CURSO DE CREACIÓN DE VIDEOJUEGOS MEDIANTE BLENDER

1. INTRODUCCION.

Este es mi segundo tutorial sobre **Blender**. En esta ocasión voy a explicar como este programa puede ser utilizado en la edición de videojuegos. **Blender**, es un programa de **3D** de uso genérico, que tiene la particularidad de poseer un motor de juegos propio, llamado **B**lender **G**ame **E**ngine (**BGE**).

En este taller abordaré varios aspectos de la edición de videojuegos. Tales como, la creación de escenarios, el uso de la luz, el modelado de objetos, la edición de materiales, el texturado realista y la animación de personajes. Sin olvidarnos, de la integración de todo ello en el motor de juegos de **Blender**, la importación y exportación de objetos, y cualquier otra técnica necesaria para producir un juego completo mediante **Blender**.

Por otra parte, voy a tratar de comunicar mi experiencia como especialista en el apartado de **Arte**, y la importancia de cuidar la historia y la estética para crear juegos que posean carácter y originalidad. Y la especial problemática que tienen todos estos temas, desde el punto de vista del desarrollo colaborativo en el entorno del **software libre**.

Para concluir esta introducción, creo que el mundo del videojuego en el entorno **Gnu/Linux** tiene un gran futuro. Lo cierto es que, aunque existen bastantes juegos sencillos, y algunos juegos muy avanzados, todavía hay espacio para muchos más. La verdad, es que las herramientas y técnicas para ello, no estaban disponibles hasta hace no demasiado tiempo.

2. COMENZANDO.

El desarrollo de un videojuego es un proceso mucho más complejo de lo que puede parecer. Se tiende a pensar, que comparado con la programación de cualquier aplicación "seria", debe ser algo así como un pasatiempo. La verdad, es que un juego es un programa normal, destinado al mundo del entretenimiento. Habitualmente, implica el uso del hardware, conocimientos y las técnicas más sofisticadas (**3D**, multimedia, inteligencia artificial, etc) precisando de un amplio grupo de desarrollo multidisciplinar. Este, suele incluir artistas gráficos, músicos, escritores, dobladores, traductores, programadores y a veces, incluso historiadores.

Actualmente, los integrantes del equipo de un gran juego, pueden superar en número a los de algunas producciones cinematográficas. Así que si has decidido comenzar un proyecto en solitario, preparate para trabajar duro.

Hace algunos años, era relativamente común el juego realizado por un solo desarrollador. El **Tetris** y el **Arcanoid** son ejemplos excelentes de ello. Hoy en día, suponen un conjunto de conocimientos, y habilidades demasiado amplias para que los pueda llevar a cabo una sola persona con éxito. Veamos a continuación como se suele organizar un proyecto de videojuego.

3. AREAS DEL DESARROLLO DE UN VIDEOJUEGO

Fundamentalmente son tres: **Concepto**, **Diseño** y **Producción**.

1- CONCEPTO.

Esta es la génesis del proyecto. Puede surgir en cualquier momento o lugar. En solitario, con unos amigos. Formalmente o no. Suceda de uno u otro modo, es conveniente desde el principio dejar claras las ideas principales del proyecto. Esta fase es muy importante, aunque a veces nos olvidemos un poco de ella. Hay que decidir el tipo de juego que vamos a desarrollar. Estrategia, aventura gráfica,

plataformas, simulación, combate en primera persona, etc. El estilo del mismo, la ambientación, el aspecto estético, etc, si va a ser un juego violento o pacífico, en que época se ambienta, si tiene un carácter optimista o tenebroso. En que difiere de otros juegos similares. Si va a tener una historia o no, etc. En esta fase conviene esbozar, no detallar mucho las cosas.

En la etapa siguiente todo ello se aclarará más. Pero es conveniente, que las lineas generales del proyecto no resulten ambiguas. Debemos entender bien las características del juego, antes de pasar al siguiente área de desarrollo. No se sabe cuanto tiempo dura esta fase. Se pueden tardar años en ello. A veces existe algo que se quiere hacer, pero no se sabe exactamente que. Pero con el paso del tiempo la idea va tomando forma.

2- DISEÑO.

En esta etapa, partiendo del concepto, se escribe el guión de la historia, y mediante algunos dibujos simples, se va estableciendo la estética del juego. Es muy importante trabajar el apartado de arte sonoro y musical. Es un error típico (que todos hemos cometido) no conceder una importancia suficiente al audio. Un juego con malos efectos de audio, mal doblado o sin doblar, o con una música vulgar desluce cualquier buen trabajo en otras áreas.

En el apartado de diseño también se establece la mecánica del juego. Es decir, perder, ganar, los personajes, los niveles, etc. Y la forma de implementar todo ello en el juego. Es decir, hay que escoger el lenguaje de programación que se va a utilizar, la plataforma de desarrollo, las librerías o **APIs** de las que nos vamos a servir, etc.

La fase de diseño es crítica. Los errores se pagan. Un lenguaje de programación inadecuado, o un motor **3D** ineficiente, lo pueden arruinar todo. Cuando todo sale mal, hay que volver a esta fase para reconsiderar de nuevo todo el proyecto.

En los grandes juegos, suele existir una fase de planificación después de la de diseño, precediendo a la de producción. Yo pienso que, en proyectos más pequeños, se puede planificar directamente desde la fase de diseño. Esto proporciona la ventaja de ayudar a ver algunas contradicciones, o errores que se hayan podido cometer en la fase del diseño.

La etapa de planificación, debe de aportar una idea bastante precisa, del esfuerzo que va a suponer el desarrollo del juego. Tras ella, se puede abordar el trabajo propiamente dicho. Si el número de personas implicadas en un proyecto es muy bajo (como suele suceder) o incluso si es una persona sola, tienen que estar resueltos todos los aspectos técnicos implicados antes de hacer nada más. Si no es así, habrá que entrar en una etapa previa "*de pruebas*" con modelos sencillos, para asegurarse de que el proyecto es viable. Aprender el lenguaje de programación, en la propia producción no es una buena idea. Tampoco lo es, utilizar un nuevo motor de juegos, que no sabemos realmente como funciona, pero que se dice que es excepcional. La problemática en el ámbito del **software libre** es bastante peculiar, así que tratare de ello de forma amplia más adelante.

3- PRODUCCION.

Se empieza a programar. Los artistas crean los gráficos y la música. Normalmente, lo más delicado suele ser la integración de todo ello, ya que todo se relaciona entre sí. El tamaño de los gráficos, su sincronización con el audio, la correcta programación, la capacidad del hardware, las exactitud de las especificaciones de la **API** y del lenguaje de programación darán lugar a nuestra obra.

El resultado, será con toda seguridad complejo, y sujeto a toda clase de errores. Así que, habrá que solucionarlos uno a uno, por el ancestral método de prueba y error. La producción es la fase más dura, pero también la más interesante si te gusta esto. Al principio todo parece sencillo, pero las personas y la acumulación de complejidad, crean un bloque de software cada vez más problemático.

Resolverlo, obteniendo finalmente un programa ágil e interesante, es un reto emocionante para un desarrollador de videojuegos bien motivado.

La fase de producción culmina siempre en una etapa de testeo. A menudo, no resulta tan estimulante como la producción. Se trata de poner a prueba el juego, en busca de errores, y realizar las modificaciones que sean necesarias para resolverlos, sin provocar daños en otras partes del programa. Este apartado de depuración suele ser tedioso. A veces, no hay manera de resolver un error. Puede tratarse de un bug de cualquiera de los elementos del código que utilicemos, habitualmente la librería o el lenguaje de programación. El tiempo que se puede tardar en limpiar de todos estos errores el código es desconocido. En función del grado de depuración, se entiende que existen dos tipos de juegos de prueba:

1- Versiones **Alpha**. Los propios desarrolladores prueban el juego y solucionan los errores graves o críticos.

2- Versiones **Beta**. Los usuarios, o un grupo escogido de ellos, prueban el juego para corregir otros errores, o mejorar algún aspecto del mismo.

También se puede hablar de una fase de post-producción. En ella, se evalúa el resultado final, y se planifican futuras mejoras o actualizaciones.

4-DISEÑO DE JUEGOS MEDIANTE SOFTWARE LIBRE

El desarrollo de juegos mediante **software libre** posee ventajas e inconvenientes. El inconveniente principal, consiste en que no se suele disponer de un gran presupuesto. Sin embargo, ofrece muchas ventajas. Para empezar, podemos estudiar e incluso reutilizar partes de software de otros programas. Si un proyecto resulta interesante, sin duda logrará colaboraciones. Así que, podemos empezar un juego en solitario, y acabar formando una comunidad en torno suyo. Muchas manos hacen más que solo dos, y muchos cerebros pueden aportar muchas ideas. Y un software interesante en algún aspecto, se puede volver con el tiempo en algo mucho más rico de lo que hubiéramos podido imaginar en un principio.

Para poder lograr un éxito así, hay que tener una buena idea, trabajarla, y hacerla pública en el momento adecuado. Cual es este momento no es fácil de decir, y si tendrá éxito tampoco. En mi opinión, el mejor momento es cuando el proyecto esta en fase de diseño avanzado, en pruebas de producción. De esta forma los voluntarios pueden comenzar la producción como tal, con el concepto y la planificación clara, pero teniendo todavía un cierto margen de maniobra para hacer aportaciones propias. Si ponemos a disposición pública, una versión terminada o casi terminada del juego sera difícil encontrar colaboraciones, y además, de existir no habrá mucho para hacer. Si lo hacemos de forma prematura, es posible que los voluntarios no entiendan bien lo que se pretende, el proyecto no capte su atención, o si lo hace no les resulte viable. Algunos podrán participar, pero produciéndose desencuentros entre el equipo debido a la carencia de una buena planificación. Hay que admitir, que no existe ninguna norma al respecto. Incluso existen desarrolladores, que prefieren iniciar el proyecto con el equipo completo desde el principio.

5- PRODUCCION DE GRAFICOS PARA VIDEOJUEGOS

En los videojuegos pueden existir varios tipos de gráficos. Al igual que en el resto de la producción gráfica, se utilizan gráficos vectoriales y de mapa de bit. Los primeros, se forman mediante fórmulas

matemáticas y se pueden escalar sin límite, ocupando poco espacio en memoria. Son útiles, si la librería o **API** que utilizamos los permite, para lineas, dibujos y mapas de color. Aunque pueden producir degradados, no logran buenos resultados fotográficos. Por el contrario, los gráficos de mapa de bit forman buenos degradados, y con buena resolución un acabado fotográfico perfecto. Se forman añadiendo miles de diminutos cuadrados de color llamados píxeles. Pero claro, ocupan más espacio en la memoria. Hasta la aparición de los gráficos en **3D**, la mayor parte de los gráficos de los videojuegos eran de este tipo.

Los gráficos en **3D** tienen un funcionamiento diferente. Un programa llamado **motor** o ***engine*** **3D** muestra una simulación de un entorno **3D**, que puede resultar convincente para nuestro cerebro, en comparación con la experiencia de visualizar un espacio real, en nuestra vida cotidiana. Los motores actuales, trabajan con gran cantidad de información, modificándola a gran velocidad. Esto nos permite crear entornos, es decir espacios, dotados de una gran credibilidad. E incluir en ellos eventos (sucesos) y personajes, para construir mediante todo ello, nuestro videojuego.

Aunque los gráficos en **3D** están triunfando actualmente, todavía existen muchos juegos realizados mediante gráficos en dos dimensiones. Incluso en los juegos en **3D**, existen partes de los mismos, que hacen uso de gráficos bidimensionales. Por ejemplo, los elementos de la interfaz, los fondos, los gráficos de presentación, la tipografía, y a menudo los efectos físicos, como fuego, humo o agua.

Los juegos **2D**, utilizan para expresar las acciones ,un tipo de gráficos de mapa de bits especiales llamados "**sprites"**. Estos, reúnen en un solo fichero, todas las acciones de un personaje o actor del juego. El programa, dibuja fragmentos del mismo en la pantalla del ordenador, a lo largo del tiempo. El resultado, es una especie de animación o película. El cerebro humano, no tiene problemas en identificar estas animaciones, como actores en el espacio del videojuego.

Los **sprites** pueden realizarse de muchas maneras. Lo normal, es utilizar programas de edición de gráficos como **The Gimp** o similares. Pero también se pueden crear, con técnicas de pintura tradicional (escaneándolos posteriormente), fotografiando maquetas o incluso objetos reales. Si se desea un control muy preciso de la perspectiva, o simular un espacio **3D** en **2D**, se puede utilizar software **3D**, como **Blender**, para obtener los gráficos.

En este taller, vamos a tratar el tema de los gráficos para videojuegos, utilizando el programa **Blender**. Veremos en primer lugar, como modelar objetos, texturarlos de una forma realista y renderizar imágenes, con destino a juegos de dos y tres dimensiones (sprites. fondos, botones, etc). Después, pasaremos a crear escenarios en **3D,** visualizarlos con el motor de juegos de **Blender**, modelar los personajes, y animarlos mediante esqueletos. Así como, las técnicas de integración de sus movimientos en el propio juego. Finalmente, veremos como combinando todo ello con las posibilidades del **Engine** de **Blender**, es posible realizar un juego.

6- ¿QUE ES BLENDER?

Blender, es un conjunto de aplicaciones integradas en un único programa, que cubren el proceso completo en la edición de videojuegos y producción de gráficos en **3D**. Incluye editor de audio, compositor de vídeo, mecanismos para obtener ficheros ejecutables, y naturalmente, un editor de videojuegos avanzado. No me parece conveniente, extenderme ahora acerca de la historia de **Blender**, ya que existe mucha información sobre ello en **internet**. Para los curiosos, ver el siguiente enlace:

http://es.wikipedia.org/wiki/Blender

Me parece más practico, considerar las posibilidades del motor de videojuegos de **Blender**, inmediatamente. De este modo, todo el mundo puede saber si el **BGE** es un software adecuado

para sus proyectos, antes de perder gran cantidad de tiempo y esfuerzo con él. De algún modo, pienso que también concede algún valor a este tutorial, ya que puede ser una buena orientación en este proceso. Por ello, pasemos a ver las ventajas e inconvenientes del tan renombrado **BGE** (motor de videojuegos de **Blender**).

-Ventajas:

1- Es libre.

2- Es gratuito.

