real.

https://www.youtube.com/watch?v=qnldtMWhspE / https://www.youtube.com/watch?v=pa2taAcaKoM

Ilustración 12 – Visita virtual Hospital General. Ciudad Real

- **Roma Reborn**

Otra visita virtual digna de mención, tanto por su calidad, como por lo ambicioso del proyecto, es **Rome Reborn**, una reconstrucción de Roma en el año 320 d.C. donde nos realizan un recorrido por la reconstrucción virtual de los principales edificios de la ciudad y por sus calles. Estamos hablando de un modelo virtual a una escala enorme, pero aún así, todos los modelos cuenta con un gran nivel de detalle. Quizá lo único que le falte al proyecto, sea un sistema interactivo que permitiese al usuario recorrer las calles libremente.

http://romereborn.frischerconsulting.com/

Ilustración 13 – Rome Reborn.

3.1.3 Visitas interactivas

Este sistema de visita virtual es, en mi opinión, el más interesante. Ya que el usuario es el que manipula el modelo tridimensional y la cámara, y tiene más libertad para decidir qué quiere ver y cómo quiere verlo. Esta interactividad hace que sea una visita mucho más atractiva, que el simple hecho de sentarse a ver unas fotos o un video.

- **Hospital universitario Los Arcos del Mar Menor. Murcia**

La visita virtual al Hospital Universitario utiliza un sistema interactivo de manipulación del modelo tridimensional. En este caso, utiliza una cámara fija y el usuario puede manipular el modelo, acercándose y alejándose, rotándolo y moviéndolo mediante el uso del ratón, como si de una maqueta se tratase. Podemos elegir entre ver una vista exterior o elegir una de las plantas interiores para ver la zonificación de la misma.

La escena exterior, tiene una grandísima calidad, que contrasta con lo limitado de las vista interiores, donde podemos decir que prácticamente es un plano con volumen donde nos van indicando los diferentes servicios del hospital.

Creo que se podría haber completado con una vista en primera persona que nos permitiese recorrer "a pie" el edificio, aunque sólo fuese exteriormente.

http://www.ffis.es/inicio/visitavirtualhospitales.php

Ilustración 14 – Visita virtual Hospital Universitario Los Arcos del Mar Menor. Murcia

- **Patrimonio español, marroquí y latinoamericano**

Fundación telefónica, cuenta en su página web con un proyecto de visitas virtuales muy interesantes. Desde esta web podemos acceder a multitud de visitas virtuales sobre el patrimonio español, marroquí y latinoamericano. Cuanta con un catálogo muy extenso de vistas, entorno a 50 proyectos, que incluyen templos, palacios, yacimientos arqueológicos, conjuntos históricos, patrimonios naturales, etc.

http://www.fundacion.telefonica.com/es/arte_cultura/arsvirtual/

Ilustración 15 – Visitas virtuales (1) Alhambra de Granada / (2) Catedral de León / (3) Anfiteatro de Mérida

Cada uno de los proyectos cuenta con su propio sistema interactivo, donde podemos encontrar diferente información sobre el patrimonio, plantas, etc. Una vez accedemos a la visita virtual, mediante el uso del teclado y el ratón, el usuario es capaz de moverse libremente por la zona elegida.

Lo más curioso de la mayoría de estos proyectos, es que utilizan modelos virtuales muy sencillos utilizando solamente formas básicas. Esta simplificación en el modelo, se ve compensada mediante el texturizado, de manera que cada uno de los polígonos está texturizado con su fotografía real. El resultado, en conjunto, queda bastante realista, ya que prácticamente estamos visualizando una fotografía. Pero dependiendo de donde nos situemos, a veces muestra un aspecto demasiado plano.

3.2 HISTORIA DEL SOFTWARE CAD

El software CAD comienza su desarrollo en la **década de los 60**, y su evolución, lógicamente, ha estado ligada a la evolución del hardware. A finales de los años 50, con el desarrollo de la segunda **generación de computadoras** y los nuevos lenguajes de programación como **FORTRAN** (Formula Translating System de IBM, 1956), se crea un clima propicio para el desarrollo del software CAD.

Uno de los proyectos más interesantes de esta época se conoce como **Sketchpad** de Ivan Sutherland. Sketchpad fue básicamente el **primer programa de dibujo por computador** y permitía la manipulación directa de objetos gráficos. Sutherland desarrollo en 1963 *"Sketchpad: A Man-Machine Graphical Communications System."* para su tesis doctoral. El sistema constaba de un **teclado**, una **pantalla CRT** (Tubo de Rayos Catódicos) y un **lápiz de luz**, que permitía dibujar directamente sobre la pantalla. Este sistema ayudó a establecer las bases del desarrollo de la **interfaz gráfica** tal y como la conocemos en nuestros días.

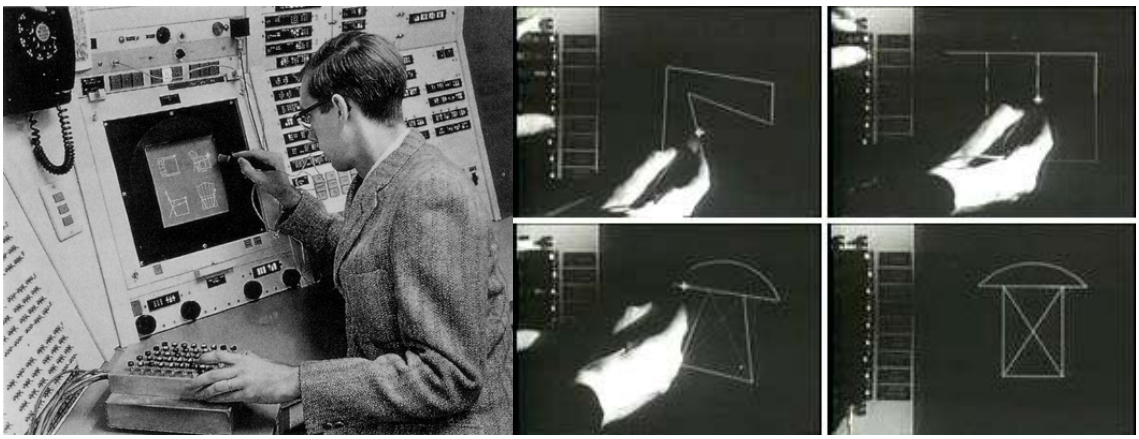


Ilustración 16 – Sketchpad – Ivan Shuterland 1963

Durante esta época, el Dr. Patrick Hanratty, conocido como el padre del CAD, diseña en IBM el software DAC (Diseño Automatizado por Computador). El **DAC-1** será el **primer programa CAD** en comercializarse y **Generals Motor** su primer usuario en 1964. Este software, supuso un enorme avance en el desarrollo de los futuros programas, ya que, según un estudio de la universidad de California en 2012, actualmente el 70% de los programas CAD3D disponibles, están basados en el código original de Hanratty.

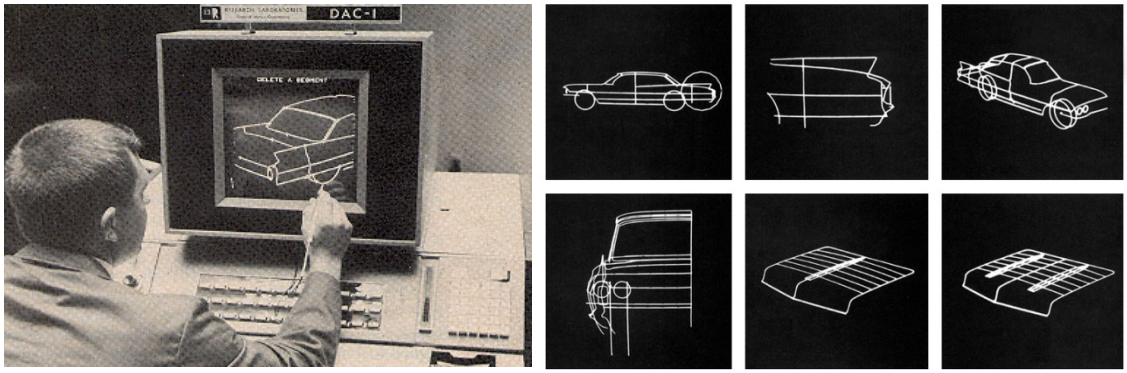


Ilustración 17 – DAC-1 – Patrick Hanratty

En 1964 empiezan a aparecer los primeros **computadores de tercera generación**, con circuitos integrados de silicio, por lo que empezaron a tener un tamaño bastante más discreto y a ser mucho más potentes. Este nuevo avance en hardware supuso un nuevo impulso en el desarrollo del software. Durante esta época comenzaron a aparecer las **primeras animaciones** por ordenador y comenzaron a comercializarse más programas de diseño asistido.

Hay que tener en cuenta que en 1965 un programa CAD costaba en torno a los 500.000 $, que sumado al precio de los computadores, suponía una **inversión** muy grande. Además, el usuario no sólo tenía que tener conocimientos sobre dibujo y matemáticas, sino que además debía saber programación. Por este motivo el uso de los programas CAD estaba prácticamente monopolizado por **grandes compañías** de vehículos, aviación e ingeniería y poco a poco iría introduciéndose en la industria del cine.

A finales de los años sesenta, en la universidad de Utah se desarrollaran importantes avances en el modelado tridimensional de formas y objetos. En 1968 Ivan Shutherland se convirtió en profesor y bajo su dirección se desarrollaron aspectos como el **sombreado Gouraud**[1] por Henri Gouraud y el **antialasing**[2][ver 3.6.8.3] desarrollado por Frank Crow. También fue importante el trabajo desarrollado por otros dos alumnos, Ed Catmull y James Blinn, que desarrollaron el **mapeado de texturas** [ver 4.4], **parches bicúbicos**[3] y **buffering Z**[4] [ver 3.6.8.2].

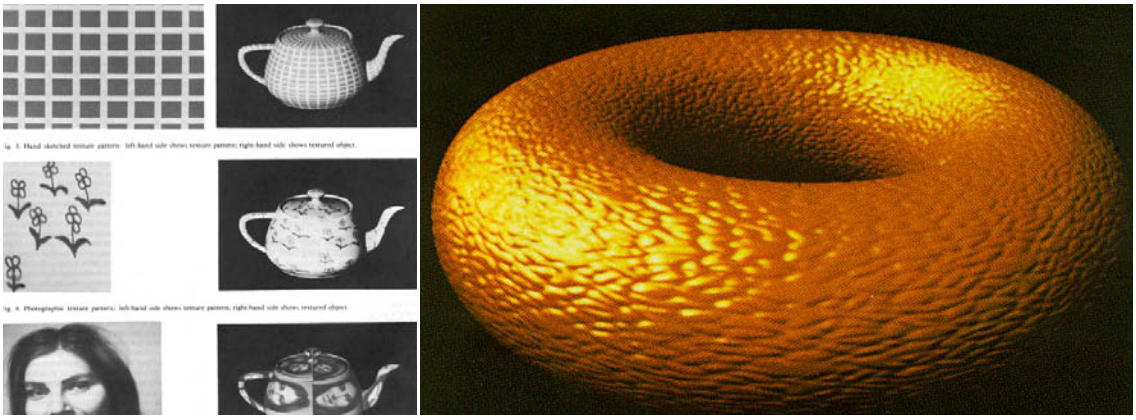


Ilustración 18 – Mapeado de Texturas y Sombreado Gouraud

En 1968 se comenzaron a usar las nuevas técnicas de **renderizado en el cine**, como por ejemplo en la película de Stanley Cubrick *"2001: Una Odisea en el espacio"*. Ed Catmull, junto con Fred Parker presentaron en 1972 la **primera animación 3D renderizada y animada** del mundo. Se trataba de un modelo de la mano izquierda de Ed Catmull.

Los años setenta, los **microprocesadores** dan paso a los computadores de cuarta generación. Esta nueva generación de computadores permite la entrada en el sector de **usuarios no especializados**. En esta década salen al mercado múltiples programas CAD como SuperPaint, CADAM, PaintBOX, Encore, Mirage, MicroCAD… y otros de modelado 3D para cálculo como **CATIA**. La **interface gráfica** comienza a ser más asequibles gracias al sistema de **Unigraphics**, pero los precios no. En 1978 los programas CAD rondan los 125.000 $.

[1] Técnica usado en gráficos 3D que simula efectos de luz y color sobre la superficie de los objetos.
[2] Algoritmo que permite evitar los artefactos y el ruido asociados a la definición de curvas y líneas, debido a la resolución finita de la imagen.
[3] Permite representar de forma sencilla superficies, mediante un número reducido de puntos de control.
[4] Gestión de las coordenadas de profundidad de los gráfico en tres dimensiones y decide que elementos son visibles y cuales ocultos.

En 1978 Steve Jobs diseña **Apple II**, y en 1982 el **ratón** se hace indispensable en el interface gráfico de Apple. **IBM** saca al mercado el terminal en **color** 3279.

Ilustración 19 – IBM 3270 –Apple II

John Walker de Autodesk, saca al mercado en 1982 la primera versión de **AutoCAD**, con la idea de hacer llegar el software CAD a los **pequeños estudios** de arquitectura e ingeniería.

En 1983, aparecen los "personal computer" o **PC**, dando lugar a la **quinta generación de computadores**. Apple desarrolla el sistema operativo **Macinthosh** sentando las bases de los **sistemas operativos actuales**. Ese mismo año, se desarrolla la interfaz gráfica de **Solid States**, sobre la que se edificará uno de los programas CAD3D más utilizados actualmente, **3D STUDIO**. En 1984 encontramos disponibles más de 40 programas de pintura y dibujo CAD. En esta década encontramos los primeros **videojuegos 3D**, aunque no maduraran hasta los años 90.

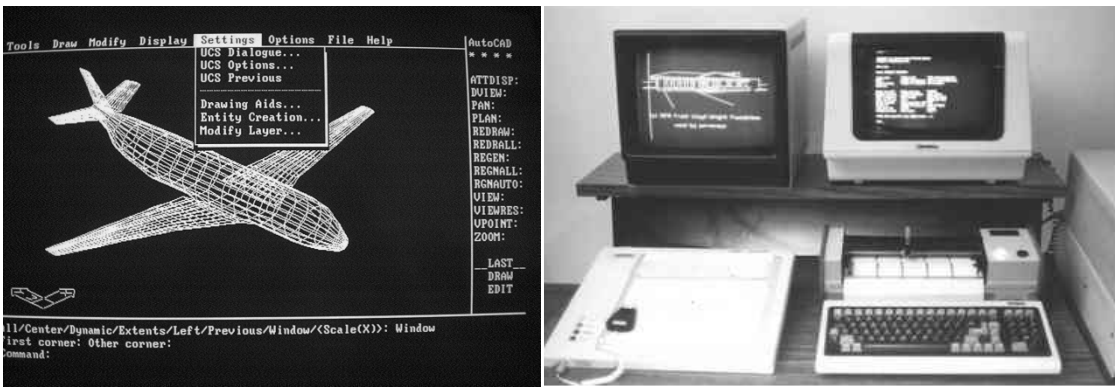


Ilustración 20 – Primera versión AutoCAD

También se realizan grandes avances en el mundo del cine y se incorporan los primeros modelos de las estaciones SGI (Silicon Graphics Inc.) a la **industria del cine**. Tenemos que destacar los resultados obtenidos por la división digital de **LucasFilm**, donde trabajaba Ed Catmull, en la saga **StarWars.** Unos años más tarde Steve Jobs comprará la división digital de LucasFilm y fundará junto con Johm Lasseter y Ed Catmull, la compañía de animación **Pixar**.

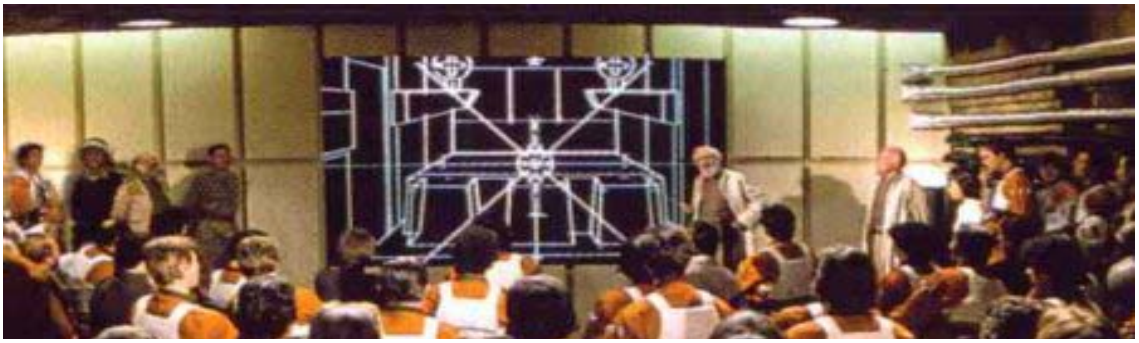


Ilustración 21 – Star Wars

En la década de los 90, empiezan a aparecer prácticamente todos los programas actuales y se **reescriben** los que ya existían. En 1991 sale al mercado la primera versión del **3DSTUDIO** en 8 bits (256 colores) y un año después la versión **AUTOCAD R12** para Windows. En 1993 aparece **MAYA**, que será comprado una década después por AutoDesk. [Autodesk, 2014]

En 1995 aparecen los sistemas de 32 bits. Ese mismo año, Pixar estrena el primer largometraje de animación por ordenador, **Toy Story**. Ton Roosendall inicia el proyecto de **BLENDER** [Blender, 2014], que pasará a ser de código abierto en 2002.

En 1997 sale a escena unos de los mejores motores de renderización actuales, **VRAY**, es un pluggin adaptable a varios programas y con su complejo sistema de cálculo de luces, es capaz de conseguir unos resultados foto realistas. Ese mismo año se termina el proyecto **Rhinoceros,** un programa muy potente para diseñar curvas, capaz de parametrizar cualquier superficie para su posterior cálculo.

En el año 2000 aparece **Sketchup**, un programa gratuito de modelado 3D muy sencillo y muy intuitivo, aunque bastante limitado, que será comprado por Google unos años más tarde.

Gracia a los sistemas de **64 bits** y la potencia desarrollada por las nuevas **CPU´s** y **GPU´s**, prácticamente todos los programas han incorporado motores de renderización capaces de realizar este proceso en **tiempo real**.

[Disseny2d1, 2010] [Mccaffrey, 2011] [Morcillo, 2011/12]

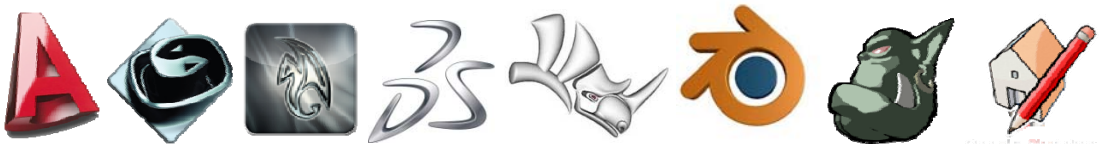


Ilustración 22 – Algunos de los programas 3D actuales.

3.3 COMPARATIVA SOFTWARE 3D

Como hemos visto existe una gran variedad de software 3D en el mercado, y cada software tiene sus ventajas y desventajas frente a los demás.

"La posibilidad de alcanzar un **trabajo de calidad** depende más de los **conocimientos** y la **creatividad** del usuario, y no tanto del software. El mejor programa 3D será aquel con el cual, el usuario se encuentre más cómodo a la hora de trabajar, teniendo en cuenta su forma y filosofía de trabajo."[García, 2009]

A continuación, haremos una pequeña descripción de dos de los programas de pago más utilizados, **Autodesk 3ds Max** y **Autodesk Maya**, y de una de las alternativas de software libre más utilizadas y potentes, **Blender**. Intentaremos ver brevemente las **ventajas y desventajas** que presentan cada uno de ellos.

3.3.1 Autodesk 3ds Max

3d Studio, es uno de los programas de animación 3D más utilizados, especialmente para **videojuegos**, anuncios de televisión, películas y arquitectura. [Chanes, 2011]

Es un **software de pago**, y su licencia de usuario cuesta en torno a los 5.000€. Podemos adquirir módulos educativos o estudiantes por unos 150€. [Autodesk, 2014]

Es una aplicación muy sólida, pero que tiene un **gran consumo de recurso**, por lo que necesitaremos un **hardware muy potente** para poder utilizarlo con fluidez. Además **no es multiplataforma**, solamente se puede ejecutar bajo **Windows**. A su favor, la gran **compatibilidad** entre diferentes **formatos CAD** (Autocad, Revit, OBJ, FBX, etc.)

Su **curva de aprendizaje** es relativamente **baja**, ya que es un programa bastante intuitivo. Un nuevo usuario podrá ser productivo en 2 o 3 meses.

Su **interfaz**, es bastante **intuitiva**, ya que utiliza un sistema ventanas y herramientas estándar, como Autocad. Aunque personalizable, la **interfaz** es bastante **rígida**.

Cuenta con más de **100 herramientas de modelado**, escultura y **manipulación** de objetos. Además el programa está basado en una arquitectura de **Pluggins**, muchos de ellos de pago, que facilitan la creación de nuestros proyectos. Además cuenta con un **buen sistema de animación, dinámica de cuerpos, cabello, ropa, fluidos, y un excelente sistema de partículas.**

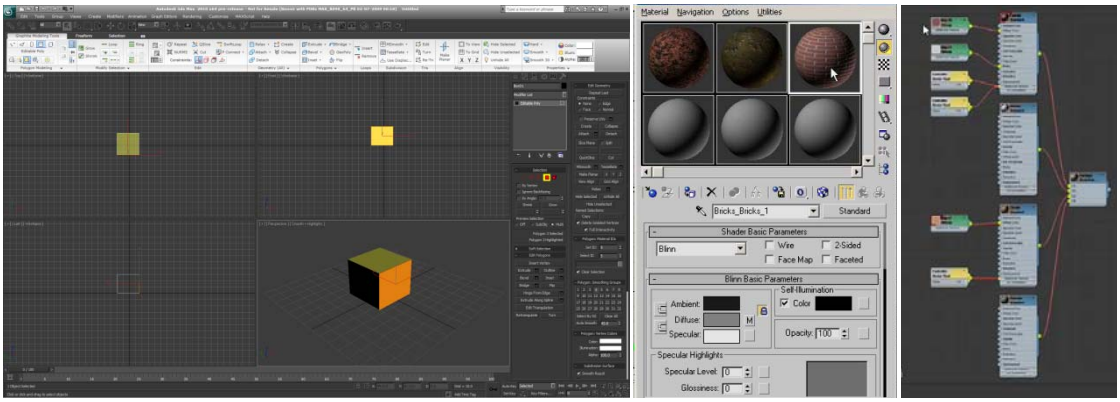


Ilustración 23 – Interfaz y Editor de materiales de 3ds Max.

El **editor de materiales es muy completo e intuitivo**. Podemos configurar las propiedades de los materiales por **Capas** o mediante **Nodos**.

En cuanto al motor de render interno, **Mentalray**, es bastante completo y se pueden obtener unos resultados muy realistas, aunque es bastante lento. Los mejores resultados se obtienen mediante el uso del motor externo de pago **Vray.**

Otra ventaja de 3ds Max, es la cantidad de **librerías y objetos** disponibles en su formato nativo, aunque la gran mayoría de ellas son de pago. [Alloza, 2010/11]

3.3.2 Autodesk Maya

Maya, es uno de los mejores programas de **animación y efectos especiales,** muy utilizado en la **industria del cine**.

Al igual que 3ds Max, es un **software de pago**, y su licencia de usuario cuesta en torno a los 5.000€. Podemos adquirir módulos educativos por unos 250€. [Autodesk, 2014]

Maya se caracteriza por su **potencia** y las posibilidades de **expansión y personalización** de su **interfaz y herramientas**. MEL (Maya Embedded Language) es el código que forma el núcleo de Maya y gracias al cual se pueden crear **scripts** y personalizar el paquete. Además es **multiplataforma**.

Su **curva de aprendizaje** es **muy lenta**, a pesar de que hay disponible una gran cantidad de documentación. Es un programa **poco intuitivo**. Un nuevo usuario no será productivo hasta los 3-6 meses.

Su **interfaz**, es **poco intuitiva**, ya que utiliza un sistema de ventanas y menús, cada uno con sus propias herramientas. El programa posee diversas herramientas para **modelado, animación, renderización, simulación de ropa y cabello, dinámicas** (simulación de fluidos), etc. Pero lo más destacable es el **sistema de animación y dinámica**.

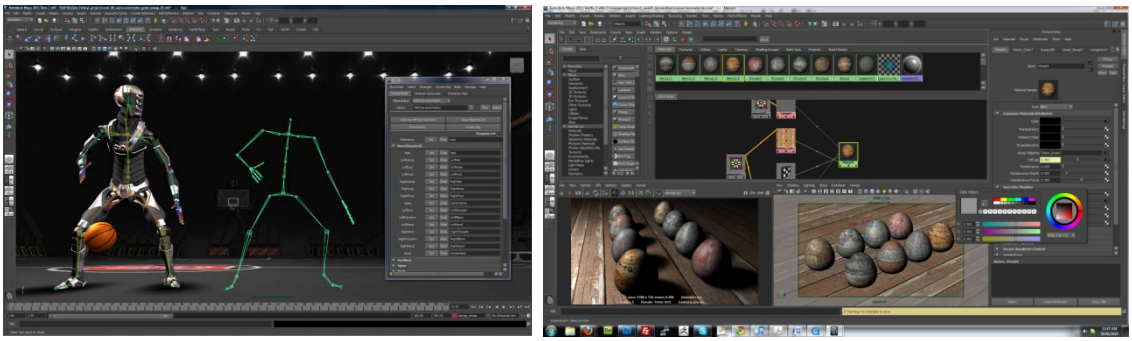


Ilustración 24 –Sistema de animación y Editor de materiales de Maya.

En cuanto al editor de materiales y motores de renderización, utiliza los mismos sistemas descritos anteriormente para 3ds Max.

La característica más importante de Maya es lo **abierto** que es al **software de terceros**, el cual puede cambiar completamente la apariencia de Maya. El mismo software se puede transformar debido a sus opciones altamente personalizables. [Autodesk, 2014]

Este aspecto ha hecho que Maya sea muy interesante para los grandes estudios que tienden a escribir mucho **código personalizado** para su producción utilizando el kit de desarrollo que viene incluido. Maya ha tenido un gran impacto en la **industria cinematográfica** como herramienta de efectos visuales, además, Maya es el único software de 3D acreditado con un Oscar.[Alloza, 2010/11]

3.3.3 Blender

Blender es un **software libre** de código abierto, y además, es totalmente gratuito. Al ser de código abierto, es un programa en constante evolución, por lo que se presenta como una **alternativa real** a los programas de pago. Aunque ha demostrado ser un software bastante potente, no cuenta con mucha popularidad dentro del sector. Esta tendencia está cambiando, poco a poco, y Blender está consiguiendo hacerse un sitio dentro de la publicidad y la animación.

Blender es **multiplataforma.** Es bastante estable y fluido, ya que consigue **gestionar muy eficientemente los recursos** del computador, por lo que no requiere un hardware demasiado potente. En su contra, la **falta de compatibilidad con otros formatos** CAD. Aunque existen algunos pluggins para importar diferentes formatos, el resultado no es siempre el esperado.

Su **curva de aprendizaje** es relativamente alta, ya que la interfaz es muy poco intuitiva. Utiliza un sistema de ventanas por funciones o tareas, que dificultan el aprendizaje, pero que permite que sea totalmente personalizable. Un nuevo usuario podrá ser productivo en 3 o 4 meses.

Cuenta con **muchas herramientas de modelado** y manipulación de objetos, además de una gran cantidad de **Pluggins** desarrollados por los propios usuarios. Cuenta también con un **sistema aceptable de animación, dinámica de cuerpos, cabellos, ropa, fluidos, y partículas.** Además de su propio motor de **videojuegos.** [Alloza, 2010/11]

Blender cuanta con dos motores internos [ver 3.4.1], **Blender Internal,** es un motor de render muy básico y con un editor de materiales bastante sencillo e intuitivo. Y **Cycles,** el nuevo motor de Blender con un editor de materiales por nodos bastante complejo, pero con unos resultados excelentes en iluminación y muy realistas, que hace de Blender una herramienta más competitiva con el resto software de pago. [Blender, 2014]

Además también tenemos motores externos como **Yafray o Luxrender,** también con unos resultados bastante realistas, pero con algunas limitaciones en el cálculo de la iluminación. **Vray** [Chaos Group, 2014] está planeando lanzar al mercado una versión para Blender, la que es una excelente noticia para los usuarios y señal de la creciente popularidad de Blender. El único problema será ver si tiene éxito un motor de render de pago en un programa gratuito.

Blender cuenta con una amplia **comunidad de usuarios**, que mantienen el contacto mediantes **foros** y páginas de **tutoriales y bibliotecas**, y siempre está dispuesta ayudar a los usuarios recién iniciados.

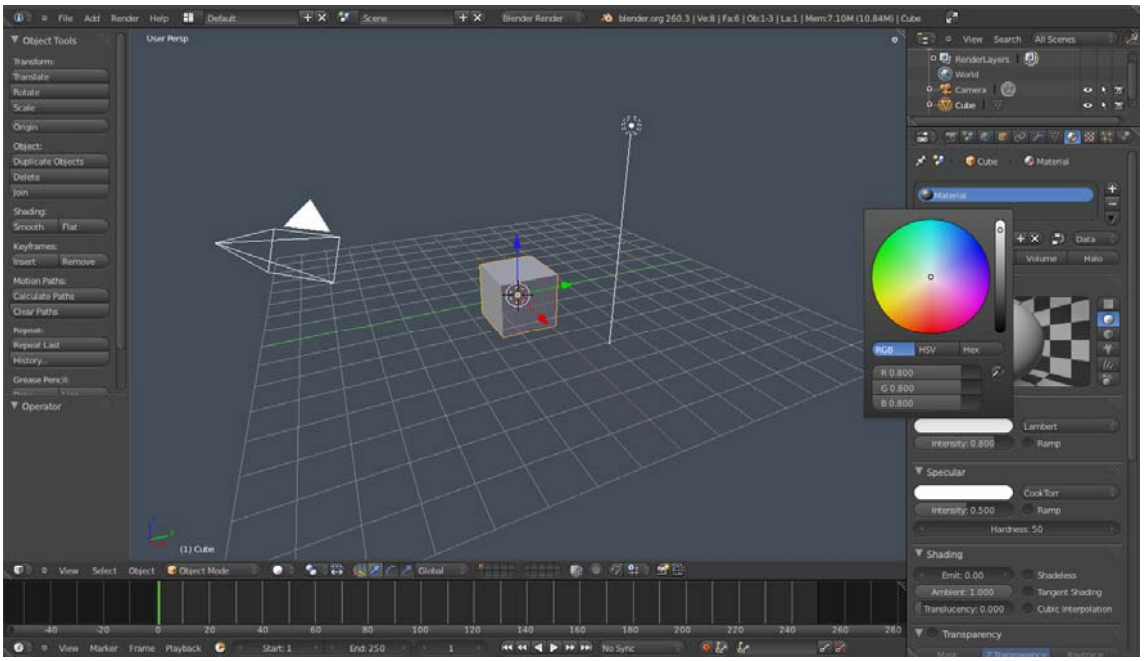


Ilustración 25 –Interfaz de Blender.

3.3.4 Tabla comparativa

	Autodesk 3ds Max	Autodesk Maya	Blender
Precio	5.000€	5.000€	Gratuito
Plataforma	Windows	Windows/MAC/Linux	Windows/MAC/Linux
Popularidad Industria	Muy buena	Muy buena	Baja
Curva de aprendizaje	< 3 meses	< 6 meses	< 4 meses
Interface	Estilo CAD	Flexible y nada intuitiva	Flexible y poco intuitiva
Documentación	Buena	Excelente	Buena
DVD entrenamiento	Buena	Muy buena	Buena
Importar/Exportar	Excelente	Muy buena	Baja
Motor de Render	MentalRay	MentalRay	Internal/Cycles
Calidad	Muy buena	Muy buena	Buena/Muy buena
Motor externo	Vray	Vray	Yafray
Calidad	Excelente	Excelente	Muy Buena
Animación	Muy bueno	Excelente	Bueno
UV tools	Muy bueno	Excelente	Excelente
Modelado	Excelente	Muy bueno	Muy bueno
Modificadores	Excelente	Muy bueno	Bueno
NURBS	Bajo	Muy bueno	Bajo
Dinámica	Muy bueno	Excelente	Muy bueno
Cabellos	Muy bueno	Muy bueno	Bueno
Partículas	Excelente	Muy bueno	Bueno
Ropa	Muy bueno	Muy bueno	Bueno
Fluidos	NO	Muy bueno	Muy bueno
Compositor	NO	NO	SI
Editor de Video	NO	NO	SI
Motor Videojuegos	Excelente	Muy bueno	Muy bueno
Scripting	Bueno	Excelente	Muy bueno
Desarrollo script	Maxscript	MEL, Python	Python
Desarrollo C/C++	Bueno	Excelente	Muy bueno

[Alloza, 2010/11]

3.4 BLENDER 2.68a

La elección del software se llevó a cabo teniendo en cuenta que necesitábamos un programa de **software libre**, ya que al tratarse de un proyecto dentro del ámbito educativo, no nos podemos permitir el coste de las licencias. Además de esta forma el proyecto podrá ser utilizado y manipulado en un futuro sin preocuparnos por el coste del software. Así que comenzamos una búsqueda en la red sobre qué programa podría ser el más indicado para la realización del proyecto, ya que necesitábamos que fuese un **software de animación** y que incluyese **un motor de videojuegos** capaz de generar la visita virtual. La alternativa que parecía más viable era **Blender**, ya que es completamente gratuito. Y según la experiencia de otros usuarios, es un programa muy potente, **comparable a otros programas comerciales** como **Maya** o **3D Studio**.

Blender no es sólo un programa de renderizado, ya que incluye dentro del programa diferentes **módulos**. Entre ellos podemos destacar el módulo de animación, el motor de videojuegos, el módulo de postproducción y el módulo de edición de videos. Por lo que podríamos realizar todo el proyecto con un **único programa** sin tener que recurrir a un software específico para cada tarea. Además, el hecho de que sea de **código abierto**, es un factor muy importante, ya que existen **miles de scripts** programados por otros usuarios que lo complementan y permite que esté en constante evolución. Blender utiliza el lenguaje de **programación Python** y para el motor de videojuegos utiliza un **sistema de nodos** muy intuitivo, que nos permite programar los objetos sin necesidad de conocer el lenguaje de programación. Este factor, hace que Blender sea más atractivo ante otras opciones como **Ogre3D**, donde tenemos que saber programar con **C++** para poder desarrollar un videojuego.

3.4.1 Motor de render de Blender.

En cuanto al **motor de render**, Blender cuenta con dos motores internos incluidos en el programa.

El primero es **“Blender internal Render”** que es el motor original del programa. Es un motor bastante básico, que realiza el cálculo de la luz mediante CPU, pero con el que se pueden conseguir fantásticos resultados utilizando un buen modelo, una iluminación adecuada y trabajando bien los materiales.

El segundo es **"Cycles"**, es el nuevo motor de Blender, y pronto sustituirá definitivamente al motor interno. Este motor realiza un cálculo de la luz mucho más complejo y con él, se puede obtener imágenes **foto-realísticas**. Es capaz de realizar el cálculo mediante **GPU nvidia**, reduciendo considerablemente los tiempo de render y siendo capaz de obtener pre-visualizaciones en **tiempo real**.

En el proyecto utilizaremos **"Blender internal Render"** ya que, de momento, es el único motor de render compatible con el **motor de videojuegos**.

3.4.2 Interfaz de Blender

La interfaz de usuario es el medio de **interacción** entre el usuario y el programa. El usuario se comunica con el programa por medio del teclado y el ratón, y el programa ofrece respuestas por medio del sistema de ventanas.

Para los usuarios familiarizados con otros softwares 3D, encontrarán que la interface de Blender es algo **diferente**, y puede resultar confusa en un principio. Pero una vez familiarizados con ella, las posibilidades de **personalización** que ofrece, nos permite trabajar extremadamente y productivamente rápido.

La interfaz de Blender está compuesta por **17 ventanas principales**, cada una con una **tarea específica** dentro del programa. Al ser una interface completamente realizado con OpenGL, cada ventana puede ser **configurada**, aumentada, disminuida y movida alrededor del espacio de trabajo. Por lo que para cada tarea, podemos configurar el espacio de trabajo con las ventanas que necesitemos, incluso distribuirla entre varios monitores, y por supuesto **guardar la configuración** para trabajos posteriores.

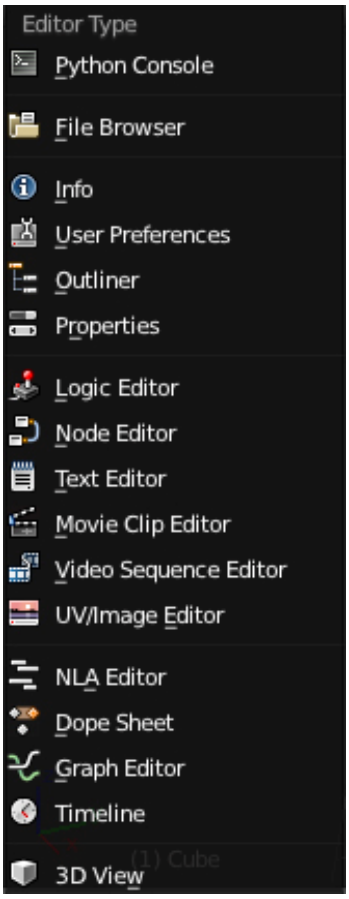


Ilustración 26 –Ventanas Principales de Blender

A continuación, vamos a ver brevemente cada una de estas ventanas:

- **Python console:** Es la consola de programación de Blender, desde ella podemos programar y cargar los diferentes pluggins del programa.

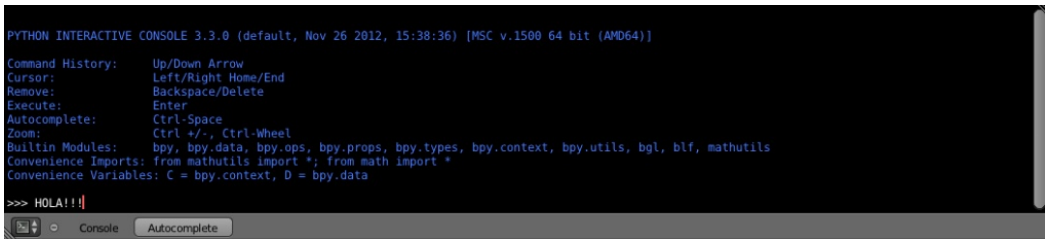


Ilustración 27 –Python console

- **File Browser:** Es el explorador del sistema. Desde ella, podemos abrir y guardar los diferentes tipos de archivos utilizados y generados con Blender.

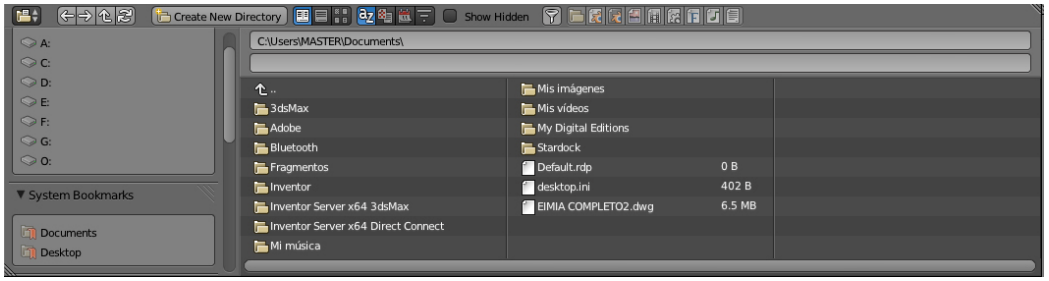


Ilustración 28 –File Browser

- **Info:** Es la barra de herramientas principal del programa, desde donde abrir y guardar los archivos de Blender, importar, exportar, añadir nuevos elementos, cambiar de escena, elegir motor de render... Además nos ofrece una serie de información útil sobre la escena, como número de vértices, caras y polígonos, memoria usada, etc. En la parte superior, podemos ver el código Python que está ejecutando el programa mientras realizamos cualquier tarea.

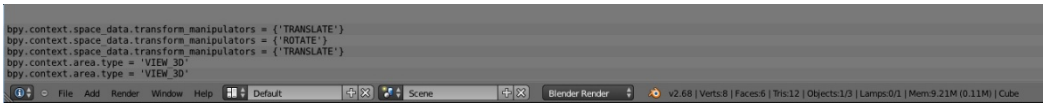


Ilustración 29 –Info

- **User preference:** Ventana desde la que podemos configurar y personalizar cualquier aspecto del programa. Desde los colores de la interface a cambiar los accesos rápidos del teclado.

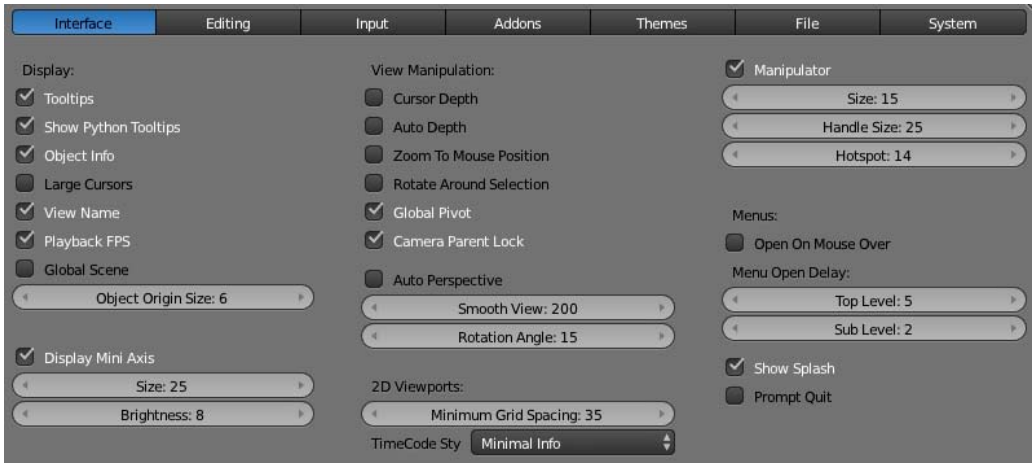


Ilustración 30 –User preference

- **Outliner:** En esta ventana nos aparecen las escenas y todos los objetos contenidos dentro de ellas. podemos seleccionar objetos, bloquearlos y desactivarlos.

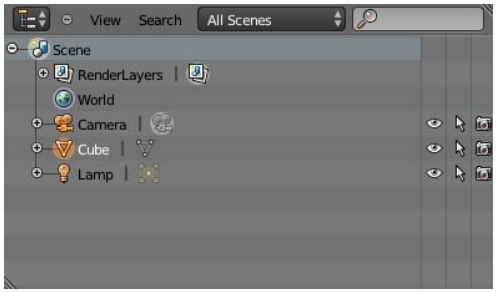


Ilustración 31 –Outliner

- **Property:** Una de las ventanas más importantes del programa. Contiene múltiples pestañas donde podremos editar las propiedades de cada objeto, además de la configuración global de la escena. Las opciones dentro de esta ventana cambiarán dependiendo del tipo de objeto que tengamos seleccionado.

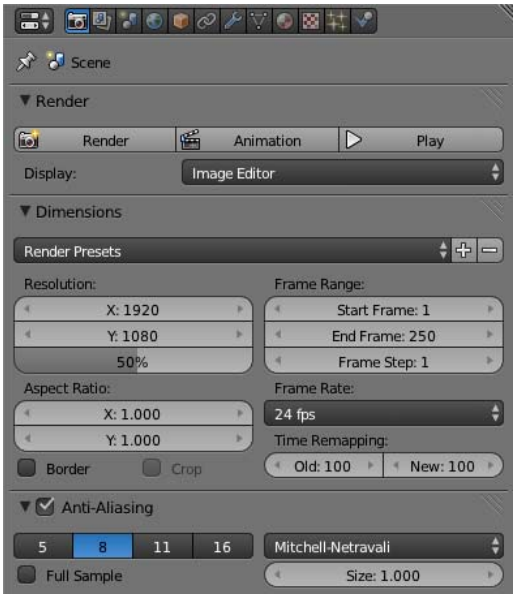


Ilustración 32 –Property

- **Logic Editor:** Desde esta ventana contralamos el motor de videojuegos. Utiliza un sistema muy sencillo de nodos y puertas lógicas para configurar las propiedades y el comportamiento de cada objeto dentro del videojuego.

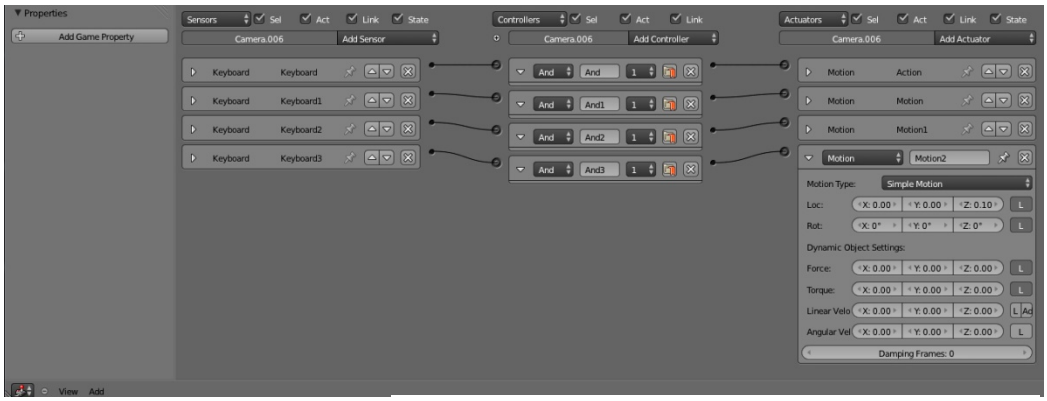


Ilustración 33 –Logic Editor

- **Node Editor:** El sistema de editor de nodos. Desde él, podemos componer la imagen final editando multitud de efectos. También lo usaremos para editar los materiales en cycles.

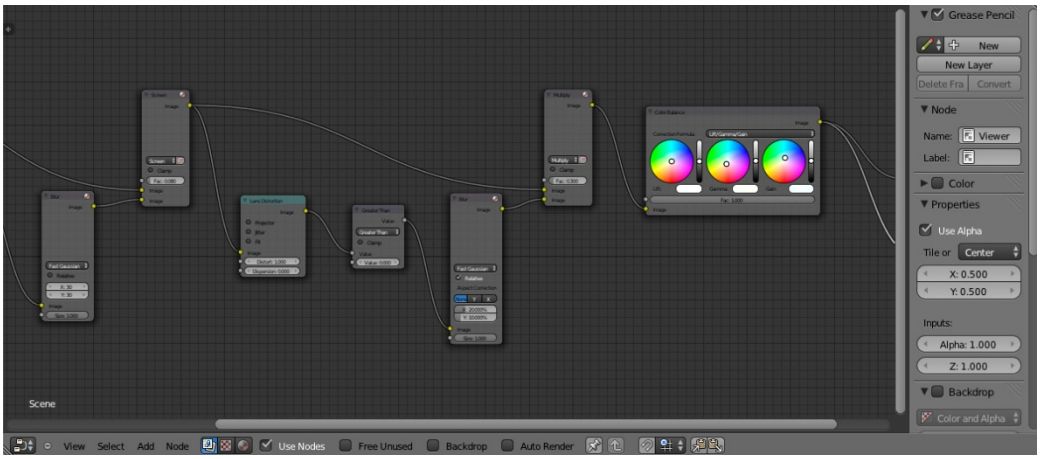


Ilustración 34 –Node Editor

- **Text Editor:** Un pequeño editor de texto donde podemos hacer notas, aclaraciones, etc. Muy útil al compartir ficheros dentro de la comunidad Blender.
- **Movie Clip Editor:** Pequeño editor de clips de video, donde podemos realizar cambios en el mismo antes de introducirlo en el Video Sequence Editor. También lo utilizamos para el rastreo de movimiento de cámara.

- **Video Sequence Editor:** Ventanas que nos permite montar diferentes clips de video para obtener la secuencia final. En el podemos encontrar diferentes canales donde podemos añadir video, audio, transiciones, etc.

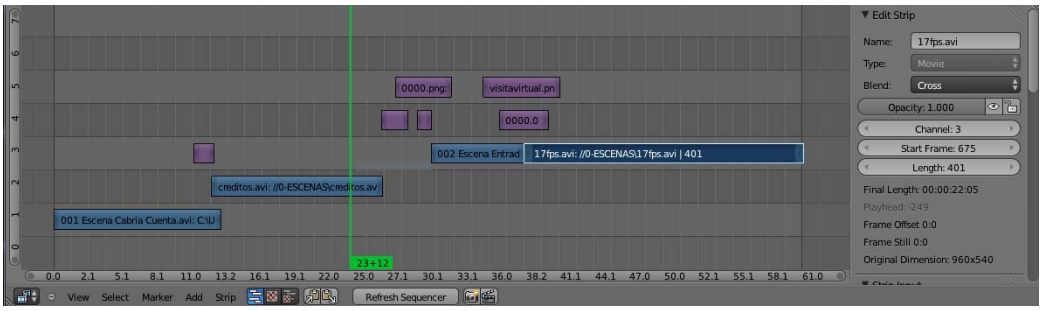


Ilustración 35 –Video Sequence Editor

- **UV/Image editor:** Es el editor de imágenes de Blender. Podemos realizar diferentes cambios en las texturas introducidas en los materiales y aplicar los mapas de texturas UV sobre los objetos [ver X.X.X]. Los resultados de la renderización y del compositor de nodos, también serán mostrados en esta ventana.

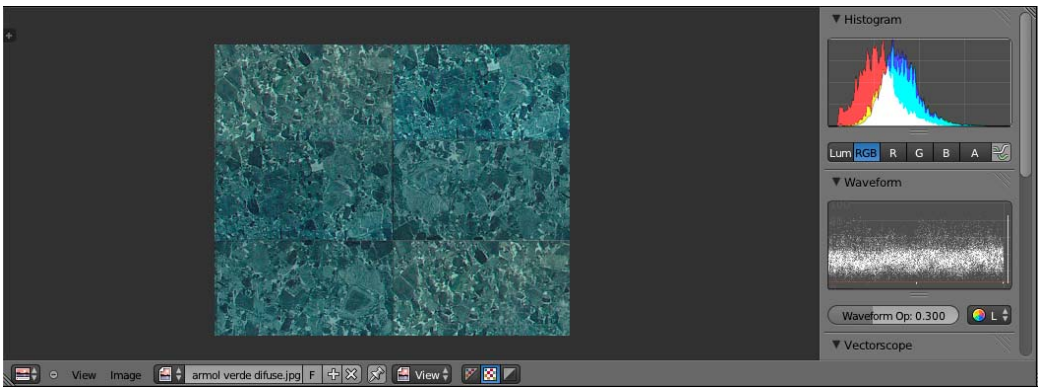


Ilustración 36 –UV/Imagen Editor

- **NLA (No Lineal Animation) Editor:** Se utiliza para mezclar acciones simples dentro de la animación, generando acciones complejas y fluidas. La ventana nos ofrece un panorama general de toda la animación que transcurre en la escena.

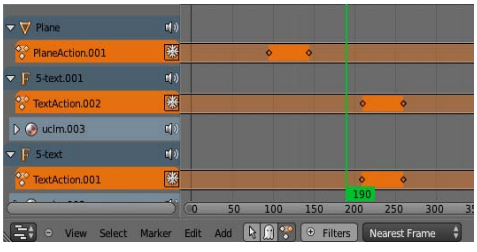


Ilustración 37 –NLA Editor

- **Dope Sheet:** Nos ofrece un panorama general de todas las acciones simples que ocurren durante la animación. Cada acción viene referida desde su frame inicial a su frame final, y separadas por objetos. Desde esta ventana podemos modificar el tiempo de cada acción, escalando, moviendo, etc.

Ilustración 38 –Dope Sheet

- **Graph Editor:** Gráfico que nos indica como varía un parámetro animado de un objeto dentro de la línea temporal de la animación. Una de las cosas más importantes que podemos realizar en esta ventana es contralar la transición inicial y final de cada acción modificando la curvatura del gráfico.

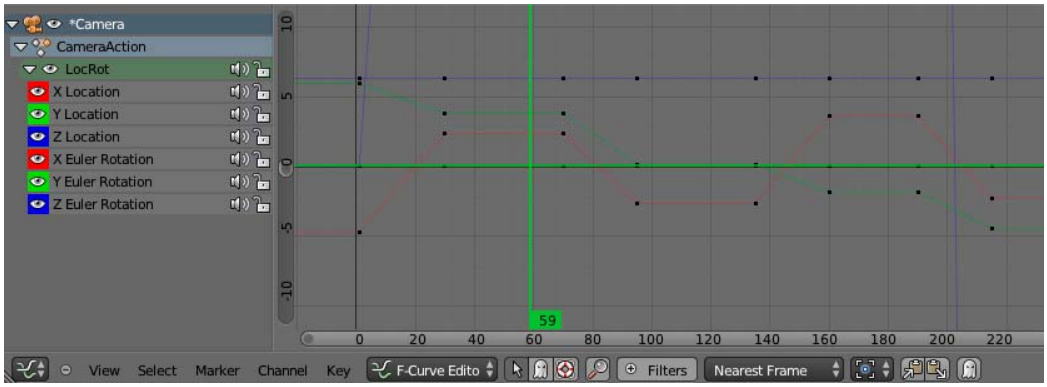


Ilustración 39 –Graph Editor

- **Timeline:** Gráfico general de la animación, donde podemos ver la duración de la misma. En este gráfico vemos señalados los frames claves de cada acción.

Ilustración 40 –Time Line

- **3D view:** La ventana principal de modelado. Es donde previsualizaremos el modelo y realizaremos todas las acciones sobre el mismo.

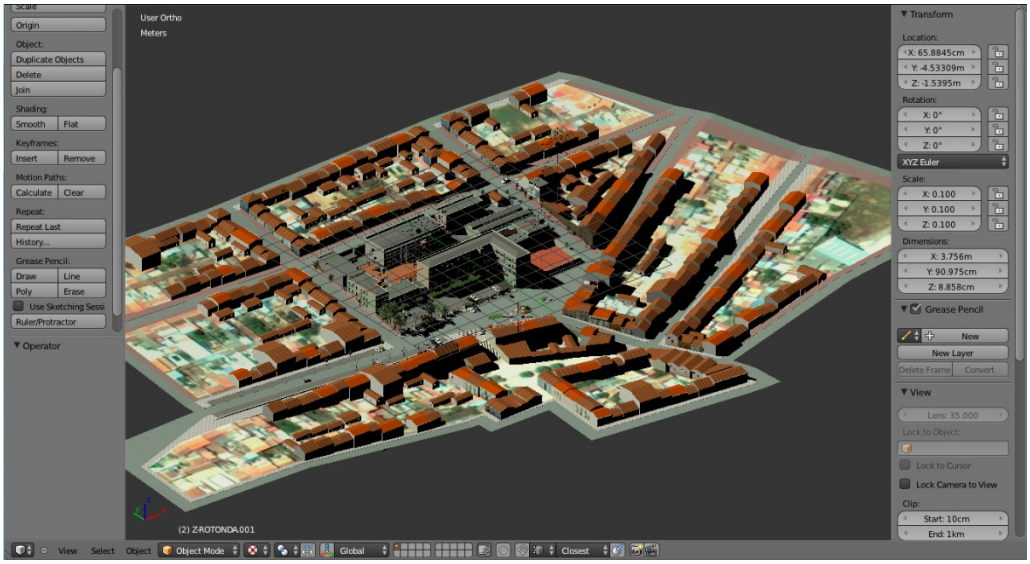


Ilustración 41 –3D view

Cada una de estas ventanas, está compuesta a su vez por cabecera, botones de contexto, paneles y controles.

- **Cabecera:** Cada ventana tiene su propio encabezado en la parte superior o en la parte inferior de la ventana.
- **Botones contextuales:** Funcionan como pestañas, permitiendo el acceso a las opciones. Normalmente están colocadas dentro del encabezado de la ventana.
- **Paneles desplegables:** Opciones agrupadas en menús desplegables.
- **Barras laterales:** Están incluidas en algunas ventanas y agrupan los diferentes paneles y controles de esa ventana. Se pueden ocultar y expandir.
- **Controles:** Permiten modificar una función, opción o un valor.
    - **Botones:** Permiten el acceso a una herramienta.
    - **Casillas de verificación:** Activar o desactivar una opción.
    - **Deslizadores:** Permiten introducir un valor decimal, pueden tener un rango limitado de valores o no.

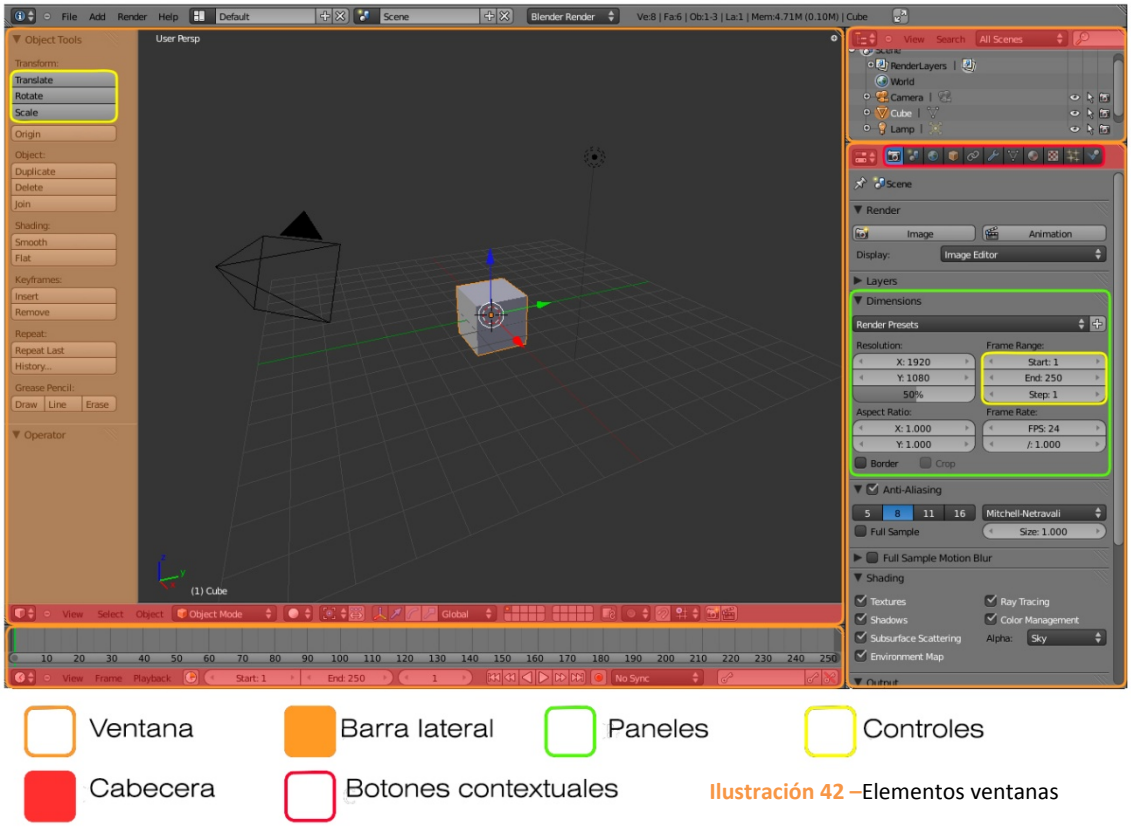


Ilustración 42 –Elementos ventanas

La interfaz de Blender se base en tres principios fundamentales de configuración:

- **No superponer:** La interfaz de usuario le permite ver todas las opciones pertinentes y herramientas de una vez, sin superponer, empujar o arrastrar ventanas por todos lados.

- **No bloquear:** Las herramientas y opciones de la interface no lanza ventanas que requieren que el usuario introduzca datos antes de ejecutar otro comando, de esta forma nunca bloquea ninguna funcionalidad de Blender.

- **No modal:** La entrada de los usuarios deberían ser tan coherente y predecible como sea posible, sin cambiar los métodos de uso general sobre la marcha.

[Blender, 2014] [Flavell, 2010] [Villar, 2014]

3.4.3 Personalizar el espacio de trabajo.

Como hemos visto, según la fase del proyecto que estemos realizando, necesitaremos una **combinación de ventanas** específicas para esa tarea. Blender trae por defecto guardado una serie de combinaciones básicas para cada una, además las podemos modificar a nuestro gusto y guardar para posteriores usos.

Cuando abrimos un archivo nuevo de Blender, nos aparece una **escena por defecto** con tres elementos básicos: Un cubo, una cámara y una luz. Además nos carga la configuración de ventanas **"default"**, donde encontramos las ventanas necesarias para comenzar con el **modelado**. Esta combinación incluye 5 ventanas; en la parte superior la ventana "**info**" (1), en la parte central la ventana "**3Dview**" (2), debajo la ventana "**timeline**" (3), y en la parte derecha la ventana **"outliner"** (4) y **"properties"** (5).

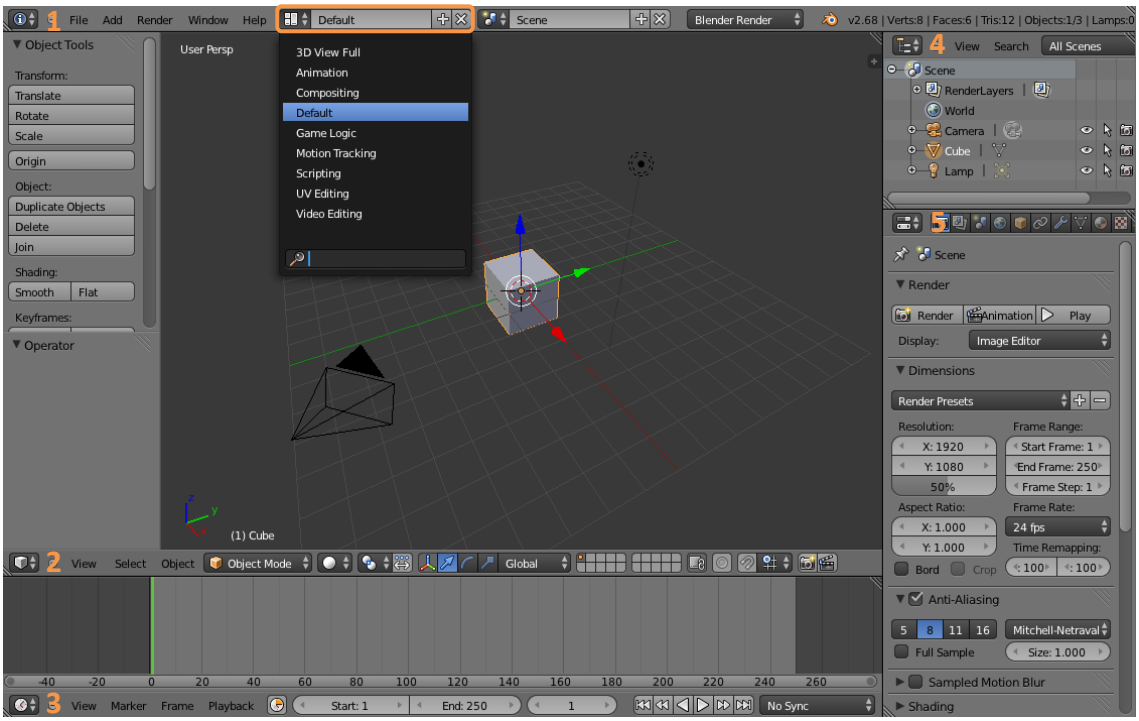


Ilustración 43 – Startup File

En la ventana superior "info" (1), encontramos el menú desplegable **"Choose screen layout"**, donde podemos encontrar otras configuraciones de ventanas, por defecto, para otras tareas.

