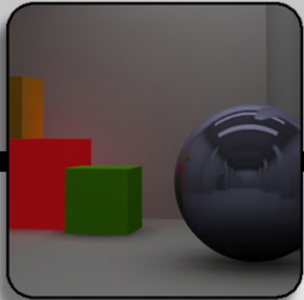


Diseño Gráfico 3D con Software Libre

Una Introducción Práctica a Blender

Carlos González Morcillo - Carlos.Gonzalez@uclm.es

Introducción a Blender (I)
Introducción a Blender (II) + GIMP
Modelado con Wings 3D y Teddy
UV-Mapping con Blender
Animación básica con Blender
Animación jerárquica y Composición
Introducción a los Esqueletos
Renderizado Realista
Modelado por Rotoscopia
Animación No Lineal
Composición de Video
OREj: de Blender a OpenGL

http://www.nicodigital.com

Prefacio :..

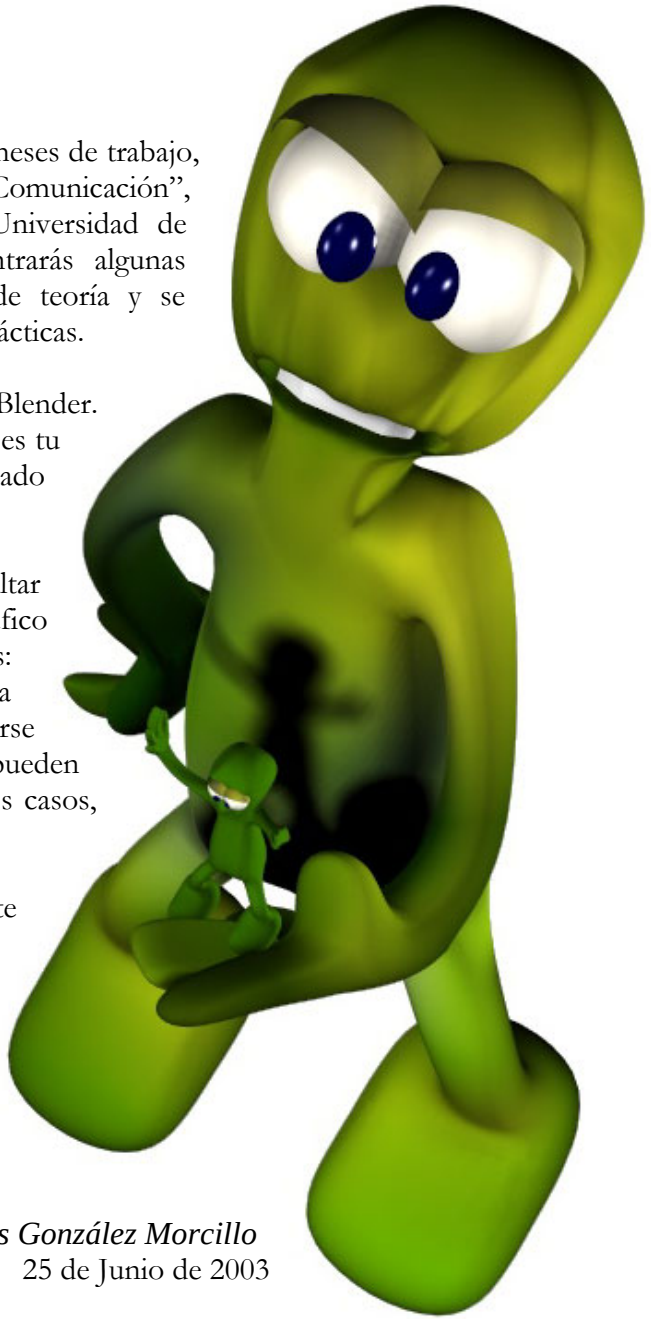
Estos apuntes que tienes en tus manos son el fruto de cuatro meses de trabajo, preparando las prácticas de la asignatura "Animación para la Comunicación", en la Escuela Superior de Informática de Ciudad Real (Universidad de Castilla-La Mancha). Por esta razón, en el texto encontrarás algunas referencias a conceptos que fueron explicados en clases de teoría y se completaron con aclaraciones comentadas en las sesiones de prácticas.

Este documento no pretende ser una guía exhaustiva de Blender. Tampoco es un manual para el totalmente inexperto en 3D. Si es tu caso, un buen punto de comienzo podría ser el manual realizado por Javier Belanche.

De cualquier forma, he considerado que este trabajo podía resultar útil para quien desee introducirse en el mundillo del diseño gráfico 3D, utilizando alguno de los sistemas operativos más populares: Microsoft Windows y Linux, sin necesidad de gastarse una fortuna en software. La mayoría de las sesiones pueden realizarse únicamente con Blender. En otras, harán falta programas que pueden encontrarse de forma gratuita en la red y, en la mayoría de los casos, podrán utilizarse indistintamente en Windows y Linux.

Será bienvenida cualquier cuestión y sugerencia sobre este documento, así como errores encontrados, a la dirección de correo: Carlos.Gonzalez@uclm.es .

Finalmente quiero agradecer a Nines y a mi familia todo el tiempo robado, a mis alumnos la ilusión aportada en clase, y a Caronte su apoyo a la hora de alojar este documento en Nicodigital y ayudar a su expansión por la red de redes.



Carlos González Morcillo
25 de Junio de 2003

Práctica 1
Introducción a Blender (I)

Esta primera práctica nos servirá como toma de contacto inicial con la herramienta. Aprenderemos a personalizar el entorno y renderizar una escena sencilla.

El interfaz que presenta Blender al arrancar es el mostrado en la Figura 1. Todas las ventanas de Blender son personalizables, en el sentido de que podemos dividir la pantalla como nos resulte más cómodo y situar en cada porción el tipo de ventana que queramos. Por defecto Blender arranca la pantalla con dos divisiones; la ventana superior de tipo *3DWindow* y la inferior de tipo *ButtonWindow*. Podemos cambiar el tipo de ventana pinchando en el botón y seleccionando alguno de los tipos que se muestran en la siguiente imagen:

La ventana 3D nos muestra un cuadrado de color rosado. Este cuadrado es un plano que crea Blender al comenzar a trabajar. El color rosa indica que está seleccionado. El triángulo inferior de color negro representa la cámara. El resto de cuadrados de la escena lo forman la cuadrícula que nos servirá para colocar los objetos de la escena con precisión.

El círculo de color rojo situado en el centro de la ventana es el puntero3D. Todos los elementos que se añadan a la escena lo harán donde se encuentre el puntero 3D.

Antes de empezar a trabajar, personalizaremos un poco el entorno de Blender. Dividiremos la ventana 3D con la distribución que se muestra en la Figura 2. Para ello, nos situaremos con el ratón en la zona de división de dos ventanas (por ejemplo, entre la zona de división entre la ventana 3D y la ventana de Botones) y con la ventana 3D iluminada (seleccionada), haremos **BDR** y elegiremos *Split Area*. Partiremos la pantalla hasta conseguir una disposición del área de trabajo como se muestra en la Figura 2. Podemos eliminar las cabeceras, así como situarlas en la parte superior de las ventanas haciendo **BDR** sobre ellas.

Para rotar el punto de vista de una ventana 3D, con **ALT** pulsada, arrastraremos el ratón con **BIR**. Podemos hacer zoom tanto en la ventana 3D como en la ventana de botones con **CONTROL** + **ALT** + **BIR**. Para desplazar el punto de vista (horizontal y verticalmente), utilizaremos **ALT** + **MAYÚSCULAS** + **BIR**.

Cambiaremos el modo de vista de cada ventana 3D mediante atajos de teclado o los botones destinados a tal efecto. Pulsando **5** en el teclado numérico, cambiaremos entre proyección ortográfica y perspectiva. Con **0**

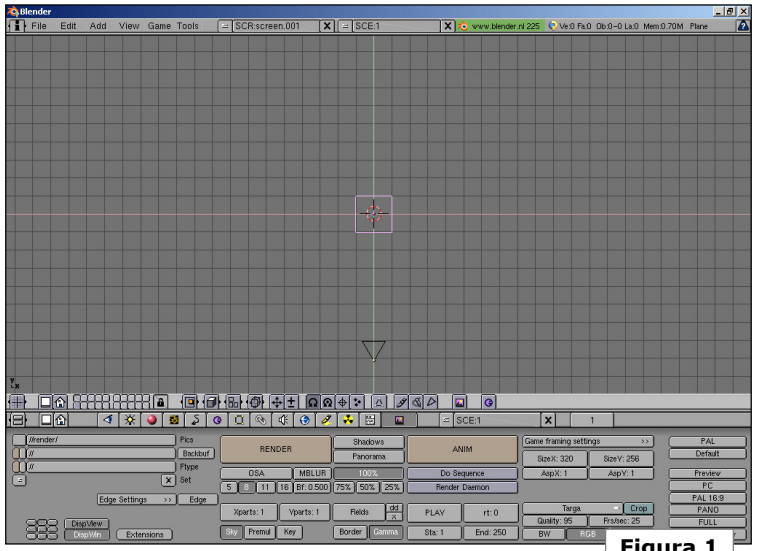


Figura 1

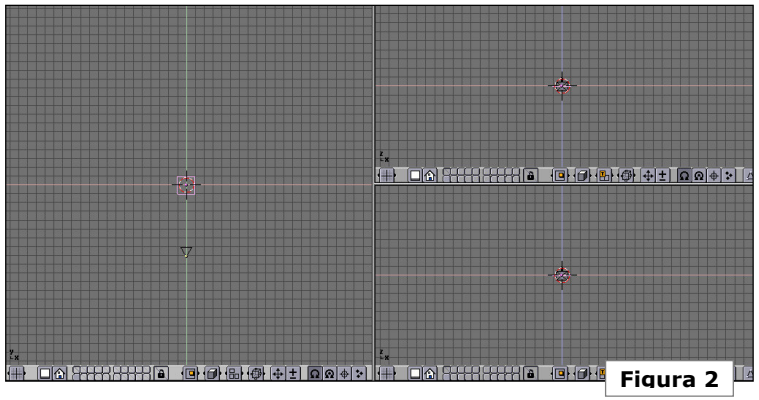


Figura 2

tendremos la vista de la cámara. **1**, **3**, **7** nos darán las vistas de la escena superior, frente, y perfil respectivamente.

Vamos a definir distintas vistas en las ventanas 3D para tener un buen control de las posiciones 3D. Configuraremos las vistas como se muestra en la Figura 3.

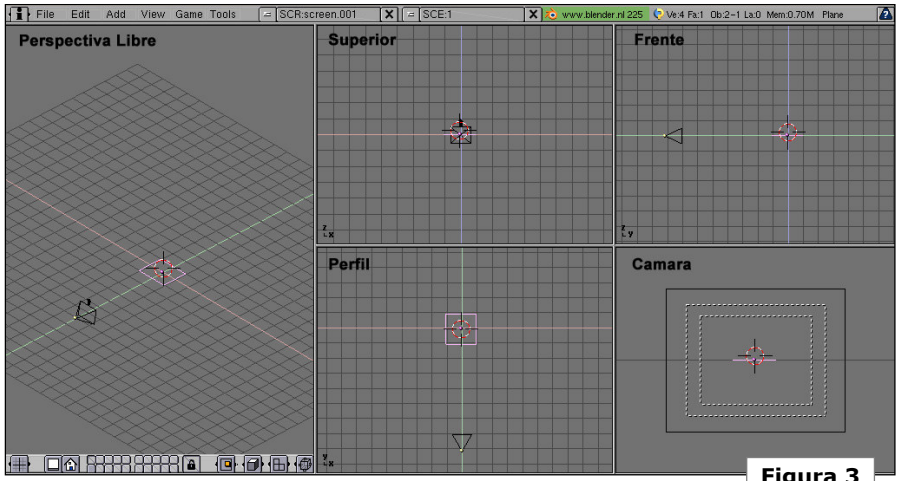


Figura 3

Con el plano por defecto seleccionado, pulsaremos **SUPRIMIR** y lo quitaremos de la escena. Añadiremos un cubo. Para ello, pincharemos en el botón de la Caja de Herramientas o a la **barra espaciadora**: **ADD** + **Mesh** + **Cube**. Cuando insertamos un objeto en la escena, lo hará en *modo de edición de vértices*. Para volver al *modo de edición de objeto*, pulsaremos **TAB**.

Algunas operaciones que podemos realizar con el objeto es rotarlo; con él seleccionado pulsamos **R** y desplazamos el ratón. Si pulsamos ahora **X**, **Y** o **Z**, delimitaremos la rotación a ese eje. Si rotamos con la tecla **MAYÚSCULAS** pulsada, lo haremos en modo de precisión. Si tenemos la tecla **CONTROL** pulsada, lo haremos de 5 en 5 unidades.

Otra operación básica es el desplazamiento (tecla **G** del inglés "*Grabber*"). Si desplazamos con la tecla **CONTROL** pulsada, lo haremos ajustándonos a la rejilla. Al igual que con la rotación, la tecla de **MAYÚSCULAS** se corresponde con el modo de precisión.

El escalado se realiza con la tecla **S**. Para restringir el escalado en alguna dimensión, pulsaremos S, después **ALT** (dejándolo pulsado) y haremos **BIR**. Según arrastremos el ratón la caja se achatará en una dimensión o en otra.

El Render se realiza a través del botón de visualización (**F10**) y pinchando en el botón de **RENDER** o bien con el atajo de teclado **F12**. ¿Por qué no sale nada?. No hemos iluminado la escena. Ocultaremos la ventana de render pulsando **F11**. Añadiremos una luz de tipo Hemi en la posición que muestra la Figura 4. Para ello, situaremos el cursor 3D en la posición adecuada ayudándonos de las vistas Superior, Frontal y de Perfil. El cursor 3D se desplaza haciendo click (sin arrastrar) en cualquier punto de la ventana 3D con **BIR**.

Añadiremos la luz accediendo a la Caja de Herramientas: **ADD** + **Lamp**. Cambiaremos el tipo de lámpara con el botón (o pulsando **F4**). Seleccionaremos el tipo de foco Hemi: Lamp | Spot | Sun | Hemi

Realizaremos un render y comprobaremos que el resultado es similar al mostrado en la

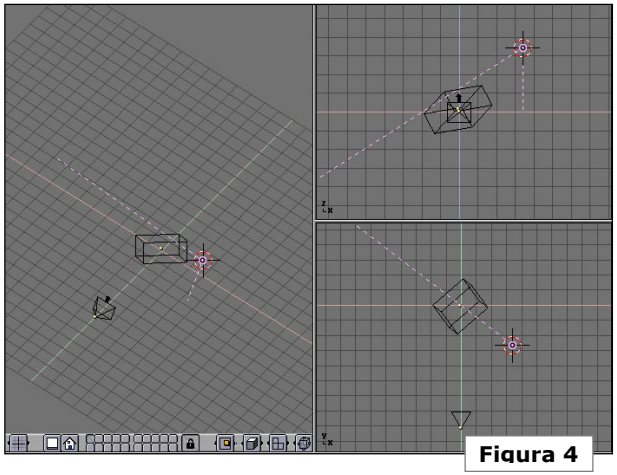


Figura 4

Figura 5. Podemos salvar el render, pulsando **F3**. El formato de la imagen que salvaremos se selecciona en el apartado del botón de visualización (**F10**).

Elegiremos, por ejemplo, JPG conservando la calidad de la imagen en un valor alto (rango 80-100).

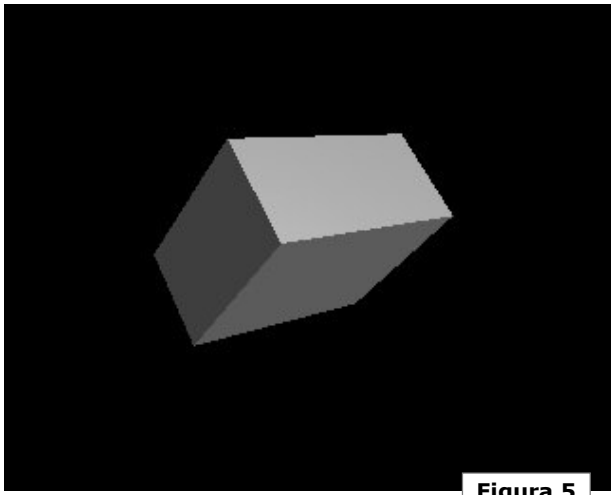


Figura 5

Práctica 2

Introducción a Blender (II) + GIMP

Como continuación de la sesión anterior, aprenderemos a crear en Blender objetos más complejos partiendo de un boceto 2D. Veremos cómo utilizar el comando de extrusión y a trabajar con curvas. Para terminar, realizaremos una composición por capas utilizando GIMP.

Arrancamos Blender y eliminamos el plano por defecto. Pulsamos **Shift+F7**. La pantalla que nos aparece nos permite cambiar la imagen de fondo, variar la distancia entre elementos de la rejilla, y los valores de *recorte* en la vista de perspectiva. Pinchamos en el botón **Background Pic**, luego en Load y cargamos la imagen *logouclm.jpg*. Hecho esto, para volver a la ventana 3D con la imagen de fondo, pulsamos **Shift+F5** o cambiamos el tipo de ventana a . Debemos obtener una vista la figura 1.

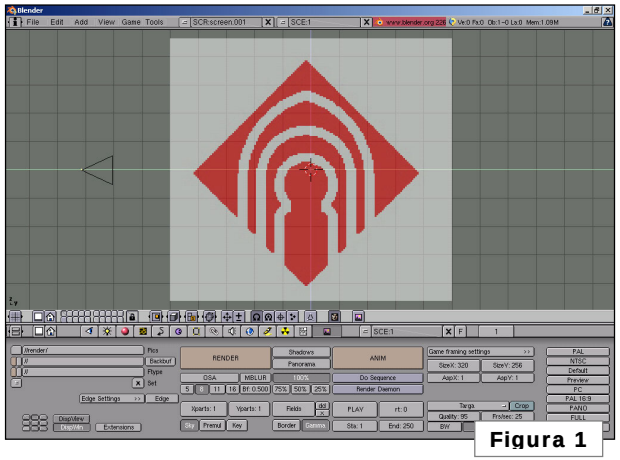


Figura 1

Vamos a trazar el contorno del logotipo por medio de una curva. Para ello, añadiremos una a la escena. Pulsamos **Barra Espaciadora**, **Add** + **Curve** + **Bezier Curve**. Con la curva añadida, y sin salir del modo de edición (sin pulsar TAB), vamos a los botones de edición (**F9**) y convertimos la curva a polígono. Esto nos permitirá trabajar de forma más cómoda. Cuando hayamos ajustado los puntos de control al logotipo, convertiremos de nuevo a cuva de Bezier, ajustando la tensión en los tramos que lo requieran.

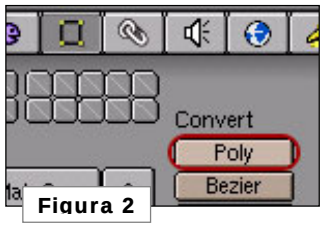


Figura 2

Por defecto, tenemos todos los vértices seleccionados (todos en color amarillo). Pulsamos la tecla **A** y deseleccionamos todos (aparecen en rosa). Ahora desplazaremos los vértices al contorno exterior del logotipo (seleccionamos cada vértice de forma individual con **BDR**, y desplazamos pinchando una vez la tecla **G**) . La curva original está formada por 6 vértices. Los situaremos en las posiciones 1 .. 6 tal y como muestra la figura 3. Podemos realizar zoom en la vista 3D (**Control** + **Alt** + **BIR**) para tener una mayor precisión a la hora de situar los vértices.

Para añadir vértices: manteniendo pulsado **Control**, haremos **BIR**. De esta forma, añadiremos los vértices 7 y 8. Para cerrar la figura (es decir, añadir la arista que une 8 con 1), pulsaremos la tecla **C**. Podríamos pensar que hacen falta más vértices, por ejemplo, entre 6 y 7. Sin embargo, no es necesario ajustar mejor la superficie en las secciones curvas, ya que lo haremos por medio de la tensión en la curva tipo Bezier.

Seleccionamos todos los vértices , pulsando la tecla **A** (deben quedar todos en color amarillo). Volvemos a los botones de edición y convertimos la curva a tipo Bezier. Deben aparecer unos segmentos de color verde. Estos son los segmento de tangente, que especifican el grado de tensión en la pendiente de cada sección de curva. Deseleccionamos todos los vértices y elegimos uno de los que requieren curvatura (por ejemplo el 6). Una vez seleccionado, pulsamos la tecla **H**. Esto nos alineará los manejadores de tensión de la cur-

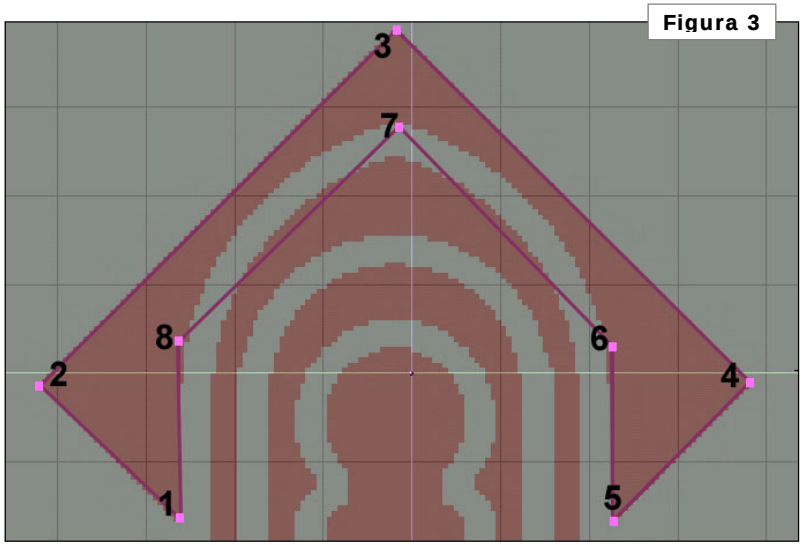


Figura 3

va, permitiendo ajustar la curvatura en cada sección de la misma.

Ajustaremos los puntos de tangente hasta obtener la geometría que se ajuste bien a la imagen de fondo. Cuando hayamos terminado, volveremos al *modo de edición de objeto* (pulsando **TAB**), y pulsaremos la tecla **Z** para obtener la representación sombreada, tal y como muestra la figura 4. Volvemos a la representación con líneas pulsando de nuevo la tecla **Z**.

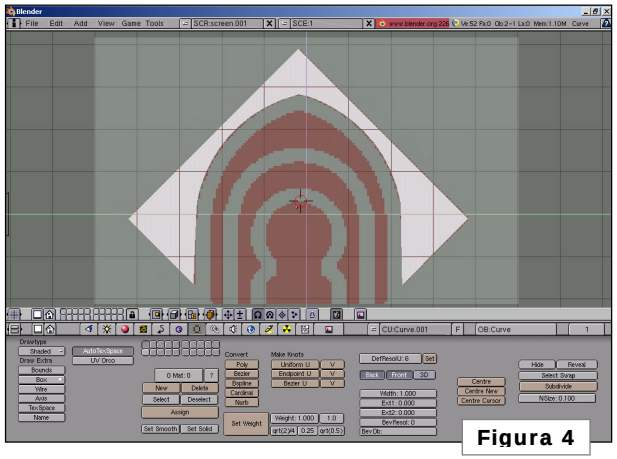


Figura 4

Añadiremos otras curvas de Bezier a la escena y procederemos de forma similar, convirtiendo a polígono, ajustando puntos y volviendo a convertir a Bezier para modelar las otras tres secciones que definen el logotipo. Cuando hayamos terminado, podemos quitar la imagen de fondo pulsando de nuevo **Shift+F7** y pinchando en *Background Pic*.

Pasaremos a extrusionar las curvas para obtener el modelo 3D. Rotaremos la vista para tener un ángulo mejor a la hora de ver la operación realizada. Elegimos en modo de selección de objeto una de las curvas (por ejemplo, la interior). Cambiamos los valores de extrusión y redondeado tal y como se muestra en la figura 5. Podemos cambiar los valores en los botones de inserción, pinchando y arrastrando sobre ellos, o bien manteniendo pulsada la tecla **Shift** y haciendo **BIR** sobre ellos. Repetiremos la operación para las otras secciones del logotipo.

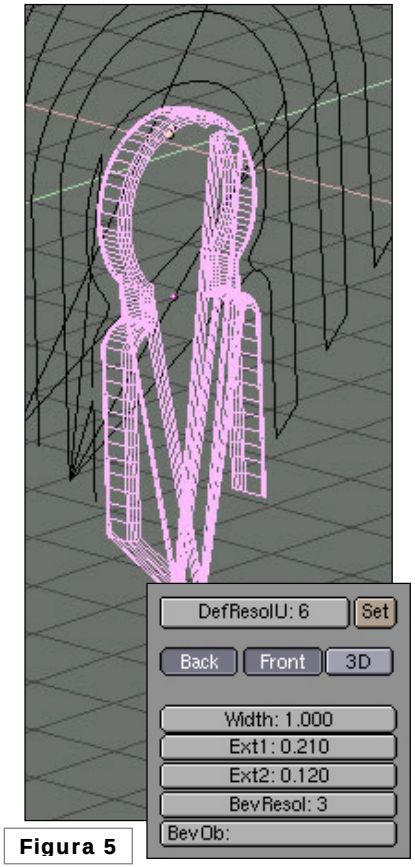


Figura 5

Vamos a aplicarle un material sencillo al modelo. Seleccionamos una de las partes y pinchamos en el botón de materiales, tal como muestra la figura 6 (o bien pulsamos **F5**). Como no hay ningún material en la escena, tendremos que añadir uno nuevo, que luego aplicaremos a todos los elementos que forman en logotipo. Se pueden crear tantos materiales como sean necesarios, pero en este ejemplo aplicaremos el mismo a todos los elementos. Para crear un nuevo material, pincharemos en el botón que tiene un guión dibujado (ver figura 6), y después en **ADD NEW**. De todas las propiedades que pueden tener los materiales, únicamente vamos a definir el color. Ajustaremos los valores del color tal y como se indica en la figura 7. En próximas sesiones de prácticas, veremos con más profundidad las opciones de materiales y texturas en Blender.

Daremos un nombre al material que hemos creado; al lado del botón que hemos pinchado para añadir el material, tendremos una caja de texto. Manteniendo pulsado **Shift**, pinchamos con **BIR** y tecleamos un nombre para el material , por ejemplo, 'UCLMRojo'. 

Aplicaremos el material que hemos creado al resto de piezas del logotipo. Para ello, seleccionaremos cada una de las partes del logotipo y, pulsando en el boton que utilizamos para añadir un nuevo material, esta vez seleccionaremos el que acabamos de crear.