3- Sistema de nodos.

4- Completo.

-Inconvenientes :

1- Rendimiento.

2- implementación incompleta.

Sobre "**software Libre**" y "**software gratuito**" ver el siguiente enlace:

http://es.gnu.org/Software_Libre

El sistema de **nodos** es una interfaz, constituida por una especie de ladrillos lógicos (llamados justamente nodos), relacionados entre sí mediante hilos. Los ladrillos, son conjuntos de datos, de un determinado tipo, y los hilos, hacen interactuar unos con otros, a gusto del usuario. De esta forma, es posible generar resultados complejos, sin necesidad de ser un experto programador. En realidad, en escenas muy complejas, es preciso hacer uso de guiones de **Python**, el lenguaje de programación que utiliza **Blender**.

La gran cantidad de funcionalidades de **Blender** es sin duda una gran ventaja, ya que nos evita el tener que instalar software adicional, para lograr un resultado completamente terminado. Una de las posibilidades mas interesantes que nos brinda, es la de generar ejecutables de forma sencilla, para la versión del sistema operativo que utilicemos. Basta con ir al menú: **File** / **Save Game as Runtime** y obtendremos un fichero ejecutable conteniendo la escena completa de **Blender**.

En cuanto a los inconvenientes, seguramente el mayor, sea la implementación incompleta del **BGE**. La razón de ello, se encuentra en la propia historia del programa. Comenzó siendo una aplicación privativa de edición de videojuegos, para convertirse en un programa libre de edición **3D** de uso general. En un momento dado del desarrollo, se reescribió el editor de materiales para modernizarlo, pero el **BGE** no se actualizo completamente. Por ello, muchas de la técnicas mas sofisticadas que pueden emplearse en **Blender**, no funcionan en el **BGE**. Afortunadamente, en los últimos tiempos asistimos a un mayor interés y esfuerzo, por el desarrollo del **BGE**, en las versiones más recientes del programa.

El rendimiento del **BGE** de **Blender**, es inferior al de los mejores motores de juegos comerciales, pero nada que no se pueda solucionar, poniendo un poco de atención a la hora de diseñar los gráficos **3D**. Es imprescindible cuidar de que los objetos tengan el menor numero de vértices posible, texturandolos

con precisión mediante texturas **UV**, y utilizando a menudo "**mapas de normales**" para incrementar su realismo.

Para poder hacerse una imagen de las capacidades del motor de juegos de **Blender** (**BGE**), he seleccionado los siguientes enlaces:

Video Bathroom en YouTube

Descargar Bathroom.blend

Esta escena, es un buen ejemplo de lo que puede llegar a lograrse. Podemos ver los vídeos en tiempo real o en el fichero de **Blender** (de extension **.blend)**. Para ello, ejecutamos **Blender**, y abrimos el fichero **Bathroom.blend** Una vez cargado, mediante la tecla "**p"** entramos en el modo de juegos. Es imprescindible disponer de un buen ordenador, una tarjeta gráfica moderna y que este configurada la aceleración gráfica por hardware.

En la "Figura 1", podemos ver una captura de pantalla de esta escena. Como puede apreciarse, el texturado posee un gran realismo. Grietas en la pintura, el relieve de las losetas, etc. La iluminación es sobresaliente. Como puede observarse, un foco de luz oscila levemente, proporcionando movimiento a las sombras que proyecta. Esta escena, esta repleta de pequeños detalles, como los metales corroídos, o la gota de agua, que cada cierto tiempo, cae del grifo de la bañera.

Son también muy destacables, los materiales blandos como la tela de la ducha. Si pulsamos el botón izquierdo del ratón, arrojaremos un pato de goma hacia delante. Este, rebotará donde lo lancemos deformándose (como un pato de goma cualquiera), hasta quedarse finalmente en reposo. Es divertido probar a lanzar patos de goma, en todas direcciones. Si apuntamos a la tela de la bañera, veremos en directo, las capacidades del **BGE** para manipular la compleja física de los objetos suaves, y sus interacciones con el resto del entorno, en el que se simulan fuerzas físicas, como la gravedad, colisiones, rebotes, etc.

figura 1

7- COMENZANDO CON BLENDER.

Este tutorial no es una guía de **Blender**. Esta orientado a la edición de videojuegos, y presupone que el usuario conoce y utiliza el programa, al menos a un nivel básico. Si no es así, estos enlaces a otros tutoriales, te ayudaran a adquirir el nivel adecuado para que puedas sacar todo el partido a este curso.

- http://wiki.blender.org/index.php/Doc:ES/Manual

- http://www.esi.uclm.es/www/cglez/fundamentos3D/index.html

Como se ha comentado en capítulos precedentes, ya deberíamos tener listo nuestro proyecto. Para empezar, hacemos uso de todo aquello que hayamos pensado para el apartado de **Arte** del mismo. Lo normal, es que este constituido por dibujos y planos de los escenarios, personajes, etc. Recapitulamos sobre ello, y ya estamos casi listos para empezar a trabajar con nuestro editor de **3D** favorito. He dicho casi... porque quizás precisemos un poco mas de trabajo previo. Me refiero, a dotarnos de todos aquellos recursos multimedia necesarios. Es decir, imágenes, texturas, efectos de sonido, música, etc.

8- MODELANDO NUESTROS PRIMEROS OBJETOS

Invariablemente, todos los cursos de **3D** empiezan por el modelado de objetos. Desgraciadamente, muchos cursos terminan con poco mas que ello. Sin restar merito a la edición de objetos, hoy en día, este trabajo supone solo una etapa más en el desarrollo de un videojuego. A menudo, se pueden conseguir los modelos ya editados en internet, o comprar galerías completas de objetos.

En cualquier caso, el modelado para videojuegos tiene particularidades con respecto al modelado “estandar”. Este último, tiene como fin la obtención de imágenes o vídeos, no renderizadas en tiempo real. El modelado para juegos se dice que es de “**baja resolución**”, en inglés "**Low poly**". Es decir, que debe de tener muy pocos vértices, pero situados de tal manera, que los resultados resulten complejos y con un alto nivel de realismo.

Para lograr este pequeño milagro, se hace uso de un tipo especial de texturas llamadas **UV**. Ya hablaremos de ello en profundidad mas adelante. Lo importante ahora, es saber que todos los juegos se texturan de esta forma (y no solo los realizados mediante **Blender**). Ello es debido, a que son los propios motores de juegos los que exigen este sistema de texturado. También es muy útil, servirse de “mapas de normales” para aumentar un poco el relieve de los materiales. Ahora, vamos a ver como hacer todo esto. Vamos a empezar modelando unos objetos sencillos. Ver “Figura 2”.

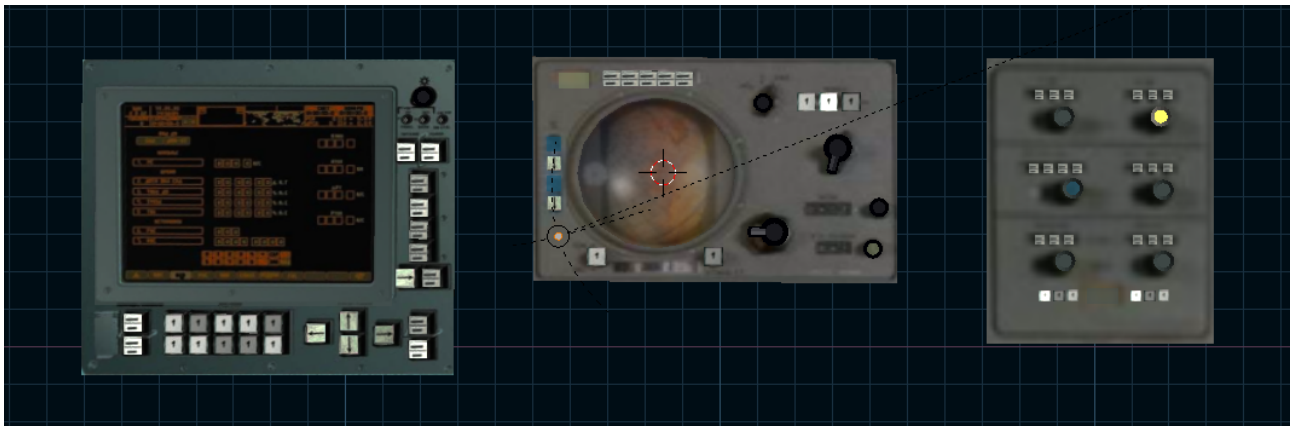

figura 2

Como puede verse, se trata de tres consolas de un vehículo espacial real (concretamente de una "**Soyuz"**). Vamos a realizar objetos **3D**, idénticos a las imágenes originales. Lo primero que hacemos, es obtener mediante el programa de retoque fotográfico **Gimp** (o de otro programa similar) una textura del frontal de cada equipo. Esto se logra recortando el frontal de la imagen y distorsionándola para adaptarla a una forma rectangular pura. Esta imagen, no nos servirá solo de textura, ya que la vamos a utilizar también, como fondo de la ventana **3D**, de tal manera, que podamos editar nuestro objeto tomando como referencia esta imagen.

Para utilizar una imagen como fondo, hacemos lo siguiente. En el menú inferior de la ventana activa **view** / **Background image** / **Use Background image** / **load** cargamos la imagen del directorio de imágenes de nuestro juego y mediante los controles **Blend**, **X offset** e **Y offset** modificamos su opacidad, su posición horizontal y vertical respectivamente. Ver "Figura 3".

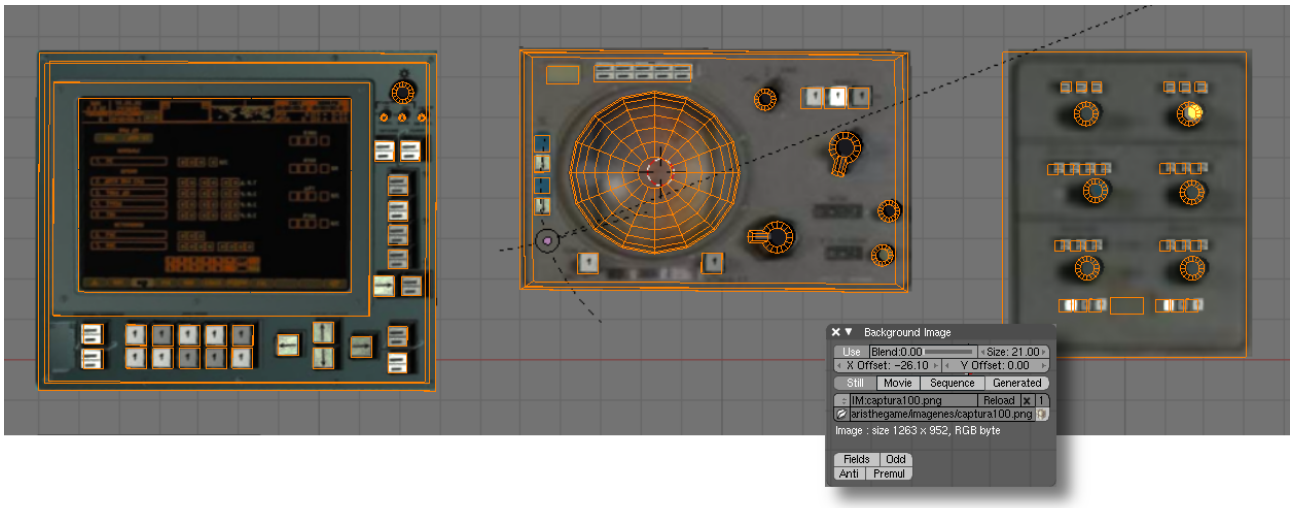

Figura 3

Después, creamos un cubo: **Add** / **Mesh** / **Cube** o utilizamos el que normalmente aparece por defecto en **Blender**. Más tarde, adaptamos el cubo a la imagen de fondo en las tres vistas. Frontal, perfil y superior. **1**, **3** y **7** respectivamente en el teclado numérico. En **3D**, siempre hay que tener mucho cuidado, de mantener las proporciones en estas tres vistas. En caso contrario, podríamos modelar un objeto totalmente correcto en una, o incluso en dos vistas, pero incorrecto en la tercera vista.

Para continuar, dotamos al objeto de biseles. El biselado. consiste en matar las aristas puras de los objetos, ver "Figura 4". Siempre que sea posible, debemos biselar los objetos. la razón es muy simple: la mayoría de los objetos que realizamos los humanos, tienen los bordes un poco redondeados. Los programas de **3D**, generan cuerpos de aristas puras, que resultan poco realistas. El biselado, se puede realizar de un modo muy simple, seleccionando las caras del cubo, extrusionándolas y reduciendo su escala un poco. El resultado, añadirá un toque de calidad a nuestro trabajo. Entre otras cosas, aparecerán los típicos brillos en los bordes, característicos de los objetos reales.

Pasemos ahora, a la edición de materiales. Empezamos creando un material base para el objeto. Para ello, vamos al panel de editor de materiales, mediante **F5** activamos el botón "**add New**", lo nombramos y en la pestaña "**Material**" seleccionamos "**Col**". Mediante los controles, el material tomará el color que queramos. El color del objeto cambiará automáticamente en la vista 3D. Es bastante sencillo.

En la pestaña "**Shaders**" encontramos varios parámetros muy interesantes. El botón "**Ref"**, arriba nos permite oscurecer o aclarar el objeto. Un valor de **0.800** es lo normal. Pero si necesitamos oscurecer un poco el objeto, basta con reducir su valor a **0.600** o menos. Debajo, vemos el botón "**Cook Torr**". Lo cambiamos por "**Phong**". Este es el mecanismo de **Blender** para generar los brillos. "**Phong**" da un resultado muy similar que la opción por defecto, y consume menos recursos en el **BGE**.

Los dos botones a la derecha son muy importantes. "**Spec**" es el tamaño del brillo, y "**Hard**" la intensidad del mismo. Los resultados los podemos ver de forma inmediata en una esfera a la izquierda de la pantalla.