- **3D view full:** Maximiza a pantalla completa la ventana 3Dview para poder modelar más fácilmente.
- **Animation:** Carga las ventanas con las herramientas necesarias para animación.
- **Compositing:** Configuración de ventanas basada en el editor de nodos para componer y retocar la imagen final del render.
- **Default:** Configuración por defecto optimizada para el modelado.
- **Game Logic:** Carga las ventanas necesarias para programar videojuegos.
- **Motion Tracking:** Utiliza el "video secuence editor" para analizar los movimientos de cámara de un clip de video y poder introducir elementos 3D en el mismo.
- **Scripting:** Configuración que nos carga las ventanas relacionadas con la programación de Script.
- **UV Editing:** Esta configuración de ventanas nos facilita la manipulación de imágenes y texturas. Es óptima para utilizar la técnica de mapeado "UV mapping".
- **Video Editing:** Carga las herramientas del editor de video de Blender.

Las **configuraciones por defecto** están pensadas para cada una de las **tareas** que puede desarrollar el programa, pero son demasiado **específicas** de cada una de ellas. Las fases de creación de un modelo no son tan cerradas, y dependiendo de la forma de trabajar del usuario necesitaremos ir combinando algunas de ellas. Por ejemplo, hay usuarios que realizan el texturizado de forma paralela al modelado, por lo tanto necesitará una **combinación** de ventanas que se ajuste a su método de trabajo.

A continuación vamos a ver cómo podemos manipular y cambiar las ventanas del espacio de trabajo. En primer lugar, podemos respetar la configuración de ventanas actual y **sustituir** una por otra. Para ello utilizaremos el primer icono de la cabecera de la ventana, donde nos aparecerá un menú desplegable con las 17 ventanas.

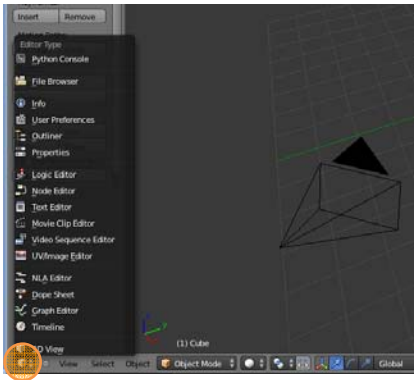


Ilustración 44 – Selección de ventana

Otras veces, nos interesará **incluir una ventana nueva** sin perder ninguna de las que ya tenemos en pantalla. Para ello tendremos que **dividir el espacio del área de trabajo** utilizado por una de ellas. Si nos fijamos bien, en la esquina inferior izquierda y en la esquina superior derecha de cada una de las ventanas, tenemos un pequeño **rayado**. Si cliqueamos sobre este rayado y **arrastramos** hacia el interior de la ventana, vemos como el área de la ventana comienza a dividirse. Esta división funciona tanto horizontal como verticalmente, según nuestras necesidades. Vamos a ver un ejemplo de división del espacio de trabajo, para ello partiremos de la configuración por defecto de Blender, y dividiremos el área destinada a la ventana 3Dview en tres ventanas.

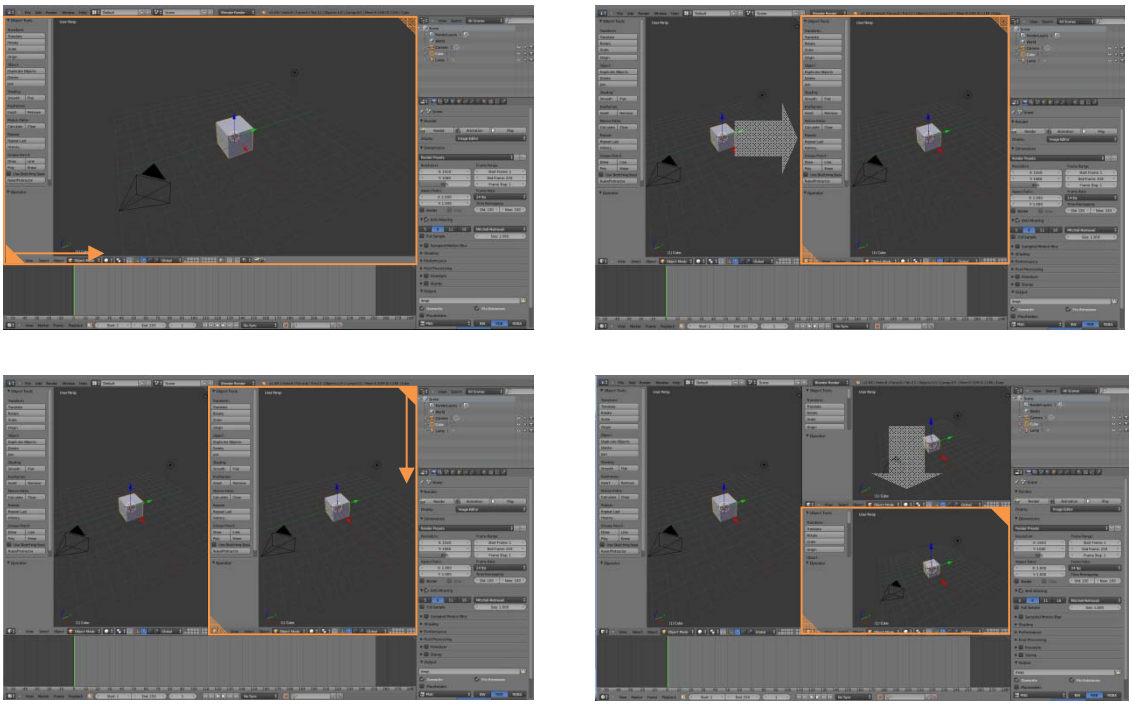


Ilustración 45 – División del espacio de trabajo

Como podemos ver, cada vez que dividimos el espacio de trabajo, se duplica la ventana que estamos modificando. Una vez dividido, siguiendo el proceso anterior, sustituiríamos la ventana duplicada por la nueva ventana.

Si queremos **eliminar** una ventana, solamente tendríamos que realizar el proceso inverso, es decir, arrastrar desde la esquina rayada hacia el exterior de la ventana. De esta forma, la ventana que estamos estirando, sustituirá a la ventana más próxima apropiándose de su área de trabajo. [Hess, 2011] [Flavell, 2010]

3.4.4 Navegación 3D básica.

A continuación, vamos a ver cómo movernos dentro de nuestra escena. Las combinaciones de teclas y botones que veremos, serán las que vienen por defecto en Blender, ya que casi todas son personalizables desde el menú de preferencias de usuario. [Villar, 2014] [Flavell, 2010] [Goás, 2009]

Para movernos en nuestra escena, necesitaremos situarnos en la ventana 3Dview. La navegación básica, al igual que en otros programas CAD la realizaremos con el ratón.

- **Órbita 3D:** La realizaremos dejando pulsado el ***botón central*** del ratón, y al arrastrar nuestro modelo comenzará a orbitar.
- **Desplazamiento:** Utilizando la combinación de teclas ***shift + botón central*** del ratón, podremos movernos por el modelo sin modificar el ángulo de la cámara.
- **Zoom in/out:** Para modificar el nivel de zoom, tendremos que girar la ***rueda central*** del ratón.

En la cabecera de la ventana, encontramos un menú desplegable llamado **view**, desde él, podemos seleccionar diferentes **vistas del modelo**: Planta, izquierda, derecha, frontal...

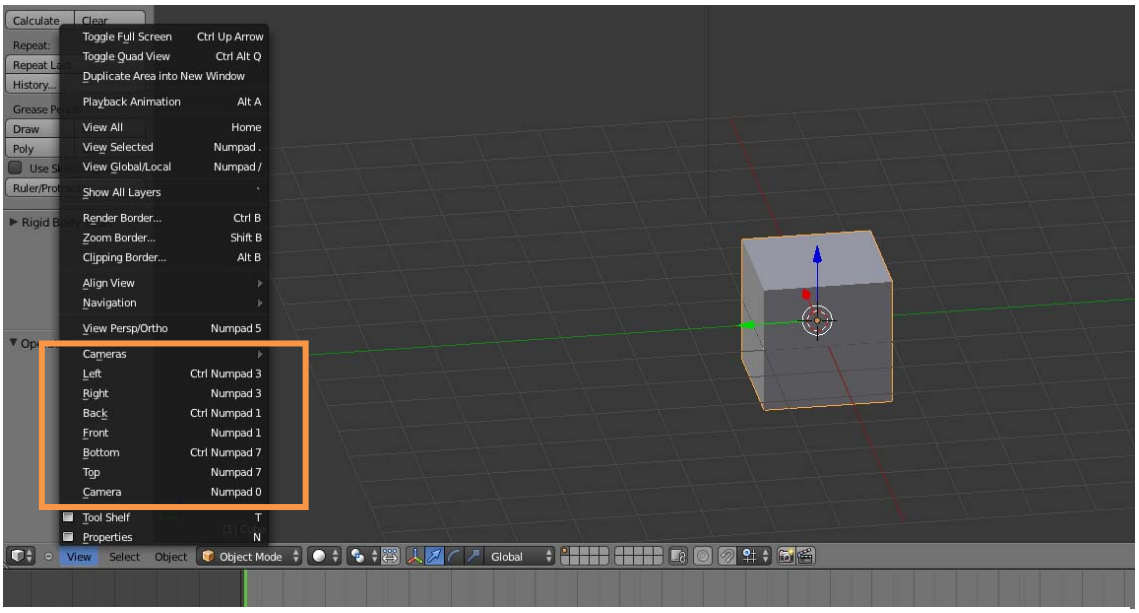


Ilustración 46 – Menú view

Podemos acceder a las mismas vistas, con los **atajos** situados en el bloque numérico del teclado.

TECLAS	VISTA	PLANOS
1	Front	(+Z,+X)
Crt + 1	Back	(+Z,-X)
3	Right	(+Z,+Y)
Crt + 3	Left	(+Z,-Y)
7	Top	(+Y,+X)
Crt + 7	Bottom	(-Y,+X)
0	Cámara Activa	

Ilustración 47 – Teclas rápidas-teclado numérico 1

Además, podemos **manipular** cada una de las vistas anteriores, con las siguientes combinaciones de teclas.

TECLAS	ACCIÓN
5	Perspectiva Ortográfica/Cónica
/	Aísla el objeto seleccionado
.	Centra el objeto seleccionado
8/2	Inclinación de la cámara
4/6	Rotación de la cámara
Crt + 8/2/4/6	Desplazamiento de la cámara
+/-	Zoom in/out

Ilustración 48 – Teclas rápidas-teclado numérico 2

La **cámara activa**, es la cámara que utilizará el programa para realizar el render. Si en nuestra escena tenemos varias cámaras, y queremos cambiar la cámara activa, seleccionaremos la nueva cámara y utilizaremos **Crt+0** para activarla. Si durante la navegación 3D, queremos situar la cámara activa en la posición de la **vista 3D actual**, presionaremos **Crt+Alt+0**.

Por último, vamos a ver los diferentes modos de visualización de nuestra escena. Cuando abrimos Blender, el modo de visualización que viene activo por defecto es **"Solid"**. Para cambiarlo, accederemos al menú desplegable "Viewport Shading" desde la cabecera de la ventana 3Dview. [Goás, 2009] [Flavell, 2010] [Villar, 2014]

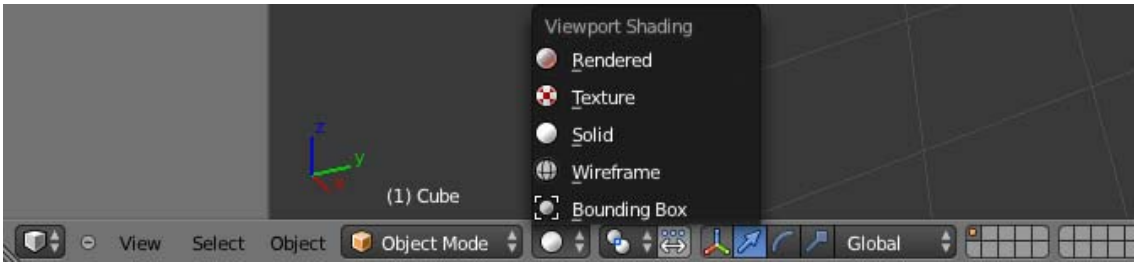


Ilustración 49 – Viewport Shading

- **Bounding Box:** Es el modo de visualización que menos recurso utiliza, ya que no muestra los objetos. Solamente visualizamos la caja de la envolvente del objeto. Es muy útil para ahorra recursos cuando tenemos objetos muy pesados es las escenas.
- **Wireframe:** Este modo, nos muestra todas las líneas de todos los objetos. Es uno de los mejores modos para modelar, ya que vemos todas las aristas del objeto que estamos manipulando.
- **Solid:** Para ver los objetos como sólidos. Las aristas de los objetos quedan ocultas dependiendo de la posición de la cámara. Utiliza un sombreado en el objeto del color base del material que tenga aplicado.
- **Texture:** Al igual que en el modo solid, vemos los objetos como sólidos. Pero podemos ver las texturas de los materiales y las luces de las escenas.
- **Rendered:** Renderiza la vista actual. Está pensado para realizar render en tiempo real con cycles. No es aconsejable utilizarlo con Blender internal, ya que los tiempos de espera pueden ser altos.

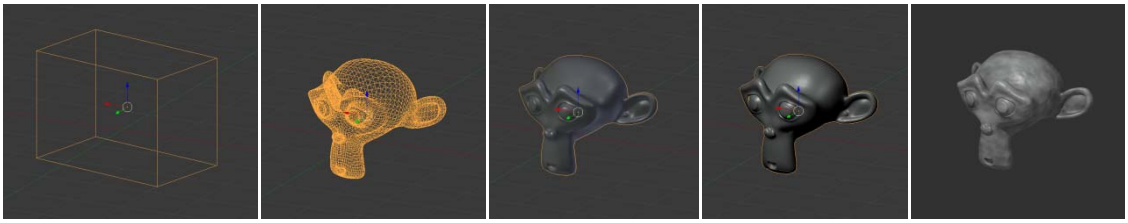


Ilustración 50 – Modos de visualización

3.4.5 Coordenadas locales y coordenadas globales.

Blender utiliza un **sistema de coordenadas** cartesianas para definir la localización y la orientación de cada uno de los objetos de nuestra escena. Dependiendo del punto de **referencia** que estemos utilizando hablaremos de coordenadas **locales** o coordenadas **globales**. El sistema de **coordenadas global**, es **común** a todos los objetos de nuestra escena, mientras que el sistema de **coordenadas locales**, es **único y exclusivo** para cada uno de los objetos. [Morcillo, 2011/12]

Cuando insertamos un nuevo objeto en Blender, este viene definido por su **ecuación implícita o paramétrica** con respecto al **origen de coordenadas local**. Mientras que su posición en el sistema global, queda definido por un **vector de posición** con referencia al **origen de coordenadas global**. [Shirley, 2009] [Theoharis, 2007]

Veamos un ejemplo, imaginemos que tenemos un sistema de coordenadas único, y una esfera de radio 1 unidad con centro situado en el punto (-2;-5;3). La ecuación implícita que define la esfera y su posición sería:

$$(x+2)^2+(y+5)^2+(z-3)^2=1$$

Para simplificar la ecuación, utilizamos el sistema de coordenadas locales. Si situamos el origen de coordenadas locales en el centro de la esfera, la **ecuación implícita de la esfera en coordenadas locales**, queda de la siguiente manera:

$$x'^2+y'^2+z'^2=1$$

Para que la esfera quede situada en su posición correcta, necesitamos un **vector de posición** del tipo (x_0; y_0; z_0), en este caso (-2;-5;3).

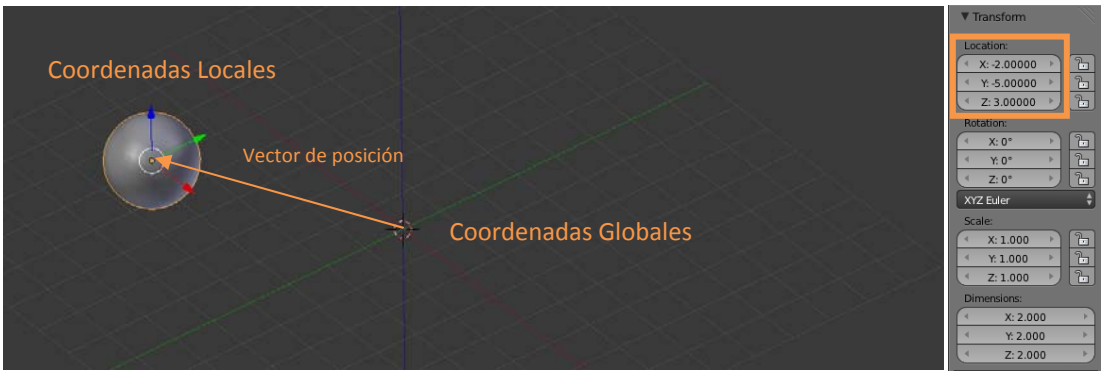


Ilustración 51 – Coordenadas Globales/Locales

Por lo tanto, si aplicamos el vector de desplazamiento a la esfera, tenemos:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} + \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} + \begin{bmatrix} -2 \\ -5 \\ 3 \end{bmatrix} = \begin{bmatrix} x' - 2 \\ y' - 5 \\ z' + 3 \end{bmatrix} \rightarrow \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} x + 2 \\ y + 5 \\ z - 3 \end{bmatrix}$$

Si sustituimos en la ecuación implícita local, obtenemos la **ecuación implícita en coordenadas globales**, que teníamos al principio:

$$(x + 2)^2 + (y + 5)^2 + (z - 3)^2 = 1$$

3.4.6 Añadir nuevos objetos

Blender incluye una serie de **objetos básicos** , primitivas, como cubos, esferas, cilindros, curvas, etc. que son muy útiles para comenzar a modelar. Podemos acceder al menú de estos objetos desde la ventana "info" en el menú desplegable **"Add"** o con la combinación de teclas "**Shift + A".**

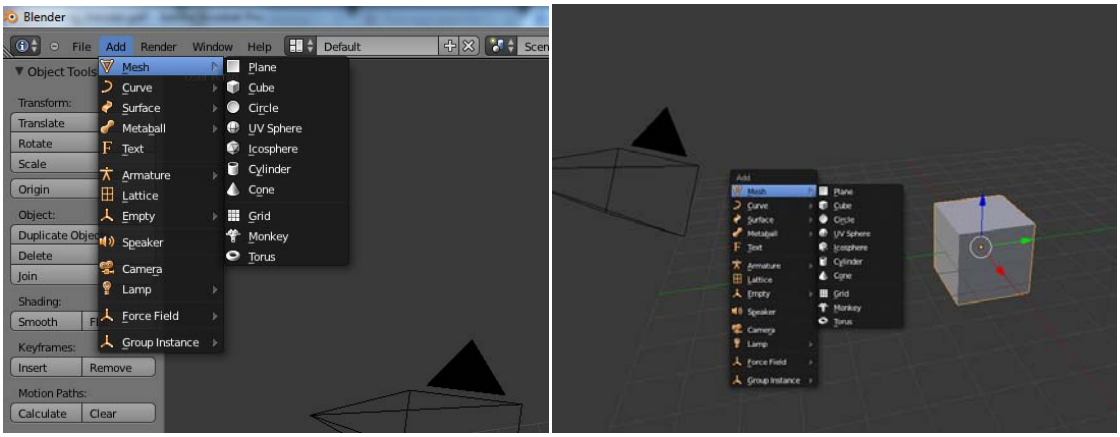


Ilustración 52 – Menú añadir objeto

Podemos añadir una gran variedad de objetos desde este menú, primitivas, luces, cámaras, texto.. Cada uno de estos objetos, tiene un punto de posicionamiento, que será el origen de las **coordenadas locales** del objeto, normalmente situado en su centro de masa o en la base, según la naturaleza del objeto, y que será utilizado como **punto de inserción**. [Goás, 2009] [Moya, 2010]

Cada vez que añadimos un objeto nuevo, será insertado en la posición del **cursor 3D**. Para mover el cursor 3D, tendremos que hacer **clic izquierdo** con el ratón en el lugar que queremos o especificar sus **coordenadas** en la barra lateral derecha de la ventana 3D view. Para volver a colocarlo en el **origen global**, utilizaremos las teclas "**Shift+C**".

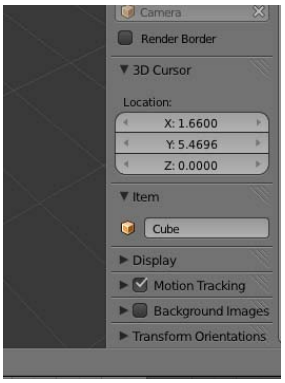


Ilustración 53 – 3D Cursor

La **barra lateral derecha** podemos desplegarla utilizando la tecla **N** o pulsando sobre la cruz que encontramos en dicho lateral. En esta barra, encontraremos opciones sobre la posición del objeto, sus coordenadas y otras opciones sobre visualización.

Al insertar el **nuevo objeto** en la escena, aparece con unas dimensiones y **propiedades determinadas**. En la barra lateral izquierda nos aparece un nuevo menú, donde podemos modificar el nuevo objeto. La **barra lateral izquierda** podemos desplegarla utilizando la tecla **T** o pulsando igualmente la cruz en dicho lateral. En esta barra encontraremos algunas herramientas básica de edición de objetos, como localización, rotación, subdivisiones, dimensiones, etc. [Hess, 2011] [Flavell, 2010] [Villar, 2014]

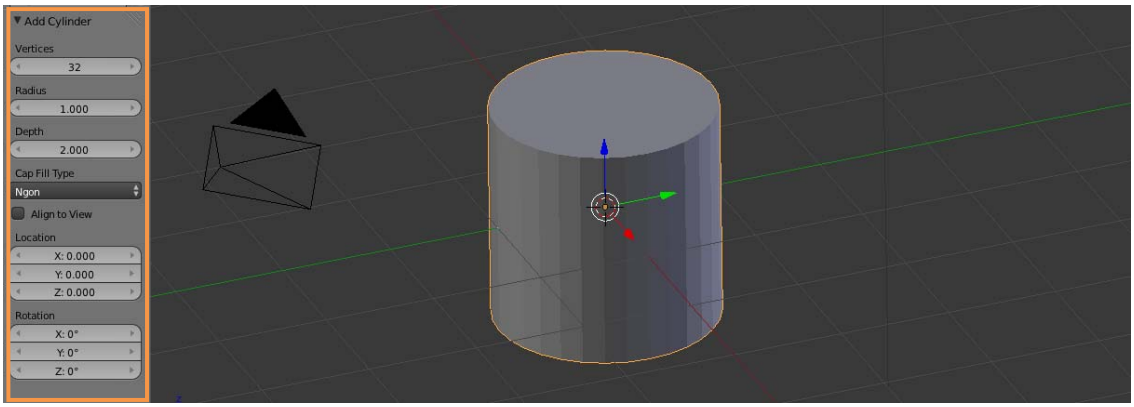


Ilustración 54 – Menú edición objeto nuevo - Cilindro

3.6.7 Transformaciones básicas: Mover, rotar y escalar.

Lo primero que tenemos que tener en cuenta, a la hora de aplicar las transformaciones básicas, es en qué sistema de referencia queremos aplicarlas. Si aplicamos cualquiera de estas transformaciones en las **coordenadas locales**, estaremos modificando la **ecuación del objeto**. Mientras que si las aplicamos en **coordenadas globales**, estaremos modificando, el **vector de posición**, la **matriz de rotación** o la **matriz de escala**. [Hess, 2011] [Flavell, 2010] [Roosendaal, 2004]

Para cambiar de un sistema a otro, utilizaremos el menú desplegable que encontraremos en la cabecera de la ventana 3d view. En el cual, podremos cambiar entre **Modo Objeto y Modo Edición**. De manera que, en modo edición, estaremos modificando la ecuación del objeto, ya que manipulamos sus puntos con respecto al origen de coordenadas locales. Mientras que en modo objeto, estaremos modificando los parámetros de localización. Podemos cambiar de un modo a otro utilizando la tecla rápida "Tabulador".

Ilustración 55 – Edit Mode / Object Mode

Tenemos dos formas de realizar cualquiera de las tres operaciones básicas, tanto en modo objeto como modo edición. Podemos utilizar los **manipuladores** disponibles en la cabecera de la 3d view. Una vez seleccionado el modificador, clicamos con el botón en izquierdo del ratón en el **eje** donde queremos realizar la operación. [Moya, 2010]

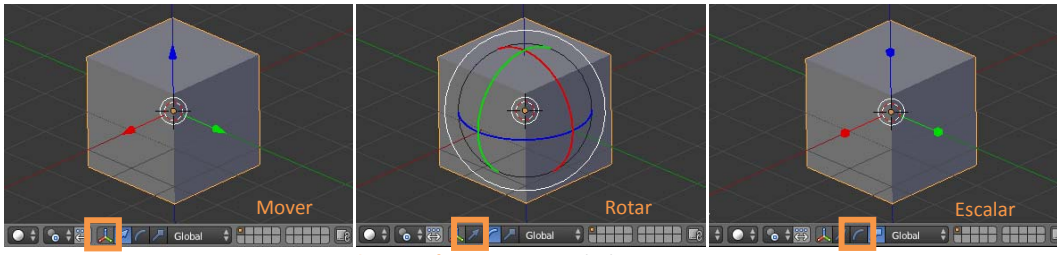


Ilustración 56 – Manipuladores

Una vez hecho esto, nuestro objeto comenzara a transformarse. Si queremos realizar la operación de una manera exacta, tendremos que introducir la distancia, el ángulo o el factor de escala con el teclado, sin soltar el botón del ratón.

También podemos utilizar las teclas rápidas, **G** para mover, **R** para rotar y **S** para escalar. Una vez pulsada la tecla, podemos restringir el eje utilizando las teclas **X**, **Y** y **Z**. Y a continuación elegimos la distancia, el ángulo o el factor de escala, de forma dinámica en la pantalla o mediante la **introducción de un valor numérico** por el teclado.

Los manipuladores, como vemos, se pueden **restringir** a un determinado eje del sistema de coordenadas, tanto global como local, para ello, elegiremos el **sistema de coordenadas** que más interese en el menú desplegable que hay al lado de los manipuladores. [Goás, 2009] [Hess, 2011] [Flavell, 2010]

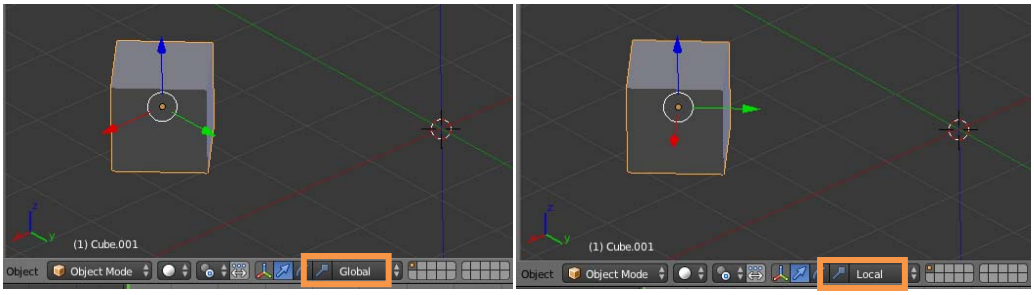


Ilustración 57 – Manipuladores Global/Local

Las transformaciones básicas, utilizan un **punto de referencia** "pivote" para aplicarse, es decir, necesitamos un centro de giro para la rotación o un punto fijo en el escalado. Por defecto, utiliza el origen de coordenadas locales del objeto. Podemos cambiar el pivote, bien moviendo el origen de las coordenadas locales, o utilizando el cursor 3D. Podemos cambiar esta opción en el menú desplegable "**Pivot Center**", en la cabecera de la ventana 3D view. [Moya, 2010]

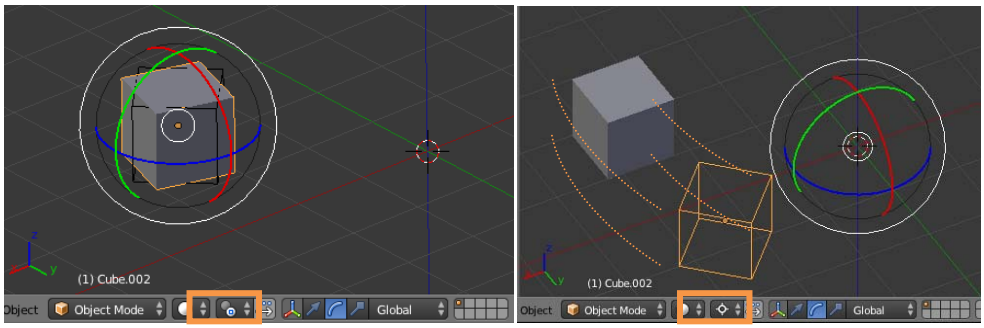


Ilustración 58 – Pivot Center

3.6.7.1 Mover.

Como vimos anteriormente, la posición de un objeto viene definida por un **vector de posición**, que nos indica la posición del origen de coordenadas locales con respecto el origen de coordenadas globales.

Cuando aplicamos un **desplazamiento** a nuestro objeto, **modificamos** su posición en el espacio y por tanto el **vector de posición**. Si tenemos un objeto situado en el punto $V_0(x_0,y_0,z_0)$ y le aplicamos un desplazamiento u(x,y,z), el nuevo vector de posición $V_1(x_1,y_1,z_1)$ será la suma de los dos anteriores. [Suffern, 2007]

$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} + \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

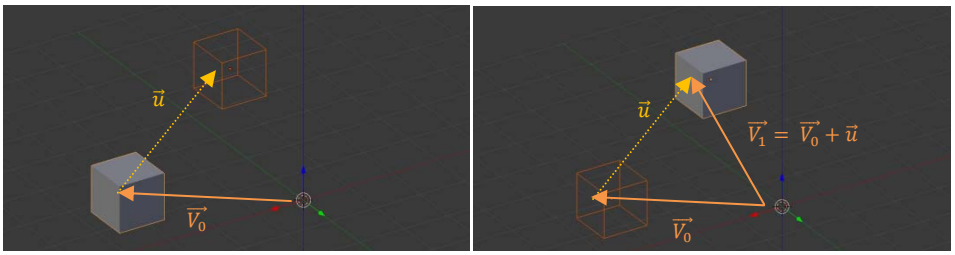


Ilustración 59 – Vectores de desplazamiento

3.6.7.2 Escalar.

Cuando aplicamos un factor de escala a un objeto, este **factor de escala** queda almacenado en la **matriz de escala.** Por lo tanto lo que estamos haciendo es multiplicar las variables de las coordenadas locales del objeto por la siguiente matriz.

$$scale(s_x, s_y, s_z) = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix}$$

Donde s_x es el factor de escala aplicado en el eje x, s_y es el factor de escala aplicado en el eje y, y s_z es el factor de escala aplicado en el eje z. [Shirley, 2009] [Theoharis, 2007]

Cuando lo multiplicamos por las variables locales del objeto, obtenemos el siguiente vector:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} x \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix} = \begin{bmatrix} x's_x \\ y's_y \\ z's_z \end{bmatrix}$$

Este vector, nos dará las coordenadas globales finales de los puntos del objeto.

$$\begin{bmatrix} x \\ y \\ z' \end{bmatrix} = \begin{bmatrix} x's_x \\ y's_y \\ z's_z \end{bmatrix}$$

En Blender, podemos ver los valores de la diagonal principal de la matriz escala, s_x, s_y y s_z en la barra lateral derecha de la ventana 3d view, en el apartado "Transform".

En el siguiente ejemplo vemos un cubo al cual le hemos aplicado un matriz escala con los siguientes valores: s_x=0.5, s_y=1 y s_z=2.

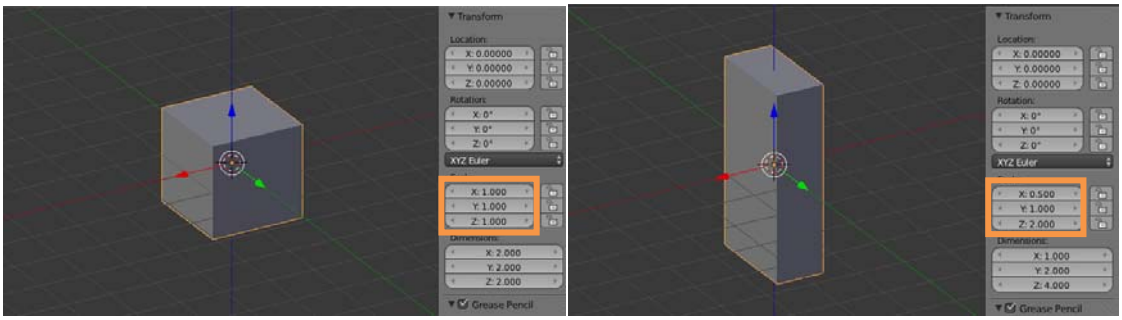


Ilustración 60 – Factores de escala

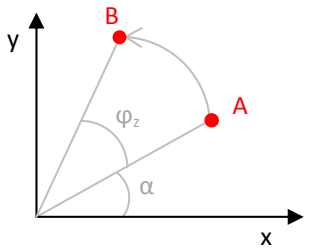
3.6.7.3 Rotar.

Cuando **rotamos** un objeto un ángulo determinado, lo hacemos en referencia a los tres ejes de coordenadas. Por lo tanto tendremos una **matriz de rotación** por cada eje.

A continuación vamos a deducir la matriz de rotación en el eje z. Para ello imaginaremos que queremos girar un punto A(x_a,y_a,z_a) un determinado ángulo φ_z hasta la posición B(x_b,y_b,z_b). Lo primero que tenemos que tener en cuenta es que, como estamos girando respecto al eje z, los puntos no cambiaran su coordenada z; por lo tanto z_a=z_b. Por trigonometría sabemos que la distancia desde el punto A, al centro de giro es $r = x_a^2 + y_a^2$; y por geometría esta distancia tiene que ser la misma desde el centro de giro al punto B. Por lo tanto tenemos: [Shirley, 2009] [Theoharis, 2007]

$$x_a = r \cdot \cos\alpha$$

$$y_a = r \cdot \sin\alpha$$



Y por trigonometría sabemos que:

$$x_b = r \cdot \cos(\alpha + \varphi_z) = r\cos\alpha\cos\varphi_z - r\sin\alpha\sin\varphi_z$$

$$y_b = r \cdot \sin(\alpha + \varphi_z) = r\sin\alpha\cos\varphi_z + r\cos\alpha\sin\varphi_z$$

Si sustituimos $x_a = r \cdot \cos\alpha$; $y_a = r \cdot \sin\alpha$:

$$x_b = x_a\cos\varphi_z - y_a\sin\varphi_z$$

$$y_b = y_a\cos\varphi_z + x_a\sin\varphi_z$$

Por lo tanto en forma matricial, y teniendo en cuanta que z_a=z_b quedaría:

$$\begin{bmatrix} x_a \\ y_a \\ z_a \end{bmatrix} x \begin{bmatrix} \cos\varphi_z & -\sin\varphi_z & 0 \\ \sin\varphi_z & \cos\varphi_z & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix}$$

Por lo tanto las matrices de rotación para cada eje serán:

En el eje Z: $$rotate(\varphi_z) = \begin{bmatrix} \cos\varphi_z & -\sin\varphi_z & 0 \\ \sin\varphi_z & \cos\varphi_z & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

En el eje X: $$rotate(\varphi_x) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi_x & -\sin\varphi_x \\ 0 & \sin\varphi_x & \cos\varphi_x \end{bmatrix}$$

En el eje Y: $$rotate(\varphi_y) = \begin{bmatrix} \cos\varphi_y & 0 & \sin\varphi_y \\ 0 & 1 & 0 \\ -\sin\varphi_y & 0 & \cos\varphi_y \end{bmatrix}$$

En la barra lateral derecha, el programa no nos muestra la matriz de rotación, nos muestra directamente el valor del ángulo girado en cada eje. Si queremos obtener la matriz de rotación, tendríamos que hacer un script en Python con el siguiente código: [Molón, 2014]

Scene= bge.logic.getCurrentScene()
ObjList= Scene.objects
*nombre matriz= ObjList['nombre del objeto'].**worldorientation***
print(nombre matriz)

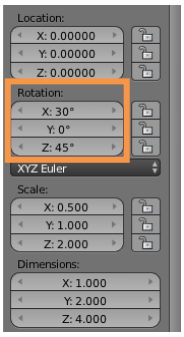


Ilustración 61
Ángulos de rotación

3.6.8 Render. Algoritmos de representación.

Durante el proceso de render, tratamos de convertir la descripción de una **escena 3D** en una **imagen 2D.** El resultado será un mapa de bits que representa el entorno 3D, por lo tanto durante la fase de cálculo, se determinará el **color** más apropiado que se asigna a cada **pixel** de la escena.

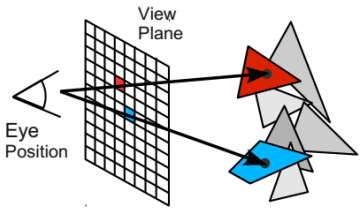


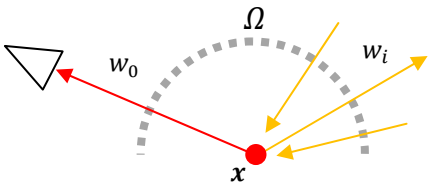
Ilustración 62
Emisión de rayos

[Shirley, 2009] [Theoharis, 2007]

Todos los métodos de cálculo pretenden dar solución a la ecuación de James Kajiya:

$$L_0(\boldsymbol{x}, \overrightarrow{w_0}) = L_e(\boldsymbol{x}, \overrightarrow{w_0}) + \int_{\Omega} f_r(\boldsymbol{x}, \overrightarrow{w_i}, \overrightarrow{w_0}) \cdot L_i(\boldsymbol{x}, \overrightarrow{w_i}) \cdot (\overrightarrow{w_i} \cdot \overrightarrow{\boldsymbol{n}})\, d\overrightarrow{w_i}$$

Donde:



$\boldsymbol{x}$ es la localización de un punto en el espacio.
w_0 es la dirección de salida de la luz hacia el observador.
w_i es la dirección inversa de la luz emitida.
L_0 es la radiación que llega al observador desde $\boldsymbol{x}$.
L_e es la radiación total emitida.
Ω es el hemisferio que contiene todos valores posibles de w_i.
f_r función de distribución de reflectancia bidireccional.
L_i es la radiación que llega a $\boldsymbol{x}$ en la dirección w_i.
$w_i \cdot \boldsymbol{n}$ es el factor de debilitamiento de la radiación debido al ángulo de incidencia.

Según el **modelo de iluminación** utilizado, distinguimos dos modelos de cálculo:

Iluminación Local: Solamente considera la **iluminación directa** desde las fuentes de luz, es el modelo de cálculo utilizado por el motor interno de Blender. Traza una línea desde la cámara a cada pixel de la escena, donde el color final será el color del objeto más cercano en esa dirección más la iluminación directa. Con este modelo de cálculo, las reflexiones y refracciones no consiguen un verdadero efecto de realismo.

Iluminación Global: Además de calcular la **iluminación directa**, se tiene en cuenta la reflexión y la refracción de la luz. La luz "rebota" entre los objetos consiguiendo un efecto de **iluminación ambiental**. Es el método de cálculo utilizado por el motor de Blender Cycles. [Morcillo, 2011/12]

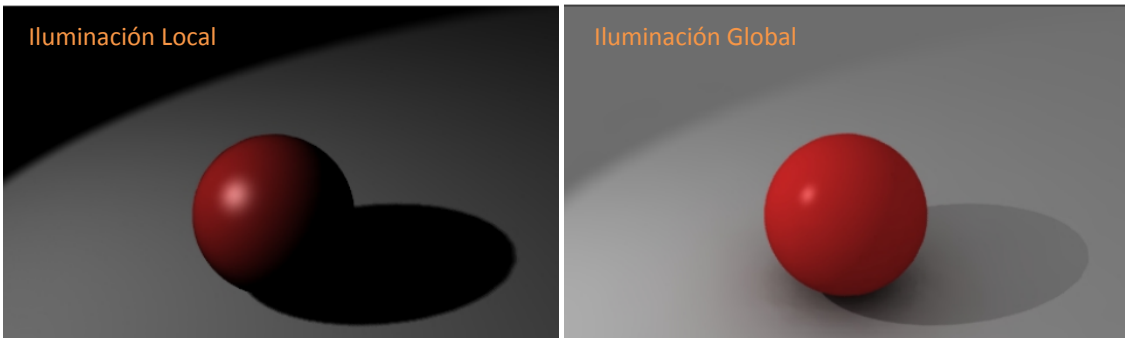


Ilustración 63 – Iluminación Local/Global

3.6.8.1 Trazado de rayos. Raytracing

El cálculo anteriormente descrito, es realizado mediante el método **Raytracing**, desarrollado en 1980 por Whitted. Genera un **rayo por cada pixel** de la imagen y calcula todos los rayos que llegan a la cámara. Cada rayo es una línea imaginaria que recorre nuestra escena **recopilando información** para la imagen final. Tenemos 4 tipos de rayos dependiendo de las circunstancias: [Morcillo, 2011/12]

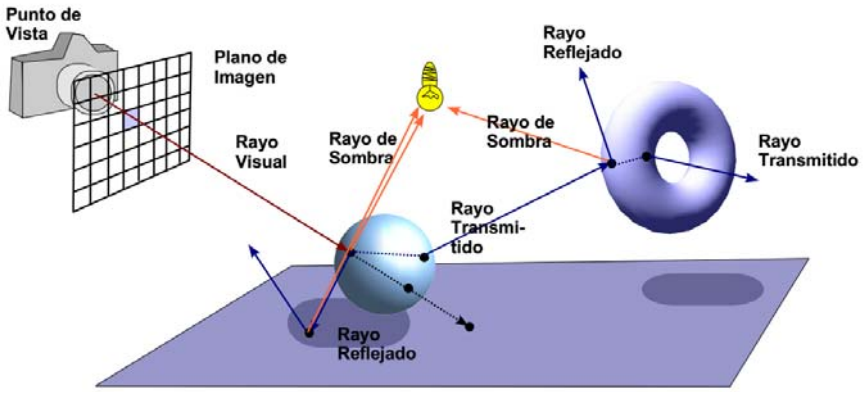


Ilustración 64 – RayTracing

Rayos visuales: Comprueba si el rayo toca a cada objeto de las escena. El resultado final será el punto del objeto más cercano a la cámara, de esta manera se establecen los objetos vistos y ocultos de nuestra escena.

Rayos de sombras: Comprueba las intersecciones entre los objetos y las luces, si hay objetos que choquen con estos rayos producirán sombras.

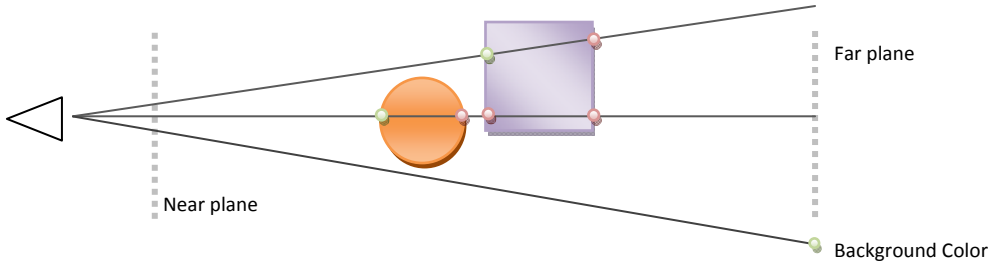
Rayos reflejados: Calcula la luz que alcanza un punto de reflexión. Obtendremos diferentes resultados dependiendo del tipo de reflexión del material.

Rayos transmitidos: Generados por los objetos transparentes.

3.6.8.2 Superficies Ocultas. Z-Buffer

Existen muchos métodos para determinar que partes de las escena es visible desde una posición determinada, algunos de ellos son, z-buffer, línea de rastreo, algoritmo del pintor, emisión de rayos, detección de la cara posterior, etc.

El método **z-buffer** o buffer de **profundidad**, es uno de los más utilizados. Se encarga de gestionar las coordenadas de profundidad de los objetos, las compara, y decide que objeto es visible en cada uno de los pixeles. [Suffern, 2007] [Shirley, 2009]



Por defecto se asigna un valor para z-buffer igual a la distancia del plano más lejano de la cámara, que tendrá el color del fondo de nuestra escena.

```
z-buffer(x,y) = max depth; and
COLOR(x,y) = background color.
```

Para cada pixel, calculará la intersección del rayo con los objetos de la escena, comparará la profundidad y almacenará en el buffer la menor de ellas, de la que finalmente obtendrá el color del correspondiente para cada pixel.

```
for(each polygon P in the polygon list)
        for(each pixel(x,y) that intersects P)
                Calculate z-depth of P at (x,y)
                If (z-depth < z-buffer[x,y])
                        z-buffer[x,y]=z-depth;
                        COLOR(x,y)=Intensity of P at(x,y);
```

3.6.8.3 Aliasing / Antialiasing

Una vez terminado el cálculo Raytracing, obtendremos una imagen formada por un mapa de bits, con la información del color de cada pixel. Pero la **resolución** de la imagen es **finita**, por lo que podemos encontrar **problemas con la definición** de curvas, objetos pequeños o muy lejanos, etc. Este fenómeno se conoce como aliasing.

[Shirley, 2009] [Theoharis, 2007]

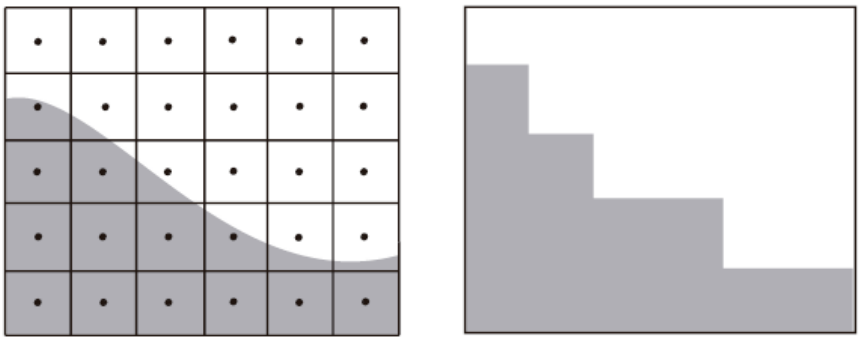


Ilustración 65 – Aliasing

Existen muchos métodos para eliminar este efecto [Suffern, 2007]. Normalmente, los métodos de antialaising realizan un **muestreo** de los pixeles y añaden pequeñas **variaciones de color**. Estas variaciones de color añaden un **ruido** en los bordes de los objetos que suavizan en gran parte el aliasing, dependiendo de la resolución de nuestra imagen y del tamaño del muestreo.

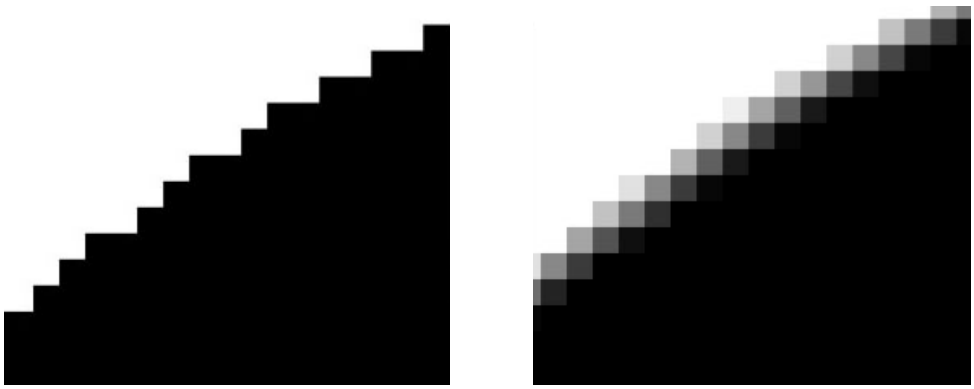
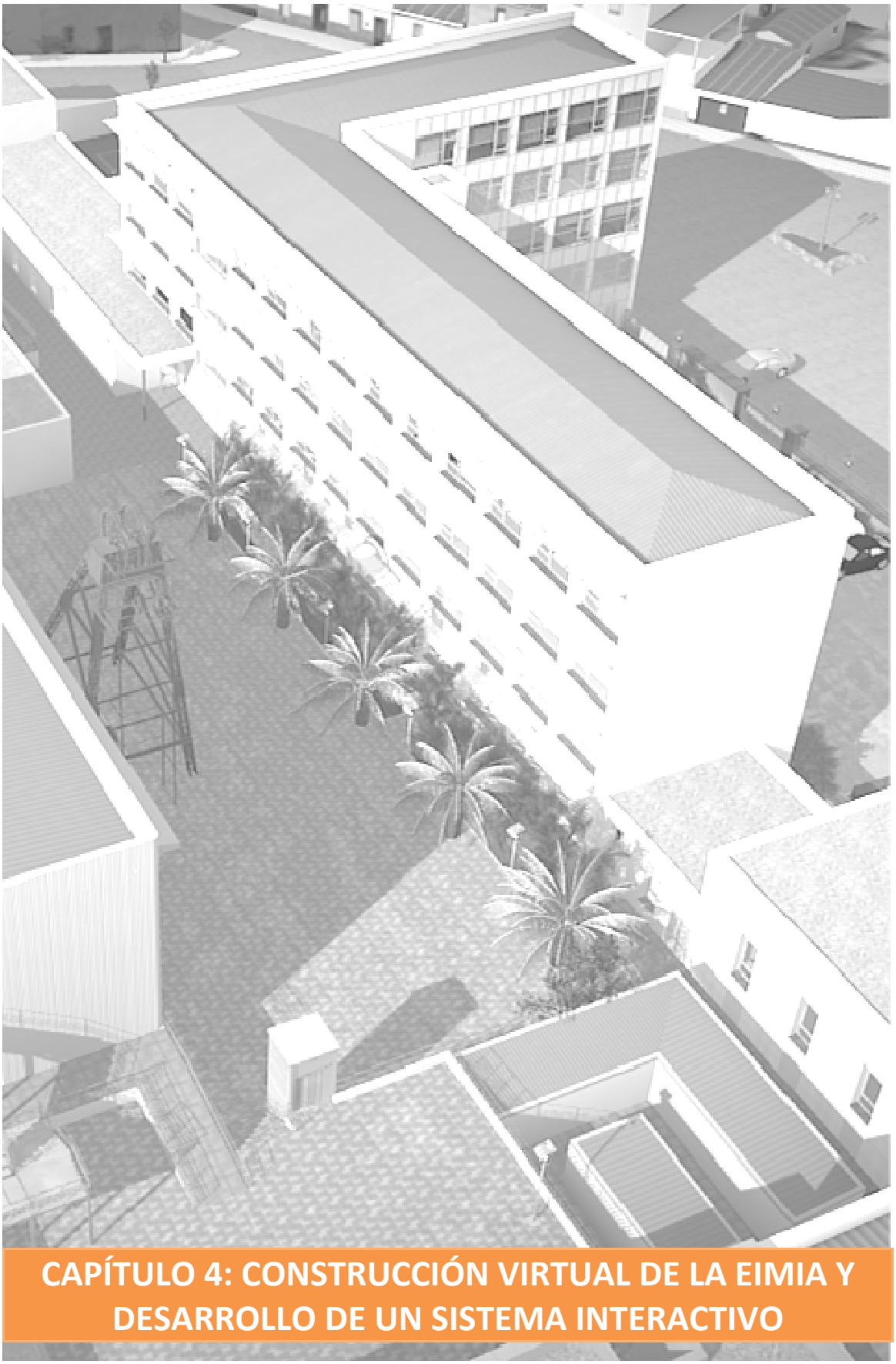


Ilustración 66 – Aliasing / Antialiasing



CAPÍTULO 4: CONSTRUCCIÓN VIRTUAL DE LA EIMIA Y DESARROLLO DE UN SISTEMA INTERACTIVO

CAPÍTULO 4: CONSTRUCCIÓN VIRTUAL DE LA EIMIA Y DESARROLLO DE UN SISTEMA INTERACTIVO

A lo largo de este capítulo, se explicarán las diferentes fases seguidas a lo largo del proyecto. En cada una, veremos las herramientas principales de cada una de ellas y cómo se han ido aplicando en el proyecto. Las fases están expuestas en un orden lógico, aunque como muchas de ellas no son fases cerradas, el orden dependerá mucho de la metodología de trabajo de cada uno. Ya que, por ejemplo, hay quien prefiere hacer la iluminación de la escena antes de texturizar, o texturizar mientras realiza el modelado.

Las fases del trabajo son las siguientes:

Modelado virtual:

- Análisis de información.
- Modelado.
- Creación y asignación de materiales.
- Fotografías de texturas y Texturizado.
- Iluminación exterior e interior.

Video de presentación:

- Guión.
- Animación Objetos y Cámaras.
- Configuración de salida de la escena.
- Postproducción y montaje.

Desarrollo de un sistema interactivo:

- Configuración del personaje/cámara.
- Configuración de la interfaz.
- Optimización del sistema interactivo.

MODELADO VIRTUAL

4.1 ANÁLISIS DE INFORMACIÓN

Antes de comenzar con el proyecto, necesitamos recopilar **toda la información posible** sobre los objetos que vamos a modelar. Esta fase es muy importante en un proyecto, ya que un error o una mala información en el punto de partida, puede suponer una gran pérdida de tiempo e incluso tener que volver a empezar desde cero.

En este caso, nos encontramos con un proyecto de una envergadura muy grande, ya que el recinto de la universidad ocupa una manzana de 8.700 m^2, está compuesto por varios edificios y tiene una topografía complicada con grande cambios de nivel.

Por tratarse de un **modelo arquitectónico** ya existente, necesitamos conocer todos los detalles y que todo esté perfectamente definido. Por eso, la información más importante que tenemos que obtener es una buena **planimetría**, ya que será nuestra "plantilla" a la hora de comenzar a modelar. Además necesitaremos **fotografías** de los exteriores e interiores de los edificios, que nos aportaran información adicional y serán muy útiles durante la fase de modelado.

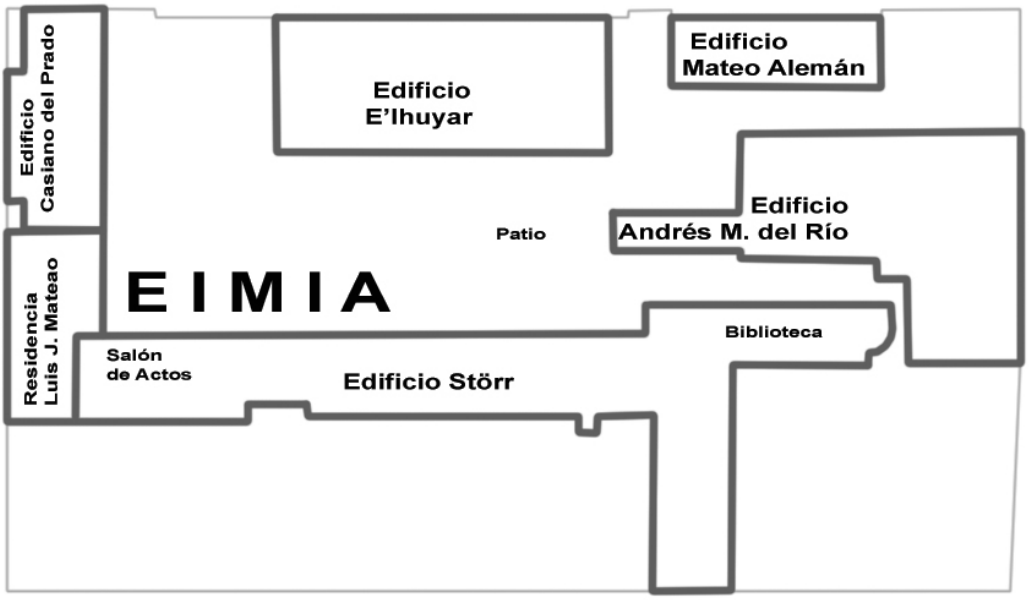


Ilustración 67 – Distribución de Edificios de la EIMIA

4.1.1 Planimetría.

En primer lugar, necesitábamos conseguir la planimetría existente de cada uno de los edificios de la universidad en formato CAD.

El primer problema que encontramos, es que no existía una planimetría actual del conjunto, es decir, teníamos todos los edificios por separado. Además, de algunos de los edificios teníamos varias planimetrías de las diferentes reformas que habían sufrido.

Por lo tanto, el primer paso fue **unificar las planimetrías** antiguas con las planimetrías de las reformas, y conseguir una única planimetría lo más actual posible.

Esto supuso un gran problema en el edificio Störr, ya que existían diferencias importantes entre las dimensiones de las planta originales y las de la reforma. Por lo fue preciso realizar **mediciones sobre el edificio** para poder unificarlas.

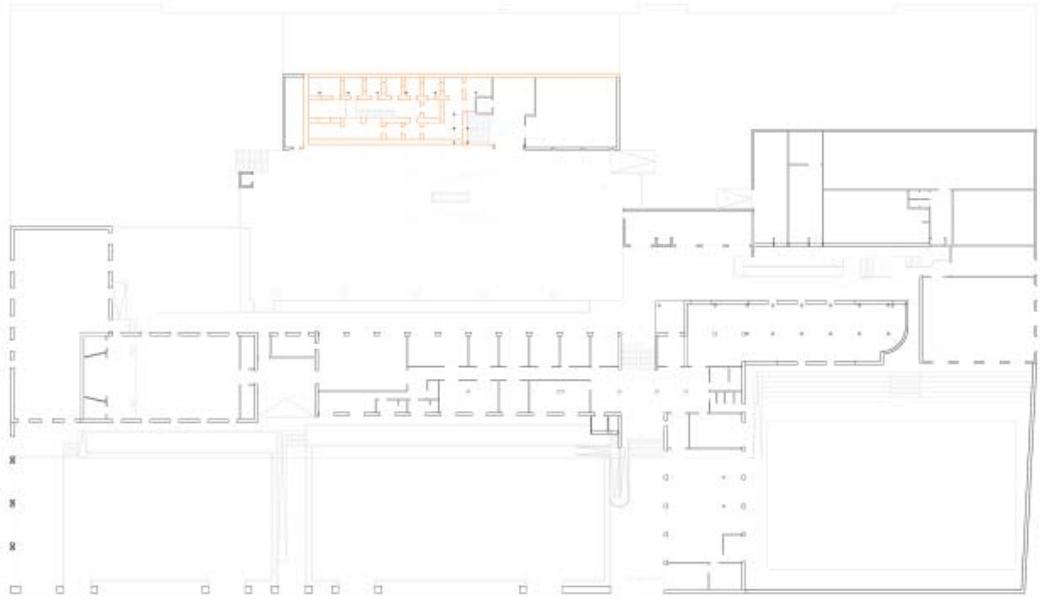
También encontramos problemas con la planimetría del edificio E'lhuyar, ya que los planos del proyecto que teníamos, no se ajustaban al proyecto final ejecutado.

Solventados estos problemas, se unificó la planimetría de cada edificio sobre uno de los planos de situación. Una vez situados todos los edificios en este nuevo plano, lo único que faltaba por dibujar era el patio interior.

Con todo esto conseguimos una planimetría bastante completa del conjunto. El siguiente paso, fue **comprobar las distribuciones** interiores de todos los edificios ya que algunas habían sufrido algunas modificaciones posteriores.

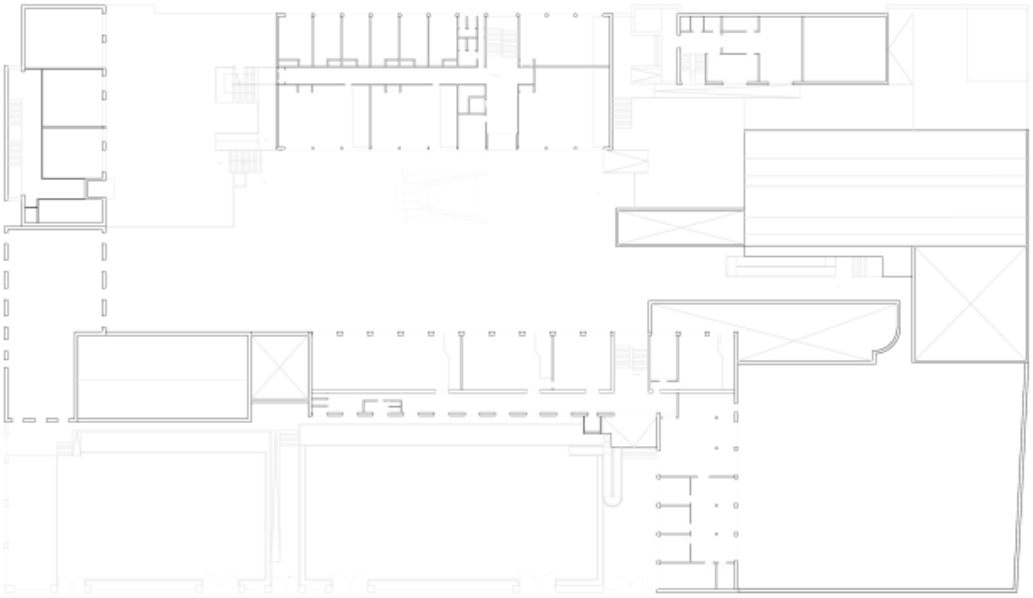
Por último, necesitábamos realizar un **estudio de los desniveles** del terreno y de los **niveles de los forjados**. Este punto era muy importante para poder modelar los edificios de manera más o menos independiente y que al final, pudieran encajar perfectamente en el conjunto.

Una vez que teníamos la planimetría del conjunto completa y bien estudiada, pasamos a dibujar las plantillas para importarlas a Blender. Para ello redibujamos toda la planimetría solamente con la información necesaria para el modelado.

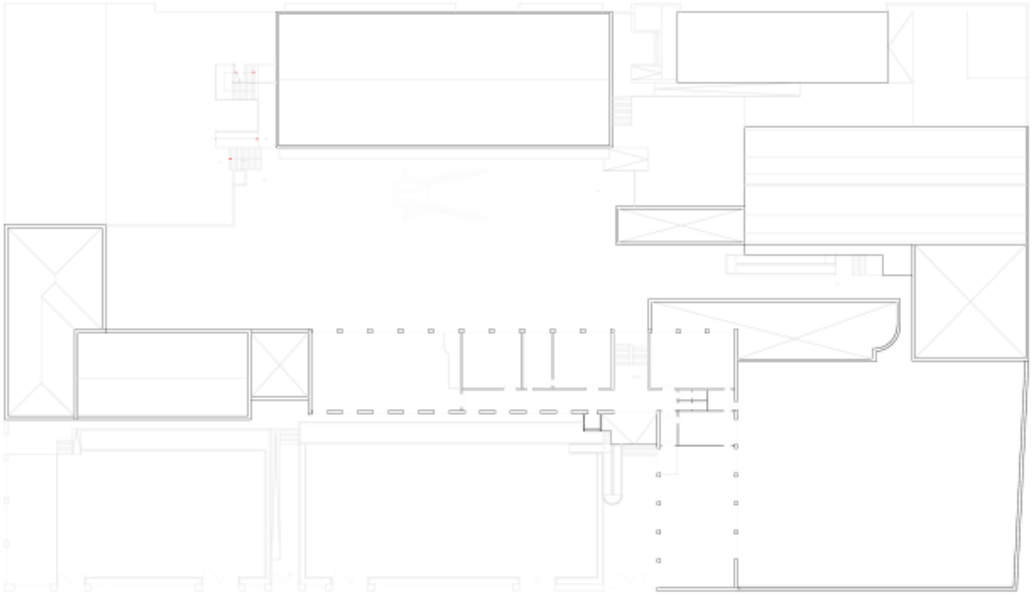


PLANTA BAJA

PLANTA PRIMERA



PLANTA SEGUNDA



PLANTA TERCERA

Ilustración 68 – Planimetría EIMIA

4.1.2 Importar las plantillas

Lo primero que tenemos que hacer es preparar las plantillas para su importación. Para ello realizaremos **una plantilla para cada planta** o nivel, por lo que tendremos un **archivo diferente** para cada una de ellas.