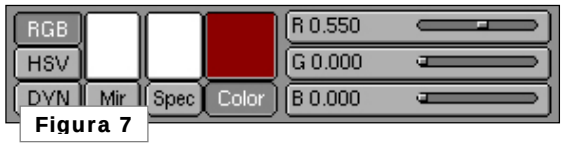


Figura 7

Figura 6

Vamos a renderizar la escena. Situaremos la cámara para obtener una vista como la que se muestra en la figura 8. Añadiremos un foco de iluminación de tipo Spot. Para ello, insertamos un objeto de tipo **Lamp** a la escena y lo cambiamos a tipo **Spot**. Orientamos el foco para que apunte al logo 3D, rotándolo con ayuda de las vistas superior, frontal y lateral. La situación del foco deberá ser similar a la mostrada en la figura 8. Probamos a renderizar la escena. Si el objeto sale muy oscuro, podemos ajustar la intensidad del foco en los botones de foco (**F4**), en el campo **Energy** a un valor superior. Ajustaremos las opciones de renderizado como se indica en la figura 9.

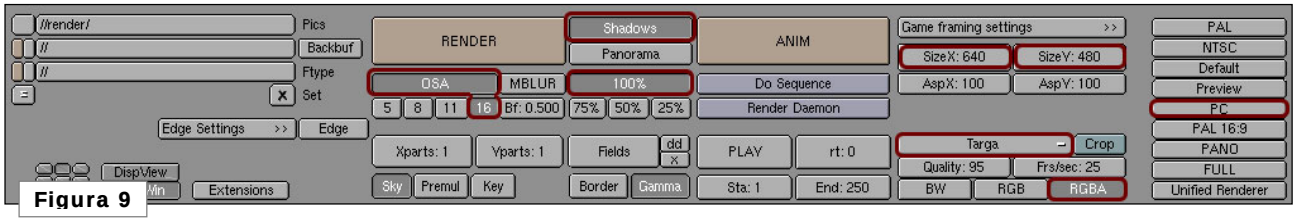


Figura 9

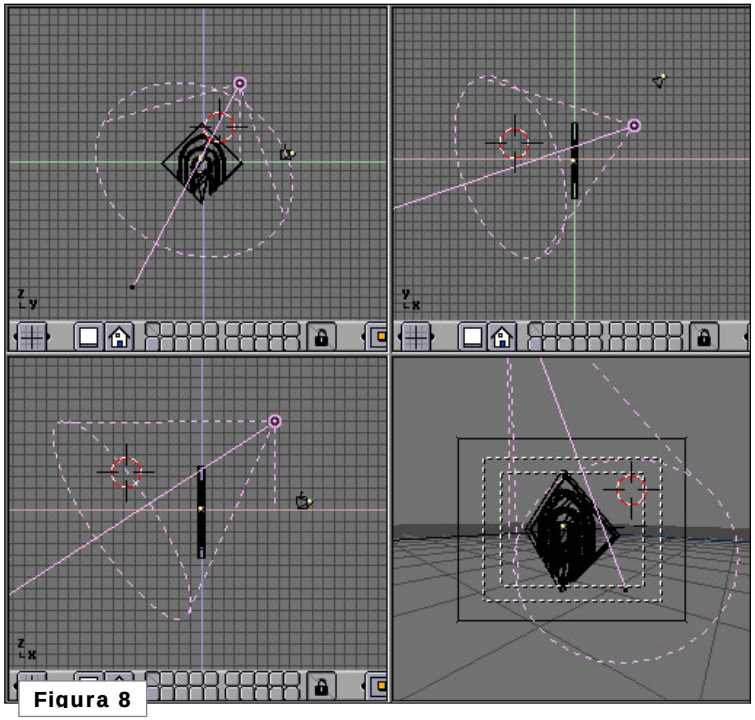


Figura 8

Activamos el botón de **Shadows** para que el motor de renderizado calcule las sombras de la escena. **OSA (OverSAmpling)** con valor 16 hace que el *antialiasing* se aplique con la mayor intensidad. Seleccionamos uno de los tamaños predefinidos pinchando en **PC** (**640**x**480**). Con 100% indicamos que el renderizado se haga al tamaño total elegido. Cuando hacemos pruebas es interesante poner este valor al 50% o al 25% para que la generación de la imagen lleve menos tiempo. Con **RGBA**, indicamos que la imagen resultado debe tener canal Alpha de transparencia. Así, tendremos que seleccionar un formato que soporte canal Alpha (como **Targa**; *.TGA). El canal Alpha codifica, en tonos de grises, qué parte de la imagen es de objeto y qué parte es el fondo. Un valor de negro indica que es fondo, mientras que blanco indica objeto sólido. Los tonos de gris indican objetos semitransparentes. El resultado de renderizar la escena debe ser similar al mostrado en la figura 10. En la esquina superior derecha de la figura 10 se ha incluido el canal Alpha de la imagen resultado.

Guardamos la imagen como *solido.tga*. Salvamos también el fichero de blender y cambiamos el material.

Nos vamos al menú de materiales y asegurándonos que estamos trabajando con **MA:UCLMRojo**, cambiamos el color del material en las tres componentes (R,G,B) a **0.200**. Además, activamos el botón de **Wire** (situado en el centro).
Como todas las secciones del logo comparten el mismo material, al renderizar de nuevo la escena comprobamos que el logo entero ha cambiado. Debemos obtener un resultado similar al de la figura 11. Sin haber cambiado ninguna opción de renderizado, guardamos la imagen resultante como *alambre.tga*.

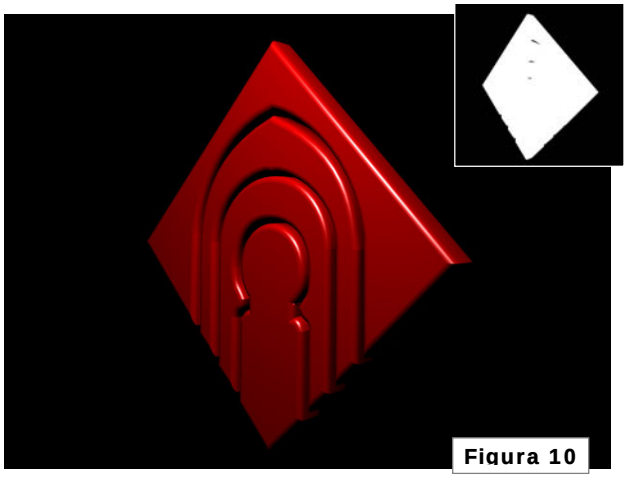


Figura 10

Salvamos el proyecto de blender y abrimos GIMP, el programa de composición y retoque

fotográfico GNU. Cargamos las imágenes solido.tga y alambre.tga. Comprobamos que el fondo de la imagen no es negro; está formado por "cuadritos". Esto indica que GIMP ha importado las imágenes con el canal de transparencia correctamente.

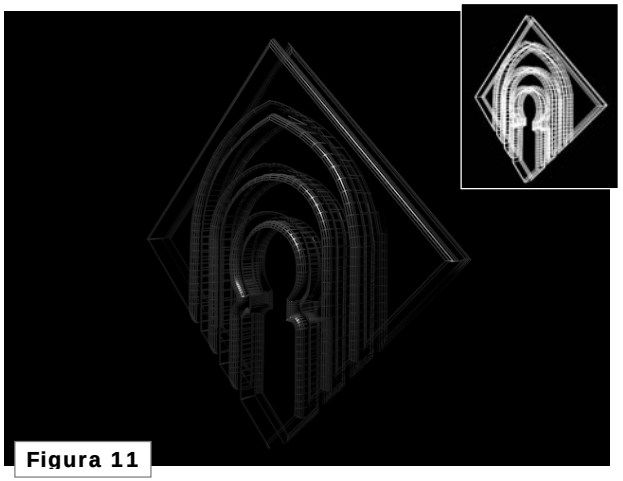


Figura 11

Recordemos que las capas actúan como "acetatos". El orden de las capas importa (al igual que si colocamos una transparencia delante de otra). Renombraremos la capa de la imagen sólida. Para ello, nos situamos sobre la capa y haciendo **BDR** + **Editar atributos de capa**, le pondremos como nombre "solido". Añadiremos una capa nueva, que esté rellena de color blanco. Para ello, pincharemos en el botón  , y le pondremos como nombre **Fondo**. La capa se ha situado delante del logo. Pincharemos en la flecha "abajo", como indica la figura 12, para ordenar las capas.

Seleccionamos la ventana donde se ha abierto el fichero alambre.tga. Hacemos **BDR** sobre la imagen y seleccionamos **editar**, **copiar** (o bien pulsamos **Control + C**). Después, nos vamos a la imagen solido, y pegaremos el logo de alambre como una nueva capa. Para ello, **BDR + editar + pegar** (o **Control + V**). Vemos que aparece en la ventana de capas como una selección flotante. Para generar una capa nueva con la imagen, pincharemos de nuevo en el botón de nueva capa  . Renombramos la capa como "alambre".

Figura 12

Situamos la capa alambre entre la capa solido y la capa fondo. En la ventana de **Selección de brocha**, seleccionamos la que se muestra en la figura 13 y pinchamos en Nuevo. Con esto, haremos una brocha personalizada partiendo de la que hemos seleccionado. Ajustamos los parámetros a: **radio = 44.8, dureza = 0.2** aprox.. Ángulo y razón de aspecto los dejamos como están. Damos el nombre que queramos a la brocha. Seleccionamos la brocha, la herramienta de borrado  , y nos vamos a la capa solido. Ajustamos la opacidad de la goma (en la ventana de **opciones de herramienta**) a 25 aprox. Hecho esto, borramos una parte de la capa sólido (la derecha, por ejemplo). Hacemos otra pasada consiguiendo un efecto de degradado, como se muestra en el resultado final (figura 14).

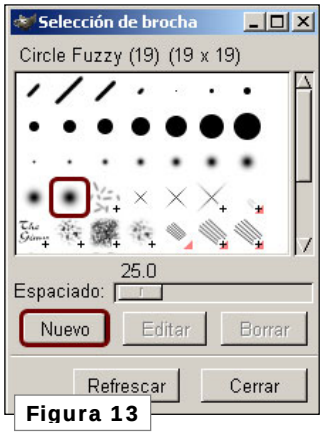


Figura 13

Para terminar, añadimos un efecto de sombra a la capa alambre. Seleccionamos la capa, nos vamos a la imagen y **hacemos BDR + Script-Fu + Sombra + Sombra Arrojada**. Podemos dejar los parámetros tal y como están y aceptamos. Vemos que, debajo de la capa alambre se ha creado una capa llamada Drop-Shadow. Si no nos gusta el resultado, eliminamos la capa (pinchando en el botón del cubo de basura), y probamos a cambiar los parámetros del script. Para terminar, pinchamos con BDR sobre la imagen, y guardamos el fichero con formato **XCF** (el formato nativo de GIMP), que permite almacenar el fichero tal y como estamos trabajando (capas, canales, etc...).

Si queremos utilizar la imagen desde otras aplicaciones, tendríamos que salvarlo en un formato como JPG. En este caso GIMP nos advierte que perderemos la información acerca de las capas, canales, etc....

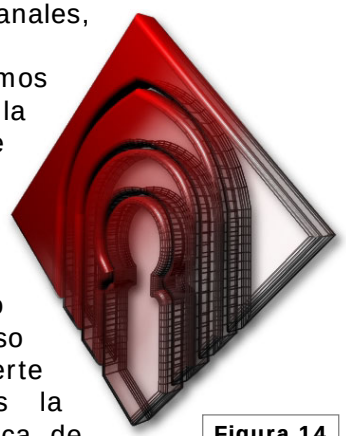


Figura 14

Práctica 3

Modelado con Wings 3D y Teddy

El objetivo de esta sesión es aprender a utilizar dos herramientas de modelado que destacan por su facilidad de uso y potencia: Wings 3D y Teddy. No debemos centrarnos en obtener un resultado exactamente igual al mostrado en las imágenes, sino en descubrir las posibilidades que nos brindan ambas aplicaciones. En la siguiente sesión veremos en profundidad las opciones de modelado de Blender.

Comenzaremos modelando un coche con Wings 3D. En esta herramienta, partimos de una forma básica (primitiva 3D) y aplicando una serie de operaciones a nivel de vértice, arista, cara o al objeto global, iremos modificándola hasta obtener el modelo final. Antes de empezar a trabajar, personalizaremos el modo de desplazar la cámara por la escena como Blender (**Alt + BIR** para rotar, **Control + Alt + BIR** para hacer Zoom...). En el menu superior; **Edit/Camera Mode/ Camera Mode [Blender]**.

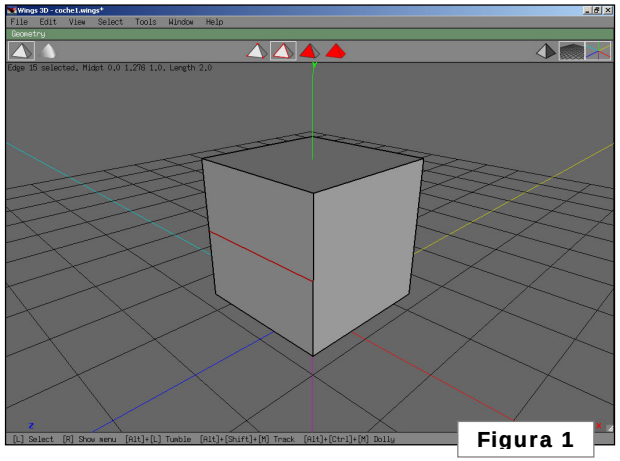


Figura 1

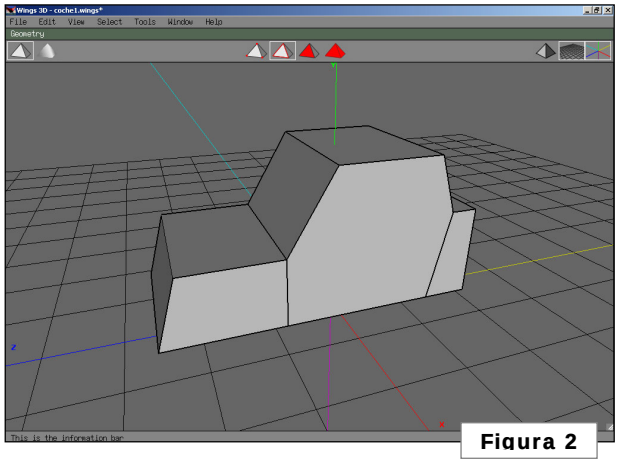


Figura 2

Añadimos un cubo a la escena con **BDR/Cube**. En la parte superior de la pantalla aparece una barra de herramientas. Las dos pirámides de la izquierda nos permiten seleccionar la forma en que se mostrará el objeto (caras o sombreado suave). La parte central nos permite seleccionar en qué nivel queremos trabajar (vértice, arista, cara u objeto). Cuando un elemento está seleccionado (vértice, arista, cara u objeto), aparece en color rojo. Podemos seleccionar de forma incremental elementos de la escena. En cualquier modo de selección que nos encontremos, podemos utilizar las teclas "**+**" y "**-**" para extender o contraer la selección realizada a los elementos vecinos [realizar la prueba seleccionando un vértice del cubo y pulsar **+** dos veces y luego **-**]. Podemos deseleccionar todos los elementos pulsando la **barra espaciadora**. Wings 3D implementa múltiples niveles de "undo/redo", que podemos recuperar con **Control + Alt + Z** y **Conrol + Shift + Z**. Con **BDR** obtenemos el conjunto de operaciones que podemos aplicar a los elementos seleccionados. Por ejemplo, partiendo del cubo que hemos insertado y seleccionando las dos aristas verticales de la cara frontal, haremos **BDR/Connect**. Con esto, obtenemos una nueva arista que conecta las dos anteriores, tal y como se muestra en la figura 1.

Moveremos la arista recién creada ligeramente hacia abajo (en el eje y), haciendo **BDR/Move/Y**. En la cara posterior del cubo, añadiremos otra aristas y la moveremos ligeramente arriba. Seleccionando las caras inferiores, extrusionaremos en la dirección del vector normal de cada cara: **BDR/Extrude/Normal**. Ajustaremos las aristas de las caras extrusionadas (desplazamientos en el eje Z), hasta obtener algo similar a la Fig 2.

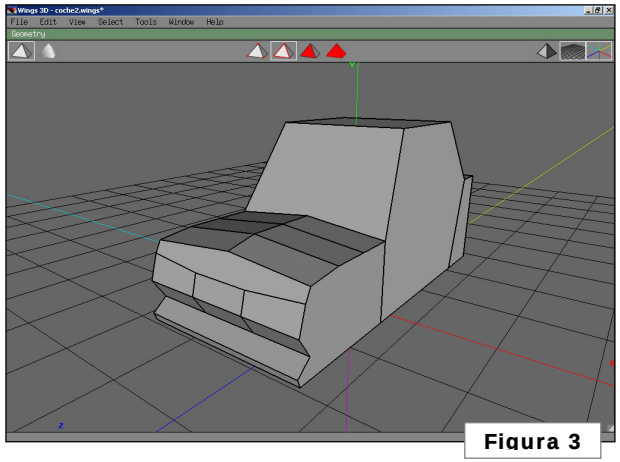


Figura 3

Seleccionaremos las dos aristas longitudinales que forman el capó del coche, y las conectaremos (obteniendo una arista central). Esta nueva arista, la levantaremos ligeramente para comenzar a darle forma. Seleccionaremos las tres aristas, y aplicaremos

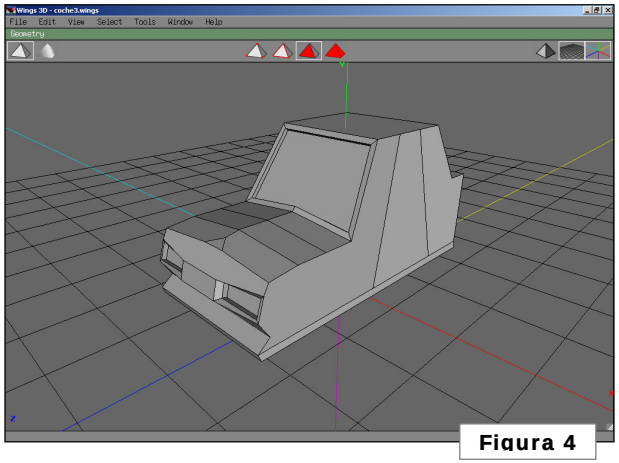


Figura 4

BDR / Cut / 3. Con esto, obtenemos dos nuevos vértices por arista (las hemos cortado en 3 trozos). Utilizaremos ahora (trabajando a nivel de vértice) la orden de **BDR / Connect**. Con esto, obtenemos cuatro nuevas aristas que unen los seis vértices creados con la orden anterior. Procederemos de forma similar para modelar el hueco de los faros y el parachoques. Desplazando las aristas obtenidas, deberemos obtener un resultado similar a la figura 3.

Seleccionaremos las caras que forman la luna delantera y los faros. Aplicaremos **BDR / Inset** y a continuación **BDR / Extrude / Normal** (hacia el interior). Realizaremos el triángulo de la zona central del capó

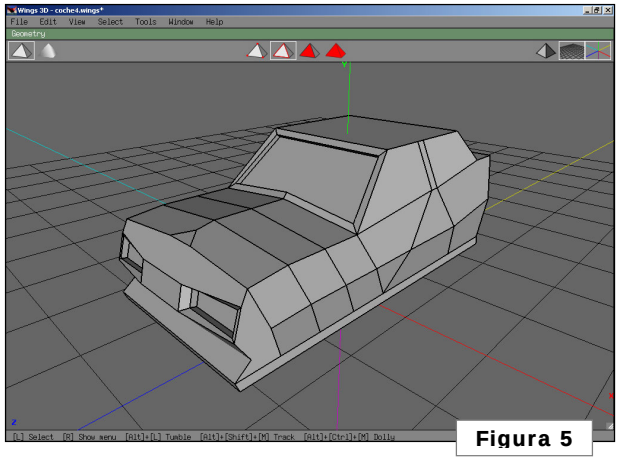


Figura 5

del capó seleccionando el vértice **BDR / Bevel**. El resultado de estas operaciones es el mostrado en la figura 4. Continuaremos dividiendo las caras laterales del coche, ajustando las aristas y vértices resultantes tal y como se muestra en las figuras 5 y 6. En este caso, además, se ha ajustado la altura del techo.

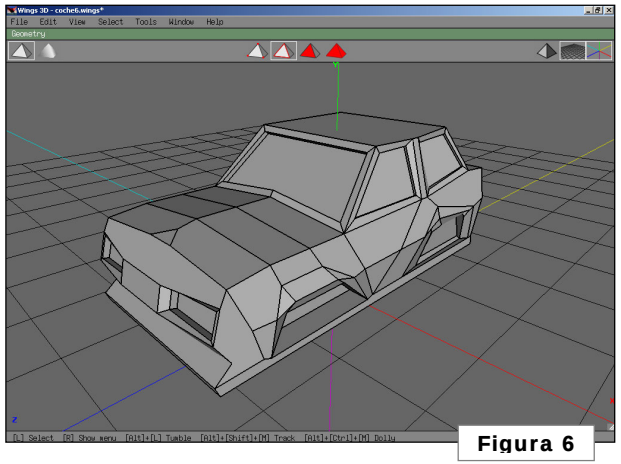


Figura 6

Resulta muy útil la herramienta Tweak (**Menú / Tools / Tweak**). Nos permite desplazar los vértices del objeto libremente (cambiando el punto de vista y pinchando-arrastrando con **BIR**). Saldremos de este modo de edición cuando hagamos **BDR**. Podemos utilizar el modo "**magnet**" (pulsando **1**), en el que desplazaremos grupos de vértices según el radio de magnetización (se puede variar pulsando **+** y **–**). Debe quedarnos claro que el desplazamiento en este modo es dependiente del punto de vista de la escena. Por tanto, para obtener el resultado esperado, tendremos que realizar multitud de rotaciones de cámara. La figura 7 muestra una captura de edición en este modo.

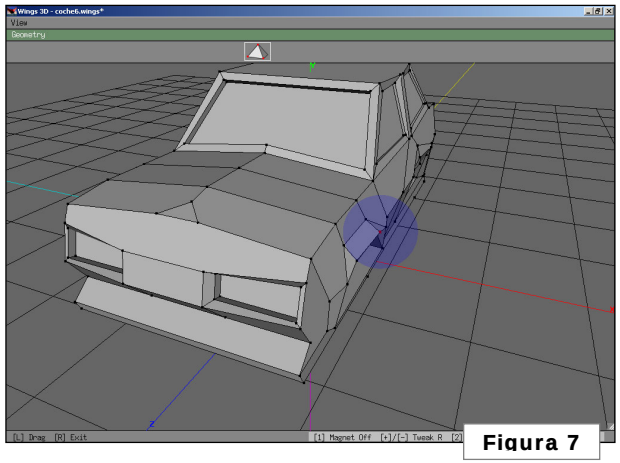


Figura 7

Para modelar las ruedas, primero añadiremos un cilindro a la escena, escalándolo respecto del eje Y. Seleccionaremos las caras superior e inferior, y después **BDR + Intrude**. Con esto, obtendremos una especie de "canuto" (ver figura 8). Insertamos otro ci-

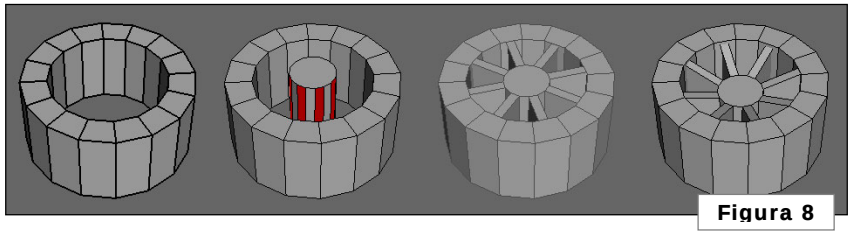


Figura 8

lindro, que escalamos para que quede en el interior del anterior, pero con la misma altura. Seleccionamos alternativamente las caras laterales y aplicamos extrusión (sin salirnos del cilindro exterior. Para terminar, seleccionamos las caras superior e inferior del cilindro interior, y las desplazamos en dirección a la normal, hacia el centro. El resultado final de la rueda es el mostrado en la figura 8.

Seleccionaremos las dos partes que forman la rueda y **BDR + Combine**. De ahora en adelante, Wings 3D tomará la rueda como un único objeto. Hacemos 3 copias más de la rueda (**BDR + duplicate**) y la situamos en el modelo del coche. El modelo del coche

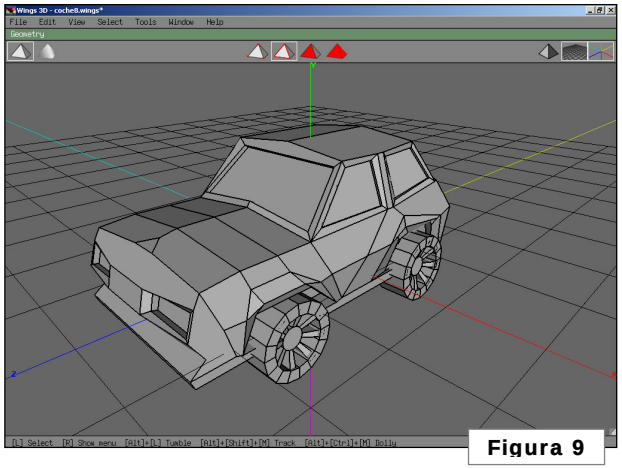


Figura 9

finalizado es el que se muestra en la figura 9. Podemos exportar el modelo para cargarlo desde Blender, en formato WRML 2.0. En Blender (versión 2.2.5; la 2.2.6 tiene un bug que no importa correctamente estos ficheros), abrimos el objeto, lo seleccionamos y pulsamos **Control + N** (para que recalcule las normales de las caras). Después, en modo de edición de vértices (pulsar **TAB**), vamos a los botones de edición (**F9**) y activamos los botones de "**draw normals**" y "**draw faces**", para comprobar que el cálculo de las mismas ha sido correcto. Podemos activar la vista en sombreado (pulsando **Z**) para comprobar que el modelo se ha importado correctamente.

Abrimos Teddy. Después de leernos el minitutorial que acompaña la distribución, haremos un personaje que importaremos en Wings 3D, para salvarlo como VRML 2.0 (que cargaremos en Blender). Podemos obtener un dinosaurio como el que se muestra en la figura 10. Aplicaremos materiales simples a los objetos y, añadiendo un plano como suelo de la escena, renderizaremos la escena en Blender obteniendo un resultado similar al que se muestra en la figura 11.

Podemos mejorar el aspecto final del dinosaurio asignando un sombreado suavizado (y no plano), pinchando en el botón "**set smooth**" del apartado de edición (**F9**). Estas opciones y muchas más las veremos en detalle en la siguiente sesión de prácticas.