El número de parámetros, que se pueden modificar en el editor de materiales es enorme. Pero para nuestra práctica actual, es suficiente con saber lo siguiente. Un material es la envoltura de un objeto, una textura es solo una forma, como esta envoltura puede ser mostrada. Ahora vamos a utilizar una textura fotográfica,para recubrir nuestro objeto. Utilizaremos para este fin, las imágenes que preparamos previamente en **The Gimp**. Nos servirán para aumentar el realismo del objeto, y de referencia, para colocar otros objetos de una forma exacta. Estos objetos, van a ser los botones y otros elementos del modelo, El sombreado "natural" de la propia imagen, puede ser un poderoso aliado, a la hora de lograr una apariencia realista, de una forma muy simple. Es importante procurar, que las sombras no sean ni demasiado duras, ni demasiado pronunciadas, para evitar entrar en contradicciones con la iluminación general de la escena.

Para texturar un objeto, hacemos una ventana nueva y abrimos mediante el ícono de la parte inferior izquierda de la ventana, el editor **UV**. En el menú de la izquierda utilizamos "**Open image**" para cargar nuestra imagen. Después, seleccionamos el objeto, cambiamos al modo de "**edición de vértices**" y pulsamos la tecla "**u"**. Sobre nuestra imagen, en el editor **UV,** veremos que se superpone la malla de nuestro objeto. Las manipulaciones sobre esta malla, no afectan de ninguna forma, a la geometría de nuestro objeto. Solamente se utiliza para modificar la posición, escala e inclinación de la textura sobre el objeto. Es decir, es un mecanismo de control para los materiales **UV**. Podemos trabajar sobre esta malla, con los mismos atajos de teclado del resto de **Blender**. "**a**" para seleccionar o deseleccionar todo, "**s**" para escalar, etc.

Una vez que tenemos texturado el objeto, vamos a hacer que el programa nos lo muestre en tiempo real con texturas fotográficas, brillos, sombras arrojadas, etc. El aspecto, sera similar a como va a visualizarse en le modo de videojuegos. Pero para lograrlo, primero tenemos que cambiar unos pocos valores por defecto del programa. Como se ha comentado al comienzo, la visualización de **Blender** y la del **BGE** no son idénticas, pero hay que intentar que sean lo mas similares posibles. Para ello, en el menú de la parte superior escogemos: "**Game** / **Blender GLSL Materials**". Hay que asegurarse, de que en el elemento del mismo menu "**GLSL Material Setting**" estén activadas todas las opciones, tales como **Light**, **Shaders**, **Shadows**, etc. También es imprescindible que la opción de representación de **Blender** sea "**Textured**".

Figura 4

Ahora, modelamos los botones y otras piezas necesarias para dar por finalizados los modelos. Las técnicas empleadas, serán las mismas que hemos utilizado para el resto. Guiándonos por la imagen de la textura, situamos todos estos elementos, de tal forma que las sombras que proyecten, den la sensación de ser propias.

9- MODELADO E ILUMINACION

En el capítulo anterior, hemos visto como modelar y texturar objetos de forma más o menos simultánea. Ahora, debemos profundizar un poco más en el proceso de texturado, mientras nos vamos introduciendo en el importante tema de la iluminación.

Existen dos tipos principales de texturas. Las procedurales y las fotográficas. Las primeras, se crean a partir de fórmulas matemáticas, y se utilizan por lo general para simular materiales naturales complejos, pero sujetos a ciertos patrones predecibles, como las vetas de la madera, la rugosidad del estuco, el agua o las nubes. Si el algoritmo en el que están basados, incluye al factor tiempo, estos materiales pueden presentar incluso animación. Naturalmente, **Blender** proporciona materiales de este tipo listos para ser utilizados. También, podemos editarlos para adaptarlos a nuestras necesidades. En cuanto a las llamadas texturas fotográficas, estas son, simplemente imágenes de mapa de bits, con las que recubrimos nuestros objetos. Aunque el programa, nos permite trabajar con imágenes de cualquier tamaño, es importante saber que la mayoría de los motores o "**engines 3D**" exigen imágenes cuadradas, con un tamaño en píxeles proporcional a la progresión 8, 16, 32 , 64, 128, 256, 512, 1024, etc.

Las texturas fotográficas por si mismas, no proporcionan buenos resultados. Pero, junto con un buen trabajo de edición de materiales, pueden producir resultados excelentes. Veamos a continuación algunas posibilidades más del editor de materiales. Como sabemos, mediante "**F5**" entramos en el editor de materiales. Bien, la pestaña "**Ramps** / **Colorband**" proporciona un interesante efecto. Este consiste en un degradado del color que escojamos. Queda bastante bien en los metales, y funciona perfectamente en el modo de videojuegos (**BGE**). Se suele utilizar en suelos, mamparos metálicos puertas de acero, y otros materiales similares.

Podemos observarlo en la "Figura 5", en los brillos dorados del suelo y de la pared de la derecha.

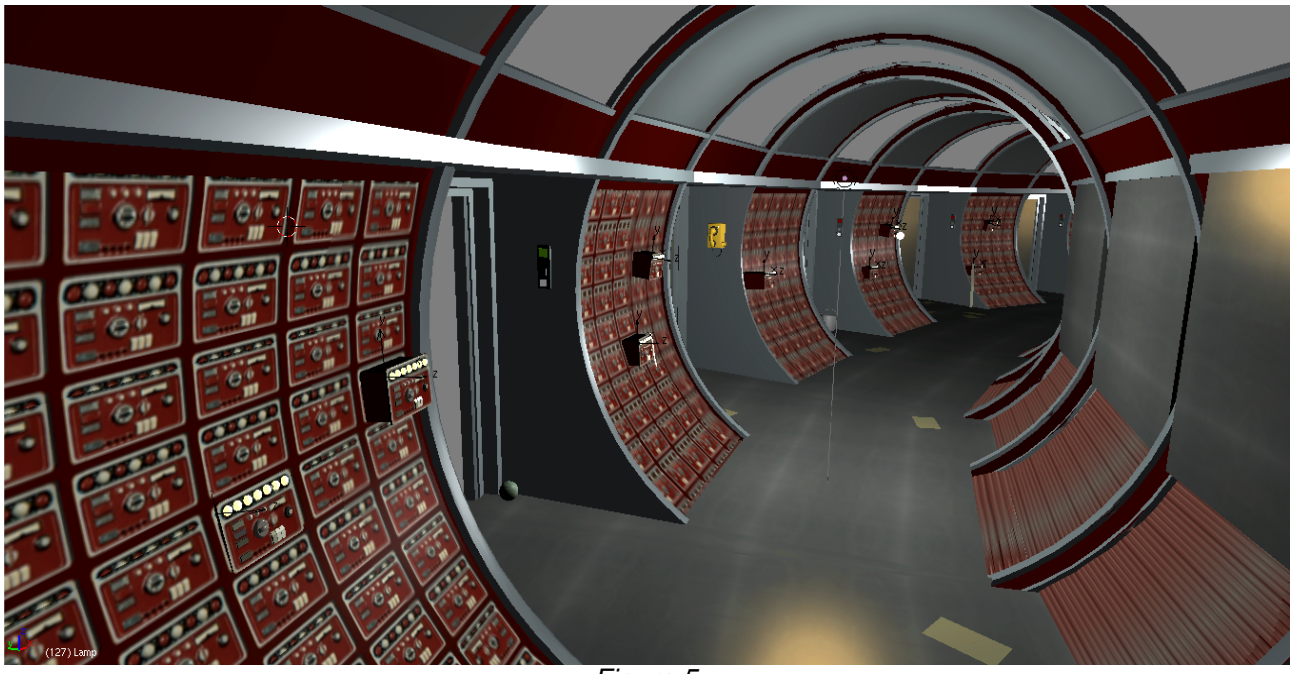

Figura 5

Otro tipo de materiales imprescindibles en un buen juego, son los transparentes. Veamos como lograrlos. En la pestaña "**Links and Pipeline**" abajo, seleccionamos el botón "**ZTrans**" para activar la transparencia, y mediante el botón **A** (**Alpha)** en la pestaña material, indicamos al programa el nivel de transparencia con valores disponibles entre **0** y **1**. Este tipo de transparencia, funciona perfectamente en el modo de juegos, siempre y cuando utilicemos el sistema de materiales **GLSL** (**Open GL**). Ver "Figura 6". Se puede obtener un resultado más interesante, utilizando en la pestaña "**Shaders**" (sombras) la opción "**Fresnel**". Asimismo, es perfectamente posible, cargar una segunda textura fotográfica, en nuestro material transparente. Se visualizara también, con el nivel de transparencia que hayamos escogido. Utilizando imágenes de formato **.png** con fondo transparente, se suelen hacer los íconos, y sistemas de señalización, tales como punteros, flechas, etc.

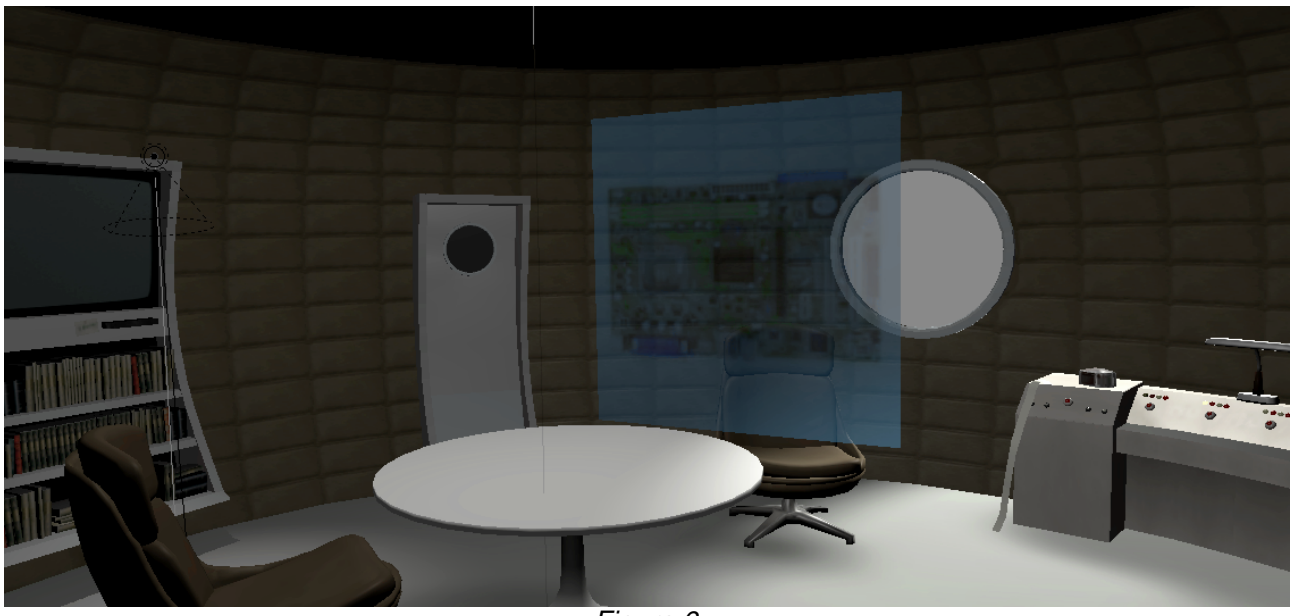

Figura 6

Afortunadamente, las capacidades del **BGE** no se acaban con todo lo visto (aunque ya permitirían hacer un buen trabajo). Vamos a ver a continuación como hacer espejos, o materiales metálicos muy pulidos, que viene a ser lo mismo. Y a dotar, a nuestros materiales de relieve mediante los llamados "**mapas de normales**".

Empecemos por el espejos. En realidad, no resulta muy práctico hacer que el **BGE** calcule toda la escena reflejada en nuestro espejo, ya que la complejidad de los cálculos ralentizaría mucho la simulación. Pero existe un sistema, que proporciona buenos resultados, de una forma mucho más sencilla. Veamos como. Primero, creamos una escena relativamente sencilla, como un pasillo por ejemplo. Modelamos un objeto sencillo para reflejar la escena, como un espejo de pared. Lo mapeamos, con una imagen adecuada, es decir que sea lo mas similar posible a lo que suponemos que nuestro espejo debería reflejar. Después, en la pestaña "**Map imput**", como sistema para aplicar las texturas, en lugar de "**UV**" escogemos "**Recfl**". El objeto parecerá reflejar la imagen, produciéndose además sutiles diferencias de la visualización, en función del ángulo de visión.

Bien, y ahora alguien podría preguntarse: ¿Y de donde sacamos una imagen que resulte apropiada para nuestro propósito? Lo mejor de todo es la respuesta. Renderizando previamente la escena, desde el punto de vista del espejo. Lo que obtengamos, lo retocamos en **The Gimp** para mejorarlo en lo posible y lo aplicamos como textura.

El tema de los **mapas de normales** requiere una explicación. Es un tema interesante, que nos permite tratar, aunque solo sea de pasada, de la evolución del software **3D** en los últimos quince años aproximadamente. Como se puede imaginar, en aquellos días solo se podían utilizar materiales muy

simples. Casi todo parecía hecho de papel o plástico coloreado. Por suerte, pronto se pudieron utilizar texturas fotográficas, y los resultados empezaron a mejorar. Pero el primitivo soporte para imágenes era muy limitado. Las imágenes no podían superar ciertos tamaños, y a menudo, había otras limitaciones, como el modo de color (indexado generalmente). Con el tiempo, las cosas mejoraron un poco más, hasta que al fin se pudieron utilizar imágenes de buena calidad para nuestros materiales.

Más tarde, se logró un método para producir relieve llamado "**Bump"** o "**Mapa de relieve**". Este, consistía en una imagen de mapa de bits (con solo dos niveles de color, el negro y el blanco), que mapeada como una segunda textura, producía un acabado rugoso. Por fin, disponíamos de un método, para simular de una forma realista, remaches, soldaduras, materiales repujados, vidrios grabados, etc.

Pues bien, un "**Mapa de normales**" es como una puesta en escena más avanzada, de este mismo concepto. Utiliza una imagen **RGB** a todo color, y es capaz de producir relieve en los tres ejes (**x**,**y**,**z**) partiendo de pequeñas desviaciones de posición, de los tres canales de color de la imagen (**Red**, **Green** y **Blur**).