Cuando Blender importa un archivo CAD, utiliza como punto de inserción el origen de coordenadas de nuestro archivo CAD. Por lo tanto necesitamos **mover** cada una de las plantillas al **origen de coordenadas**, utilizando un punto de nuestro dibujo como **referencia**. Es muy importante que este punto de referencia sea el mismo en todas las plantillas.

Tenemos que tener en cuenta el formato, ya que Blender, no importa archivos ".wdg". Por tanto, tenemos que **guardar las plantillas como ".dxf"**. Otro factor importante, es que Blender, **solamente importa líneas**, por lo que tendremos que evitar utilizar polilíneas o descomponerlas antes de la importación.

Una vez que tenemos preparadas las plantillas, abrimos Blender. Ahora tenemos que activar el módulo de Blender que nos permite importar archivos ".dxf", para ello entramos en las **preferencias de usuario** (Crt + Alt +U). Entramos en la pestaña "**Addons**", y en categorías seleccionemos "**Import-Export**". Nos aparece una lista con todas las opciones de importación y exportación, desde ahí, activamos "**Import Autocad DXF Format**". Ahora, en el menú **File-Import**, nos debe aparecer **Autocad(.dxf)**. [Moya, 2010]

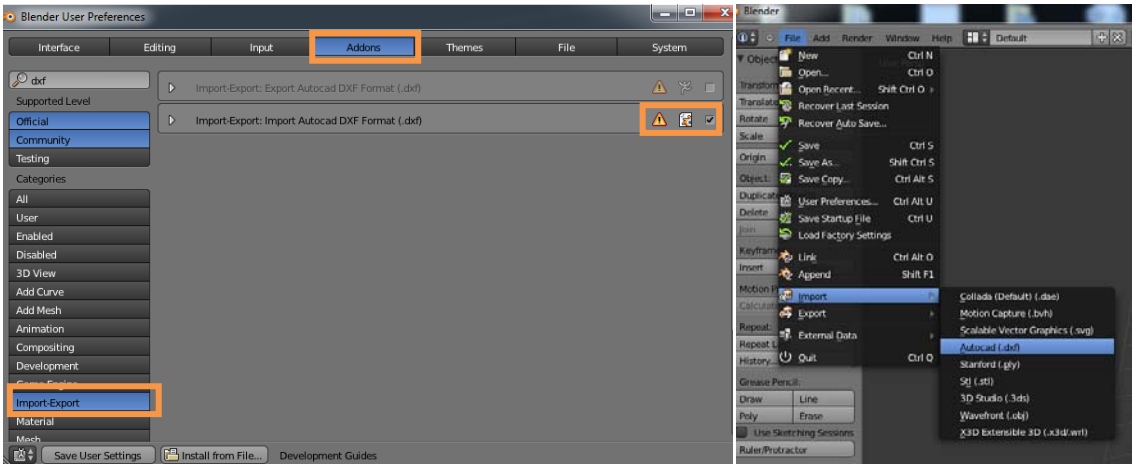


Ilustración 69 – Preferencias de Usuario-Importar

Cuando le damos a importar Autocad DXF, nos aparece un explorador para seleccionar el archivo. Abajo a la izquierda, tenemos las opciones de importación. Es muy importante desactivar la opción "**Replace scene**" y seleccionar **codec:iso-8859-15**. Una vez hecho esto, aceptamos en el botón **Import DXF v.0.1.6** y ya nos aparece nuestra plantilla dxf en la escena, con nuestro punto de referencia situado en el origen de coordenadas.

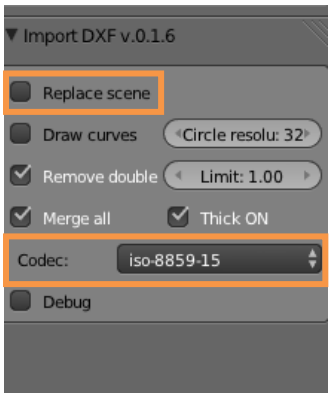


Ilustración 70 – Importar

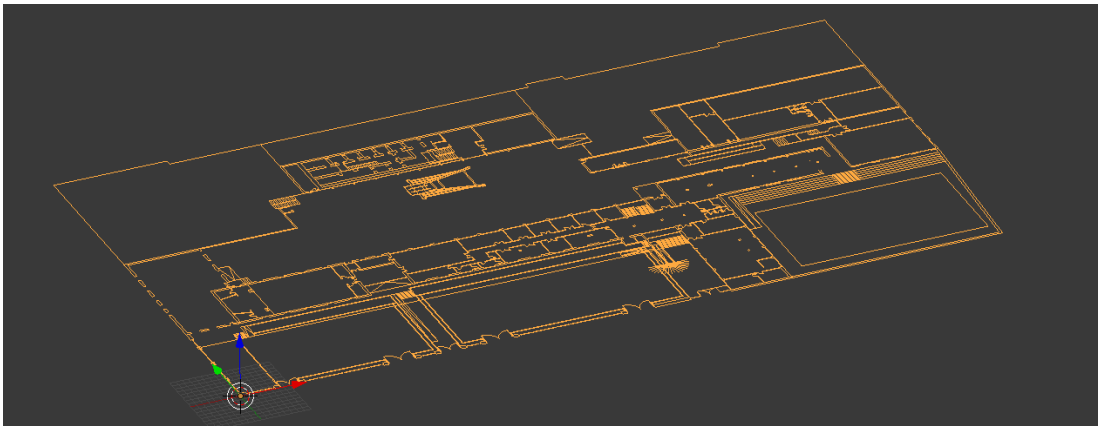


Ilustración 71 – DXF importado

Repetimos la operación con cada una de nuestras plantas, las renombramos desde la ventana **Outliner** y le **asignamos** a cada una de ellas su **posición en Z**. Para ello, seleccionamos la plantilla que queremos mover, pulsamos la tecla rápida G (mover), restringimos el movimiento al eje Z pulsando Z e introducimos el valor de desplazamiento mediante el teclado.

Una vez colocadas todas las plantillas en su lugar correcto, tendremos una plantilla tridimensional para comenzar a modelar los edificios. Desde la ventana Outliner, podemos activar o desactivar la visualización de cada una de ellas para trabajar por plantas cómodamente.

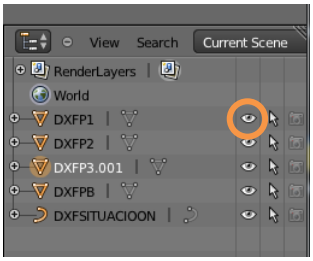


Ilustración 72 – Desactivar / Activar Plantillas

El resultado final de nuestra plantilla será el siguiente.

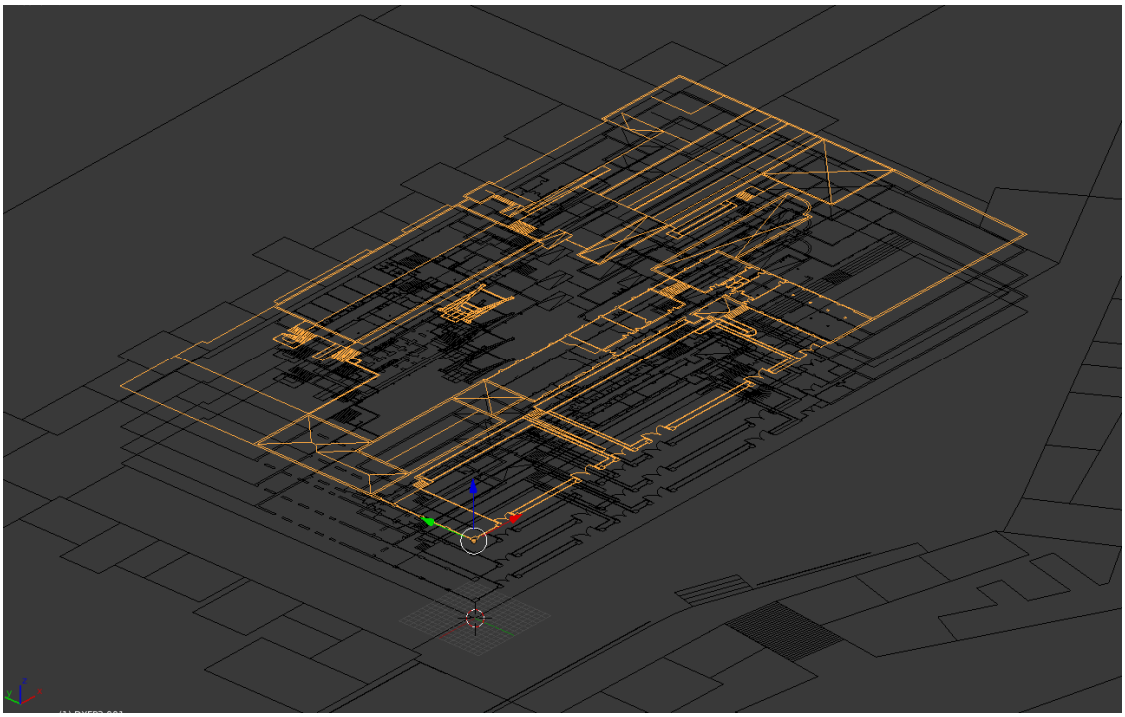


Ilustración 73 – DXF importado

Por último, Blender nos importa las plantillas como **curvas**, por lo tanto "no tienen vértices". Así que necesitamos convertir nuestras plantillas en **mallas** (Objeto poligonal), para que el programa nos reconozca los **vértices** y así poderlos usar de **referencia**. Para ello seleccionamos nuestra plantilla y desde la cabecera de la ventana 3dview seguimos la ruta: [Moya, 2010]

Object → Convert to → Mesh from Curve

4.2 MODELADO

Cuando modelamos un objeto en un programa CAD, lo que estamos haciendo es **reconstruir la geometría** de dicho objeto en un espacio virtual tridimensional. Blender nos proporciona una gran variedad de herramientas para modelar.

Al igual que un **proceso de fabricación** de un objeto real, elegimos el método de fabricación dependiendo de la geometría, material, etc., cuando modelamos un objeto, el proceso o las herramientas de modelado que utilicemos dependerán sobre todo de la geometría.

4.2.1 Organización de la escena

Durante la fase de modelado, es muy importante **organizar la escena**, ya que nos facilitará el trabajo sobre todo cuando tengamos muchos objetos. Cuando creamos un nuevo objeto en Blender, le asigna un nombre por defecto, y un número en el caso de entidades iguales. Por ejemplo "cube","cube.001","cube.002"... Así que para evitar problemas a la hora de seleccionar objetos, es muy importante **nombrar** todo bien. También podemos **crear grupos** de objetos pulsando "**Crtl+G**". En la ventana **Properties** podemos alternar entre ver la lista de objetos o la lista de grupo. [Villar, 2014]

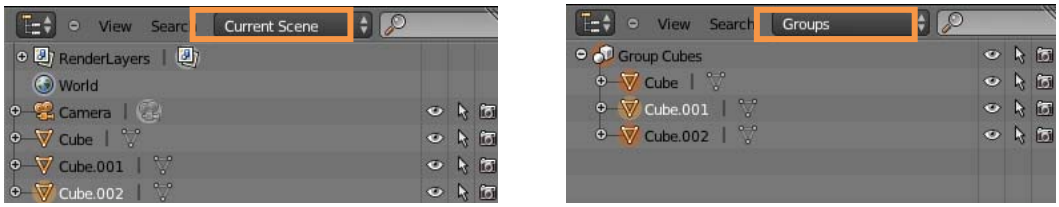


Ilustración 74 – Lista de objetos / Lista de grupos

Otra herramienta que nos permite organizar la escena son las **capas,** que se encuentran en la cabecera de la ventana 3Dview. Podemos mover varios objetos de capa seleccionándolos y pulsando la tecla "**M**". Podemos activar y desactivar capas clicando sobre cualquiera de ellas.

Ilustración 75 – Capas en Blender

4.2.2 Puntos de referencia. Snap.

Cuando estemos manipulando un objeto, tanto en **modo objeto** como en **modo en edición**, necesitaremos el **sistema de referencia automática o snap**, para situarlo con exactitud dentro del modelo.

El sistema de referencia automática de Blender, es algo diferente a la de otros programas CAD. Por ejemplo, en Autocad, cuando movemos un objeto, seleccionamos un punto de origen y otro de destino. Sin embargo en Blender, cuando movemos un objeto, este comienza a moverse por la escena siguiendo la estala del ratón. Si tenemos activado el sistema de referencias y acercamos el puntero del ratón al punto de destino nuestro objeto se situará sobre él. Pero el programa utiliza un **criterio proximidad de puntos**, por lo que colocará sobre el punto destino el punto de nuestro objeto que esté más próximo en la posición anterior. Por tanto, a veces, tendremos que mover previamente nuestro objeto para colocarlo en la posición deseada.

Podemos activar el sistema de referencia pulsando el icono de imán situado en la cabecera de la ventana 3Dview. Además, en el menú desplegable podremos elegir la clase de referencia destino: Incremento, vértice, arista, cara o volumen. [Flavell, 2010] [Goás, 2009] [Moya, 2010]

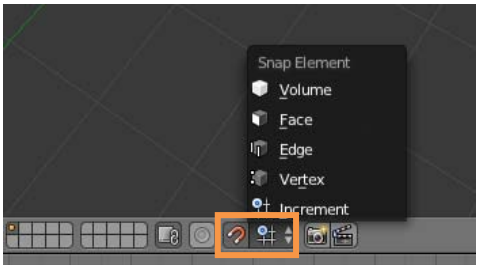


Ilustración 76 – Sistema de referencia

4.2.3 Modo edición

Cuando queramos modificar o modelar la geometría de un objeto, utilizaremos siempre el **modo edición**, ya que en modo objeto como vimos en el capítulo anterior solamente podemos aplicar las operaciones básicas de movimiento, giro y escalado.

Al activar el modo edición, nos aparecen unos iconos que representan jerárquicamente los elementos de una malla: **Punto, Arista y Cara**. [Flavell, 2010]

Ilustración 77 – Modo edición

A la derecha de los iconos de selección, hay un icono "**limit selecction to visible**". Desactivándolo, podemos seleccionar puntos, aristas y caras ocultas desde la perspectiva actual.

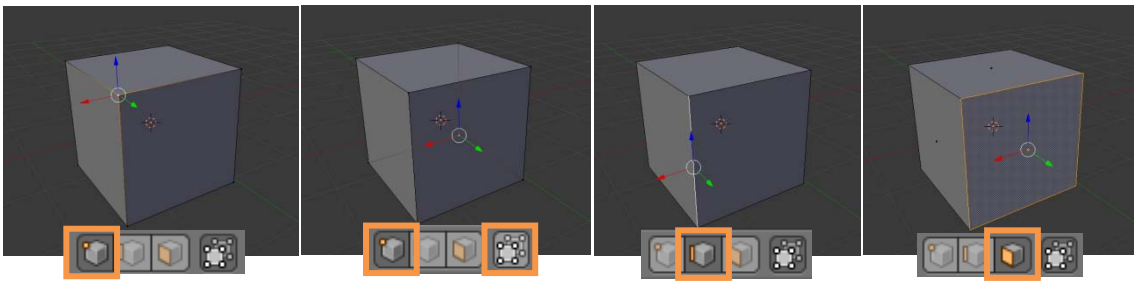


Ilustración 78 – Modo edición: Punto / Punto oculto / Arista / Cara

Seleccionando el icono de selección correspondiente, podremos seleccionar con el botón derecho del ratón: puntos, ejes o caras. Una vez seleccionado el elemento, podremos **moverlos, girarlos, escalarlos** o aplicarle cualquier **otra herramienta** de modelado. [Hess, 2011] [Flavell, 2010]

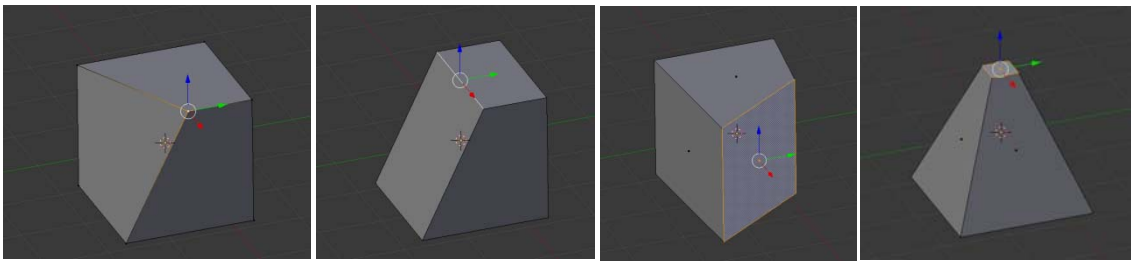


Ilustración 79 – Operaciones básicas
(1) mover punto - (2) mover arista - (3) rotar cara - (4) escalar cara

4.2.4 Opciones de modelado básicas.

4.2.4.1 Extrusión / Extrude.

Es una de las herramientas más potentes de modelado, y es muy útil combinada con el giro y el escalado. Al igual que en el proceso de extrusión en fabricación, esta herramienta nos permite crear objetos con una **sección transversal ya definida** y fija.

Además, establece una jerarquía entre punto, arista, superficie y volumen. Ya que si extruimos un **punto**, obtenemos una arista; si extruimos una **arista**, obtenemos una **superficie** y si extruimos una superficie, obtenemos un **prisma**.

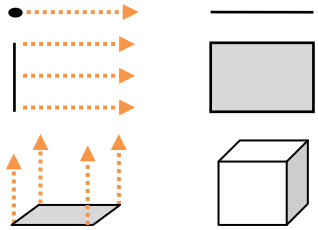


Ilustración 80 – Extrusión

Para realizar una extrusión en Blender, en modo edición, seleccionamos el objeto que queremos **extruir** y pulsamos la tecla rápida "**E**". Automáticamente comienza la extrusión en el **eje perpendicular** al objeto. Como siempre podemos cambiar esta restricción a cualquier eje o extruir libremente sin restricción alguna. Si necesitamos que la extrusión sea exacta, solamente tenemos que introducir la longitud de la misma mediante el teclado. [Flavell, 2010] [Goás, 2009] [Villar, 2014]

En la siguiente ilustración, vamos a ver cómo crear una **botella** rápidamente con el comando **extruir**, combinado con **escalar**. En este ejemplo, vamos a partir de un **cilindro**. (1) Agregamos un cilindro a nuestra escena, lo seleccionamos y entramos en modo edición. Seleccionamos la **cara superior**. (2)(3) **Extruimos** y **escalamos** la cara sucesivamente dándole forma a la botella. (4) Con los mismos comandos hacemos la forma del tapón.

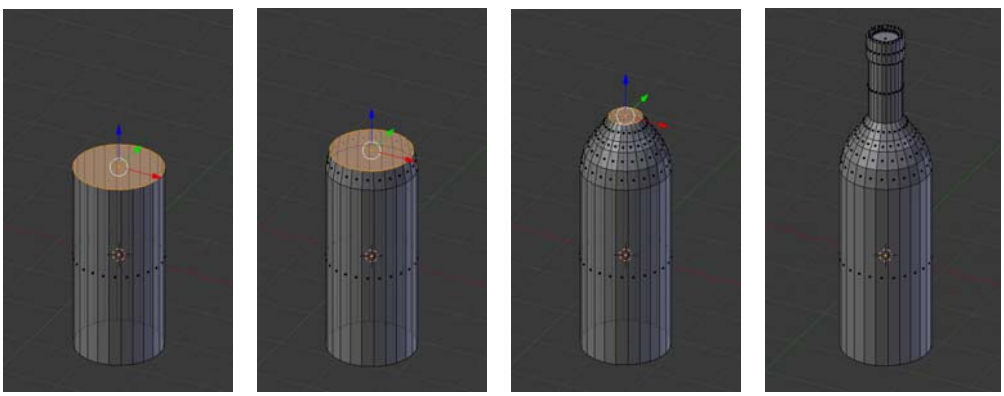


Ilustración 81 – Modelado de una botella mediante extrusión / escalar

4.2.4.2 Rellenar / Fill.

Utilizamos el comando Fill, para crear **elementos intermedios** de unión. Su acceso rápido es la tecla "F". Para utilizarla, tenemos que seleccionar **dos o más elementos** y pulsar la tecla "F", entonces nos creará un elemento intermedio dependiendo de la jerarquía. Si utilizamos Fill entre dos puntos, nos creará una arista; si lo hacemos con tres puntos o si utilizamos un punto y una arista, creará el triangulo que definen; si lo utilizamos entre dos o más rectas coplanarias, creará el polígono que definen; etc.

En el caso anterior de la botella, imaginemos que en lugar de partir de un cilindro, queremos partir de una **circunferencia**. Cuando insertamos la circunferencia en Blender y (1) activamos el modo de visualización sólido, nos damos cuenta de que son una serie de aristas delimitando la longitud de la misma, pero que no existe el plano interior. Por lo tanto tendríamos que generarlo nosotros. (2) Primero entraríamos en modo edición y seleccionaríamos todas las aristas. (3) Con las aristas seleccionadas, pulsamos la tecla "F". Por tratarse de muchas aristas coplanarias y cerradas, nos creará la **superficie delimitada por las mismas**. (4) Ahora, seleccionamos esta nueva cara y extruimos, creando el cilindro de partida del punto anterior. [Flavell, 2010] [Goás, 2009]

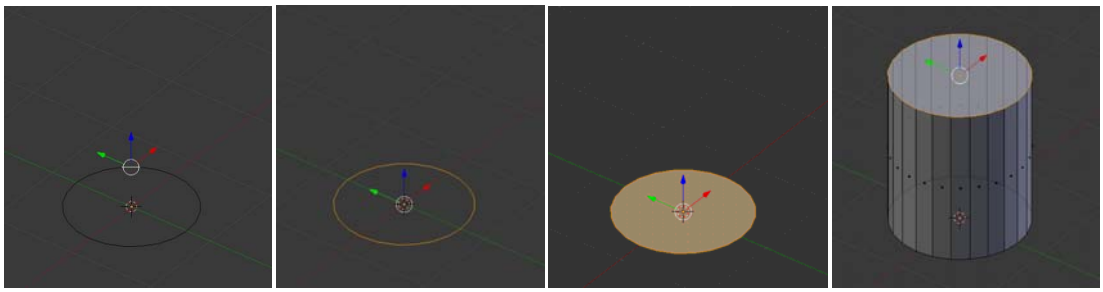


Ilustración 82 – Crear cara con Rellenar / Fill

4.2.4.3 Revolución / Spin.

Es otra herramienta de modelado muy potente, ya que nos permite generar **superficies de revolución**, a partir de un malla poligonal plana. Utiliza los mismos criterios que la extrusión y genera una **extrusión circular** con respecto a **un eje**.

Cuando seleccionamos un elemento y ejecutamos el comando spin, en la barra desplegable izquierda ("T"), encontramos las **opciones de la revolución**. En primer lugar encontramos los **pasos** (steps), elegimos de cuantos **segmentos** queremos que esté compuesta nuestra curva de revolución. Cuanto mayor sea, mejor resultado obtendremos pero a cambio de un mayor número de caras. También podemos elegir el **ángulo** de la revolución, el **centro**, que por defecto será la posición del 3Dcursor y el **eje** sobre el que queremos efectuar el giro. [Goás, 2009] [Villar, 2014] [Hess, 2011]

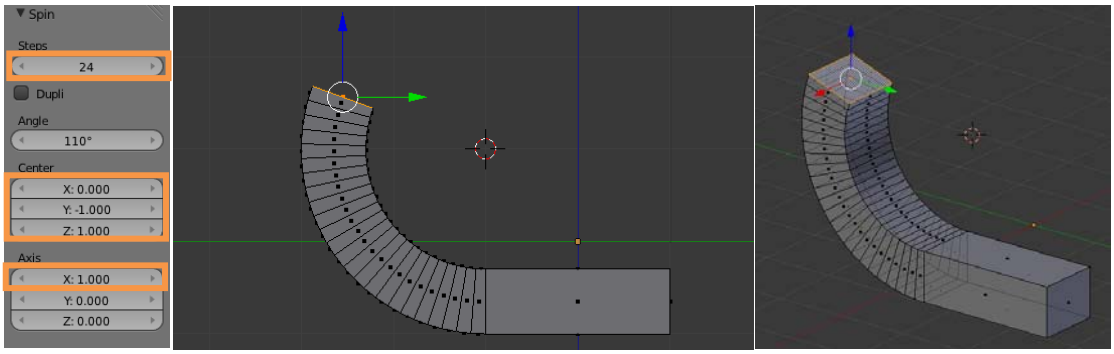


Ilustración 83 – Spin

Seguiremos con el ejemplo de la botella, esta vez utilizando el eje Z, como eje de revolución. (1) Insertamos un **plano**, y desde el modo edición lo rotamos 90º sobre el eje X. Lo escalamos en Z hasta obtener la forma del cuerpo de nuestra botella. (2) Eliminamos tres de sus lados y nos quedamos con la **arista vertical**. (3) Seleccionamos el **punto superior**, y vamos realizando extrusiones poco a poco, **dibujando el perfil** de la botella. (4) Ejecutamos el comando el spin "Alt+R" y realizamos una **revolución** con 24 pasos de **360º** y en el **eje Z**. (5) Utilizamos rellenar/fill para cerrar la cara superior y la inferior.

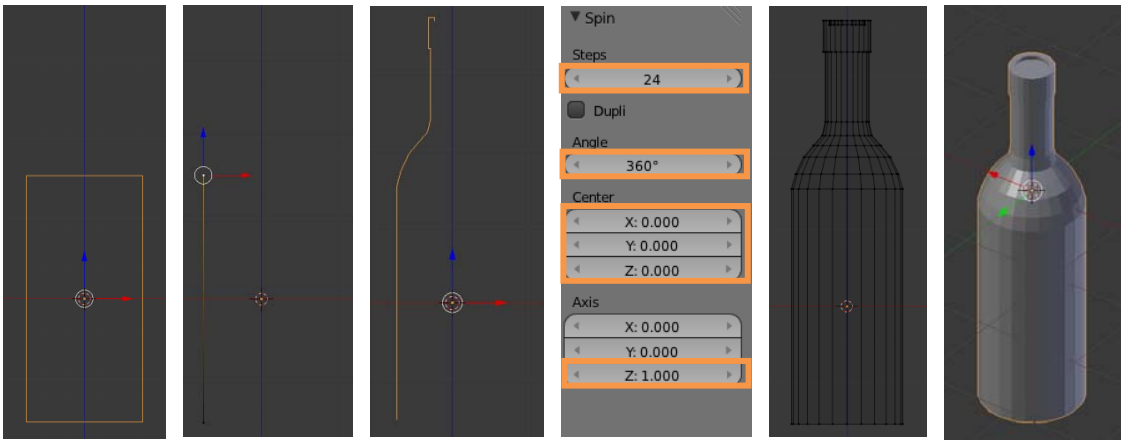


Ilustración 84 – Modelado de una botella mediante revolución.

4.2.4.4 Biselado / Bevel.

Como su nombre propio indica nos realiza uno o varios **cortes oblicuos en las aristas** de un objeto. Biselar es una herramienta muy importante cuando estamos modelando, ya que si observamos a nuestro alrededor por un momento, muy pocas veces encontraremos aristas perfectas. Esta herramienta ayuda a que nuestro modelo sea mucho **más realista**, ya que, aparte de suavizar las aristas, nos generará unas **reflexiones** en los brillos bastante más interesantes. El único problema, es que obtenemos un **número elevado de caras**, por lo tanto si abusamos de esta herramienta, tendremos unos tiempos de render muy grandes.

En modo edición, seleccionamos la arista que queremos biselar y utilizamos las teclas "Crtl+B". Nos aparecen las opciones del biselado en la pestaña izquierda, donde podemos elegir la **distancia** de biselado "Offset" y el **número de segmentos**. [Flavell, 2010]

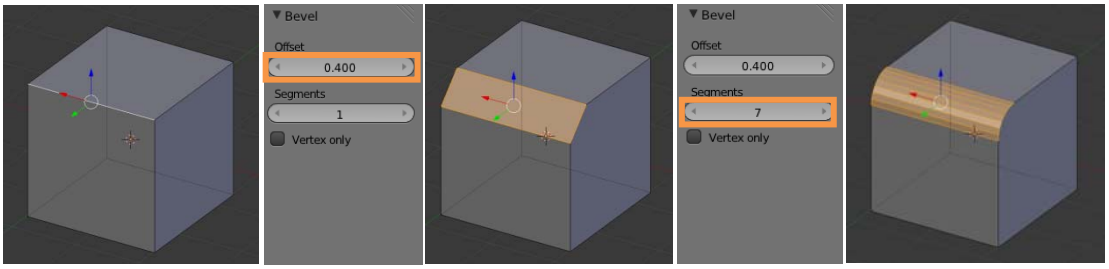


Ilustración 85 – Biselado.

Vamos a utilizar esta nueva herramienta en nuestra botella, para ello partimos de la botella generado mediante revolución.

Si observamos el modelo de nuestra botella, vemos que, tanto las aristas de la base, como las de la boca de la botella, presentan un aspecto irreal, ya que forman 90º exactos y tienen un aspecto muy pronunciado.

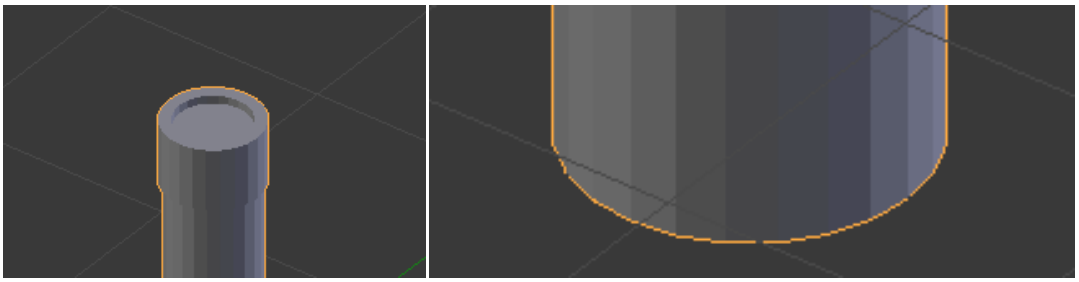


Ilustración 86 – Aristas sin biselar.

Empezaremos por la base, (1) seleccionamos las aristas de la base. (2) Aplicaremos un bisel de 0,35 de distancia y 3 pasos.

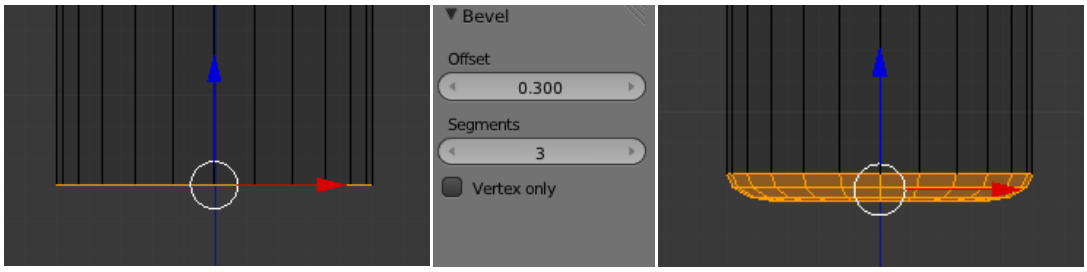


Ilustración 87 – Aristas base biselada.

Realizaremos la misma operación en la boca de la botella. (1) seleccionamos las aristas a biselar. (2) Aplicamos un bisel de 0,03 de distancia y 2 pasos.

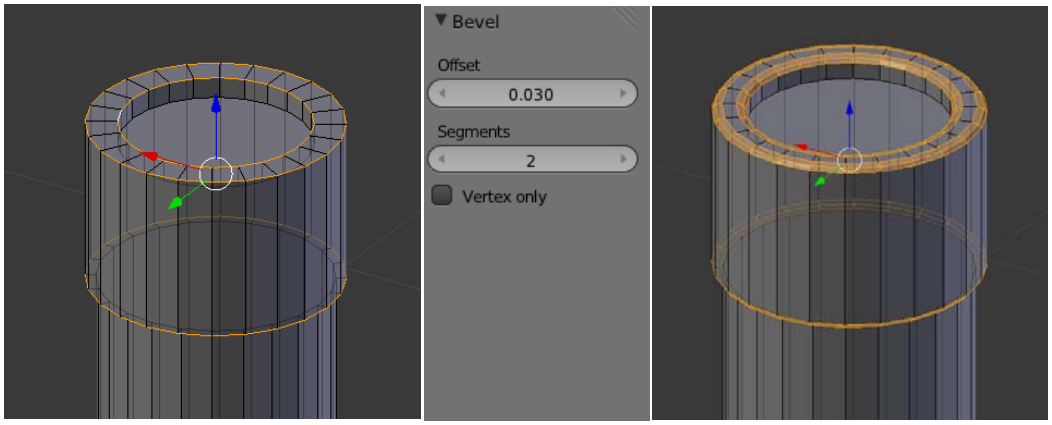


Ilustración 88 – Aristas base biselada.

4.2.4.5 Subdivisiones, Sección y Cortes / Subdivide, Loop cut and knife

Estas herramientas, las utilizaremos para **dividir y cortar** polígonos o volúmenes. En algunas ocasiones, necesitaremos cortar un polígono o subdividirlo para continuar con el proceso de modelado o bien para indicar un cambio de material.

Estas herramientas las podemos encontrar en el panel izquierdo "T" de la ventana 3dview. [Flavell, 2010] [Villar, 2014]

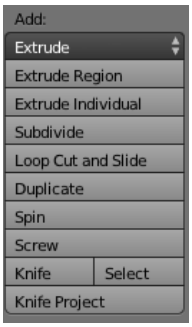


Ilustración 89 - Mesh tools

Subdivide:

Con el comando subdividir, podemos dividir en **partes iguales** una arista o una superficie. Para ello elegiremos los elementos que queremos subdividir y seleccionaremos el número de secciones. En la imagen vemos cómo afecta el comando subdividir dividir con tres secciones: a una **arista**; a un **plano en una única dirección**, seleccionando dos aristas enfrentadas; y a **un plano en dos direcciones**, seleccionando la superficie.

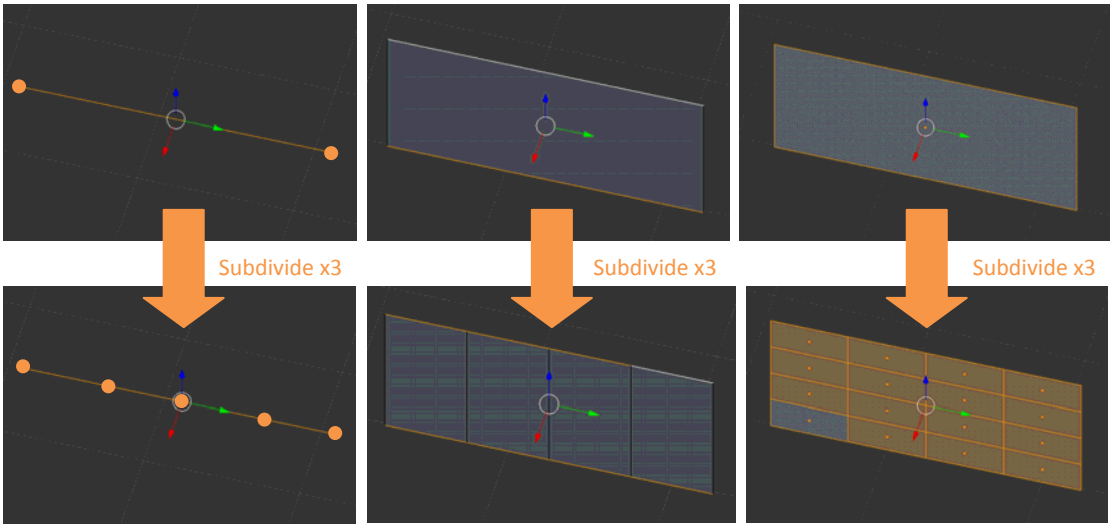


Ilustración 90 – Subdividir / Subdivide.

Loop cut:

Esta herramienta realiza **secciones perpendicularmente** a las direcciones principales de nuestro objeto. Las divisiones serán igual, pero podemos desplazarla hacia un lado o el otro de nuestro objeto.

En primer lugar, elegimos el objeto sobre el que queremos realizar las secciones. Pulsamos "Crtl+R" y elegimos la **dirección** de las secciones. Una vez seleccionada la dirección podemos desplazarla sobre el objeto. En las opciones de Loop cut podemos variar el **número de cortes** y volver a **desplazarlos** con la barra "Factor" donde en el valor 0 estarán centrados y en los valores 1 y -1 estarán ajustadas a los extremos de nuestro objeto.

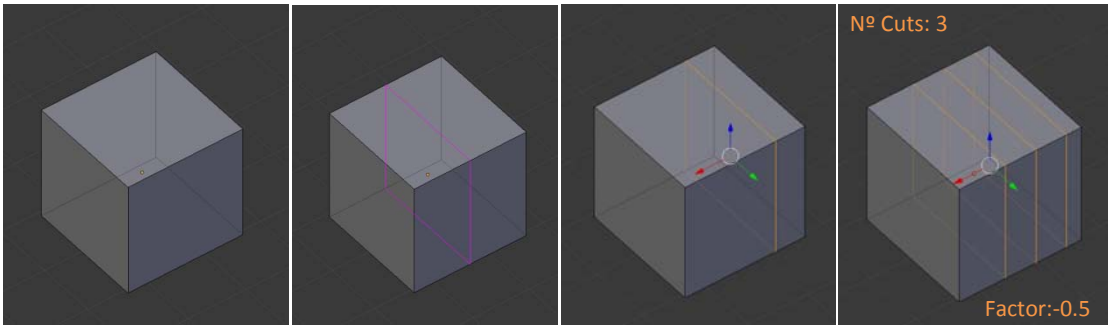


Ilustración 91 – Secciones / Loop Cut and Slice.

Knife:

La herramienta cortar, nos permite **seccionar** una o varias caras de manera **manual**, eligiendo los puntos iniciales, finales e intermedios. El sistema de corte depende de la posición de la vista actual, por lo que es recomendable utilizar una de las vistas ortogonales. La tecla rápida de acceso es la tecla "K" y una vez definida la sección aceptaremos con la "barra espaciadora". Podemos utilizar la tecla "X" para cortar las caras ocultas y la tecla "C" para restringir el ángulo de corte en múltiplos de 45^{o}.k

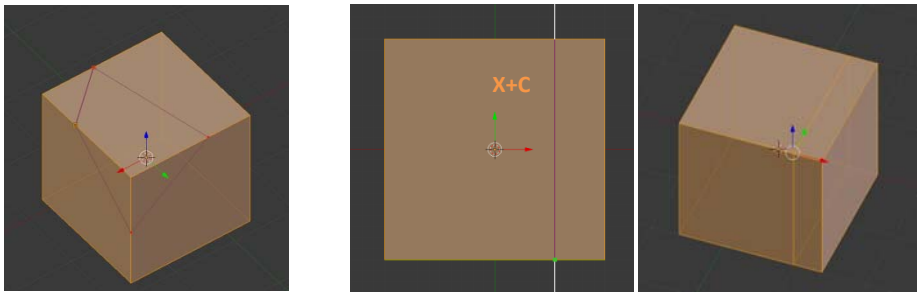


Ilustración 92 – Ejemplos del uso de "knife".

4.2.4.6 Separar, Juntar y Duplicar.

Si queremos **separar** parte de un objeto, en otro nuevo, tendremos que **seleccionar** en el **modo edición** los objetos que queremos separar y pulsar la **tecla "P"**. Se abrirá un pequeño menú, donde elegiremos separar la selección. Los objetos separados, se renombren automáticamente como el objeto original seguidos de una numeración del tipo 001.

Para **juntar** dos objetos en un único objeto, tendremos que hacerlo desde el **modo objeto**. Seleccionaremos los objetos utilizando "Shift + Botón derecho del ratón" y utilizaremos las **teclas "Crt + J"** para unirlos. El nombre del nuevo objeto, será igual que el del elemento activo, es decir, el que hemos seleccionado el último. [Villar, 2014]

Si lo que queremos es **duplicar** un objeto existente, tenemos varias formas:

- **Shift + D** : El objeto se **duplica como objeto independiente**.
- **Alt +D** : El objeto se **duplica anidado**. Es decir, cualquier cambio en la geometría o los materiales de algunos de los dos, le afecta al otro.

En cualquiera de los dos casos, el objeto duplicado se llamara igual que el objeto original seguido de una numeración del tipo 001.

En la figura, el cubo de la izquierda es el original "Cube", el del centro está copiado independientemente "Cube.001" y el de la derecha "Cube.002" está anidado a "Cube". Si modificamos la geometría del cubo original "Cube", vemos como "Cube.002" cambia automáticamente mientras que "Cube.001" permanece igual.

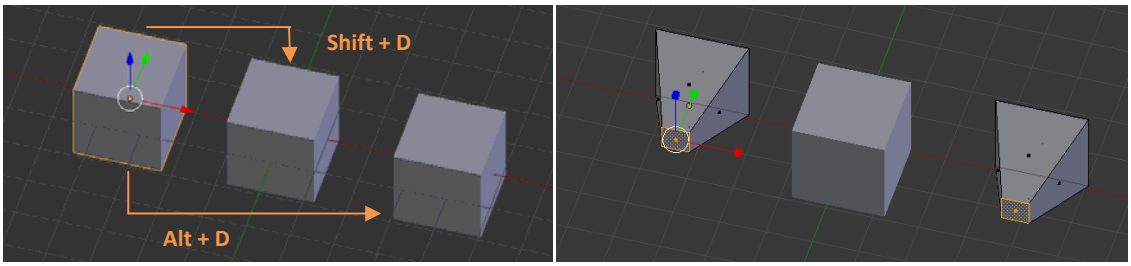


Ilustración 93 – Duplicar Objetos

4.2.5 Modificadores.

Los modificadores son operaciones que aplicamos sobre nuestro objeto sin que este pierda su geometría, a no ser que lo apliquemos de una forma definitiva. Podemos encontrarlos en la ventana de propiedades, en la pestaña con forma de llave inglesa.

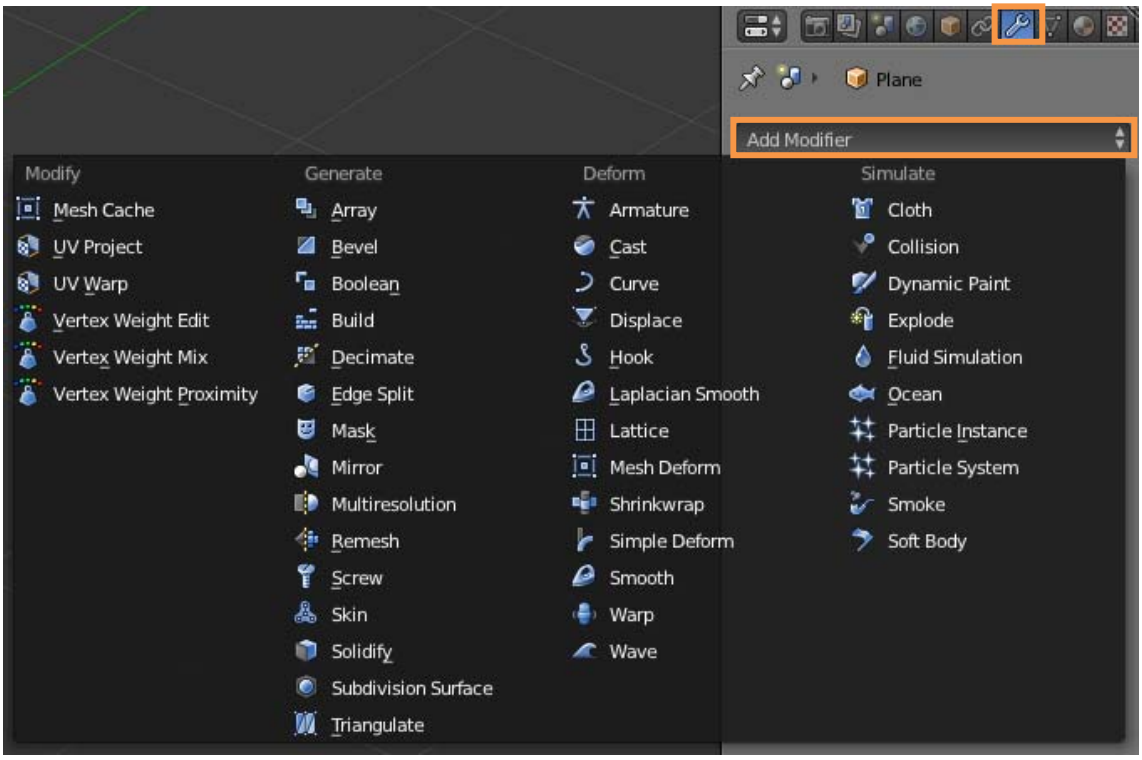


Ilustración 94 – Lista de modificadores

Podemos aplicarle a un objeto tantos modificadores como consideremos necesario. Pero tenemos que tener en cuenta el orden de aplicación, ya que un orden incorrecto nos puede dar resultados no deseados.

En la pestaña de los modificadores, aparece para cada objeto, los modificadores activos. Utilizando las flechas de la derecha, podemos variar el orden de aplicación y sus diferentes efectos. [Flavell, 2010] [Villar, 2014]

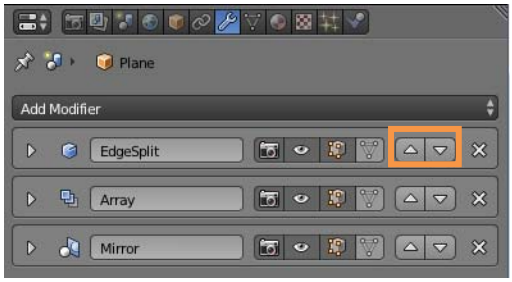


Ilustración 95 – Modificadores activos

4.2.5.1 Suavizar Objeto /Smooth and Edge Slide.

Cuando generamos superficies de revolución mediante mallas poligonales, realmente estamos trabajos con **poliedros** en lugar de esferas, cilindros, etc. El problema es que por muchos vértices que pongamos a nuestro objeto, siempre tiene un efecto poligonal y no de superficie continua. Para solucionar este problema utilizamos el efecto "smooth", que **suaviza** el objeto transformándolo en una **superficie continua** y no poligonal. En el panel izquierdo de la ventana 3Dview, tenemos las opciones de Shading: Smooth y Flat.

Si continuamos con el ejemplo de la botella de los apartados anteriores podemos ver cómo funciona este efecto. (1)En primer lugar, activamos el modo de vista sólido, para ver el efecto poligonal descrito anteriormente. (2) seleccionamos nuestra botella y clicamos sobre el botón "smooth". Podemos ver como desaparece el efecto poligonal. (3) Sin embargo, observamos que hace un efecto extraño en las superficies planas, como en el tapón, cuando forman 90^{o}. Aquí es donde entra el modificador **Edge Slide**, que establece un **ángulo máximo** entre caras para el efecto smooth. (4) Aplicamos el modificador Edge Slide con un ángulo inferior a 90^{o}. (5) Y podemos ver como el tapón vuelve a ser completamente plano. [Roosendaal, 2004] [Villar, 2014]

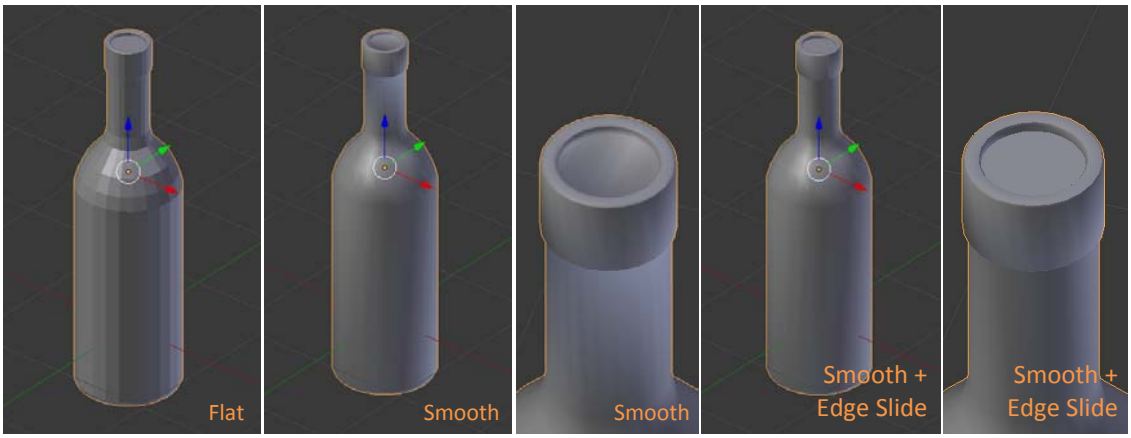


Ilustración 96 – Smooth + Edge Slide

4.2.5.2 Subdivisiones /Subdivision and Decimate.

Otra forma para suavizar los objetos, es utilizar el modificador **Subdivision Surfaces**. Este modificador subdivide las caras de nuestro objeto, reduciendo el efecto poligonal, pero **incrementando** notablemente el número total de **caras** y de **vértices**. El resultado, combinado con el efecto smooth y el modificador edge slide, es un objeto perfectamente definido y suavizado, pero a un alto coste de recursos y tiempo a la hora de renderizar. [Roosendaal, 2004]

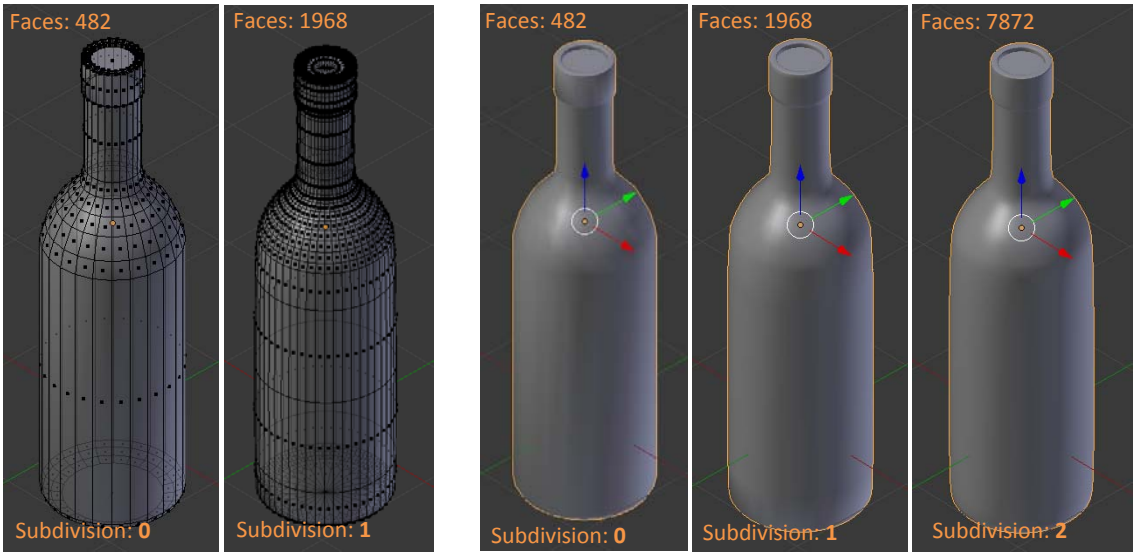


Ilustración 97 – Subdivision Surfaces

Existe otro modificador para realizar la operación inversa, **Decimate**. Los utilizamos para **simplificar** los objetos. Según la composición de nuestra escena, tendremos objetos en un primer plano, que sí será necesario que tengo una gran definición, pero existirán otros objetos que probablemente, no necesiten un elevado número de caras para quedar correctamente definidos.

Para aplicar el modificador, lo seleccionamos de la lista y elegiremos la opción del centro, **Un-Subdivide**, después elegimos el número de iteraciones del modificador. En algunos casos, es importante revisar bien el modelo, ya que al simplificar las subdivisiones, es probable que se pueda eliminar alguna de las caras de los objetos.

4.2.5.3 Biselado / Bevel.

Funciona igual que la herramienta de modelado [ver 4.2.4.4] , pero por tratarse de un modificador, aplica el mismo bisel a todas las aristas del objeto por igual. [Villar, 2014]

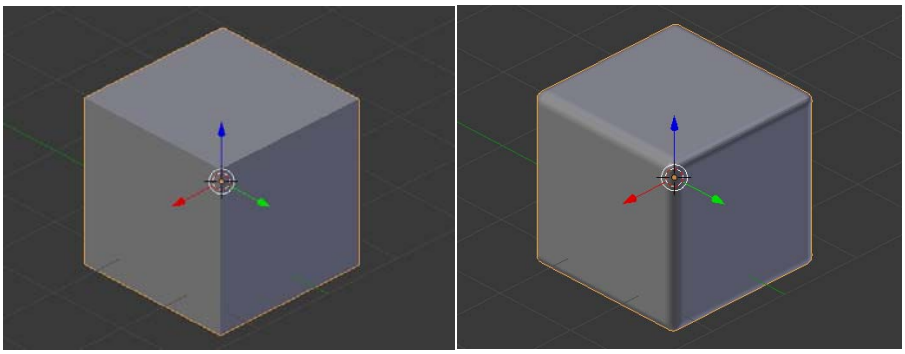


Ilustración 98 – Modificador Bevel

4.2.5.4 simetría / Mirror.

Este modificador, nos permite modelar **objetos simétricos**. Utiliza como centro de simetría, el **origen de coordenadas locales** de un objeto y podemos definir cualquier de los tres ejes como **ejes de simetría**. Uno de los aspectos más interesantes de este modificador, es que realiza la simetría en **tiempo real**, es decir, mientras estamos modelando nuestro objetos, vemos los cambios en el lado simétrico. [Villar, 2014]

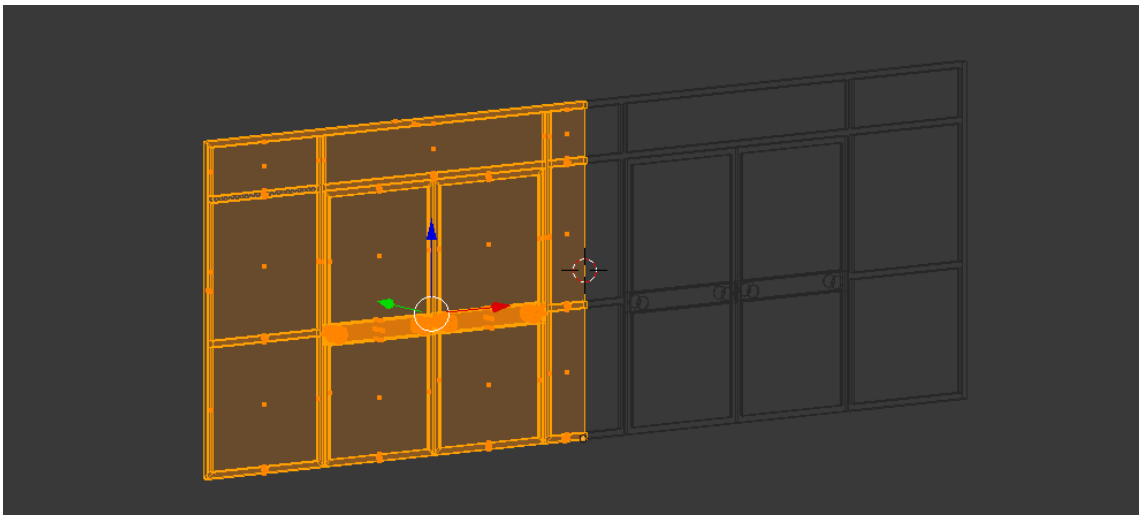


Ilustración 99 – Mirror. Puerta entrada salón de actos EIMIA.

4.2.5.5 Organizar / Array.

El modificador **Array**, nos permite reproducir objetos a una **distancia constante**. Podemos ordenar el objeto alineado a una **dirección** o alineado a una **curva**. Es muy útil cuando tenemos en nuestro modelo algún **objeto modulado** o cuando tenemos que repetir objetos de una forma ordenada.

Podemos **encadenar** tantos modificadores array como queramos, de manera que podemos repetir un objeto "n" veces en un eje y "m" veces en otro. Veamos un ejemplo con nuestra botella, si tuviésemos que colocarlas en una estantería, utilizaríamos dos modificadores array en direcciones paralelas. [Villar, 2014]

Ilustración 100 – Modificador Array.

4.2.5.6 Operaciones Booleanas.

Este modificador está basado en el **álgebra de Boole,** que define las intersecciones entre conjuntos. Es una herramienta para **sumar, restar e intersecar** sólidos entre sí, formando uno nuevo. En una operación booleana tenemos un **sólido activo**, sobre el que se realiza la operación, y los **sólidos operadores**. [Roosendaal, 2004] [Villar, 2014]

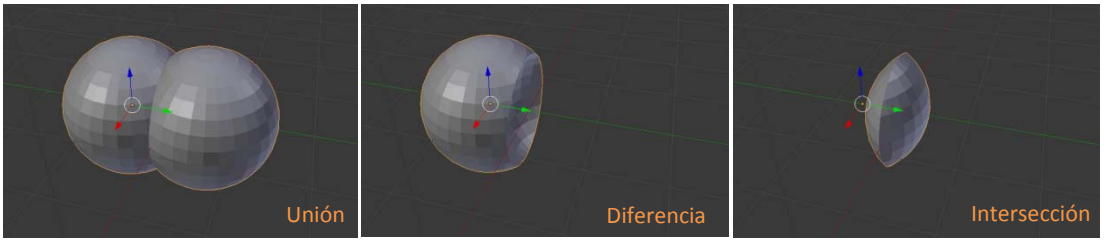


Ilustración 101 – Operaciones Booleanas.

4.2.6 Modelado Edificios.

Para comenzar con el modelado de los edificios, necesitamos tener las plantillas dxf importadas como vimos en el apartado 4.1.2.

4.2.6.1 Tabiques y Fachadas.

Para modelar los tabiques y las fachadas de los edificios, partiremos de un **plano**. Este plano lo ajustaremos a unos de los muros, e iremos **extruyendo sus lados**, de manera que dibuje en el plano XY todos los **muros**, marcando todas las puertas y ventanas. Una vez que nuestro muro es una superficie, **extruiremos en Z**, marcando también todos los puntos importantes, rodapié, alfeizar ventana, altura ventana, alturas de puertas, etc. [Moya, 2010]

De esta forma, obtenemos todos los muros de una planta como una única maya.

En nuestro proyecto, hemos separado los **tabiques** y las **fachadas** como elementos independientes. Ya que una de los recursos que queremos utilizar, es la posibilidad de mostrar los interiores del edificio en una vista desde el exterior, desactivando las fachadas. Por lo tanto nos interesa que sean **objetos independientes**.

Comenzaremos con los **tabiques**. Insertamos un plano y lo ajustamos a uno de los tramos del tabique moviendo sus aristas. Una vez colocado, comenzaremos a extruir sus aristas siguiendo los tabiques y parando en cada cambio de dirección, en cada puerta, etc.

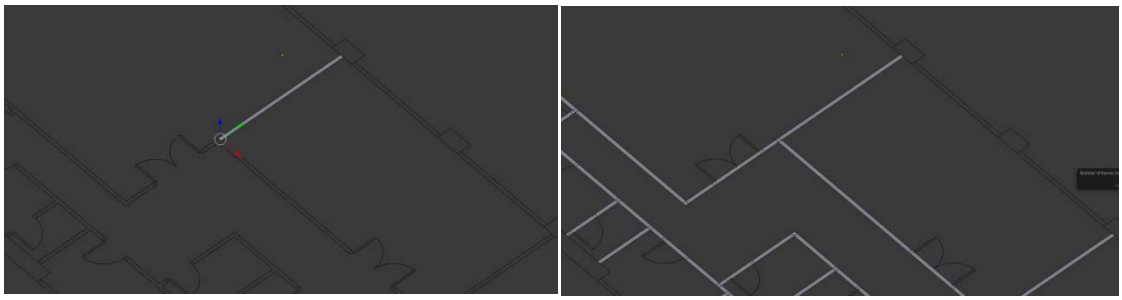


Ilustración 102 – Dibujando los tabiques.

Una vez que hemos cubierto con nuestra superficie todos los tabiques, eliminamos las superficies correspondientes a las puertas. Y ya tenemos nuestra superficie preparada para comenzar la extrusión.

Antes de extruir, tenemos que tener muy claro los puntos claves en Z y las alturas de extrusión de los mismos. En este caso tenemos, nos interesa marcar a una altura de 10 cm el rodapié para poder asignarle más tarde su correspondiente material. El siguiente punto, será la altura de las puertas a 2,10 m sobre el nivel del suelo, por lo tanto la segunda extrusión será de 2 metros, ya que hemos subido los 10 cm del rodapié.

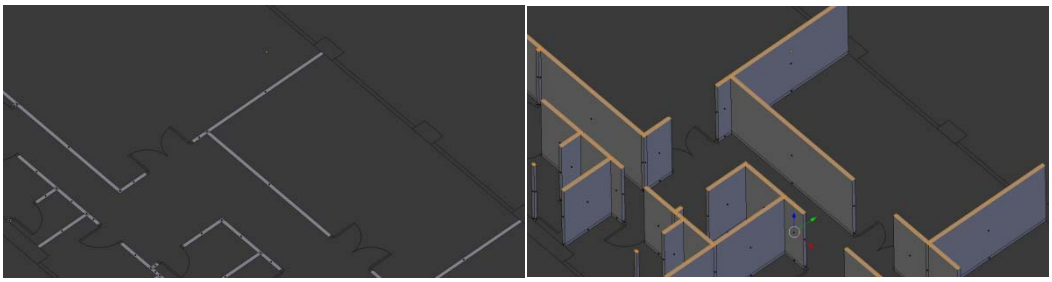


Ilustración 103 – Extrusión tabiques I.

Finalmente realizamos una última extrusión hasta la altura del techo.

Ya solamente nos falta cerrar la parte superior de las puertas, para ello extruimos unos de sus laterales. Es conveniente que borremos las caras que quedan en el interior del muro tras esa extrusión. Por último, seleccionamos todos los vértices de nuestro objeto y desde el panel desplegable izquierdo, en Mesh Tools seleccionamos "remove dobles". De esta forma borramos posibles puntos duplicados y dejamos el objeto como una malla única y soldada.