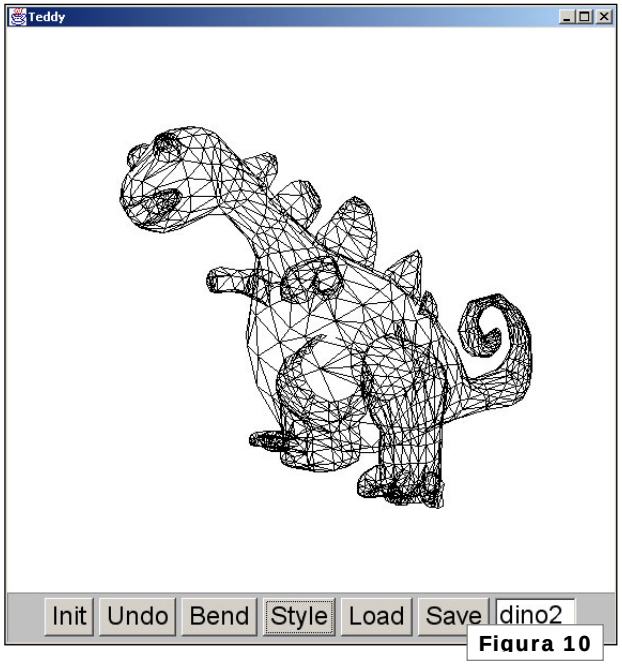


Figura 10

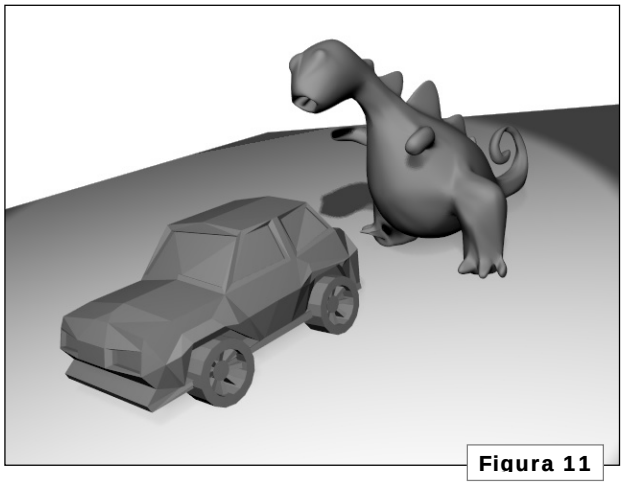


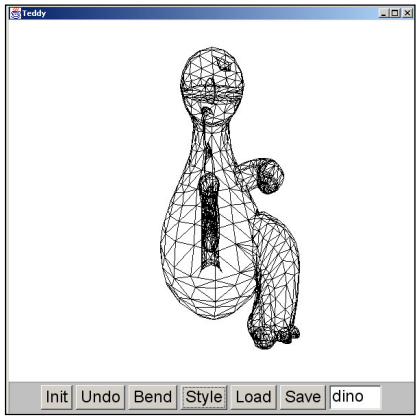
Figura 11

Práctica 4

UV-Mapping con Blender

Adelantamos la sesión de texturizado mediante UV-Mapping en Blender para poder crear personajes completos cuanto antes y utilizarlos en nuestros proyectos de animación. En esta sesión aprenderemos a pintar la "piel" de nuestro modelo utilizando esta técnica, vista en teoría que permite un posicionamiento muy preciso. El objetivo es texturizar un modelo complejo creado con cualquier técnica de modelado.

Partimos del modelo creado en Teddy mostrado en la figura 1. Sólo nos hemos ocupado de construir la mitad del personaje. La otra mitad la conseguiremos por "espejo" del este modelo en Wings 3D. Podéis encontrar el modelo en el fichero *dino.obj*, no obstante es aconsejable que intentéis construir vuestro propio personaje similar al mostrado en la figura 1.



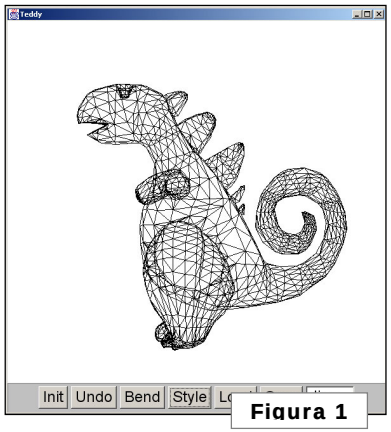


Figura 1

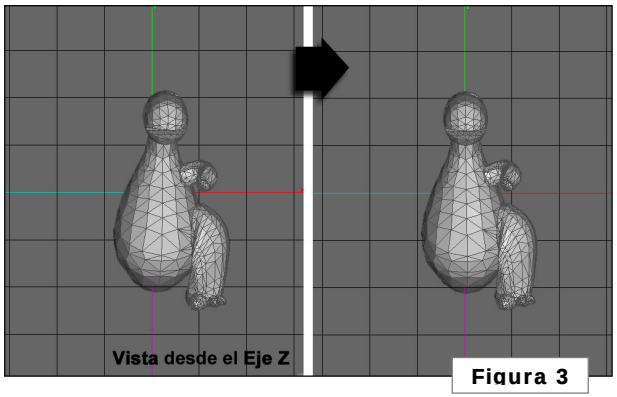


Figura 3

Importamos el modelo en Wings. Activamos la vista de los ejes y el modo ortogonal en la parte superior derecha de la ventana. El modo ortogonal nos elimina el efecto de perspectiva. Esto nos permite seleccionar de forma más precisa conjuntos de caras y trabajar sin distorsiones en la vista 3D.

Pulsando las teclas **x**, **y**, **z** veremos el modelo desde estos ejes. Rotaremos y desplazaremos el modelo para que se quede perfectamente centrado respecto del plano YZ, tal y como se muestra en las figuras 2 y 3. Es importante situar correctamente el modelo en la operación anterior porque nos facilitará la siguiente selección de caras.

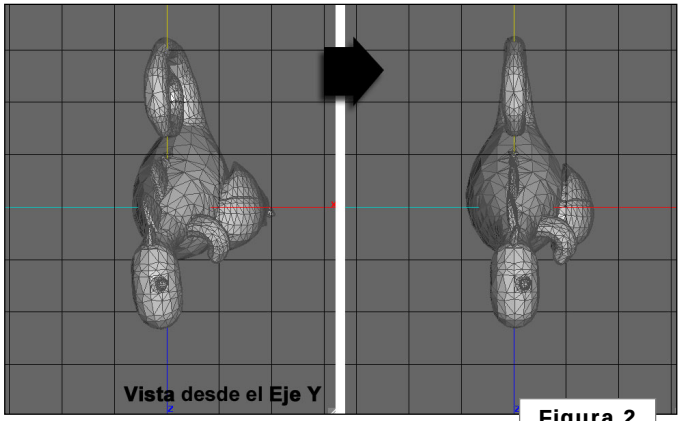


Figura 2

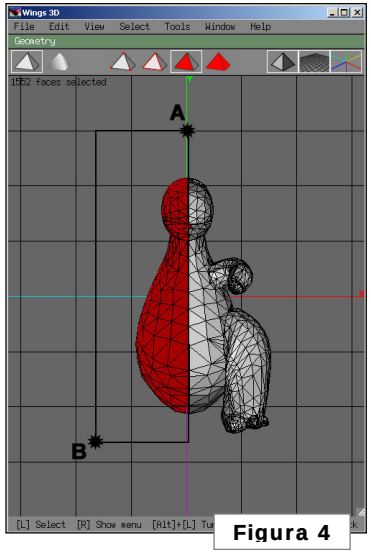


Figura 4

Con la vista desde el eje Z, y el modo de selección a nivel de cara, seleccionaremos la mitad izquierda del modelo pinchando y arrastrando desde la posición **A** de la figura 4, a la posición **B**. Obtendremos una selección de caras bastante precisa, pero habrá que rotar la cámara y ajustar la selección manualmente, sobre todo en la zona de las escamas de la espalda y en la cola. Eliminaremos las caras sobrantes y borraremos (tecla **Suprimir**) la selección. El modelo debe quedarnos como muestra la figura 5. Para evitarnos efectos indeseables con la herramienta de espejo, aplanaremos primero la cara resultante (aún seleccionada en color rojo) mediante **BDR + Flatten + Normal** (alisar la cara respecto de su vec-

tor normal). Hecho esto, utilizamos la herramienta de espejo con **BDR + Mirror** para obtener el modelo del dinosaurio simétrico. Exportamos el modelo como VRML y lo importamos en blender siguiendo los pasos explicados en la práctica anterior.

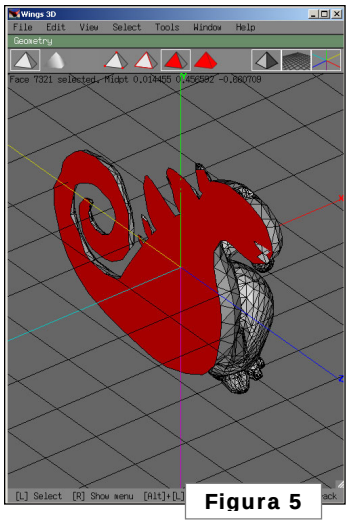


Figura 5

Configuramos Blender para obtener una división de ventanas como se muestra en la figura 6. La ventana de la derecha la cambiamos a tipo *ImageWindow* (**Shift + F10**, o seleccionando el icono). Antes de continuar, veamos algunas opciones del entorno que nos van a ser útiles en esta práctica. En la cabecera de la ventana 3D, distinguimos los iconos que se muestran en la figura 7. Con el icono **(a)** podemos seleccionar el modo de visualización, en perspectiva o vista ortográfica. La vista ortográfica no agranda los objetos cuanto más cerca están de la cámara. Podremos también cambiar entre estos tipos de vistas pulsando el **5** del teclado numérico. El icono **(b)** nos permite seleccionar la vista frontal, lateral, etc... Tiene el mismo efecto que las teclas **1**, **3**, **7** que hemos utilizado en prácticas anteriores. El icono **(c)** nos permite seleccionar el modo de sombreado del objeto mientras trabajamos con él. Nos interesará el primer modo empezando por arriba (detallado, que nos muestra la textura aplicada al modelo), y el segundo empezando por abajo (la representación de alambre que hemos utilizado en prácticas anteriores). Los iconos **(d)** y **(e)** que ya hemos utilizado, nos permiten hacer **zoom** y desplazar la **vista**. El icono **(f)** nos permite pasar a modo de edición de vértices (equivalente a la tecla **Tab**). Con **(g)** pasamos al modo de pintado de vértices y **(f)** nos permite pintar la textura directamente sobre el modelo. El icono **(i)**, equivalente a pulsar la tecla **f**, nos permite pasar al modo de selección de caras. Por último, **(j)** sirve para renderizar la vista actual. Con **F3** podremos salvarla en el formato que hayamos seleccionado en el menú de visualización (**F10**).

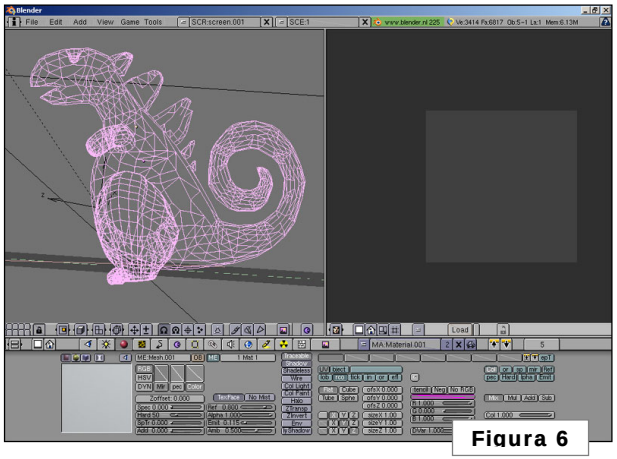


Figura 6

a b c d e f g h i j

Figura 7

Pasamos al modo de selección de caras (pulsando la tecla **f**). Vemos que el modelo se ha puesto de color blanco, y el icono **(i)** de la figura 7 aparece seleccionado. Seleccionamos las caras que forman el cuerpo del dinosaurio, dejando las extremidades para más adelante. Para realizar la selección de una forma más sencilla, y sin salir del modo de selección de caras, pulsaremos **Tab**. Todos los vértices aparecerán seleccionados (en color amarillo), y de igual forma las caras asociadas.

Pulsando dos veces la tecla **B**, activaremos el modo de selección de vértices mediante un "*pincel*". El cursor aparecerá delimitado por una circunferencia. Podremos añadir caras a la selección pinchando y arrastrando **BIR**. Para quitar caras a la selección, pinchar y arrastrar con **BIR + ALT** pulsado. Podemos aumentar el radio del pincel con la

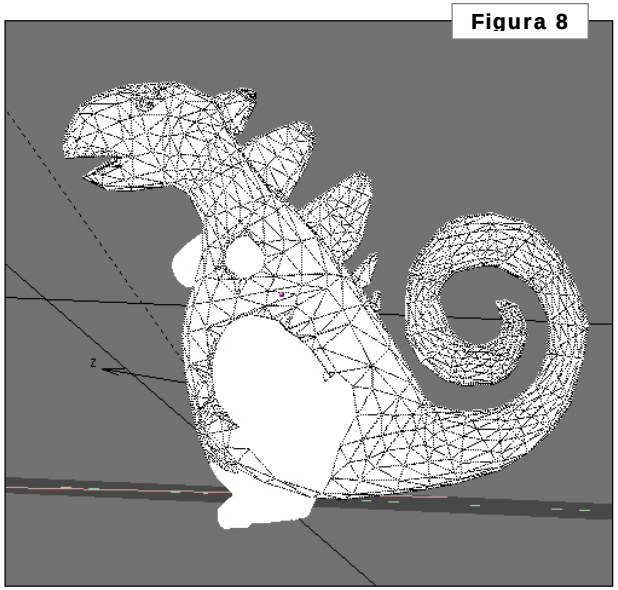


Figura 8

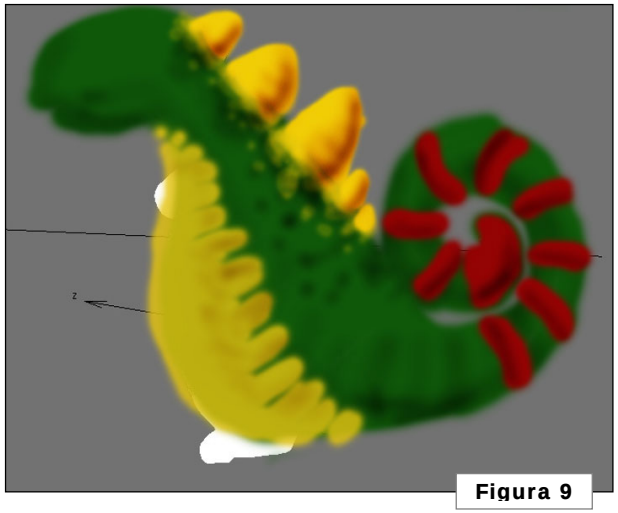


Figura 9

tecla **+**, y disminuirlo con la tecla **-**. Abandonaremos el modo de selección con esta técnica haciendo un **BDR**. Teniendo en cuenta que la selección de caras "*atraviesa*" el modo, seleccionaremos las vistas que nos hagan falta para dejar inactivas las caras de las extremidades del dinosaurio. Podemos volver al modo de edición de caras, pulsando Tab. El resultado de la selección, debe ser el mostrado en la figura 8. Las extremidades se quedan totalmente blancas porque no hay selección de caras. No pasa nada si no hemos seleccionado perfectamente el recorte de las extremidades, ya que las podremos seleccionar cuando "pintemos" las extremidades.

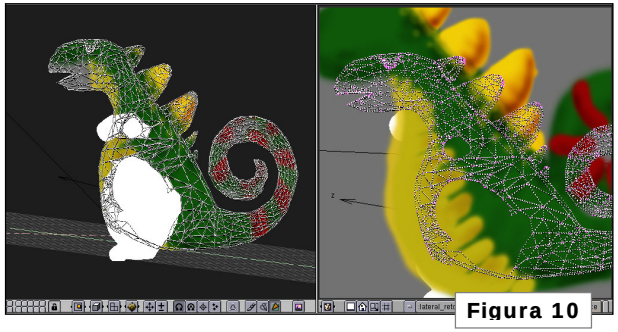


Figura 10

Rotamos el modelo en la vista 3D hasta obtener una posición totalmente lateral, de forma que coincidan las aristas de forma simétrica (tal y como muestra la figura 8). Para ello, tendremos que activar la vista ortogonal. Hecho esto, renderizamos la vista y salvamos con el nombre de *lateral.jpg*. Cargamos el fichero en GIMP y creamos una nueva capa. Pintaremos sobre esta capa, con un color verde la piel del dinosaurio. Nos "saldremos" a propósito fuera del área a colorear, ya que esto nos facilitará el situar la textura más adelante. Crearemos una nueva capa donde dibujaremos de color amarillo los huesos que salen de la espalda. Añadiremos detalles más oscuros jugando con la transparencia del pincel. Puede resultar útil variar la transparencia de la capa para tener una visión general de la zona del dinosaurio sobre la que estamos pintando. Cuando hayamos obtenido algo similar a la figura 9, salvaremos el fichero como *lateral_pintado.jpg*.

Cargamos este fichero en la ventana de imágenes de Blender.

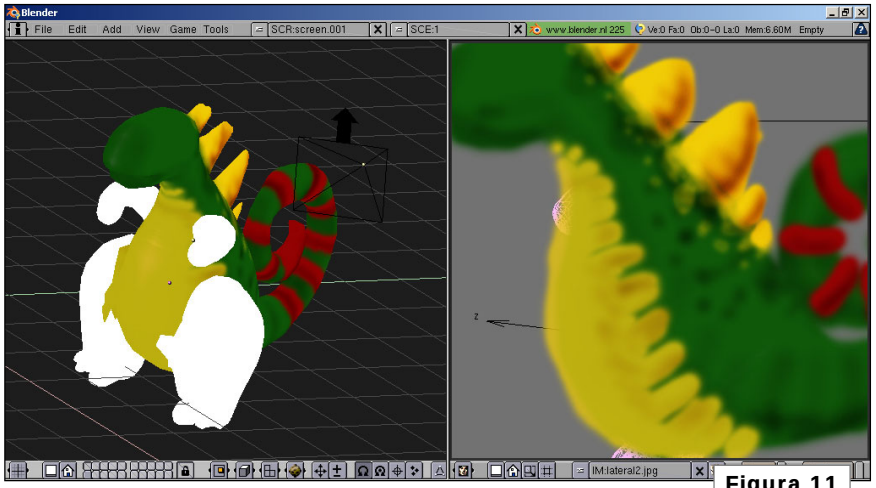


Figura 11

Pinchamos en el botón Load. La imagen aparecerá como fondo de la ventana. Podemos maximizar esta vista (y volver a su estado original) mediante el icono ▣, o pulsando **Control + Flecha Abajo**. En la ventana 3D, con las caras del cuerpo seleccionadas, y la vista que se indicó en la figura 8, pulsamos la tecla "**U**", y seleccionamos **UV Calculation / From Window**. Vemos que el conjunto de caras seleccionadas se han "copiado" en la ventana de la imagen. Tendremos que ajustar los vértices para que se cuadren bien a la textura. Pero antes, activaremos la vista detallada (pinchando en el icono **(c)**, la primera que aparece

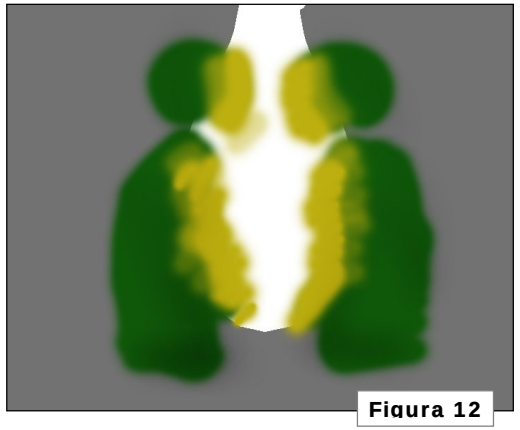


Figura 12

Figura 13

empezando por arriba. El modelo muestra una representación con la textura aplicada, como se muestra en la figura 10.

Maximizamos la ventana derecha, y ajustamos los vértices. Para ello, los seleccionamos todos (tecla **A**). Podemos escalar el modelo con **S**. Recordemos que para limitar el escalado en alguna dirección, pulsaremos **Alt** cuando procedamos al escalado. También podemos rotar los vértices con **R**. Cuando los vértices se ajusten perfectamente a la textura, minimizaremos de nuevo la ventana de imagen y deberíamos obtener una representación similar a la figura 11.

Procedemos de forma similar con las extremidades. Nos aseguraremos de que, al seleccionar las caras, no nos dejemos ninguna que no seleccionáramos en el paso anterior. Situamos el modelo frontalmente y renderizamos la vista.

Coloreamos en GIMP obteniendo una textura similar a la figura 12. Aplicamos la textura frontalmente.

Añadimos una esfera para crear un ojo. Ajustamos el tamaño, la situamos y le aplicamos un material (**F5**). El ajuste de materiales en Blender será visto en detalle en próximas sesiones. De cualquier forma, y para terminar este modelo, daremos el *recetario* para texturizar el ojo. A este material le asociaremos una textura (**F6 + Botón Image**). Activamos las opciones de Texturas y Material, tal y como se muestra en las figuras 13 y 14.

Para situar con más exactitud el ojo con la textura aplicada, activaremos la vista detallada y lo rotaremos hasta conseguir la posición deseada.

Figura 14

Para terminar, y muy importante, añadiremos un material genérico al modelo del dinosaurio, activando el botón de **TexFace**. También activaremos el botón de **VcolPaint** y **VcolLight**. Si entramos en el modo de pintado de vértices, en el menú "*Paint Buttons*" , podremos añadir sombras adicionales al modelo (como en los dedos del modelo, y en las caras de unión entre las extremidades y el cuerpo...). Para ello, deberemos activar el icono **(g)** y el modo de sombreado detallado. En el menú de Paint Buttons, podremos elegir el color con el que pintaremos los vértices del modelo. En zonas donde queremos que el sombreado sea suave, es aconsejable utilizar el pincel de pintado de vértices con el botón Soft pulsado.

Concluiremos la práctica realizando un render sencillo del modelo. Podéis encontrar un entorno donde probar el modelo en el fichero "*escenariorender.blend*". En este escenario está definido un plano (que servirá de suelo) y un par de focos con una configuración básica para realizar las pruebas.

Práctica 5

Animación Básica con Blender

Comenzamos la parte de animación en prácticas viendo la forma de trabajar con curvas IPO en Blender. De ahora en adelante alternaremos sesiones de animación (huesos, restricciones, etc...) con nuevas técnicas de modelado y texturizado.

Vamos a comenzar realizando una animación muy sencilla, manipulando directamente las curvas de animación. Cargamos en Blender el fichero "*dino_prac5.blend*". Seleccionamos el modelo del dinosaurio y vamos a intentar rotarlo respecto del eje Z. Como se puede apreciar, hay un problema con los ojos y los párpados, que no siguen el movimiento del cuerpo. Hay que indicar a Blender la jerarquía existente entre estos objetos; el cuerpo del dinosaurio será el "*objeto padre*" de los ojos y los párpados. Para indicar esto, seleccionamos primero un elemento hijo (por ejemplo, un párpado), y con **Shift** pulsado, seleccionamos después el cuerpo. Hecho esto, hacemos **Control + P** / **Make Parent**. Aparece una línea punteada indicando la jerarquía, tal y como se ve en la figura 1. Procedemos de igual forma con el otro párpado y las esferas de los ojos.

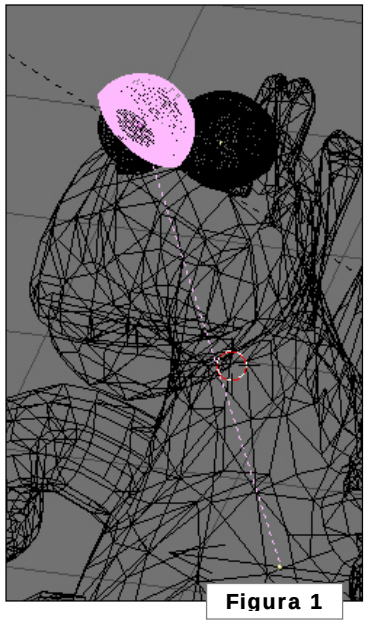


Figura 1

Figura 2

Ahora, si rotamos el cuerpo del dinosaurio, se rotarán con él los ojos y los párpados, pero no al contrario (los padres no heredan posición de los hijos). Vamos a realizar la animación que se puede ver en el fichero "*anim_dino.mpg*". Rotaremos el dinosaurio respecto del eje Y 360º, intentando que el vídeo pueda repetirse indefinidamente (sin cambios de velocidad al empezar y al finalizar la animación).

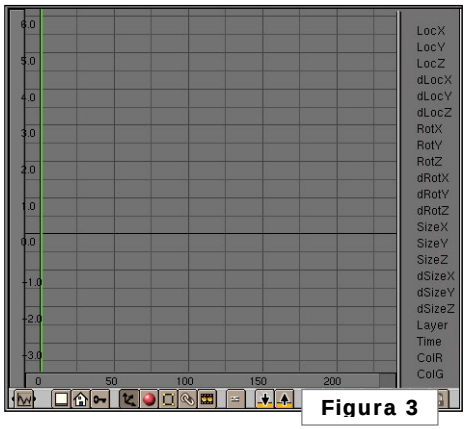


Figura 3

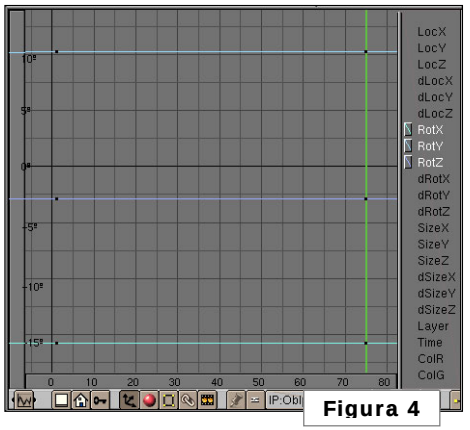


Figura 4

Las curvas de movimiento en Blender nos permiten representar la posición, rotación y tamaño de cada componente (X,Y,Z) de un objeto a lo largo del tiempo. Recordemos que, en animación, el tiempo lo medimos en *frames* por segundo. El número de frame actual en Blender se representa con un botón, en la parte derecha de la cabecera principal, como se muestra en la figura 2. Podemos avanzar el número de frame de 1 en 1 con las flechas **Izquierda** y **Derecha** de los cursores y con las flechas **Arriba** y **Abajo**, de 10 en 10 frames.

Nos situamos en el primer frame de la animación. Cambiamos una ventana al tipo "*Editor de curvas IPO*" . Aparece una imagen como la mostrada en la figura 3. En el eje de abcisas se representan los frames, y en el de ordenadas los valores que pueden tomar los objetos respecto de las magnitudes indicadas en la columna de la derecha (Localización, Rotaciones...). La línea verde vertical (que podemos arrastrar para ver la evolución de nuestra animación), representa el frame actual. De momento no vamos a comentar todas las opciones de esta herramienta, ya que volveremos sobre ella en próximas sesiones. Con el cuerpo del dinosaurio seleccionado, definiremos el **frame 1** como clave (pulsando la tecla **i**). Aparece un menú preguntándonos qué tipo de información vamos a manejar en nuestro frame clavel. Indicaremos que sólo vamos a variar la rotación (**Rot**).