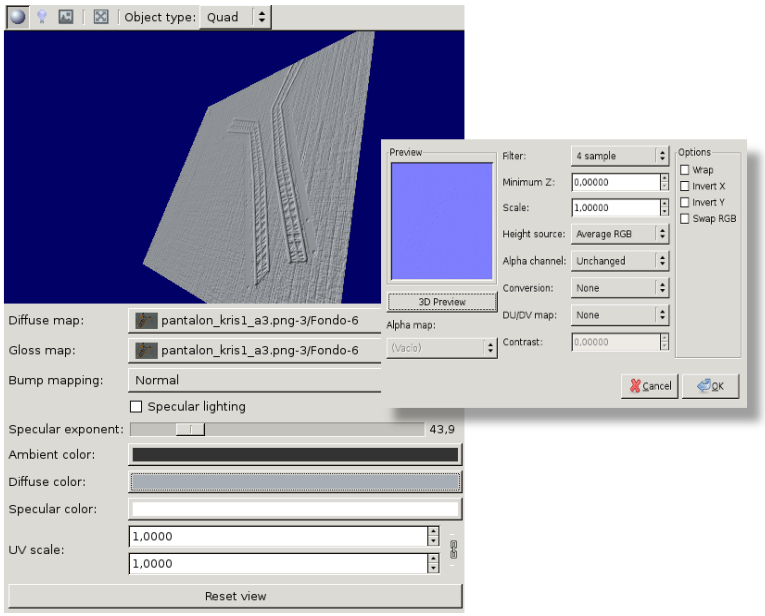

Figura 7

Para poder utilizar un mapa de normales, primero debemos disponer del propio mapa. Para ello, utilizamos **The Gimp**. Mapeamos en **UV** nuestro objeto con una textura fotográfica. Hecho esto, la abrimos en el **Gimp**, y generamos el mapa mediante: **Filtros** / **Mapa** / **Normalmap**. Si no dispones de este filtro por defecto en **Gimp**, lo tendrás que instalar previamente. En **Gnu/Linux,** el procedimiento es bastante simple. Basta hacer una búsqueda entre los paquetes disponibles, para nuestra distribución e instalarlo. El filtro es capaz de visualizar la textura con relieve en **3D** ("Figura 7**"**).

Entre los parámetros que es posible modificar, el más importante es "**Scale**". Este representa la altura de la rugosidad. Aunque también se puede modificar esto en **Blender**, es mejor hacerlo en **Gimp,** ya que el filtro nos permite un control mas preciso. Un valor de 15.00000 suele ser adecuado, pero depende de la textura y de lo que pretendamos conseguir.

Veamos ahora, como incorporar el mapa de normales a un material en **Blender**. Pulsamos "**F5**" para trabajar con el editor de materiales, y en la pestaña "**Texture**" añadimos una segunda textura en el mismo material. Cargamos una imagen, que "casualmente" sera el mapa de normales que creamos en **The Gimp**. Los mapas de normales tienen un color azulado. Pero, por el momento, **Blender** aun no

sabe que esta textura es un mapa de normales, así que tenemos que decírselo de algún modo. Ver “Figura 8”, a continuación.

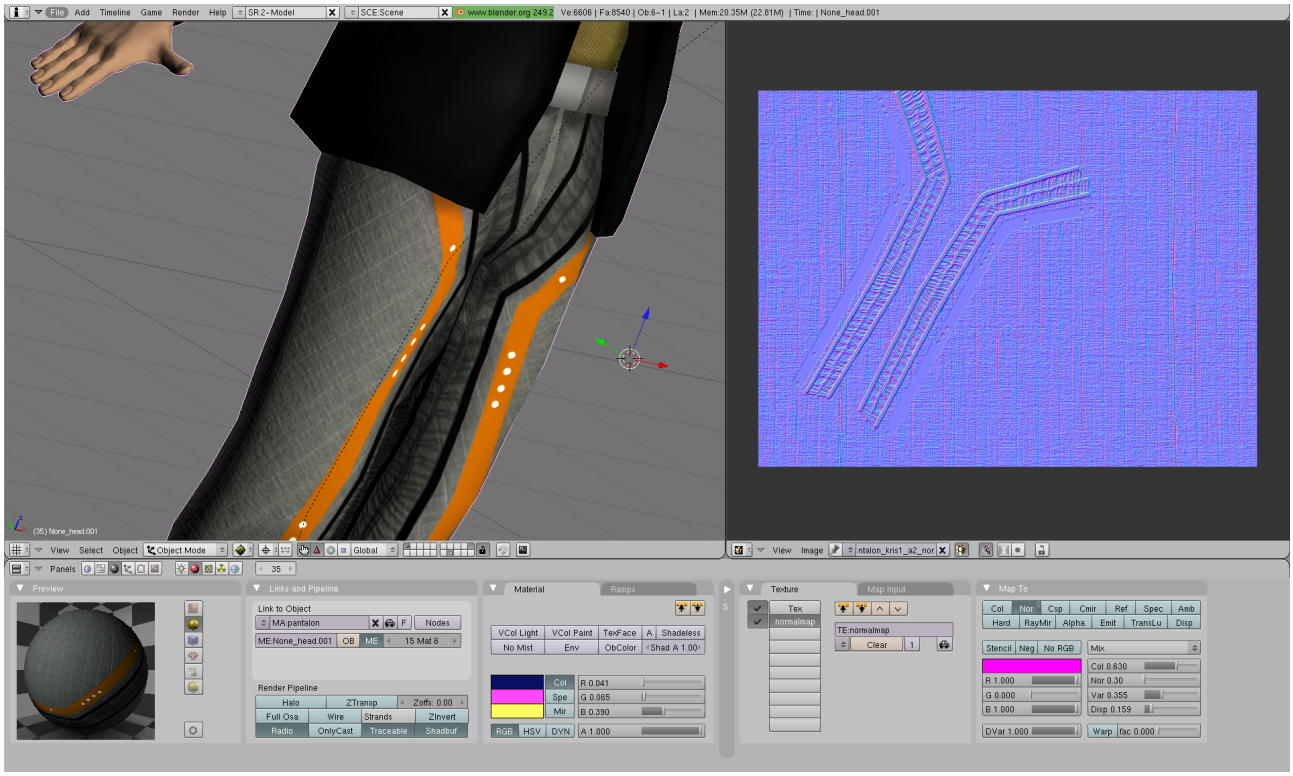

Figura 8

Como se ha comentado previamente, en la pestaña “**Texture**” creamos una textura nueva para el mismo material. Mediante “**u**” la mapeamos en el “**editor UV**”, (en la imagen arriba a la derecha) y en la pestaña “**Map To**” seleccionamos “**Nor**”. Finalmente, en “**Texture Buttons**” (**F6**) en la pestaña “**Map image**” seleccionamos “**Normal Map**”. Ver “Figura 9”.

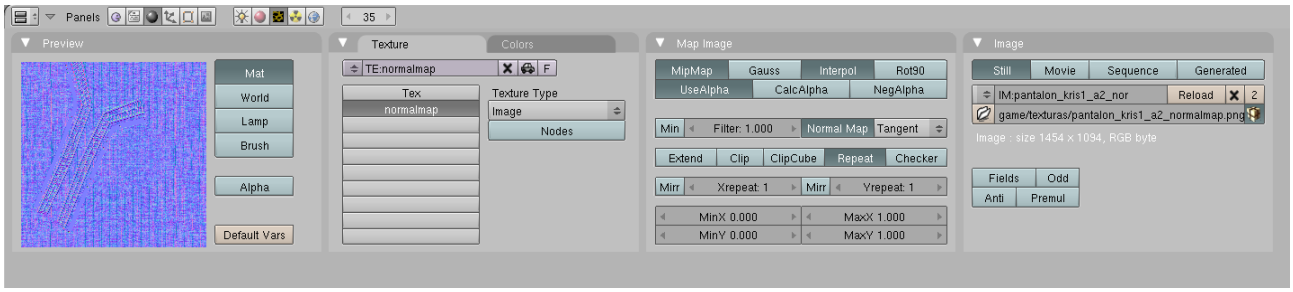

Figura 9

Ahora, que ya conocemos las principales técnicas de edición de materiales, vamos a ver como iluminar nuestras escenas. Esta es, una fase de nuestro trabajo de gran importancia. Una escena bien modelada y texturada se arruina con una mala iluminación.

De hecho, un trabajo de iluminación sobresaliente, puede salvar una escena deficiente. A menudo, no se concede a la iluminación la importancia que realmente tiene. Esto tiende a ocurrir en muchos cursos, donde no suele haber tiempo suficiente para profundizar en todas las materias.

La iluminación está muy relacionada con el **Arte**. Esta, aporta al juego un carácter o aspecto determinado, que debe estar armonizado con el diseño estético del mismo. Pensemos en el mundo del cine. Como se sabe, algunas películas son tenebristas. Otras, por el contrario poseen una iluminación intensa. Las hay muy saturadas de color, y otras prácticamente insaturadas. El aspecto visual puede ser muy nítido o borroso y desvaído. Todo ello, tiene el mismo efecto en un juego.

Por otra parte, es aconsejable evitar las iluminaciones excesivas e impersonales, a las que tenemos tendencia los técnicos. Esto, resulta tan científico, como poco interesante. En mi opinión, las cosas deben visualizarse en función del ambiente estético del juego. Y siempre hay que procurar que este sea rico e interesante. La iluminación puede ayudar mucho a ello, ya que es uno de los elementos más expresivos en el mundo de la imagen.

Dicho esto, parece un buen momento para pasar de la teoría a la práctica. Lo primero que debemos saber, es que la iluminación es un aspecto crítico, desde el punto de vista del rendimiento del **BGE**. Como consecuencia de ello, no es buena idea utilizar más de tres o cuatro focos en una escena ya que, de hacerlo, la simulación perdería mucho rendimiento.

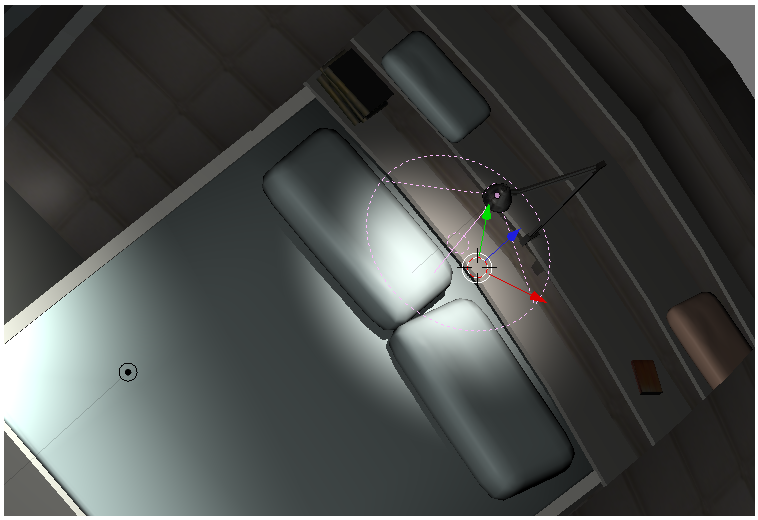

Figura 10

Normalmente, se utilizan como máximo tres focos. El primero, suele ser un punto de luz de tipo "**Spot**". Este tipo de foco, es el único en **Blender** capaz de producir sombras proyectadas en el **BGE**. Estas, son aquellas (ver Figura 10) que unos objetos proyectan sobre otros. El segundo foco, suele ser de tipo "**Lamp**". Este, es un punto de luz que emite en todas direcciones. Se utiliza como relleno, debido a que los focos de tipo "**Spot**", solo iluminan una parte de la escena. Su iluminación, se dice que es "dura". La combinación de ambos tipos de luz, proporciona buenos resultados. Si el espacio de nuestro escenario es muy grande, nos veríamos obligados a incluir un foco "**Lamp**" adicional. No es buena idea prescindir del tercer foco ,aumentando la intensidad de los dos primeros. Pero tampoco lo es, iluminar una escena con un número excesivo de puntos de luz.

Veamos a continuación, los pasos que hay que seguir en **Blender**, para que las sombras proyectadas se puedan visualizar en el **BGE** (Figura 10).

Comenzamos creamos una escena simple, y la iluminamos con dos focos, uno de tipo "**Lamp**" y otro de tipo "**Spot"**. Como actor, podemos servirnos de un cubo. Seleccionamos el foco "**Spot**" y pulsamos **F5**. Desde esta parte del programa, podemos editar las propiedades del foco de luz que tenemos seleccionado. La energía y distancia a la que actúa, se ajustan mediante los botones "**Energy**" y "**Dist**" respectivamente. El color se puede modificar en los botones inmediatamente inferiores.

Lo que nos ocupa en estos momentos, se realiza desde la ventana, a la derecha, llamada "**Shadow and Spot**". Las luces de tipo "**Spot**", permiten utilizar halos en los bordes del área (habitualmente circular) que iluminan. El tamaño y suavidad de este halo, entre otras cosas, puede ser ajustarse desde esta ventana. Este efecto, funciona en el **BGE**. En esta misma ventana, a la izquierda, disponemos de dos botones "**Ray shadow**" y "**Buf shadow**". El primero es para utilizar técnicas de trazado de rayos y no funciona en el **BGE**. El segundo si lo hace. Por tanto, Lo activamos.

Eso es todo. Si hemos configurado correctamente **Blender**, podremos ver las sombras proyectadas

en la ventana **3D**. Si no es así, se debe seguramente, a que el foco de luz no esta a una altura adecuada. Así que, lo desplazamos hasta que finalmente se visualicen.

Es también buena idea, utilizar luces de colores. Las luces calientes (rojas, naranjas y amarillas) ayudan a resaltar los objetos del primer plano. Las luces frías (azules, violetas y verdes) producen una sensación de gran profundidad, así que las utilizamos en los fondos, Una buena combinación de colores, puede volver atractivo un escenario gris y sin vida. Experimenta cuanto se te ocurra, ningún tutorial podrá sustituir nunca a tus propias investigaciones.

10- MODELANDO UN PERSONAJE EN BAJA RESOLUCION

Con todo visto hasta ahora, deberíamos poder modelar, texturar e iluminar una escena. Llegados a este punto, es interesante que modeles tu propia escena. Es conveniente que esta sea simple, pero que este resuelta de un modo correcto. Lo mejor es una habitación con algunos muebles sencillos. Intenta hacer un espejo, y algunos elementos de metal, cristal y madera. La arquitectura, generalmente da más facilidades, que el modelado orgánico. Recuerda, que es conveniente utilizar biseles en las aristas de los objetos, que no haya vértices duplicados y que los objetos no se solapen. El objetivo es lograr una malla continua, no un agrupamiento de objetos superpuestos.