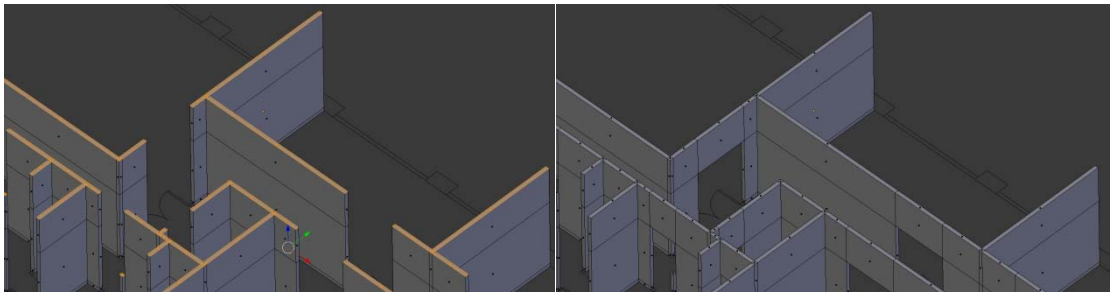


Ilustración 104 – Extrusión tabiques II.

Para las **fachadas**, seguimos el mismo procedimiento. Dibujamos el muro de fachada extruyendo un plano, y parando en ventanas y puertas. Esta vez las alturas de extrusión son diferentes. Primero extruimos 10 cm de rodapié y después 65 cm hasta la altura del alfeizar de la ventana.

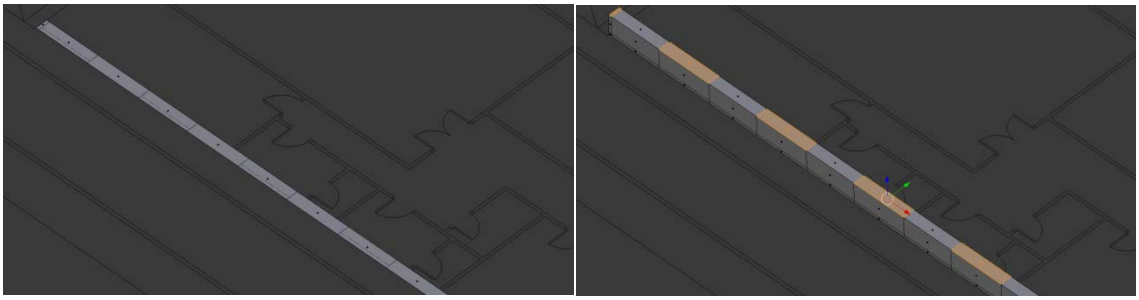


Ilustración 105 – Extrusión fachadas I.

Ahora, deseleccionamos las superficies correspondientes a las ventanas y continuamos con la extrusión. Tenemos que extruir 2,4 m de altura de ventana y 17 cm hasta la altura del techo. Finalmente, al igual que con los tabiques, cerramos las partes superiores de las ventanas.

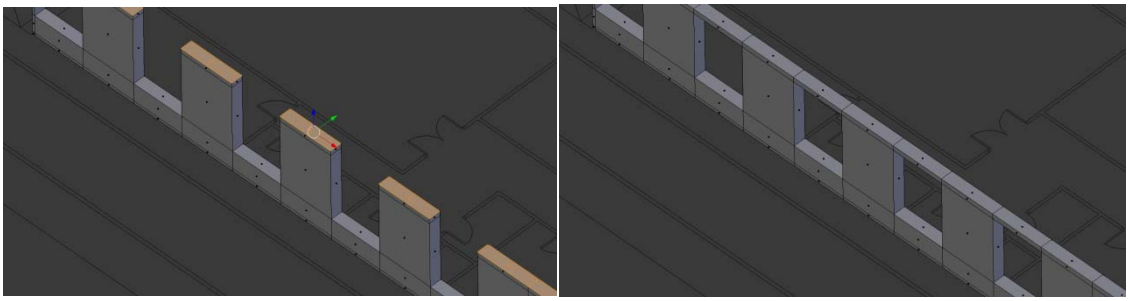


Ilustración 106 – Extrusión fachadas II.

4.2.6.2 Molduras, puertas y ventanas.

Para las molduras de la fachada, las puertas y las ventanas, utilizaremos el mismo proceso descrito anteriormente, ya que son elementos con una geometría bastante simple y se pueden realizar mediante extrusión.

Para las **carpintería**s, es conveniente realizar un .dxf como plantilla, de manera que nos facilite el dibujo en Blender. Una vez importado, redibujaríamos la geometría partiendo de un plano y extruiríamos los diferentes elementos.

Ilustración 107 – Carpinterías.

La fachada del edificio Störr, tiene una **moldura exterior** muy modulada. Por lo tanto podemos utilizar un modificador array y solamente tendríamos que modelar un solo módulo.

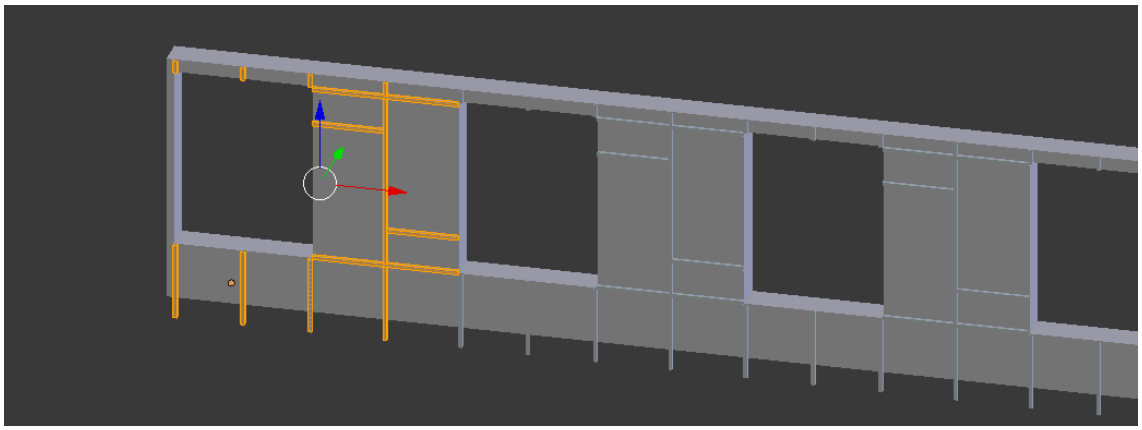


Ilustración 108 – Módulo fachada con Array.

Para los **letreros de entrada**, utilizaremos la herramienta de texto. Añadimos un nuevo objeto, y seleccionamos **Text**. Si entramos en el modo edición, podemos editar el texto. [Villar, 2014]

Ahora en la ventana de propiedades, nos aparece una pestaña con las propiedades del texto. Desde aquí, podemos modificar las propiedades del texto, como tipo de letra, tamaño, etc. Pero además podemos jugar con la geometría, extruir, achaflanar, etc.

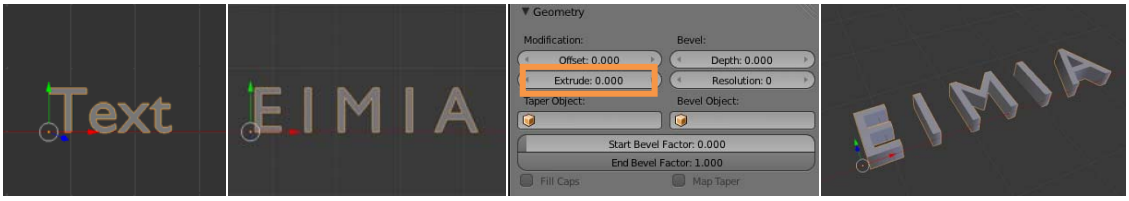


Ilustración 109 – Extrusión de un texto.

4.2.6.3 Mobiliario.

En cuanto al mobiliario, tenemos varias opciones. Podemos **modelar** nuestros propios muebles, o utilizar las **bibliotecas** gratuitas de la comunidad Blender, disponibles en la web, como en *www.blenswap.com*.

En el proyecto, como estamos reproduciendo un edificio existente, consideré que era más apropiado modelar los muebles. Ya que utilizar un mobiliario distinto del actual, le restaría realismo a las escenas.

Solamente se ha recurrido al uso de bibliotecas en el caso de objetos como los extintores, árboles, coches, sillas de oficinas, ordenadores o las máquinas del taller de fabricación.

Ilustración 110 – Objetos de biblioteca.

El resto de objetos y mobiliario ha sido modelado y texturizado basándonos en las fotos de los mismo.

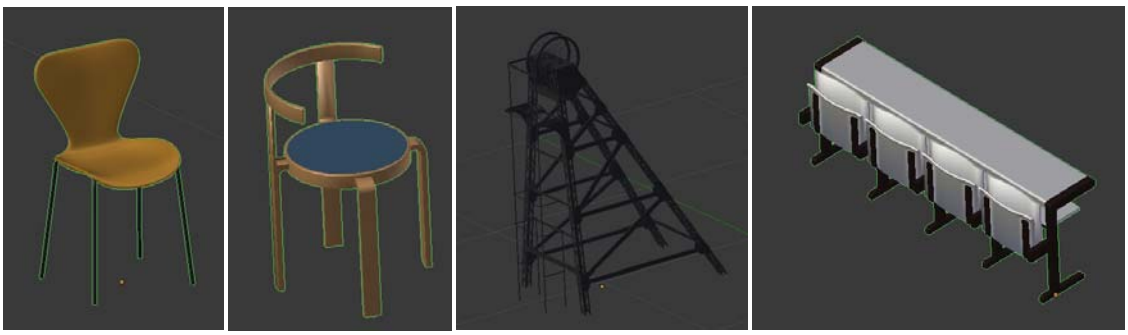


Ilustración 111 – Objetos modelados.

Vamos a ver un ejemplo con las sillas de la biblioteca.

Partiremos de un círculo, del cual vamos a borrar la mitad, ya que como es un objeto simétrico, utilizaremos el modificador Mirror.

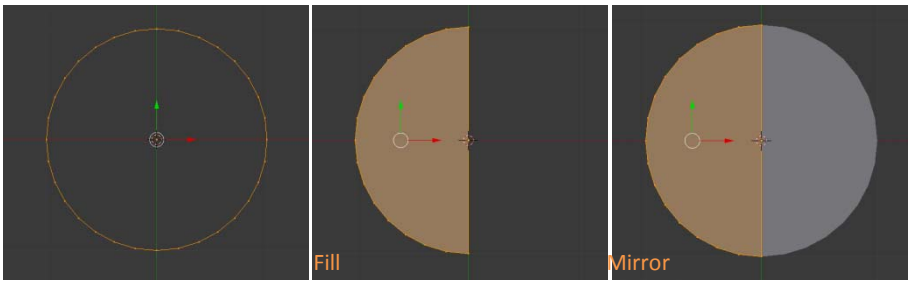


Ilustración 112 – Modelado silla I.

Extruiremos el círculo, para crear la forma del asiento. Una vez hecho esto, volvemos a extruir con z=0 y escalamos el polígona para crear el cambio de material del asiento. Utilizaremos la orden Bevel para achaflanar el contorno superior del asiento.

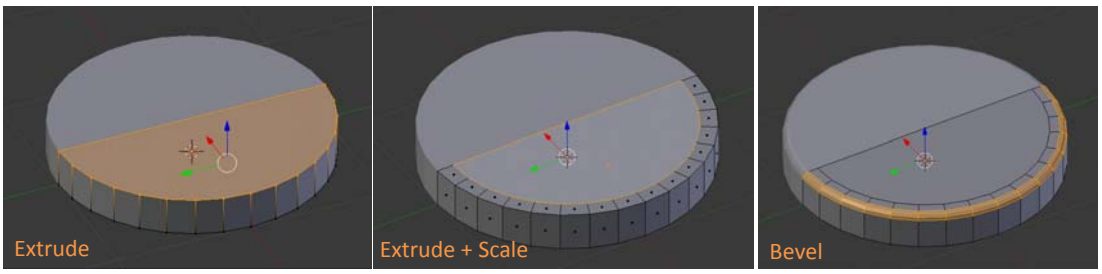


Ilustración 113 – Modelado silla II.

Ahora vamos a hacer el respaldo. Seleccionamos un cuarto de circunferencia de la parte del asiento y duplicamos las caras. Las escalamos respecto al 3DCursor y las movemos hasta su posición aproximada. Una vez colocadas las extruimos y les aplicamos un chaflán en el perímetro exterior.

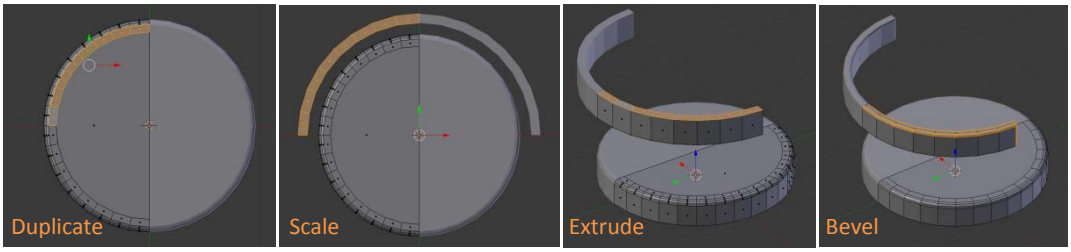


Ilustración 114 – Modelado silla III.

Continuaremos con las patas traseras de la silla. Seleccionamos las caras inferiores del respaldo que se encuentra situada a 45º y la extruimos hasta la altura del suelo. Realizaremos dos cortes en la parte interior a la altura del asiento y extruiremos esas caras para unir las patas con el asiento. Aplicaremos un chaflán, en el encuentro de esta última pieza.

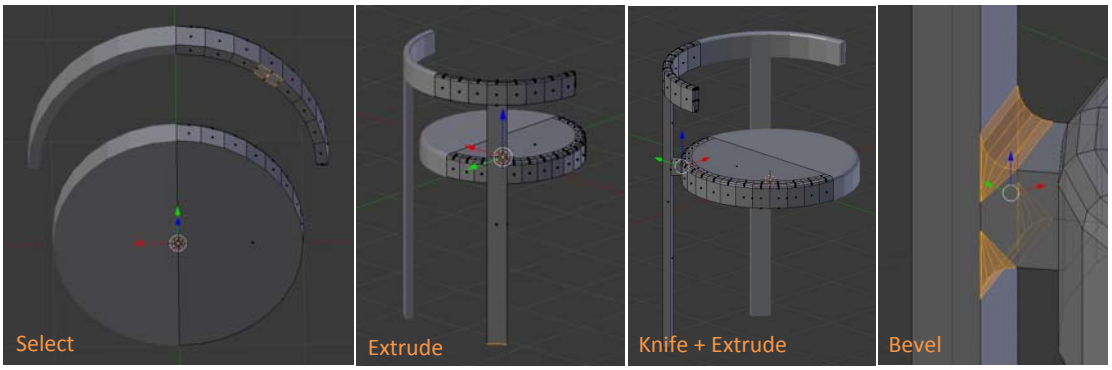


Ilustración 115 – Modelado silla IV.

Ya solamente faltan las patas delanteras. Seleccionamos las caras laterales donde está situada la pata. Las extruimos con Z=0 y escalamos en su posición. Una vez escaladas realizamos una pequeña extrusión para hacer la parte horizontal de la pata.

Ahora colocamos esta pieza en posición frontal y realizamos un spin de 90º. Finalmente extruimos hasta el suelo.

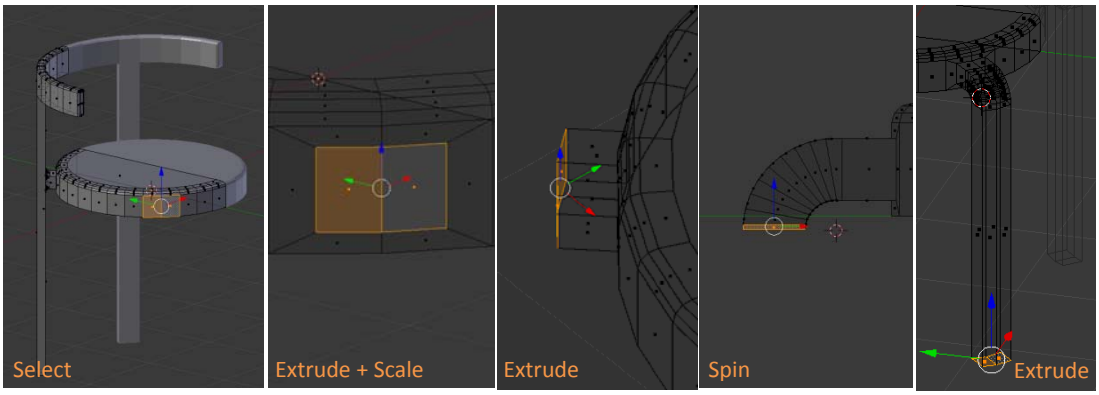


Ilustración 116 – Modelado silla V.

Ya tenemos el modelo completo, ahora pasaremos a suavizarlo mediante el comando Smooth y el modificador Edge Split.

Ilustración 117 – Modelo silla.

4.2.7 Modelo Final.

En la siguiente imagen, podemos ver el aspecto del modelo una vez terminado, algunos elementos, aparecen con colores, debido a que hemos ido aplicando materiales a medida que se modelaba.

Ilustración 118 – Modelo Tridimensional.

4.3 CREACIÓN Y ASIGNACIÓN DE MATERIALES.

Cuando creamos un material para un modelo virtual, estamos definiendo una serie de propiedades físicas que definirán el **comportamiento de la luz** al incidir sobre las superficies donde esté aplicado. Estas propiedades básicas son:

- Diffuse / Difusión de Luz
- Specular / Reflexión especular
- Shading / Sombreado
- Transparency / Transparencia
- Mirror / Espejo
- Shadow / Sombras

4.3.1 Crear un nuevo material.

Para crear un nuevo material, lo haremos desde la ventana de propiedades, en la pestaña "**material**". Podemos crear un material utilizando el botón "+" o eliminarlo pulsando el botón "-". Al igual que ocurre con nuestros objetos, es muy importante nombrar correctamente cada material, de esta forma los tendremos organizados y nos será más fácil trabajar con escenas grandes.

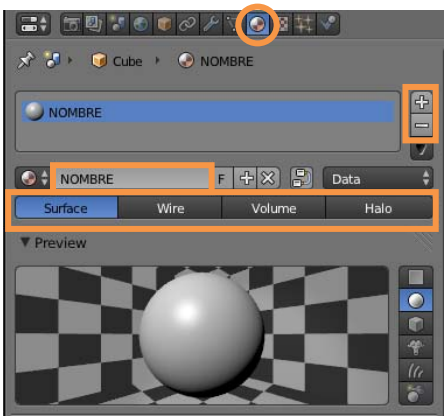


Ilustración 119 – Pestaña Materiales.

Podemos elegir entre cuatro tipos de materiales. El primero es "**Surface**", es un material **sólido** y sobre el que definiremos las propiedades físicas antes descritas. Es el más indicado para generar materiales realistas. Los otros tres, son para crear diferentes efectos. "**Wire**" renderiza en modo alámbrico, "**Volume**" crea un efecto humo y "**Halo**" nos muestra los vértices del objeto iluminados. [Blender, 2014]

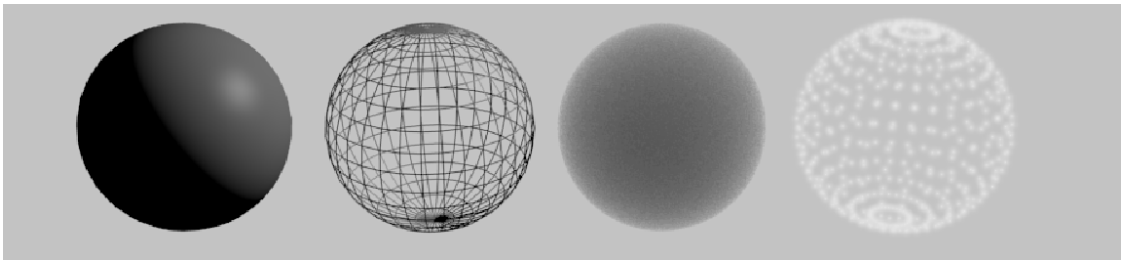


Ilustración 120 – Materiales: Surface / Wire / Volume / Halo.

4.3.2 Difusión de la luz.

Cuando la luz impacta sobre una superficie, se produce una posterior **irradiación** por efecto de la **difusión**. Esta irradiación es muy dispersa (isotrópica), por lo que la cámara verá la misma cantidad de luz sobre el objeto independientemente del ángulo de visión incidente. Por lo tanto la luz difusa es **independiente del punto de vista**. Si una superficie solamente refleja la luz de forma difusa, tendrá un **aspecto mate**.

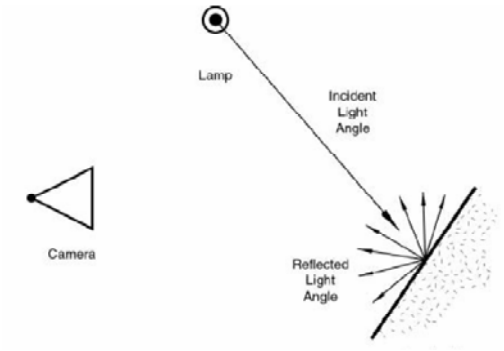


Ilustración 122 – Fenómeno de Difusión.

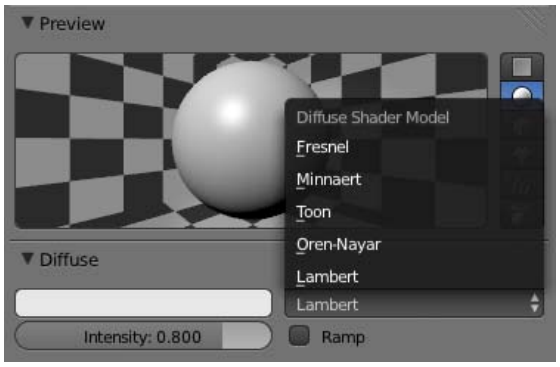


Ilustración 121 – Opciones de Difusión.

En las opciones, podemos seleccionar el **color básico** del material, que es el color usado por el sombreador de difusión (diffuse shader) y la **intensidad** de luz que es difundida. Blender ha implementado 5 modelos matemáticos diferentes para calcular la difusión: Lambert, Oren-Nayar, Toon, Minnaert y Fresnel. [Roosendaal, 2004] [Blender, 2014]

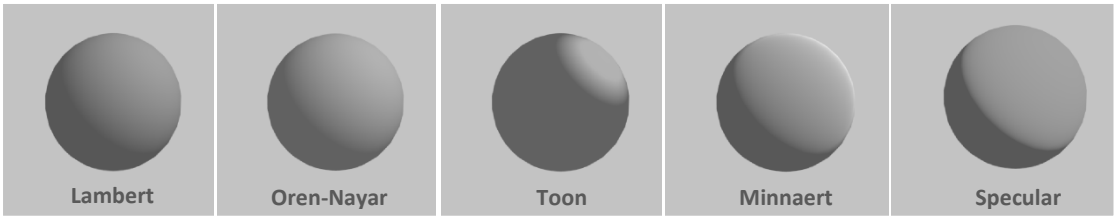


Ilustración 123 – Diffuse shader

4.3.3 Reflexión especular.

Al contrario que la difusión, la reflexión especular es **dependiente del punto de vista.** La luz que impacta sobre una superficie especular se verá **reflejada con un ángulo** de salida igual al ángulo de incidencia, lo que hace el ángulo de visión cobre mucha importancia. Este tipo de reflexión crea **finos y brillantes reflejos**, haciendo que la superficie parezca **pulida**. Es importante aclarar que el fenómeno de reflexión especular no tiene nada que ver con la reflexión que deberíamos de ver en un espejo.

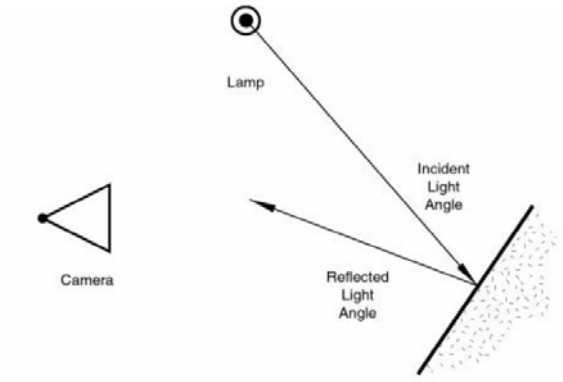


Ilustración 125 – Fenómeno de Reflexión.

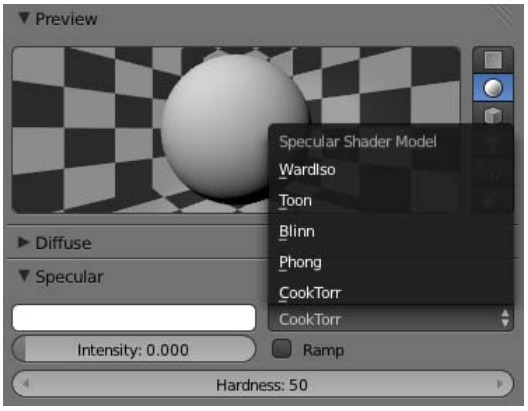


Ilustración 124 – Opciones de Reflexión.

En las opciones, podemos elegir el **color del brillo** especular, la **intensidad** de la luz que se refleja y la **dureza** del mismo, que regula la anchura del brillo especular. Al igual que para la difusión, tenemos diferentes métodos de cálculo para la reflexión especular: Cook Torr, Phong, Blinn (incluye **índice de refracción**), Toon y Wardiso. [Roosendaal, 2004]

Ilustración 126 – Specular shader

4.3.4 Sombreado.

En el panel Shading, encontramos algunas opciones que modifican el comportamiento de los sombreadores de difusión.

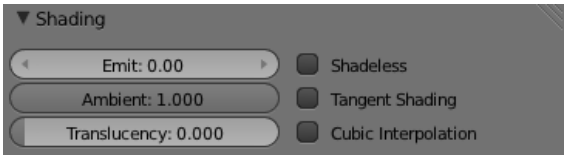


Ilustración 127 – Opciones de Sombreado.

El valor de **emisión** (Emit), hace que la superficie emita luz por sí misma, lo que hace posible que podamos ver el objeto aún sin luces en nuestra escena. Lo podemos utilizar para simular luces encendidas.

El valor **ambiental** (Ambient), modifica la de difusión producida por la luz ambiental de la escena. Un valor cercano a cero, hará que las superficies que no reciban luz directa sean mucho más oscuras.

El valor de la **translucidez** (Translucency), determina la cantidad de luz directa que atraviesa el objeto para iluminar las caras en sombra, generando una pequeña sensación de que el objeto es translúcido.

Además, tenemos la opción de desactivar los sombreadores, **Shadeless.** Por lo que la superficie se representará con un color difuso homogéneo independientemente de donde estén situadas las fuentes de luz.

La opción **cubic interpolation**, suaviza la difusión, suavizando las transiciones entre las zonas iluminadas y las zonas en sombra. Por última la opción **Tangent Shading**, genera un efecto anisótropo en la difusión. [Roosendaal, 2004] [Blender, 2014] [Simond, 2012]

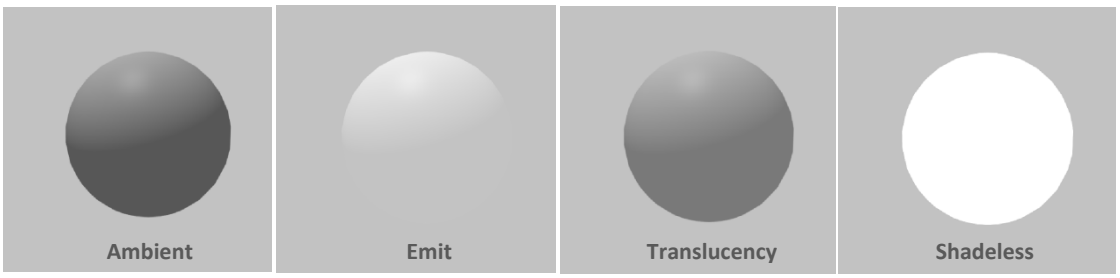


Ilustración 128 – Shading

4.3.5 Transparencia

Si queremos que nuestra superficie sea transparente y deje pasar la luz, tendremos que activar el panel **Transparency.** En este panel, podemos seleccionar entre tres métodos de cálculo de la transparencia: Mask, Z transparency y Raytrace.

El método **Mask** y **Z transparency**, están basados en el método de cálculo **Z buffering** [ver 3.6.8.2]. En el método Mask solamente podemos elegir el grado de transparencia modificando el valor Alpha del material, siendo "0" completamente transparente y "1" completamente opaco. Con Z transparency, podemos elegir el grado de transparencia del material modificando el valor Alpha, pero además podemos elegir el valor de transparencia de los brillos especulares. Además el método Z transparency, no deja activar un efecto muy interesante, el efecto **Fresnel**. Este efecto, varía el grado de transparencia dependiendo de la posición del punto de vista. Si pensamos en un cristal, cuando lo miramos de formas perpendicular es muy transparente, sin embargo, al inclinarlo comienza a comportarse más como un espejo y pierde transparencia, esto es el efecto fresnel.

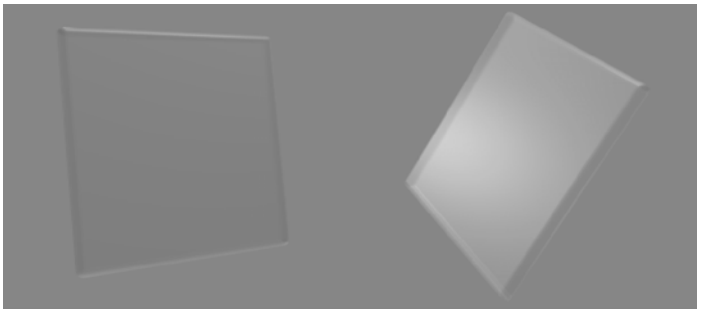


Ilustración 129 – Efecto Fresnel

El método **Raytrace,** utiliza el modelo de cálculo de **trazado de rayos** [ver 3.6.8.1] para calcular las superficies transparentes. Es el método de cálculo con el que se obtienen resultados más realistas, pero también el **más lento**. Tiene muchas opciones de configuración, pero quizá la opción más interesante de este método de cálculo, es que nos permite elegir el **índice de refracción** del objeto. [Roosendaal, 2004] [Blender, 2014]

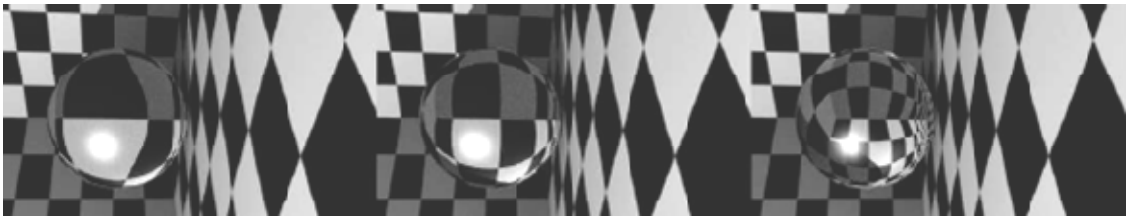


Ilustración 130 – Diferentes IOR

4.3.6 Espejo.

Si necesitamos una superficie reflectante, activaremos el panel **Mirror**. Estas superficies se calculan mediante Raytrace [ver 3.6.8.1]. El rayo que sale de la cámara, rebota en la superficie reflectante, hasta impactar en otro objeto de donde obtiene la información de color, por lo tanto el resultado es la obtención d una **imagen reflejada** en dicha superficie.

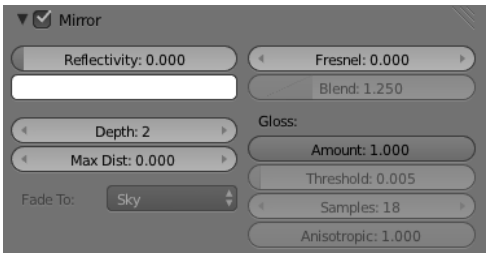


Ilustración 131 – Mirror

Desde el panel mirror, podemos elegir la **cantidad** de reflexión del Raytrace mediante el deslizador "reflectivity", además del **color** de la misma. Esto es muy útil para objetos metálicos como el oro, que aunque reflejan la imagen, mantienen su color. Podemos elegir la **profundidad** o la máxima distancia del reflejo hasta la que se calcularán los rayos reflejados.

Otro factor importante, al igual que en la transparencia, es el efecto **fresnel**, por lo que la claridad del reflejo dependerá de la posición del objeto con respecto el punto de vista. [Roosendaal, 2004] [Blender, 2014] [Simond, 2012]

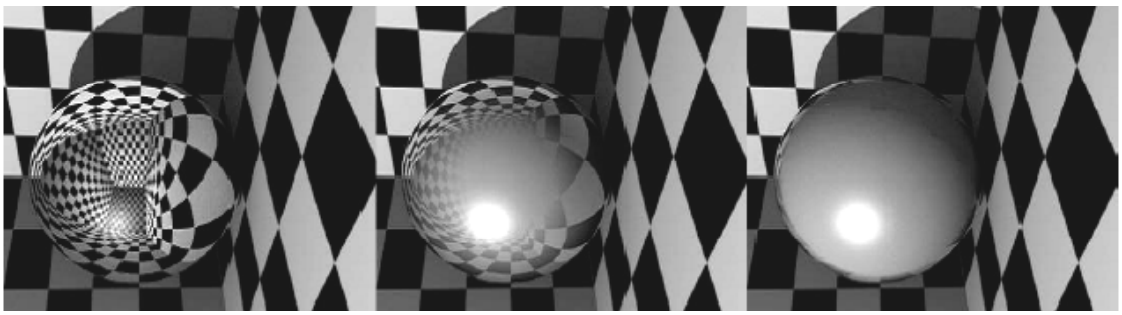


Ilustración 132 – Diferentes valores para el efecto fresnel

4.3.7 Sombras.

Por último, desde el panel **shadow**, podemos activar y decidir cómo le afecta, o como no le afecta, a nuestro material los **rayos translúcidos** o **transmitidos** y los **rayos sombra** [ver 3.6.8.1].

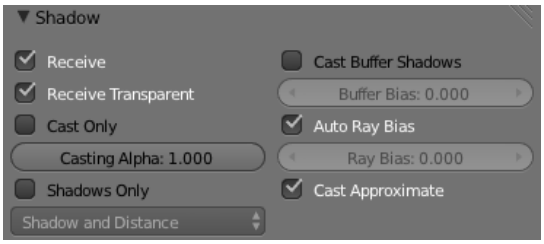


Ilustración 133 – Shadow

Si queremos que nuestro material **reciba sombras** producidas por otros objetos, tenemos que tener activada la opción "**receive".**

Si queremos que nuestro objeto se **ilumine** por los rayos translúcidos, tendremos que utilizar la opción "**receive transparent**". Esta opción es muy importante en materiales utilizados en el interior de una escena cerrada donde tengamos ventanas, ya que la luz proveniente del exterior será toda producida por rayos translúcidos.

Podemos utilizar la opción "cast only", si queremos que nuestro objeto no aparezca en la escena, pero sin embargo si queremos que aparezca su sombra.

Si activamos la opción "shadows Only", solamente se renderizará la parte de nuestro objeto que esté en sombra, es decir que no reciba luz directa ni luz procedente de un objeto transparente. [Roosendaal, 2004] [Blender, 2014]

4.3.8 Aplicar un material a un objeto.

Ahora vamos a ver cómo aplicar los materiales a un objeto, para ello vamos aplicarle materiales a la botella del capítulo anterior. Hemos configurado una escena simple, con varias luces y un plano como base, para poder ir viendo cómo quedan nuestros materiales una vez renderizados. En la imagen vemos como queda la escena sin materiales aplicados. [Roosendaal, 2004] [Blender, 2014]

Ilustración 134 – Escena sin materiales

Para comenzar, seleccionamos el objeto y seleccionamos la pestaña materiales, en la ventana propiedades.

Creamos un **nuevo material**, como vimos anteriormente, utilizando el botón "+". Cuando creamos un nuevo material, este se aplica automáticamente a todo el objeto. Comenzaremos creando un material tipo vidrio para la botella. Para ello, utilizaremos un efecto difuso y un brillo especular en tonos verdosos, y le añadiremos el efecto transparencia.

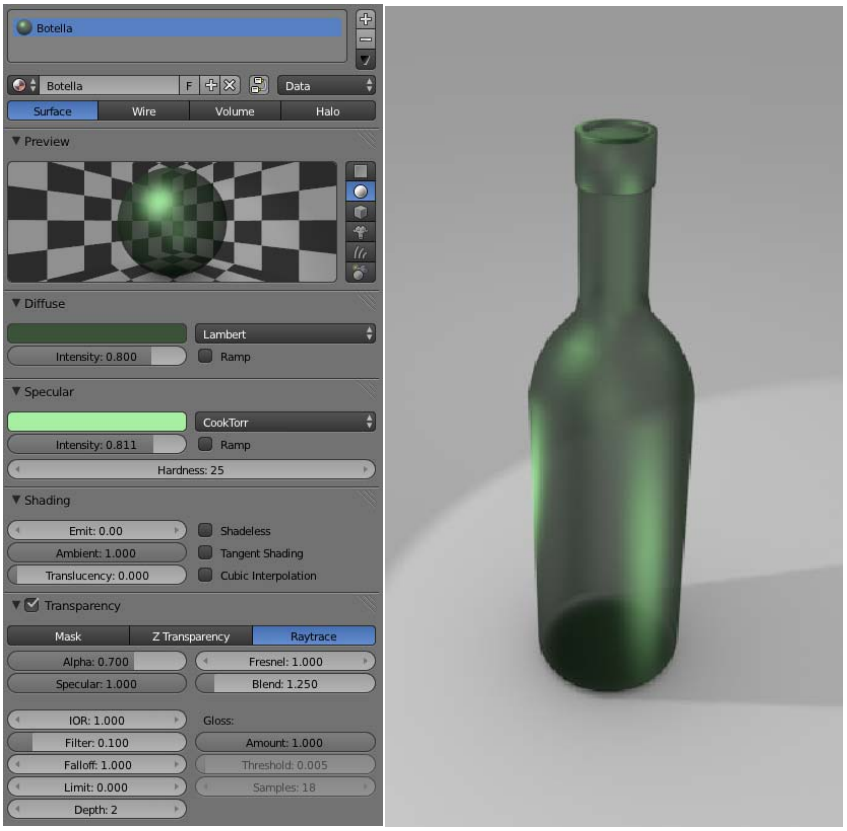


Ilustración 135 – Material vidrio verde.

Ahora crearemos un nuevo material dentro del mismo objeto para la etiqueta. Utilizaremos un material satinado en tonos rojos. Una vez que tenemos creado el nuevo material, entramos en modo edición y **seleccionamos las caras** donde queremos aplicar el nuevo material y utilizamos el botón **asignar** "Assign", bajo el nombre del material.

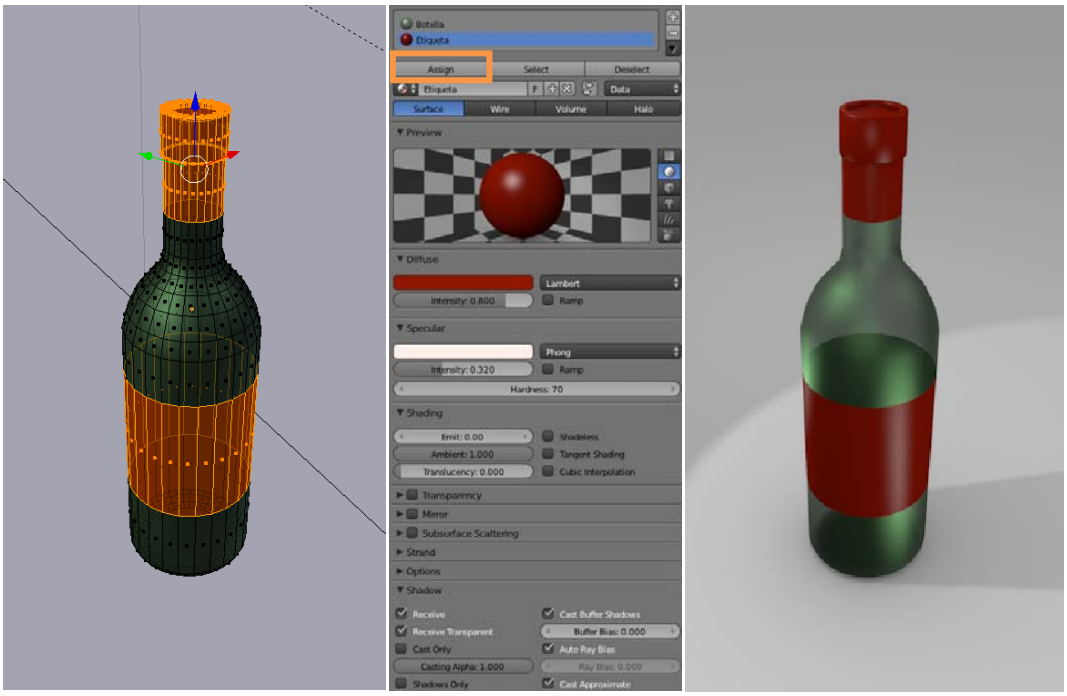


Ilustración 136 – Asignar un segundo material.

Por último, vamos a añadirle el líquido. Pero antes tendremos que modelarlo. Seleccionaremos las caras de la botella que estarán en contacto con el líquido. Las duplicamos "Ctrl+D" y las escalamos un poco hacia el interior de la botella. Por último, cerramos la cara superior con el comando fill "F". Volvemos a seleccionar las caras correspondientes y le aplicamos un nuevo material que simule el líquido.

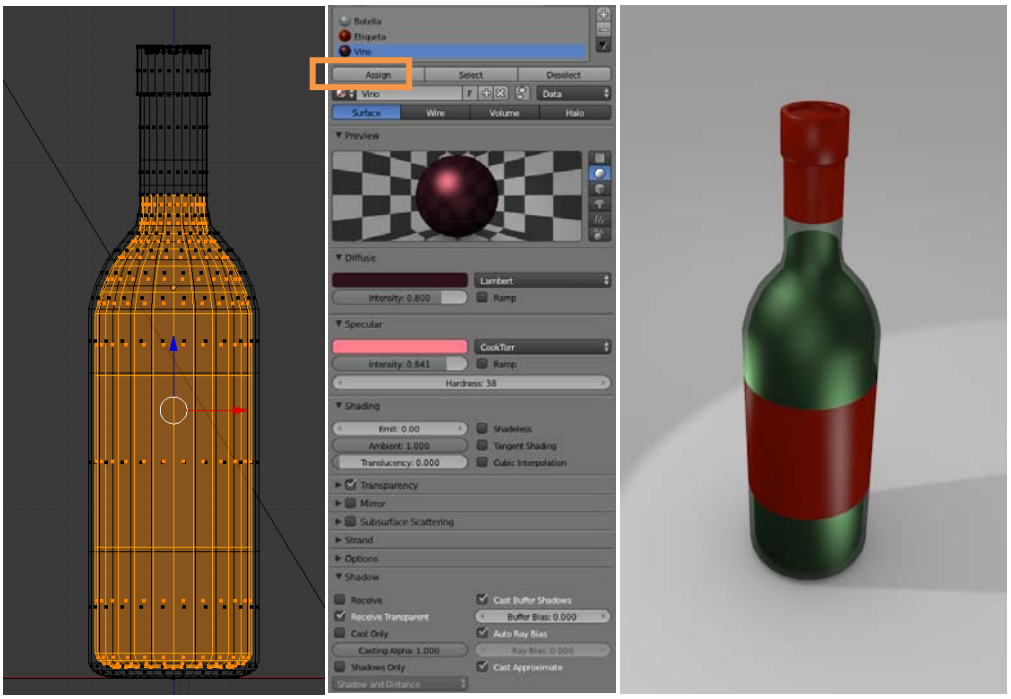


Ilustración 137 – Resultado final.

4.4 TEXTURIZADO

Con las herramientas disponibles en el editor de materiales, podemos definir las **características físicas básicas** de cada material y cómo les afecta la luz. Con estos ajustes, conseguimos materiales suaves y uniformes. Pero esto no es suficiente para definir la gran mayoría de los materiales, ya que existen otras características como **rugosidad**, cambios de **color**, distintas **intensidades** de brillo, **texturas**, **suciedad** y un largo etcétera, donde la falta de uniformidad es lo común.

Para poder definir todas estas propiedades y poder dotar a nuestros materiales de más **realismo**, necesitamos el uso de **imágenes reales y texturas**.

Las **textura**s, son un conjunto de imágenes superpuestas en **capas**. Cada una de ellas **modifica** una, o varias, de las **propiedades básicas** de nuestro material, añadiendo más matices a cada una de ellas. Podemos utilizar tantas texturas como necesitemos para definir correctamente un material. Lógicamente, cuantas más texturas utilicemos obtendremos materiales más pesados y mayores tiempos de render. [Guru, 2014]

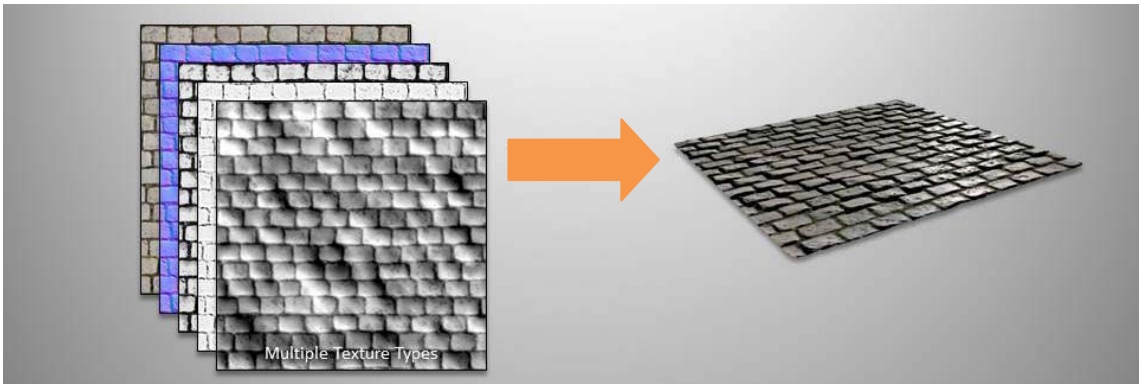


Ilustración 138 – Material generado mediante texturas..

En Internet, podemos encontrar millones de **bibliotecas de texturas** preparadas para aplicar a nuestros materiales. Pero cuando estamos definiendo el modelo de un edificio concreto como en este proyecto, en ocasiones, es mejor generar nuestras propias **texturas mediante fotografías** reales de nuestro edificio. El uso de bibliotecas lo dejaremos para materiales estándar como paredes interiores o falsos techos, donde además, es bastante difícil obtener una buena textura a partir de una fotografía.

4.4.1 Fotografiando texturas.

Cuando queremos crear un **material realista**, una de las texturas más importantes es la que modifica el **color difuso**, ya que será la que determine el color y el aspecto de nuestro material.

Por lo tanto, si queremos reproducir un material existente, tendremos que realizar una fotografía de dicho material. Para poder realizar una buena fotografía de un material, tenemos que tener en cuenta ciertos aspectos.

En primer lugar, tenemos que tener en cuenta qué es una textura y cómo trabaja. Una textura es una fotografía plana de un material, que se repetirá un número determinado de veces sobre la superficie aplicada. Por lo tanto, las características principales que le tenemos que exigir a la fotografía de la textura es que sea una fotografía **ortogonal, homogénea** y **repetible.**

Para realizar un **fotografía ortogonal** de un material, tendremos que colocarnos lo más **perpendicular** posible al mismo a la hora de realizar la fotografía. Este punto es de los más difíciles, ya que no siempre será posible colocarnos en dicha posición. Además debemos de intentar no acercarnos mucho, ya que nuestra fotografía se vería afectada por la **deformación esférica** del objetivo.

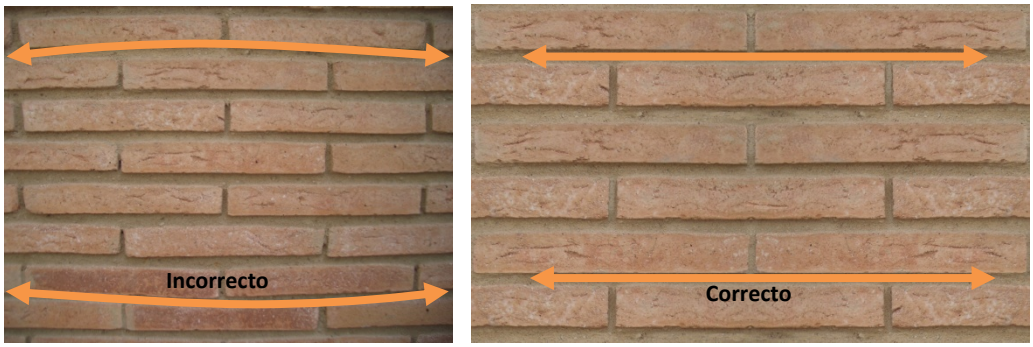


Ilustración 139 – Deformaciones de texturas.

Cuando no podamos realizar la fotografía de forma ortogonal, la realizaremos intentando que la **deformación por perspectiva** sea mínima. Finalmente, tendremos que modificarla posteriormente, mediante un programa de edición de imagen y utilizar un corrector de perspectiva.

El segundo aspecto, también es muy importante, la fotografía tiene que ser **homogénea**. Por lo tanto tenemos que **evitar** que en nuestra fotografía salgan **brillos** especulares, propios del material, cambios bruscos de **iluminación** o que una parte esté en **sombra** y otra no. De esta manera, será mucho más fácil conseguir hacer la textura repetible.

Como decíamos al principio, una textura es una fotografía que se repite, por tanto tenemos que asegurar la **continuidad** en sus extremos para conseguir un efecto continuo y homogéneo.

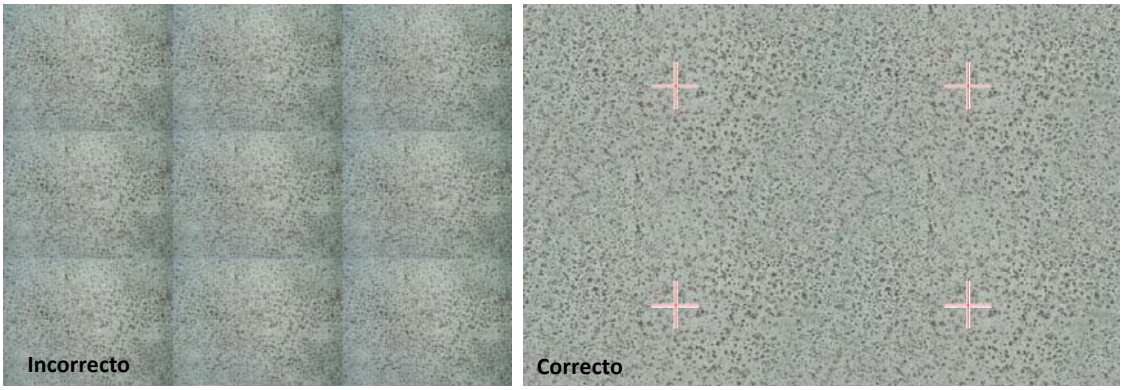


Ilustración 140 – Continuidad en las texturas.

Para conseguir una buena textura de repetición, necesitaremos manipularla mediante un editor de imágenes tipo Photoshop o similar. Vamos a ver un ejemplo de cómo conseguir una buena textura partiendo de una fotografía. [Vila,2014]

Ilustración 141 – Fotografía textura muro.

Para el ejemplo vamos a utilizar una de las texturas utilizadas en la ruinas de la Real Cárcel de Forzados. Se trata de la textura superior de dichas ruinas, que por su posición fue imposible realizar la fotografía de manera ortogonal. Por lo tanto, tendremos que realizar una corrección de la perspectiva y después asegurarnos de que funcionará como textura de repetición. Corregir la perspectiva, es un proceso sencillo. Solamente tenemos que seleccionar la herramienta "**Recortar con perspectiva**" y dibujar el área de la foto que queremos recortar.

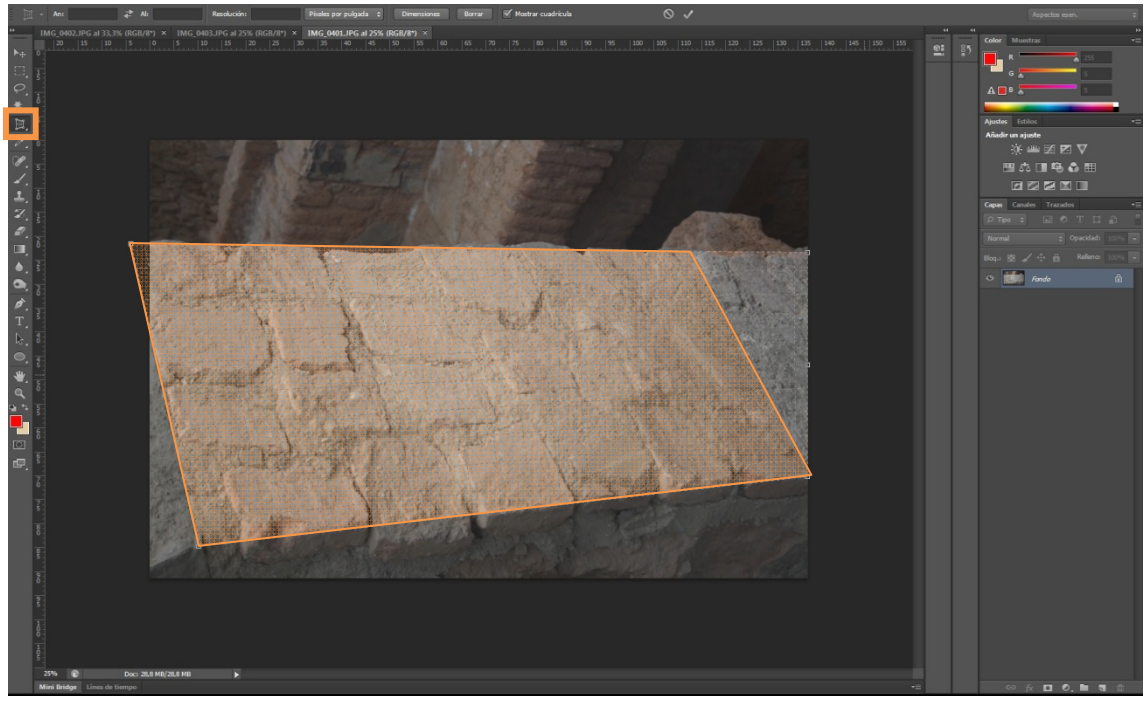


Ilustración 142 – Recortar con perspectiva.

Una vez seleccionada el área, aceptamos y obtenemos la textura en **proyección ortogonal**. Es difícil conseguir un buen resultado a la primera, ya que tenemos que conseguir dibujar la forma en perspectiva, por lo que seguramente necesitemos varios intentos antes de conseguir el efecto deseado.

Ilustración 143 – Textura en proyección ortogonal.

Ahora debemos comprobar, si la textura funciona como textura de repetición. Para ello aplicamos un **filtro desplazamiento** "Offset" con los siguientes valores:

Horizontal = "píxeles en x" /2 **Vertical** = "píxeles en y" /2

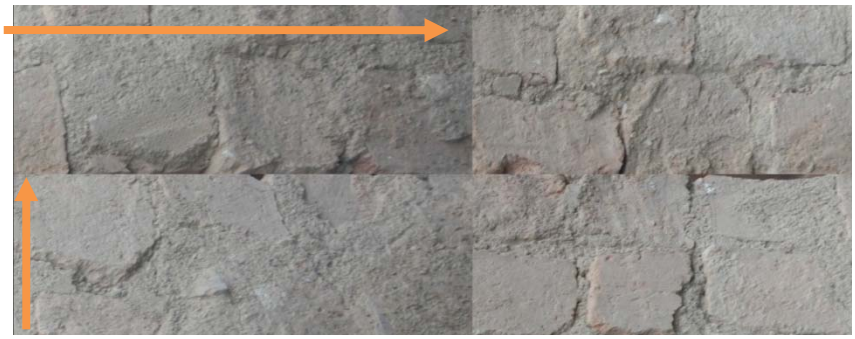


Ilustración 144 – Filtro desplazamiento.

Podemos apreciar a simple vista que la textura **no es repetible**. Pero ahora gracias al filtro desplazamiento podemos asegurar que es continua en los extremos de la fotografía. Por lo tanto solamente tenemos que arreglar la "cruz" que queda en la parte central. Podemos usar el **tampón de clonar**, o alguna herramienta correctora parecida, para ir copiando elemento de la imagen en la parte central y poco a poco hacer desaparecer ese corte tan brusco. El resultado sería algo así:

Ilustración 145 – Textura corregida.

Por lo tanto, nuestra textura ya tendría que ser repetible, al menos en cuanto al dibujo, ya que podemos observar a simple vista unos **cambios de tonalidad e intensidad lumínica todavía muy bruscos**. Para ver la apariencia que tendría la textura, aumentamos el tamaño del lienzo en un 200% y copiamos nuestra imagen 9 veces.

Ilustración 146 – Textura repetida.

Para solucionar esta variación tonal, vamos a realizar un **proceso por capas y canales** completamente editable. En primer lugar copiamos nuestra imagen a un archivo nuevo. A este archivo le aplicamos un modo **color Lab** (Imagen→Modo→Color Lab).

A simple vista este modo, no ha cambiado nuestra imagen, pero si vamos a la pestaña de canales, veremos por un lado la **información de color** (a/b) y por otro la **información de luminosidad** (Lightness). Ahora lo que tenemos que hacer es arrastrar este canal hasta nuestra imagen original.

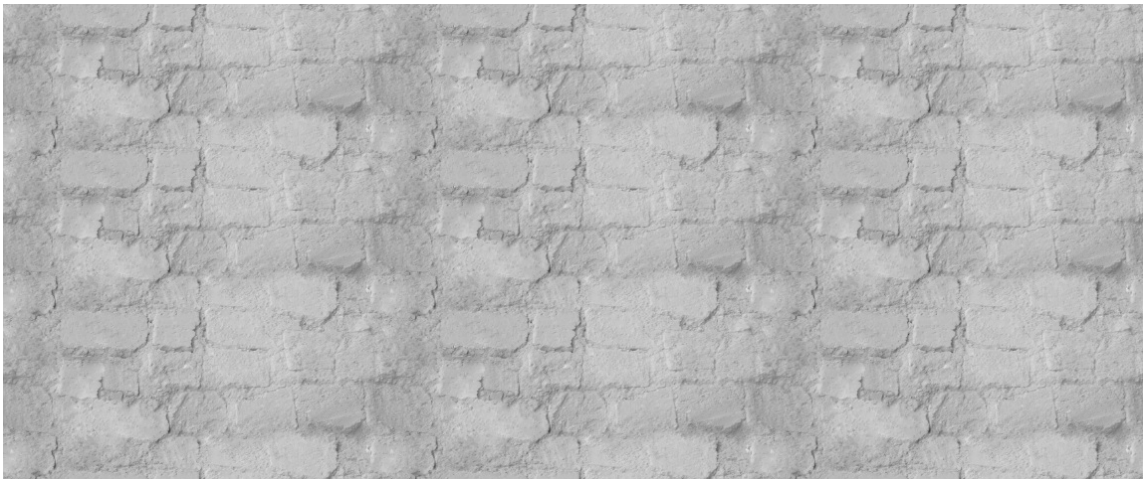


Ilustración 147 – Canal Luminosidad.

Ahora en los canales de la imagen original, tendremos un **nuevo canal Alfa** con esta información. A continuación seleccionamos el canal Alfa y le aplicamos un **desenfoque Gaussiano** con un valor bastante alto. La idea es conseguir que se pierda el detalle superficial.

Ilustración 148 – Desenfoque Gaussiano.

Una vez hecho esto, abrimos el ajuste de **Niveles** (Imagen→Ajustes→Niveles) y hacemos un ajuste **Auto.** Ahora podemos renombrar el canal Alfa con el nombre "**Selector de luces**". **Duplicamos** este canal e i**nvertimos** los colores de la copia y renombramos este canal como "**Selector de sombras**".

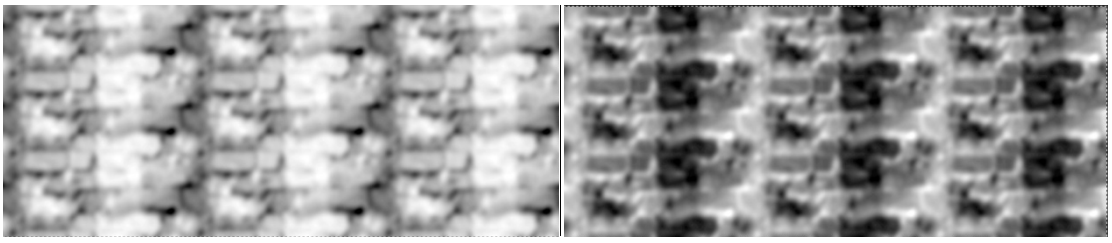


Ilustración 149 – Selector de luces / Selector de sombras

Volvemos a la paleta de capas y **cargamos el contenido** del canal selector de luces: Selección→Cargar selección→Selector de luces. Y le aplicamos una **capa de ajuste de niveles**, a esta capa podemos llamarla "**Niveles de luz**". Modificando los niveles hacemos que las zonas más claras, sean menos claras. Tenemos que repetir la operación cargando el contenido del canal selector de sombras, y nombramos la nueva capa de ajuste como "**Nivel de sombra**". Desde esta capa haremos que las zonas más oscuras, sean menos oscuras. De esta manera tenemos que intentar **igualar las luces y las sombras**, modificándolas lo menos posible, para que se igualen de un modo "equidistante".

Ilustración 150 – Luces y sombras igualadas.

Como vemos la imagen presenta un aspecto más homogéneo, pero hemos perdido algo de información del color en el proceso. Primero recortaremos la imagen a su tamaño original, y utilizando **ajustes correctores del color**, saturación, brillo y contraste, modificamos el resultado hasta obtener un color más parecido al original.

Finalmente este es el **resultado** de nuestra textura.

Ilustración 151 – Textura homogénea y repetible.

Una vez que tenemos nuestra textura de color difuso preparada, podemos generar las otras texturas que necesitamos de dos formas. Manualmente utilizando programas de edición de imagen o automáticamente utilizando programas específicos como CrazyBump.

4.4.2 Tipos de texturas.

Mediante texturas, podemos **manipular** muchos de los **parámetros básicos** de los materiales. En la imagen, vemos todos los parámetros que podemos configurar mediante texturas. Cuando insertamos una textura, tendremos que seleccionar aquí la **propiedad** a la que afecta, la **intensidad** de la propia textura y cómo se mezclará (**Blend**) con los parámetros básicos o con otras texturas.

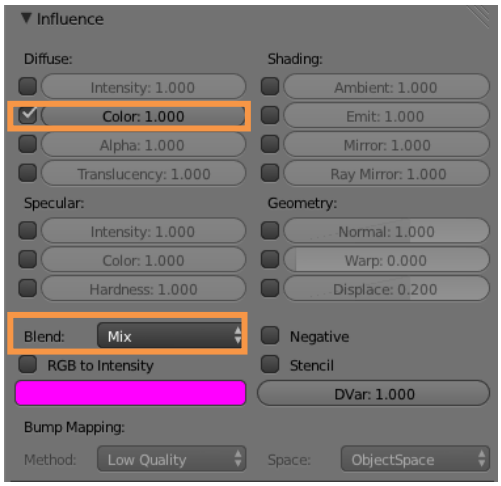


Ilustración 152 – Influencia de las texturas.