Como queremos que la animación dure 3 segundos (a 25 fps), definiremos otro **frame clave** en el **75**. Al igual que antes, también de tipo **Rot**. Obtendremos una imagen como se muestra en la figura 4. En este tipo de ventana podemos hacer zoom y desplazarla de igual forma que en una ventana 3D. Si al hacer zoom o desplazar nos "perdemos", podemos dejar que Blender ajuste la vista pulsando la tecla **Inicio**.

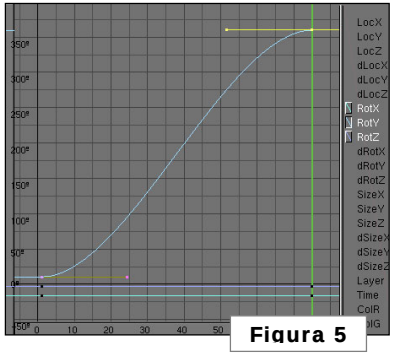


Figura 5

Seleccionamos la curva IPO llamada **RotY**, pasamos al modo de edición de objeto (tecla **TAB**) y ajustamos los puntos de control para obtener una curva similar a la mostrada en la figura 5. Naturalmente, en el eje Y tenemos que llegar al valor 360 (giro completo en el eje Y). Nos ayudaremos del Zoom para situar el valor exactamente en 360.

Podemos ver una previsualización de la animación, si nos situamos en el frame 1 y pulsamos **Alt + A** en cualquier vista 3D. Paramos la animación con **Escape**. Al principio y al final de la animación la rotación es más lenta que en el medio. Como queremos que la animación se pueda repetir en un bucle sin que se noten los cortes, ajustaremos la pendiente de la curva IPO en los puntos extremos, desplazando los puntos de tangente. Debemos obtener un resultado como el que se muestra en la figura 6.

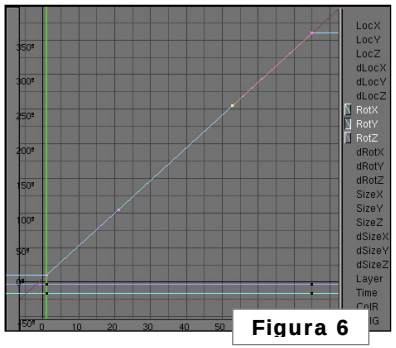


Figura 6

Si previsualizamos la animación, comprobamos que ahora el resultado es el que esperamos. Vamos a generar el vídeo AVI. Disminuimos la resolución de salida a **320x240**. Elegimos como formato de salida **AVI Jpeg**. Ajustamos el rango de frames que queremos renderizar con los botones **Sta: 1** y **End: 75**. El directorio donde se va a generar el vídeo por defecto es /**render** (indicado en el apartado **Pics**, arriba a la izquierda). Hecho esto, pinchamos en **ANIM** para iniciar el proceso y esperamos un rato ☺.

En la segunda parte de la práctica, también emplearemos curvas IPO, pero no generando el movimiento directamente sobre ellas, sino utilizándolas únicamente para dar los ajustes finales. Cargamos el fichero "*oreto_prac5.blend*". Moveremos la cámara por la escena, insertando frames clave en los puntos donde haya que realizar un cambio de dirección. Blender se encargará de interpolar las posiciones intermedias. El vídeo que debemos obtener como resultado (*anim_oreto.mpg*) muestra el proceso final (con post-producción, que veremos en prácticas sucesivas). Debemos

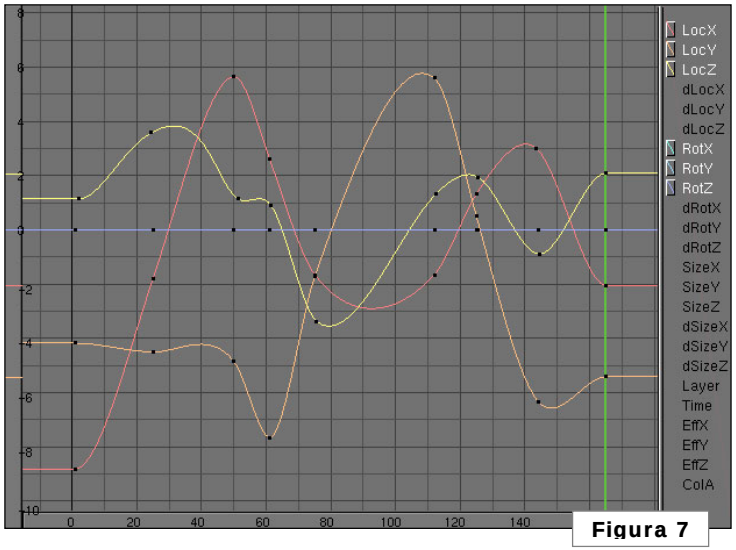


Figura 7

ocuparnos de obtener un movimiento de cámara similar. Nos centraremos en colocar la cámara en las posiciones clave (respecto de la localización y la rotación). Para facilitar que la cámara apunte siempre al logotipo, añadiremos un objeto de tipo Empty en el centro del logotipo (**Espacio + Add + Empty**). Seleccionamos la cámara primero y luego con **Shift** pulsado, el Empty recién creado. Pulsamos **Control + T (Make Track)**. Corregimos el cambio de orientación con **Alt + R (Clear Rotation)**.

En la figura 7 tenemos las curvas IPO asociadas a la cámara. Podemos ver las posiciones donde se han insertado frames clave (en el 1, 25, 50, 61, 75, ..., 165). Al igual que en la primera parte, obtenemos una previsualización de la animación con **Alt+A**. Si los resultados son correctos, renderizamos con los parámetros vistos anteriormente.

Práctica 6

Animación Jerárquica y Composición

En esta sesión aprenderemos los conceptos básicos de animación jerárquica en Blender, y realizaremos una composición sencilla de video.

Comenzaremos modelando el objeto que se muestra en la figura 1. Todos los componentes del modelo deben ser independientes (añadiremos a la escena 5 cilindros, rotando y escalando hasta conseguir la configuración mostrada).

Renombraremos los objetos que forman la escena, con el fin de poder identificarlos más fácilmente, tal y como se muestra en la figura 2. Para ello, seleccionaremos en el modo de edición de objeto cada uno de ellos, y teclearemos el nombre en la sección «**OB:** » de los botones de edición (**F9**).

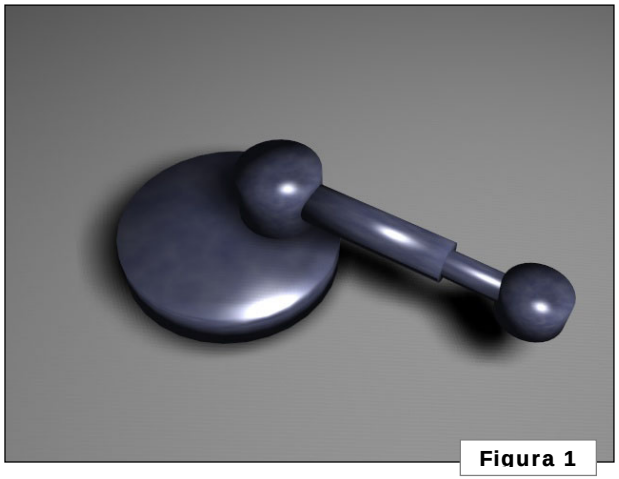


Figura 1

Podemos indicar a Blender que muestre el nombre de los objetos si activamos el botón de la parte inferior izquierda de esta ventana: **Name**. Establecemos las relaciones de jerarquía entre PistonA1 - PistonA2 y PistonB1 - PistonB2. Queremos que, cuando movamos las bases de los pistones (A1 y B1), los extremos tengan el mismo movimiento. Por tanto, en la jerarquía tendremos que PistonA2 es hijo de PistonA1. Para indicar esta jerarquía a Blender, seleccionamos primero el elemento hijo (PistonA2), y con **Shift** pulsado, seleccionamos el padre (PistonA1). Pulsamos **Control + P (Make Parent)**. Ahora, si rotamos PistonA1, el giro también lo sufrirá PistonA2, pero no al revés. Realizamos la misma operación con la otra parte del pistón (B1, B2). Si nos equivocamos, podemos eliminar la relación de parentesco pulsando **Alt + P**.

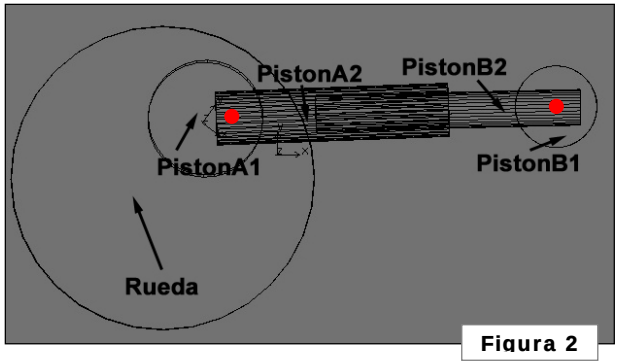


Figura 2

Cambiaremos los centros de los objetos PistonA2 y PistonB2 para que las rotaciones se hagan respecto del extremo (en vez de estar situado en el centro geométrico del objeto). Para ello, situaremos el puntero 3D (círculo de color rojo que movemos con **BIR**), a la posición 3D donde queramos situar el nuevo centro y en los botones de edición (**F9**) pincharemos en **Centre Cursor**. Colocaremos los centros en los puntos rojos que se muestran en la figura 2.

Pasamos a establecer una relación de dependencia entre objetos. Asociaremos el movimiento de los dos pistones, haciendo que se sigan el "rastro". Esta técnica ya la hemos utilizado en sesiones anteriores, cuando incluíamos un objeto Empty para situar la cámara. El primer objeto que seleccionamos es el "*dependiente*" (que seguirá el movimiento del segundo; el "*principal*"). Seleccionamos primero **PistonA1** y luego (con **Shift** pulsado) **PistonB1** (en este orden) y creamos el track (**Control + T**). Repetimos la operación, pero ahora seleccionando primero **PistonB1** y luego **PistonA1**. En caso de que se produzcan efectos raros en la rotación de los objetos, pulsamos **Alt + R** y ajustamos de nuevo la rotación de PistonA1 y PistonB1.

Probamos a mover PistonA1. Si tenemos que el movimiento lo siguen perfectamente el resto de piezas, hemos terminado este paso. En caso contrario, tendremos que ajustar en los botones de animación (**F7**) el eje con el que vamos a apuntar al otro objeto (al otro piston). Habitualmente, habrá que cambiar de **Track Y** a **Track X**. Podría hacer falta cambiar también el eje que apunta "arriba" (vector Up). Podemos ver los ejes de cada objeto si en los botones de edición pinchamos en el botón **Axis** (encima de **Name**).

Activaremos el botón de **PowerTrack** (en los botones de animación; **F7**) al pistón A1, para quitarle la rotación (cuando el objeto "Rueda" gire, no se tendrá en cuenta esta rotación, sólo la posición).

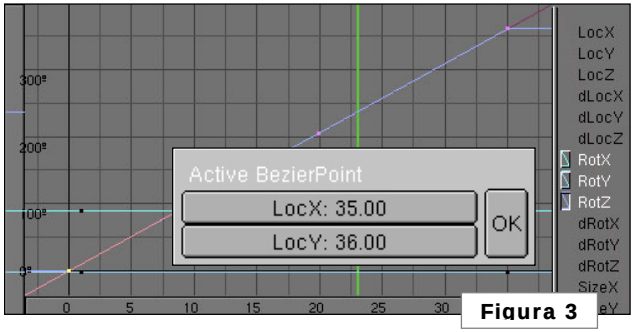


Figura 3

Añadimos un Empty en el centro geométrico de PistonA1, y hacemos padre (este Empty) de PistonA1. Hecho esto, hacemos hijo al Empty de Rueda. Si rotamos Rueda respecto del eje Z, el mecanismo general debería funcionar.

Pasamos a hacer una rotación de 360º de la rueda. Procederemos de forma similar a como se explicó en la práctica anterior con el dinosaurio. Podemos especificar un valor numérico si pulsamos la tecla **N** (tal y como muestra la figura 3). Si ponemos el valor "36" como **Loc Y** de la gráfica "**Rot Z**", tendremos un giro de 360º (suponiendo que el punto origen esté en el 0). Si no está en el 0, sumaremos 36 al valor que tengamos en el **LocY**. En el ejemplo, se ha utilizado un ciclo de 35 frames.

Pasamos a renderizar 2 animaciones con distintos materiales (resolución de 320 x 240). Añadimos un plano debajo de la escena, para que sirva como "suelo" y recoja las sombras arrojadas. Por ejemplo, podemos hacer un render utilizando la técnica "*Toon Shading*"; con el aspecto de los dibujos animados. Para ello, en el material (**F5**) activamos la opción "shadeless", para que no arroje sombras. En la ventana de render (**F10**), activamos el botón **Edge**, y en **Edge Settings** al parámetro **Eint** (Intensidad de la arista), le damos un valor alto (100).

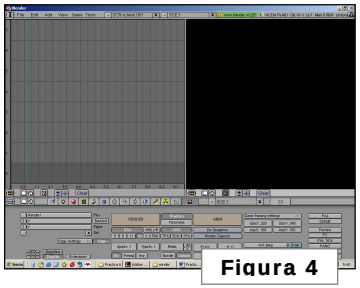


Figura 4

En definitiva; hemos obtenido distintos vídeos con la animación de nuestro pistón. Pasamos al siguiente paso; componerla en Blender. Para ello, abrimos una nueva sesión de Blender y dividimos la pantalla en 2 partes. Cambiamos las dos ventanas a tipo . En una de ellas, activamos el botón . Debemos obtener una representación como muestra la figura 4. En la parte derecha obtendremos una previsualización de la composición que estamos realizando, y en la izquierda iremos añadiendo los vídeos, imágenes, transiciones y efectos. Para añadir un vídeo o imagen (en el vídeo de ejemplo se ha utilizado una para el texto), pulsamos **Shift + A**. Podemos duplicar cualquier elemento que aparezca en esta ventana pulsando **Alt+D**. Pinchando en los triángulos izquierdo y derecho de un vídeo, se puede "escalar". Añadimos los 2 vídeos que hemos realizado, los colocamos uno encima del otro, tal y como muestra la figura 5. Con los dos seleccionados, podemos añadir un efecto. Podemos experimentar con todas las transiciones que incorpora blender, e incluso descargar plugins para añadir nuevas funcionalidades a la aplicación.

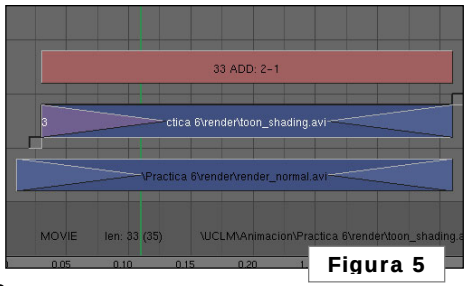


Figura 5

Con un efecto seleccionado, si pulsamos la tecla **c** podemos cambiar el tipo de efecto, o el sentido de aplicación del efecto (A→B o B→ A). En algunos efectos, como las transiciones Cross, GammaCross, etc... el orden importa. Con la tecla **N** podemos renombrar los elementos de nuestra composición.

Una vez que tenemos un bloque de nuestra animación definido, si seleccionamos todos los elementos que intervienen (vídeos, imágenes, efectos, etc...) y pulsamos la tecla **M**, podemos agruparlos en una única Metatransición. Con **Alt+M** podemos deshacer este paso.

Una vez tenemos la secuencia creada, la generaremos en un AVI, activando el el botón «do sequence» de los botones de renderizado (**F10**) y pinchando luego en Anim. Utilizando las herramientas descritas en esta práctica, intentar conseguir un resultado similar al que se puede ver en el fichero *resultado_p6.mpg*.

Práctica 7

Introducción a los esqueletos

Con esta práctica nos introduciremos en el uso de esqueletos (*armatures*) en Blender. Partiendo de un modelo sencillo (un brazo robótico), aprenderemos a asociarle, a cada sección, un hueso del esqueleto. Añadiremos restricciones a los elementos y finalmente generaremos un movimiento del robot mediante cinemática inversa.

Comenzaremos modelando las diferentes piezas que formarán nuestro brazo. Debemos obtener una geometría similar a la mostrada en la figura 1. Antes de ponernos manos a la obra, debemos tener en mente que al modelo deberá asociársele un esqueleto. Resultará mucho más sencillo realizar esta operación (y el propio modelado), si en la construcción del objeto empleamos una postura "cómoda". Un ejemplo de construcción del modelo se muestra en la figura 2. Así, el esqueleto se situará de forma sencilla a lo largo del eje X, siguiendo la geometría del brazo.

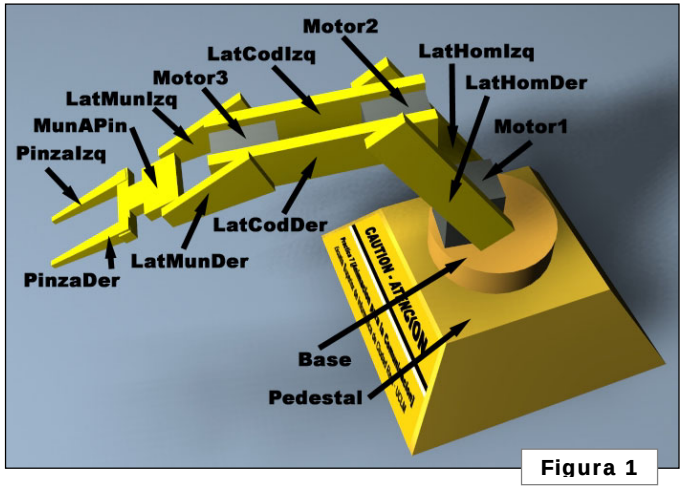


Figura 1

Recordemos que se pueden duplicar objetos con **Shift + D**. Será útil definir todos los componentes del robot como mallas independientes. Para ello, deberemos salirnos del modo de edición de vértices cada vez que añadamos una parte del robot a la escena. Nombraremos cada componente del robot como se muestra en la figura 1. Será importante tener claros los nombres asignados a cada parte a la hora de gestionar la jerarquía y las asignaciones de parentesco a cada hueso del esqueleto. Por tanto, se recomienda seguir el mismo convenio de nombrado que se utiliza en este documento.

Una vez modelado el robot, y renombradas las piezas, añadiremos el esqueleto. Para ello, pulsaremos **Barra Espaciadora + ADD + Armature**. Construiremos un esqueleto formado por 4 huesos, centrado en el interior del brazo robótico, tal y como se muestra en la figura 3. Los tres primeros están ajustados a las articulaciones del brazo del robot (hombro, codo y muñeca). La última, muy pequeña, se utilizará para gestionar la cinemática inversa del modelo. Una vez creado el

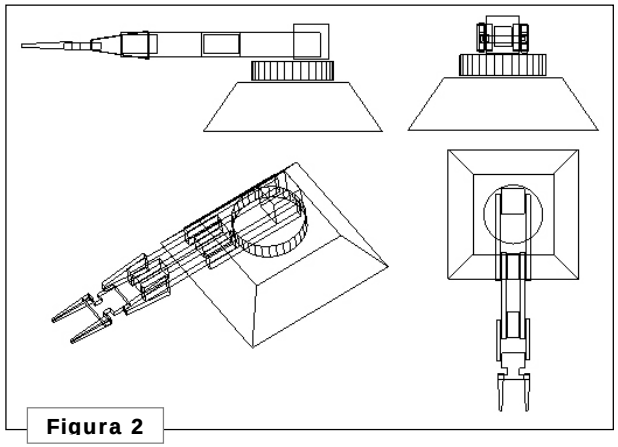


Figura 2

esqueleto, entramos en el modo de edición del objeto (**TAB**). El color rosa del esqueleto seleccionado será más intenso. Seleccionamos todos los huesos del modelo (tecla **A**), cambiando su color a amarillo. Vamos a los botones de edición (**F9**) y cambiamos el nombre de los huesos como se muestra en la figura 4. Al igual que antes, se recomienda seguir el convenio en el nombrado de los

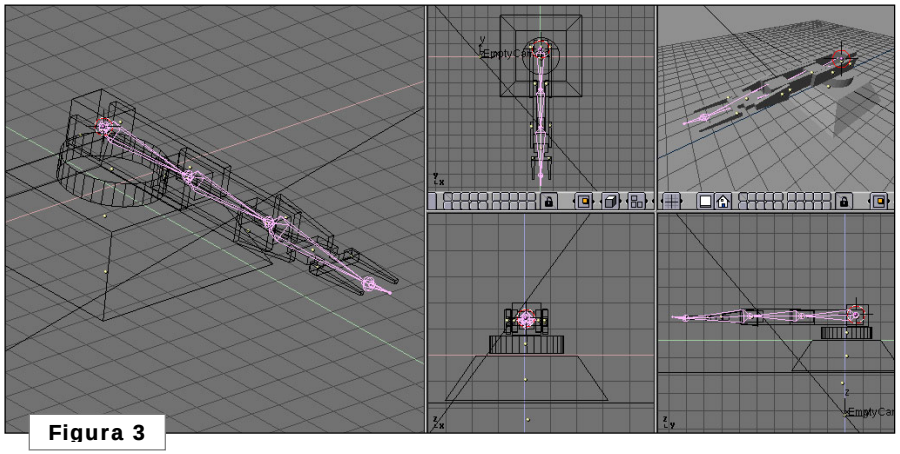


Figura 3

huesos. Pasamos a definir las jerarquías entre elementos, y a conectar las diferentes partes del robot con el esqueleto que hemos creado. Todo el proceso que vamos a seguir a continuación, está resumido en el esquema de la figura 5. Las relaciones de parentesco ("*es padre de*"), están representadas por una flecha que va del padre al hijo. Los huesos del esqueleto están representados en color azul. Los nodos hijo de la jerarquía, están representados por un rectángulo con las esquinas sin redondear. Los elementos auxiliares (*empty*) que se han utilizado, aparecen en color naranja. Por último, las restricciones entre elementos se representan con una flecha punteada, con una circunferencia negra en el origen.

Recordemos que para establecer una relación de parentesco entre elementos, seleccionaremos siempre **primero el** elemento **hijo**, y **después**, con **Shift** pulsado, el **padre**. Pulsaremos **Control + P + Make Parent**. En caso de equivocarnos, podremos eliminar el parentesco con **Alt + P**. De esta forma, seleccionaremos, por ejemplo, primero el elemento Base, y después Pedestal y realizaremos la jerarquía. Procederemos de igual modo con Motor1 y Base, el Esqueleto (completo) y la Base, y MunAPin (Muñeca a Pinza) con las dos partes de la pinza.
Las relaciones de parentesco con los huesos se realizan de forma similar. Primero seleccionamos el hijo, y después el esqueleto (completo). Pulsamos **Control + P + Use Bone** y seleccionamos el hueso que corresponda en cada caso.

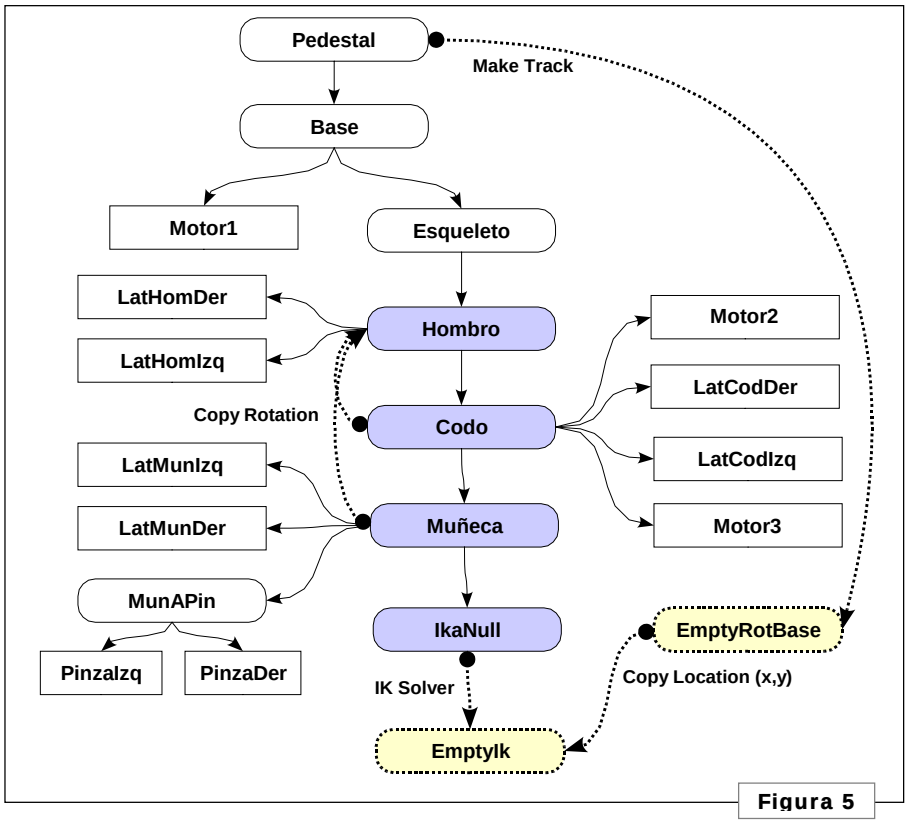


Figura 5

Añadiremos un objeto Empty a la escena que nombraremos *EmptyIk*. Este objeto nos servirá para situar el extremo del robot y, por cinemática inversa, Blender calculará la rotación necesaria para cada articulación del robot. Situaremos el puntero 3D cerca del hueso *IkaNull* y situaremos ahí el nuevo objeto Empty. Entraremos en el *modo de pose* del esqueleto (que veremos con más profundidad en próximas sesiones), que entre otras cosas nos permite seleccionar individualmente cada hueso. Para cambiar a modo de pose, con el esqueleto seleccionado (en color rosa claro), pulsaremos **Control + Tabulador**. También podemos acceder a este modo pinchando en el icono de la cabecera de cualquier ventana 3D. El esqueleto pasará a color azul claro.

Figura 4

En este modo, seleccionaremos el último hueso (*IkaNull*), que pasará a color verde claro. Vamos al menú de restricciones, pinchando en el botón . Añadimos una restricción nueva (**ADD**). El tipo de restricción será **IK Solver**, sobre el nuevo Empty (tecleamos en el campo **OB: EmptyIk**).
Hecho esto, podemos mover el EmptyIk por

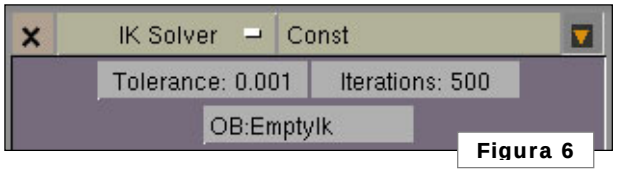


Figura 6

la escena, y el brazo seguirá el movimiento del mismo. Sin embargo, vemos que el brazo no se comporta correctamente. Las articulaciones no realizan los giros de forma realista, debido a que no tienen restricciones de giro aplicadas.