Ahora vamos a comenzar a modelar nuestro primer personaje. Para ello, tenemos que hacer, en primer lugar, un montaje fotográfico como el de la “Figura 11”. Como se puede observar, en la parte izquierda esta situado nuestro personaje de espaldas, con las piernas un poco separadas y los brazos extendidos, pero sin llegar a estar en cruz. Esta, es una postura típica en los personajes **3D**. Esta postura, es por así decirlo, la extensión máxima del modelo. La creamos ya así, y posteriormente, mediante la animación, produciremos posiciones menos forzadas. Podríamos también, forzarla más, pero los resultados podrían no ser muy buenos. En la parte derecha de la imagen, vemos a nuestro modelo de perfil.

Figura 11

Las técnicas, para crear imágenes de este tipo son muy variadas. Algunas imágenes, pueden encontrarse en **internet**, pero claro, las tiene todo el mundo y seguramente no sean adecuadas para nuestros propósitos. Podemos utilizar la cámara fotográfica y convencer a algún amigo para que pose. Descargar de un repositorio fotográfico, o combinar fragmentos de imágenes de fuentes variadas. Es lo que normalmente hago yo mismo. Incluso se pueden pintar a mano.

Ahora, ejecutamos **Blender**. Hacemos dos ventanas nuevas, que ocupen cada una más o menos la mitad de la pantalla. Ver “Figura 12”. Creamos un cubo nuevo y cambiamos al modo de edición mediante el tabulador, y en la pestaña "**Modifiers**" activamos la opción de modelado en espejo:

/ Add Modifier / Mirror

Aparentemente, no ha pasado nada. Pero si movemos el objeto, seleccionando mediante la tecla "**a**" , y desplazándolo hacia la derecha, veremos que aparece otro cubo a la izquierda. Este, es la réplica automática del primero. Ahora, un pequeño detalle muy importante. Nuestro plan, es modelar solo la mitad del personaje partiendo de un solo cubo, pero que la otra mitad, se replique automáticamente. Para que las cosas vayan bien, el eje de simetría de la figura tiene que estar exactamente entre los dos cubos. Para lograrlo, hacemos “**zoom**” en esta zona y movemos nuestro cubo hasta que coincida exactamente con la réplica. Normalmente en **Blender**, se puede hacer “**zoom**” con la rueda del ratón. Para centrar la vista se utiliza la tecla "**c**".

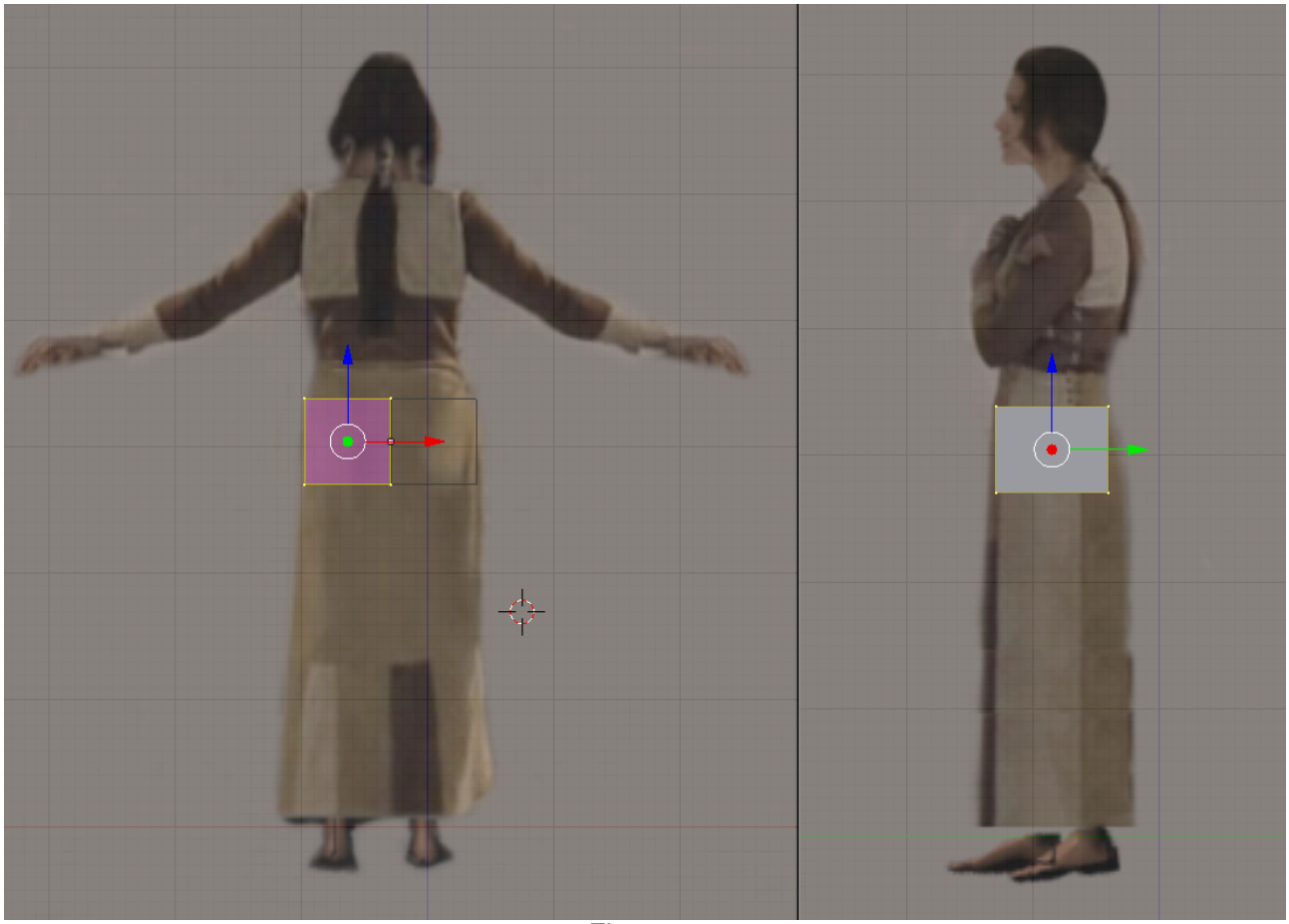

Figura 12

Después extrusionamos el cubo desde la cintura hacia arriba. Extrusionar, consiste en seleccionar los vértices de la parte superior del cubo, y desplazar una copia de los mismos. De este modo, se crean nuevos volúmenes. La técnica es bastante sencilla. Seleccionamos lo que queramos. Es aconsejable utilizar la tecla "**b**" ya que nos permite hacer grandes selecciones mediante un rectángulo. Más tarde, con la tecla "**e**", realizamos la extrusión propiamente dicha. Podemos extrusionar tantas veces como queramos. Y a la vez, escalamos, y desplazamos los vértices como nos interese para ajustarnos a

nuestra imagen de fondo. En la ventana de la derecha, disponemos del perfil de nuestro personaje. La tecla "**3**", selecciona esta vista. Es imprescindible, que ajustemos los vértices al modelo, al menos en las vistas frontal y lateral.

Al final, obtendremos un torso parecido al de la “Figura 13”. La verdad, es que se parece a una armadura medieval. Pero no debemos de asustarnos por ello. Hay que tener en cuenta, que el personaje se recubrirá finalmente, con un material muy detallado, que mejorará mucho su aspecto. Además, recordemos que este tipo de personajes (en baja), están diseñados para visualizarse a una cierta distancia, y no deben emplearse en primeros planos.

Figura 13

Hemos visto antes, como seleccionar vértices con la tecla “**b**”. Para hacer lo mismo con vértices independientes, se seleccionan simplemente, con el botón derecho del ratón. Se añaden a la selección previa, con la tecla de “**mayúsculas**”. Se restan, haciendo “click” dos veces con el botón derecho del ratón. También, es posible editar en modo de aristas y en modo de caras. Para ello, disponemos de unos íconos en la parte inferior de la ventana de **Blender**. Utilizamos un modo u otro, en función de nuestras necesidades.

Si no obtienes un resultado óptimo a la primera, vuelve a intentarlo. El modelado de personajes es todo un arte, y la práctica ayuda mucho a mejorar los resultados. Comprende, que estas creando un personaje completamente original, no adaptando un modelo preexistente. Los resultados, si persistes, serán algo único y especial.

Ahora, vamos a continuar modelando los brazos. En la “Figura 14”, se puede observar el modelo en modo de edición, con los brazos terminados. Para poder extrusionar los brazos partiendo del torso, primero debemos de eliminar algunas caras. En la vista de perfil, seleccionamos aquellas caras desde las que parte el brazo. Para saber cuales son estas, nos fijamos en la imagen de fondo, que estamos tomando como referencia. Después las borramos, de tal forma que ahora podremos ver el interior del torso.

Aprovechando esta circunstancia, nos fijamos en algunas caras que hay en el interior del torso. No nos sirven para nada, y solo podrían provocarnos problemas, así que, las eliminamos también. Entonces, en modo de edición de aristas, seleccionamos el borde de la abertura que hemos realizado para el brazo. Finalmente, extrusionamos las veces que sean necesarias, hasta llegar al final del mismo.

Como continuamos trabajando con el modificador "**Mirror**" se habrá creado un brazo simétrico al que hemos modelado manualmente. (“Figura 14”).

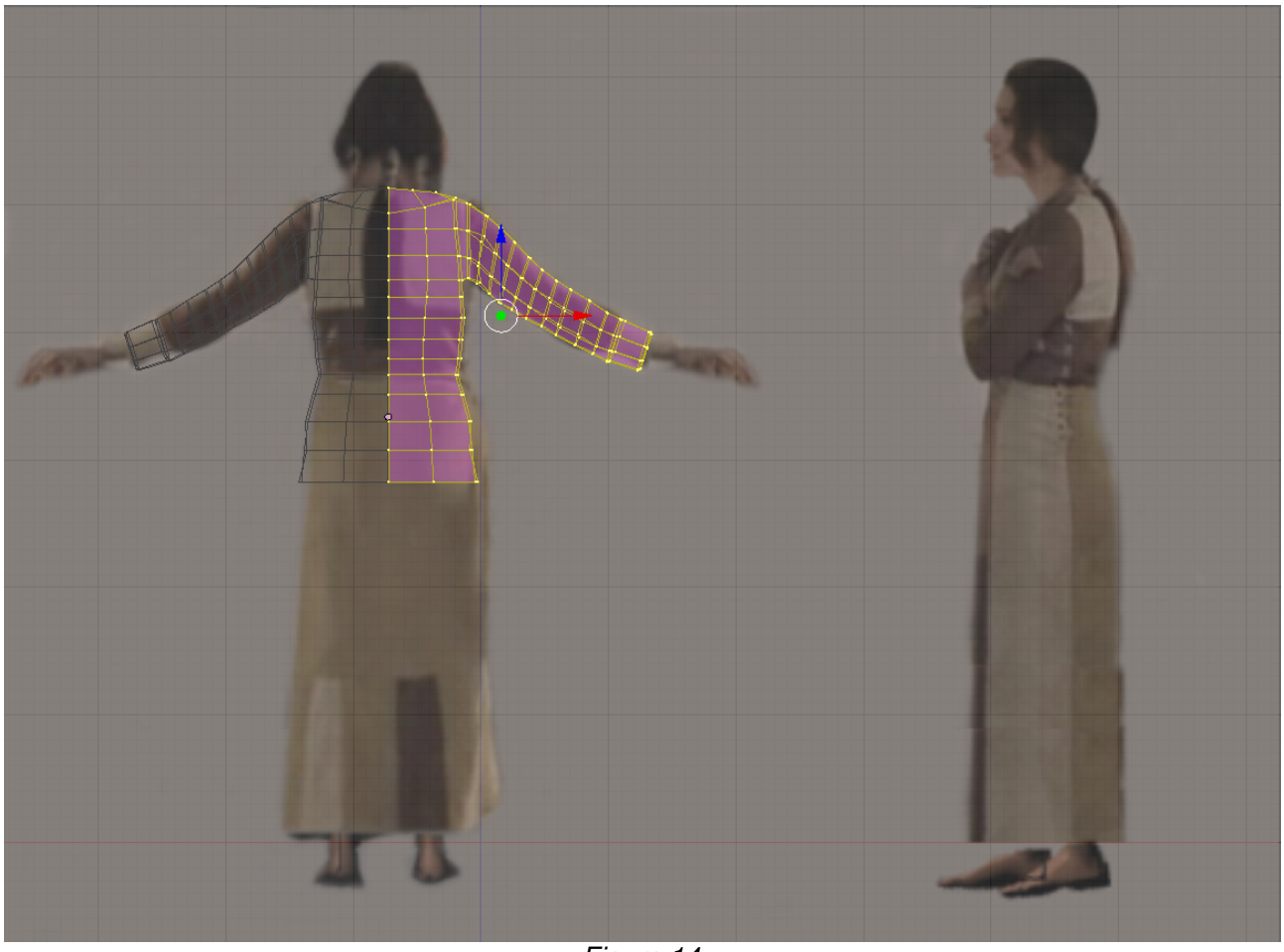

Figura 14

El procedimiento para crear las piernas es similar. Con algunas pequeñas diferencias. Es preciso vaciar los huecos para las piernas dejando una separación entre ellos . Más tarde, seleccionando las aristas de los huecos de las piernas, vamos extrusionándolas. En esta etapa del modelado, es muy importante conceder mucha atención a la vista de perfil. En cuanto a la articulación de la rodilla, es aconsejable, que haya más vértices en esta zona, para facilitar su animación. Hay quienes, optan por hacer un triangulo en el perfil, para que la articulación se flexione mejor.

Finalmente, seleccionamos toda la figura, y en modo de edición activamos el botón "**Set Smoth**" o sea suavizar. Ahora, veremos nuestro modelo con formas suaves. Aprovechamos para depurarlo.

A veces, es preciso no respetar la imagen de fondo, para obtener un mejor resultado en **3D**. Hecho todo lo cual, nos faltan la cabeza, las manos y los zapatos para terminar. Estos elementos requieren un modelado mas delicado. Para modelar cabezas existen programas especiales.

Faceworx es uno de ellos. Este, nos permite adaptar una cabeza genérica en **3D**, partiendo de dos imágenes de referencia, una de frente y otra de perfil.