Podemos dividirlos en cuatro grupos según el parámetro que modifican:

Difusión:

- Controlar la intensidad del color difuso base - **Intensity** -
- Sustituir o mezclar el color difuso base - **Color** -
- Modificar la intensidad o las zonas transparentes -**Alpha**-
- Modificar la intensidad o las zonas translúcidas - **Translucency** -

Especular:

- Controlar la intensidad del brillo especular - **Intensity** -
- Sustituir o mezclar el color del brillo especular base - **Color** -
- Controlar la dureza del brillo especular - **Hardness** -

Sombreadores:

- Controlar la intensidad de la luz ambiente - **Ambient** -
- Modificar la intensidad o las zonas de emisión de luz - **Emit** -
- Sustituir o mezclar el color del reflejo - **Mirror** -
- Modificar la intensidad o las zonas reflectantes - **Ray Mirror** -

Geometría:

- Modificar las normales de una superficie - **Normal** -
- Modificar las coordenadas de las texturas en el canal - **Warp** -
- Desplazar una superficie - **Displace** -

Para agregar una **nueva textura** a un material, utilizaremos la pestaña "**Texture**" situada en la ventana de propiedades, junto a la pestaña de materiales. Si pulsamos sobre el botón "**New**" nos aparecerá una textura nueva. Al igual que con los materiales, es importante llevar un orden e ir nombrando la diferentes texturas que utilicemos.

Lo primero que tenemos que hacer es elegir el tipo de textura que queramos, normalmente "**Image or movie**", el resto de opciones son texturas incluidas en Blender que se generan de forma automática. Si pulsamos en **open**, podremos cargar en el programa la textura que queramos.

Desde aquí podemos modificar el color, brillo y contraste de la **imagen original** en "Colors". Seleccionar el **tipo de proyección** que queremos aplicar en "Mapping". Y seleccionar las **propiedades** donde va a actuar la textura.

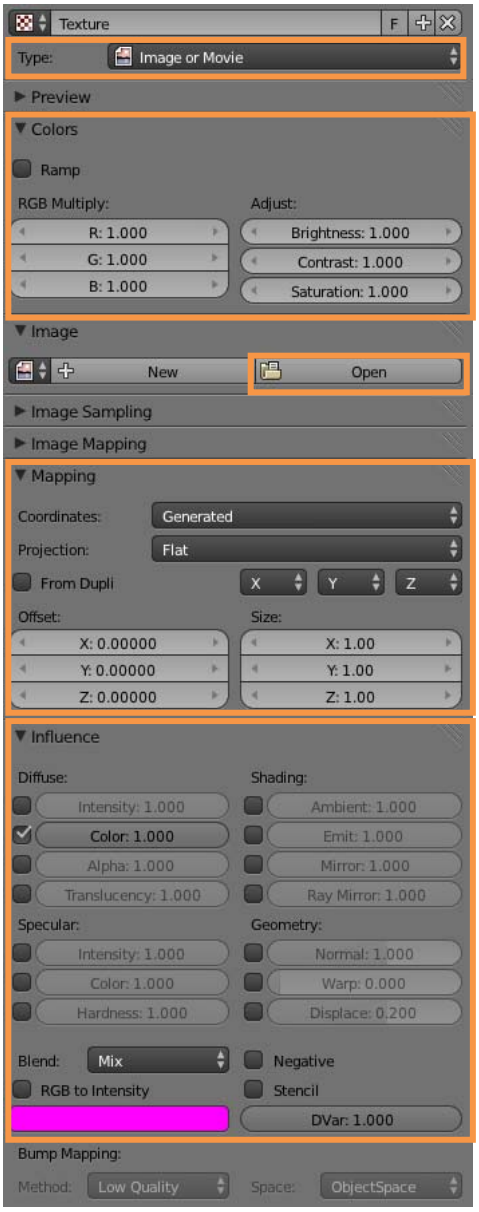


Ilustración 153 – Texture.

Como vemos, al poder incluir una o varias texturas en todos estos parámetros, podemos **complejizar** un material todo lo que queramos. A continuación vamos a ver alguno de los mapas de texturas más comunes en los materiales.

4.4.2.1 Color Difuso.

Es una de los más importantes, ya que será la textura que le dará **color** a nuestro material y donde se tienen que definir todos los **detalles** del mismo. Suelen ser **fotografías reales** del material, y de su calidad dependerá en gran grado el resultado final de un material. [Roosendaal, 2004]

Ilustración 154 – Texturas de color difuso utilizadas en el proyecto.

A parte de la textura de color del material, incluimos en este grupo otras texturas como las **texturas de suciedad**. Estas texturas añaden un realismo extra a nuestro material, ya que añaden imperfecciones, manchas, etc.

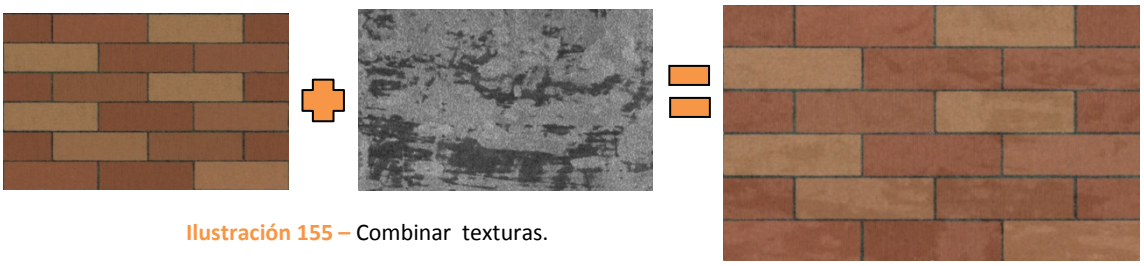


Ilustración 155 – Combinar texturas.

También podemos usas texturas de color para darle más realismo a **materiales metálicos**. Como el brillo especular depende de la posición del observador y del foco de luz, en ocasiones aparecerán como elementos sólidos poco realistas. Podemos utilizar una textura que le de aspecto metálico y simule los brillos especulares.

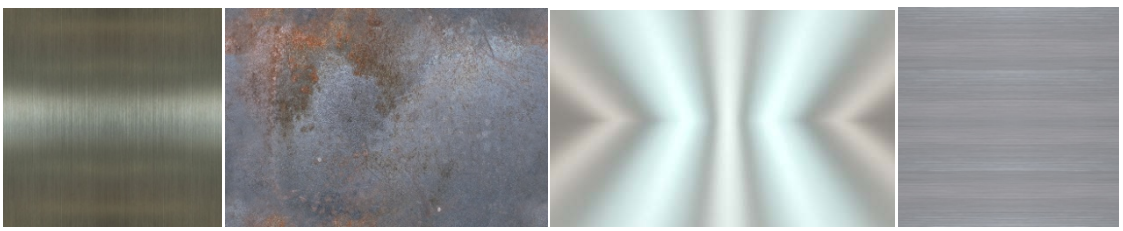


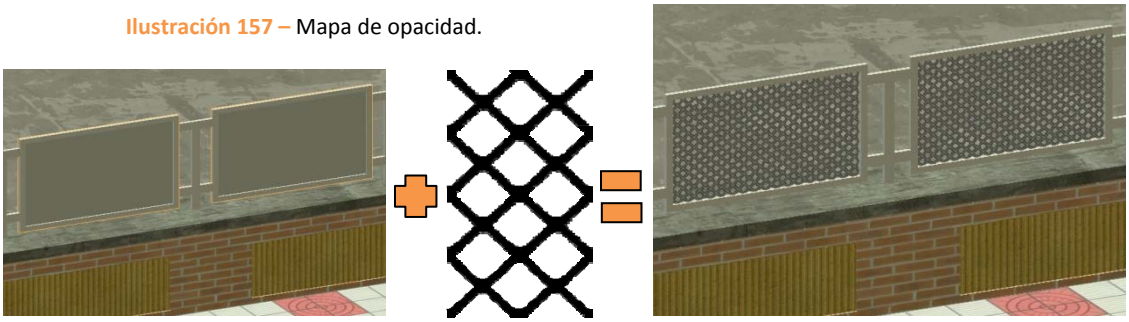
Ilustración 156 – Texturas metálicas.

4.4.2.2 Mapa de opacidad.

Utilizamos estas texturas para **definir partes** de nuestro material que son **transparentes**. Utilizamos imágenes ".png" que guardan la información alfa de la imagen. Cuando cargamos estas imágenes en el **canal alfa** del material, solamente afectará a las zonas que coincidan con el alfa de la textura. [Villar, 2014]

Por ejemplo, para una reja, podemos modelar todo el objeto, con el incremento de caras que esto supone, y aplicarle un material sólido. O podemos utilizar un plano para el modelo y definir los huecos mediante el uso de un mapa de opacidad.

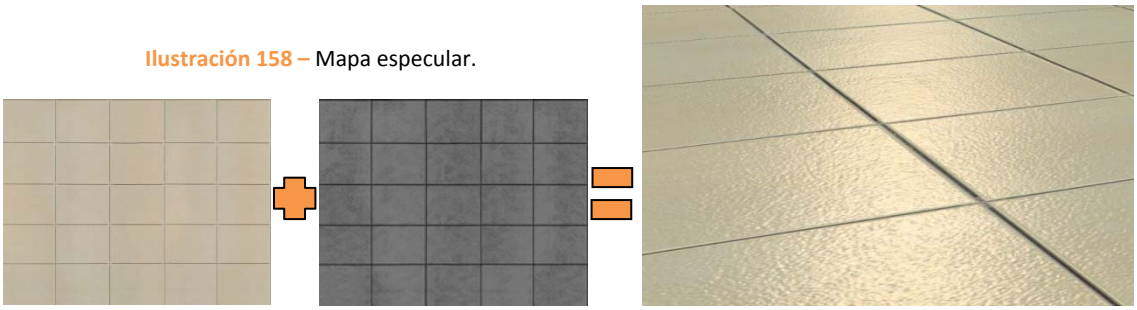
Ilustración 157 – Mapa de opacidad.



4.4.2.3 Mapa especular.

Son texturas que modifican la intensidad del **brillo especular**. Suelen ser imágenes iguales que las difusas pero en **escala de grises**. Esta imagen modifica la información de la intensidad del brillo de la superficie pixel por pixel. Donde **negro** es "0" y **blanco** es "1". Por tanto, el valor proporcionado por el color de cada pixel **multiplica** el valor básico de brillo especular. En el ejemplo del aplacado, vemos como las juntas están en negro, para que no brillen, mientras el aplacado presenta diferentes tonalidades para enriquecer el brillo especular. [Villar, 2014]

Ilustración 158 – Mapa especular.



4.4.2.4 Mapa de normales y Mapa de relieve.

Ambos mapas, el de normales y el de relieve (bump), añaden detalles a la textura de la superficie de un material en forma de **rugosidad**, haciendo que la geometría de la superficie parezca mucho **más detallada** de lo que en realidad es. La información del color de cada pixel de estas texturas modifica el **ángulo de salida** de la luz reflejada en ese punto, generando un **efecto visual** en el que parece que la superficie **sobresale** o queda **hundida** con respecto la posición de la superficie original.

[Flavell, 2010] [Roosendaal, 2004]

- Los **mapas bump** o de relieve, son textura en **escala de grises**, donde los tonos más claros representan la **altura** añadida a la superficie en ese punto. Este tipo de mapas, se pueden hacer fácilmente usando un programa de edición de imagen en 2D.

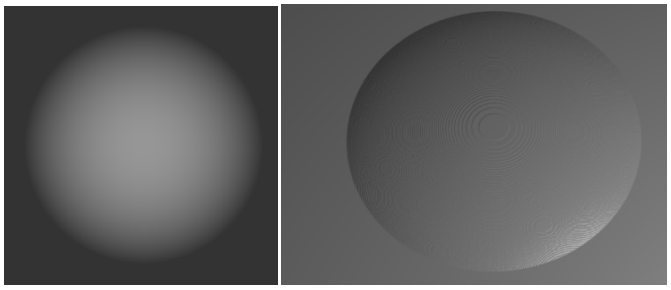


Ilustración 159 – Mapa de relieve "Bump".

- Los **mapas de normales** , son texturas **RGB**, en lugar de los datos de altura, el color de cada pixel representa el **ángulo de inclinación** de la superficie. Como tenemos la información de **tres colores**, el ángulo de salida de la iluminación se simula desde **tres ángulos** diferentes. Aunque se puede conseguir el mismo efecto con los dos mapas, los mapas de normales son físicamente más **exactos**.

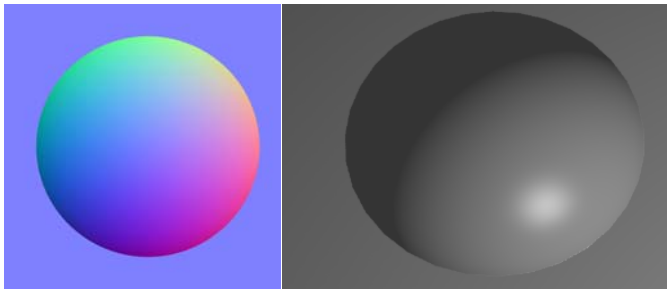


Ilustración 160 – Mapa de normales.

4.4.2.5 Mapa de desplazamiento.

Los mapas de desplazamiento, realizan la misma acción que los dos anteriores. Son texturas en **escala de grises**, donde el color del pixel indica la **altura** del punto. La diferencia con los mapas de relieve, es que no es un efecto mediante el cálculo de la luz, sino que realmente **desplaza el punto**. Para poder desplazar el punto, necesitamos que la **superficie** esté **subdividida**. Cuantas más subdivisiones tenga la cara, mejores resultados obtendremos. El problema es que para alcanzar una calidad razonable, necesitamos muchas subdivisiones, lo que implica un gran número de caras que ralentizará mucho el proceso de cálculo del render. [Guru, 2014] [Simond, 2012]

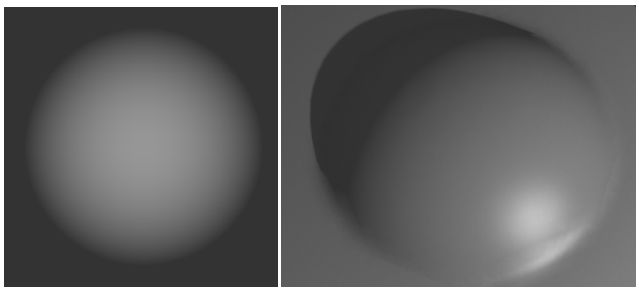


Ilustración 161 – Mapa de desplazamiento.

4.4.2.6 Ejemplo material con textura.

Vamos a ver las texturas utilizadas en el suelo del patio de la EIMIA. Para este material, vamos a utilizar un mapa **difuso** de color, un mapa difuso de **suciedad**, un mapa **especular**, y un mapa de **normales**.

Si vemos el aspecto del material, solamente con el mapa de color, observamos que su aspecto es muy plano y que el brillo especular afecta a todas las partes por igual.

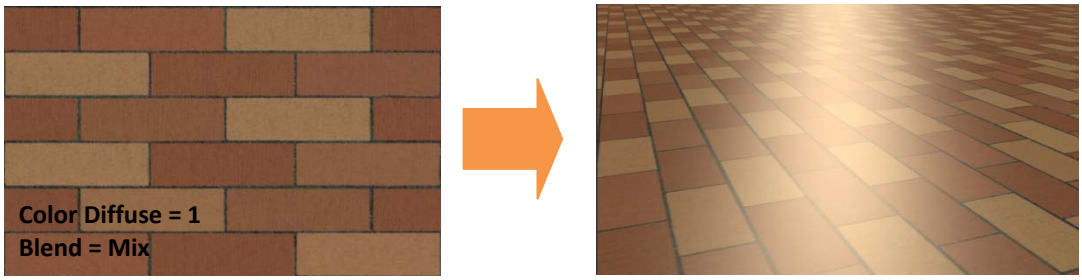


Ilustración 162 – Mapa de color difuso.

Vamos a corregir primero el brillo especular. Si utilizamos el mapa especular, para modificar tanto el color del brillo como la intensidad, vemos como las baldosas comienzan a brillar más en su perímetro, mientras que las juntas apenas brillan. Pero el aspecto, sigue siendo muy plano.

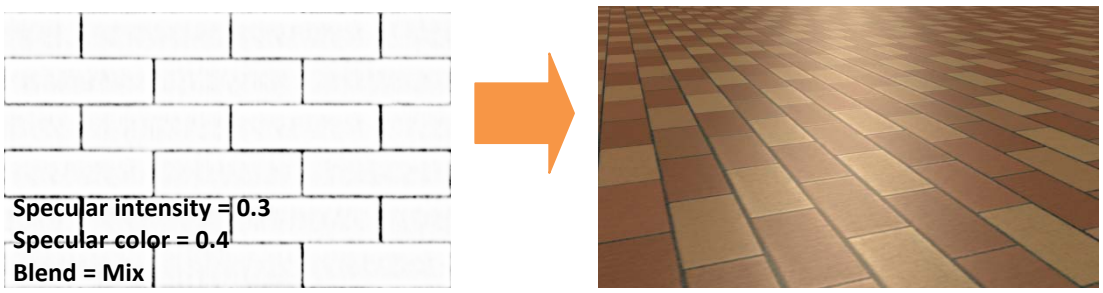


Ilustración 163 – Mapa especular.

Si añadimos el mapa de normales, para modificar la geometría del material, vemos como su aspecto es mucho más realista ya que las juntas parecen hundirse.

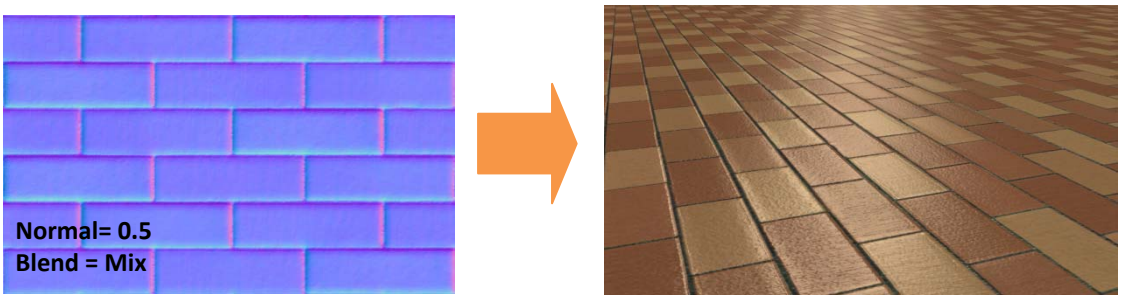


Ilustración 164 – Mapa normales.

Por último, ya que este material tiene que cubrir una superficie muy grande, aplicaremos un mapa de suciedad para añadirle más variaciones y que parezca menos homogéneo. Lo aplicaremos en color difuso y en el brillo especular.

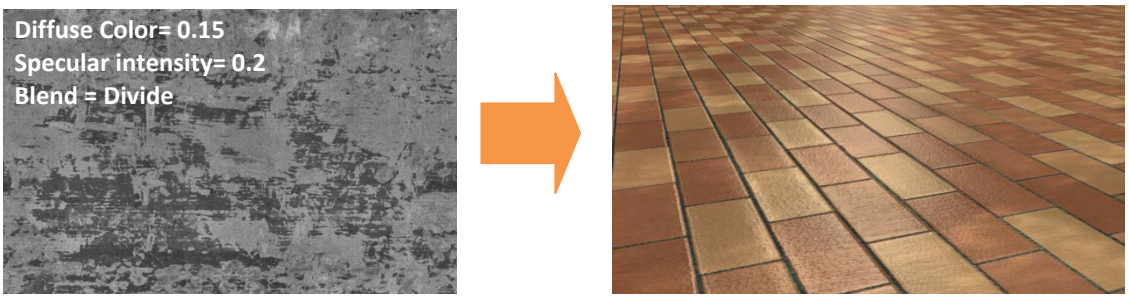


Ilustración 165 – Mapa suciedad.

4.4.3 Mapeado de una textura.

El mapeado de una textura, es la forma en el que **la imagen se proyecta sobre el objeto** en el que está aplicado. Podemos seleccionar el método de mapeado en la pestaña de textura, en el menú "Mapping". Vamos a ver dos de las técnicas más utilizadas. El **método estándar** o generado y el **mapeado UV**.

El método más adecuado para un modelo determinado, dependerá de la forma general del objeto. [Flavell, 2010] [Roosendaal, 2004]

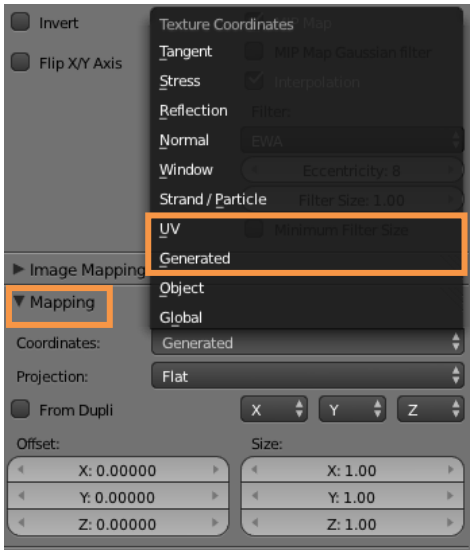


Ilustración 166 – Mapping I.

4.4.3.1 Método general.

Con el método general, proyectamos la textura bidimensional sobre el objeto tridimensional, con una forma establecida. Tenemos cuatro tipos diferentes de proyección estándar: **Flat** (plano), **Cube** (cúbico), **Tube** (tubular) y **Sphere** (esférico).

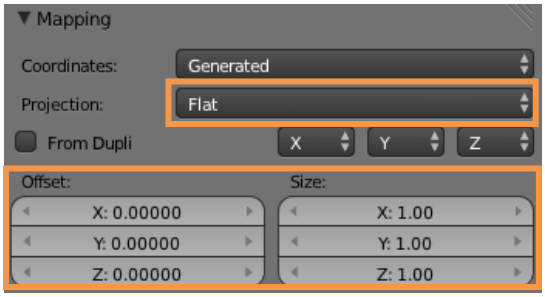


Ilustración 167 – Mapping II.

Una vez elegido el método de proyección, podemos **mover** el origen de la textura modificando los valores de X, Y y Z que aparecen en las barras deslizadoras en la opción "Offset". También podremos **escalar** el tamaño en la opción "Size" donde podemos introducir el número de veces que se repite la textura en cada uno de los ejes. A valores grandes, la textura será más pequeña. [Roosendaal, 2004]

- **Mapeado Plano**:

  Es el mejor método para texturizar caras planas simples. La textura se proyecta paralela a un plano, el problema es que cuando la aplicamos a un volumen, se proyectará sobre una de sus caras, y el último pixel de esa cara se copia a lo largo de las caras perpendiculares. También produce efectos interesantes sobre las esferas.

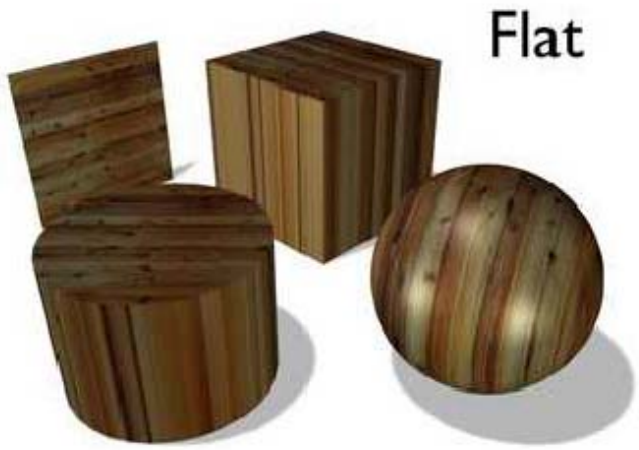


Ilustración 168 – Mapeado Plano.

- **Mapeado Cúbico**:

  El mapeado cúbico, proyecta la textura desde tres planos ortogonales. Normalmente da los mejores resultados si se utiliza sobre paralelepípedos y objeto no demasiado curvos u orgánicos. Es un bueno método de proyección de texturas, aunque a veces nos puede dar problemas de continuidad del material en algunas de sus caras.

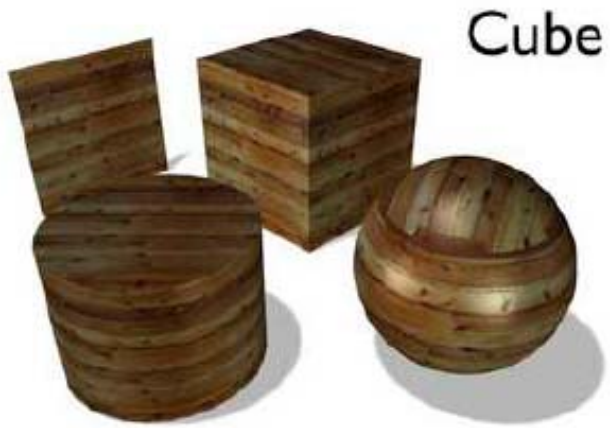


Ilustración 169 – Mapeado Cúbico.

- **Mapeado Tubular:**

  El mapeado tubular, proyecta la textura en forma de cilindro alrededor del objeto, como una etiqueta en una botella. Normalmente sólo obtendremos buenos resultados en cuerpos cilíndricos o esféricos. El principal problema de esta proyección, son las caras superiores e inferiores, donde los píxeles del perímetro se repite de forma radial.

Ilustración 170 – Mapeado Tubular.

- **Mapeado Esférico:**

  El mapeado esférico proyecta la imagen con forma de esfera alrededor del objeto, como si de un mapa del planeta tierra se tratase. Obtenemos los mejores resultados, obviamente, al aplicarlo sobre esferas. Pero podemos obtener efectos interesantes sobre cilindro y objetos orgánicos.

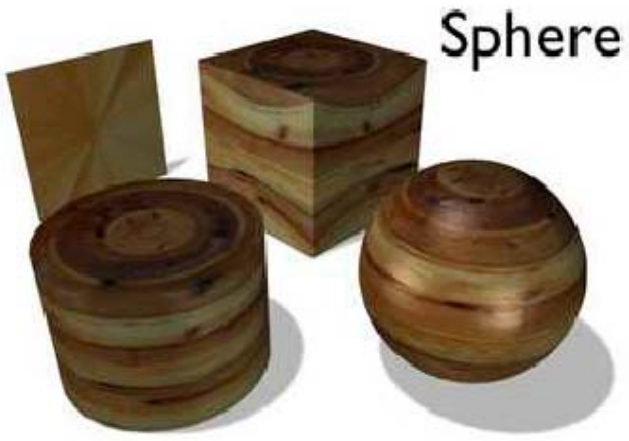


Ilustración 171 – Mapeado Esférico.

4.4.3.2 UV mapping.

El sistema de mapeado UV es el método más potente de mapeado, ya que tenemos **control absoluto** de la posición de cada una de nuestras superficies con respecto a nuestra textura. El proceso es justo al contrario que en el método general, ya que, en vez de proyectar una imagen 2D sobre un objeto 3D, **proyectamos las caras** de nuestro objeto 3D sobre un plano y lo **superponemos en la imagen** 2D. [Flavell, 2010]

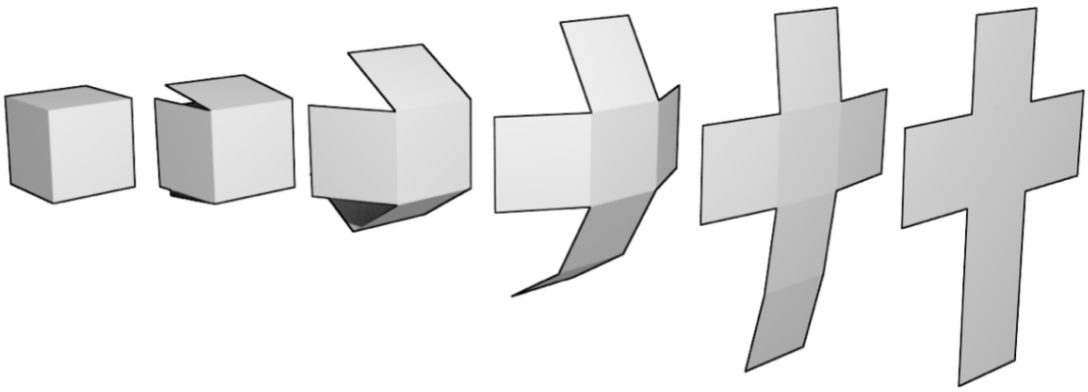


Ilustración 172 – Concepto de mapeado UV.

Como vemos en la figura anterior, consiste básicamente en hacer un desplegable de nuestro volumen. En primer lugar vamos a ver cómo sería el proceso manualmente, para poder controlar todos los aspectos de esta herramienta.

Vamos a aplicarle una textura de baldosa cerámica a un cubo. Para ello utilizaremos una escena con un cubo y un material. El material tiene una textura en el color difuso y con coordenadas de mapeado UV.

En primer lugar, para poder visualizar en tiempo real la textura, necesitamos activar el modo de visualización "**texturizado**". Para ello, utilizaremos la combinación de teclas rápidas "Alt + Z". En la barra desplegable de la derecha, dentro de la ventana 3Dview, en las opciones de visualización, tenemos que activar en **Shading** el **modo GLSL**.

Para este proceso, es conveniente dividir la pantalla en dos ventanas de trabajo, una parte para el **3Dview** y otra para **UV/image editor**.

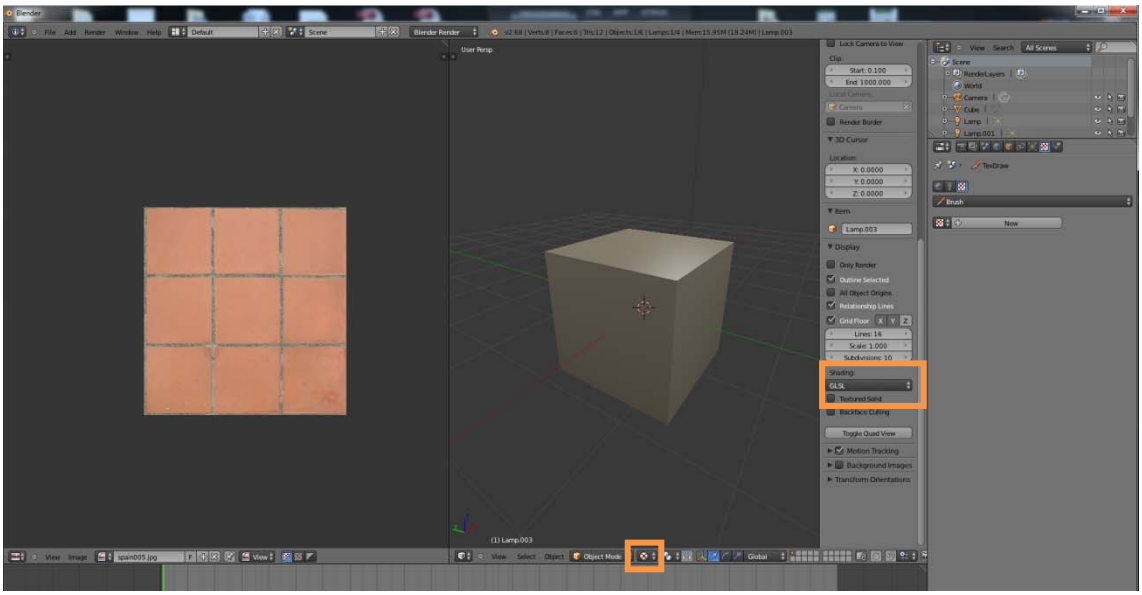


Ilustración 173 – UV/image editor y 3Dview.

Lo que tenemos que hacer en primer lugar, es **marcar las costuras** por las que vamos a desplegar el cubo. Según el esquema del desmontable, necesitamos tres costuras en cada base y una costura vertical.

Entramos en **modo edición** y seleccionamos las aristas correspondientes. Una vez seleccionadas, abrimos el menú de "edge" mediante las tecla "Crtl + E" y seleccionamos "**mark seam**". Ahora cuando generemos el mapa UV, estas serán las costuras por donde desplegaremos el cubo.

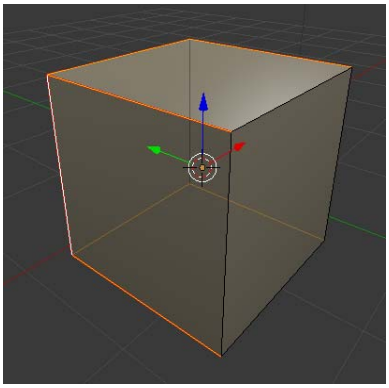


Ilustración 174 – Mark seam.

Para generar el mapa UV, seleccionamos todas la caras y pulsamos la tecla "U". y nos aparece el menú "**UV maping**" donde podremos ver las diferentes opciones para general el mapa UV. Utilizaremos la primera opción, **Unwrap**. Una vez hecho esto veremos cómo nos aparece el desplegable del cubo sobre nuestra textura. El **mapa UV** generado, puede ser **movido**, **escalado**, **rotado**, etc. como si de cualquier otro objeto se tratase. De esta forma podemos ajustarlo donde queramos con respecto de la textura. [Villar, 2014]

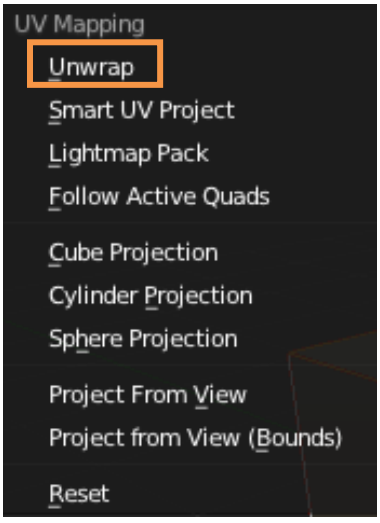


Ilustración 175 – Unwrap.

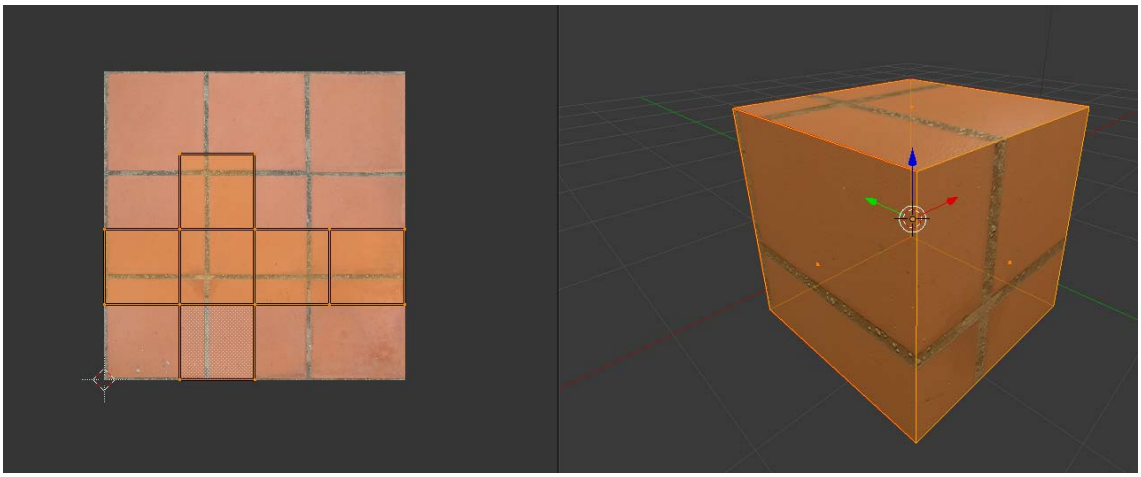


Ilustración 176 – Mapa UV.

A continuación, vamos a ajustar el mapa UV, de manera que cada uno de las caras ocupe la totalidad de la textura, para ello, escalamos y movemos nuestro UV. En la imagen vemos como la textura queda perfectamente colocada.

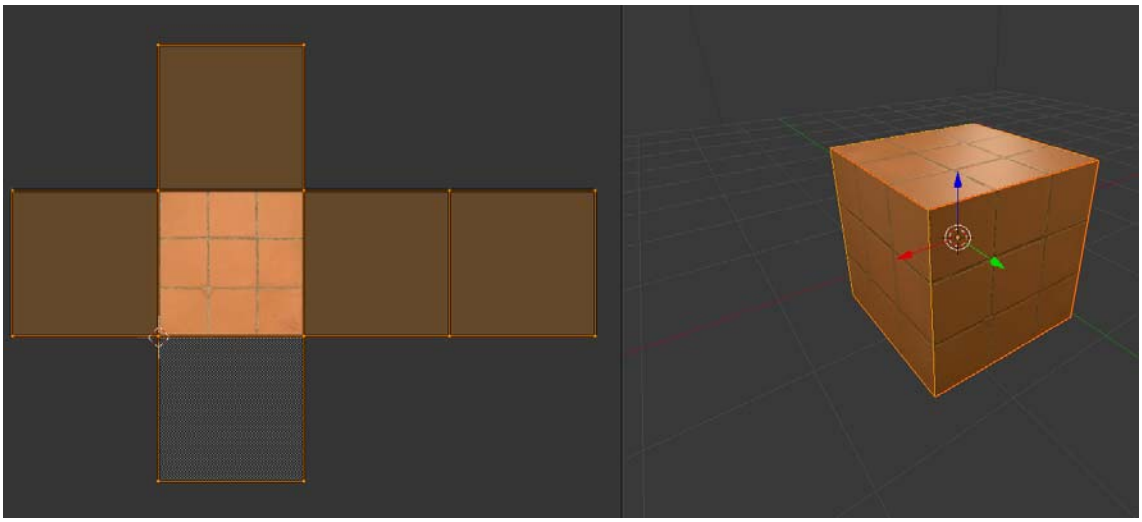


Ilustración 177 – Mapa UV escalado.

Dependiendo de la geometría del objeto que estemos mapeando, podemos **simplificar** el proceso. Si vemos las opciones del menú "UV mapping" tenemos tres tipos de **proyecciones**: Cúbica, Cilíndrica y Esférica. Utilizando estas opciones podemos realizar la proyección de nuestro objeto, sin necesidad de marcar las costuras. Aunque el que mejores resultados ofrece de los tres sin marcar las costuras es la **proyección cúbica.**

En el proyecto, en la gran mayoría de los objetos, tabiques, fachadas, ventanas, etc. , sería muy complicado decidir todas las costuras necesarias. Por lo tanto se han realizado utilizando "**Cube projection**". De esta forma, todas las caras son proyectadas en las tres direcciones ortogonales. El único problema es que las proyecta una encima de otras, por lo que en ocasiones, necesitamos ajustar algunas de ellas para asegurar la continuidad del material en todas sus caras.

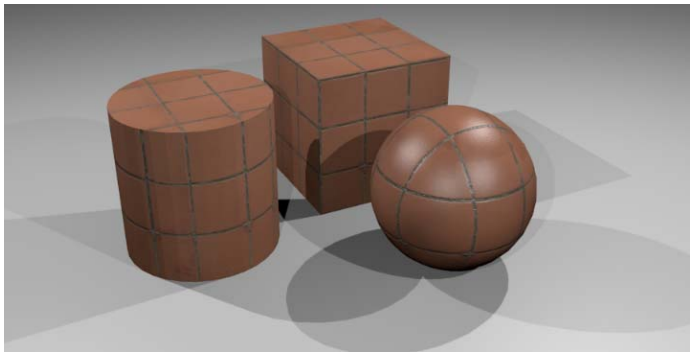


Ilustración 178 – Mapa UV proyección cúbica.

4.4.4 Modelo texturizado.

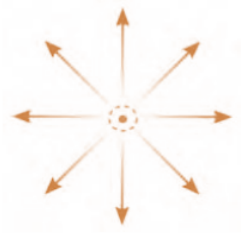
Una vez aplicadas todas las texturas, este es el aspecto que presenta nuestro modelo. Ya solamente necesitaríamos colocar las cámaras y las luces para comenzar el proceso de renderizado.

Ilustración 179 – Modelo texturizado.

4.5 ILUMINACIÓN

La iluminación de nuestra escena, es el punto crucial de la configuración del render. De la iluminación va a depender la **calidad de nuestra imagen**, ya que de ella dependen los brillos, las sombras, etc.

En primer lugar, vamos a ver los diferentes **elementos de iluminación** que tenemos disponibles en Blender. [Flavell, 2010]

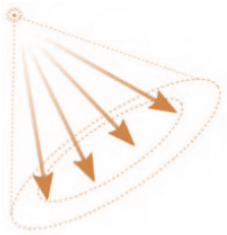


Point

Esta luz emite en **todas direcciones** desde un único **punto**. Podemos simular una vela, una bombilla, etc.

Sun

Es una **luz direccional**. Inunda de luz una escena en un **ángulo** dado. Es independiente de su posición.



Spot

Similar a una linterna, desde un punto emite un **cono de luz** que arroja un círculo sobre la superficie donde está dirigida.

Hemi

Se puede utilizar como **luz ambiental**. La luz es emitida en una **dirección principal**, pero está acompañada de **rayos secundarios** en forma de hemisferio. **No genera sombras**.

Area

La luz es emitida por una **superficie** en una **única dirección**. El tamaño de la superficie podemos configurarlo nosotros.

Aunque cada una de ellas posee características diferentes, tienen una serie de parámetros en común.

Podemos elegir el **color** de la luz, esto muy útil para poder simular diferentes tipos de luz y generar diferentes ambientes en nuestras escenas, como un amanecer, un atardecer, etc.

También podemos configurar la **intensidad** de emisión. Este valor dependerá mucho del tipo de luz, de su tamaño en el caso de las luces superficiales y del tamaño de nuestra escena. Por lo tanto no existen unos valores claros de referencia.

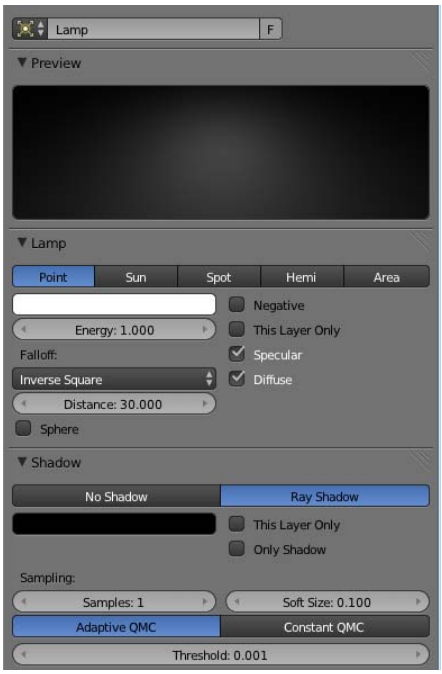


Ilustración 180 – Lights.

Además, podemos elegir la **distancia** desde el foco emisor donde comienza a desvanecerse. Esta opción no la encontramos en la luz "sun", ya que se considera infinita en distancia y superficie.

Una opción bastante interesante, es que podemos elegir si queremos que afecte solamente al color **especular**, al color **difuso** o ambos. Esto es muy útil en las luces ambientales, ya que a veces nos crean reflejos que no nos interesan en una escena determinada.

Por último, tendremos que configurar el tipo de **sombra**. Normalmente seleccionaremos el método de trazado de rayos "**Ray shadows**". Desde el panel de sombra podemos elegir el **color** y la **calidad**. La calidad de la sombra depende del número de "**samples**" que utilicemos para su cálculo y del método de cálculo, cuantos más samples más calidad y mayor tiempo de render. El modo "Adaptive QMC", es más lento pero ofrece mayor calidad, mientras que el método "Constant QMC" es más rápido pero puede producir ruido en la sombra.

Además también podemos configurar el borde de **penumbra** de la sombra con el parámetro "**soft size**".

4.5.1 Cámaras.

Antes de seguir con la iluminación, tendremos que colocar las cámaras en la posición desde donde vamos a realizar la imagen del modelo. Utilizaremos dos cámaras, una para comprobar la iluminación exterior y otra para la iluminación interior de una de las aulas.

Para añadir una nueva cámara, seguimos el mismo procedimiento que para los objetos, "Add → Camera" . Podemos manipular la cámara de la misma manera que cualquier objeto, moviendo y rotando. De esta forma colocamos la cámara en la posición aproximada desde donde queremos realizar la imagen.

Una vez colocada la cámara, tenemos que seleccionarla como **cámara activa**, desde la que se va a realizar el render. Para ello, con la cámara seleccionada pulsamos "Crtl+0", ahora cuando pulsemos la tecla "0" en el teclado numérico, nos cambiara la vista de la ventana 3dview a la posición de la cámara.

Una vez que estemos en vista de cámara, vemos un rectángulo que representa el área del modelo que se verá en la imagen. Podemos mejorar el encuadre, si con la cámara seleccionada pulsamos "G" y modificar el ángulo de la vista, rotando sobre un eje pulsando "R" o rotar la cámara sobre sí misma pulsando dos veces la tecla "R".

Ilustración 181 – Camera

En la **pestaña cámara** de la ventana propiedades, podemos modificar los aspectos básicos de nuestra cámara. Desde aquí podemos elegir entre una cámara en **perspectiva** o una cámara **ortográfica**, modificar el **angular** del objetivo y modificar la distancia del **plano cercano y lejano**. [Flavell, 2010]

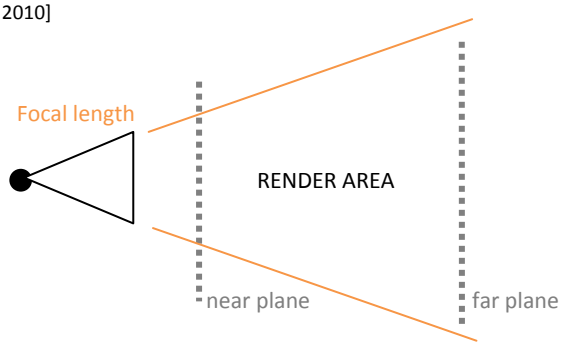


Ilustración 182 – Zona de render

4.5.2 Iluminación Exterior.

El sistema de iluminación utilizado por el motor interno de Blender, es un sistema de **Iluminación Local** [ver 3.6.8] . Por lo tanto tendremos que simular la luz global mediante otro sistema.

Para las pruebas de iluminación, sustituiremos todos los materiales de la escena por un **material** preparado para **testear la luz**. Este material será de un tono blanco-gris con un valor en el brillo especular muy pequeño del orden de 0,1 a 0,2. En la ventana propiedades, en la pestaña "Layers", podemos sustituir todos los materiales de la escena por el nuevo material en la opción Material.

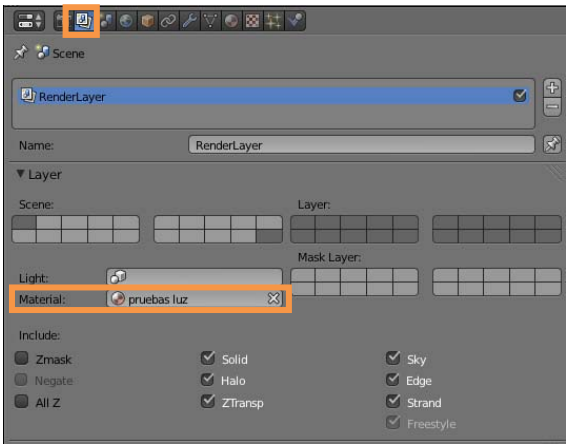


Ilustración 183 – Layer

Para la iluminación exterior, utilizaremos un **sol** como fuente de **luz principal**, y un sistema secundario de **iluminación global**, bien utilizando otras fuentes de luz o mediante la herramienta **oclusión de ambiente** de Blender. [Guru, 2014]

Empezaremos añadiendo un sol a nuestra escena, para ello utilizaremos el proceso de siempre "Add→Lamp→Sun". Y lo colocamos con la herramienta mover y rotar en la posición deseada. Como veíamos en la tabal anterior, lo importante no es la posición si no el **ángulo**.

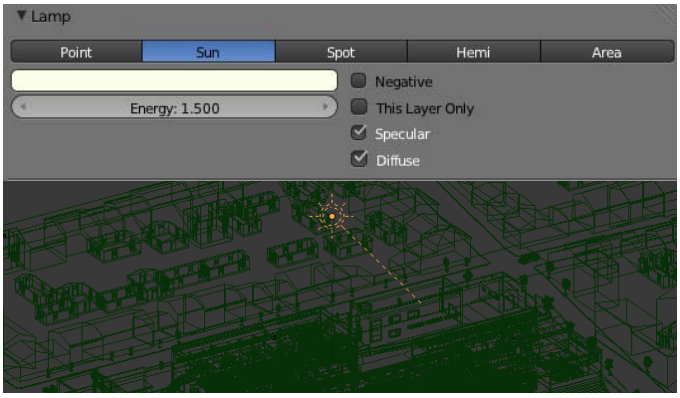


Ilustración 184 – Sun

La luz del sol, es una luz intensa y con un color ligeramente amarilla, por lo tanto seleccionaremos un color blanco pero con un tono amarillento. En cuanto a la intensidad, utilizaremos un valor de 1,5. Las sombras las calcularemos con ray shadows con un sampling de 12.

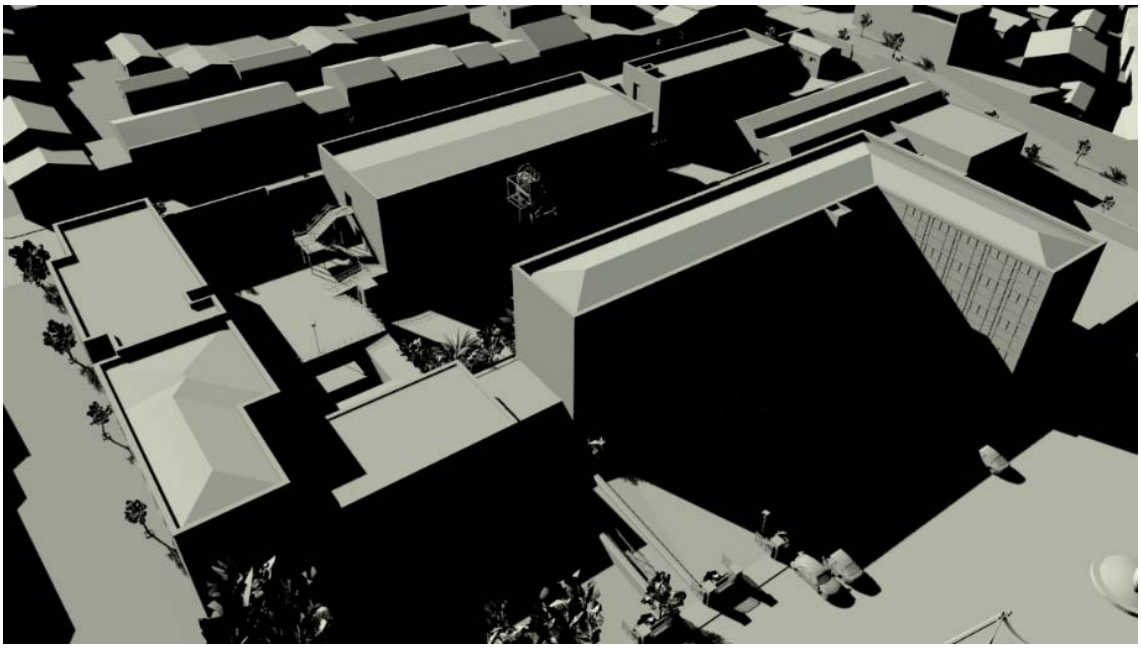


Ilustración 185 – Iluminación local

Como podemos observar en la imagen anterior, solamente están iluminadas las zonas que reciben **iluminación directa**. Por lo tanto ahora vamos a ver cómo **simular la luz global**. Para ello, tenemos que generar un "nuevo mundo".

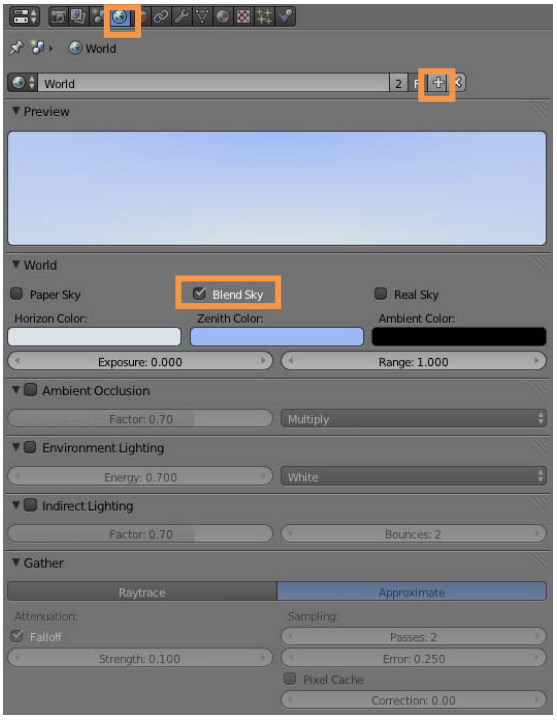


Ilustración 186 – World

En la ventana propiedades, tenemos una pestaña llamada "**world**". Si pulsamos en el icono "+", crearemos un nuevo mundo.

Desde esta pestaña, podemos elegir el color del cielo, utilizando un Blend Sky, que nos hará un degradado entre "Horizon color" y "Zenith Color".

Ahora tenemos tres opciones que vienen desactivadas por defecto: Ambient oclussion, Enviroment Lighting y Inderect Lighting.

La **oclusión ambiental**, es un método de cálculo de sombras polígono a polígono, simula las sombras que producen unos objetos sobre otros por cercanía. Activaremos esta opción para enriquecer las sombras, con una intensidad del 0,7. Y en el menú desplegable de la derecha seleccionaremos "Multiply" para que oscurezca esas zonas. Al activar esta opción, automáticamente se desbloque el panel Gather, donde seleccionaremos, el método del Raytrace. Aunque podemos reducir bastante el tiempo de render utilizando el modo Aproximado que realiza menos cálculos.

La **luz ambiental**, genera una luz homogénea para toda la escena, no se trata de un cálculo de luz mediante rebotes, simplemente suma una cantidad de luz fija a la calculada por el Raytrace. De esta forma, nunca tendremos superficies en la oscuridad absoluta como ocurría antes con la iluminación local. El color de esta luz será blanco y le aplicaremos una intensidad de 0,7.

Con la **luz indirecta** podemos calcular los rayos de luz que rebotan en un objeto y llegan a los objetos próximos. Podemos controlar el número de rebotes y la intensidad de los rayos reflejados. Por ahora esta opción la dejamos desactivada, ya que ralentiza mucho el tiempo de render y más interesante para la iluminación de interiores.

Si desactivamos momentáneamente el sol, veremos los cambios que produce esta configuración en nuestra escena.

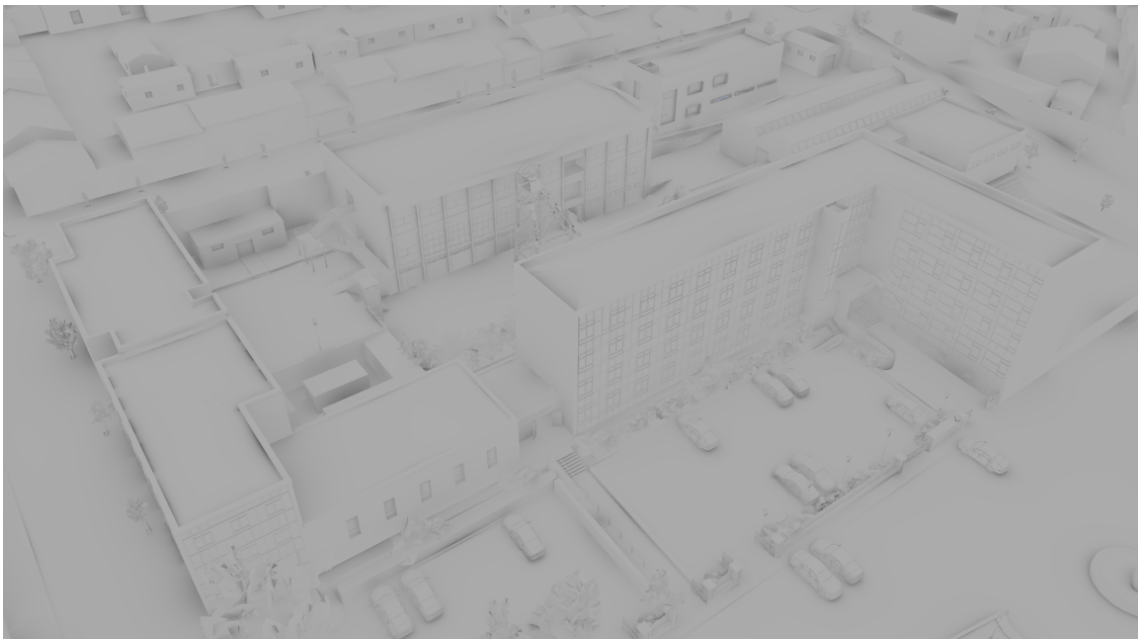


Ilustración 187 – Simulación de luz ambiental

Ahora volvemos a activar el sol y renderizamos para combinar las dos imágenes. Ahora tendremos que tener un efecto de **luz global**:

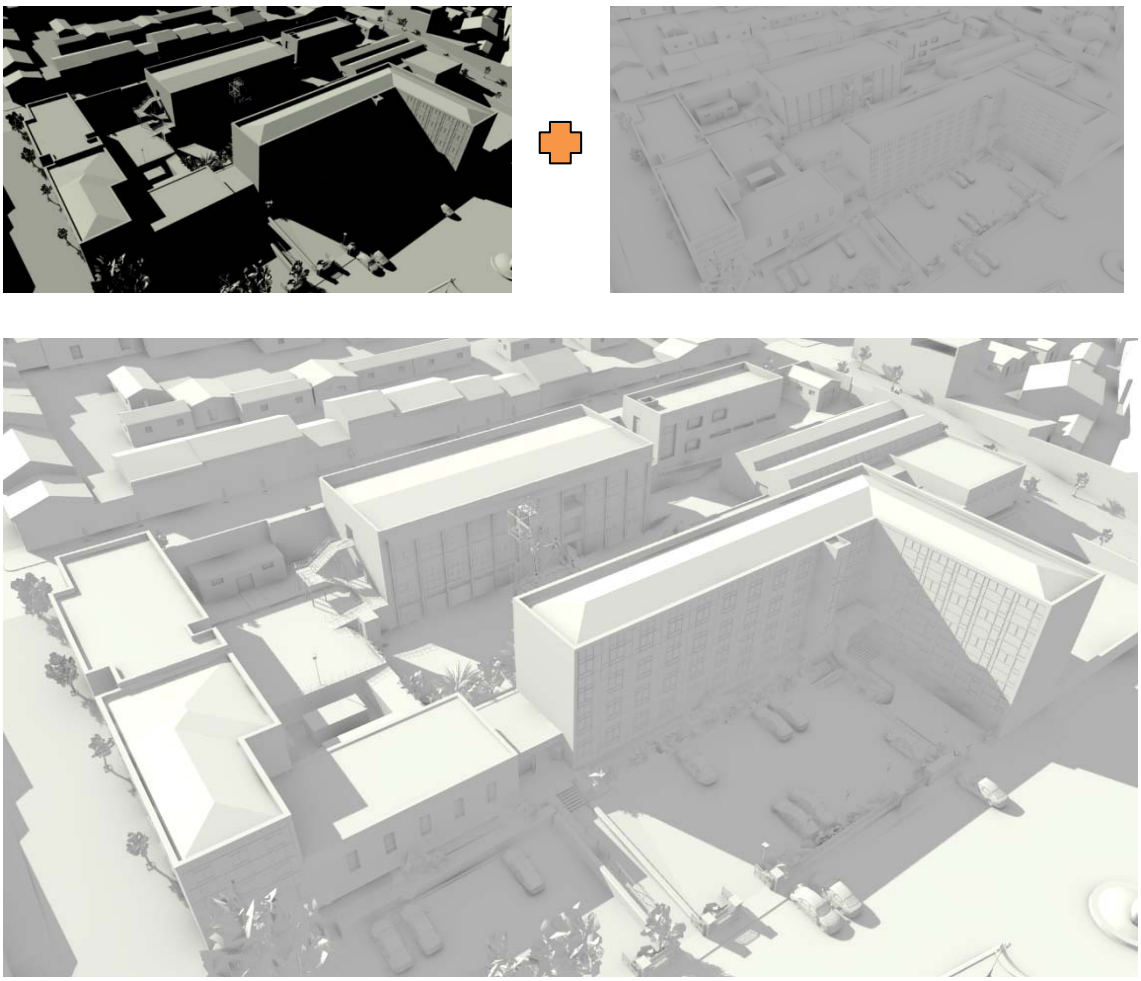


Ilustración 188 – Simulación de luz global

Una vez obtenida la simulación de luz global, podemos mejorar los resultados mediantes el uso del **compositor de nodos**. Cuando abrimos el compositor de nodos y activamos la opción "use nodes", veremos la siguiente pantalla: [Flavell, 2010] [Guru, 2014]

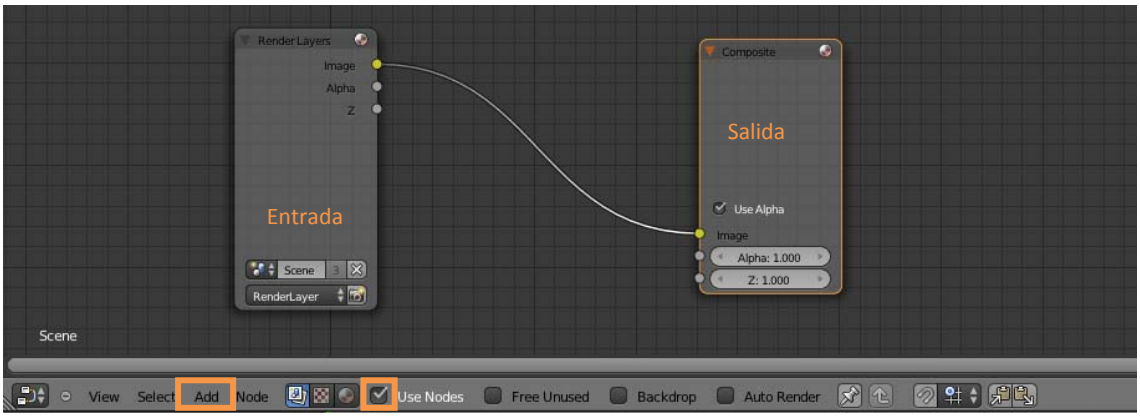


Ilustración 189 – Compositor de Nodos

El uso del compositor de nodos, nos permite retocar las imágenes como si de un programa de edición de imagen se tratase. Además nos permite crear una gran diversidad de efectos e incluso mezclar escenas diferentes.

La base de funcionamiento es muy sencilla. Tenemos un canal de **entrada** "Render Layers" y otro de **salida** "Compositor". Como vemos, estos dos nodos están unidos por una línea que los conecta, por lo que la información de salida es igual a la de entrada. Nosotros tendremos que añadir **nodos intermedios** en esa línea para modificar la imagen inicial, como filtros, modificadores de color, distorsiones, etc.

Para añadir un nuevo nodo lo haremos desde el menú desplegable "Add". Aunque la base del funcionamiento es sencillo, la cantidad de herramientas y opciones que posee hace que sea uno de los módulos más potentes y más complejos de Blender.

La principal ventaja que tiene el retocar las imágenes directamente desde el compositor, es que cuando realizamos un video. Ya que aplica las modificaciones a todos los fotogramas y no tenemos que retocarlos luego en un editor de video, si no queremos, pues ya salen retocados de fábrica.

En primer lugar, vamos a aplicar un **filtro Sharpen**, para marcar un poco más las aristas de los objetos. Para añadirlo accedemos a la ruta: Add→Filter→Filter. Y seleccionamos en el menú desplegable el filtro sharpen. No utilizaremos un valor muy alto, ya que si no hay demasiado contraste en los contornos de los objetos.