Seleccionamos, en modo pose, el hueso "*Codo*" y añadimos una restricción para copiar la rotación del hueso "*Hombro*". Con esto, conseguiremos *parcialmente* que los tres huesos giren a la vez. Como se muestra en la figura 7, tendremos que definir el objeto que contiene al hueso llamado "*Hombro*" (en nuestro caso, "*Esqueleto*"). Repetiremos la misma operación con el hueso "*Muñeca*".

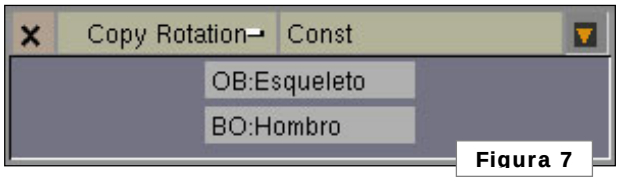


Figura 7

Crearemos un nuevo objeto vacío, al que llamaremos EmptyRotBase. Nos servirá para definir la rotación que se tiene que aplicar al objeto Base (recordemos que es padre de casi toda nuestra jerarquía), cuando movamos EmptyIk. De esta forma, el robot siempre apuntará al objeto EmptyIk, y no tendremos ninguna rotación extraña en las articulaciones. Situaremos el nuevo Empty, como se muestra en la figura 8, a la misma altura que la base (para que al hacer el Track entre ellos, la Base no se "incline" hacia ningún lado; únicamente rote respecto del eje Z).
La idea es hacer que este nuevo Empty sirva de "sombra" al EmptyIk con el que controlamos el movimiento del brazo. Para conseguir este efecto de "sombra", añadiremos una restricción de copiar la localización del EmptyIk, como se muestra en la figura 9. La localización se copiará respecto del eje X e Y. La altura en el eje Z la mantendremos fija.

Hecho esto, añadiremos un seguimiento (Track) de la base respecto del Empty. Para ello, recordemos que habrá que seleccionar primero el objeto dependiente, y después el principal, pulsando **Control + T**. Si hemos seguido los pasos correctamente, podemos mover el *EmptyIk* y el resto de la geometría del robot seguirá el movimiento.

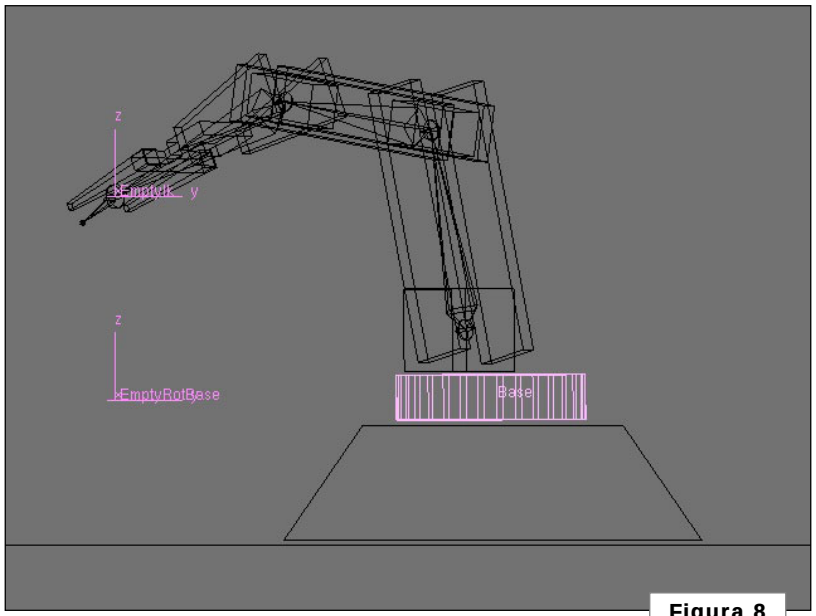


Figura 8

Para animar el modelo, bastará con insertar frames clave que guarden la posición (*Loc*) del objeto *EmptyIk* en el tiempo (**Control + I**). La interpolación del movimiento de este objeto nos producirá (por cinemática inversa) el movimiento en el resto de articulaciones. Se puede ver el modo de trabajar de la cinemática inversa en el fichero esqueleto_p7.mpg.

Figura 9

Añadiremos luces y texturas, e intentaremos conseguir un movimiento similar al mostrado en el vídeo resultado_p7.mpg. Este fichero tiene una curva IPO asociada que puede verse en la figura 10. Es muy importante finalizar correctamente esta práctica para concluir con éxito futuras sesiones.

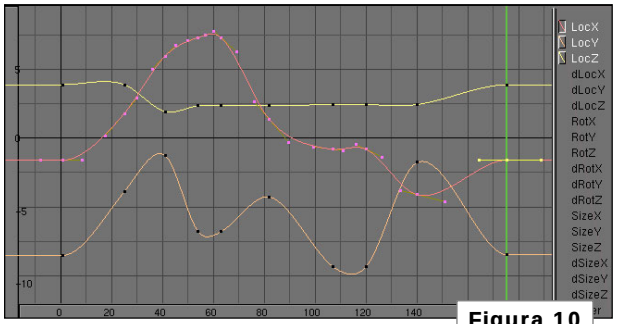


Figura 10

Práctica 8

Renderizado Realista

Completaremos los conceptos vistos en teoría sobre Radiosidad y Trazado de Rayos en esta sesión. Utilizaremos las opciones de radiosidad en Blender, junto con los mapas de entorno para simular reflexiones. También renderizaremos una escena en PovRay mediante el plugin *PovAnim*, cambiando manualmente los parámetros de materiales y acabado final.

Comenzaremos la primera parte de la práctica construyendo el escenario que puede verse en la figura 1. La iluminación de la escena vendrá dada por las áreas de luz de las superficies superiores de la habitación. Recordemos que en Radiosidad, la **iluminación** vendrá generada por **superficies**. De esta forma, crearemos planos a los que asignaremos un material que emita luz.

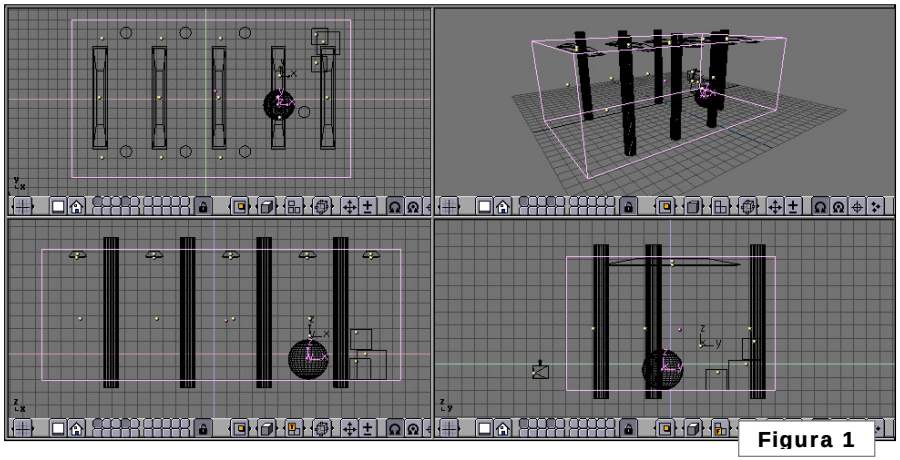


Figura 1

Es muy importante la dirección del vector normal ya que Blender "lanzará" la luz en ese sentido. Por tanto, tendremos que asegurarnos que el vector normal de cada foco apunta "*hacia el suelo*", tal y como muestra la figura 2. Para comprobar que es así, seleccionamos cada "foco", y en modo de edición de vértices, accedemos a los botones de edición (**F9**) y activamos el botón **Draw Normals**. Podemos ajustar el tamaño de representación del vector normal en **Nsize**. En caso de que la normal esté invertida, podemos ajustarla con el botón **Flip Normals** o rotando la cara.

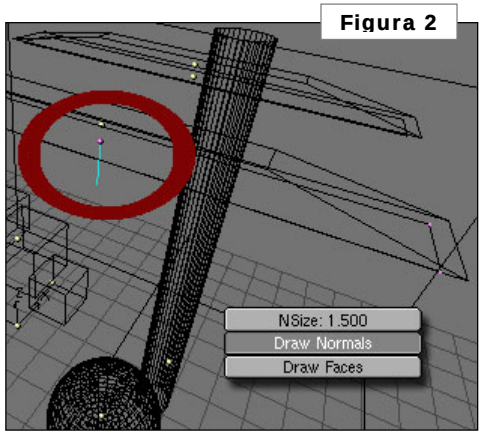


Figura 2

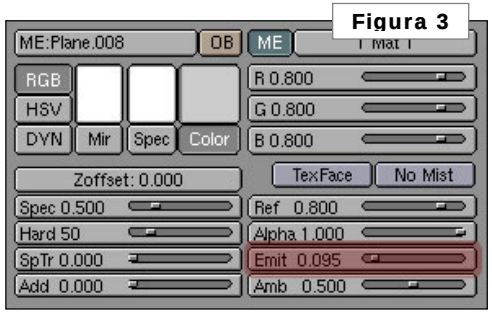


Figura 3

Los elementos que emiten luz tendrán el campo **Emit** del material con un valor mayor que 0. Los focos de la práctica tienen un valor de 0.095 (ver figura 3). A la caja de color rojo se le dio un valor de 0.020.

A la esfera, de momento, le asignaremos un material azul por defecto. El mapa de entorno se calculará después de la radiosidad.

Accedemos al menú de radiosidad con el botón . Seleccionamos todas las mallas que forman nuestra escena y pinchamos en el botón **Collect Meshes**. Aparecen en la vista de la cámara unos cuadrados blancos y azules. Los azules representan el tamaño mínimo y máximo de los "*Elementos*" (recordemos que cada *Parche* está formado por un conjunto de *Elementos*), y los blancos representan los "*Parches*".

Hecho esto, pinchamos en **Limit Subdivide**, seguido de **Subdiv Shoot Element** y **Subdiv Shoot Patch**. Finalmente pinchamos en el botón **Go**. De esta forma comienza el proceso de cálculo de la solución de radiosidad. Podemos parar el proceso en cualquier momento pulsando **Escape**, quedándonos con la aproximación que se ha conseguido hasta ese instante. Ajustaremos los valores para obtener mejor precisión (como muestra la Figura 4). Para ello, primero liberaremos la solución anterior (pinchar en el botón **Free Radio Data**).

El tamaño de los parches (**PaMax** - **PaMin**) determina el detalle final (cuanto más pequeño sea, más detallado será el resultado final), pero incrementamos el tiempo de cál-

culo. De forma similar ocurre con el tamaño de los elementos (**ElMax** – **ElMin**). En **Max Iterations** indicamos el número de pasadas que Blender hará en el bucle de Radiosidad. Un valor 0 indica que haga las que estime necesarias. **MaxEl** indica el número máximo de elementos para la escena. **Hemires** es el tamaño del hemicubo (técnica utilizada para calcular el parámetro H visto en teoría, de cálculo de visiblidad).

Una vez terminado el proceso de cálculo, podemos añadir la nueva malla calculada reemplazando las creadas anteriormente (**Replace Meshes**) o añadirla como nueva a la escena (**Add new Meshes**). Elegiremos esta segunda opción. Antes de deseleccionar la malla, la moveremos a otra capa. Las capas en Blender se utilizan de forma similar a como se hace en cualquier otro programa de diseño. Pueden resultar muy útiles cuando la complejidad de la escena crece. Incluso son imprescindibles para algunas operaciones (como en los mapas de entorno). Las capas activas (visibles) en cada momento pueden verse en la cabecera:

En este caso, sólo la primera capa está activa. Podemos seleccionar múltiples capas como activas pinchando sobre ellas con la tecla **Shift** pulsada.

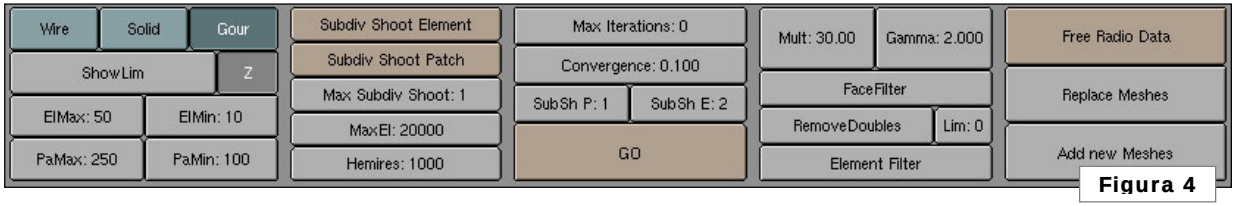


Figura 4

Movemos los objetos seleccionados (en nuestro caso, la nueva malla) a una nueva capa pulsando la tecla **M**. Nos aparecerá una ventana emergente donde nos pedirá la capa destino del movimiento:

Movemos la malla de radiosidad a la segunda capa y liberamos la memoria ocupada por estos datos (**Free Radio Data**).

Activamos únicamente la segunda capa y renderizamos. Tenemos que obtener una imagen similar a la figura 5.

Vamos a añadir el mapa de entorno a la esfera, para que refleje el resto de objetos de la habitación. Para ello, primero separaremos la malla del resto, moviéndola a la capa 3. En modo de edición de vértices, seleccionamos uno de los que forman la esfera. Pulsamos la tecla **L** y seleccionamos, de esta forma, todos los que están enlazados con ese directamente (los que forman la esfera). Separamos este conjunto de vértices como un objeto nuevo pulsando **P**. Movemos la esfera a la tercera capa (tecla **M** con la esfera seleccionada).

Hacemos las dos capas visibles a la vez.

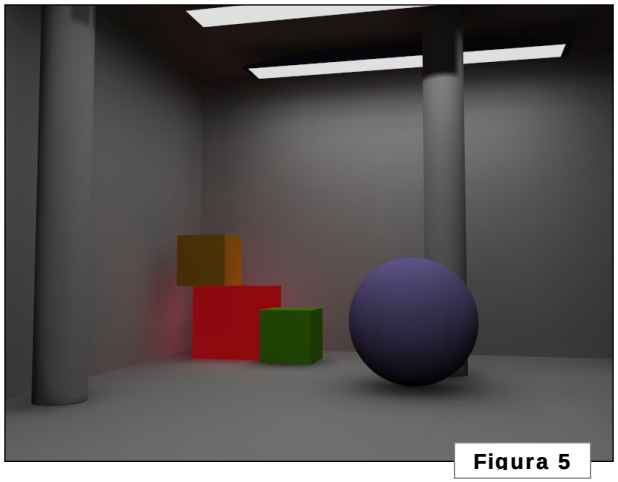


Figura 5

Añadimos un Empty en la capa 2, situado en el centro de la esfera, al que llamaremos "*EmptyEntorno*". Nos servirá para calcular el mapa de entorno cúbico, como si estuviéramos situados en ese punto del espacio.

Al material azul de la esfera le asociamos una textura (F6). Pinchamos en el botón de **EnvMap**. Los botones azules (**Static**, **Anim** y **Load**) indican si el mapa de entorno se va a calcular sólo una vez (en caso de que los objetos del mundo no se muevan), **Anim** (en caso de que haya desplazamiento de los objetos) o **Load** (para cargar el mapa de entorno precalculado en una imagen). **Free data** sirve para forzar a que se recalcule el mapa. En **Ob:** indicaremos el objeto que nos servirá de origen para calcular el mapa. En nuestro caso será **EmptyEntorno**. Con **CubeRes** indicamos la resolución, en píxeles, de cada pared del cubo. Al lado, tal y como muestra la Figura 6, nos indica qué capa(s) no queremos renderizar. Es muy importante

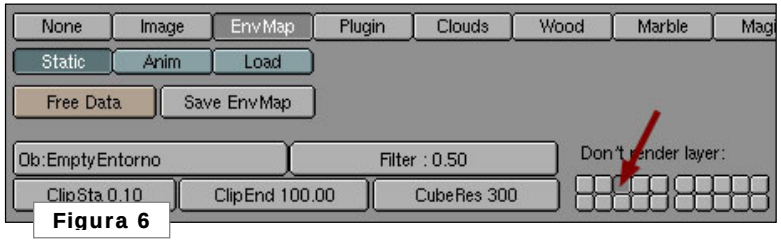


Figura 6

no renderizar las capas donde se encuentren los objetos a los que vamos a aplicar el mapa de entorno. De otra forma, el mapa obtenido renderizaría el interior del objeto (en nuestro caso la esfera). Así, indicamos que la capa 3 no la vamos a tener en cuenta. Finalmente es posible que tengamos que ajustar los parámetros de brillo y contraste del mapa para obtener un resultado realista. En el menú de materiales (**F5**) habrá que indicar que queremos usar el vector de reflexión como coordenadas de textura (**Refl**).

El mapa de entorno para la escena es el mostrado en la figura 7. El resultado del render final debe ser similar a la figura 8.

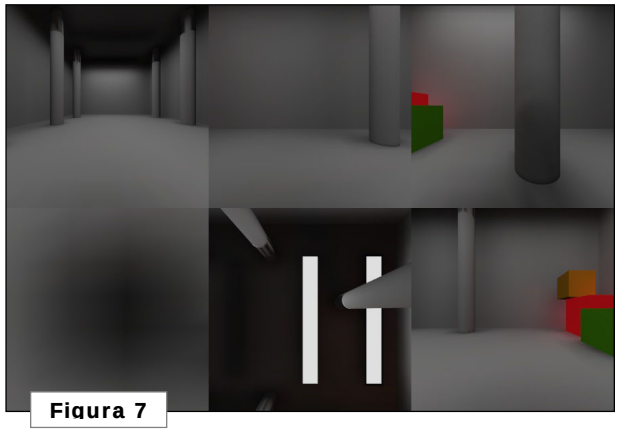


Figura 7

Exportando Blender :: PovRay

En esta segunda parte de la práctica, renderizaremos una escena realizada en Blender con un motor de render externo: PovRay. Comenzamos abriendo el fichero logouclm.blend. Añadimos las letras "UCLM" con Wings3D. De nuevo en blender, abrimos una ventana de tipo Texto y cargamos el fichero "*lanc_povanim225i_05.py*". Debemos tener en el mismo directorio donde se encuentre el fichero .blend, el fichero de Python compilado "*povanim225i_05.pyo*". De igual forma, tenemos que tener instalado el intérprete de Python en nuestra máquina. Hecho esto, ejecutamos el script (**Alt+P** en la ventana de texto).

Únicamente modificaremos la opción referente al tipo de salida (para evitar la compatibilidad con MegaPov, y obtener una salida de PovRay estándar). Para ello, pulsaremos

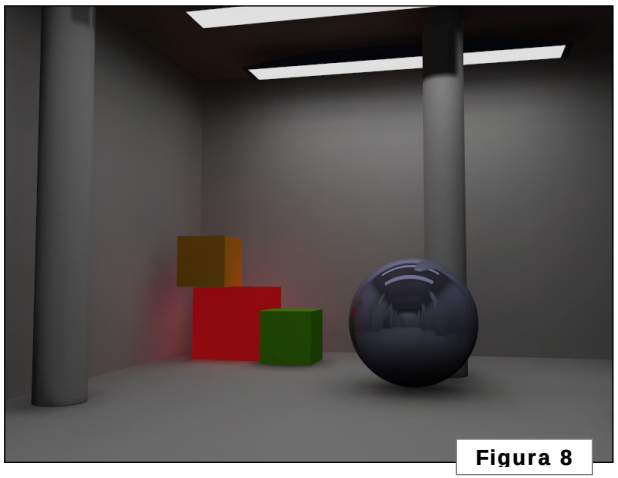


Figura 8

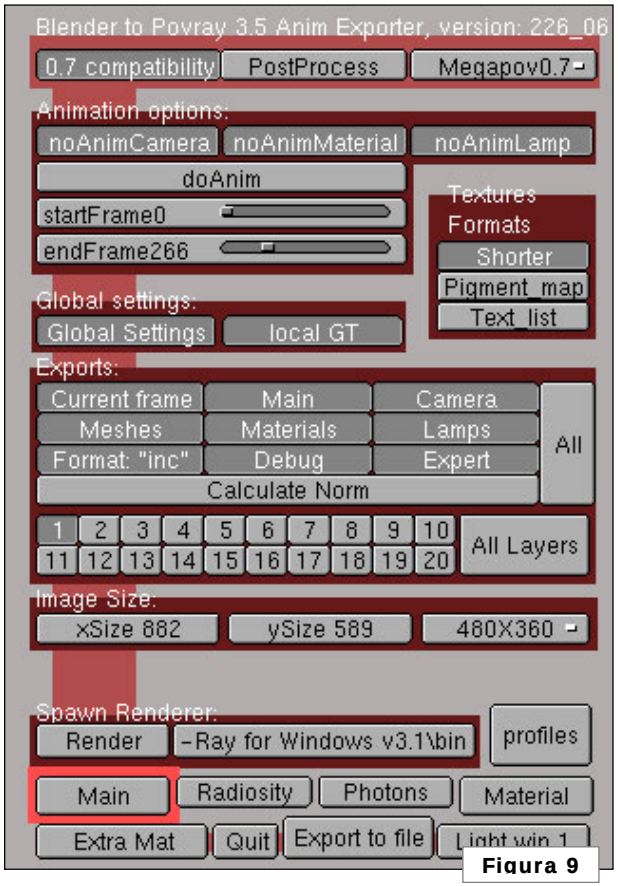


Figura 9

en el botón **0.7 compatibility** para desactivarlo, cambiamos la resolución de salida a **800x600** y seguidamente pinchamos en **Export to file**. En la página web oficial del script podemos encontrar una descripción completa de todos los parámetros que se pueden exportar.

En el mismo directorio de trabajo, se habrá creado un directorio llamado "povanim".

Este directorio contiene varios ficheros llamados:

- **cam[*logouclm*].inc:** Contiene la descripción de la posición de la cámara.
- **ini[*logouclm*].ini:** El fichero con todos los parámetros de renderizado. Será el que lanzaremos en PovRay.
- **lamp[*logouclm*].inc:** Fuentes de luz.
- **main[logouclm].pov:** Descripción principal de la escena. Carga todos los ficheros .inc.
- **mat[*logouclm*].inc:** Materiales de los objetos que forman la escena.
- **mesh[*logouclm*].inc:** Define los vértices que forma la geometría de la escena. Si la escena contiene objetos complejos (como nuestro ejemplo), cargará los conjuntos de triángulos definidos en la subcarpeta "mesh".

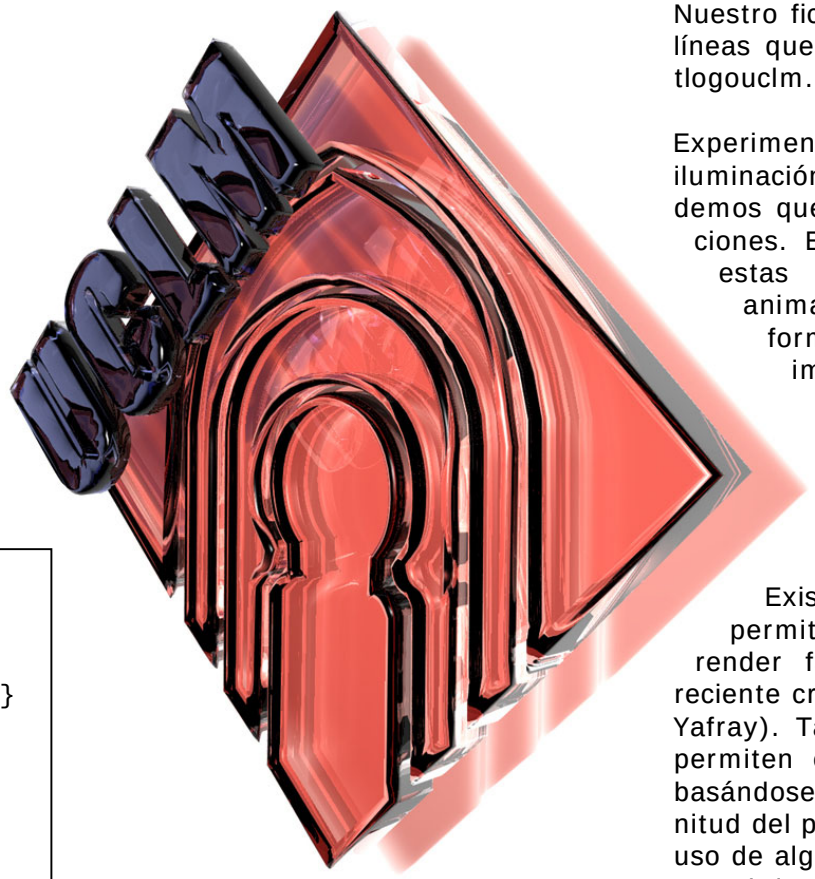
Probaremos a renderizar *inilogouclm.ini*. Probablemente habrá que corregir algún error mínimo en la exportación de materiales (bump mapping).

```
#include "glass.inc"
#include "colors.inc"

// Material para el logo
#declare UCLMRojo_pig=pigment{color Clear }
#declare UCLMRojo_fsh=finish {F_Glass2}

// Material para las letras UCLM
#declare Material_pig=pigment{color Clear}
#declare Material_fsh=finish {F_Glass8}
```

matlogouclm.ini

```
Width=800
Height=600
...
Antialias_Depth=4
Antialias=On
Antialias_Threshold=0.02
```

inilogouclm.ini



Hecho esto, procedemos a personalizar la escena con nuestros propios valores de materiales y opciones de renderizado.

En el fichero inilogouclm.ini, ajustaremos la resolución del render y el valor de antialiasing (cuanto menor sea, más calidad de render final).

Nuestro fichero de materiales contendría las líneas que pueden verse en el listado "matlogouclm.ini".

Experimentar con valores de materiales, iluminación y tipos de renderizado. Recordemos que PovRay permite generar animaciones. El plugin que hemos utilizado en estas prácticas permite exportar las animaciones generadas en Blender. El formato de salida será un conjunto de imágenes (TGA o BMP típicamente). Tendremos que utilizar algún programa para generar el vídeo en estos casos. Hay multitud de herramientas por línea de comandos que realizan esta tarea.

Existen plugins para Blender que permiten exportar a otros motores de render famosos (como Lightflow), o de reciente creación y amplia aceptación (como Yafray). También existen herramientas que permiten el render distribuido en PovRay, basándose en PVM o PMI. Según la magnitud del proyecto puede ser necesario hacer uso de alguna herramienta de este tipo para que el tiempo de render sea manejable.

Práctica 9

Modelado por Rotoscopia

En esta sesión modelaremos un personaje utilizando superficies de subdivisión mediante la técnica de rotoscopia. Esta técnica, muy utilizada en el ámbito de la animación de personajes por computador, se basa en moldear el personaje 3D utilizando, al menos, dos vistas realizadas a mano.