Estas imágenes, sirven por una parte como texturas **UV**, y por otro, como referencia para que podamos adaptar una cabeza "**de canón**" a las particularidades de nuestro modelo.

Este programa, nos permite modelar mediante tiradores, con la forma de los elementos básicos de la cabeza humana. Es decir, ojos, boca, cejas, orejas, etc. Basta ajustar estos a las imágenes, para obtener resultados muy convincentes ("Figura 15").

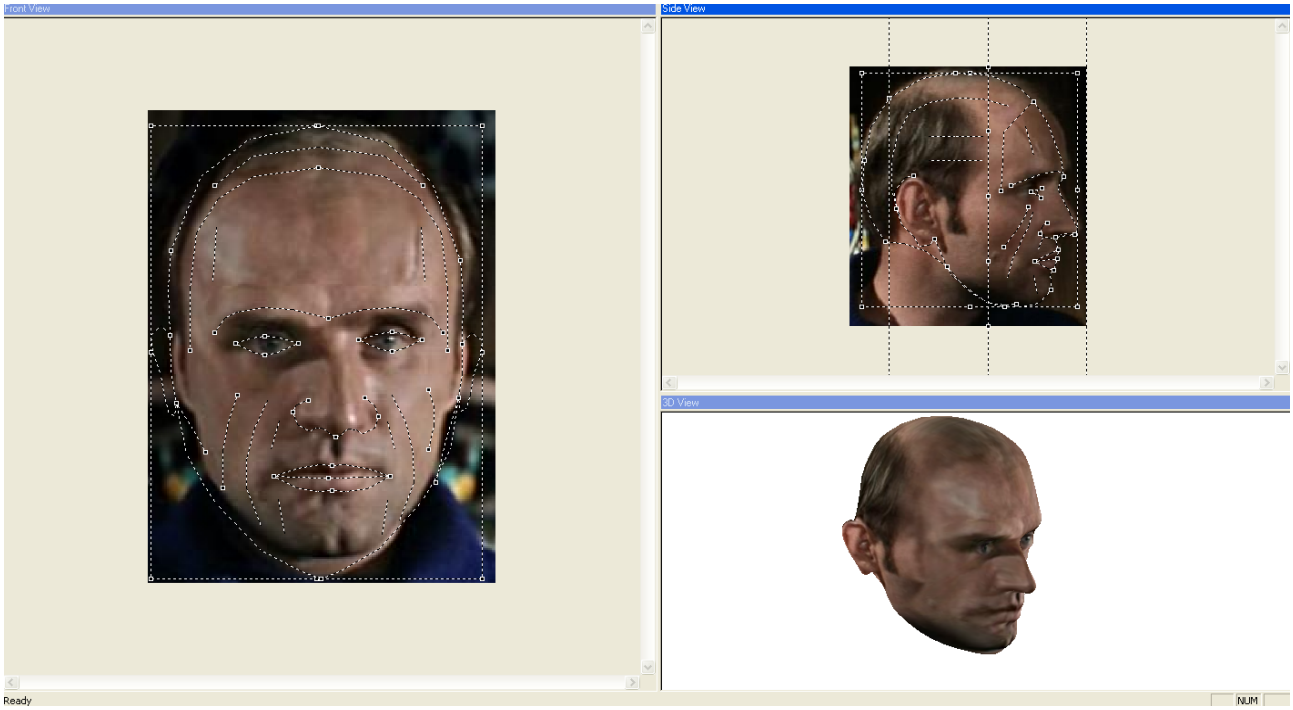

Figura 15

Las manos, se pueden modelar en **Blender** de un modo similar al cuerpo. Aunque hay que admitir, que suponen un esfuerzo considerable. Si estamos ansiosos por terminar, podemos buscar otras opciones. El programa libre, "**Makehuman**" es una buena alternativa.

El modelo humano básico, incluido por defecto de este programa, presenta un gran detalle, sin una poligonización excesiva. Lamentablemente, los personajes carecen de vestidos, por lo que normalmente no terminan de servirnos. La cabeza, tiene ojos independientes, boca, dientes e incluso lengua. Ver "Figura 16". Las manos, están muy bien modeladas. "**Makehuman",** proporciona con sus modelos, esqueletos perfectamente integrados en los mismos. La exportación a otros programas se realiza mediante el formato "**.obj**".

Afortunadamente, **Blender** es capaz de importar el formato "**obj**" sin problemas. En el caso de las cabezas modeladas con **Faceworx**, ni siquiera perderemos la textura **UV**. Ello es una gran ventaja, ya que lo normal, al reutilizar objetos entre programas es perder ,al menos, las texturas. Son también muy típicos otros tipos de errores, como la pérdida de la escala o la modificación de la geometría.

Continuamos. Importamos un cuerpo de "**Makehuman**" de este modo:

File / **Importar** / **Obj**

En modo de edición de vértices, seleccionamos solamente las manos. Invertimos la selección y borramos todos los demás vértices. Ahora volvemos a seleccionar las manos y las desplazamos y modificamos para adaptarlas a nuestro modelo.

Hacemos otro tanto, con la cabeza que modelamos con anterioridad en “**Faceworx**” y ya tenemos nuestro personaje completo. Mediante "**Ctrl j**", podemos unir todas las partes en un solo objeto. El resultado es, el de la "Figura 17". Pero esta no es una forma muy perfecta de modelado. Algunas partes, como las manos, se superponen al cuerpo. Recordemos que nuestro objetivo, es lograr una malla limpia y continua, sin superposiciciones, ni elementos inútiles.

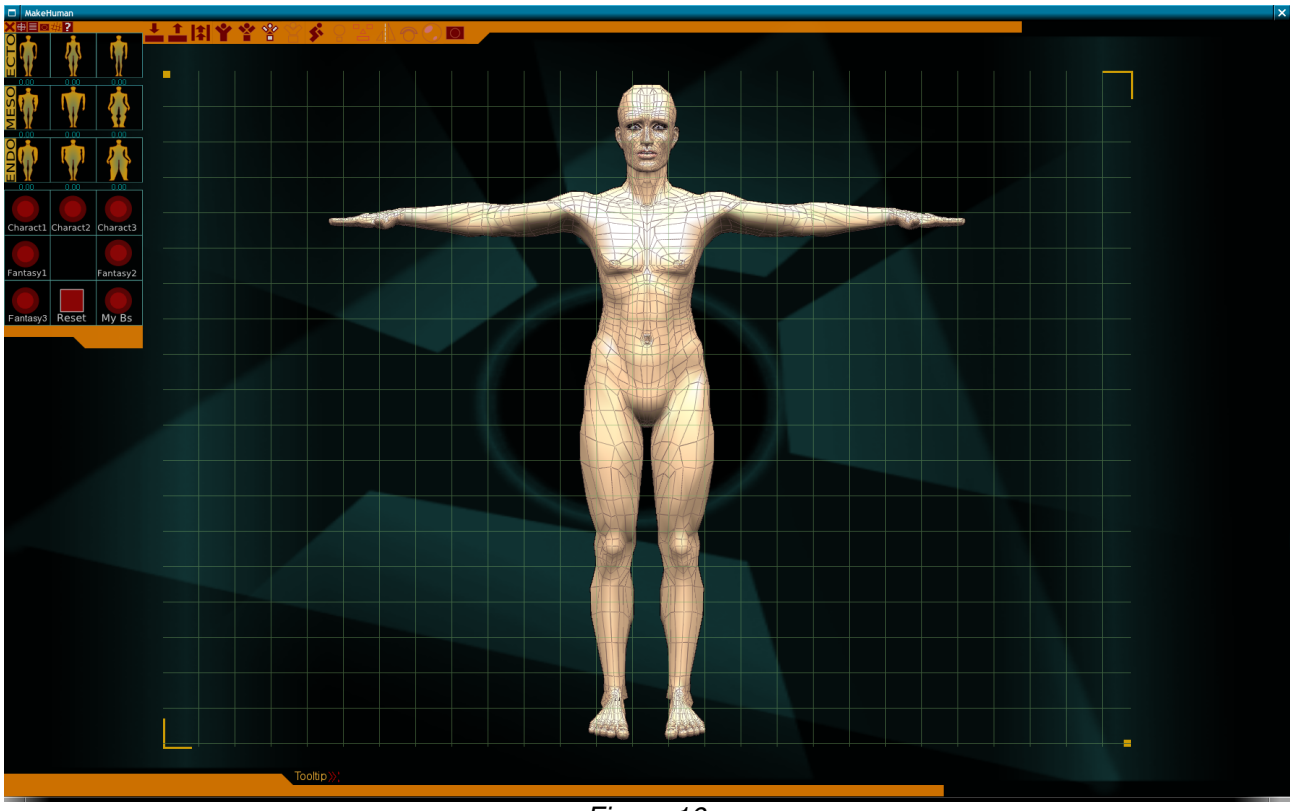

Figura 16

Lo que se debe hacer, es ampliar mucho la vista **3D**, y pacientemente, formar caras nuevas con vértices, tanto de la mano como del brazo. hasta llegar a cerrar la malla totalmente, manteniendo una forma coherente con nuestro diseño. Los vértices que queden fuera de la envolvente de la figura, se eliminan. Hay que hacer lo mismo con la cabeza, aunque previamente hay que dotarla de cuello, pues “**Faceworx**” no nos lo facilita. El cuello, lo modelamos como casi todo hasta ahora. Seleccionamos el nacimiento del cuello y extrusionamos, modelándolo con un poco de habilidad.

Nuestro personaje está básicamente terminado. Pero es posible mejorarlo bastante. Existen muchos modos de perfeccionarlo, pero no todos los métodos son adecuados, para un personaje en baja resolución. Lo más práctico suele ser trabajar sobre las texturas **UV**. Normalmente, se genera una sola textura fotográfica, con todas las texturas que utiliza el personaje. La posición de cada fragmento de la textura es irrelevante, siempre y cuando estos no se solapen entre sí.

Más tarde, seleccionando el personaje completo, ajustamos cada fragmento de la malla, a nuestra nueva textura. De este modo, es muy sencillo modificar el aspecto de nuestro modelo, simplemente retocando la textura en un editor de gráficos.

Como se puede imaginar, cambiar el color de la ropa, o algo similar, es ahora bastante fácil. También puede ser interesante, ayudar al modelado de la figura, oscureciendo la imagen en algunas zonas. Esto, puede realizarse con un pincel de baja opacidad en **The Gimp**, resaltando hábilmente los bordes de cada fragmento de la textura.

Pero **Blender** es un programa muy avanzado, y nos depara una grata sorpresa: Permite la pintura

directa sobre el objeto **3D**. Para ello, dispone de un modo especial de pintura. Para cambiar a este modo seleccionamos “**Texture Paint**”. Para pintar, utilizamos un pincel aplicándolo sobre la parte del modelo que queramos, mediante el botón izquierdo del ratón. El pincel como tal, es invisible.

Para obtener buenos resultados, es imprescindible editar los pinceles. En modo de edición (**F9**), desde el panel “**Paint**”, modificamos su tamaño, color y opacidad. Con un poco de práctica y paciencia, se pueden obtener acabados asombrosos. En este tipo de edición, la preparación artística tradicional (dibujo y pintura) es determinante. Si no tienes este tipo de formación, puedes intentarlo de todas formas. Algunas personas, poseen aptitudes naturales para este tipo de cosas. En cualquier caso, con un poco de práctica seguramente terminarás por producir resultados aceptables.

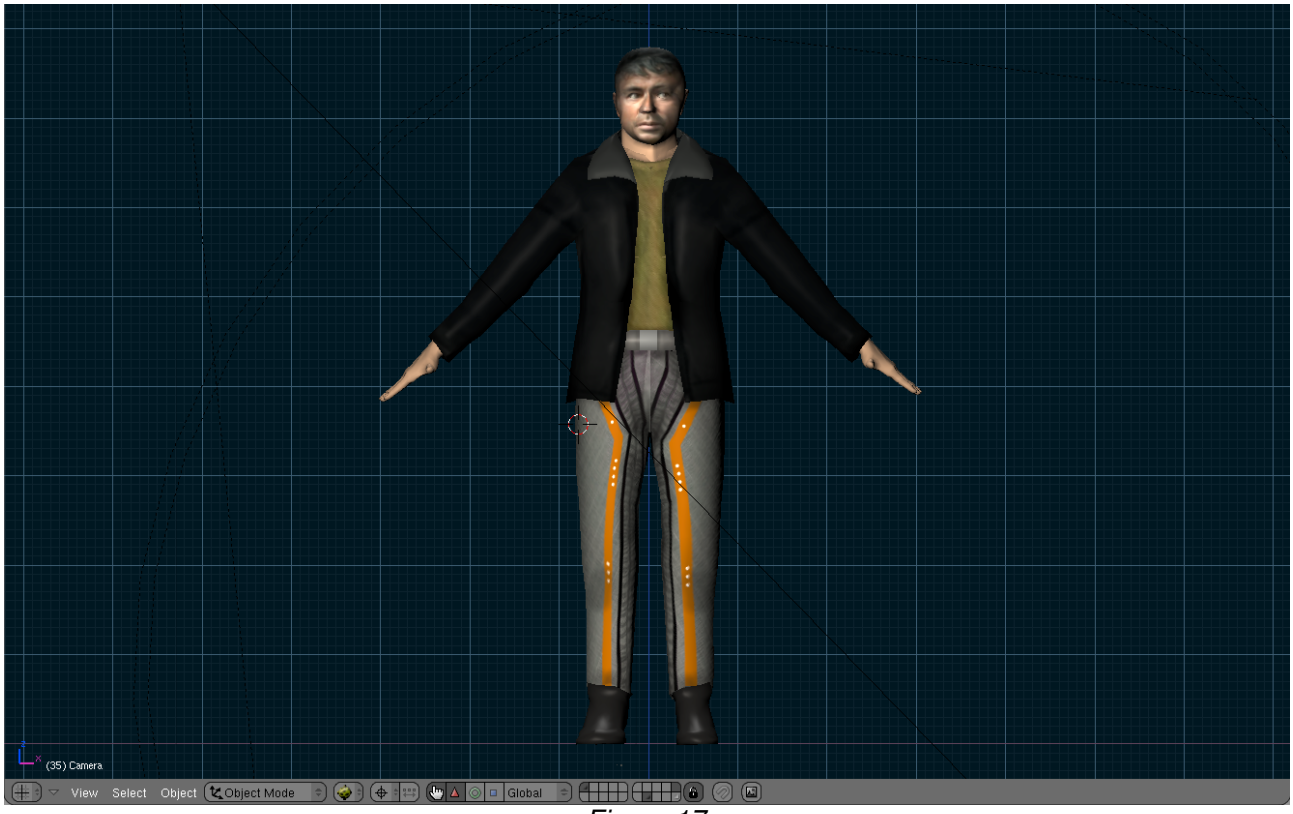

Figura 17

Si mediante la tecla “**p**”, ejecutamos el modo de videojuegos, veremos que nuestro personaje se visualiza exactamente igual que en la vista **3D** de **Blender**. Este era nuestro objetivo. El programa nos ofrece un modo de "**escultura**" adicional, con el cual se pueden realizar modelados extraordinarios. Lamentablemente, requieren mallas muy finas, y no se pueden realizar modelos adecuados para el **BGE**.