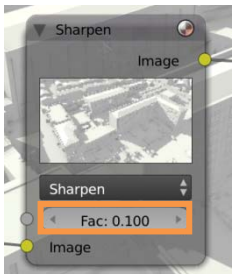


Ilustración 190 – Filter

Ahora le añadiremos un **efecto Vignetting**, para oscurecer un poco los márgenes de nuestra foto y centrar la atención en nuestro edificio. Para ello tendremos que usar un efecto **Lens Distorsion** (Add→Distorsion→Lens distorsión) que nos deforma la imagen a una elipse. Un operador **Greater Than** (Add→Converter→Math\Grater than) que transformara la elipse en blanco y la parte de las esquinas en negro. Y un filtro Blur **Fast Gaussian** (Add→Filter→Blur\Fast Gaussian) que suavizará los bordes. Este grupo en conjunto generará una máscara lista para aplicarla a la imagen y quedaría de la siguiente manera

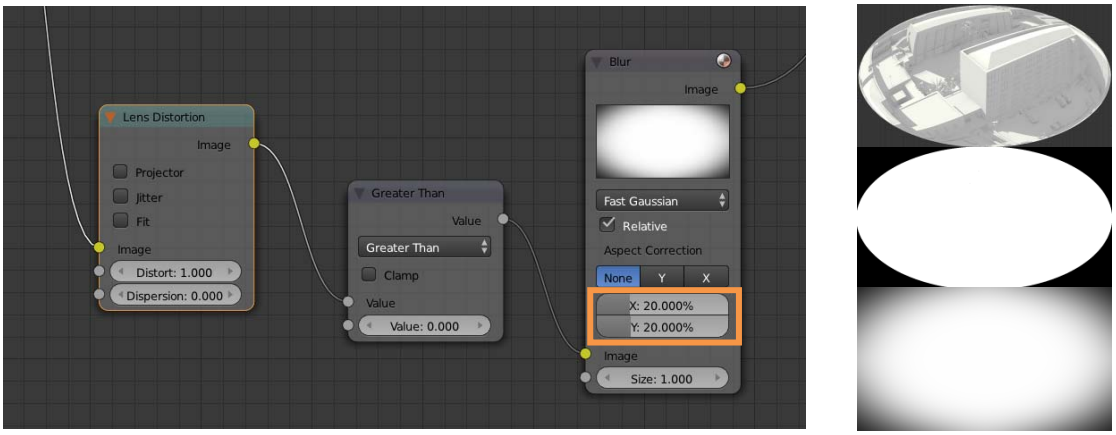


Ilustración 191 – Efecto Vignneting

Para aplicar esta mascara a la imagen original, utilizaremos un **mezclador de color Multiply** (Add→color→Mix\Multiply). Los mezcladores, tienen dos canales de entrada y uno de salida, por tanto en la primera entrada tendríamos que conectar la imagen que sale del filtro sharpen y en la segunda entrada la máscara de salida del filtro blur. Según el coeficiente que usemos para la mezcla, el efecto vignetting será más o menos intenso.

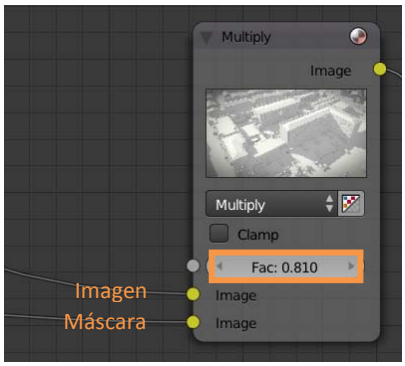


Ilustración 192 – Mix

A continuación, el resto de **modificaciones de color** que vamos a conector lo haremos de una manera lineal. Añadiremos un modificador de **Gamma** (Add→color→Gamma).Añadiremos también un modificador de las **curvas RGB** (Add→color→RGB curves) y un control de **saturación** (Add→color→Hue Saturation Value). Estos parámetros funcionan igual que en cualquier programa de edición de fotografía, por tanto la configuración dependerá mucho del usuario.

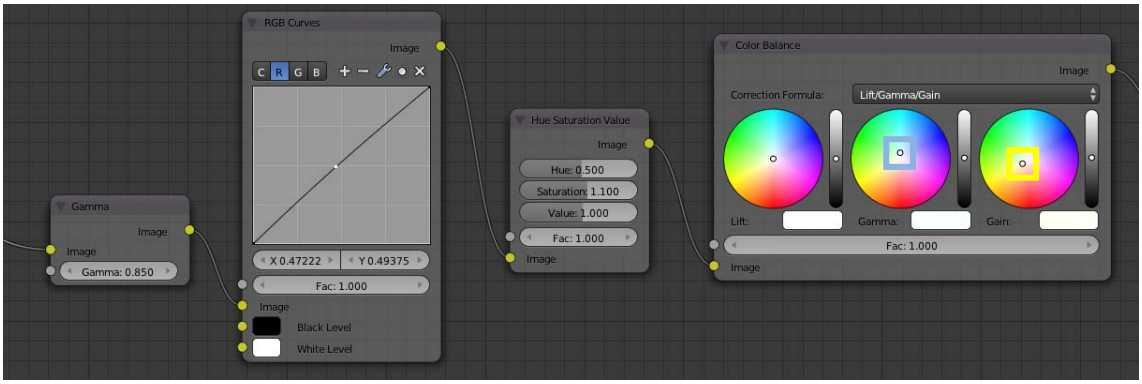


Ilustración 193 – Correctores de color

El último nodo que añadiremos, será **balance de color** (Add→color→Color Balance), donde haremos la siguiente. En "**Gamma**", le daremos un pequeño toque **azulado** al blanco, para corregir la luz ambiental que por defecto es blanca y en "**Gain**" un leve tono amarillo para acentuar algo más el efecto de la luz del sol.

Por lo tanto el compositor de nodos visto en conjunto quedaría de la siguiente manera:

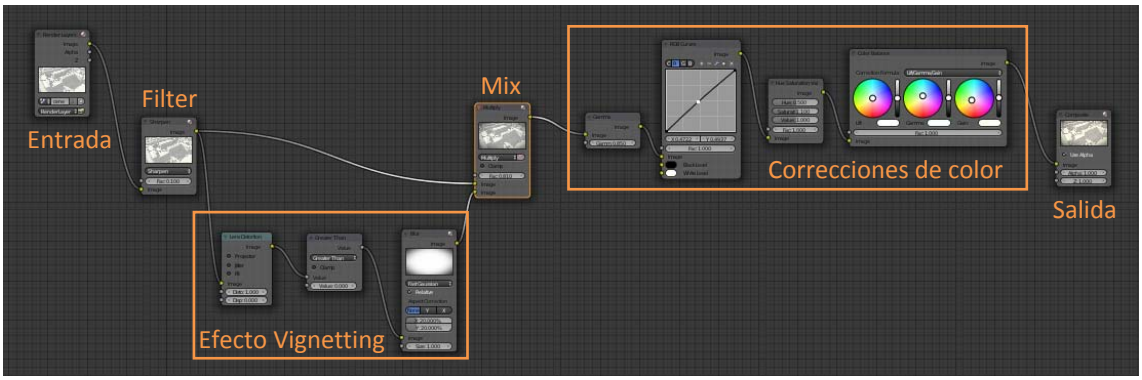


Ilustración 194 – Compositor de nodos

El **resultado final** se muestra en la siguiente figura:

Ilustración 195 – Iluminación exterior.

4.5.3 Iluminación Interior.

Para la iluminación interior el proceso será muy parecido. Utilizaremos un **sol**, para simular la **luz directa** y el sistema de **oclusión ambiental** para simular la **iluminación global**.

El problema en el interior, cuando utilizamos únicamente un sol para iluminar la escena, es que hay muy pocos objetos que reciban realmente luz directa, por lo que apenas tendremos sombras, brillos especulares y en general, tendremos **poca definición** en la imagen. Para solventar dicho problema, necesitaremos una fuente de **iluminación directa secundaria**, que simule la **luz ambiental** que entra por las ventanas.

Para comenzar nos aseguramos que está activado el **material de testeo** de la luz, en la pestaña **render layers**. Tenemos que tener cuidado, ya que al sustituir el material, las **ventanas no son transparente**s, por tanto tendremos que desactivarlas.

Una vez hecho esto, **colocamos un sol** como fuente de luz principal. Podemos utilizar el mismo sol que en la iluminación exterior y girarlo de tal forma que nos aseguremos que la luz directa entra por alguna de las ventanas.

El resultado de la iluminación local producido por el sol, tendrá este aspecto:

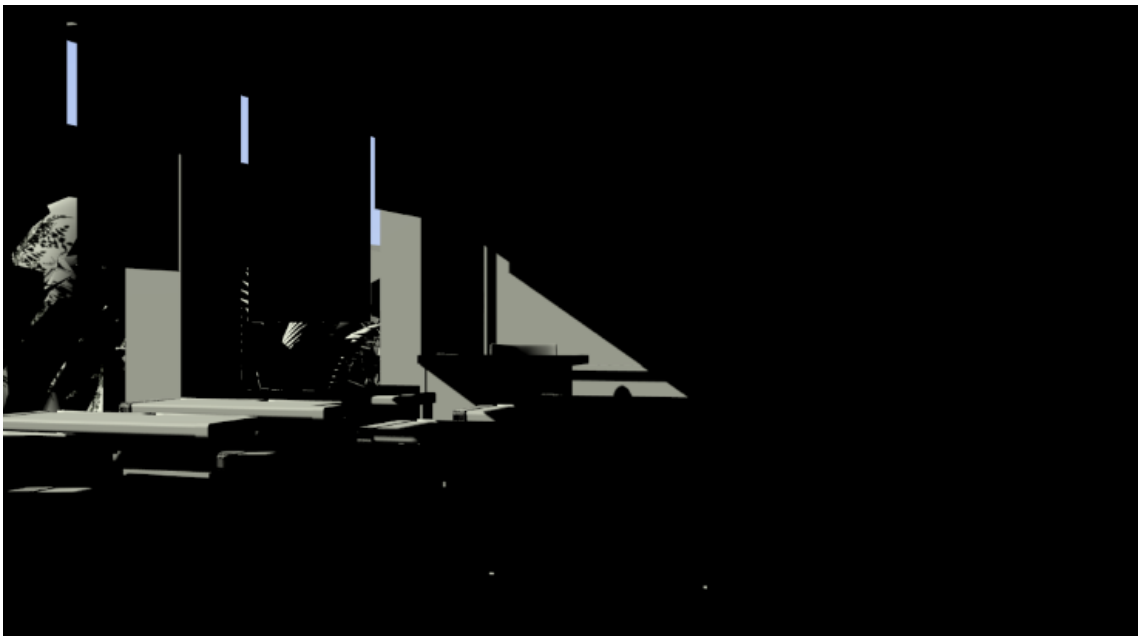


Ilustración 196 – Iluminación local interior.

Al igual que ocurría en el exterior, solamente se iluminan, los elementos que reciben luz directa. Como vemos, la proporción de la imagen que recibe luz directa en muy pequeña, por lo que no bastará con la oclusión ambiental para conseguir una definición decente de la escena.

Necesitamos incluir una **fuente de iluminación secundaria**, para ello utilizaremos una **luz de área** que simula la luz ambiental que entra por una ventana: Add→Lamp→Area.

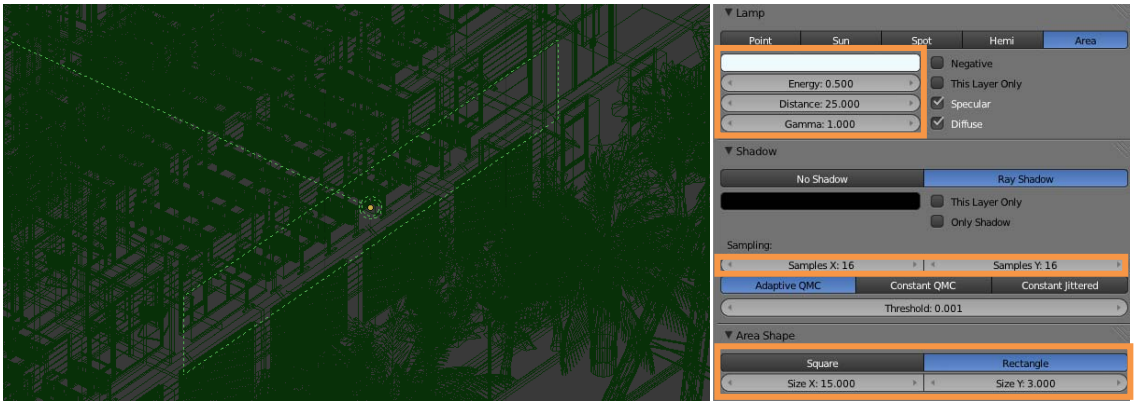


Ilustración 197 – Luz área.

La idea es colocar esta luz área, con un tamaño igual al **área de las ventanas** y situarla paralela a las mismas. Para cambiar el **tamaño**, es importante realizarlo desde las opciones del objeto, no escalando la luz. Para ello seleccionamos la opción **rectángulo** e introducimos los **valores de longitud en X e Y**. En cuanto al color de la luz, utilizaremos un blanco con un leve tono **azulado** para simular la luz ambiental. La intensidad de la misma, dependerá mucho del tamaño que tenga la luz y la distancia de desvanecimiento que le pongamos.

Para la iluminación interior del resto de salas, tendremos que repetir esta operación. Podemos poner un sol estándar que nos sirva para la gran mayoría de los interiores, sin embargo, la luz secundaria tendremos que ir colocándola una a una. Además es muy importante utilizar nombres claros, ya que necesitaremos activarlas y desactivarlas continuamente, dependiendo del render que estemos realizando.

El efecto que estamos buscando con esta luz secundaria, sería el de la luz que entra por la ventana un **día nublado**. Desactivaremos el sol momentáneamente, para ver el efecto que nos produce esta nueva luz área, y renderizamos.

Ilustración 198 – Iluminación local interior secundaria.

Por tanto, ya hemos conseguido que nuestra escena, esté correctamente iluminada mediante **luz directa**.

Finalmente, activamos la **oclusión ambiental** con los mismos parámetros que para la iluminación exterior, pero esta vez, sí activamos la opción "**indirect lighting**", para que nos calcule los rebotes de luz.

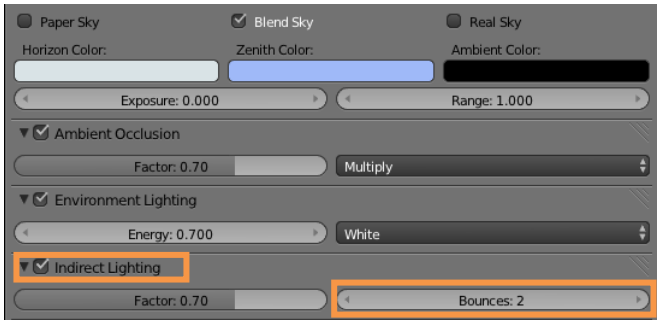


Ilustración 199 – Oclusión ambiental interior

Renderizamos la imagen únicamente con la **oclusión ambiental**. En la imagen vemos el efecto que produce en una escena interior.

Ilustración 200 – Simulación iluminación ambiental

Si activamos todas las luces para superponer las capas tenemos:

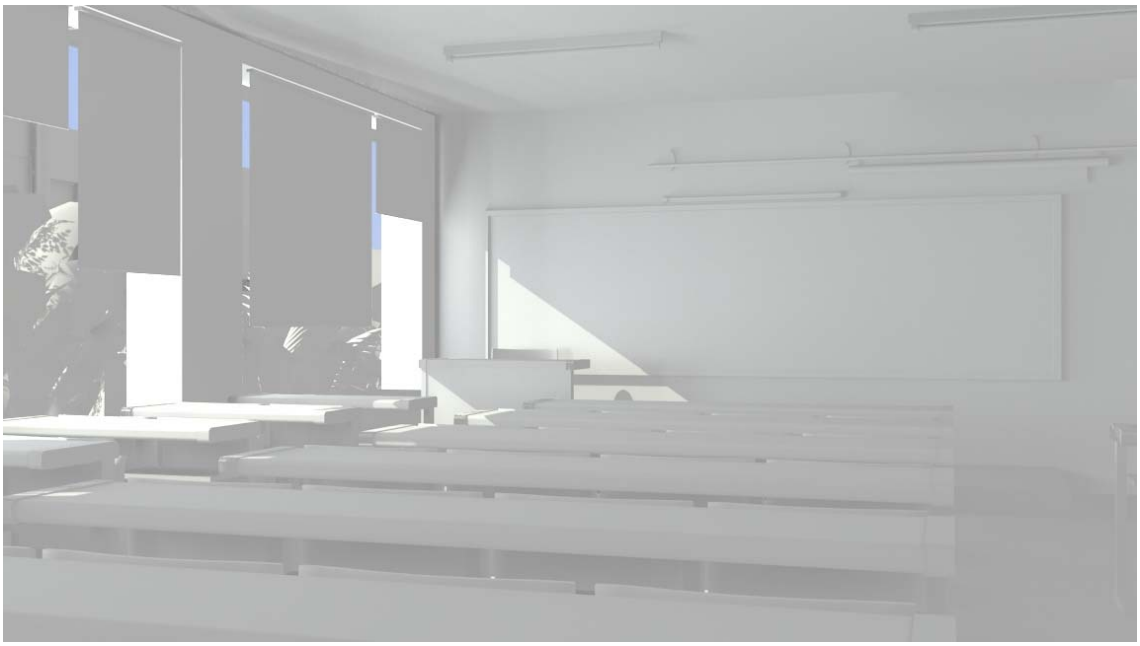


Ilustración 201 – Simulación luz global interior

Por último activamos los retoques de imagen del **compositor de nodos**:

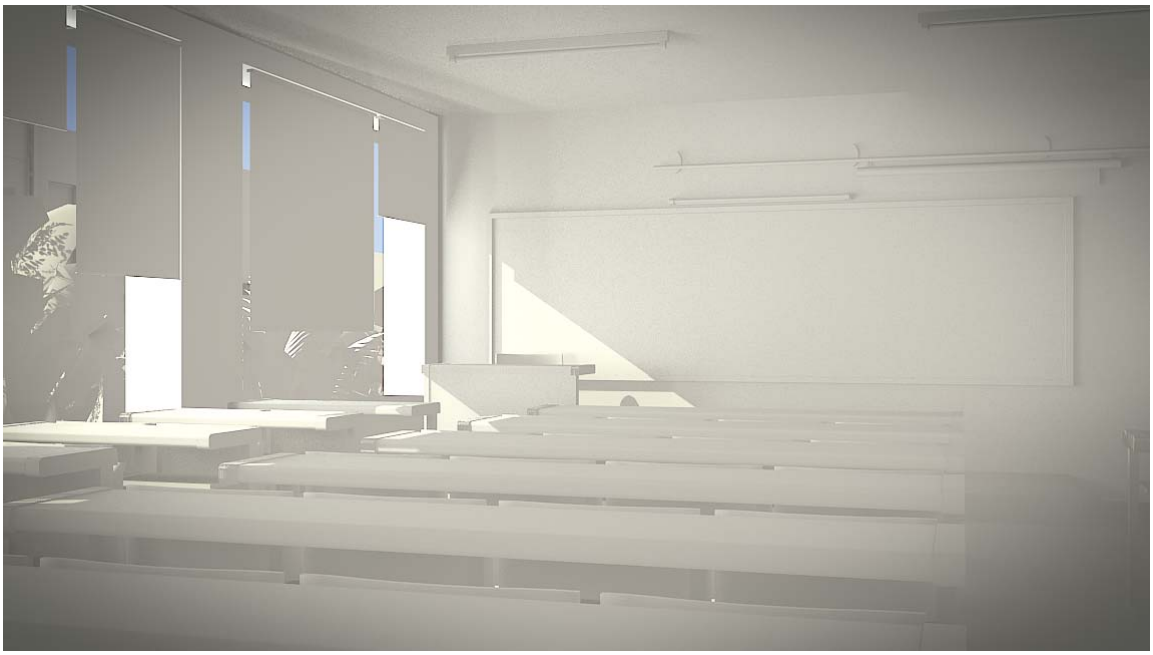


Ilustración 202 – Iluminación interior

4.5.3 Iluminación y Texturas.

Una vez que hemos terminado las pruebas de iluminación, tendremos que activar las texturas para ver el resultado final. Una vez renderizado con las texturas, daremos los últimos ajustes tanto a la iluminación como a los materiales.

Ilustración 203 – Iluminación y texturas

VIDEO DE PRESENTACIÓN

4.6 GUIÓN.

Antes de comenzar con la animación de nuestra escena, es importante tener un guión de la animación y el audio grabado. En el guión detallaremos tanto el texto para el audio, como las acciones que ocurren durante la animación.

De esta manera, al tener el audio grabado y una idea clara de lo que queremos hacer, nos resultará mucho más fácil ajustar el tiempo de la animación y saber el momento exacto en el que ocurren las cosas.

A continuación podemos ver el guión para el video de presentación de la EIMIA:

- Cuenta atrás / escena cabria patio / créditos con planos de situación.

- Escena1. Visual fachada principal

La Escuela de Ingeniería Minera e Industrial de Almadén, se encuentra situada en la Plaza Manuel Meca de la ciudad de Almadén, al suroeste de la provincia de Ciudad Real.

- Escena2. Fachada Academia de Minas

Durante el reinado de Carlos III en 1777, fue fundada la Academia de Minas de Almadén, bajo la dirección de Enrique Cristóbal Störr.

En esta época inicial, salen de sus aulas profesores y alumnos tan brillantes como Fausto E'lhuyar y Andrés Manuel del Río.

- Escena3. Vista aérea exterior recorriendo y marcando los diferentes edificios. (55s)

En 1973, la Escuela se traslada a su ubicación actual. La Escuela dispone de una superficie de 8.700m^2, y desarrolla su actividad docente en 5 edificios.

El Edificio Enrique Cristóbal Störr, donde se desarrollan la gran mayoría de las funciones del centro.

El Edificio Andrés Manuel del Río, que alberga varios de los laboratorios de prácticas más importantes de la escuela.

El Edificio Mateo Alemán, que alberga el Instituto de Geología Aplicada, donde se desarrollan actividades y proyectos de investigación.

El Edificio Fausto E'lhuyar, construido como aulario sobre las ruinas de la Real Cárcel de Forzados.

Y El Edificio Casiano del Prado dedicado a aulas para cursos de postgrado y laboratorios.

Además, cuenta con la Residencia Universitaria Luis Mateo, integrada en la Escuela y con acceso directo a la misma.

- Escena 4. Comienzo en fachada principal hasta el hall. Vista general del hall parando en el cartel informativo de las titulaciones disponibles.

La Escuela de Ingeniería Minera e Industrial de Almadén, inicia en el curso 2010-2011 los nuevos títulos de Grado, dentro del proyecto común de Espacio Europeo de Educación Superior.

Los nuevos títulos de grado están estructurados en cuatro cursos académicos:

- Grado en Ingeniería de los Recursos Energéticos.

- Grado en Ingeniería de la Tecnología Minera.

- Grado en Ingeniería Eléctrica.

- Grado en Ingeniería Mecánica.

- Escena 5. Edificio Störr sobre plano de la escuela apareciendo planta por planta e indicando las principales estancias (biblioteca, salón de actos, laboratorios,...) y marcando las zonas de aulas, administración y departamentos. Al final aparece la fachada completa completando el edificio.

El edificio Enrique Cristóbal Störr cuenta con 3.340 m^2 construidos en cuatro plantas.

- En planta baja, encontramos algunos de los servicios del centro.

  -Como el servicio de publicaciones.

  -Cafetería.

  -O la Biblioteca, que cuenta con 180 m^2, donde podemos encontrar más de 7800 monografías, 1300 proyectos fin de carrera y 55 publicaciones periódicas en curso. Además de 614 volúmenes en el fondo antiguo, originario del siglo XVIII.

  -El resto de la planta, está destinado a las actividades administrativas y de dirección. Anexo al edificio principal, encontramos el Salón de Actos, donde se desarrollan los actos académicos y conmemorativos de la Escuela.

En las plantas superiores, es donde se desarrolla la actividad docente del centro.

- En planta primera se encuentra:

  -El Laboratorio de Ciencias de la tierra.

  -El departamento de Ingeniería Geológica y Minera.

  -La sala de conferencias.

  -Y 2 Aulas.

- En planta segunda se encuentra:

  -El Laboratorio de docencia y de investigación de Química.

  -Los departamentos de Ingeniería Química, Químico-Física, Química orgánica y Química analítica.

  -Y 2 Aulas.

- Y En planta tercera se encuentra:

  -El Laboratorio de Electricidad.

  -El departamento de Ingeniería Eléctrica, Electrónica y Automática.

  -1 Aula

  -El departamento de Expresión gráfica.

  - El aula de dibujo asistido por ordenador.

  -Y el Aula de dibujo.

- Escena 6. Vistas interiores Störr. Muy breves.

Salón de Actos, Cafetería, Biblioteca, Sala juntas, Lab Electrónica, Lab. Química, Aula.

- Escena 7. Edificio Andrés M. del Río sobre plano de la escuela aparece junto al edifico anterior sin cubierta y sin la fachada norte, permitiendo ver los diferentes talleres y el aula de informática.

El edificio Andrés M. del Río, es una nave de 500 m^2 concebida para talleres, pero que en la actualidad alberga varios laboratorios de prácticas:

-El laboratorio de ingeniería de los procesos de fabricación

-El laboratorio de metrología y automatización

-El laboratorio de tecnologías mineras

-Así como el aula de informática

- Escena 9. Vistas interiores Andrés M. del Río. Muy breves.

  Taller, Aula de Informática.

- Escena 8. Continuando escena 7.Edificio E'lhuyar sobre plano de la escuela apareciendo planta por planta e indicando las principales estancias (ruinas, museo minero,...) y marcando las zonas de aulas y departamentos. Al final aparece la fachada completa completando el edificio.

El edificio E'lhuyar cuenta con 560 m^2 construidos en 3 plantas. El edificio se encuentra construidos sobre los antiguos calabozos de la Real Cárceles de Forzados, que forman parte del conjunto declarado por la UNESCO patrimonio de la Humanidad en 2012.

El edificio en planta baja cuenta con el centro de interpretación de la cárcel de forzados. Además podremos acceder a un gran espacio a doble altura donde podemos visitar las ruinas de los antiguos calabozos.

Asociado a este espacio, en planta primera, encontramos una gran sala de exposición del museo minero Francisco Holgado, donde se hace un repaso a la historia de la Minería de la Comarca de Almadén.

En la segunda planta, se encuentra el aulario, que cuenta con:

- El aula magna

- El aula informática

- 2 Aulas.

- Además de los departamentos de Mecánica Aplicada, Economía, Informática y Matemáticas.

- Escena 9. Vistas interiores E'lhuyar. Muy breves.

  Ruinas, Pasillo P2, Aula.

- Escena 10. Continuando la escena 8, continúan apareciendo los edificios sobre el plano completando todo el complejo. Visto desde el patio, la cámara vuela y termina con una última vista aérea.

- Escena 11. Créditos

4.7 ANIMACIÓN OBJETOS Y CÁMARAS.

Para la animación de nuestra escena, utilizaremos las herramientas más sencillas dentro de las opciones de animación de Blender. La animación consiste en especificar la **posición** y movimientos de nuestros objetos a lo largo de una **línea temporal**. Obviamente cuando tenemos objetos o personajes animados [Hess, 2011], esta tarea se complica bastante. En nuestro caso, no tenemos personajes que animar, ya que se trata de mostrar las instalaciones de un edificio, por lo tanto solamente realizaremos **movimientos** simples de **objetos** y de **cámaras**.

4.7.1 Keyframes.

Para definir el movimiento de cualquier objeto, utilizaremos los fotogramas claves o keyframe. Un **keyframe**, es una **referencia** que le damos al programa, sonde situamos nuestro objeto en una posición del espacio en un momento determinado de la línea de temporal. Cuando un objeto tiene varios keyframes, el programa automáticamente genera una trayectoria continua de transición entre ambos. [Roosendaal, 2004]

Blender ofrece una cantidad innumerable de parámetros de animación, ya que podemos incluir keyframes prácticamente en **cualquier opción o manipulador** del programa. Podemos animar parámetros básicos como posición, rotación y escala, hasta cualquier propiedad de un material, incluido las texturas.

Para poder explicar el proceso de una manera sencilla, vamos a ver un ejemplo de animación de una cámara donde sustituiremos nuestro edificio por un cubo.

Suponemos que nuestra animación va a durar 10 segundos a 24 fotogramas por segundo, por lo tanto nuestra escena tendrá que tener 240 fotogramas. Podemos configurar la duración de la escena en la ventana "**Timeline**".

Ilustración 204 – Timeline: duración escena

En el gráfico, podemos ver en gris claritos los frames pertenecientes a nuestra escena.

En la siguiente imagen, vemos la que será la posición inicial y final de nuestra cámara, y que serán las posiciones que utilicemos para generar los fotogramas claves.

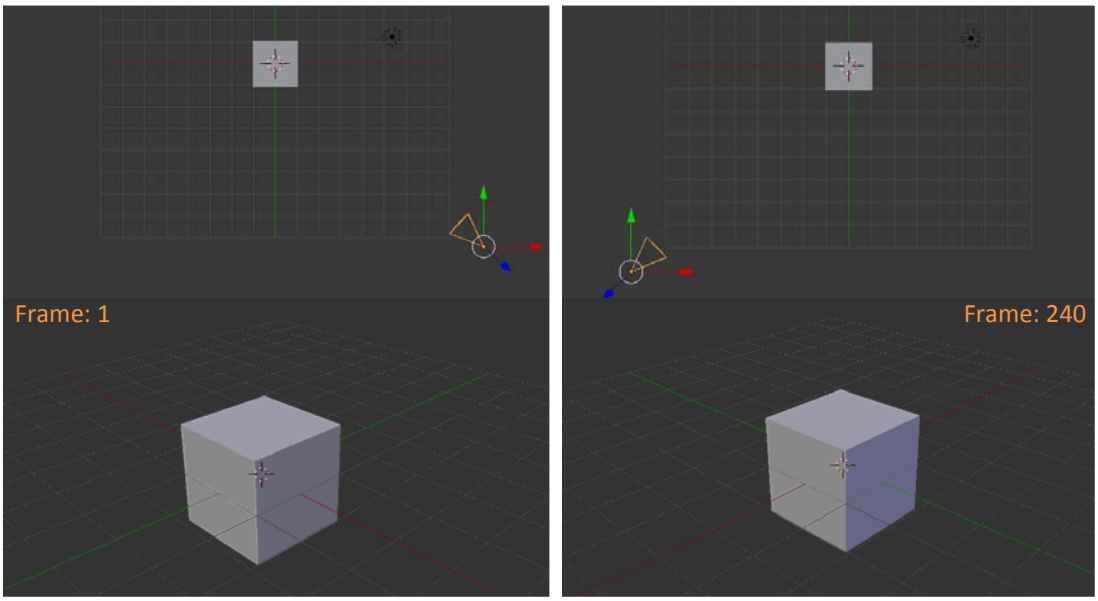


Ilustración 205 – Animación, posición inicial y final

Para insertar un fotograma clave, primero tenemos que especificar qué **tipo de clave** queremos. Para este caso, queremos modificar la posición y la rotación de la cámara, por lo que tendremos que seleccionar una llave tipo "**LocRot**" que nos guardará para cada frame el vector posición y la matriz de rotación del objeto. Para seleccionarlo, utilizamos el menú desplegable de la ventana timeline.

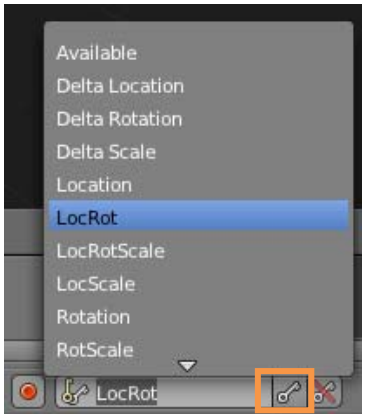


Ilustración 206 – Tipos de claves

Una vez seleccionado el tipo de clave, iremos al frame 1 y seleccionamos la cámara, entonces, para guardar la posición utilizamos el icono con forma de llave. Podemos ver cómo aparece **marcada** la clave en el gráfico del **timeline** mediante una línea amarilla. Ahora realizaremos la misma operación en el frame 240. Seleccionamos la cámara, la movemos a su posición final y volvemos a pulsar sobre la llave.

El programa nos genera entonces, automáticamente el movimiento de transición.

Podemos desplazarnos por el timeline, para ver el resultado obtenido.

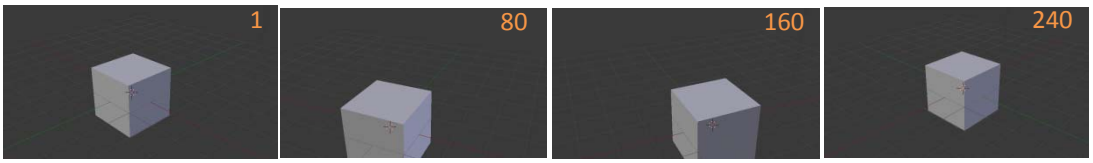


Ilustración 207 – Trayectoria automática

Como vemos en la imagen, la trayectoria generada por el programa tiene un problema de encuadre, ya que se pierde parte del cubo en los fotogramas intermedios, por lo tanto, será necesario introducir **keyframes intermedios**. Siempre que sea posible, tenemos que tratar de generar una trayectoria limpia, por lo que tendremos que utilizar el menor número de keyframes intermedios, para que el movimiento de la cámara sea lo más suave posible.

Para introducir un keyframe intermedio, nos iremos a uno de los frames centrales, en este caso al 120. Entonces movemos la cámara de manera que el cubo quede perfectamente encuadrado. Una vez esté listo, volvemos a pulsar sobre la llave.

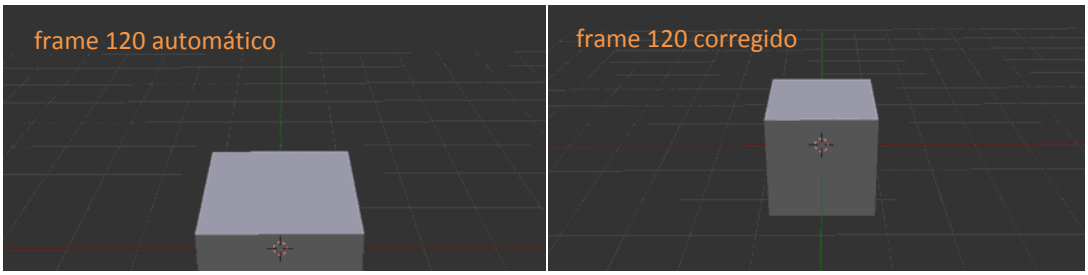


Ilustración 208 – Keyframe intermedio

Por tanto la animación ahora queda así:

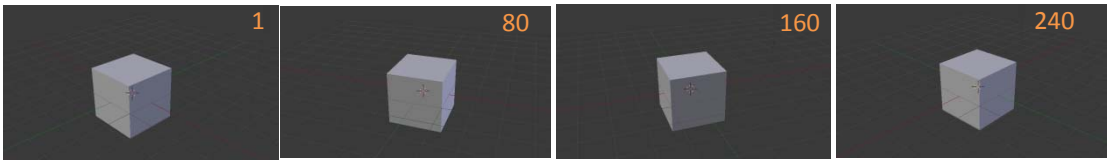


Ilustración 209 – Trayectoria corregida

4.7.2 Graph editor.

La ventana "**graph editor**", nos permite **manipular** las acciones de nuestra animación de una forma gráfica. Además podemos modificar algunos aspectos de la generación automática de trayectorias. [Flavell, 2010]

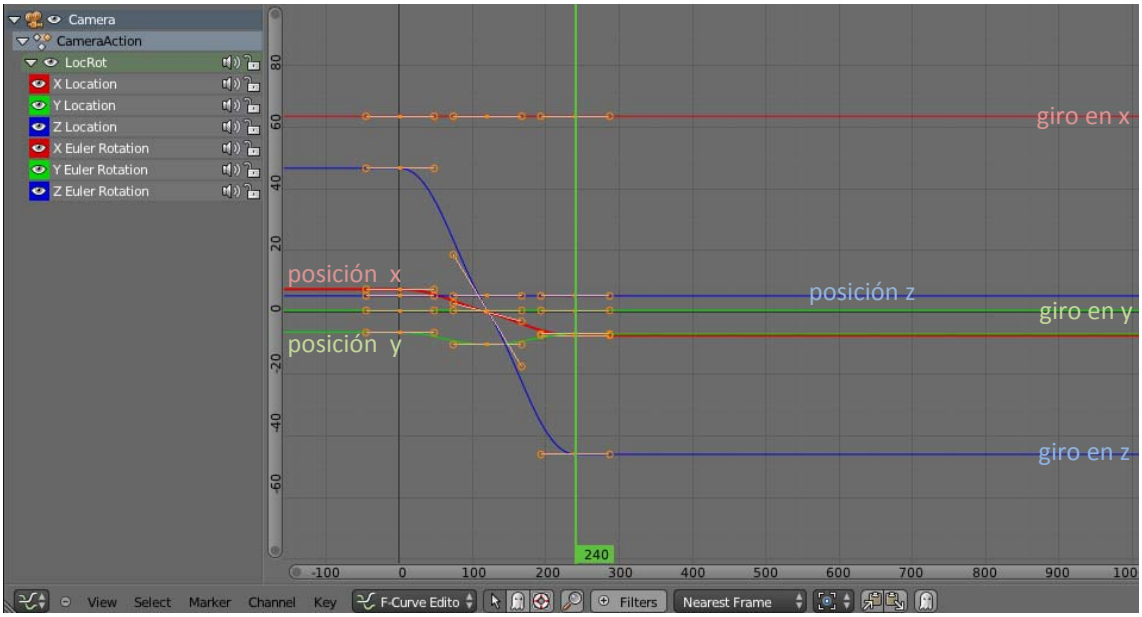


Ilustración 210 – Graph editor

En la izquierda, vemos el nombre del objeto que estamos animando. Como la clave que hemos utilizado es una clave "LocRot", en la gráfica nos aparecen los datos correspondientes a la **localización** en cada eje y a la **rotación** con respecto cada eje. En la parte izquierda podemos desactivar las gráficas que no necesitemos para la edición.

Nuestra cámara, solamente se mueve en el plano X e Y, y gira solamente respecto al eje Z, como podemos ver en la gráfica. Por eso, la gráfica de giro en X y en Y, y la de la posición Z, no varían su valor, por lo tanto son horizontales.

Podemos ver, que los puntos de inflexión que aparecen en la gráfica, coinciden verticalmente en los fotogramas con claves. Moviendo estos puntos, podríamos modificar cualquiera de estos parámetros de una manera gráfica.

Si le damos al "play" en la ventana "timeline", podemos visualizar nuestra animación y comprobar que efectos han tenido nuestros cambios.

Si observamos la animación tal y como nos la ha generado Blender, vemos que los movimientos, comienzan y terminan de una forma muy lenta, y es en la parte central donde toma más velocidad. Esto es debido a la curvatura de la gráfica en los **puntos de inflexión**. Podemos modificar la curvatura de los punto de inflexión desde el menú desplegable "Key→Handle type". Tenemos dos opciones principales:

- **Automatic**: Es el método por defecto, crea una curvatura que **suaviza los movimientos al comienzo y al final** de cada acción.
- **Vector**: Se produce un cambio contante en la variable, produciendo un movimiento a **velocidad constante**.

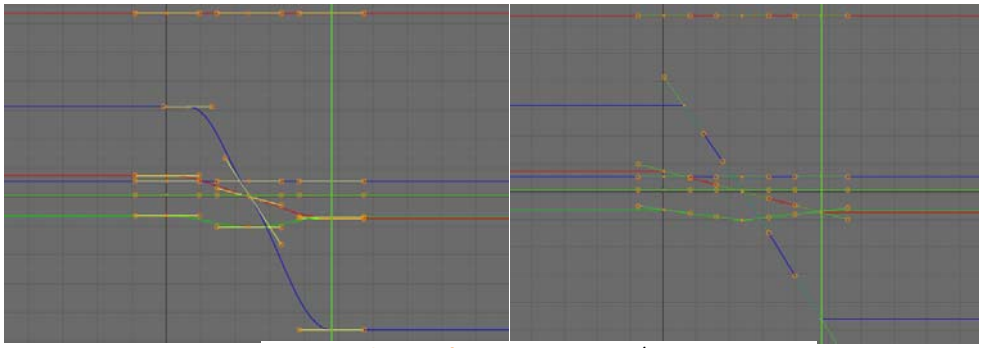


Ilustración 211 – Curvatura / vector

Por lo tanto, podemos mezclar dichos efectos o modificar nosotros manualmente la curvatura, para generar diferentes tipos de movimiento y diferentes velocidades.

4.7.3 Dope Sheep.

Desde esta ventana, podemos **manipular los keyframes** y las acciones de una forma sencilla y gráfica. En la parte izquierda, al igual que en el graph editor, nos aparecen los objetos que tienen animaciones. Y en la derecha, un timeline con los fotogramas claves de cada objeto. [Flavell, 2010]

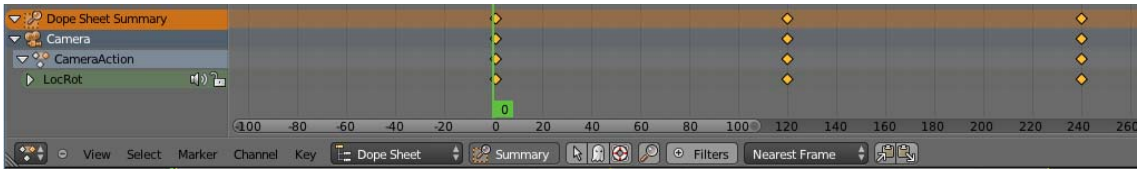


Ilustración 212 – Dope Shit

Desde esta ventana, es muy sencillo **retrasar** o **adelantar** una acción o incluso **copiarla** y **escalarla**. En nuestro ejemplo, imaginamos que queremos conservar el movimiento completo, pero que la cámara pare durante cuarenta fotogramas en la posición central. Por lo tanto, la cámara se mueve del frame 1 al frame 100, se ditiene durante 40 frames, y continúa su movimiento hasta el frame 240.

En primer lugar, tendríamos que seleccionar el frame central y moverlo al frame 100. Podemos hacerlo arrastrando o utilizando la tecla "G" e introducir un valor numérico mediante el teclado. Una vez desplazado el frame central, copiaríamos con "Shift+D" hasta el frame 140.

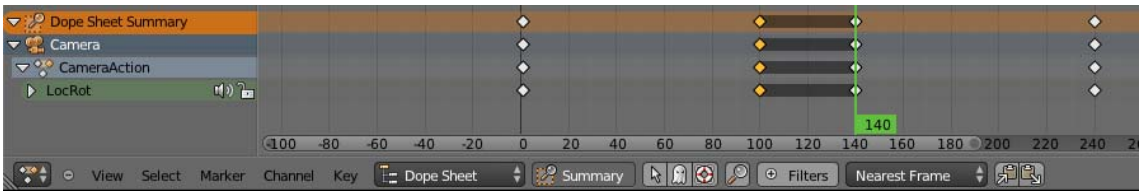


Ilustración 213 – Dope Shit modificado

En la parte central, aparece un sombreado, que nos indican que entre estos frames, todas las variables mantienen un valor constante, lo que nos ayuda bastante a la hora de localizar acciones.

Mediante estas herramientas, podemos generar animaciones simples muy completas.

4.7.4 Simulador de física.

El simulador de física, es una herramienta muy poderosa para crear animaciones complejas. Podemos simular colisiones, humo, fluidos, partículas, ropa, viento, etc.

Para el proyecto, lo hemos utilizado para modelar y **animar las banderas** de entrada del edificio Störr. Para ello, hemos utilizado un **simulador de ropa** "Cloth" y una fuerza externa, en este caso un simulador de **viento**. [Hess, 2011]

Comenzaremos modelando una bandera, de la manera más simple posible, mediante un cilindro esbelto, y un plano. Es importante que sean dos objetos separados.

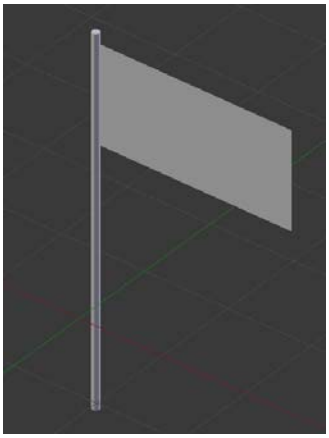


Ilustración 214 – Bandera

El plano que simula la bandera, tendremos que **subdividirlo** varias ves, ya que cuantos más puntos tenga el programa para calcular la acción, mejores resultados obtendremos.

Una vez subdividido el plano, tendremos que crear un grupo de vértices para poder indicarle al simulador los puntos de la bandera que serán fijos. Para ello entramos en el modo editar, y seleccionamos la arista superior e inferior del borde de la bandera más cercano al mástil. Desde la pestaña de "Object Data" en "**Vertex Groups**", creamos un nuevo grupo y le asignamos los vértices seleccionados.

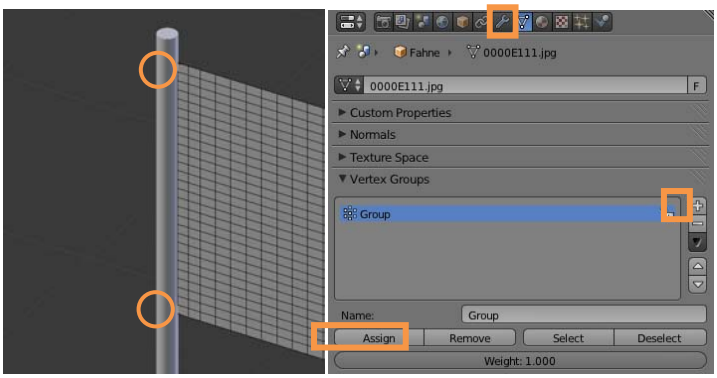


Ilustración 215 – Vertex Group

Desde la ventana de propiedades, seleccionamos la pestaña "**Physics**" donde nos aparecen todos los efectos que podemos simular en Blender. Para la bandera, vamos a seleccionar el simulador "**Cloth**". Nos aparecen una serie de modificadores que definen el **comportamiento físico del material**, como elasticidad, flexión, masa, resistencia estructural, etc. En la imagen podemos ver los valores que le hemos dado a cada uno de ellos para simular una tela. Es muy importante en este caso, activar la opción de fijado "**pinning**" y seleccionar el grupo de vértices que hemos creado anteriormente, para que sujete nuestra bandera al mástil.

Para el mástil, utilizaremos un simulador de colisión, de manera que la bandera choque contra él y no lo traspase.

Si pulsamos el botón play del timeline, comenzará la animación de la ropa y vemos como poco a poco, nuestra bandera pasa de ser un plano rígido a una tela afectada por la gravedad.

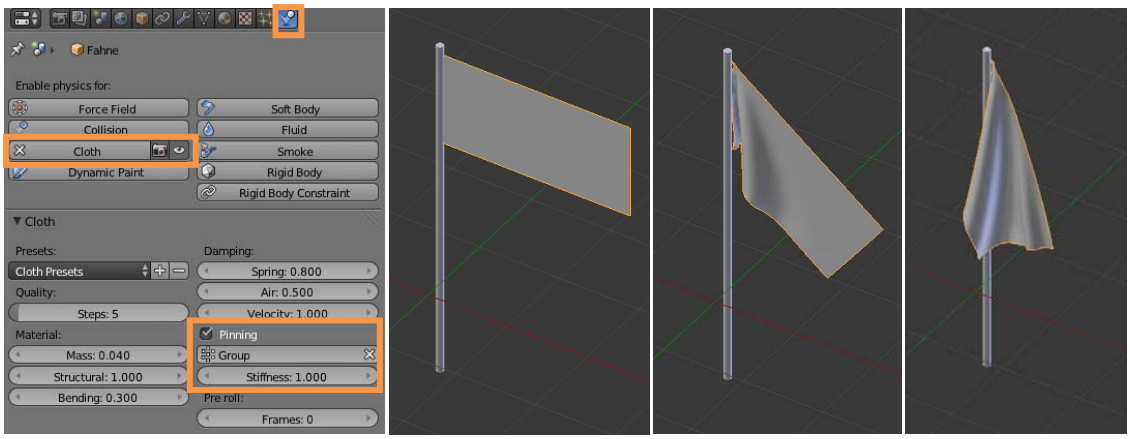


Ilustración 216 – Simulador de ropa

Si solamente quisiésemos modelar la bandera, esperaríamos a que la gravedad termine de ejercer su acción y desde la pestaña de modificadores, aplicaríamos definitivamente el modificador de ropa, para que el cambio fuese permanente.

La idea es que la bandera durante la animación tenga un leve movimiento simulando la acción del viento. Para ello necesitamos introducir una **fuerza externa "wind"**. Lo haremos desde Add→Force Field→wind.

El parámetro más importante para nuestro viento será la **fuerza**. Podemos realizar diferentes pruebas para ver cuanta fuerza necesitamos para mover de una forma realista nuestra bandera. Además, podemos introducir diferentes **keyframes** en este parámetro de forma que la fuerza del viento cambie con el tiempo y no sea constante.

Para insertar un keyframe, tenemos que poner el valor que queramos en el parámetro fuerza, y pulsando con el botón derecho del ratón elegir la opción "insert keyframe".

Una vez tenemos nuestro viento configurado y colocado en la dirección correspondiente, volvemos a visualizar la animación.

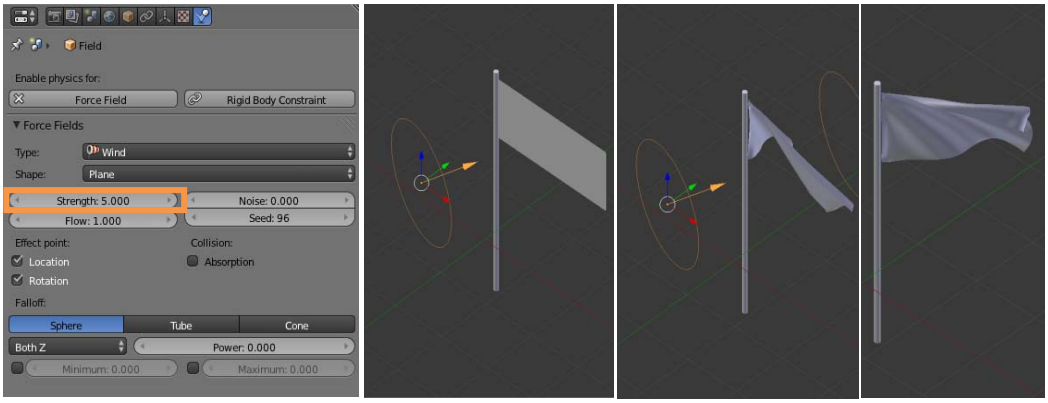


Ilustración 217 – Simulador de ropa + viento

Una vez que ha terminado la animación, tenemos que guardar el cálculo hecho por el programa, de manera que no tenga que repetirlo cada vez que iniciamos la visualización de la animación. Para ello, desde las opciones del modificador "Cloth", realizamos una copia de la memoria cache mediante el botón "Bake".

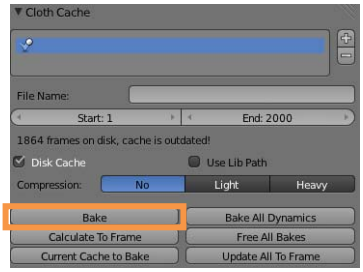


Ilustración 218 – Cloth cache

4.8 CONFIGURACIÓN DE SALIDA.

Una vez tenemos nuestra animación terminada, tenemos que configurar las opciones de salida de nuestra imagen.

Para acceder a estas opciones, utilizaremos la pestaña "**render**" en la ventana de propiedades.

En el primer panel "**render**", tenemos varios botones. El primero "**render**", nos renderiza el **frame actual** de nuestra escena. El segundo "**animation**" nos renderiza un **intervalo de frames** especificado en el panel inferior. Y el tercero "**play**", sirve para ejecutar un **previsualización** de nuestro **videojuego**, en el caso de que estemos trabajando con el motor de videojuegos.

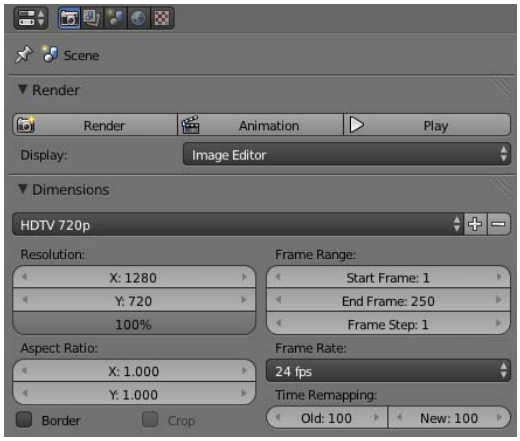


Ilustración 219 – Render

En el panel "**Dimensions**", especificamos el **tamaño de salida** de nuestra imagen. En nuestro casa, para las imágenes y el video hemos seleccionado el formato de salida **HDTV 720p** con unas dimensiones de 1280x720 píxeles.

Desde este panel, podemos especificar el número de frames que queremos renderizar y el **ratio de frames por segundo** que queremos utilizar en nuestra animación. En nuestro caso hemos utilizado un ratio de **24 fps**.

En el panel "**Anti-Aliasing**", podemos seleccionar el **método** de suavizado que queremos utilizar y el **número de muestras** "sample" que queremos tomar por píxel, cuanto mayor sea el muestreo, mejor calidad y más tiempo de renderizado.

Ilustración 220 – Anti-Aliasing

Desde el panel de salida "**Output**", podemos seleccionar el **formato de salida** de nuestra imagen o de nuestra animación y la **carpeta de destino**. Podemos seleccionar entre varios formatos de **imagen y video**.

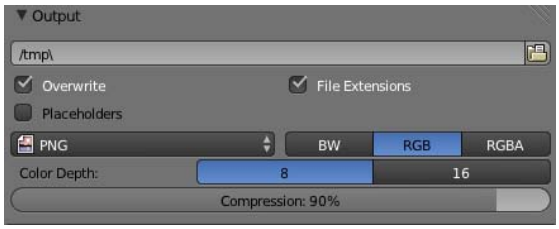


Ilustración 221 – Output

Para renderizar una animación, podemos seleccionar directamente un formato de video, pero no es recomendable, ya que si el proceso falla, perderemos todo el progreso de render. Lo mejor para una animación, es seleccionar un formato de imagen, por lo tanto el programa nos calculará frame por frame, y en caso de fallo, solamente tenemos que continuar desde donde se produjo el fallo. Con este sistema, tendremos una imagen por cada frame, y podremos montar el video sin problemas en cualquier editor de video o en el propio Blender con el secuenciador de imágenes. Por lo tanto nuestro formato de salida será **".PNG"** en RGB.

Otro panel importante para nosotros, es el panel "**Performance**". Donde tenemos una serie de opciones para ahorrar memoria, recursos, etc. y poder reducir los tiempos de render.

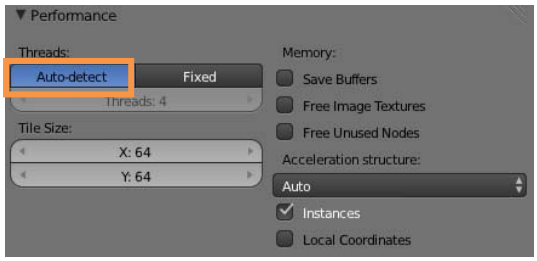


Ilustración 222 – Performance

Dado que para la realización de los render vamos a usar el servicio de **Supercomputación de la UCLM**, es muy importante seleccionar la opción de **autodetectar microprocesadores** y no introducirlo manualmente.

Una opción que nos puede bajar mucho los tiempos de render es la opción "**Tile Size**", que determina el **número de píxeles** que renderiza a la vez cada **CPU** o cada **GPU**.

No es necesario que este valor forme un cuadrado, aunque es recomendable que los números utilizados sean múltiplos de 2. Cuando calculemos por CPU, funcionarán mejor valores bajos, mientras que para GPU utilizaremos valores más altos. [Guru, 2014]

En la página web, *www.blenderguru.com*, se han realizado unas pruebas con distintos valores de "tile size", tanto para GPU como para CPU. Estos son **valores orientativos**, ya que dependerán mucho del tipo de microprocesador, del tipo de tarjeta gráfica y de la escena tridimensional en sí misma. Pero puedo decir, que merece la pena realizar unas cuantas pruebas para ver con que tamaño funciona más rápido en nuestra computadora. En mi caso, y renderizado mediante GPU, el formato con el que más velocidad he obtenido ha sido 128x128.

Tile Sizes (GPU)		
X	Y	Render Time (seconds)
256	256	197
128	128	199
64	64	199
480	540	201
480	270	201
512	512	202
960	540	206
120	68	218
32	32	503
16	16	1769

Tile Sizes (CPU)		
X	Y	Render Time (seconds)
16	16	430
32	32	431
64	64	449
120	68	452
128	128	471
256	256	488
480	270	800
512	512	1166
480	540	1231
960	540	2340

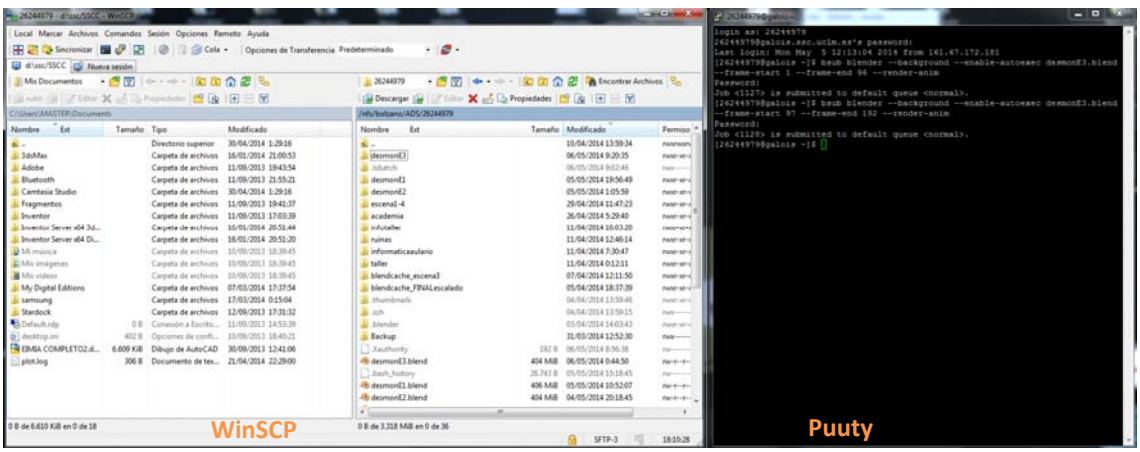
Results based on a 960 x 540 render

Ilustración 223 – Performance

4.8.1 Servicio de Supercomputación de la UCLM.

Para el renderizado de las escenas, hemos utilizado el **Servicio de Supercomputación de la Universidad**. Para poder lanzar los render de forma remota hemos necesitado dos programas: **WinSCP** y **Puuty**.

Ilustración 224 – WinSCP / Puuty



WinSCP: Este programa es un **explorador de archivos**. Tras iniciar sesión en el servidor, nos aparece la ventana dividida en dos partes. En la izquierda, un explorador de archivos locales, donde podemos buscar en nuestro disco duro. En la derecha, un explorar de archivos remoto, conectado a nuestra carpeta de usuario del servicio de supercomputación *\nfs/bolzano/ADS/"nºDNI"*. Tendremos que mover nuestros archivos de Blender a esta carpeta para poder lanzar los comandos de una forma remota. Y una vez terminado el proceso, mover las imágenes renderizadas de nuevo a nuestro disco duro. Es importante configurar la carpeta de salida de nuestro archivo Blender y colocar la ruta de nuestra carpeta:

\nfs/bolzano/ADS/"nºDNI"/carpeta salida/

Putty: Es una consola para la **ejecución de comandos** de forma remota. Desde aquí tendremos iniciar sesión para poder comunicarnos con Blender y que renderice la escena que queremos. Para ello utilizaremos la siguiente línea de comandos:

bsub *blender --background --enable-autoexec "nombre fichero".blend --frame start "nº" --frame end "nº" --render-anim*

Donde tendremos que especificar el nombre de nuestro fichero, el frame inicial y el frame final de nuestra animación. Para sacar el mayor partido al servicio de supercomputación, es mejor dividir nuestra animación en intervalos pequeños de frames, para que se rendericen en paralelo.

Una vez lanzado el comando, se nos facilitará un **número identificador** del trabajo. Podemos seguir el estado de nuestros trabajos utilizando el comando: *bjobs*

Nos aparecerá la siguiente información:

JOBID	USER	STAT	QUEUE	FROM_HOST	EXEC_HOST	JOB NAME	SUBMIT_TIME
968	xxxxxxx	RUN	normal	fermat	compute-4-3		Apr 28 11:38

Ilustración 225 – bjobs

4.9 POSTPRODUCCIÓN Y MONTAJE.

Para realizar el montaje del video final, utilizaremos la ventana "**Video Sequence Editor**". Abriremos un nuevo archivo en Blender, y pondremos en el **modo de ventanas**, la opción "**Video Editing**". Esta opción nos cargará las opciones necesarias para la edición de video. Personalmente creo que también es necesario tener la ventana de propiedades visible, para poder configurar la salida del video y el número de frames por segundo. Así que podemos duplicar alguna de las ventanas para sacar esta ventana.

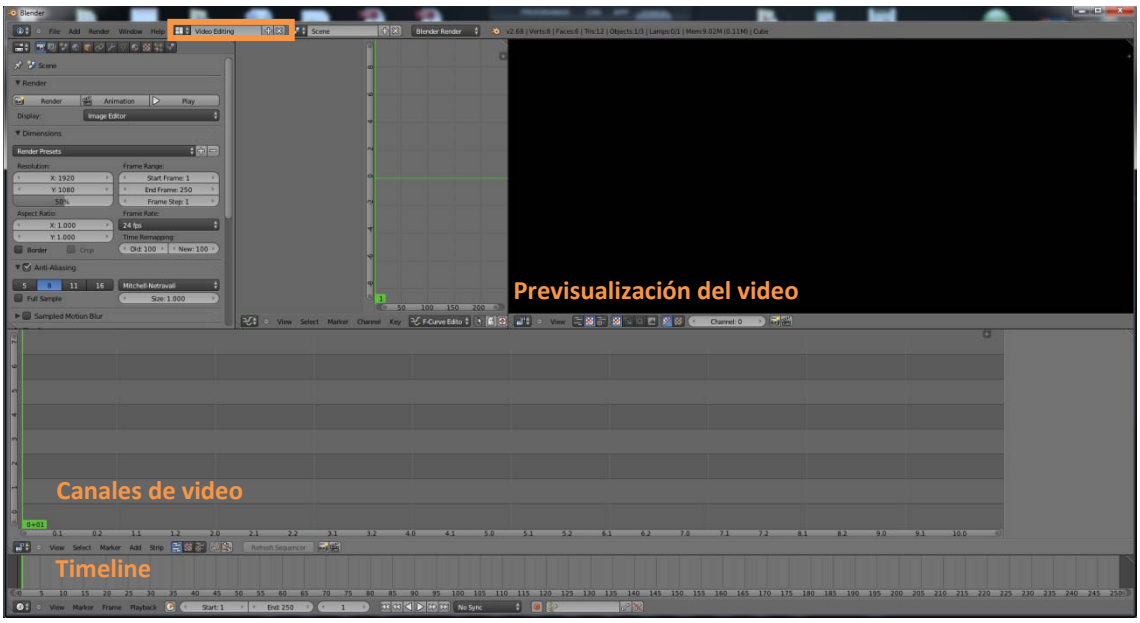


Ilustración 226 – Video Editing

El funcionamiento es muy similar a cualquier programa de edición de video. Tenemos una serie de canales, donde iremos añadiendo **imágenes**, **clips de video** y **música**. Hay que tener en cuenta que el programa leerá estos canales de arriba a abajo, por lo que si ponemos una imagen en el canal superior, no podremos ver las imágenes que estén en los canales inferiores.