Cargaremos los bocetos del personaje que pueden verse en la figura 1. Dividimos la pantalla en 3 ventanas. A la izquierda, cargaremos la imagen de fondo "*frente.jpg*", y en el centro "*lateral.jpg*". La parte de la derecha la dejaremos para ir rotando el modelo, seleccionar vértices, etc... (ver figura 2). La imagen de frente se cargará en la vista frontal (**tecla 1** del teclado numérico) y la vista lateral con la **tecla 3** del teclado numérico.

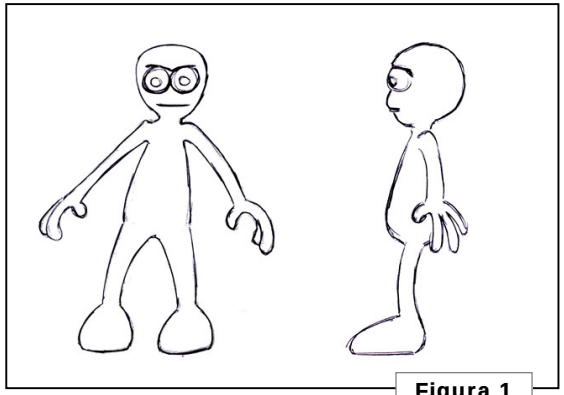


Figura 1

Añadimos un cubo a la escena y lo ajustamos para comenzar a modelar un pie del personaje, como se muestra en la figura 2. Activaremos las opciones de subdivisión de la malla (botones de edición, **F9**).

El valor de la izquierda (2 en nuestro caso), es el número de subdivisiones que hace Blender para mostrar la malla en modo de trabajo. Únicamente sirve para hacernos una idea de cómo va a quedar. El valor de la derecha (más importante), es el número de subdivisiones que se harán a la hora de renderizar. De esta forma, ajustaremos el valor de la derecha para que sea mayor que el de la izquierda (y éste, según sea nuestro procesador).

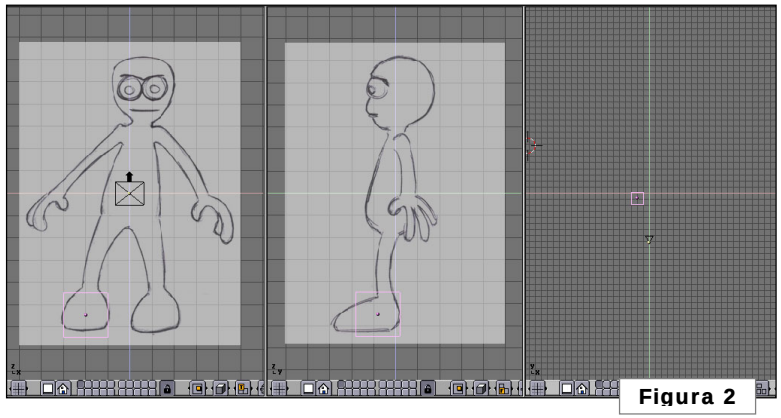


Figura 2

Puede ser útil activar el modo de sombreado (tecla **Z**) en alguna vista, para tener una visión más precisa del estado del modelo. La selección de vértices puede realizarse de forma individual, o con la tecla **B** (o **BB**). Las operaciones básicas sobre los vértices serán Extrusionar (**E**), con al menos 3 vértices seleccionados, mover (**G**), rotar (**R**) y escalar (**S**).

La idea básica es ir "calcando" los bocetos, ajustando los vértices de la geometría de alto nivel. **Es importante conservar el menor número de vértices posible en el modelo a subdividir**. Cuanto más aumentemos la geometría del modelo, más difícil resultará desplazar los grupos de vértices. Con esta idea en mente, comenzamos a generar un pie del modelo, como se muestra en la figura 3. Para extrusionar una cara (sacar la pierna de la parte superior del pie), seleccionamos los 4 vértices que formarían el "talón" y pulsamos **E**.

Al llegar a la parte de unión entre las piernas, realizaremos un espejo de la pierna izquierda. Para ello, seleccionamos la malla, pulsamos **Shift + D**, luego **S** (escalado) y sin escalar la malla, pulsamos la tecla **Y**. Esto hace un espejo de la malla respecto de ese eje. Obtenemos un resultado similar a la figura 4. Pasamos a unir los objetos generados para continuar modelando el torso del personaje.

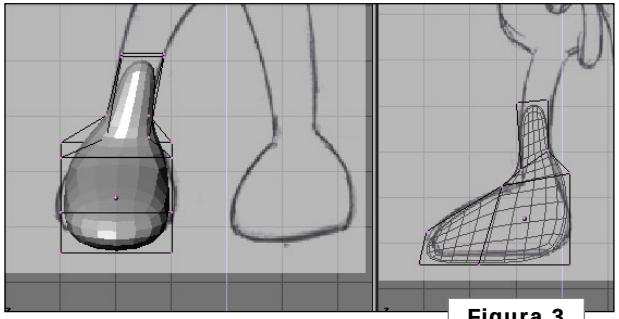


Figura 3

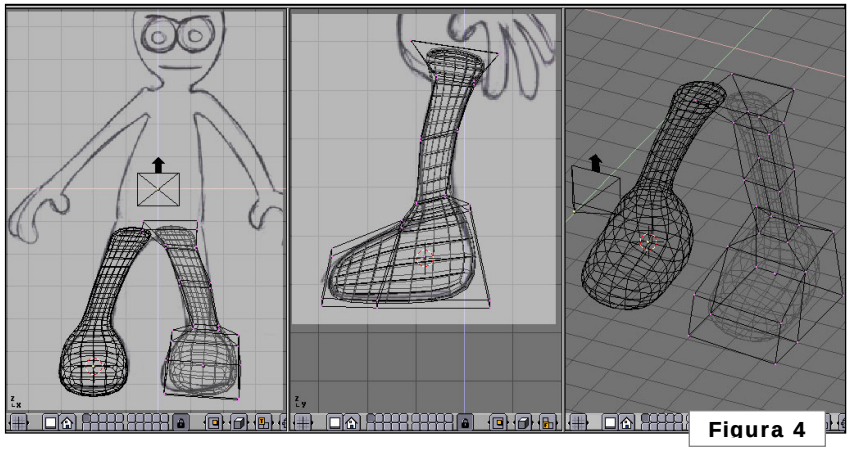


Figura 4

Con las dos piernas seleccionadas, unimos las mallas (**control + J**). Entramos en modo de edición de vértices y situamos cerca los vértices que deben ser comunes en ambas piernas (los dos vértices superiores de cada pierna). Tal y como se muestra en la figura 5. Seleccionamos los 4 vértices que queremos "fusionar", y ajustamos en los botones de edición (**F9**) el valor **Limit** que nos gradúa la distancia que debe haber entre vértices continuos a eliminar. Hecho esto, pinchamos en **Rem Doubles** y Blender debe informarnos que ha eliminado 2 vértices.

Continuamos modelando el cuerpo del personaje, siempre teniendo en mente que debemos utilizar el menor número de polígonos posible. Hay que ver también las partes donde "interesa" generar nuevas caras. Por ejemplo, para modelar el torso (figura 6 arriba), hemos realizado tres extrusiones cuando habría bastado con dos. De esta forma, hemos obtenido unos vértices donde, en el paso siguiente, generamos los brazos.

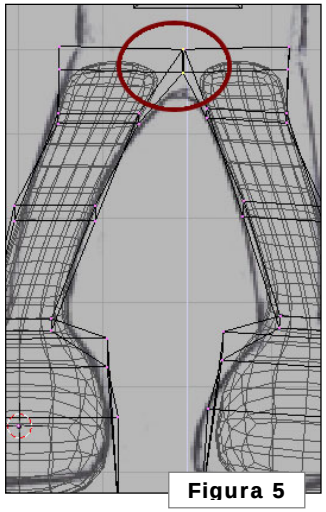
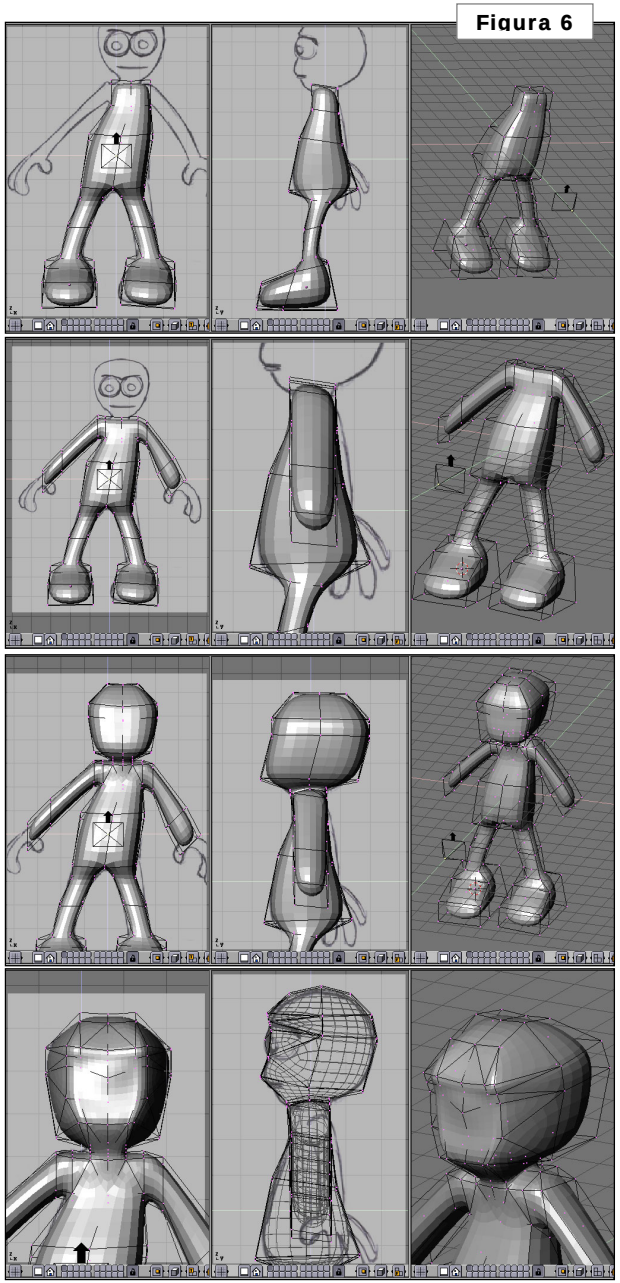


Figura 5

Necesitamos generar nuevos vértices para sacar el cuello y, de ahí, la cabeza. Para ello, resulta útil la herramienta "**Subdivide**" (botones de edición, **F9**). Es importante utilizar esta herramienta sólo cuando sea necesario. El número de vértices crece muy rápido y el modelo podría hacerse inmanejable. También resulta útil la opción de "**Smooth**", del mismo bloque de botones. Con este botón suavizamos una superficie, eliminando la apariencia de estar generada de forma simétrica. Esta opción la utilizaremos únicamente cuando la geometría base esté totalmente definida, para añadir un toque de naturalidad a la superficie.

Generaremos una mano del modelo y realizaremos la otra por "espejo" (de forma similar a como trabajamos con la pierna). En la figura 8, se muestra el proceso de ajustar los vértices para que "coincidan" con la muñeca. De forma similar, pinchando en **Rem-Doubles** juntaremos las superficies en una. Blender realizará la interpolación de la malla resultante.

Figura 6



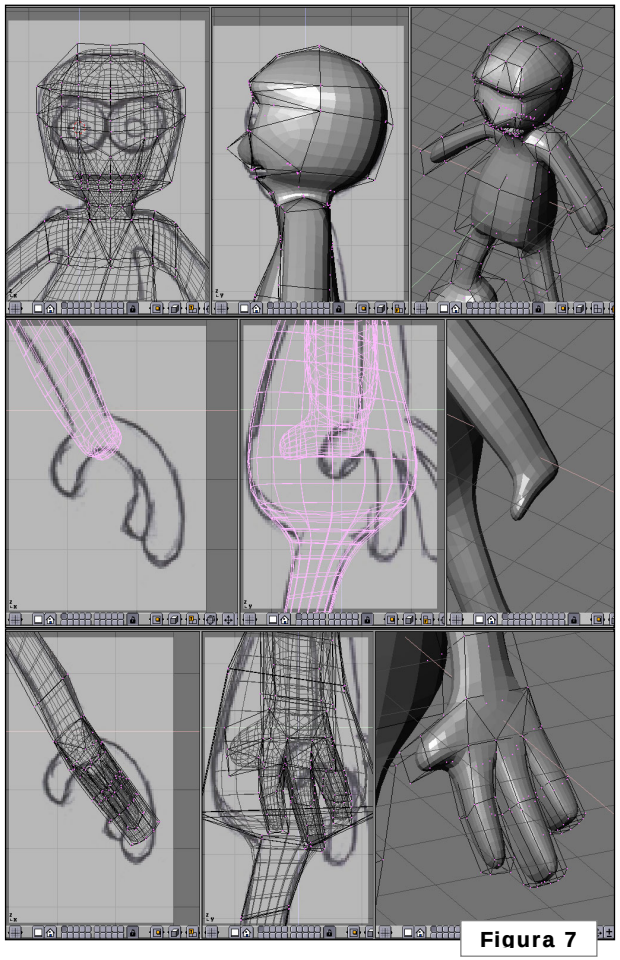


Figura 7

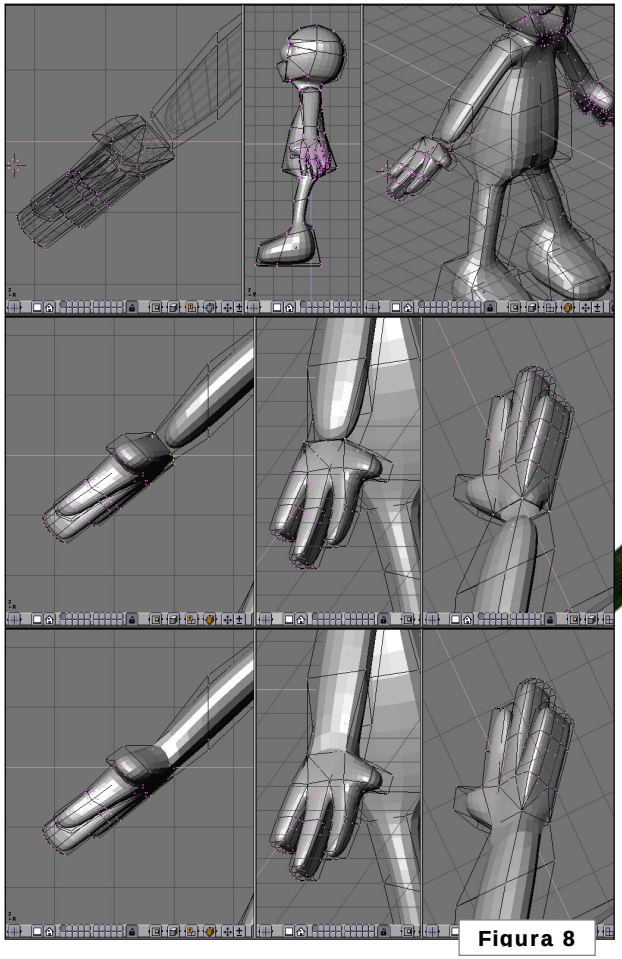
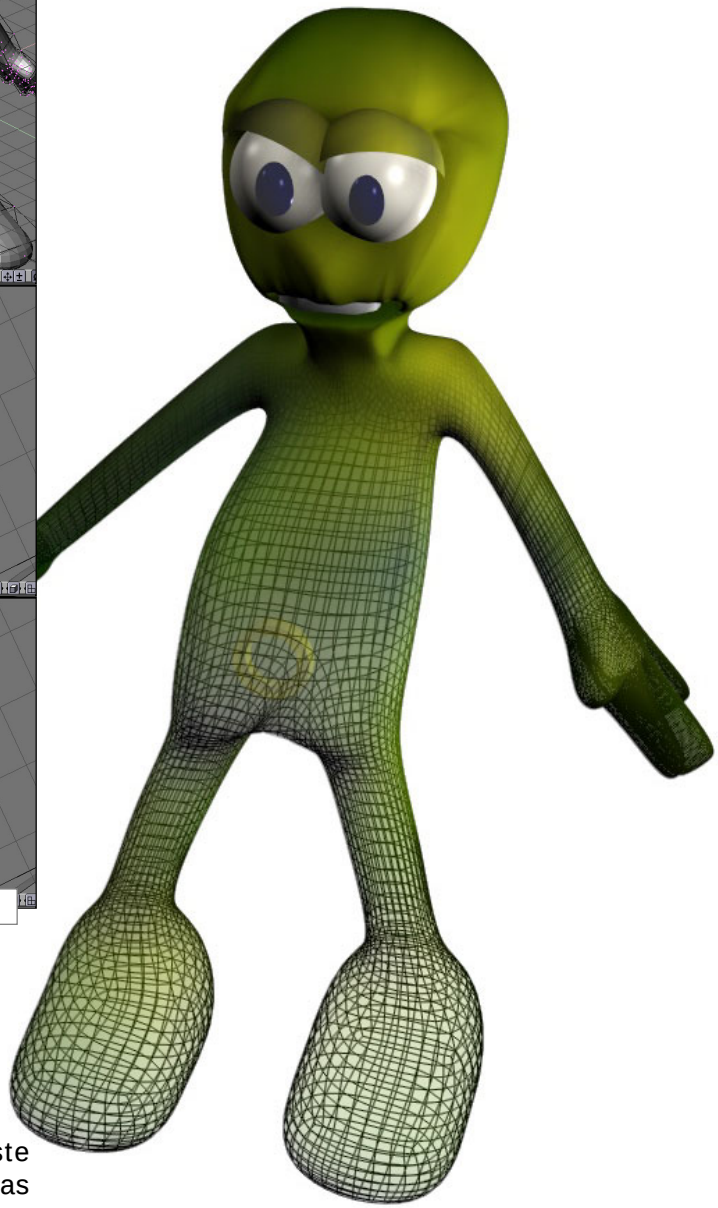


Figura 8



Por último, añadimos los ojos, párpados y dientes del modelo. Para las texturas se ha utilizado un color verde de material y después se han pintado los vértices (**Vertex Paint Mode**), como se explicó en la práctica 4 (Modelado del dinosaurio).

Con esta técnica podemos construir modelos de personajes complejos. Partiendo de unos bocetos 2D sencillos generamos la malla 3D mediante rotoscopia. Al emplear superficies de subdivisión podemos ajustar el nivel de suavizado de la superficie final. Habitualmente un valor de subdivisión de 3-4 en el renderizado, da mallas muy suavizadas. Animaremos este personaje en la siguiente sesión de prácticas con NLA.

Práctica 10

ANL y Esqueletos Avanzados

La animación de personajes digitales requiere definir un esqueleto interno al que aplicaremos Cinemática Inversa para calcular la posición de todas las articulaciones. Éstas generarán unas deformaciones que habrá que trasladar a la malla del modelo 3D. Por último se aplicará un movimiento al esqueleto, construyendo una secuencia de animación.

Así pues, en esta sesión identificamos dos objetivos; en primer lugar, construir un esqueleto para nuestro personaje, que asociaremos a una malla. Indicaremos a Blender qué vértices de la malla tendrá que deformar junto al esqueleto. En segundo lugar, construiremos una secuencia de animación para el esqueleto. Podremos realizar esta secuencia mediante la técnica de animación lineal empleada en sesiones anteriores; definiendo la posición de todos los puntos finales (*IK Solver*) en cada frame clave, o bien definiendo acciones en el módulo de Animación No Lineal. Utilizaremos esta segunda alternativa, mucho más cómoda para construir animaciones complejas.

Primera parte: Construcción del esqueleto.

Antes de definir el esqueleto del modelo creado en la práctica anterior, veamos cómo asociar un Armature a una malla de polígonos, y la convención de nombrado que utiliza Blender.

Comencemos con un ejemplo sencillo; arrancamos blender, eliminamos el plano por defecto e insertamos un cubo. Extrusionamos la cara lateral y, activando las superficies de subdivisión, construimos una malla como la mostrada en la figura 1.

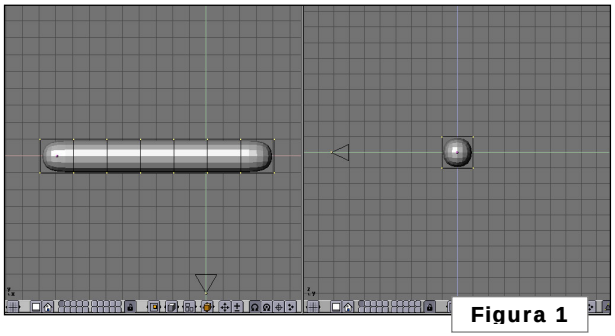


Figura 1

Añadimos un Armature formado por 3 huesos principales (**hueso1**, **hueso2**, **hueso3**) y un cuarto hueso pequeño que utilizaremos para resolver la cinemática inversa (**huesonull**). Con el esqueleto todavía en modo de edición de huesos (color rosa oscuro, accedemos con **Tab**), añadiremos un quinto hueso (independiente de los anteriores, aunque forma parte del esqueleto), que utilizaremos para calcular la cinemática inversa (**huesoik**). Obtendremos una configuración similar a la figura 2.

Es importante que todos los huesos formen parte de un único objeto Armature. Por tanto, cuando añadamos nuevos huesos al esqueleto, lo haremos siempre desde el modo de edición del objeto.

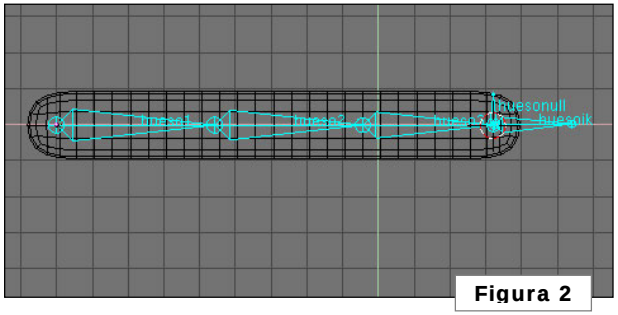


Figura 2

Podemos activar la representación del nombre de cada hueso pinchando en el botón **Draw Names** de los Botones de Edición (**F9**). Nombraremos los huesos como se explicó en la práctica 7, y añadiremos una restricción (en modo pose del esqueleto; **Control + Tab**) de tipo IkSolver al **huesonull** con objetivo en **huesoik**. Ahora podemos mover (en modo pose) **huesoik**, y el resto del esqueleto se calculará de forma automática mediante cinemática inversa.

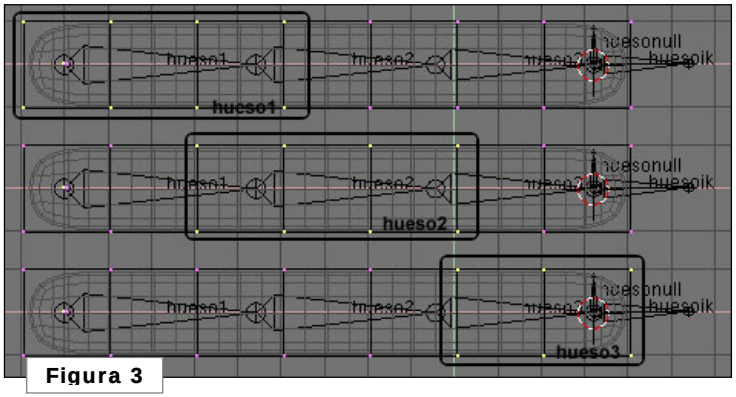


Figura 3

Falta por indicar a blender cómo deformar la malla. Para ello, crearemos grupos de vértices con el mismo nombre que los huesos del esqueleto. Asociaremos a cada conjunto de vértices un hueso de los anteriores. Entramos en modo de edición de vértices del objeto y seleccionamos el grupo de vértices que están situados sobre el **hueso1** (ver figura 3). En los botones de edición (F9), crearemos un nuevo grupo de vértices (pinchamos en **new**). Tecleamos el nombre del primer hueso y, con los vértices del primer hueso seleccionados, pinchamos en **Assign**. Si nos equivocamos al asignar vértices, podemos quitar los erróneos con **Remove**. El botón **Select** sirve para mostrar los vértices seleccionados de un grupo y **Deselect** para deseleccionar los vértices del grupo. Creamos un grupo para cada hueso y asignaremos los vértices que se muestran en la figura 3.

Si nos fijamos, hay vértices que pertenecen a dos grupos. Es buena práctica que los vértices situados en la zona de unión de dos huesos, pertenezcan a los dos grupos de vértices asociados. Blender calculará cómo deformarlos para que el resultado final en la malla sea suave.

Hecho esto, haremos la malla hija del esqueleto (**Control + P**) → **Use Armature**. Entramos en modo de pose para el esqueleto y movemos el hueso **huesoik**. Si todo ha ido bien, se deberá aplicar la deformación en el modelo (ver figura 4). Vamos a aplicar el mismo método al fichero *p10modelo.blend*.

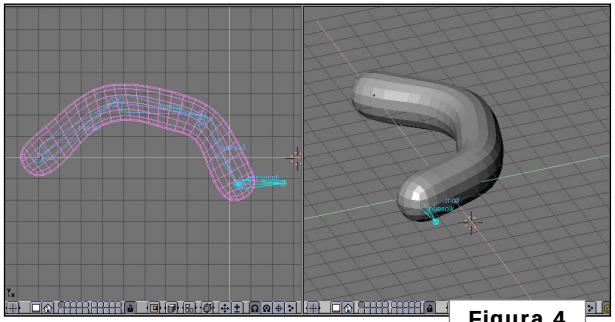


Figura 4

Comenzaremos por el esqueleto de las piernas. Los huesos que tengan simétricos en el otro lado del cuerpo, como por ejemplo las piernas y los brazos, los nombraremos acabando en **.R** (los huesos de la derecha) y en **.L** (los de la izquierda). Esto nos servirá para "*copiar*" y "*pegar*" posiciones simétricas en la segunda parte de la práctica. De momento nos ocuparemos de mantener la notación exacta y consistente.