11- ANIMACION BÁSICA

Llegados a este punto, se supone que seriamos capaces de modelar, texturar e iluminar en espacio, y poblarlo con nuestros propios personajes. Aunque esto es divertido en si mismo, resultaría más vivo e interesante, si ocurriera algún tipo de acción o evento. Los personajes "congelados", hacen que perdamos interés por ellos, en poco tiempo. Y algo parecido viene a suceder con los escenarios. Pero todo puede cambiar, si animamos a nuestros actores. Llamamos actores, no solo a los personajes con forma humana, sino a cualquier objeto que sea programado, para realizar algún tipo de acción en la simulación.

Una cámara, que movemos para visualizar el entorno es un actor del juego. Un foco de luz, que cambia de intensidad a lo largo del tiempo, también es otro actor. Este tema es muy complejo, por ello vamos a empezar por lo mas sencillo. Vamos a animar la cámara. Nos sera muy útil, ya que ya va siendo hora de ver nuestro escenario, desde una vista subjetiva, y de desplazarnos por él como queramos.

Seguramente, ya dispongamos de una cámara en nuestra escena. La vista de cámara, se visualiza en la ventana **3D** mediante la tecla "**0**". Si no nos gusta lo que vemos, cambiamos la posición de la cámara desde cualquier vista. La Seleccionamos y mediante la tecla "**F4**" vamos al editor de videojuegos de **Blender**.

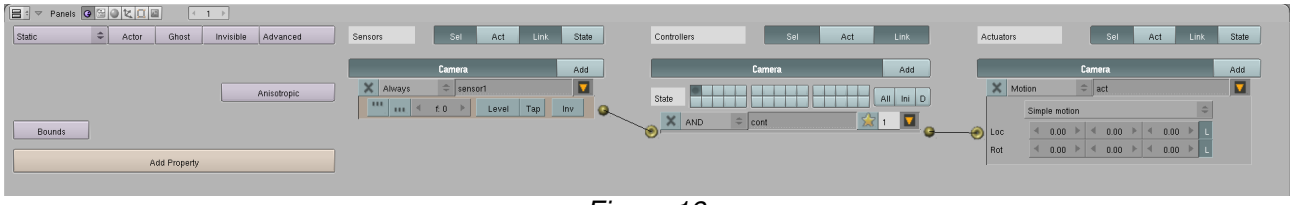

Figura 18

Bien. La pantalla, horizontalmente, se divide en cuatro grupos de botones. Ver "Figura 18". El primero desde la izquierda, nos sirve para escoger las características físicas del actor y su comportamiento en la simulación. Es decir, si se ve afectado por la física del juego (comportamiento dinámico), o no (estático). Si es rígido o blando. Su peso aparente, su capacidad para atravesar o colisionar con otros objetos, etc.

Los otros tres paneles, se utilizan para definir con precisión, aquello que nuestro actor va a hacer. Se denominan “**sensores**”, “**controladores**” y “**actuadores**”. Un sensor, viene a ser la entrada de datos de nuestro objeto o actor. Esta, puede ser un conjunto de datos previos o un periférico, como el teclado o el ratón. Los **controladores**, hacen referencia al tipo de datos que se van a emplear, y los **actuadores**, a las acciones que nuestros actores van a realizar.

No es tan difícil como aparenta. Básicamente, todo ello se reduce a escoger un tipo de sensor y dotarle de un actuador adecuado. Veamos un ejemplo de ello, animando una cámara. Para ello, seleccionamos la cámara, y abrimos el editor de videojuegos (**F4**).

Por defecto, en el panel de la izquierda veremos que esta seleccionado "**Static**". Es adecuado, en este caso. La cámara, no sera afectada por la física del **BGE**. Si seleccionásemos "**Dynamic**", esta caería hasta el suelo, por efecto de la gravedad, y no es interesante que ello suceda en este ejercicio.

En el panel "**Sensors**", mediante "**Add**", creamos un sensor nuevo. Hacemos lo mismo, en los paneles "**Controllers**" y "**Actuators**". Después, unimos los tres bloques de datos resultantes con hilos, como puede verse en la "Figura 18". Esto se hace, con el botón izquierdo del ratón, arrastrando desde los pequeños círculos, en la parte derecha de cada bloque de datos.

Ahora, escogemos el tipo de sensor. La opción "**Always"**, por defecto en el programa, significa que la acción se realice siempre. En nuestro caso, elegimos "**Keyboard**" o sea teclado. El programa, nos permite pulsar la tecla que queramos, guardándola como sensor. El siguiente panel, hace referencia al tipo de datos, por defecto "**AND**". Es adecuado, en esta situación.

El panel siguiente "**Actuators**", nos va a permitir, asignar acciones a nuestra cámara. En este caso, vamos a utilizar un conjunto de acciones muy sencillas. Adelante, atrás, izquierda y derecha, mediante los cursores del teclado. De momento, solo tenemos un sensor. Le asignamos el cursor "adelante", y hacemos otros tres sensores nuevos para las teclas de cursor "Atrás" "Izquierda" y "Derecha". En el panel "**Actuators**", tendremos que crear tres actuadores adicionales. Y unir con hilos, cada actuador con su controlador y su actuador, como puede verse en la “Figura 18”.

Pero todavía nos falta, estudiar con un poco más de detenimiento, el panel "**Actuators**". Vamos a mover la cámara. Seleccionamos, por ejemplo, "**sensor 1**", que nos servirá para mover la cámara frontalmente. Se presupone, que este sensor está relacionado mediante un hilo con su Controlador, y su Actuador. En el panel "**Actuators**", seleccionamos "**Motion**", desplegando un menú, tal y como puede observarse en en la "Figura 19" a continuación.

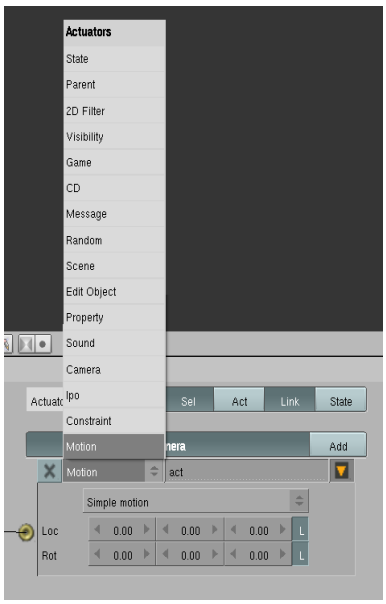

Figura 19

Como resultado de esta elección, ahora aparece una tabla con dos filas y tres columnas. Las filas, son "**Loc**" y "**Rot**" y las columnas son, de izquierda a derecha, los ejes **x**, **y**, z, respectivamente.

Escribimos **-0.03** en "**Loc z**". O sea, la tercera posición de la primera fila (**Loc**). Los valores tienen que ser bajos para lograr movimientos suaves. Ver "Figura 20".

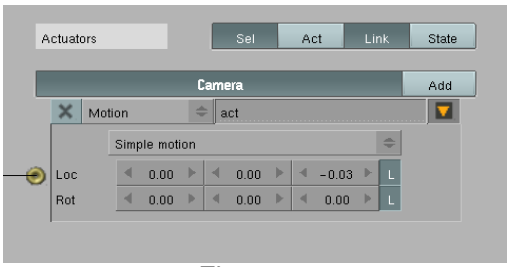

Figura 20

Si ahora, ejecutamos el **Blender game engine** (**BGE**), mediante la tecla "**p**", podremos mover la cámara hacia adelante, mediante la tecla de cursor "**adelante**". El resto del ejercicio, consiste en repetir esta técnica con los otros tres sensores, para lograr finalmente gobernar la cámara en los dos ejes (**X**,**Y**), con las cuatro teclas de cursor. Podemos enriquecer el ejercicio, y de paso practicar, creando sensores nuevos, para movimientos verticales y rotaciones de la cámara. También, podemos cambiar el sensor por otro periférico, como el ratón o incluso un joystick (palanca de control de un avión).

El actuador "**Motion**", es interesante para comprender los principios más básicos de la animación, pero es bastante limitado, y no se utiliza mucho. Es útil, para algunas cosas como, precisamente el movimiento de la cámara, y los movimientos continuos, como una hélice de avión o una noria. Estos últimos, se logran utilizando el sensor "**Always**".

En la práctica, se suele trabajar con curvas “**Ipo**” y con el llamado “**Action Editor**”. El primero, es un potente editor, que permite manipular cualquier tipo de movimiento, mediante curvas cartesianas. De hecho, puede hacer muchas otras cosas, como modificar la escala, el color, la luz, o incluso el propio material. El “**Action Editor**”, editor de acciones en nuestra lengua, es similar al editor **Ipo**, pero especializado en la animación de personajes.

Por fortuna, ambos tipos de curvas se pueden utilizar en el **BGE**, lo cual nos va a permitir crear, en nuestros juegos, animaciones de gran complejidad. Pero, como siempre, no nos queda más remedio que empezar por lo más sencillo. Así que, vamos a animar un cubo mediante la técnica estandar de **Blender**, y veremos, como esta animación queda reflejada en el editor **Ipo**. Para ello, creamos dos ventanas horizontales, la izquierda para la vista **3D,** y la derecha para el editor **Ipo**, como puede observarse en la “Figura 21”.

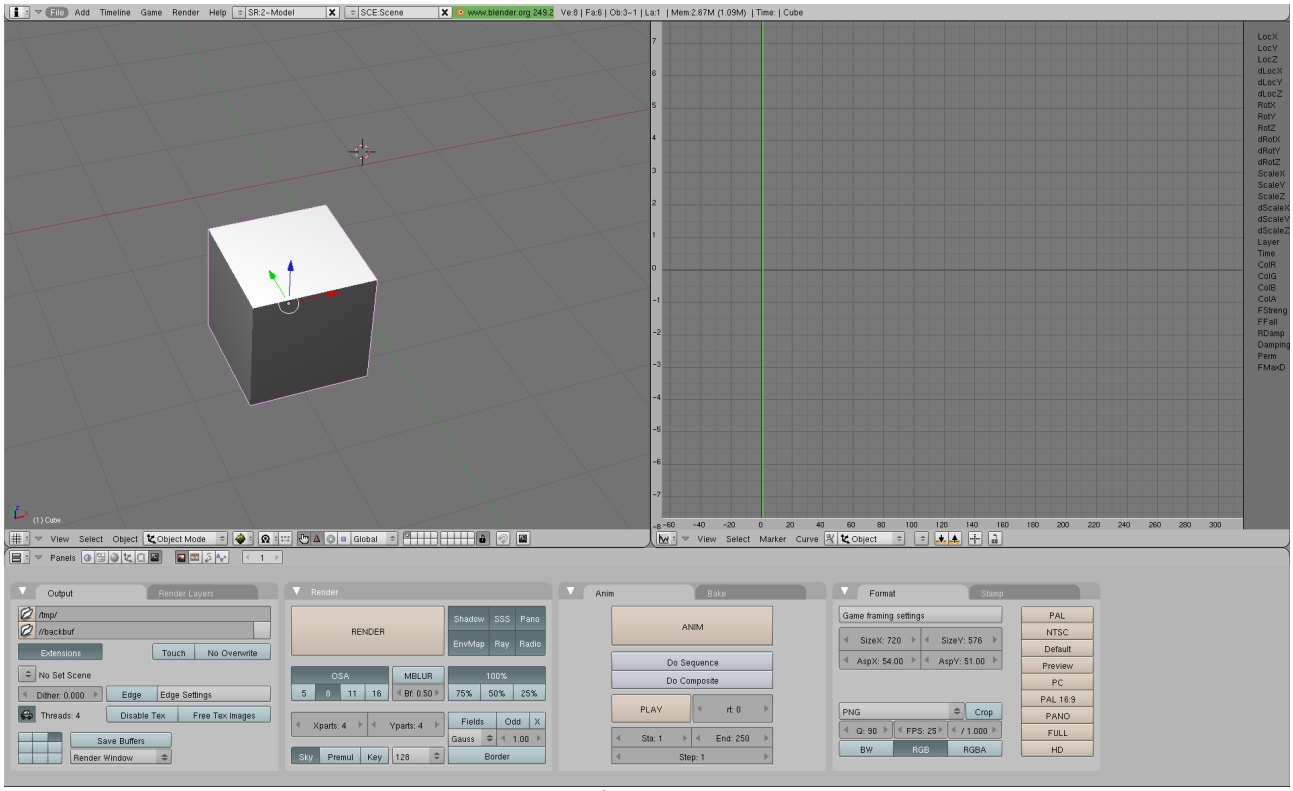

Figura 21

A continuación, asegurándonos de estar en el fotograma **1**, pulsamos la tecla “**i**” desde la vista **3D**. En el menú que emerge, seleccionamos “**LocRot**”. Esto genera un fotograma clave. Después, vamos al fotograma **30**. Desplazamos el cubo a la posición que deseemos, y pulsamos la tecla “**i**”. Seleccionamos de nuevo “**LocRot**”, y ya tenemos un nuevo fotograma clave, esta vez en el fotograma **30**. Para poder ver nuestra primera animación en tiempo real, pulsamos “**Alt a**”. El cubo, realizara el movimiento programado en la vista **3D**, una y otra vez. Para salir de este modo de visualización, pulsamos la tecla **Esc**.