4.9.1 Añadir un clip de video.

Para añadir un nuevo clip de video o un conjunto de imágenes, desde la ventana Video Sequencie Editor, seguimos la secuencia "Add→Image". Se abrirá el explorador de archivos de Blender, donde tendremos que buscar la carpeta donde tenemos nuestra escena. Tenemos que recordar, que renderizamos las escenas frame a frame, por lo que tendremos que seleccionar **todas las imágenes** para que el programa nos cree automáticamente un **clip de video**.

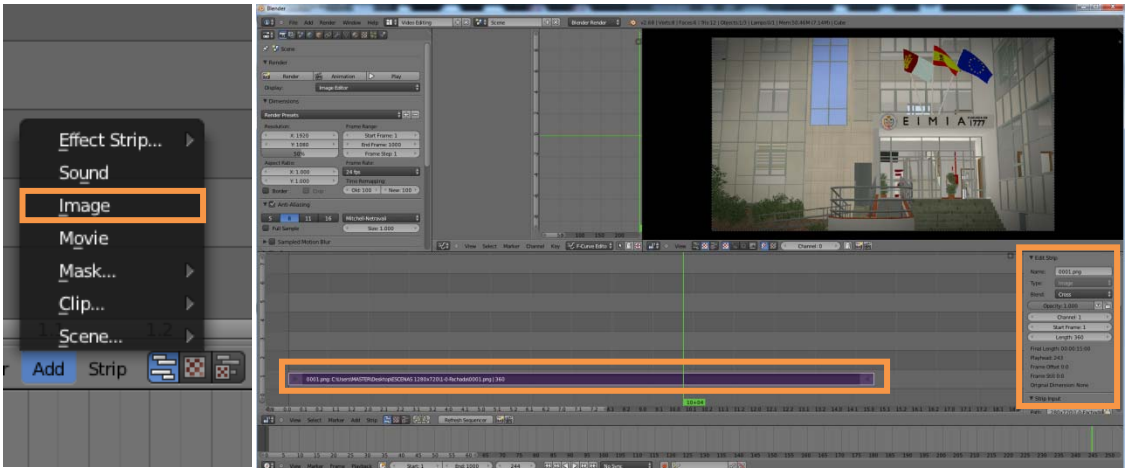


Ilustración 227 – Añadir secuencia

Al insertar las imágenes, nos aparece la escena en un "**strip**" en el Video Sequence editor, donde podemos ver los frames que ocupa, el frame inicial y final, etc. Este Strip, podemos **desplazarlo** para que comience en el frame que queramos. También podemos **recortar** el principio o el final, moviendo las flechas de los extremos.

En la derecha, nos aparece una ventana con las **propiedades del strip**. Desde aquí podremos manipular parámetros como opacidad, velocidad, correcciones de color, etc.

Vamos a agregar un **segundo strip**, para continuar el video, para ellos seguimos el mismo procedimiento de antes. Este nuevo strip, lo vamos a colocar a continuación del anterior pero en el segundo canal.

4.9.2 Fundido de clips de video.

Si ponemos el segundo strip justo a continuación del primero, el cambio en ese frame será muy brusco, por tanto tenemos que decidir cómo queremos hacer la transición. En este caso vamos a hacer un fundido de 2 segundos entre los dos clip. Por lo tanto, movemos el segundo strip 48 frames para la izquierda.

Ilustración 228 – Clips superpuestos

En la ventana de previsualización, podemos ver como la imagen predominante es la del canal 2. Para poder hacer el fundido, tendremos que jugar con la **opacidad** del segundo strip. Para ello tendremos que animar este parámetro, ya que como comentábamos en el apartado de animación, casi todas las propiedades en Blender pueden ser animadas mediante **keyframes** [ver 4.7.1].

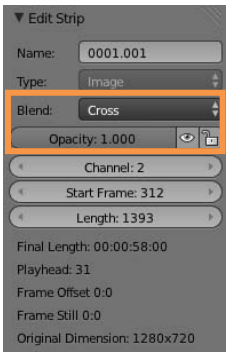


Ilustración 229 – Edit Strip

Para hacer la animación, nos situamos en el primer frame del segundo strip, y bajamos la opacidad a "0". Pulsamos con el botón derecho del ratón y seleccionamos "insert keyframe". Ahora repetimos la operación pero nos situamos en el último frame del primer strip y subimos la opacidad del segundo strip en 1.

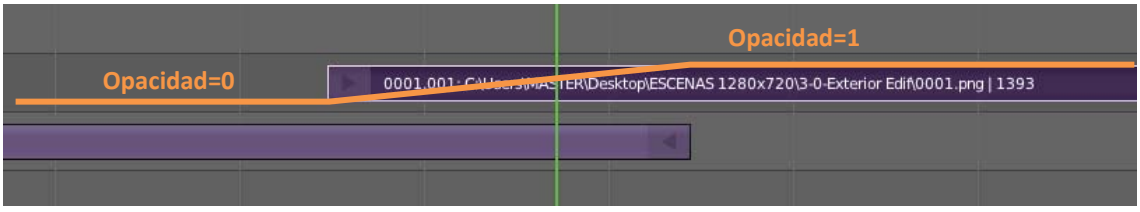


Ilustración 230 – Animación de opacidad

Ahora, en la previsualización, vemos como los dos videos se funden, para crear una transición más suave entre ellos.

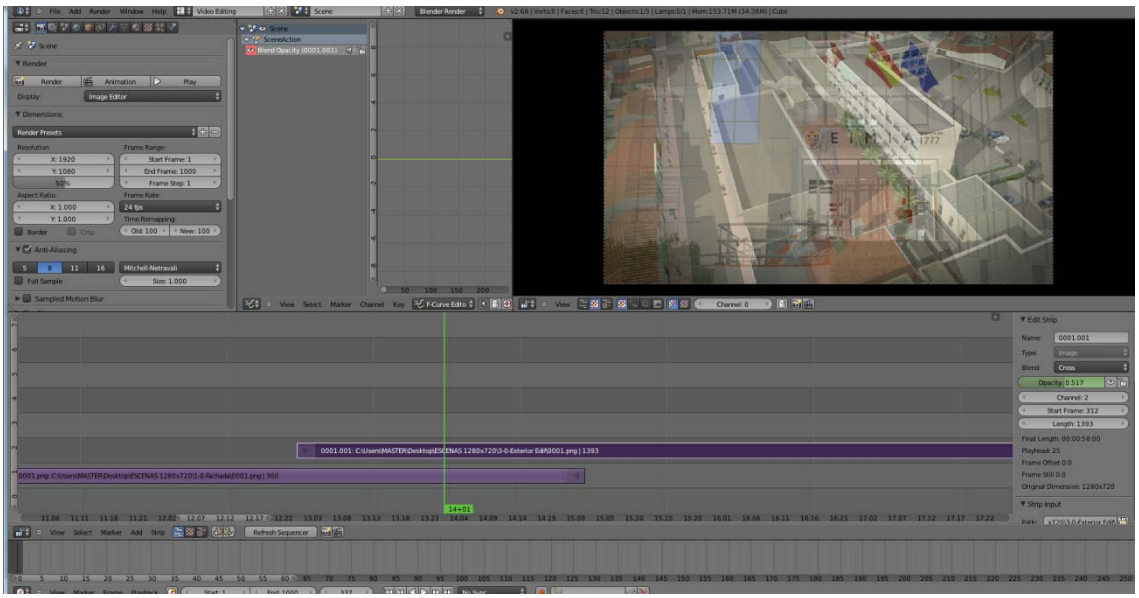


Ilustración 231 – Clips fundidos

4.9.3 Canal Alpha.

Otro efecto de fundido bastante interesante, es el uso de imágenes png con **canal alpha**. Para utilizarlo pondremos la imagen png en un canal superior a nuestro video. Si utilizamos la opción de fundido "**alpha over**", veremos la imagen en primer plano y el video a través del canal alpha de la imagen.

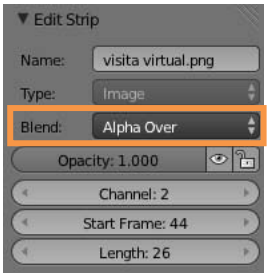


Ilustración 232 – Alpha over

Imaginemos que queremos introducir un título a la secuencia anterior. Prepararíamos una imagen con el texto del título y el fondo en el canal alpha. Al superponerlo en un canal superior, obtenemos el siguiente efecto.

Ilustración 233 – Superposición del canal alpha

4.9.4 Capa de ajustes.

Podemos generar una capa de ajustes que **afecte a todos los strip** de nuestro video. Esto muy útil si queremos crear un filtro o un efecto que afecte a la totalidad del video para homogeneizarlo.

Para crear esta capa, tenemos que seguir la ruta "Add→Strip Effect→Adjustment layers". Nos aparece un strip de **color verde**, para poder diferenciarlo claramente de los strip de video. Es importante que este strip esté en el canal superior, ya que solamente afecta a los clips que tiene en canales inferiores.

Esta capa de ajuste, la ajustamos para afecte a la totalidad de la escena, o al tramo que nos interese. En la ventana de propiedades de strip, podemos seleccionar entre diferentes **ajustes de color**. [Villar, 2014]

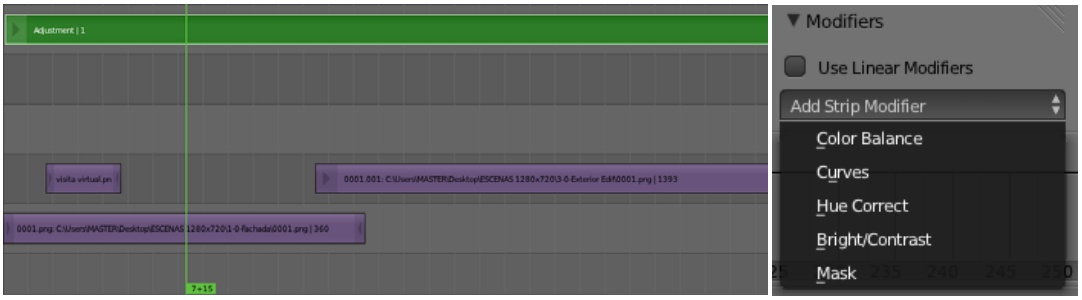


Ilustración 234 – Capa de ajustes

4.9.5 Audio.

Para insertar un archivo de **audio**, lo haremos de la misma manera que las imágenes, "Add→Sound", y seleccionamos el archivo de audio. Aparece en un color **cian**, para diferenciarlo de las imágenes. Este tipo de strip, podemos moverlo, recortarlo, etc. al igual que hacíamos con las imágines. Además desde las propiedades de strip, podemos modificar su volumen y otros aspectos del audio.

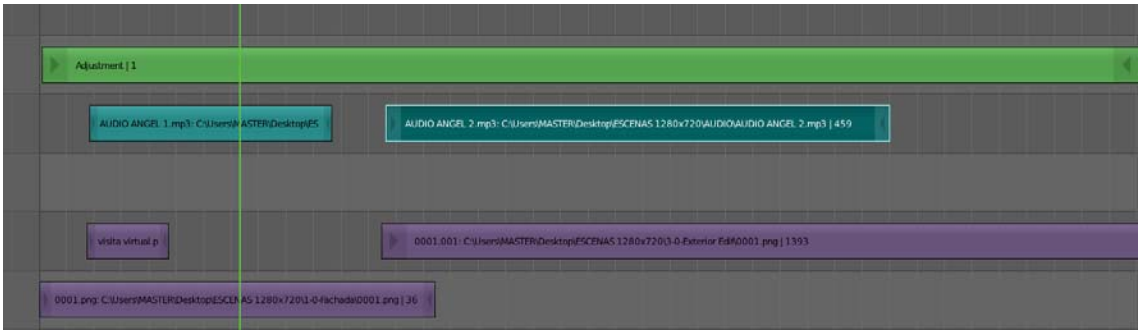


Ilustración 235 – Inserción del audio

4.9.6 Formato de salida.

Una vez tenemos todo nuestro video montado, tendrá un aspecto similar a la imagen.

Ilustración 236 – Video final

Al igual que hacíamos para renderizar las escenas, ahora tenemos que elegir las opciones de salida de nuestro video. Tendremos que especificar la **resolución** y los **frames por segundo**, es importante que estos parámetro de salida, sean iguales que los de las escenas de entrada.

En formato de salida, esta vez seleccionaremos uno de los **formatos de video**, según las características de compresión de cada uno de ellos. En este caso hemos elegido AVI con H.264 que es un código de video de alta compresión.

También tenemos que seleccionar el código de **compresión del audio**. En este caso hemos seleccionado mp3, que es el más común y además el formato del audio original.

Por último, seleccionamos la **ruta de salida** y pulsamos sobre el botón animación.

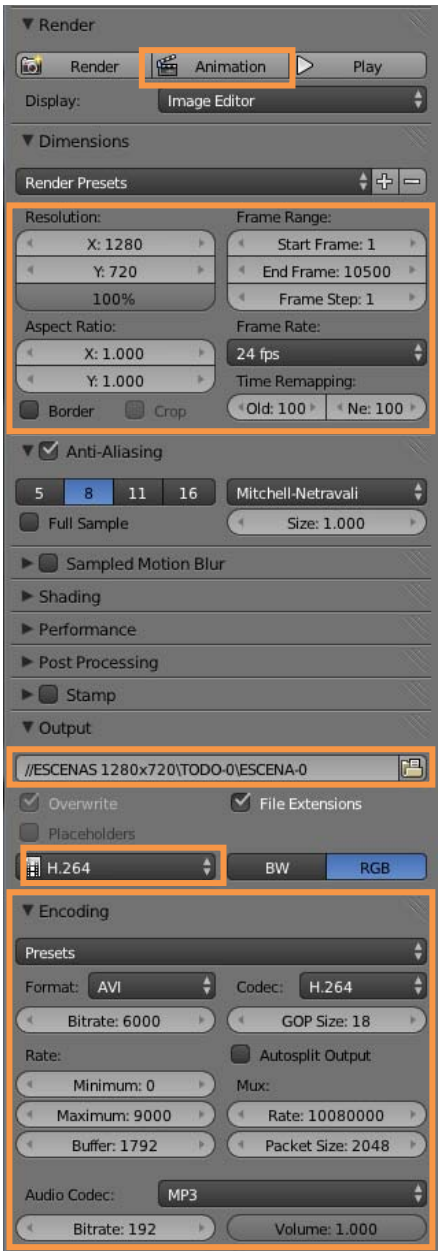


Ilustración 237 – Configuración de salida

SISTEMA INTERACTIVO

4.10 BLENDER GAME ENGINE

A lo largo de esta sección, vamos a ver algunas herramientas básicas para generar un **sistema interactivo** con Blender. En este caso, queremos crear un juego que nos permita recorrer el modelo virtual creado anteriormente, por lo que no se trata de un sistema muy complejo.

El **motor de videojuegos de Blender** "BGE", es bastante inferior al de los motores de juegos comerciales. En gran parte esto se debe a la **implementación incompleta** del mismo. Blender comenzó siendo una aplicación privativa de edición de videojuegos, y evolucionó hasta convertirse en un programa libre de edición 3D de uso general. En un momento dado de esta evolución, se reescribieron partes del programa y del editor de materiales para modernizarlo, pero el BGE no se actualizó completamente. Por ello, algunas de las técnicas que pueden emplearse en Blender, no funcionan en el BGE. Aunque por suerte, en las últimas versiones se es a poner mayor interés en el desarrollo de este motor. [Becerro, 2010] Pero aún con estas limitaciones, podemos generar muy buenos resultados, cuidando el modelado y los materiales.

Una de las principales ventajas del BGE, es el **sistema de nodos** que utilizamos para programar. El sistema de nodos es una interfaz constituida por **ladrillo lógicos**, que relacionamos entre sí mediante hilos. Los ladrillos, son conjuntos de datos de un determinado tipo y los hilos hacen **interactuar** unos con otros a gusto del usuario. De esta forma podemos generar resultado complejos, sin necesidad de ser un experto programador. Aunque también haremos uso de guiones de Python, el lenguaje de programación que utiliza Blender.

Para el desarrollo de un videojuego, al igual que para las animaciones, tenemos que tener muy claro el **concepto** de lo que queremos hacer antes de comenzar. Por lo tanto vamos a definir las **características** que queremos que tenga nuestro **sistema interactivo**:

- Pretendemos crear un videojuego en **primera persona**, que nos permita recorrer el modelo virtual creado. Por lo tanto, necesitaremos un **personaje.** Este personaje, solamente necesita interactuar con el medio, es decir chocar con las paredes, subir escaleras, etc. También, lógicamente, necesitará interactuar con el usuario. La idea es que el **usuario** pueda **mover** este personaje y a su vez elegir hacia dónde mira, es decir, que sea capaz de cambiar el **punto de vista**.
- Como cualquier videojuego, necesitamos también crear una **interfaz**. Para ello, vamos a desarrollar una **pantalla de inicio**, que nos de acceso al videojuego.
- La visita virtual, comenzará en un punto fijo, pero sería interesante que, mediante un plano, el usuario pudiese elegir el **punto de inserción** dentro del modelo. Así que vamos a integrar un **plano dentro de la interfaz**, para ese fin

Antes de empezar, tenemos que **activar BGE**. Para ello en la ventana "info", cambiaremos el motor de render por el motor de videojuegos.

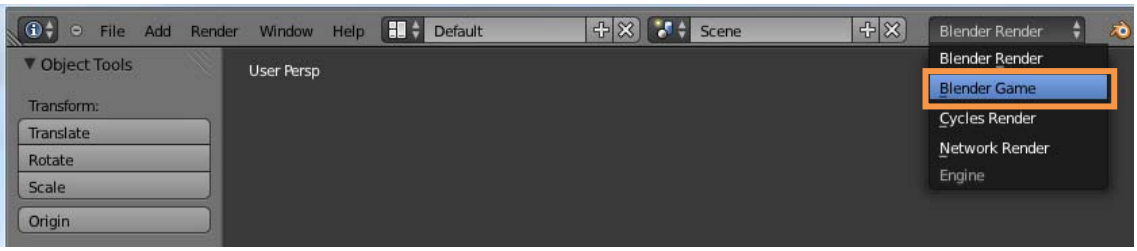


Ilustración 238 – BGE

Una vez activado, veremos que algunas pestaña de la ventana "property", varían algunas de sus funciones, como el editor de materiales y las opciones de configuración de los elementos de iluminación.

4.10.1 Editor de materiales.

En el editor de materiales, aparecen dos nuevos paneles: Game Setting y Physics. Desde ellos podremos modificar algunos parámetros relacionados con el motor de juegos y el simulador de física, como fricción y elasticidad.

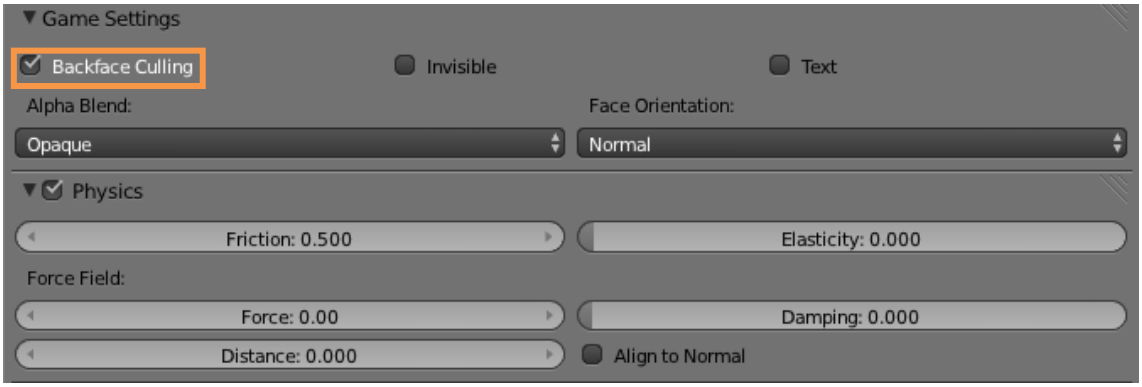


Ilustración 239 – BGE Editor de materiales.

Cuando creamos un material nuevo, por defecto viene activada la opción **Backface Culling**. Si esta acción está activada, cuando vemos una superficie en el motor de videojuegos, solamente es visible por uno de sus lados. Por lo que si en un objeto tenemos caras mal orientadas, estas caras serán completamente transparentes. Esto supuso un gran problema, pues cuando realicé el modelo, no tuve en cuenta esta opción y algunas de las caras no aparecían en el videojuego. Así que tuve que desactivarla de los 368 materiales, uno a uno.

En cuanto a las propiedades físicas de los materiales, las dejaremos como están, ya que por la naturaleza de nuestro juego no necesitamos interactuar con los materiales.

4.10.2 Iluminación.

Para la iluminación, hemos dejado el **sol** de la iluminación de la escena exterior, que será el encargado de generarnos las sombras. El tipo de luz "sun", tiene una diferencia importantísima en el BGE, y es que **no es infinita**. Por lo que tendremos que jugar con los parámetros de las sombras y la posición del sol para asegurarnos que cubre la totalidad de la escena. Para su configuración hemos utilizado los siguientes parámetros:

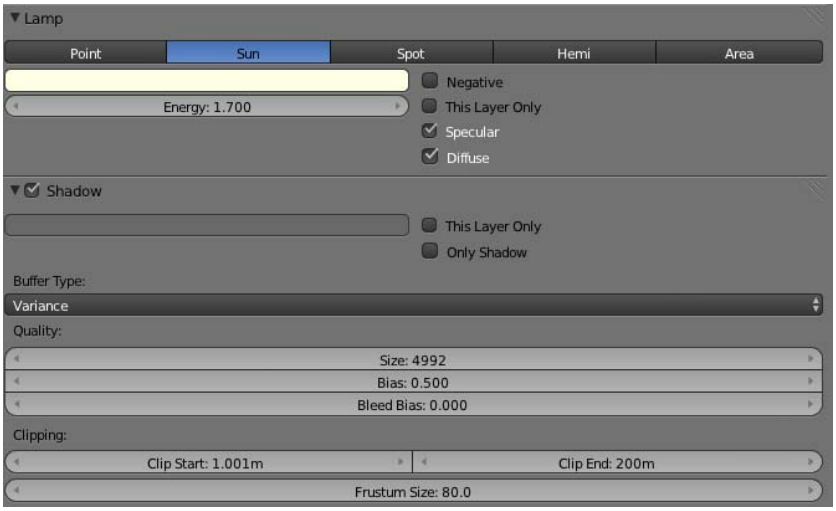


Ilustración 240 – BGE Sun.

Como ahora, no podemos utilizar parámetros como la oclusión ambiental, necesitaremos utilizar unas **luces secundarias** para iluminar las zonas en sombra. En este caso he utilizado cuatro luces Hemi, situadas en las esquinas, para asegurar que la luz llega de **todas direcciones**. [Becerro, 2010]

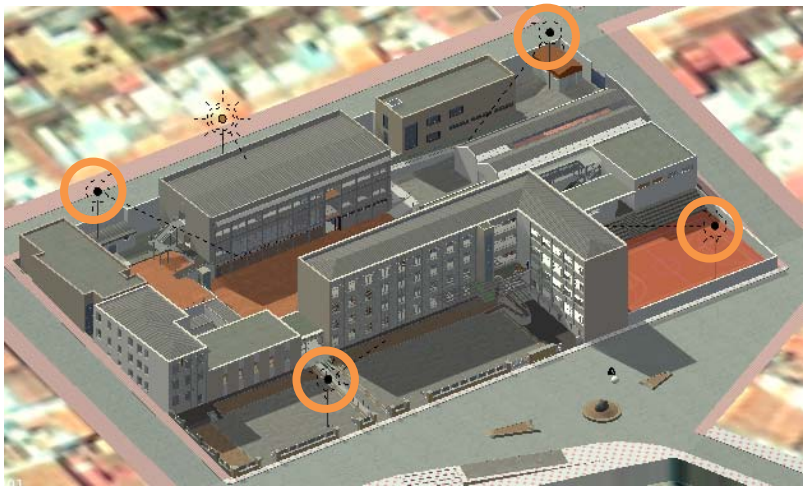


Ilustración 241 – BGE Iluminación de la escena.

4.11 CONFIGURACIÓN DEL PERSONAJE

Una vez reconfigurada nuestra escena, podemos configurar a nuestro personaje. Para ello, tendremos que configurar un **elemento dinámico asociado a una cámara**, que será la que nos proporcione la imagen del recorrido.

Como la figura de nuestro personaje es algo simbólico, no modelaremos un personaje, si no que usaremos algún **elemento simple** como un cubo o una esfera. Tras realizar algunas pruebas, nos hemos decantado por la **esfera**, ya que opone menos resistencia a la hora de subir escalones.

En primer lugar, insertamos en la escena una esfera de unos 40 cm de diámetro, que sería lo que ocupa el perfil de una persona y la renombramos como "personaje". Acto seguido añadimos la cámara. Tenemos que tener en cuenta que los ojos de una persona de estatura normal se encuentra a una altura aproximada de 1,60 - 1,70 metros. Por lo tanto teniendo en cuenta que la esfera "caminará" por el suelo desde su punto inferior, colocamos la cámara a esa altura y bien centrada con la esfera.

Ahora tenemos que anidar la cámara al esfera, para que la siga en su movimiento. Para hacerlo, seleccionamos la cámra y desde la ventan de propiedades, en la pestaña objeto, elegimos en "Parent" nuestro objeto personaje. Y aparecerá una línea de puntos entre la cámara y la esfera. [Penagos, 2012]

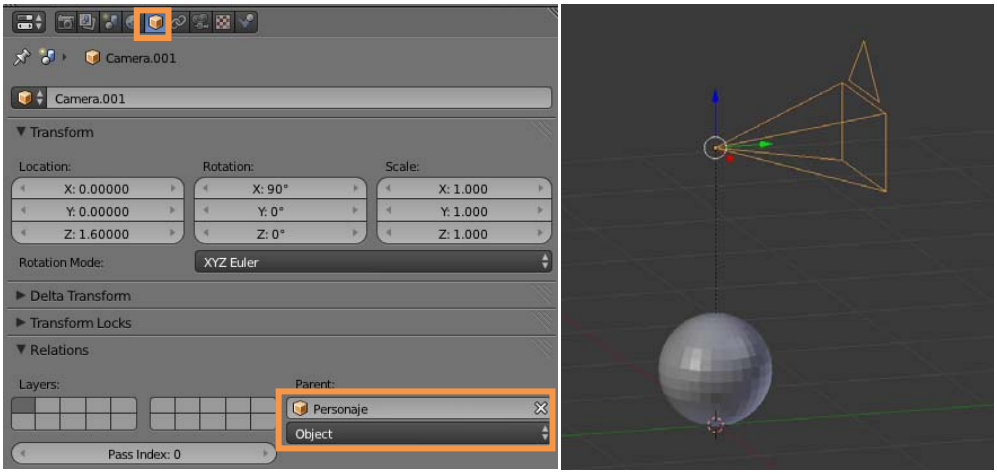


Ilustración 242 – BGE Personaje.

Por último, tenemos que indicarle al programa que la esfera va a ser nuestro personaje. Si seleccionamos la esfera y vamos a la pestaña "Physics" de la ventana de propiedades, encontramos la opción "Physics Type". Tenemos que seleccionar la opción "**Dynamic**", para indicarle que es un objeto que se va a mover. Una vez que hemos hecho esta selección, tenemos que activar la opción "**Actor**" y "**Invisible**", ya que nuestro personaje, no se va a ver.

También tenemos que cambiar la opción "**radio**". Esta opción hace referencia al contorno de nuestro objeto, con ella, podemos decidir a qué distancia chocamos con los demás objetos. De manera que la ajustamos al diámetro de nuestra esfera. [Gallardo, 2010]

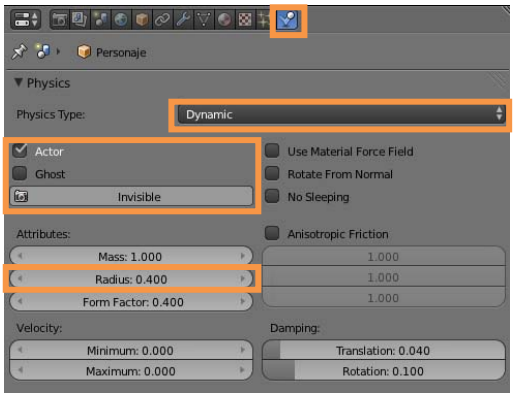


Ilustración 243 – BGE Física del Personaje.

Con esto, ya tenemos listo nuestro personaje para comenzar a programarlo. Así que vamos a definir las acciones y los controladores que utilizaremos para manejar nuestro personaje.

- **Teclado:** Usaremos el teclado para movernos **adelante, atrás y lateralmente**. Lo normal para estas acciones en este tipo de juegos es utilizar las teclas de dirección o la cruceta W-A-S-D. Por lo tanto configuraremos nuestro personaje con las dos opciones.
- **Ratón:** Usaremos el ratón para **mover la cámara y girar** nuestro personaje, de esta forma el usuario podrá mover el punto de vista en una posición fija.

He elegido esta combinación de teclas y ratón, ya que es la configuración estándar de cualquier juego en primera persona.

Ya tenemos todo listo para comenzar con el editor de lógica. Abrimos la ventana "Logic Editor" que tiene la siguiente apariencia.

Ilustración 244 – BGE Logic Editor

A la izquierda tenemos un **menú de propiedades**, donde podemos crear propiedades para la escena como un contador o una variable determinada. Por ahora, no utilizaremos este menú.

En la parte central de la ventana vemos tres columnas: **Sensores, controladores y Actuadores**. Aquí, mediante el sistema de nodos, configuraremos las acciones de nuestro juego.

Comenzaremos programando el movimiento de avance con la tecla de dirección "↑" y la tecla "**W**". **Seleccionamos la esfer**a y añadimos un **sensor "keyboard"**. En la casilla "key" pulsamos la tecla "↑" y aparecerá el texto "Up Arrow". Podemos renombrar a este ladrillo lógico como "adelante 1". Añadimos un segundo sensor de teclado, pero esta vez uitlizamos la tecla "W". A este sensor, lo renombramos como "adelante 2".

Ahora tenemos que añadir un **controlador** para estas teclas. Como queremos que funcione una u otra, tendremos que usar una puerta **lógica "or".** Unimos los dos sensores con el controlador.

Ya solamente nos falta definir la acción. Para ello, tenemos que fijarnos en la posición de la esfera. La dirección de avance será hacia el frente de la cámara. Por lo tanto, tenemos que avanzar en la coordenada local **Y** de la esfera.

Por lo tanto, añadimos un actuador del tipo "**motion**" que nos permite modificar la localización y la rotación de un objeto. En la localización (Loc) añadimos un avance de 0,1 en el eje Y.

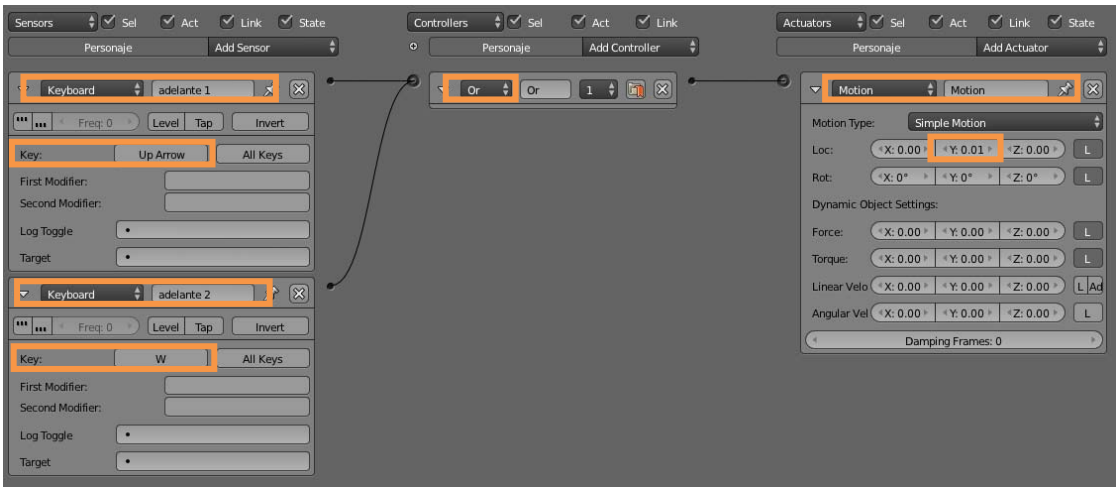


Ilustración 245 – Programación del movimiento de avance

Si pulsamos la tecla "**P**" para **previsualizar el videojuego**, vemos como nuestro personaje avanza cuando pulsamos la tecla "W" o la tecla "↑".

Por lo tanto, haremos lo misma operación para el resto de las acciones del teclado, teniendo en cuanta que el movimiento de retroceso tendrá un actuador de -0.1 en el eje Y, el movimiento lateral a la izquierda tendrá un actuador de -0.1 en X y el movimiento lateral a la derecha tendrá un actuador de 0.1 en X.

Una vez terminado el proceso, vemos cómo queda el editor de lógica.

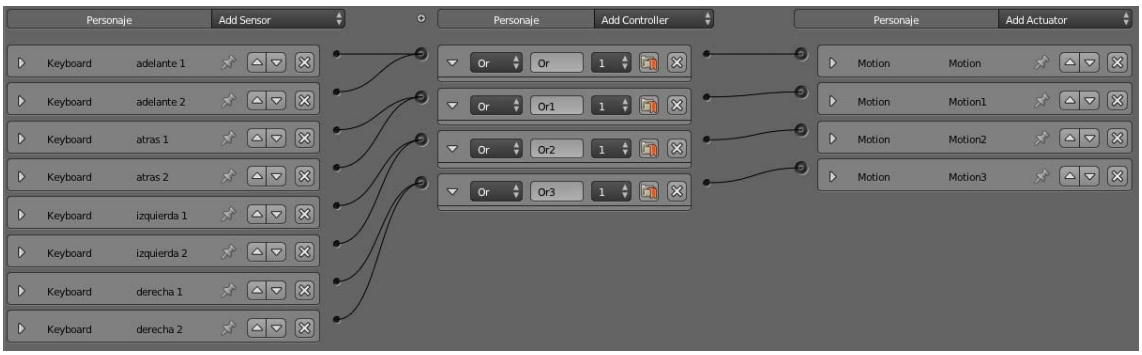


Ilustración 246 – Programación del teclado por nodos

Como vemos es muy sencillo programar acciones simples mediante los nodos del editor de lógica de Blender. Con esto ya tendríamos configurado el teclado. Ahora vamos a ver cómo realizaríamos la misma acción mediante **código Python**.

Para utilizar Python, tenemos que abrir el editor de textos de Blender, en la ventana "Text editor". El código para programar el teclado con las acciones anteriores es:

```
### Importamos el módulo de lógica de blender game engine BGE" ###
from bge import logic
def main():    ### Definimos programa principal ###
   controller = bge.logic.getCurrentController()  ### Definimos el controlador ###
   Player = controller.owner    ### Definimos el objeto ###
### Definimos las teclas ###
wKey = bge.logic.KX_INPUT_ACTIVE == keyboard.events[bge.events.WKEY]
upKey = bge.logic.KX_INPUT_ACTIVE == keyboard.events[bge.events.UPARROWKEY]
sKey = bge.logic.KX_INPUT_ACTIVE == keyboard.events[bge.events.SKEY]
downKey = bge.logic.KX_INPUT_ACTIVE == keyboard.events[bge.events.DOWNARROWKEY]
aKey = bge.logic.KX_INPUT_ACTIVE == keyboard.events[bge.events.AKEY]
leftKey = bge.logic.KX_INPUT_ACTIVE == keyboard.events[bge.events.LEFTARROWKEY]
dKey = bge.logic.KX_INPUT_ACTIVE == keyboard.events[bge.events.DKEY]
rightKey = bge.logic.KX_INPUT_ACTIVE == keyboard.events[bge.events.RIGHTARROWKEY]
### Definimos las acciones ###
   if wKey or upKey:
      Player.applyMovement((0, 0.1, 0), True)

   if sKey or downKey:
      Player.applyMovement((0, -0.1, 0), True)
```

```
    if aKey or leftKey:
        Player.applyMovement((-0.1, 0, 0), True)


    if dKey or rightKey:
        Player.applyMovement((0.1, 0, 0), True)
main()   ### Terminamos el programa principal ###
```

Guardamos el texto como "**teclado.py"**. Este código, podríamos utilizarlo en cualquier juego abriendo directamente este archivo.

Como vemos, para poder realizar la misma acción con Python, necesitamos conocer muy bien los **comandos del módulo lógica de Blender Game Engine**. Para aplicarlo a nuestro personaje, tendríamos que hacerlo desde el editor de lógica.

Tendríamos que crear un **sensor** del tipo "**Always**" para que se ejecute siempre el script y activar el botón TRUE level. Necesitamos también un **controlador** del tipo **Python** y cargar ahí nuestro archivo "teclado.py". En este caso, no es necesario un actuador, ya que está especificado en el código python.

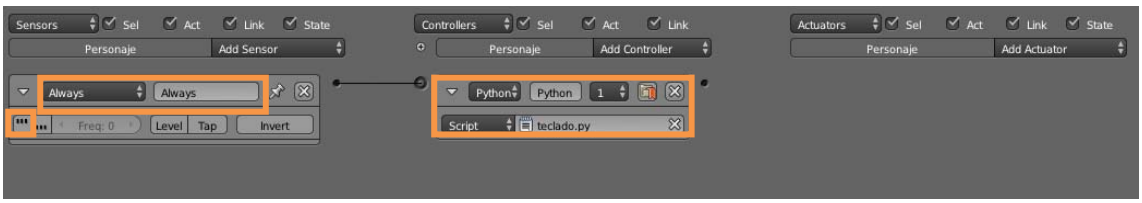


Ilustración 247 – Programación del teclado por Python

Para programar el movimiento de **giro y movimiento de la cámara** con el ratón, necesitaremos utilizar un código Python, ya que es una acción bastante complicada como para hacerla por nodos.

El código utilizado ha sido obtenido de *www.tutorialsforblender3d.com* y es el siguiente:

```
def main():  ### Definimos el programa principal
        ### Configuramos los valores por defecto del movimiento del ratón
        Sensitivity =  0.0005
        Invert = 1
        Capped = False
        ### Obtenemos los contralodores
        controller = bge.logic.getCurrentController()
        ### Obtenemos el objeto donde está aplicado el script
        obj = controller.owner
        ### Obtenemos la resolución de pantalla
        gameScreen = gameWindow()
        ### Obtenemos el movimiento de ratón
        move = mouseMove(gameScreen, controller, obj)
        ### Cambiar la sensibilidad del ratón?
        sensitivity =  mouseSen(Sensitivity, obj)
        ### Invertir el movimiento del ratón?
        invert = mousePitch(Invert, obj)
        ### usar mouse look
        useMouseLook(controller, capped, move, invert, sensitivity)
        ### Centrar ratón en la pantalla
        centerCursor(controller, gameScreen)
```

```
############################################################
### definir ventana de juego
def gameWindow():

    ### Obtener ancho y alto de la ventana
    width = bge.render.getWindowWidth()
    height = bge.render.getWindowHeight()

    return (width, height)
############################################################
### definir la función del movimiento del ratón
def mouseMove(gameScreen, controller, obj):
    ### Obetener el sensor llamado "MouseLook"
    mouse = controller.sensors["MouseLook"]
    ### Obtener ancho y alto de la pantalla de juego
    width = gameScreen[0]
    height = gameScreen[1]
    ### Distancia movida desde el centro de la pantalla
    x = width/2 - mouse.position[0]
    y = height/2 - mouse.position[1]
    ### Iniciar ratón si no se arrancó por primera vez
    if not 'mouseInit' in obj:
        obj['mouseInit'] = True
        x = 0
        y = 0
############################################################
```

```
############################################################
### Definir sensibilidad del ratón
def mouseSen(sensitivity, obj):
	### Comprobar si hay una propiedad de ajustes añadida
	if 'Adjust' in obj:
		### No queremos ajustes con valores negativos
		if obj['Adjust'] < 0.0 or obj['Adjust'] == 0:
			obj['Adjust'] = sensitivity
		### Ajustar sensibilidad del ratón
		sensitivity = obj['Adjust'] * sensitivity
	### Devolver sensibilidad
	return sensitivity
############################################################
### Definir inversión del movimiento del ratón
def mousePitch(invert, obj):
	### Comprobar si hay una propiedad de ajustes añadida llamada Invert
	if 'Invert'in obj:
		### Invertir?
		if obj['Invert'] == True:
			invert = -1
		else:
			invert = 1
	### Devolver valor
	return invert
############################################################
```

```
############################################################
### Definir useMouseLook
 def useMouseLook(controller, capped, move, invert, sensitivity):
        ### Arriba/Abajo
        upDown = move[1] * sensitivity * invert
        ### Izquierda/Derecha
        leftRight = move[0] * sensitivity
        ### Obtener actuadores
        act_LeftRight = controller.actuators["LeftRight"]
        act_UpDown = controller.actuators["UpDown"]
        ### Configurar valores
        act_LeftRight.dRot = [ 0.0, 0.0, leftRight]
        act_LeftRight.useLocalDRot = False
        act_UpDown.dRot = [ upDown, 0.0, 0.0]
        act_UpDown.useLocalDRot = True
        ### Usar actuadores
        controller.activate(act_LeftRight)
        controller.activate(act_UpDown)
################################################
### Definir centro del cursor
def centerCursor(controller, gameScreen):
        ### Extraer ancho y alto de la pantalla de juego
        width = gameScreen[0]
        height = gameScreen[1]
        ### Obtener sensor llamado MoouseLook
        mouse = controller.sensors["MouseLook"]
        ### Obtener posición del cursor
        pos = mouse.position
```

```
        ### Centrar cursor si es necesario
        if pos != [int(width/2), int(height/2)]:
                bge.render.setMousePosition(int(width/2), int(height/2))
        else:
                # Obtener actuadores
                act_LeftRight = controller.actuators["LeftRight"]
                act_UpDown = controller.actuators["UpDown"]
                # Apagar actuadores
                controller.deactivate(act_LeftRight)
                controller.deactivate(act_UpDown)
##################################################
import bge ### Importamos BGE logic
main()### Iniciamos el programa principal
```

Como vemos este script es bastante más complicado. Este script, tenemos que aplicarlo en la cámara. Creamos un **sensor "mouse"** de **movimiento** y tenemos que ponerle el nombre que nos indica el código, "**MouseLook**". También tenemos que crear un **controlador "Python"** donde cargaremos el script. Ahora tenemos que crear un **actuador "motion"** en la **cámara**, llamado "**UpDown**" y otro actuador en la **esfera** llamado "**LeftRight**".

El editor de lógica debe quedar así:

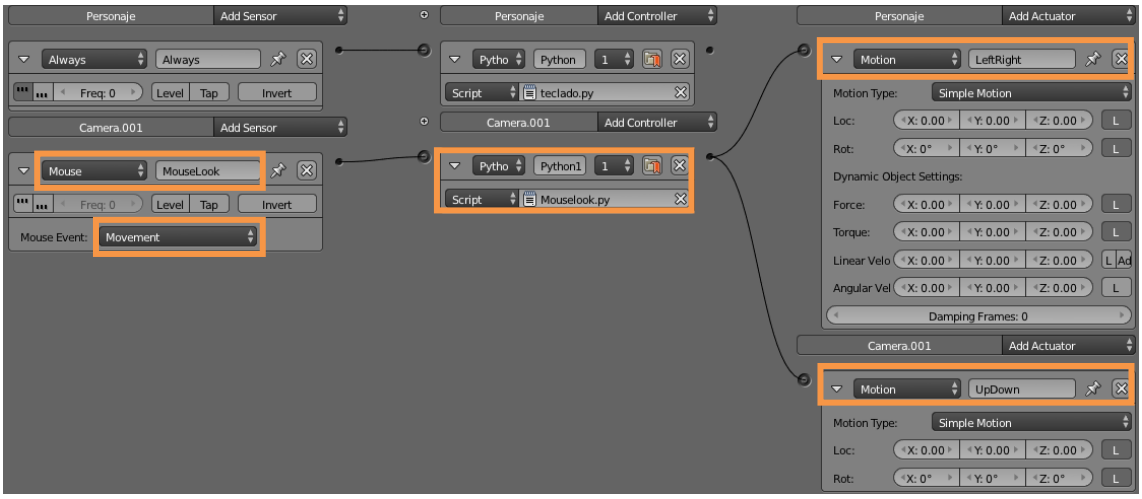


Ilustración 248 – Programación del ratón por Python

Por lo tanto, una vez terminada la configuración de las acciones de nuestro personaje, utilizamos la tecla "P" para previsualizar el juego. Lo primero que tenemos que hacer, es **comprobar** todas las teclas y las **acciones** del ratón para que todo funcione correctamente. Una vez que hemos comprobado que nuestro personaje funciona correctamente, tenemos que puede moverse por todo el **modelo tridimensional**. Por lo tanto nos dedicamos a recorrer el modelo varias veces en busca de errores.

Es muy importante que para que nuestro personaje pueda **colisionar** contra tabiques, suelos, muros, y demás **objetos de la escena**, que estos estén configurados en la pestaña de **física** como **objetos estáticos** "Static". Esto no debe suponer un problema, ya que es la configuración **por defecto** de cualquier objeto al insertarlo en la escena.

Por último, tenemos que **acotar el campo de movimiento** de nuestro personaje. Es decir, pone un límite físico invisible, de manera que el usuario no pueda salir de ese "recinto". Crearemos un cubo que "rodeará" el contorno de la Escuela y desactivaremos la opción de visualización en el render desde la ventana de la ventana Outliner.

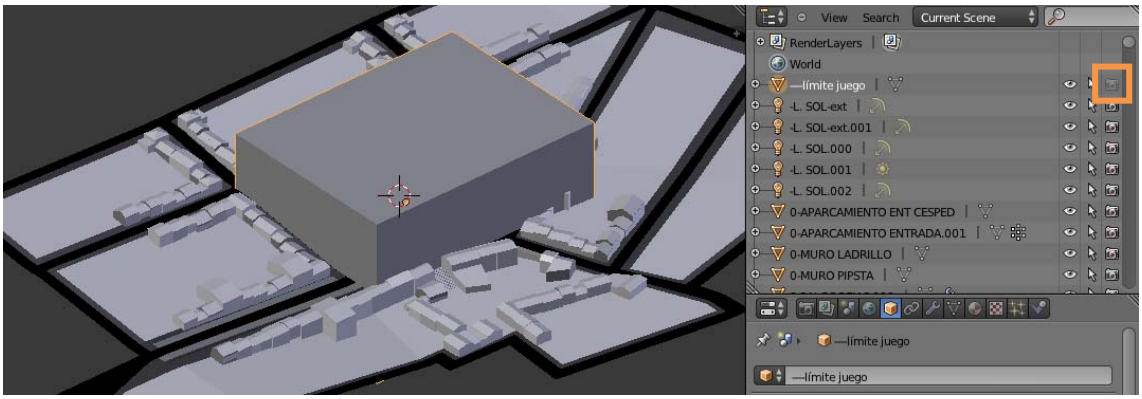


Ilustración 249 – Límite videojuego

Por lo tanto, ya te tenemos configurado nuestro sistema interactivo para que el usuario pueda interactuar con el modelo tridimensional. Ahora vamos a ver cómo crear una interfaz simple para un videojuego.

4.12 CONFIGURACIÓN DE LA INTERFAZ

En primer lugar, tenemos que pensar cómo queremos que sea nuestra interfaz. Para ello, realizamos un esquema o mapa de la misma. De esta forma, podremos ver las acciones que tendremos que programar para cada una de las pantallas.

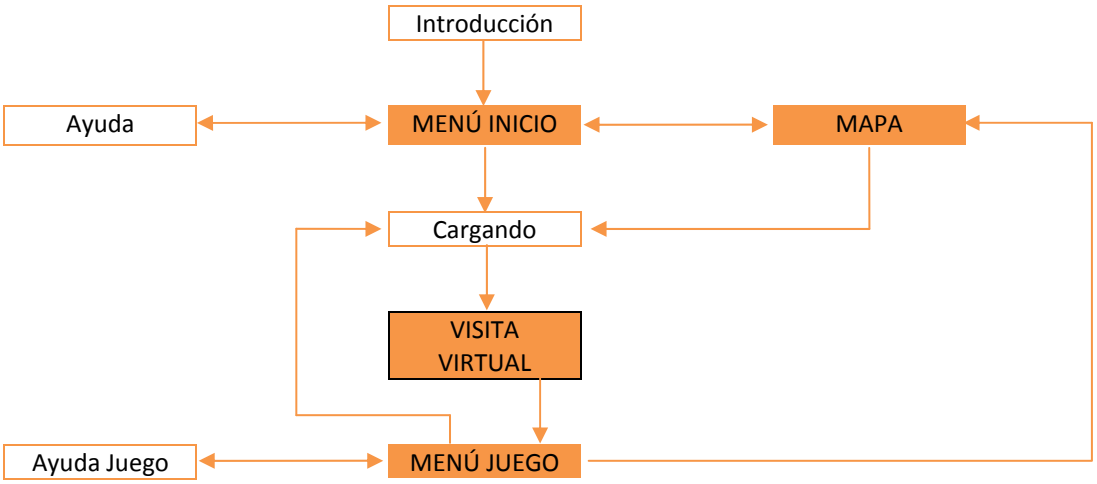


Ilustración 250 – Mapa interfaz

En primer lugar, tendremos una **pantalla inicial** con unos **créditos**, que nos llevará al menú de inicio. Desde este **menú** podremos elegir 4 opciones. Podremos **iniciar el juego** o acceder a una pantalla de **ayuda** donde se explicarán los controles de la visita. También podremos cargar una pantalla **mapa** con el plano de la Escuela y elegir el punto de inserción en el modelo tridimensional. Y por último la opción **salir**.

Tanto el mapa, como iniciar el juego llevan a una **pantalla de carga**. Debido al tamaño del modelo tridimensional, el tiempo de carga oscila entre 40 y 60 segundos, por lo tanto creí conveniente que apareciese esa pantalla, en lugar de un fondo negro donde parecía que el juego "no funcionaba".

Una vez en el juego, podemos pausarlo para acceder al **menú de juego**, que será igual que el de inicio pero con la opción **continuar el juego** en lugar de iniciar el juego. Este menú nos obliga a crear algún sistema de **guardado y carga de la posición** del jugador en el momento que decide entrar a este menú, por si decide continuar donde lo dejó.

Para poder guardar y cargar la posición, necesitaremos crear un **script en Python**, que nos cree un fichero **"posición.txt"**, donde guardaremos los datos del **vector de posición** y de la **matriz de rotación**. Este archivo será leído cada vez que iniciamos el juego y guardado cada vez que salgamos de él.

El método de trabajo será por **escenas**. Cada uno de los apartados anteriores, tendrá su propia escena dentro del archivo de Blender.

4.12.1 Pantallas de inicio.

En primer lugar, renombraremos la escena donde se encuentra el modelo y el personaje como "juego". Esto podemos hacerlo desde la ventana info. Si pulsamos sobre el botón "+", crearemos una **nueva escena vacía**. A esta escena la llamaremos "0-Intro".

Ilustración 251 – Administrar escenas

Para la introducción, vamos a mostrar unos créditos muy breves. Constará de tres pantallas que se reproducirán en orden y en las que aparecerán estas imágenes:

Ilustración 252 – Créditos iniciales

Configuraremos la escena de la siguiente manera. Crearemos un **plano** con las proporciones de las imágenes y lo **copiaremos** dos veces, variando levemente su **posición en el eje Z**. Crearemos tres **materiales** con las imágenes y las aplicaremos en orden a los planos. Una vez hecho esto, colocaremos una **cámara** en posición vertical, frontalmente a los planos.

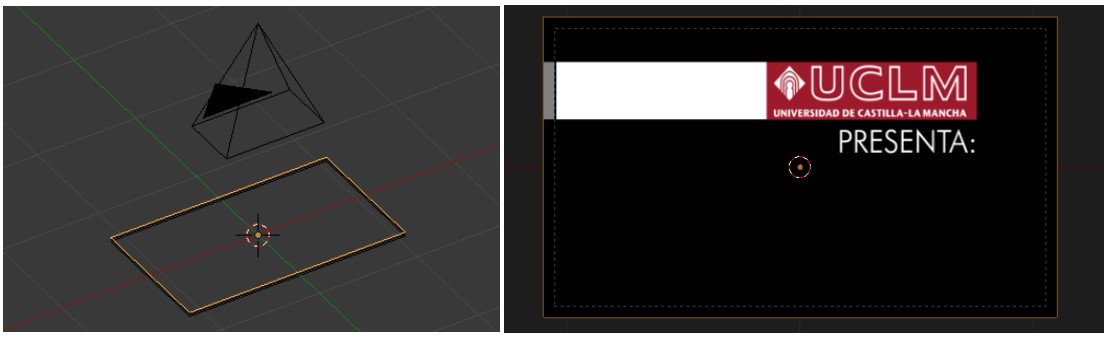


Ilustración 253 – Configuración de la escena inicial

Ahora, tenemos que programa la escena para que los planos vayan desapareciendo cada cuatro segundos. Para poder hacer esto, tenemos que crear una **propiedad tiempo** para la escena en las propiedades del editor de lógica.

Pulsamos sobre **Add Game Property** y elegimos un "**timer**" y lo llamamos "tiempo". Lo configuramos para que empiece la cuenta en 0.000.

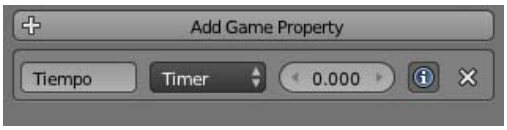


Ilustración 254 – Timer

Ahora seleccionamos el primero de los planos en desapareces. Le añadimos un **sensor** de propiedades "**Property**" y lo configuramos de la siguiente manera. En tipo de evaluación "**Evaluation Type**" seleccionamos "**Equal**". En propiedad, seleccionamos nuestro timer "**tiempo**". Y en valor lo configuramos para que realice una acción cuando tiempo sea igual que **4**. Ahora añadimos un **controlador** tipo "**And**" y añadimos un **actuador** de visibilidad "**visibility**" donde indicaremos la opción "**End Object**".

Por lo tanto, cuando el cronómetro llegue al valor de 4 segundos, este plano desaparecerá y dejará visible el inferior.

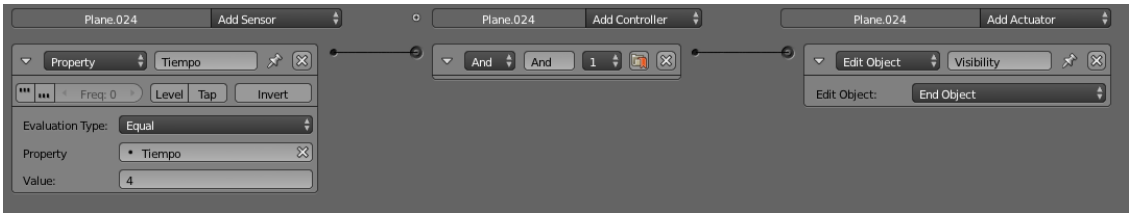


Ilustración 255 – Programar visibilidad mediante tiempo

Haremos lo mismo para el segundo plano, pero con un valor de 8 segundo. Para el **tercero** utilizaremos la misma configuración con un valor de 12 segundos, pero esta vez el **actuador** no será de visibilidad sino de **escena.** Configuraremos este actuador para cambiar de escena seleccionando "Set scene" y seleccionaremos la escena siguiente, en este caso "1-Menu" que tendremos que crear previamente. [Penagos, 2012]

Ilustración 256 – Programar cambio de escena mediante tiempo

4.12.2 Pantalla de Menú.

Ahora tendremos una **nueva escena vacía** llamada "1-Menú". Desde donde podremos elegir diferentes opciones, como comentábamos anteriormente. Para esta escena, utilizaremos un **plano**, donde colocaremos la imagen del menú, y **una cámara** en posición frontal al mismo.

Ilustración 257 – Pantalla Menú

Necesitaremos también crear un cursor para realizar las selecciones. Para este juego, hemos creado un circulo que rodeará los puntos blancos de la imagen.

A continuación, tendremos que crear una animación para que nuestro botón selección pueda moverse. Para ello nos posicionamos en el frame 1, primera opción, situamos el cursor sobre "comenzar" y añadimos un keyframe de localización "Loc". En el frame 2, los posicionaremos sobre "mapa" y añadimos otro keyframe. Repetimos la operación en el frame 3 para "ayuda" y en el frame "4" para salir. [Penagos, 2012]

Ilustración 258 – Cursor selector

Antes de continuar configurando la escena, este parece un buen momento para que el juego cree el fichero **posición.txt** con la posición inicial del jugador, que nos situará frente a la puerta de entrada del edificio Störr. [Molón, 2014] Para ello utilizaremos un **sensor "always"** y un **controlador "Python"** con el siguiente script al que llamaremos "reiniciar.py" y lo aplicaremos a nuestro cursor:

```
### Abrir archivo
saveFile = open("posicion.txt", "w")
### Cabecera
saveFile.write("Este es un archivo valido" + "\n")
### Escribimos datos del vector de posición y la matriz de rotación por filas
saveFile.write("7.53053" + "\n")
saveFile.write("-35.67547" + "\n")
saveFile.write("-5.81934" + "\n")
saveFile.write("1" + "\n")
saveFile.write("0" + "\n")
saveFile.write("0" + "\n")
saveFile.write("0" + "\n")
saveFile.write("1" + "\n")
saveFile.write("0" + "\n")
saveFile.write("0" + "\n")
saveFile.write("0" + "\n")
saveFile.write("1" + "\n")
## Cerramos archivo
saveFile.close()
print("posicion reiniciada")
```

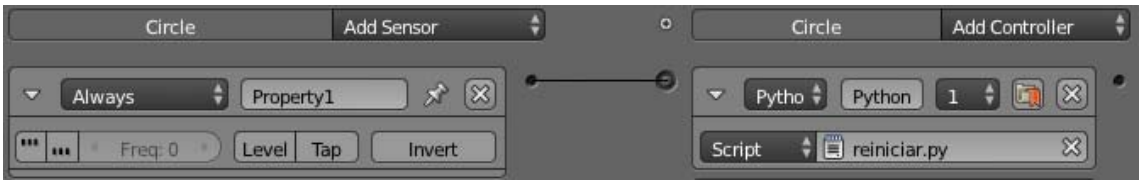


Ilustración 259 – Cargar posición inicial

Con este script, cada vez que iniciemos el juego se creará el archivo con los datos de posición y se borraran datos guardados de otras partidas.

Ahora tenemos que seguir configurando las acciones de nuestro cursor. [Penagos, 2012]

Nuestro cursos dispone de cuatro posiciones, por lo tanto necesitamos un contador que cuente en que posición nos encontramos dependiendo de las acciones que realice el usuario. Cuando aparece el menú, frame 1, nuestro cursor se encuentra en la primera opción, por tanto nuestro contador debe empezar en la posición 1.

Para crear un contador, creamos una propiedad "Integer" en las propiedades del editor de lógica. Lo renombramos como "opción" y lo situamos en el valor 1.

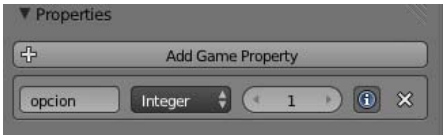


Ilustración 260 – Integer

Para movernos por el menú, utilizaremos las teclas "izquierda" y "derecha", y la tecla "Intro" para confirmar la selección. Por lo tanto, cuando el usuario pulse la tecla derecha, tenemos que sumarle +1 al contador, hasta un máximo de 4 y cuando el usuario pulse la tecla izquierda sumarmos -1 al contador hasta un mínimo de 1.

Para la **tecla derecha**, creamos dos sensores, un **sensor "keyboard"** donde especificaremos la "**flecha derecha**" y un **sensor "propety" de intervalo** donde indicaremos que solamente funcione esta tecla en las posiciones 1,2 y 3.

Uniremos estos dos sensores mediante un **controlador "And"**. Y por último las acciones. Cuando pulsamos la tecla derecha, tienen que pasar dos cosas, que sumemos +1 al contador y que nuestro cursor, se mueva a la posición dos.

Añadimos primero un **actuador "property"**, seleccionamos la propiedad donde queremos actuar, en este caso **"opción"** y pondremos el modo añadir **"Add"** con un **valor de 1**.

En segundo lugar, añadimos un **actuador "Action"**, donde selecionaremos la propiedad **"opción"** y la **animación de nuestro cursor**. De esta manera asociamos el valor de esta propiedad a los frames de nuestro cursor.

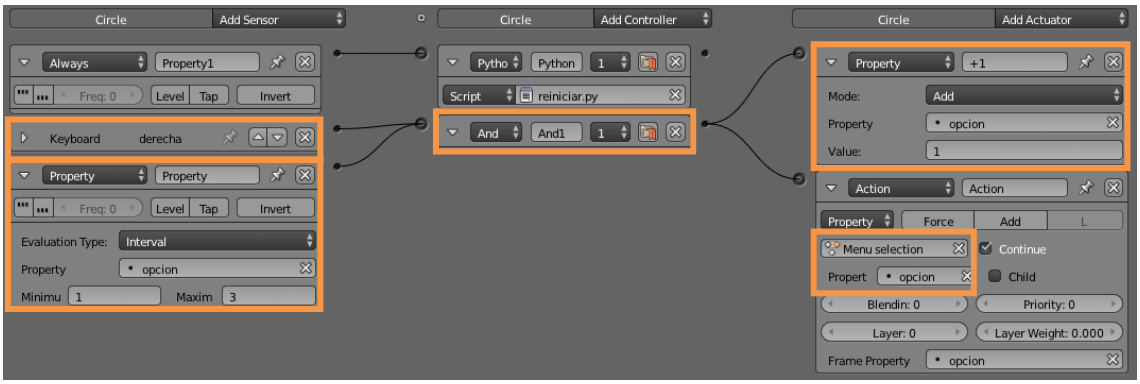


Ilustración 261 – Configuración cursor menú

Para la **tecla izquierda**, realizaremos la misma configuración, pero en el intervalo del sensor "property" le especificaremos que solamente funcione de la **posición 2 a la 4**. Y en el actuador "Property" pondremos un **valor de -1**.

Ahora ya podemos cambiar el cursor de posición y el program es capaz de detectar en que posición se encuntra, por lo tanto, ahora tendremos que configurar la tecla para **seleccionar la opción**, en este caso la tecla **"intro"**.

Tenemos que añadir al cursor, un **sensor "keyboard"** donde introducimos la tecla "intro" y **4 sensores "property" de igualdad**, uno para cada valor de las opciones. Renombraremos esto cuatro sensores con el nombre de su acción: (1) Empezar juego, (2) Mapa, (3) Ayuda y (4)Salir.

Acontinuación, agregamos **4 controladores "And", uno para cada sensor "property"**, y unimos el bloque lógico de la **tecla "intro" con los cuatro controladores**.

Por último creamos los actuadores. Necesitaremos **3 actuadores** de cambio de escena **"scene"** para las opciones comenzar juego, mapa y ayuda. Y **actuador "game"** con la opción **"Quit Game"**, para salir del juego.