Añadimos como primer Armature **Fémur.R**, **Tibia.R** y **PiernaNull.R**. Después, y recordemos que tenemos como objetivo tener un único Armature con todos los huesos, sin salir del modo de edición, añadimos una nueva cadena de huesos: **Pie.R** y **PieNull.R**. Finalmente, añadimos como cadenas de un único hueso: **DedosPie.R**, **IkDedosPie.R**, **IkTobillo.R** e **IkPie.R** (ver Figura 5).
Hecho esto, emparentamos (Botones de edición F9, modo de edición - **Tab** - con todos los huesos seleccionados, elegimos en el botón desplegable el nombre del padre en el campo **child of**; ver figura 6) los dos huesos IK pequeños; **IkDedosPie.R** e **IkTobillo.R** con el controlador principal del pie: **IkPie.R**. Al hueso **DedosPie.R** añadimos una restricción "*copy location*" hacia el hueso **IkDedosPie.R**. Al hueso **PieNull.R** un *IkSolver* hacia **IkDedosPie.R**. Al hueso **Pie.R** un "*copy location*" hacia **PiernaNull.R**. Finalmente a **PiernaNull.R** un *IkSolver* hacia **IkTobillo.R**. Si todo ha ido bien, tendremos un comportamiento como se ve en la Fig 7.
Crearemos grupos de vértices como se explicó con el primer ejemplo de la práctica, nombrando los grupos igual que los huesos

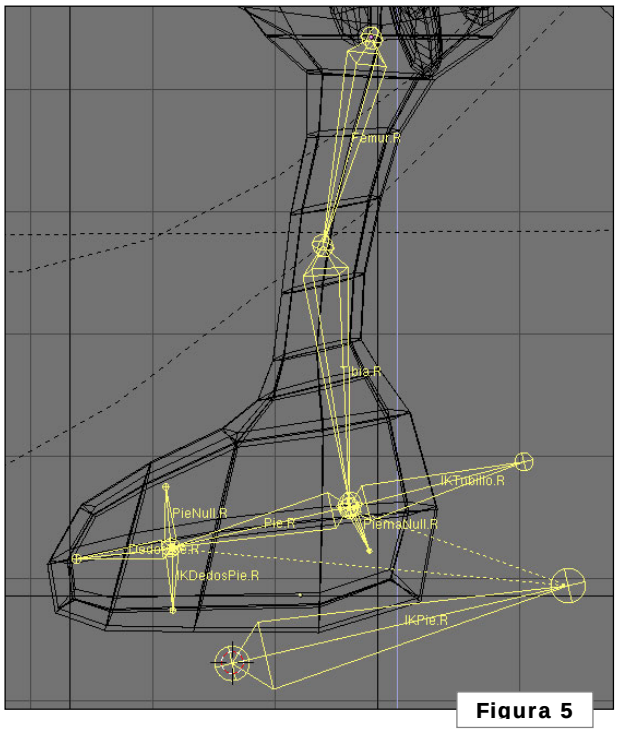


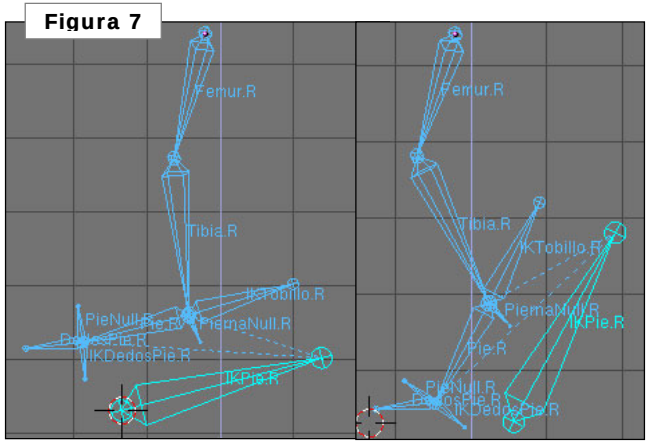
Figura 5

Figura 6

que los deformarán: **Fémur.R**, **Tibia.R**, **Pie.R** y **DedosPie.R**. Al terminar, podemos emparentar la malla con el esqueleto de la pierna. La deformación debe verse en este punto. A partir de ahora, todos los huesos que añadamos al esqueleto y grupos de vértices en la malla tendrán resultado directo en el modo de animación "pose".

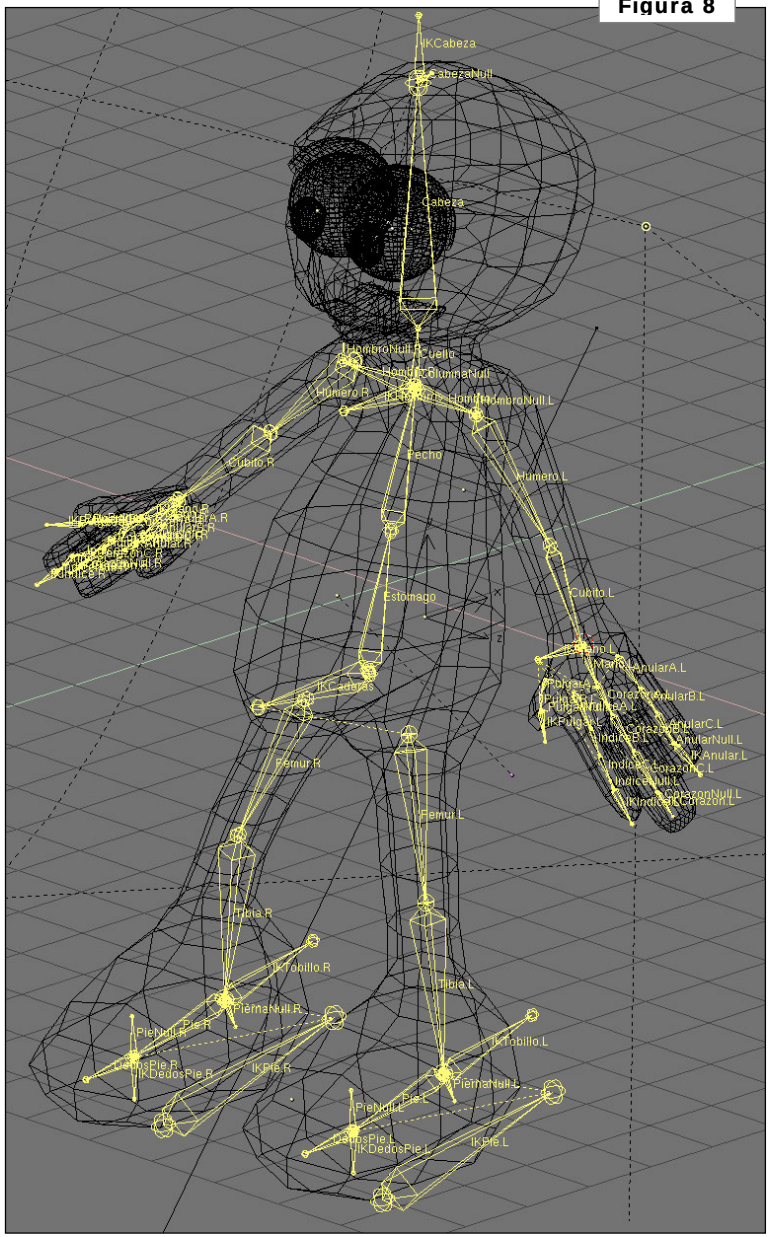
En la figura 8 puede verse una captura de la jerarquía que tenemos que construir.
La figura 9 muestra los nombres exactos de los huesos y sus relaciones de parentesco. Añadiremos una cadena de huesos para formar la columna vertebral (**Estomago**, **Pecho** y **ColumnaNull**). Para limitar el inicio y el fin de la columna, insertamos 2 nuevos huesos (**IkCaderas** e **IkHombros**). Añadimos una restricción de tipo *CopyLocation* a **Estomago**, respecto de **IkCaderas**. A **ColumnaNull** un *IkSolver* sobre **IkHombros**. De ese punto sacaremos dos cadenas de huesos (una para cada lado): **Hombro.R** y **HombroNull.R**. **Hombro.R** tendrá un *CopyLocation* de **IkHombros** y **HombroNull.R** servirá de origen para el *CopyLocation* de **Humero.R**. De forma similar se definirán los huesos y restricciones para el lado izquierdo.

Figura 7



Contruiremos los brazos como una cadena de 3 elementos: **Humero.R**, **Cubito.R** y **Mano.R**. Añadiremos un *IkSolver* (**IkMano**) sobre **Mano.R.** Añadiremos nuevas cadenas de huesos para los dedos. Índice, Corazón y Anular tendrán 3 huesos (el dedo pulgar sólo 2) más un cuarto hueso (*null) que servirá como origen del *IkSolver*. Añadiremos un hueso independiente para cada dedo, que servirá como destino del *IkSolver*. Además, emparentaremos el origen

Figura 8



de cada cadena de huesos con **Mano.R**. Una captura del esqueleto de esta parte puede verse en la figura 10.

Finalmente podemos ocultar algunos huesos para que, cuando estemos en modo de edición de pose, sólo nos aparezcan los que vayamos a utilizar. En la figura 9 aparece el conjunto de huesos que serían de utilidad a la hora de animar el personaje (básicamente los IK-Solver de todas las cadenas de huesos): IKPie.*, IKDedosPie.*, IKTobillo.*, IKCaderas, IKHombros, IKCabeza, IKMano.*, IKPulgar.*, IKIndice.*, IKCorazon.* e IKAnular.*

Puede resultar de utilidad tener el esqueleto en una capa distinta del resto de elementos. Esto nos permitirá ocultarlo cuando no estemos trabajando directamente con él. Después de ocultar un subconjunto de los huesos del esqueleto, obtenemos una configuración como la mostrada en la figura 11.

Segunda parte: Animación No Lineal.

Abrimos dos ventanas nuevas en blender. Una que contendrá "acciones" y otra de tipo NLA. En animación no lineal, definiremos las acciones como conjuntos de posiciones clave de un subconjunto de huesos. Así, en el movimiento de "andar", definiremos la acción de "mover las piernas" y la acción de "mover los brazos". De la combinación de ambas acciones en el compositor de NLA (Non Linear Animation) obtendremos el movimiento de caminar.

Vamos a añadir el movimiento de mover las piernas al caminar. En la ventana de acciones, crearemos una nueva pinchando en el icono y seleccionando (**ADD NEW**). Cambiaremos el nombre por "*caminar piernas*". Nos situamos en el frame 1 y movemos **IKPie.R** e **IKPie.L** hasta obtener una posición similar a la figura 12. Hecho esto, añadimos un frame clave (sólo con esos dos huesos seleccionados, únicamente intervienen dos manejadores en la acción) sobre LocRot. En la ventana 3D, al lado del icono de

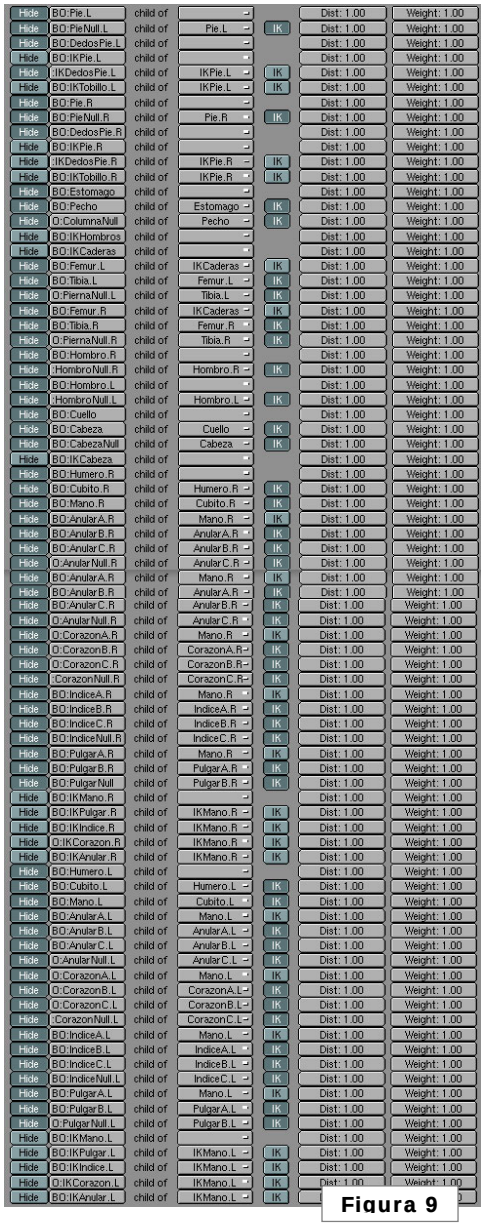


Figura 9

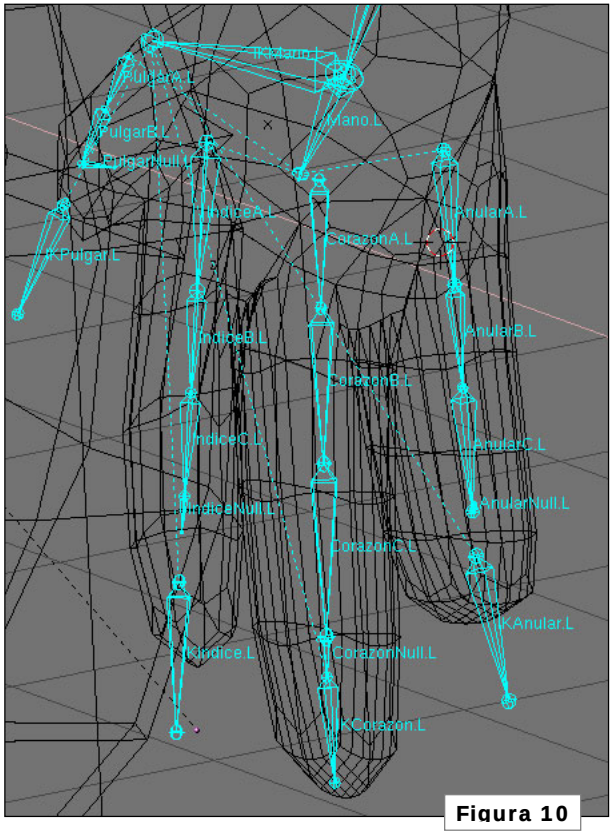


Figura 10

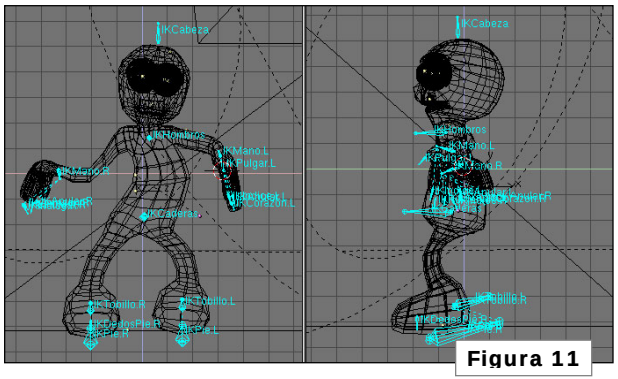


Figura 11

pose , aparecen 3 iconos que nos permiten copiar , pegar y pegar invertida la pose actual. Es en el pegado invertido donde entra en juego la notación especial de Blender. Podemos "pegar" la posición de las piernas invertida siempre que hayamos seguido la notación .R y .L para huesos simétricos. De esta forma, pinchamos en copiar la posición actual y nos desplazamos al frame 14. Pegamos la posición invertida e

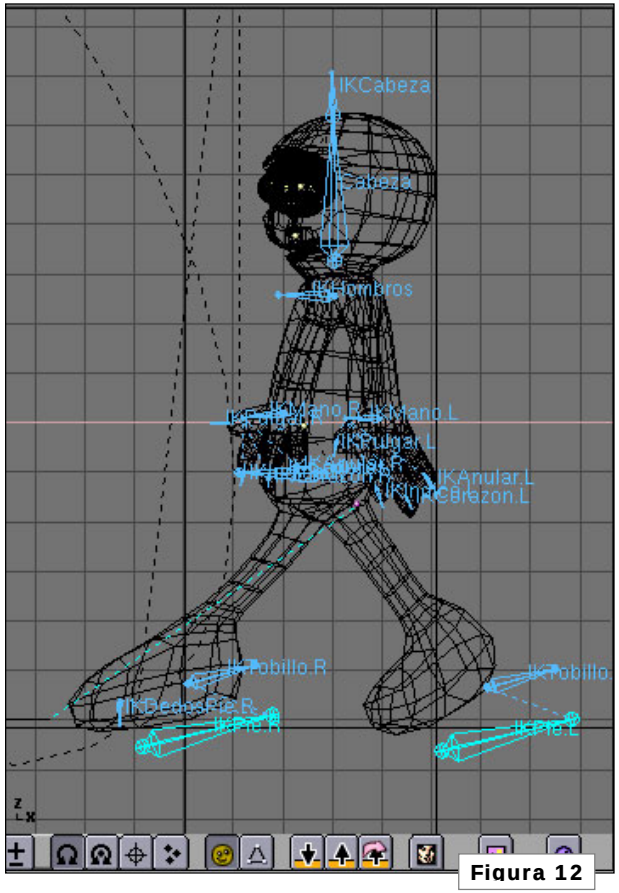


Figura 12

insertamos frame clave. Nos vamos al frame 28 y pegamos la posición normal . Insertamos de nuevo un frame clave. Si reproducimos la acción, tendremos un movimiento parecido a andar, pero sin levantar los pies del suelo. Bastará con que nos situemos en los frames intermedios (7 y 21) y añadamos nuevos frames clave con la pierna que corresponda levantada. Finalmente, obtendremos una acción en la ventana de acciones como se muestra en la figura 13. En la ventana NLA aparecerán una serie de cuadritos indicando los frames clave (ver figura 13, abajo). Nos interesa convertir estos frames a franjas NLA; que son barritas amarillas que podemos escalar, suavizar y com-

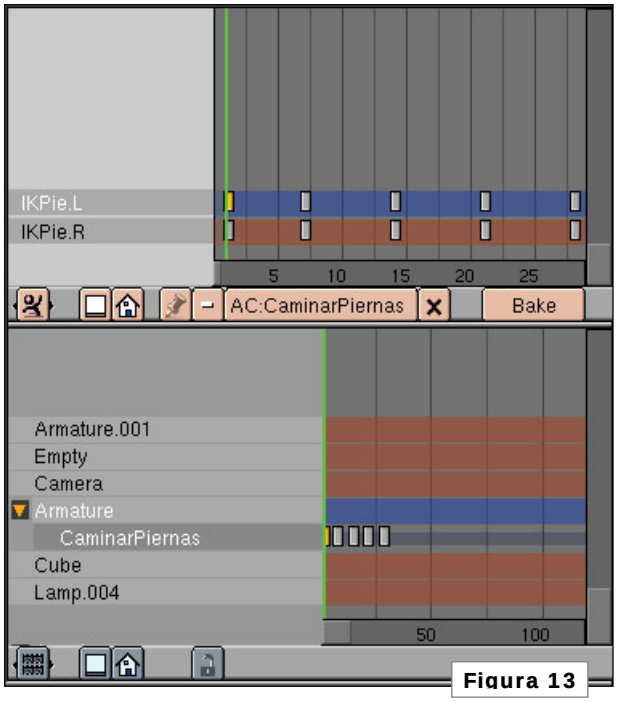


Figura 13

poner. Para ello, en la ventana NLA seleccionamos todos los cuadros clave y pulsamos **Control + A → Action to NLA Strip**. Hecho esto, podemos "estirar" la duración de la acción de caminar al número de frames que queramos (con la tecla **S**, como cualquier objeto), o desplazarlo en la línea de tiempo (tecla **G**). Crearemos nuevas acciones (como la de mover los brazos, taparse la cara, girar la cabeza...) de forma similar. Cuando hayamos creado la barrita NLA, podemos eliminar la acción de la ventana NLA para crear nuevas, pero no antes!. Si eliminamos los frames clave antes de pasar la acción a NLAStrip, estaremos eliminando la propia acción. Podemos añadir acciones ya creadas a la ventana NLA pulsando **Shift + A**.

Podemos ver cómo quedó la composición de acciones de esta práctica en la figura 14. La parte inferior, que tiene frames clave directamente sobre el objeto Lamp.004, se corresponde con el efecto del foco que ilumina la cara del muñeco al comenzar la animación. A continuación analizaremos algunas opciones de composición.

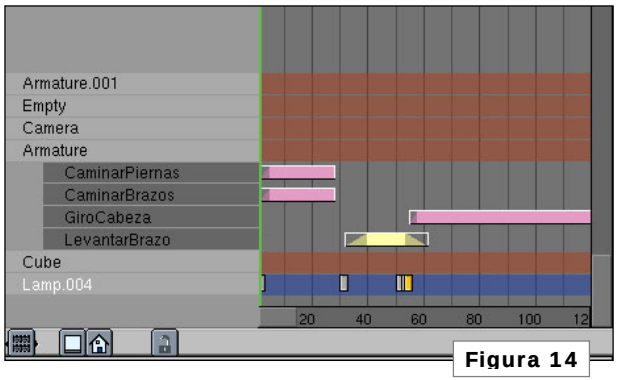


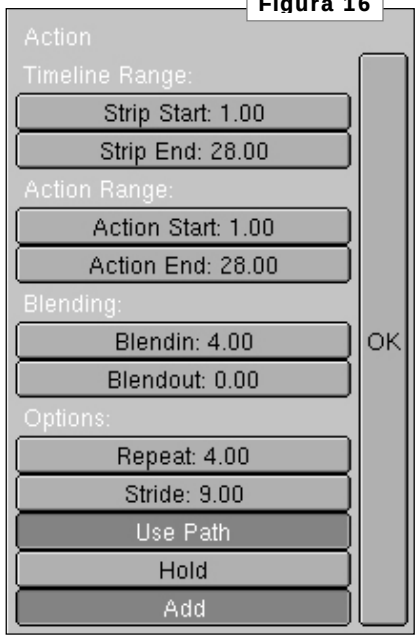
Figura 14

Figura 15

¿Cómo hacer que el personaje siga un camino? Necesitamos añadir un elemento de tipo **Curve / Path**. Ajustamos los puntos de control y en los botones de animación (F7), indicamos los parámetros que se muestran en la figura 15. En **PathLen** indicaremos la longitud en frames del Path. Activaremos los botones de **CurvePath** y **CurveFollow**.

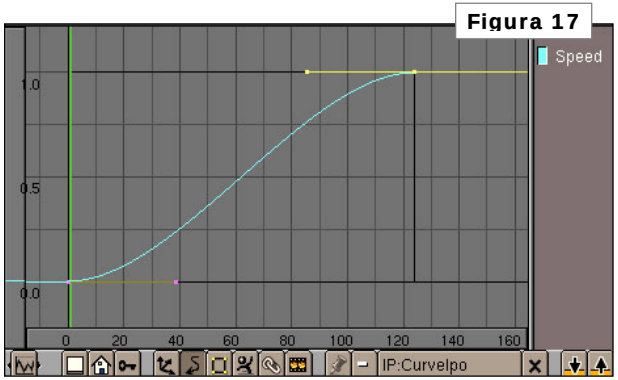
Hecho esto, hacemos al Esqueleto hijo del Path. En la ventana NLA, seleccionamos la acción "*caminar piernas*" y pulsamos la tecla **N**. Con esto, accedemos a las propiedes específicas de la acción (ver figura16). Las propiedades de **Strip Start** y **Strip End** indican el momento en el que la acción comienza y termina en la planificación NLA actual. **Action Range** hace mención a la duración (en frames) de la acción (independiente de dónde empiece). **Blendin** y **Blendout** son el número de frames que damos para mezclar la acción con las siguientes. Es un factor de suavizado que aplica Blender en la interpolación entre acciones. **Repeat** indica el número de veces que vamos a repetir la acción. **Stride** es un parámetro muy importante si estamos utilizando caminos (paths), como en nuestro caso. Indica a Blender el número de unidades que avanza el modelo en cada ciclo. Debemos ajustar este parámetro para evitar que el modelo "patine" sobre el suelo. Es útil pinchar en el botón **PrintLen** (ver figura 15), que nos da la longitud de un camino para calcular el valor de **Stride**. Si activamos **UsePath**, la acción se va a sincronizar con el avance del path. En nuestro ejemplo, activaremos este botón únicamente en las acciones que tengan que ver directamente con el acto de caminar; es decir, "*caminar piernas*" y "*caminar brazos*". El botón de **Hold**, si está activo, nos conserva la última posición alcanzada por una acción. Por último, el parámetro **Add** indica que el movimiento final resultará de la composición de esta acción con el resto (sus efectos se suman). Por lo general, este parámetro tendrá que estar siempre activo para todas las acciones.

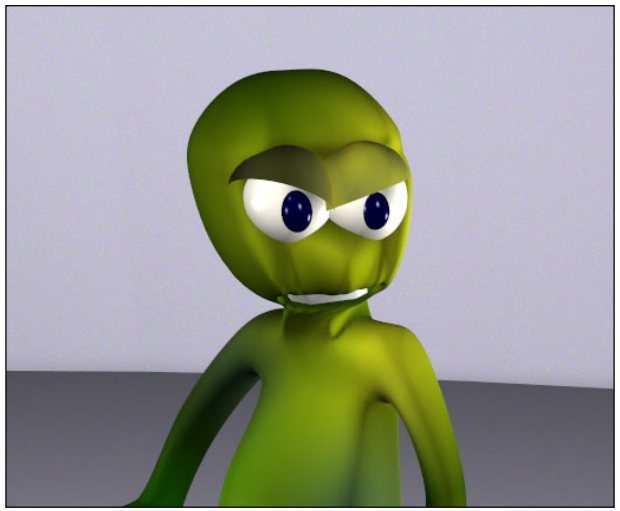
Figura 16



Por último, recordaremos que Blender permite trabajar hasta el mínimo nivel de detalle mediante curvas IPO. Podemos, por ejemplo, variar la velocidad del personaje que sigue sobre el path accediendo a la curva IPO asociada al camino (ver figura 17).

Figura 17





Práctica 11

Composición de Vídeo

En esta última sesión aprenderemos a manejar un programa de edición de vídeo: Zwei-Stein. Sincronizaremos un conjunto de vídeos, generados en prácticas anteriores, con efectos de transciones y rótulos mediante imágenes.

Al ejecutar el programa nos encontramos con el interfaz que se muestra en la figura 1.

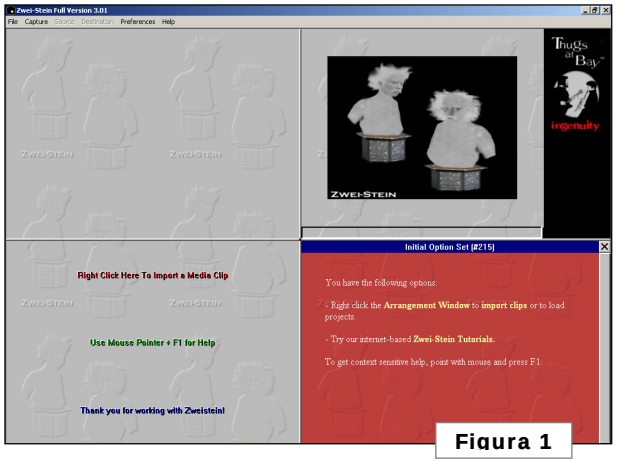


Figura 1

La ventana aparece dividida en 4 partes. Arriba a la derecha está la zona de previsualización. En esta parte podemos reescalar los vídeos que componemos, desplazarlos, etc... Además, ofrece una visión del resultado que obtendremos. Abajo a la derecha aparece la ventana de ayuda. Arriba a la izquierda está la zona de filtros y propiedades. Por último, abajo a la izquierda aparece la típica línea de

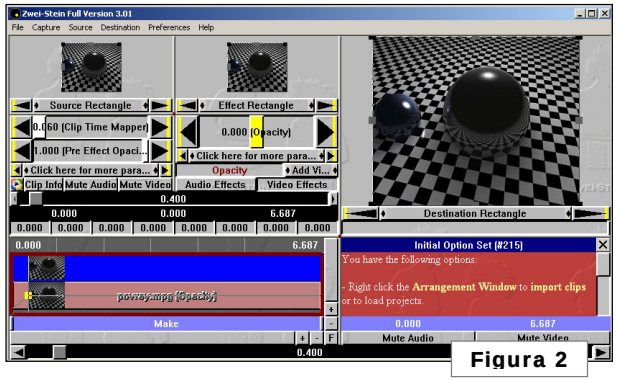


Figura 2

tiempo de los programas de composición de vídeo.

Antes de comenzar a cargar los vídeos, configuraremos el proyecto. En **Destination → Video Format** elegiremos **352 / 288 (CIF PAL)** , y en **Destination → Frames Per Second 25.** Guardamos el proyecto.

Para comenzar, cargaremos un vídeo haciendo BDR sobre la zona de la línea de tiempo **BDR → Import Video Clip**. Cargaremos "*povray.mpg*" (ver figura 2). Las imágenes y sonidos se importan con esta misma opción. La aplicación soporta AVI, BMP, JPG y WAV.

Podemos hacer "zoom" sobre la línea de tiempo pinchando en los botones + y – de su esquina inferior derecha. Pinchando en F el programa realiza el ajuste óptimo. Distinguimos nuevas partes del interfaz (ver figura 3). En **(a)** se definen los parámetros de entrada del vídeo (imagen o sonido). Entre otras cosas, podemos "recortar" únicamente la parte de un vídeo que nos interese utilizar. En **(b)** gestionaremos los filtros. En el vídeo de ejemplo, se han utilizado, principalmente, el de opacidad (**Basic effects → Opacity**) y el **RGB Key**, que nos permite definir qué color tomar como transparente (muy útil en rotulación). **(c)** nos define la posición, dentro del vídeo seleccionado, en que nos encontramos. Es un valor local. El valor global del vídeo se encuentra en la barra inferior de la ventana. La aplicación mide el tiempo en segundos (no en frames, como Blender). Podemos acceder a un valor exacto de tiempo haciendo doble clic con **BIR** sobre la barra de tiempo. Podemos especificar un valor numérico exacto prácticamente sobre cualquier

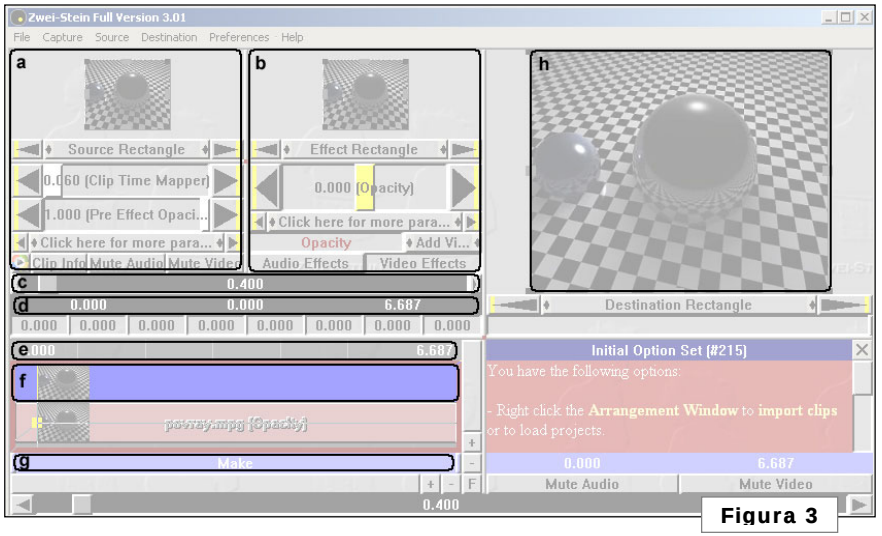


Figura 3

parámetro (barras deslizantes, etc...) con este método. **(d)** está formado por tres valores: punto de comienzo (en el tiempo general de la composición), comienzo local del vídeo (si lo hemos recortado por el principio, será distinto de cero) y punto final. Dando valores exactos a estos parámetros, podemos ajustar con total precisión las uniones entre partes. **(e)** permite desplazarnos por el tiempo general del vídeo. Encima de **(e)**, podemos definir "atajos" para llegar a ciertas partes del vídeo. **(f)** ofrece una previsualización del frame actual. La barra **(g)** define el intervalo que vamos a generar de vídeo. Cuando realicemos pruebas, sólo generaremos el intervalo sobre el que estemos probando. Cuando queramos generar el vídeo completo, haremos **BDR** sobre **(g)** + **Auto Adjust Make Strip**. Por último, **(h)** ofrece una previsualización del resultado final y un mecanismo para reescalar y desplazar los elementos que intervienen en la composición.

¿Cómo gestiona Zwei-Stein los frames clave y las interpolaciones? Para esta aplicación, al igual que la mayoría de editores de vídeo, las interpolaciones entre frames clave son lineales. Por tanto, los parámetros cambiarán a velocidad constante entre claves definidas. Para introducir un frame clave sobre cualquier parámetro, bastará con cambiar el parámetro en el frame actual. Por ejemplo, si quisiéramos introducir un degradado en la transparencia del primer vídeo, nos iríamos al segundo 0.0 y cambiaríamos el valor del filtro Opacity a –1. Después, avanzamos un poco (por ejemplo, segundo 0.5) y cambiamos el valor a 0. Observamos que aparece una línea de representación sobre el objeto vídeo, que indica la variación de ese parámetro. Los puntos amarillos indican frames clave. En la ventana de previsualización también aparecen cuadros amarillos cuando tenemos un frame clave. Podemos realizar operaciones básicas sobre los frames clave en **(a)** y **(b)**, accediendo con **BDR**. Generamos el vídeo resultado (delimitado por la franja *Make*) accediendo a **Destination/Export/Video for Windows *.AVI**. Un codec que da muy buen resultado es el de *Intel (IYUV)*.

Para terminar, en la figura 4 se muestra el esquema final resultante del vídeo generado en esta práctica.

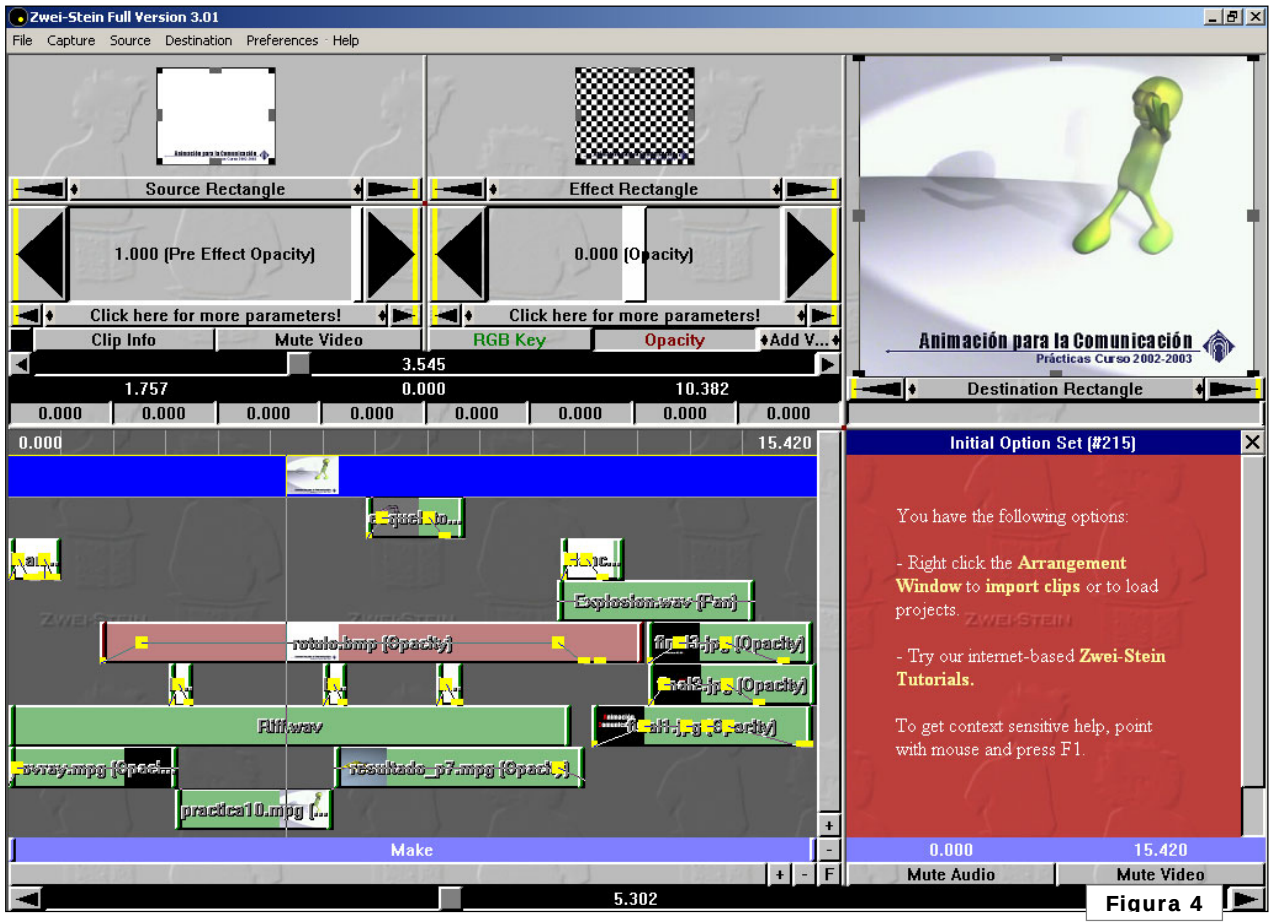


Figura 4

Especificación OREj v.0.2

Formato de Intercambio Blender - OpenGL

Esta propuesta de MetaFormato 3D, nace con la intención de ser el punto de partida para quien desee desarrollar videojuegos utilizando Blender, partiendo de una base sólida para la generación de Animaciones y Modelos 3D y posteriormente utilizarlos en la API 3D por excelencia: OpenGL. A su vez, se pretende crear un nexo entre las asignaturas de *Animación para la Comunicación* e *Informática Gráfica* en la Escuela Superior de Informática de Ciudad Real.

En esta primera versión pública (0.2), el formato almacena la **geometría del objeto** y permite texturizado mediante **mapeado UV** (UV Mapping). En futuras especificaciones del formato, se almacenarán movimientos del objeto (mediante puntos clave de la trayectoria, descritos con Splines), esqueletos internos y materiales asociados. No obstante, esta primera versión, totalmente funcional, será compatible con posteriores revisiones.

Como sabemos, el **mapeado UV** consiste en asignar a cada vértice del modelo, una coordenada (entre 0 y 1) del fichero de texturas. Esto nos permitirá variar la resolución de la textura de forma dinámica, sin tener que variar la información del mapeado. De esta forma, si estamos trabajando con una tarjeta gráfica poco potente, podremos tener una textura, por ejemplo, de 256x256 píxeles. Si tenemos un hardware mejor, podríamos trabajar con la textura a mayor resolución (512 x 512, 1024 x 1024...).

Recordemos que **Blender no** pone ninguna **restricción** a la hora de trabajar con **tamaños de texturas**, mientras que **OpenGL requiere** texturas cuyas dimensiones sean de la forma **2^n píxeles**. Así pues, en Blender nos adaptaremos a utilizar texturas con estas dimensiones.

```
# Objeto OREj: cubo
# Vertices Totales: 8
# Caras Totales: 6
v 1.0000 1.0000 -1.0000
v 1.0000 -1.0000 -1.0000
v -1.0000 -1.0000 -1.0000
v -1.0000 1.0000 -1.0000
v 1.0000 1.0000 1.0000
v 0.9999 -1.0001 1.0000
v -1.0000 -1.0000 1.0000
v -1.0000 1.0000 1.0000
f 4 3 2 1
f 6 7 8 5
f 2 6 5 1
f 3 7 6 2
f 4 8 7 3
f 8 4 1 5
t 0.015625 0.99609375 0.015625 0.51171875 0.5 0.51175 0.5 0.99675
t 1.0 0.99609375 0.51175 0.9921875 0.515625 0.50325 1.00395 0.5075
t 0.515625 0.015625 0.9921875 0.015 0.9921875 0.4925 0.525 0.4875
t 0.4921875 0.00725 0.49605 0.4975 0.0115 0.49675 0.0075 0.011775
t 0.49609375 0.0078125 0.5 0.5 0.00785 0.50390625 0.0035 0.01175
t 0.01175 0.492 0.01171875 0.01171875 0.49215 0.011 0.4925 0.4975
```

Listado 1

Un ejemplo de fichero en formato OREj, lo podemos ver en el listado 1. Los comentarios comienzan con '#'. La descripción de los vértices (que irán seguidos, en el mismo bloque), comenzarán con la letra "**v**" y a continuación sus coordenadas x, y, z relativas al centro del objeto. A continuación irán las caras del objeto (de cualquier número de vértices, siempre superior a 3), que comenzarán con la letra "**f**", seguido de los índices de los vértices descritos anteriormente. Por ejemplo, `f 4 3 2 1` definirá una cara formada por los vértices número 4, 3, 2 y 1. Recordemos que el orden en que se dan los vértices es importante a la hora de calcular el vector normal.

El orden es tal y como será necesario en OpenGL; el vector normal saliendo hacia afuera, si describimos los vértices en contra del giro de las agujas del reloj. Por último tenemos la parte de descripción de coordenadas de textura UV (comenzando con la letra "**t**"). Hay dos coordenadas por cada vértice (la primera corresponde al parámetro U y la segunda al V). El orden en que aparecen las coordenadas de textura (t) es el mismo que el orden en que aparecieron las caras (f). Así, en el ejemplo del Listado 1:

```
t 0.015625 0.99609375 0.015625 0.51171875 …
f          4          3          …
```

Para el primer vértice (4) de la primera cara tenemos que su componente U vale 0.015 y la componente V vale 0.996.

```
import Blender
import math, sys
from Blender import NMesh, Object
from math import *

PATH ="c:/"      # La ruta donde se salvara el OREj

object = Object.GetSelected()
objname = object[0].name
meshname = object[0].data.name
mesh = NMesh.GetRaw(meshname)

filename = PATH + objname + ".orj"
file = open(filename, "w")
std = sys.stdout    # Asociamos stdout al fichero .orj
sys.stdout = file

print "# Objeto OREj:", objname
print "# Vertices Totales:", len(mesh.verts)
print "# Caras Totales:", len(mesh.faces)

# Escribimos toda la lista de vertices (v x y z)
for vertex in mesh.verts:
    x, y, z = vertex.co
    print "v %f %f %f" % (x, y, z)
# Escribimos toda la lista de caras (f v1 v2 v3 ...)
for face in mesh.faces:
    print "f",
    face.v.reverse()    # Invertimos las normales
    for vertex in face.v:
        print vertex.index + 1,
    print
# Lista de parametros uv de cada cara (t u1 v1 u2 v2...)
for face in mesh.faces:
    nvertex = 0
    face.v.reverse()
    print "t",
    for vertex in face.v:
        print face.uv[nvertex][0], face.uv[nvertex][1],
        nvertex = nvertex + 1
    print

sys.stdout = std
file.close()

print "Fichero salvado:", filename
```

Listado 2

Desarrollo en Blender

El proceso básico es el mismo que se explicó en la segunda parte de la práctica 4 de Animación para la Comunicación. Recordaremos que, una vez tengamos la geometría construida, pasaremos a texturizar el modelo con UV Mapping. Dividiremos la pantalla y en una parte cargaremos el fichero de texturas (ventana de tipo). Una vez cargada la imagen, asignaremos a cada cara del modelo una porción de la textura. Para ello, pulsaremos la tecla "**F**" e iremos al modo de selección de caras. El modelo aparecerá en color blanco. Seleccionaremos una cara (o un conjunto de ellas) y pulsaremos la tecla "**U**", iremos a "**UVCalculation**" y seleccionaremos "**From Window**". Ajustaremos los vértices de la cara hasta que estén situados correctamente en la textura. Para ver el resultado, activaremos el modo de sombreado detallado en la ventana 3D. En la figura 1 podemos ver un ejemplo de un cubo texturizado utilizando esta técnica.

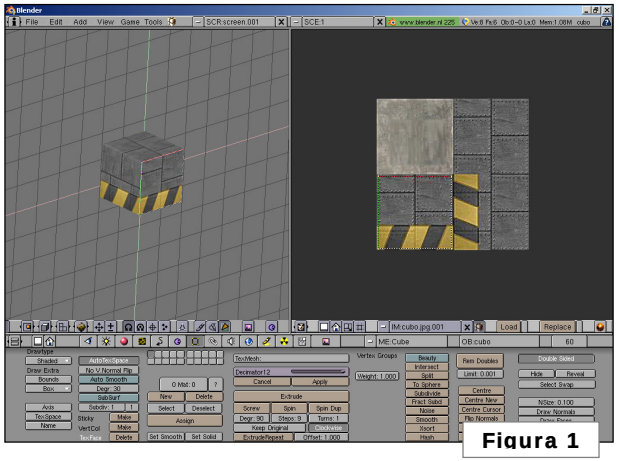


Figura 1

Como tenemos intención de exportar el modelo para trabajar en tiempo real con él, deberemos cuidar el número de polígonos que lo forman. Es de gran utilidad la herramienta de disminución de número de polígonos de Blender. Accedemos a ella con los botones de edición (F9). Podemos disminuir el número de triángulos con la barra deslizante. En la ventana 3D aparecerá el modelo con la pérdida de detalle que tenemos. Cuando el resultado sea adecuado, pincharemos en Apply.

Cuando tengamos el modelo terminado, cargaremos el Script, realizado en Python, que se muestra en el Listado 2. Este script nos permite exportar el objeto a formato OREj. Para cargarlo, cambiaremos una ventana a tipo texto y pincharemos en el botón con un guión + **Open New**. Cambiaremos la variable "PATH" del Script con la ruta completa donde queremos que se guarde el fichero .orj. El fichero se guardará con el mismo nombre que tenga el objeto en Blender.

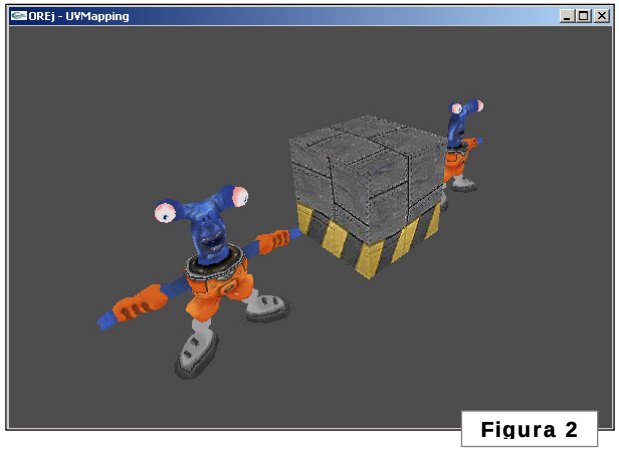


Figura 2

Ejecutaremos el Script con el objeto que queramos exportar seleccionado y, situándonos en la ventana de texto, pulsaremos **Alt+P**. En la ventana de MS-DOS de Blender aparecerá un mensaje como el siguiente:

```
Fichero salvado: c:/cubo.orj
```

Las texturas que utilizaremos en el programa de ejemplo de OpenGL estarán en formato RAW (con 24 bits de color). Esta primera versión del programa requiere que las texturas se guarden invertidas (la parte superior abajo), po resultar más cómodo a la hora de trabajar en OpenGL. Se podría rotar la textura en OpenGL, pero resulta más eficiente guardarla ya rotada (y nos evitamos ese paso en código).

Desarrollo en OpenGL

Una vez se tienen los archivos .orj y .raw se puede realizar un programa que haciendo uso de la librería OpenGl despliegue los objetos descritos con UVMapping como mallas de polígonos.

Listado 3

```
#include <GL/gl.h>
#include <GL/glu.h>
#include <stdlib.h>
#include <stdio.h>

#define MAXPUTOS 5000
#define MAXLINEA 200

typedef struct{
  float x;
  float y;
  float z;
}OrjVertice;

typedef struct{
  int v1; float uv1, vv1;
  int v2; float uv2, vv2;
  int v3; float uv3, vv3;
  int v4; float uv4, vv4;
}OrjCara;

typedef struct{
  GLuint     textura;
  OrjCara    caras[MAXPUTOS];
  OrjVertice vertices[MAXPUTOS];
  int        ncaras;
  int        nvertices;
}OrjObjeto;

void cargarTextura(OrjObjeto *obj,
                   int ancho,
                   int alto,
                   const char *ruta);
void cargarObjeto(OrjObjeto *obj,
                  const char *ruta);
void desplegarObjeto(const OrjObjeto *obj);
```

Para dicha tarea es necesario crear las estructuras necesarias en memoria que alberguen las caras del objeto y las coordenadas UVMapping de cada cara respecto de la textura.

Esto se puede realizar de muy diferentes maneras, a priori la más adecuada parece ser aquella que mantenga la información de los vértices separada de la información de las caras, ya que de esta manera se evitan duplicidades innecesarias y es mas fácil realizar un análisis de consistencia.

Se entrega una librería denominada OREj en su versión 0.2 (vamos casi ni ha nacido aún) que implementa las estructuras de memoria y los mecanismos necesarios para el despliegue de dichos objetos.

La librería ofrece una interfaz consistente de un tipo de datos y tres funciones. La estructura de datos **OrjObjeto** es la que permite manejar la información del objeto, en ella se contienen otras estructuras que guardan la información de la textura, los vértices y las caras del objeto con los valores UVMapping. Estas estructuras están definidas en el archivo `orej.h` (ver Listado 3).

Las funciones son tres:

- **cargarObjeto**: se le pasa como parámetro un puntero a objeto válido y el fichero que contiene la descripción del objeto a representar. Si el fichero contiene errores los notificará y abortará la ejecución del programa.
- **cargarTextura**: idem a la anterior carga y asocia una textura al objeto indicado.
- **desplegarObjeto**: se le pasa como argumento un puntero a objeto válido que ya haya sido cargado (textura y modelo) y despliega el objeto en el origen de coordenadas actual.

Se acompaña un proyecto en el lenguaje C formado de 3 archivos: `orej.h`, `orej.c` (ver en listado 4 un fragmento de la librería) y un *main* a modo de ejemplo de cómo usar las primitivas ofrecidas por esta librería. La salida por pantalla del programa de ejemplo puede verse en la Figura 2.

En un futuro se desea incluir mas primitivas que permitan la animación de los objetos OREj mediante Splines así como la constante mejora de esta librería que dado su embrionario estado no se puede asegurar su estabilidad en todos los entornos ni su correcto funcionamiento, salvo de momento en la plataforma Win32 en la que ha sido probada.

Para su uso no es necesario editar los archivos `orej.h` ni `orej.c` tan solo incluir el .h en los archivos necesarios y enlazar el resultado con el archivo `orej.o` para resolver las dependencias.

Listado 4

```
#include "orej.h"

void cargarTextura(OrjObjeto *obj, int ancho, int alto,
                   const char *ruta){
  GLubyte *imagen;
  FILE *fich;
  int i, j, c;

  if(obj==NULL) return;
  /* Reserva para la imagen */
  if(!(imagen = malloc(ancho*alto*3*sizeof(GLubyte)))){
    printf("no hay memoria para la textura\n");
    exit(-1);
  }
  if(!(fich = fopen (ruta, "rb"))){
    printf("no se encuentra el archivo %s\n", ruta);
    exit(-1);
  }
  /* Lectura de la imagen */
  fread(imagen, ancho * alto * 3, 1, fich);
  /* Asociar la imagen */
  glGenTextures(1, &(obj->textura));
  glPixelStorei(GL_UNPACK_ALIGNMENT,1);
  glBindTexture(GL_TEXTURE_2D, obj->textura);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
  glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, ancho, alto, 0, GL_RGB,
               GL_UNSIGNED_BYTE, imagen);
  /* Liberar recursos */
  fclose (fich);
  free(imagen);
}

void cargarObjeto(OrjObjeto *obj, const char *ruta){
  int   i=0, linea=0;
  char  cadaux[MAXLINEA];
  float a,b,c,d,e,f,g,h;
  FILE  *fich;

  if(obj==NULL) return;
  if(!(fich = fopen (ruta, "r"))){
    printf("no se encuentra el archivo %s\n", ruta);
    exit(-1);
   }

  while (fgets(cadaux, MAXLINEA, fich)!=NULL){
    linea++;
    switch(cadaux[0]){
      case '#': break;
      case 'v': sscanf(&cadaux[2],
                       "%f %f %f",
                       &obj->vertices[obj->nvertices].x,
                       &obj->vertices[obj->nvertices].y,
                       &obj->vertices[obj->nvertices].z);
                obj->nvertices++;
                break;
      case 'f': a = b = c = d = -1;
                sscanf(&cadaux[2], "%f %f %f %f",
                                   &a, &b, &c, &d);
                obj->caras[obj->ncaras].v1 = (int) --a;
                obj->caras[obj->ncaras].v2 = (int) --b;
                obj->caras[obj->ncaras].v3 = (int) --c;
                obj->caras[obj->ncaras].v4 = (int) --d;
                obj->ncaras++;

                if(a<0 || b<0 || c<0){
                  printf("Error de formato en '%s', linea:
                            '%d'\n", ruta, linea);
                  printf("Cara '%d' un vertice es invali
                             do\n", obj->ncaras-1);
                  exit(-1);
                }
                break;
      case 't': a = b = c = d = e = f = g = h = -1;
                if(i>obj->ncaras){
                  printf("Error ‘%s', linea: '%d'\n", ruta, linea);
                  printf("valor uv no asociado a cara\n");
                  exit(-1);
                }
                sscanf(&cadaux[2],
                       "%f %f %f %f %f %f %f %f",
                       &a, &b, &c, &d, &e, &f, &g, &h);
                obj->caras[i].uv1 = a;
                obj->caras[i].vv1 = b;
                obj->caras[i].uv2 = c;
                obj->caras[i].vv2 = d;
                obj->caras[i].uv3 = e;
                obj->caras[i].vv3 = f;
                obj->caras[i].uv4 = g;
                obj->caras[i].vv4 = h;

                if(obj->caras[i].v4>0 && (g<0 || h<0)){
                  printf("Error '%s', linea: '%d'\n", ruta, linea);
                  printf("Cara '%d' valor uv invalido\n", i);
                  exit(-1);
                }
                if(a<0 || b<0 || c<0 || d<0 || e<0 || f<0){
                  printf("Error '%s', linea: '%d'\n", ruta, linea);
                  printf("Cara '%d' valor uv invalido\n", i);
                  exit(-1);
                }
                i++;
                break;
    }
  }
  fclose (fich);
}
```