Por lo tanto el editor de lógica de nuestro cursor, queda de la siguiente manera:

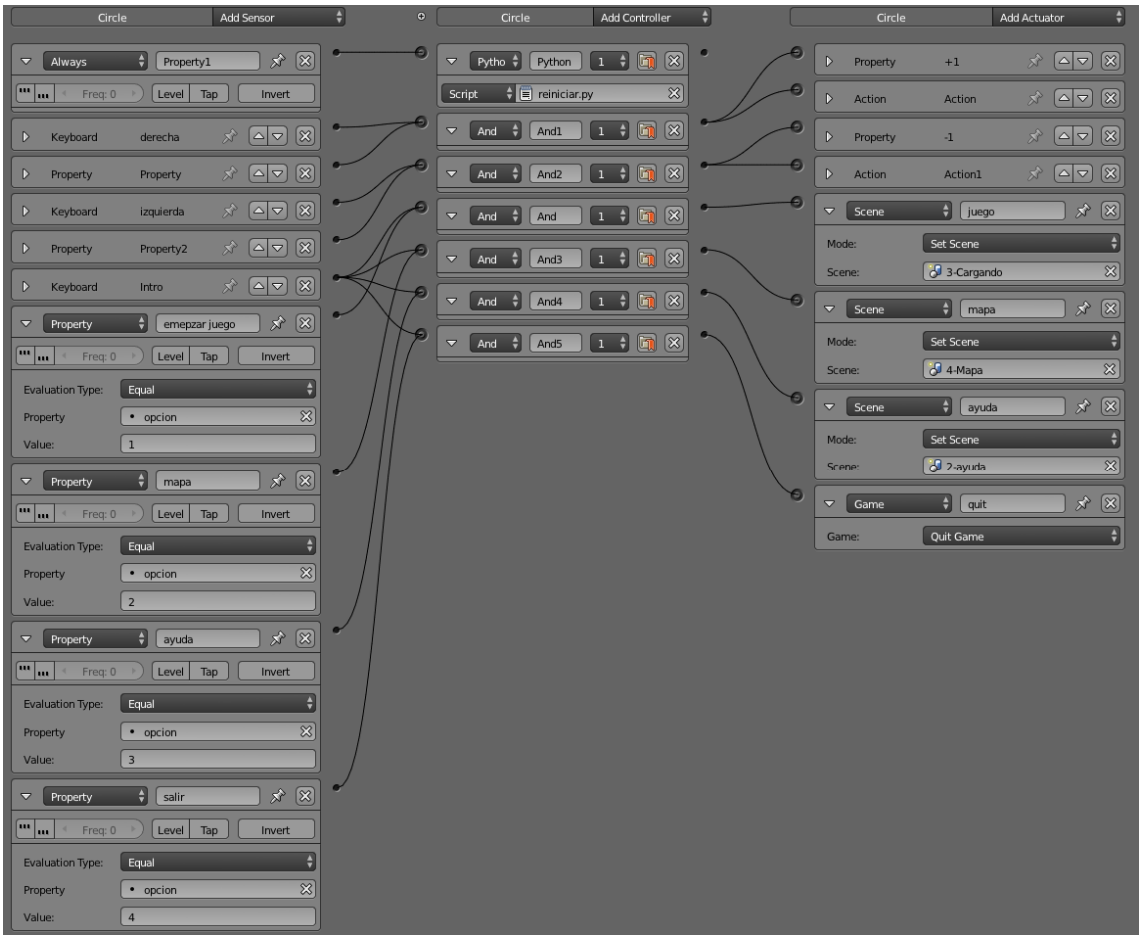


Ilustración 262 – Configuración final del cursor menú

Para configurar el **menú de juego**, duplicaremos esta escena y cambiaremos la imagen por la del menú de juego, que es igual, pero en lugar de comenzar, pone continuar. Tendremos que eliminar el controlador "reiniciar.py", ya que ya tenemos creado el archivo "posición.txt". Y cambiar las escenas de destino en función del organigrama de la interfaz.

4.12.3 Pantalla ayuda.

Esta pantalla, no requiere una programación compleja, ya que solamente será una imagen donde mostraremos al usuario los controles del juego. Por lo que solamente tendremos que crear un sensor de "keyboard" con la tecla "Esc" unido a un controlador "And" y con un actuador "scene" para volver al menú correspondiente.

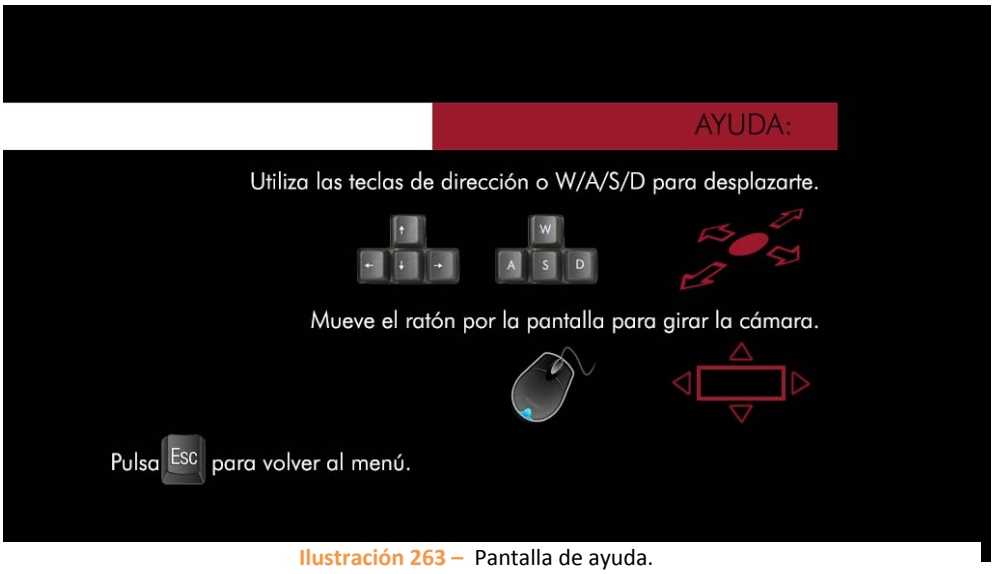


Ilustración 263 – Pantalla de ayuda.

4.12.4 Pantalla de carga.

Esta pantalla, aparece mientras se carga el juego. Es una pantalla de transición por lo que únicamente tendremos que añadir un **sensor "always"**, un **controlador "And"** y **actuador de cambio de escena "scene"** hacia el juego. Por lo tanto, en cuanto cargue esta página, automáticamente comenzará a cargarse el juego.

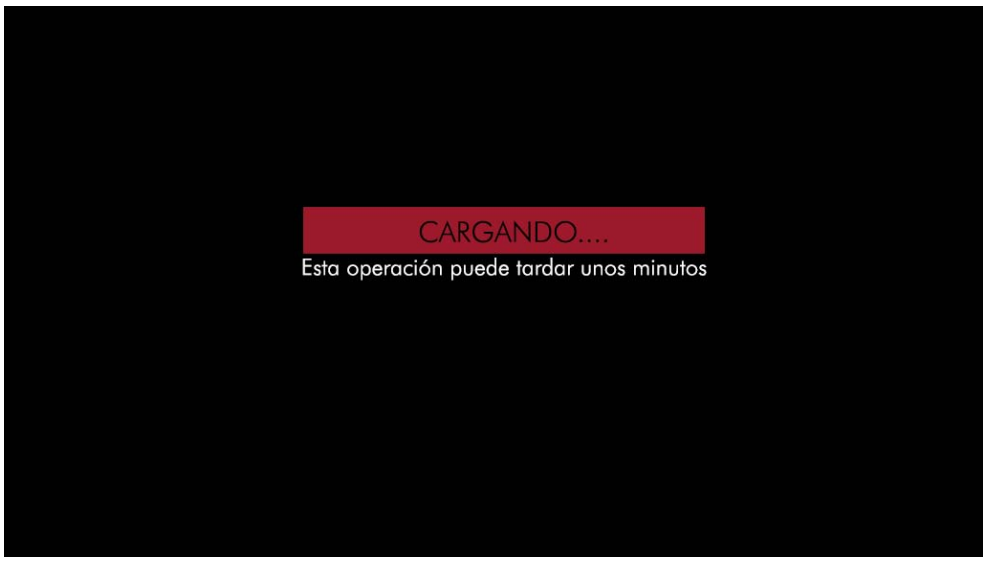


Ilustración 264 – Pantalla de carga

4.12.5 Mapa.

Mediante esta pantalla, el usuario podrá acceder de forma automática a diferentes puntos preestablecidos de la Escuela. En la imagen podemos ver las diferentes opciones. En total, he preestablecido 12 puntos. Esta escena es un poco más compleja, ya que tendremos que preparar un script de python para cada uno de los puntos. Cada uno de estos script, reescribirá el fichero "posición.txt" con el vector de posición y la matriz de rotación correspondiente a esa posición.

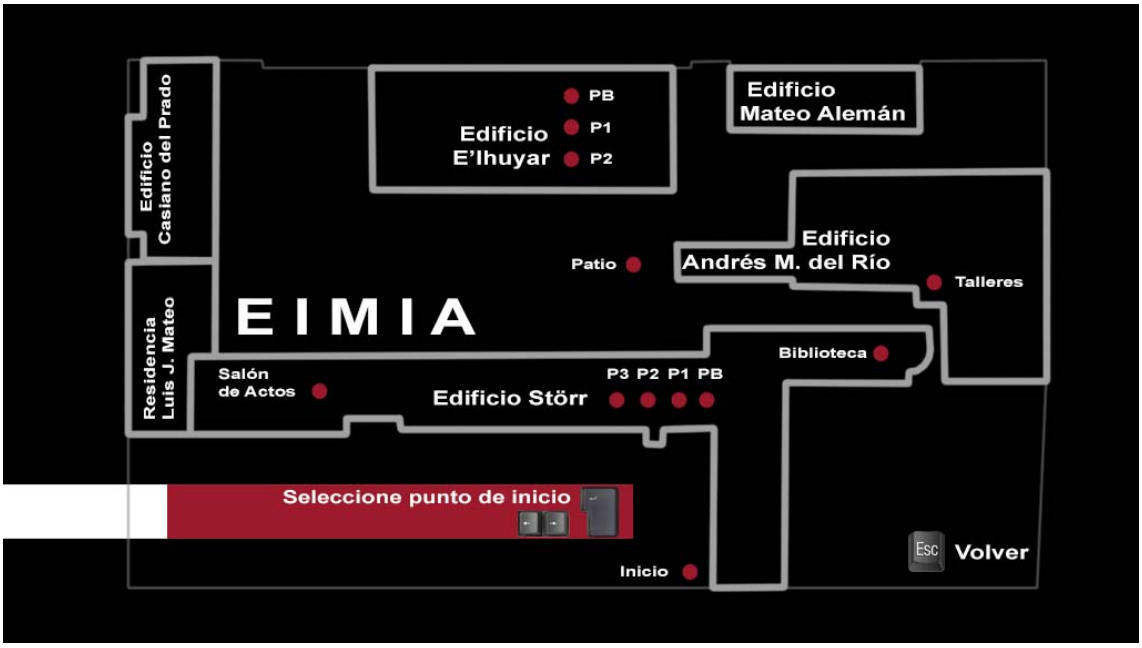


Ilustración 265 – Pantalla de mapa

El sistema de funcionamiento, será igual que para la **escena del menú principal,** tendremos un **cursor** y tendremos que preparar doce fotogramas con la posición correspondiente a cada uno de los puntos. La **estructura del editor de lógica** para las teclas izquierda y derecha, será **igual que en el menú**, pero cambiando los intervalos de valores. Además tendremos que programar la **tecla "Esc"** para volver al menú principal y la **tecla "intro"** para iniciar el juego.

Ahora, utilizaremos la siguiente fórmula para modificar el archivo "posición.txt". Crearemos **12 sensores del tipo "property" de igualdad**, uno para cada uno de los puntos y los uniremos a **12 controladores "Python"**, cada uno con un script que modificará el archivo "posición.txt" para escribir el **vector de posición** y la **matriz de rotación** correspondiente.

Este sería el script para el salón de actos:

```
### Abrir archivo
saveFile = open("posicion.txt", "w")
### Cabecera
saveFile.write("Este es un archivo valido" + "\n")
### Escribimos datos
saveFile.write("-41.61815" + "\n")
saveFile.write("4.6168" + "\n")
saveFile.write("-2.85818" + "\n")
saveFile.write("0" + "\n")
saveFile.write("-1" + "\n")
saveFile.write("0" + "\n")
saveFile.write("1" + "\n")
saveFile.write("0" + "\n")
saveFile.write("0" + "\n")
saveFile.write("0" + "\n")
saveFile.write("0" + "\n")
saveFile.write("1" + "\n")
## Cerramos el archivo
saveFile.close()
```

De esta forma, cada vez que pasemos por un punto, reescribimos el fichero posición. Cuando aceptemos la opción y vayamos al juego, éste cargará la información de ese fichero.

De esta forma, el editor de lógica de esta pantalla, queda de la siguiente forma:

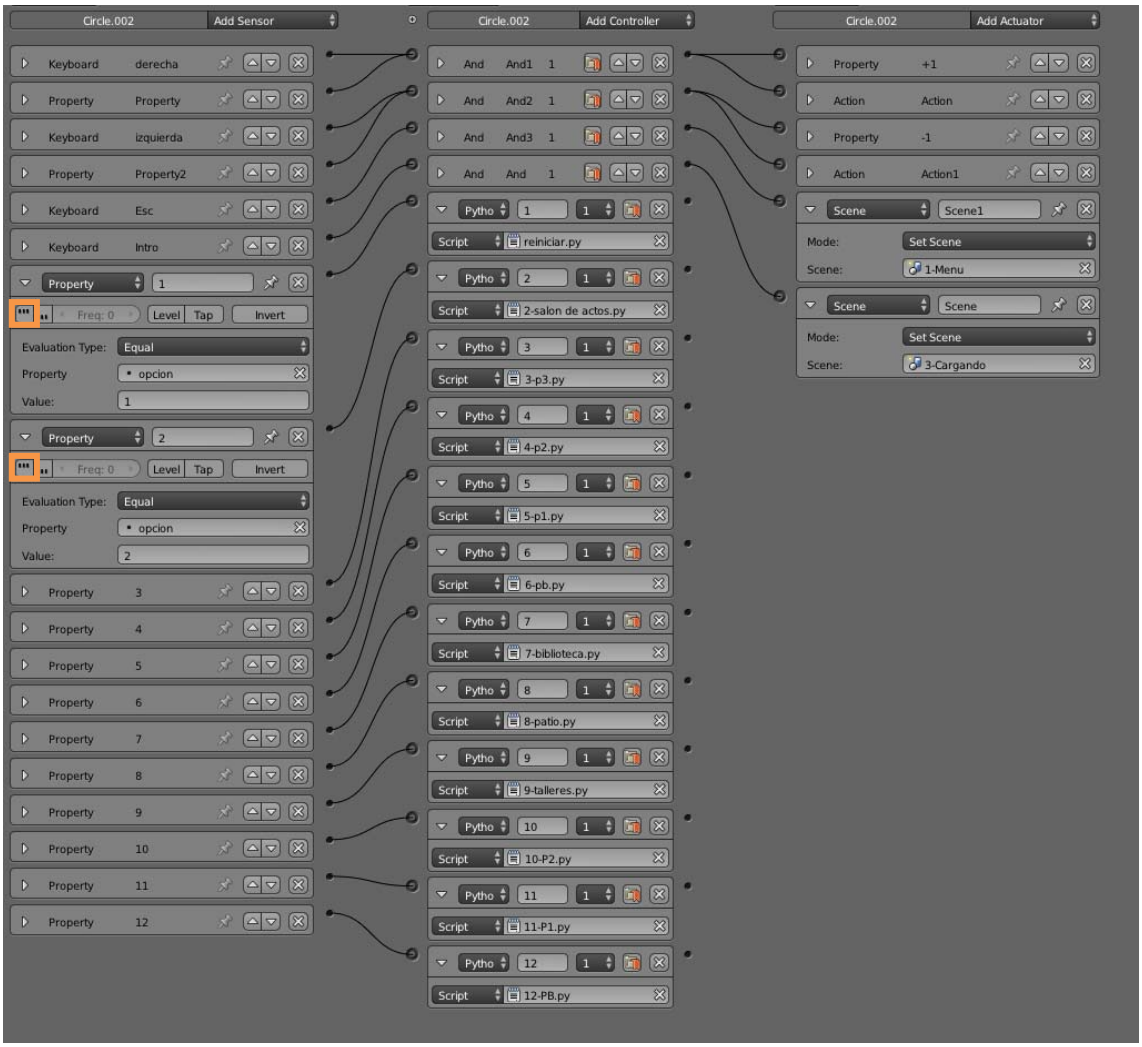


Ilustración 266 – Configuración final de la pantalla mapa

4.12.6 Pantalla de juego.

En el juego, ya tenemos configurado la cámara y el personaje. Ahora tenemos que añadirle las opciones correspondientes para interactuar con la interfaz.

La pantalla juego, lo primero que debe hacer, es **leer** la información que hemos ido guardando en el fichero **"posición.txt"** y **cargarla** en las propiedades de nuestro **personaje**. Por lo que tendremos que añadir un **sensor "always"** y un controlador "python" con un script de carga.

```
##Importamos la lógica de BGE
import bge
GL = bge.logic    ##resumimos el comando
Scene = GL.getCurrentScene()
Control = GL.getCurrentController()
objList = Scene.objects  ##obtenemos la lista de objetos de la escena
## y seleccionamos el personaje, tendremos que cambiar el nombre del objeto para usarlo en otro programa.
cube= objList['Personaje']
##Abrimos el archivo lectura
loadFile = open("posicion.txt", "r")
#Check cabecera
header = loadFile.readline()
#borramos el /n
header = header[0:-1]
#recuperamos datos del fichero
if header == "Este es un archivo valido":
   x= float(loadFile.readline()[0:-1])
   y= float(loadFile.readline()[0:-1])
   z= float(loadFile.readline()[0:-1])
   rot1= float(loadFile.readline()[0:-1])
   rot2= float(loadFile.readline()[0:-1])
```

```
    rot3= float(loadFile.readline()[0:-1])
    rot4= float(loadFile.readline()[0:-1])
    rot5= float(loadFile.readline()[0:-1])
    rot6= float(loadFile.readline()[0:-1])
    rot7= float(loadFile.readline()[0:-1])
    rot8= float(loadFile.readline()[0:-1])
    rot9= float(loadFile.readline()[0:-1])
Posicion=[0,0,0]
PosicionXY=[0,0]
OrigenXY=[7.53053,-35.67547] #punto de inserción por defecto
Rotacion1=[0,0,0]
Rotacion2=[0,0,0]
Rotacion3=[0,0,0]
Posicion[0]=x    #reconstruimos el vector de posición
PosicionXY[0]=x
Posicion[1]=y
PosicionXY[1]=y
Posicion[2]=z
Rotacion1[0]=rot1#reconstruimos la matriz de rotación
Rotacion1[1]=rot2
Rotacion1[2]=rot3
Rotacion2[0]=rot4
Rotacion2[1]=rot5
Rotacion2[2]=rot6
Rotacion3[0]=rot7
Rotacion3[1]=rot8
Rotacion3[2]=rot9
```

```
if PosicionXY == OrigenXY:   #comprobamos si hay datos para cargar
   print("nada que cargar")
else:
     #asignamos datos al personaje
   cube.worldPosition=(Posicion)  #vector de posición
   cube.worldOrientation[0]=(Rotacion1) #matriz de rotación
   cube.worldOrientation[1]=(Rotacion2)
   cube.worldOrientation[2]=(Rotacion3)
   print(cube.worldOrientation)
## Cerramor el fichero
loadFile.close()
```

Una vez cargada la posición de nuestro personaje, comienza el juego. El usuario podrá pulsar la tecla **"Esc" para volver al menú**. Por lo tanto tenemos que configurar dos acciones: El **cambio de escena** y **salvar** los nuevos datos de **posición actual**.

Para el **cambio de escena**, utilizaremos como siempre un **sensor "keyboard"** con la tecla "Esc", unido a un **controlador "And"** y con un **actuador "scene"** que nos redirige al menú del juego.

Para **salvar la posición**, tendremos que unir el **sensor "keyboard"** con un controlador "Python" que contendrá el siguiente script para guardar la posición:

```
##Importamos la lógica de BGE

import bge

GL = bge.logic  ##resumimos el comando

Scene = GL.getCurrentScene()

Control = GL.getCurrentController()

objList = Scene.objects ##obtenemos la lista de objetos de la escena

cube= objList['Personaje'] ##seleccionamos nuestro personaje


cubePosi = cube.worldPosition ##obtenemos vector de posición

print(cubePosi)

cubeRot = cube.worldOrientation ##obtenemos matriz de rotación

print(cubeRot)

##descomponemos la matriz de rotación en vectores

cubeRot0=cubeRot[0]

cubeRot1=cubeRot[1]

cubeRot2=cubeRot[2]

##Abrimos el archivo posición.txt

saveFile = open("posicion.txt", "w")

##Escribimos cabecera

saveFile.write("Este es un archivo valido" + "\n")
```

```
##escribimos los datos

saveFile.write(str(cubePosi[0]) + "\n")

saveFile.write(str(cubePosi[1]) + "\n")

saveFile.write(str(cubePosi[2]) + "\n")

saveFile.write(str(cubeRot0[0]) + "\n")

saveFile.write(str(cubeRot0[1]) + "\n")

saveFile.write(str(cubeRot0[2]) + "\n")

saveFile.write(str(cubeRot1[0]) + "\n")

saveFile.write(str(cubeRot1[1]) + "\n")

saveFile.write(str(cubeRot1[2]) + "\n")

saveFile.write(str(cubeRot2[0]) + "\n")

saveFile.write(str(cubeRot2[1]) + "\n")

saveFile.write(str(cubeRot2[2]) + "\n")

##Cerramos el archivo

saveFile.close()
```

Con estos dos scripts, el usuario puede ir al menú de juego y volver al juego, sin perder la ubicación actual de su personaje.

4.12.7 Añadir música de fondo.

Para añadir una pista de audio de fondo para cualquiera de las escenas, tendremos que hacerlo mediante un **sensor "always"**. Este irá unido a un **sensor "And"** y a un **actuador "Sound"**, donde cargaremos la pista de audio correspondiente. Si queremos que la pista se repita cuando llegue al final, tendremos que seleccionar como modo de reproducción **"Loop End"**.

En el juego hemos utilizado una pista para el menú y otra pista para el juego, conseguidas a través de la página web www.musicalibre.es.

Pista Menú: Fin del reino; Autor: S.Salazar; Álbum: SALROM

Pista Juego: Dark Marimba; Autor: Javier Kimenez; Álbum: Mis bandas sonoras

4.13 FORMATO DE SALIDA

Ya tenemos la visita virtual y la interfaz completamente programada. Por lo tanto es hora de crear un ejecutable del juego que no necesite de Blender para funcionar.

En la pestaña **Render** de la ventana propiedades, podemos **configurar la resolución** de salida, el antialaising y otras opciones de visualización. También cambiaremos la tecla de salida del juego, que por defecto es "Esc", por la tecla "F1", ya que "Esc" lo hemos utilizado para acceder a los menús.

Una vez configurados los ajustes de salida, tendremos que ir a **preferencias de usuario** "Crt + Alt + U" y en la pestaña **Addons**, dirigirnos a **Game Engine** y habilitar "**Save As Game Engine Runtime**". De esta forma, desde el menú exportar podremos guardar el juego como un ejecutable independiente. Ahora nos iremos a la **primera escena** de nuestro juego, seleccionamos la **vista de cámara** mediante la tecla "0" del teclado numérico y **exportamos** el juego. Sacaremos dos versiones del mismo, una para sistemas operativos de 32 bits y otro para 64 bits.

Export → Save As Game Engine Runtime

4.14 OPTIMIZACIÓN

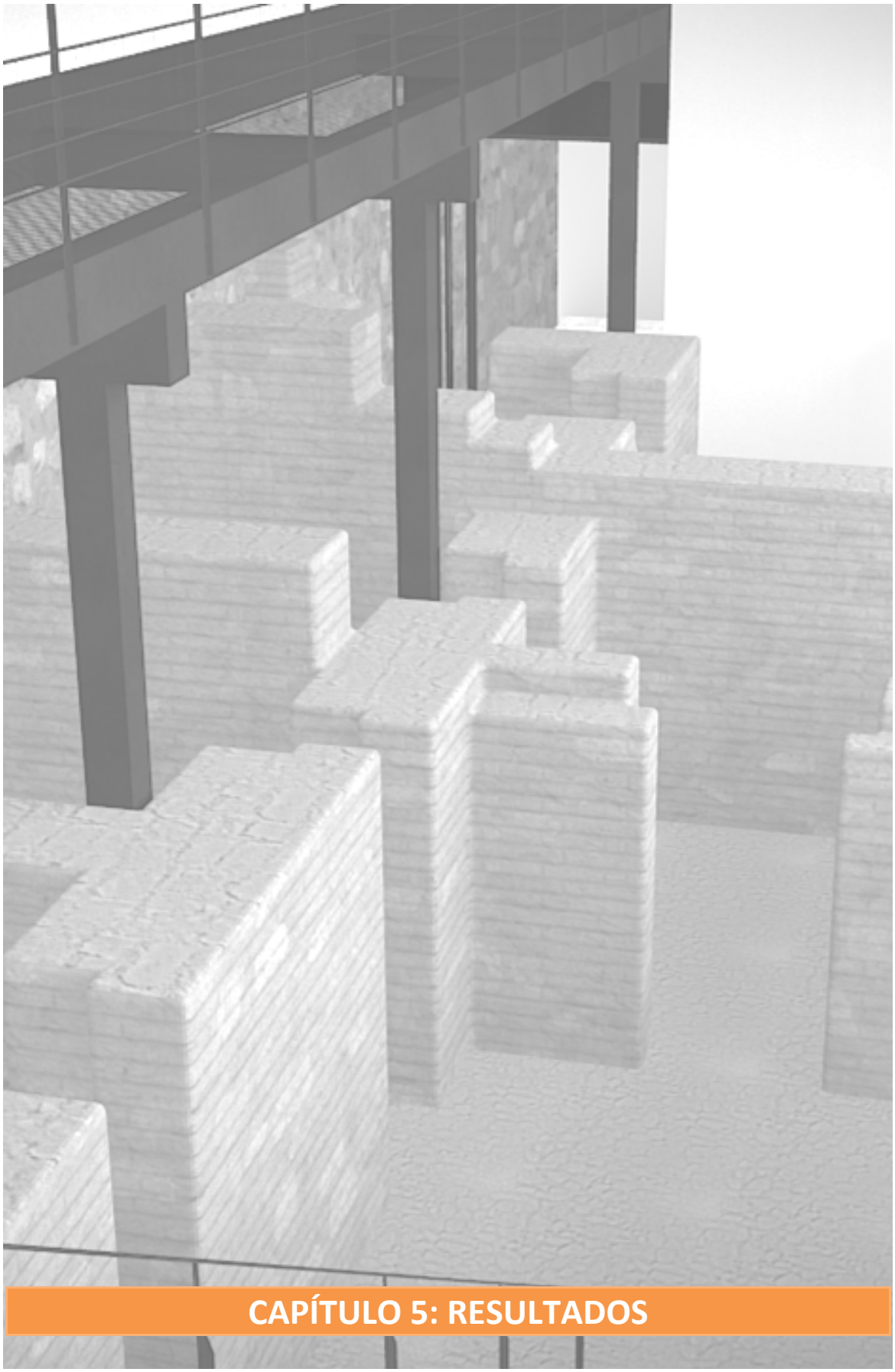
Debido al gran tamaño del modelo tridimensional y la gran cantidad de texturas empleados, nuestra sistema virtual, para funcionar con fluidez, consume una gran cantidad de recursos de hardware. Por lo que tendremos que intentar optimizarlo.

Los principales problemas, viene por el punto de partida del proyecto, ya que hemos intentado utilizar el mismo modelo tridimensional para realizar el video que para realizar el sistema virtual.

Cuando estamos desarrollando un sistema virtual de este tipo, tenemos que intentar que los objetos tengan el **menor número de caras posibles**. Por lo tanto, he intentado reducir dentro de lo posible el número total de caras del modelo. Simplificando mobiliario y restringiendo el número de aulas a las que tenemos acceso. También hemos simplificado los modelos de los árboles y hemos suprimido los coches del aparcamiento, ya que estos dos objetos tenían un elevadísimo número de caras.

Otro problema, proviene de los materiales, ya que he usando **texturas** de bibliotecas y tomadas con la cámara, y estas tienen una resolución muy elevada entorno a 2500x1900 píxeles. En un videojuego, se suelen utilizar texturas muchísimo más pequeñas, en torno a **256x256 píxeles.** Por tanto, para el archivo de Blender del sistema virtual, tuve que ir reduciendo cada una de las texturas a una resolución mucho más baja. Tras hacer esto, notamos una mejora muy importante en la estabilidad del juego.

Por último, en nuestro modelo teníamos muchas **texturas de relieve** "bump". Estas texturas cargan mucho el renderizado en tiempo real, por lo que desactivamos aquellas que no eran estrictamente necesarias. En otros casos, fueron **sustituidas por mapas de normales**, que son más ligeras en este tipo de procesos.



CAPÍTULO 5: RESULTADOS

CAPÍTULO 5: RESULTADOS

Una vez **terminadas todas las fases** de nuestro proyecto, es hora de recapitular y comprobar que hemos cumplido los objetivos que nos planteábamos al comienzo de este trabajo.

5.1 MODELO TRIDIMENSIONAL

El **primer objetivo**, era realizar un **modelo tridimensional** de la Escuela, que nos permitiese realizar el video de presentación y el sistema interactivo. Tras la fase de modelado, podemos ver que el modelo tridimensional creado, cumple con las expectativas, ya que refleja de forma exacta tanto la geometría de los exteriores e interiores, como la geometría del mobiliario y demás elementos que componen las escenas. Como dato, decir que el modelo tridimensional una vez terminado, está compuesto por **1984 objetos**. En total contienen **3,6 millones de vértices** y **3,2 millones de caras** y **6,2 millones de aristas**.

Una vez realizado el modelo tridimensional, le hemos añadido la **iluminación** y los **materiales**. Gracias al uso de texturas reales fotografiadas, las imágenes finales presentan un aspecto bastante realista y se aproximan mucho a la realidad material de las diferentes escenas. Por otro lado, hemos conseguido una iluminación muy real, teniendo en cuenta las limitaciones que presenta el motor interno de Blender.

Estos dos aspectos, son fundamentales a la hora de conseguir un **resultado realista**, ya que, unos materiales irreales y una mala iluminación, pueden echar a perder todo el esfuerzo invertido en la obtención de la geometría. Para el modelo de la escuela, se han creado y utilizado un total de **368 materiales**, donde hemos utilizado más de **260 texturas**.

A continuación se muestra algunas imágenes del modelo.

Ilustración 267 – Fachada principal Edificio Störr

Ilustración 268 – Vista aérea

Ilustración 269 – Interior clase. Edificio Störr

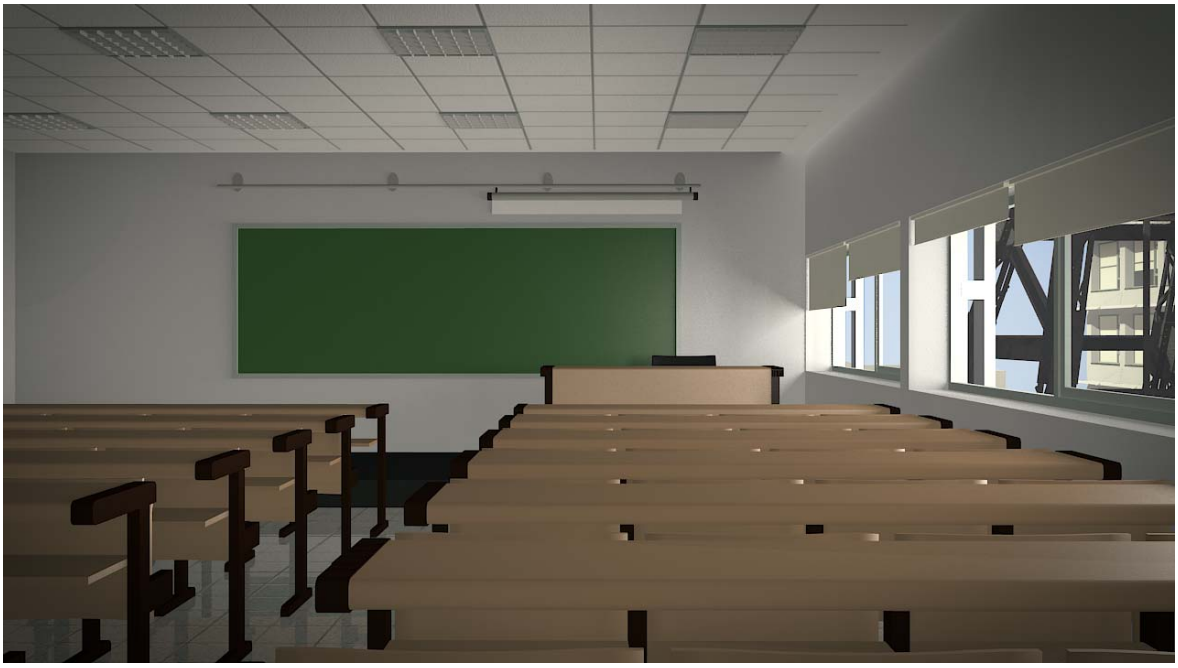


Ilustración 270 – Interior clase. Edificio E`lhuyar

Ilustración 271 – Interior Edificio E`lhuyar

Ilustración 272 – Calabozos Real Cárceles de forzados. Edificio E`lhuyar

Ilustración 273 – Interior Biblioteca. Edificio Störr.

Ilustración 274 – Interior Sala de Juntas. Edificio Störr

Ilustración 275 – Interior Laboratorio de Química. Edificio Störr

Ilustración 276 – Interior Laboratorio de elctricidad. Edificio Störr

Ilustración 277 – Interior Laboratorio de elctricidad. Edificio Andrés Manuel del Rio.

Ilustración 278 – Interior Hall. Edificio Störr

5.2 VISITA VIRTUAL. VIDEO PRESENTACIÓN.

Una vez completado modelo tridimensional, el **segundo objetivo** era utilizar dicho modelo para generar una animación para dar a conocer las instalaciones y promocionar la Escuela mediante un **video de presentación**.

Se trataba de generar un producto audiovisual con un alto contenido de información. Por lo tanto se realizó un guión sobre el contenido y cómo iba a contarse dicho contenido. Teníamos que aprovechar las posibilidades que ofrece trabajar con un modelo tridimensional, ya que estas nos permiten modificar la realidad de un edificio y mostrarlo de formas que serían imposibles mediante un vídeo convencional.

Ilustración 279 – Vista del Edificio Störr por plantas.

Tras el proceso de animación y postproducción, descrito en el capítulo anterior, se ha obtenido un **video de 8 min**, donde se recorren desde distintos puntos de vistas los edificios y las principales instalaciones del centro. El video posee una calidad bastante buena de imagen y acorde con la información que se está contando en todo momento. Por lo que creo que cumple con las expectativas de un producto de estas características.

Para la realización del video, se han utilizado un total de **11520 fotogramas**. A continuación mostramos algunas de las imágenes donde podemos ver varios de los recursos empleados.

Ilustración 280 – Vista del Edificio E'lhuyar.

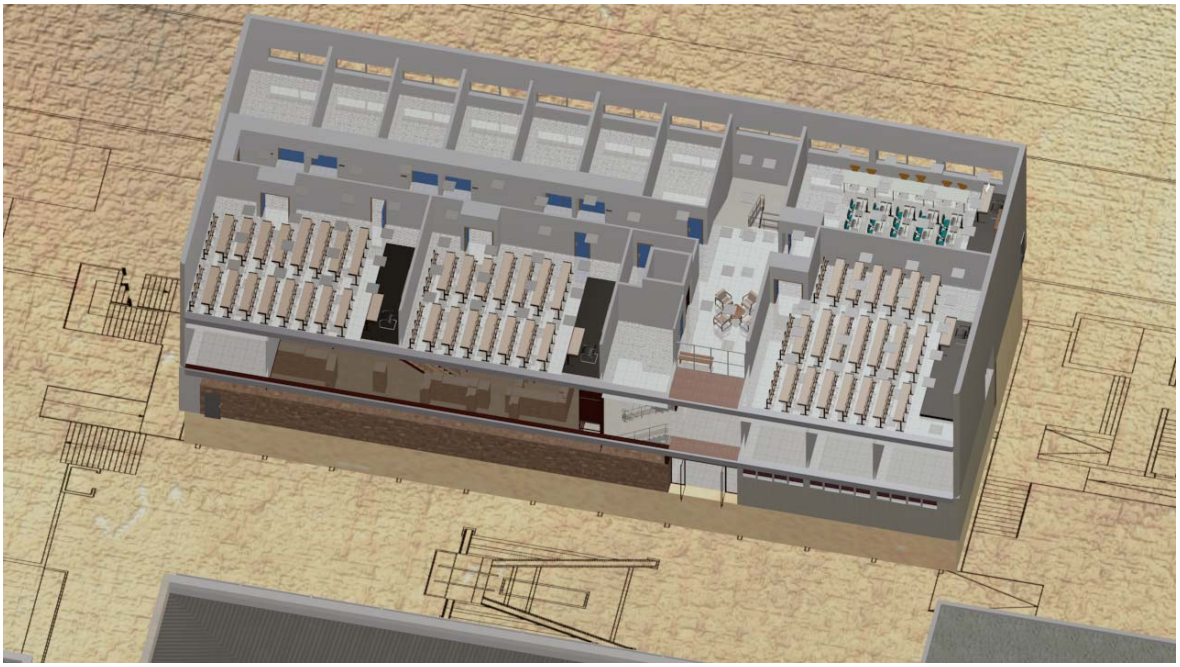


Ilustración 281 – Vista del Edificio E'lhuyar por plantas.

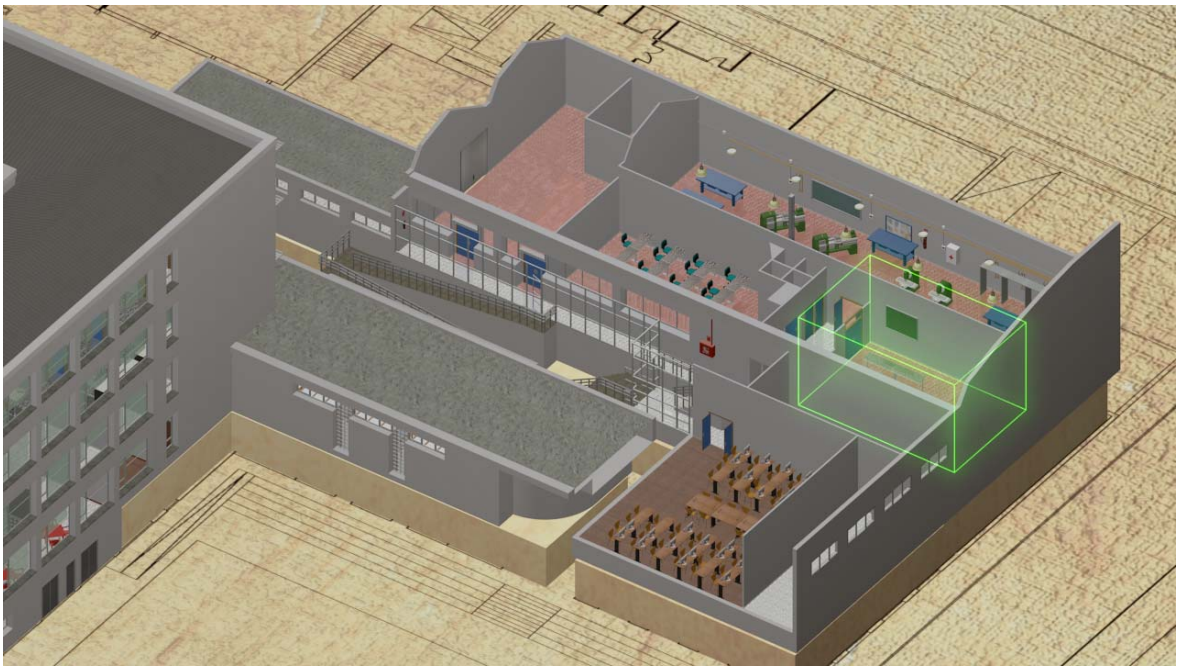


Ilustración 282 – Vista del edificio Andrés Manuel del Río por plantas.

Ilustración 283 – Vista interior del Aula de informática.

5.3 VISITA VIRTUAL. SISTEMA INETRACTIVO.

Por último, como **tercer objeto**, se propuso la realización de una visita virtual mediante el uso de un **sistema interactivo** como elemento de promoción de la Escuela. Este objetivo, desde mi punto de vista, era el más interesante como elemento de promoción, ya que el usuario deja de ser un espectador y participa activamente de la visita.

Tras el proceso de **optimización y programación**, se ha logrado generar una visita virtual donde el usuario es completamente libre en sus acciones. Y puede recorrer las instalaciones tanto exterior como interiormente. Además se ha acompañado el sistema interactivo con una interfaz acorde a las necesidades de un programa de estas características.

El **producto final**, es un sistema de promoción bastante completo y atractivo, donde cualquier usuario podrá conocer las instalaciones del centro de una forma amena y divertida. Además gracias a la calidad del modelo, y a las técnicas de iluminación y texturizas empleadas, se ha conseguido un producto con una calidad gráfica excelente.

A continuación, se muestran algunas imágenes capturadas del sistema interactivo.

Ilustración 284 – Fachada principal edificio Störr./ Sistema Interactivo

Ilustración 285 – Fachada a patio edificio Störr./sistema interactivo

Ilustración 286 – Cabria patio./sistema interactivo

Ilustración 287 – Fachada a patio edificio Mateo Alemán./sistema interactivo

Ilustración 288 – Fachada a patio edificio E'lhuyar./sistema interactivo

Ilustración 289 – Interior Clase./sistema interactivo

Ilustración 290 – Interior Cafeteria./sistema interactivo

5.4 DIAGRAMA DE GANTT.

En este diagrama vamos a ver el tiempo dedicado a cada una de las **fases del proyecto**.

		MESES											
Nº	FASES	1	2	3	4	5	6	7	8	9	10	11	12
1	Curva de aprendizaje	■	■	■									
2	Análisis información			■									
3	Modelado				■	■	■	■	■				
4	Texturizado					■	■	■	■				
5	Iluminación Exterior					■			■				
6	Iluminación interior								■				
7	Animación									■	■		
8	Render										■		
9	Postproducción										■	■	
10	Programación										■	■	
11	Memoria								■	■	■	■	■

Podemos diferenciar tres bloques, un primer bloque de **aprendizaje del programa y análisis de la información**. Un segundo bloque correspondiente al **modelado, texturizado e iluminación** del modelo. Y un tercer bloque correspondiente a la fase de **animación, renderizado y programación.**

En el segundo bloque, podemos ver que es en el que más tiempo se ha invertido. En primer lugar por el tamaño del proyecto, y en segundo porque de él van a depender los resultados finales, por lo tanto merece la pena invertir el tiempo necesario.

En el último bloque, la fase de renderizado y postproducción, nos genera una cantidad enorme de tiempos muertos debido a los procesos de cálculo del computador. Por lo tanto nos permiten, en cierta manera llevar varias fases en paralelo.

Por último, comentar que gracias a poder usar el servicio de Supercomputación, la fase de render, se ha visto reducida a 3 - 4 semanas, mientras que un ordenador personal, estaría alrededor de dos a tres meses mínimo.

5.5 ESTADÍSTICAS

Por último, acompañar los resultados de unas estadísticas con la información generada.

Modelo:

OBJETOS	VÉRTICES	ARISTAS	CARAS
1.984	3.682.733	6.224.881	3.217.594

Texturizado:

MATERIALES	TEXTURAS
368	260

Video:

TIEMPO DE VIDEO	FRAMES	TAMAÑO IMAGENES
480 segundos	11.520	13,2 GB

Tiempo de Render en SSC:

FRAMES	TIEMPO TOTAL*	MEDIA POR FRAME	TIEMPO EN PARALELO*	MEDIA POR FRAME
1.984	1.442,56 h	43,6 min / frame	490,35 h	14,8 min /frame
	60,08 días		20,43 días	

*No incluye tiempo de pruebas, ni escenas repetidas.

PC utilizado para el modelado:

CPU: Intel Quad Core Q84000 2,66 GHz
RAM: 8 GB DDR3 a 1333 MHz
GPU: Nvidia Geforce GTX 650 - 1 GB GDDR5
SO: Windows 7 64 bits

CAPÍTULO 6: CONCLUSIONES

CAPÍTULO 6: CONCLUSIONES

Este proyecto, se planteaba como medio de aprendizaje de las técnicas y herramientas de modelado 3D. El modelado 3D, es una disciplina que cada día posee más recursos, y que poco a poco ha ampliado su campo de aplicación a multitud de disciplinas.

En este caso, se planteaba su uso como herramienta para la promoción de la escuela, mediante la generación de un video de presentación y una visita virtual interactiva.

Cuando comenzamos el proyecto, se presentaba como un gran reto. Primero, debido a lo complejo del modelo tridimensional, ya que tenía que modelar la totalidad del complejo de la Escuela, y sus espacios interiores. Y segundo, debido a que el trabajo se iba a desarrollar mediante la herramienta Blender, un software completamente desconocido para mí.

Así que, ese fue el punto de partida, antes de comenzar con el proyecto, aprender a utilizar Blender. Para ello, decidí realizar una prueba con un modelo pequeño utilizando los planos del pabellón de Barcelona de Mies Van Der Rohe. Esta toma de contacto con Blender, me permitió habituarme a la interfaz y comprender cómo se aplicaban en Blender los principales conceptos sobre renderización.

Ilustración 291 – Render realizado con Blender. Pabellón de Barcelona

Gracias a los tutoriales y los foros de la comunidad Blender, conseguí rápidamente la mayor parte de la información y las herramientas que necesitaba para comenzar con el proyecto. Así que tras completar el proceso de aprendizaje y familiarizarme con el software, empecé de lleno con el proyecto de la Escuela.

Así que comencé analizando la planimetría y las fotografías, para poder generar las plantillas en dos dimensiones, que nos servirían de base para el modelo 3D. Como comentaba en capítulos anteriores, esta fase tampoco fue fácil, pues toda la planimetría del centro, estaba por separado y supuso prácticamente redibujarlo todo.

El proceso de modelado, texturizado e iluminación de la escuela, fue un proceso muy largo, en parte por las dimensiones del proyecto y en parte por mi inexperiencia con el programa. Pero todo el tiempo invertido, se vio recompensado cuando comenzaron a renderizarse las primeras imágenes.

En general, creo que la calidad de las imágenes obtenidas, es bastante buena. Sobre todo teniendo en cuenta que hemos usado el motor interno de Blender, que como comentábamos en los capítulos iniciales tiene muchas limitaciones en cuanto al cálculo de luz y materiales.

Así que con el modelo terminado, llagamos a las fases finales, la realización del video de la Escuela y la programación de una visita virtual mediante un sistema interactivo. Personalmente, estaba muy interesado en estas dos fases, pues nunca había realizado un proceso de animación virtual, ni de programación de un videojuego.

La fase de animación y realización del video, fue casi como hacer un pequeño corto. Tuve que hacer un guión y un pequeño storyboard, buscando la mejor forma de contar las instalaciones. Y después llevar estas ideas al programa. Esta fase costó muchas pruebas desde distintos ángulos, distintas velocidades de los objetos, recorridos de cámara, etc.

El resultado final, creo que es bastante interesante, ya que se ha mostrado la escuela desde diferentes puntos de vista y utilizando varios de los recursos que nos permite la animación virtual, como desmontar edificios, marcar partes, vistas aéreas, etc.

Para la programación del juego, aunque Blender nos facilita mucho la tarea programar mediante el editor de lógica, fue necesario hacer una gran investigación sobre lo que se podía y no se podía hacer, y lo más importante cómo.

Para ello, se realizó un guión con las características que necesitábamos tanto para el juego como para la interfaz, y poco a poco, mediante el editor de lógica y scripts realizados en Python, fui cumpliendo cada uno de ellos. Finalmente creo que el sistema programado, se ajusta bastante bien a las necesidades de un programa con estas características, y la calidad gráfica final, ha sido una de las cosas que más me ha sorprendido del motor de juego de Blender. En mi opinión, el sistema virtual creado, es una herramienta muy potente de promoción, ya que siempre es más atractivo para el usuario participar activamente en una visita virtual que ser un simple espectador al que le mandamos una información.

En cuanto a Blender, me ha parecido una herramienta muy potente y muy interesante por la cantidad de módulos que trae incorporados. Ya que excepto para generar los planos y las fotografías, el resto del proyecto se ha realizado íntegramente con él. Aunque en algunos aspectos, es más limitado que otros softwares de pago, la gran cantidad de recurso que posee y la gestión que realiza del hardware del computador, hace de Blender una herramienta a tener en cuenta en el mundo del modelado 3D.

Personalmente, he disfrutado mucho realizando este proyecto. Ya que en este tipo de trabajos se mezclan una gran cantidad de disciplinas para poder llevarlos a cabo, dibujo, ingeniería, arquitectura, animación, fotografía, programación, etc., y además como veíamos al principio, es una utilidad con aplicaciones en muchas otras disciplinas como cine, publicidad, ingeniería, arquitectura, medicina, fabricación, etc. Por lo tanto creo que ha sido un proyecto bastante interesante, ya que con los conocimientos adquiridos, podemos ser capaces de representar y animar cualquier objeto existente o diseñado por nosotros.

En cuanto al producto final estoy bastante satisfecho con la calidad obtenida, y creo que se han logrado todos los objetivos propuestos inicialmente. Además el modelo tridimensional que hemos generado, puede ser utilizado en un futuro, para otras actividades promocionales de la escuela.

También, se pueden proponer diferentes trabajos utilizando como base este modelo. Por un lado, se pueden proponer proyectos de ampliación, añadiendo los departamentos e instalaciones que no se han modelado interiormente. Y por otro lado, se pueden proponer nuevos sistemas interactivos o mejorar el actual, de forma que se pueda interactuar con otros usuarios y crear una especie de red social virtual.

ANEXOS

Anexo 1: Manual de usuario

Contenido del DVD:

1- Video presentación EIMIA.
2- Sistema Interactivo Visita Virtual EIMIA.
- Versión SO 32 bits "\VVEIMIAx32".
- Versión SO 64 bits "\VVEIMIAx64".

Instalación:

1- Introduzca DVD en la unidad DVD-ROM.
2- Copie la carpeta correspondiente a su Sistema Operativo a su disco duro.
3- Para ejecutar el programa abra la carpeta y ejecute al archivo:
- SO 32 bits "\VVEIMIAx32\VV EIMIA x32.exe"
- SO 64 bits "\VVEIMIAx64\VV EIMIA x64.exe"

Desinstalación:

- Para desinstalar el programa, simplemente elimine la carpeta

Guía rápida:

- **Interfaz:**

  Durante el menú de inicio, utilice las teclas de dirección "← →" para desplazarse entre las diferentes opciones. Utilice la tecla "Intro" para aceptar la selección.

- **Juego:**

  Andar: Utilice las teclas de dirección, o las teclas W, A, S y D, para desplazarse.

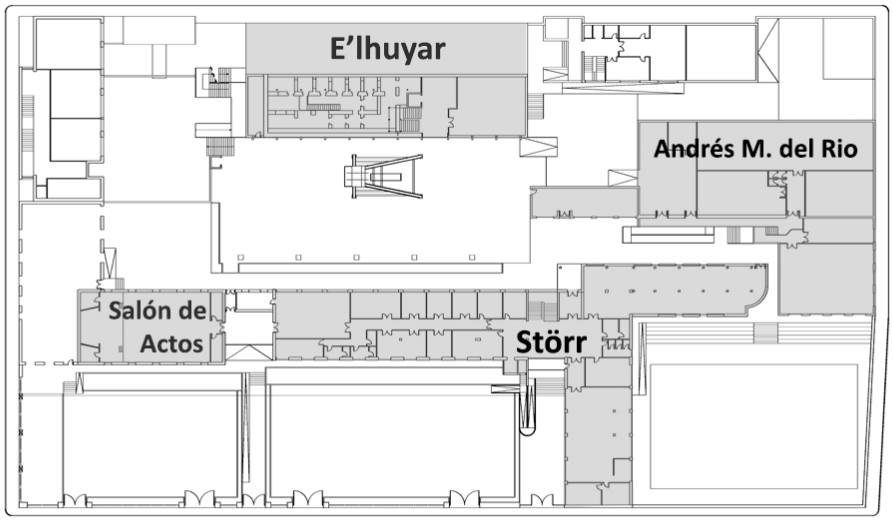
  Mirar: Para girar la cámara, mueva el ratón en la dirección deseada.

  Volver a Menú: Para volver al menú utilice la tecla "ESC".

  Salir: Para salir del juego utilice la tecla "F1".

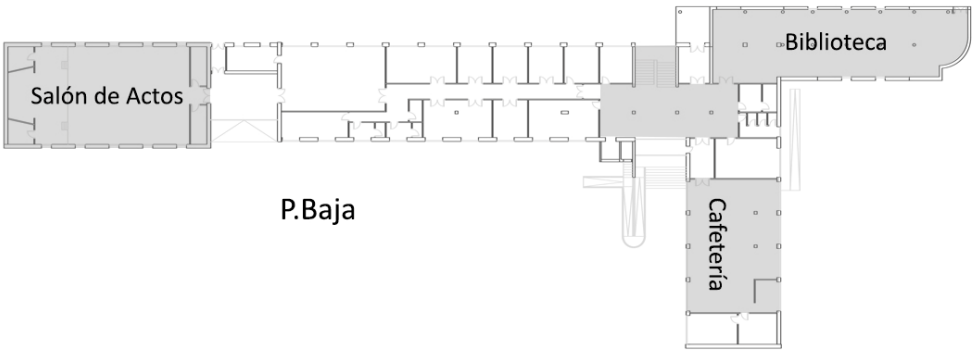
Visita Virtual:

Durante la visita virtual, podrá desplazarse libremente por las instalaciones de la Escuela de Ingeniería Minera e Industrial de Almadén. A continuación incluimos los planos de la Escuela, indicando las instalaciones que podrá visitar y que le servirán de guía durante su visita.

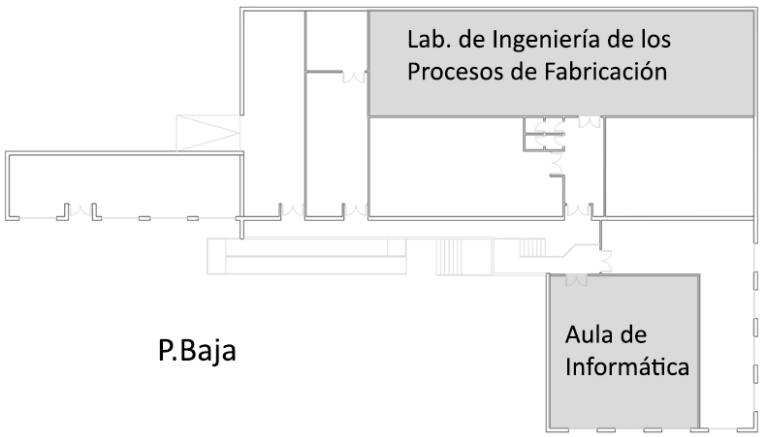


Edificios visitables

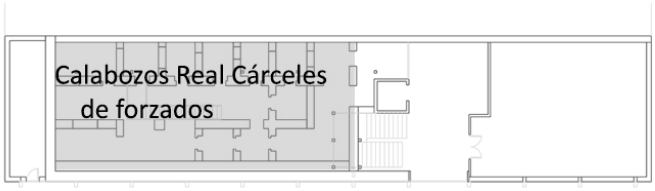
Edificio Störr:



Edificio Manuel del Rio:



Edificio E'lhuyar:



P.Baja

P.Primera

P.Segunda

Requisitos Mínimos:

- CPU Quad Core a 2GHz
- 4 GB de memoria RAM
- GPU 1 GB
- Versión SO 32 bits

Requisitos Recomendados:

- CPU Intel i3 a 3GHz
- 8 GB de memoria RAM
- GPU Nvidia 2 GB
- Versión SO 64 bits

Anexo 2: Modelos 3D

Modelos obtenidos de la página web ***www.blendswap.com***:

Modelo	Enlace	Autor
Chair	http://www.blendswap.com/blends/view/918	Jo_Anna
Paimio Chair	http://www.blendswap.com/blends/view/22987	Hugoferreira
Low-Poly Chair set	http://www.blendswap.com/blends/view/70219	DennisH2010
Lattice Chair	http://www.blendswap.com/blends/view/23472	Rosenth
Tree Next Gen	http://www.blendswap.com/blends/view/70949	Zuendholz
Low Poly foliage	http://www.blendswap.com/blends/view/59269	EugeneKiver
Arbusto	http://www.blendswap.com/blends/view/66876	Jese-Mx
Samsung TV 55	http://www.blendswap.com/blends/view/72709	DennisH2010
PC	http://www.blendswap.com/blends/view/740	Modelyna
Bottles 1.0	http://www.blendswap.com/blends/view/14882	Elbrujodelatribu
Glass	http://www.blendswap.com/blends/view/71654	BananaBoy
Fire extinguiser	http://www.blendswap.com/blends/view/63723	DennisH2010
Pool table	http://www.blendswap.com/blends/view/67131	Jeffie
Car	http://www.blendswap.com/blends/view/49721	Johntheboom
Coffe maker	http://www.blendswap.com/blends/view/4520	Broghtonpiers
Microwave	http://www.blendswap.com/blends/view/63691	Rich33584
Drink Bar	http://www.blendswap.com/blends/view/71639	B2przemo

Modelos obtenidos de la página web ***archive3d.net***:

Modelo	Enlace	Autor
Lathe 3D model	http://archive3d.net/?a=download&id=a86866ec	Wang
Drill	http://archive3d.net/?a=download&id=c9dc072a	May Ernst
Lathe 3d Model	http://archive3d.net/?a=download&id=3215459c	Carlo Giancarlo

Anexo 3: BIBLIOGRAFÍA

Apuntes:

[Morcillo, 2011/12] Morcillo, C. G. (2011/12). Computer Animation and Rendering. Apuntes Escuela Superior de Informática, Universidad de Castilla la Mancha.

[Penagos, 2012] Penagos, J. R. (2012). ¿Cómo crear un videojuego con BGE? Apuntes Informática y Sistemas, Universidad del Guindío.

Libros:

[Chanes, 2011] Chanes, M. (2011). *Autodesk 3ds Max 2011.* Anaya Multimedia.

[Flavell, 2010] Flavell, L. (2010). *Beginning Blender.* Apress.

[Hess, 2011] Hess, R. (2011). *Blender Diseño y Creatividad.* Anaya Multimedia.

[Möler, 2008] Akenine Möler, T., Haines, E., & Hoffman, N. (2008). *Real-Time Rendering.* A.K.Peters.

[Moya, 2010] Moya, R. (2010). *Breve guía para modelar arquitectura en Blender 2.55*.

[Mullen, 2010] Mullen, T. (2010). *Mastering Blender.* Sybex.

[Mullen, 2012] Mullen, T. (2012). *Mastering Blender 2nd edition.* Sybex.

[Roosendaal, 2004] Roosendaal, T., & Selleri, S. (2004). *Blender 2.3 Guía.* No Starch Press.

[Shirley, 2009] Shirley, P., & Marschner, S. (2009). *Fundamentals of Computer Graphics.* A.K.Peters.

[Simond, 2012] Simonds, B. (2012). *Blender Master Class.* Sybex.

[Suffern, 2007] Suffern, K. (2007). *Ray Tracing from the Group Up.* A.K.Peters.

[Theoharis, 2007] Theoharis, T., Papaioannou, G., Platis, N., & Patrilkalakis, N. (2007). *Graphics and Visualization: Principles & Algorithms.* A.K.Peters.

TFG:

[Alloza, 2010/11] Alloza, M. (2010/11). Blender VS Autodesk 3ds Max. TFG Ingeniería Multimedia.

Artículos:

[Armoza, 2013] Armoza, M. (2013). Impresora 3D fabrica un motor a reacción a escala. *Replikat* .

[Becerro, 2010] Becerro, A. (2010). Curso de creción de videojuegos mediante blender. http://es.gnu.org/~littledog/3d/blendergame1.pdf

[Caicoya, 1999] Caicoya, C. (1999). Museo Guggenheim Bilbao. *Revista de obras públicas* .

[Disseny2d1, 2010] Disseny2d1.(2010). *Evolución de las aplicaciones gráficas por ordenador.* Obtenido de: http://www.slideshare.net/disseny2d1/004-evolucin-aplicacionesgrficos-ordenador

[Gallardo, 2010] Gallardo, E. (2010). Recorridos virtuales BGE. http://coolrocker911.files.wordpress.com/2010/07/recorridos-virtuales-y-game-engine-blender.pdf

[García, 2009] García, G. (2009). *Herramientas de modelado 3D.* Obtenido de http://www.slideshare.net/gbgarcia/herramientas-de-modelado-3d

[Goás, 2009] Goás, J. H. (Noviembre de 2009). Guía de iniciación a Blender 2.5

[Hernán, 2009] Hernán, H. (2009). Desafíos y tendencias en el diseño de videojuegos. *Revista Internacional de Comunicación Audiovisual, Publicidad y Estudios Culturales* .

[Mccaffrey, 2011] Mccaffrey, T. J. (2011). *La historia del software de animación.* Obtenido de http://www.ehowenespanol.com/historia-del-software-animacion-sobre_311944/

[Santos, 2013] Santos, R. G. (2013). Las tecnologías de prototipado rápido en cirugía. *Revista Cubana de Estomatología* .

Web:

[Artist, 2014] *Blender's Artist.* (2014). Obtenido de http://blendersartist.org

[Autodesk, 2014] *Autodesk*. (2014). Obtenido de http://www.autodesk.com

[Blender, 2014] *Blender*. (2014). Obtenido de http://wiki.blender.org

[Chaos Group, 2014] *Chaos Group*. (2014). Obtenido de http://www.chaosgroup.com

[Guru, 2014] *Blende Gurur.* (2014). Obtenido de http://www.blenderguru.com

[Molón, 2014] *Físico Molón.* (2014). Obtenido de http://www.fisicomolon.com

[Vila,2014] Vila, C (2014). *Eterea estudios.* Obtenido de http://www.etereaestudios.com

[Villar, 2014] Villar, O. (2014). *Blendtuts.* Obtenido de http://blendtuts.com/

Visitas virtuales:

Edificio de viviendas Granato	https://www.youtube.com/user/3dproyecta/videos
E.S. de Informática, Ciudad Real	http://www.inf-cr.uclm.es/virtual/
FFIS de la Región de Murcia	http://www.ffis.es/inicio/visitavirtualhospitales.php
Fundación Telefónica	http://www.fundacion.telefonica.com/es/arte_cultura/arsvirtual/
Hospital General, Ciudad Real	https://www.youtube.com/watch?v=qnldtMWhspE https://www.youtube.com/watch?v=pa2taAcaKoM
Machu Picchu, Perú	http://panoramas.pe/machupicchu100.html
Rome Reborn	http://romereborn.frischerconsulting.com/
Universidad de Jaén	http://www.ujaen.es/visita-virtual-ujaen/index.html