Otra forma de ver la animación en la ventana **3D** es, utilizando la linea de tiempo del editor **Ipo**. Esta es, una linea vertical verde, que podemos desplazar manualmente a la posición en el tiempo que queramos. Tiempo y número de marcos o fotogramas, viene a ser lo mismo. Ya que una película utiliza normalmente 25 fotogramas por segundo (“Figura 21”).

La curva **Ipo** se crea automáticamente, al animar cualquier objeto de forma convencional, en **Blender.** Pero desde el editor, podemos hacer todas la manipulaciones de la misma que queramos. La curva esta constituida por vértices. Podemos seleccionarlos con el botón derecho del ratón. Si queremos

seleccionar todos los vértices de una curva, pulsamos “**a**”. Para borrar utilizamos la tecla “**Supr**”. Para hacer “**zoom**”, giramos la rueda del ratón y para desplazarnos simplemente la pulsamos (Se logra el mismo efecto, pulsando simultáneamente los dos botones del ratón). Nada de ello es demasiado misterioso, teniendo en cuenta, que se trata de los mismos atajos de teclado, que **Blender** emplea en muchas tareas.

El eje horizontal, representa el número de fotogramas, y el vertical el valor numérico del movimiento. La modificación de las curvas se realiza, creando vértices nuevos, mediante la tecla “**Ctrl**” en la posición que queramos. Ver “Figura 22”, a continuación:

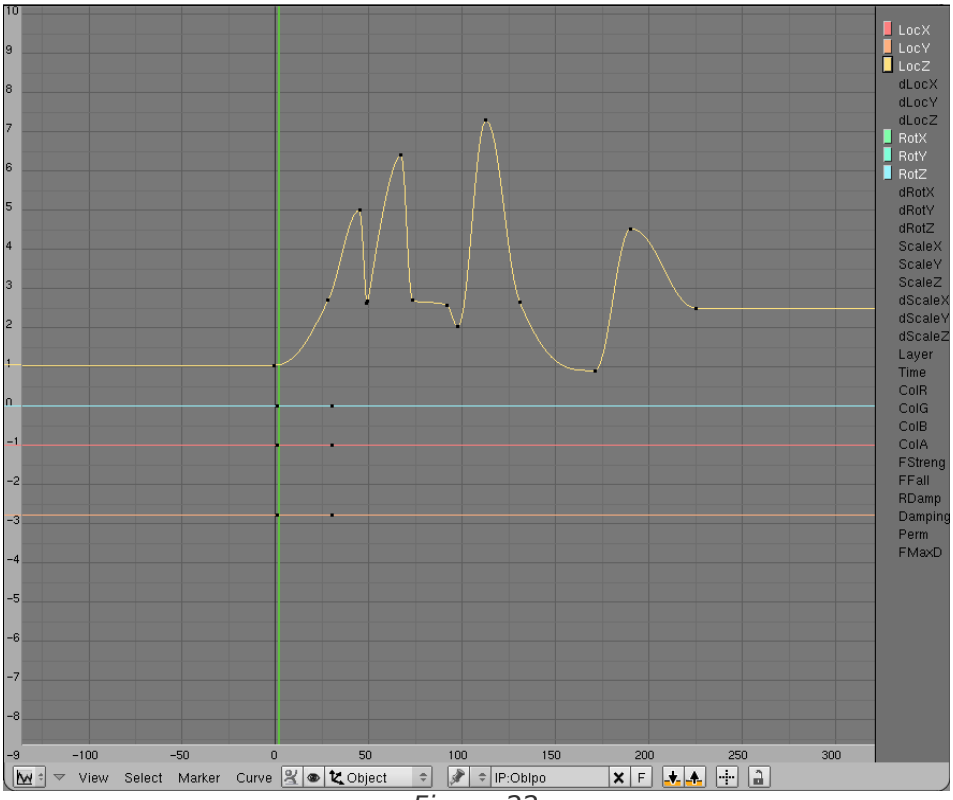

Figura 22

Si observamos el editor **Ipo**, con un poco más detenimiento, veremos cuatro curvas de colores diferentes. Tres de ellas son llanas, y la que se encuentra en la parte superior, presenta alteraciones. Esta última es el eje **Z**, que es el único eje animado. Como se puede interpretar de esta gráfica, el objeto al que esta vinculado ascenderá y descenderá, tantas veces como la forma de esta curva.

Es importante, fijarse en la parte derecha de la ventana del editor **Ipo**. Como se ve, existe una columna, que empieza por “**FmaxD**” abajo, y termina por “**LocX**”, en la parte superior. Son los parámetros, a los que se puede aplicar una curva **Ipo**. En el ejemplo de la “Figura 22”, tenemos activados solo algunos de ellos. Los que están resaltados en color blanco. También tienen un cuadrado de color idéntico, al de la curva que representan. Si nos fijamos, la curva amarilla es: “**LocZ**”, o sea, movimiento en el eje **Z**.

El editor **Ipo** es tan complejo, que perfectamente podría ser un programa independiente. Vamos a analizarlo un poco más. En la parte inferior de la ventana, disponemos de cuatro menús emergentes: **View**, **Select**, **Marker** y **Curve**. En estos, encontraremos todo aquellos elementos, que podemos necesitar para editar nuestras curvas, sin utilizar atajos de teclado. El siguiente menú, hacia la derecha (Figura 22), sirve para seleccionar el tipo de elemento al que se le va a aplicar la curva. En este caso “**Object**”. Pero existen otros tipos, como **Word** (mundo), **Shape** o **Sequence**. Objetos especiales, como un foco de luz o una cámara poseen sus propios modos.

Estos modos, son algo así como tipos especializados de curvas **Ipo** para un elemento de **Blender**. Existen incluso, los modos "**material**", y "**textura**". Por ello, el editor **Ipo**, se emplea habitualmente para modificar materiales, cambiar el color y la intensidad de los focos de luz, y realizar cualquier tipo de combinación posible, entre todos estos elementos. De ahí, la complejidad y potencia de este editor.

Si ahora pulsamos la tecla "**p**", para ejecutar el **BGE** veremos... Que no pasa nada. Es normal. Para que la animación funcione desde el juego, es necesario crear en el modo de edición de juegos (**F4**) un **sensor**, un **controlador** y un **actuador** . El actuador tiene que ser: "**Ipo**". Ver "Figura 23".

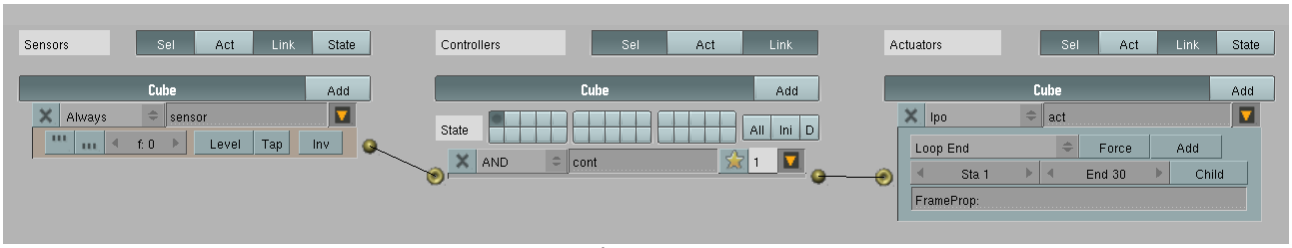

Figura 23

Como se puede ver en la imagen, hay que indicar el primer y último fotograma de la animación, en este caso **1** y **30**. También, debemos indicar la forma de ejecución, por defecto "**play**", o sea, que se ejecute una sola vez. Pero hay más posibilidades. En nuestro ejemplo, estamos utilizando "**Loop End**" que hace que se repita una y otra vez la misma acción.

12- CREANDO UN ESQUELETO

Ahora, que ya conocemos las técnicas básicas de animación, estamos preparados para afrontar retos más ambiciosos. Nuestro objetivo inicial ahora, es animar un personaje de forma humana, haciendo que camine, de una forma más o menos natural. Más adelante, veremos como modificar vértices, mediante curvas, para lograr efectos más sutiles.

Para animar personajes, se utilizan unas estructuras especiales llamadas "**esqueletos**", en inglés "**armature**". Estos, están constituidos por unidades más simples, llamadas huesos. Hasta aquí, nada es demasiado sorprendente. Casi parece, que nos refiriésemos a esqueletos naturales. A pesar de su parecido aparente, los esqueletos **3D**, tienen una función completamente diferente. Esta es, facilitar la animación de un personaje, transfiriendo el movimiento del propio esqueleto. Editamos la animación sobre este, y la malla del personaje le sigue. Es decir, el personaje hereda el movimiento del esqueleto. Existe, por tanto, una especie de parentesco. El esqueleto es el padre, y el personaje, el hijo.

Los buenos esqueletos, están formados por pocos huesos, pero situados de tal manera, que permitan animar un actor de forma sencilla, y con buenos resultados. Para construir un esqueleto, primero cargamos una escena que contenga uno de los personajes, que modelamos con anterioridad. Después, creamos un objeto especial, proporcionado por **Blender**, llamado precisamente "**armature**".

Barra espaciadora / Add / Armature

La visualización estandar del programa, representa al hueso como un prisma piramidal, con una esfera en cada extremo ("Figura 24"). Estas esferas, son las rótulas de las articulaciones. Así que, debemos situarlas en una rodilla, el codo, el hombro, etc. Si teniendo seleccionado nuestro hueso, vamos al modo de edición (**F9**), hallaremos un panel específico, para trabajar con huesos ("Figura 25"). El aspecto del hueso, es solo una forma de visualización ("**Octahedron**"), entre otras. "**Stick**", lo muestra como un simple segmento y se suele utilizar cuando ya tenemos terminado el esqueleto.

Para simplificar la visualización. "**Envelope",** nos permite ver el área de influencia del hueso, sobre la malla, y "**B-Bone**" nos lo muestra en modo de caja o cubo.

El proceso de generar el esqueleto completo, partiendo de un solo hueso, consiste en extrusionar huesos nuevos, partiendo del primero. Al final, todos los huesos acabaran dependiendo unos de otros. Esto es esencial, para que el movimiento del personaje tenga algo de fluidez. Esto es imprescindible, ya que los elementos que constituyen el esqueleto, deben influenciarse unos a otros, durante la animación.

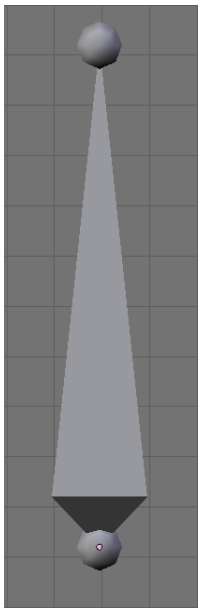

Figura 24

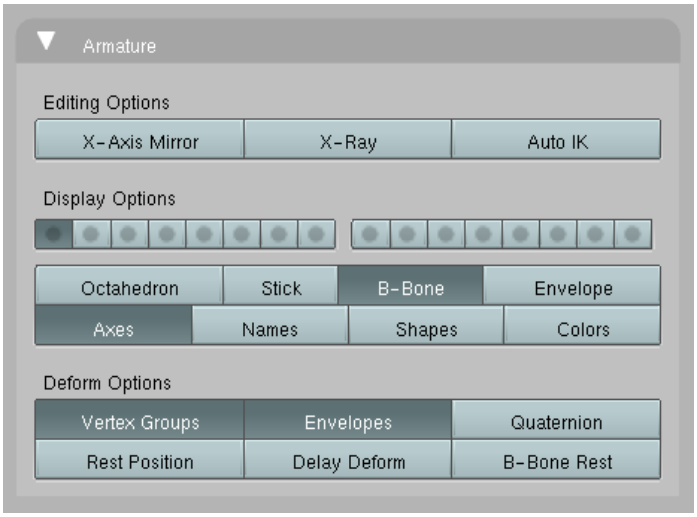

Figura 25

En la parte superior del panel "**Armature**", se encuentra un apartado muy interesante. Su denominación es "**Editing Options**". Contiene los subapartados "**X-Axis Mirro**r", "**X-Ray**" y "**Auto IK**". El primero, nos permite el modo de espejo al extrusionar el hueso. Es la misma técnica que empleamos al modelar nuestro primer personaje. Y la emplearemos también para el esqueleto. El segundo apartado, "**X-Ray**", se traduce aproximadamente como **rayos X**. Provoca, que el hueso se visualice siempre, aunque se encuentre (como suele suceder), en el interior de un personaje.

"**Auto IK**", activa un tipo especial de animación, llamado "**cinemática inversa**". Como su nombre indica, consiste en animar de forma invertida. Es decir, aplicando el movimiento sobre los huesos situados en los extremos. Como es lógico, el resto del esqueleto se vera obligado a seguir a estos. Si el esqueleto está bien realizado, se pueden lograr complejas animaciones muy rápidamente. Pero es una técnica bastante avanzada, y requiere disponer primero de un buen esqueleto.

Bien, vamos a realizar nuestro esqueleto. Para ello, ejecutamos **Blender**, y cargamos una escena que contenga un buen personaje. Yo dispongo de uno llamado **Kris** ("Figura 26"). Creamos el primer hueso, y lo situamos como puede verse en la imagen. Después seleccionamos la rótula inferior del hueso (la esfera), y en modo de edición (**F9**), activando "**X-Axis Mirro**r" y "**X-Ray**", llevamos a cabo la primera extrusión (**e**).

Si todo ha ido bien, deberíamos haber creado dos nuevos huesos, simultáneamente. Situarlos correctamente, no es tarea fácil. Afortunadamente, gracias al efecto espejo (**Mirror**), solo tenemos que preocuparnos de un hueso. Como se da la circunstancia, de que el personaje es también simétrico, el segundo hueso, se ajustará perfectamente al lado inverso de la figura.

Estos primeros tres huesos vienen a formar la cadera. Partiendo de los nuevos huesos, hacemos una segunda extrusión para las piernas. La parte final de estos huesos, tiene que quedar situada en la

articulación de la rodilla. Después, mediante una nueva extrusión, llegamos hasta el talón, para finalizar las extremidades inferiores, mediante la extrusión de los pies.

Durante todo el proceso, es muy conveniente tener tres ventanas **3D** abiertas: la vista superior, la frontal y la lateral. Hay que asegurase, de que los huesos se sitúan aproximadamente, en los ejes de las extremidades. Y que las rótulas, lo hacen exactamente en las articulaciones.

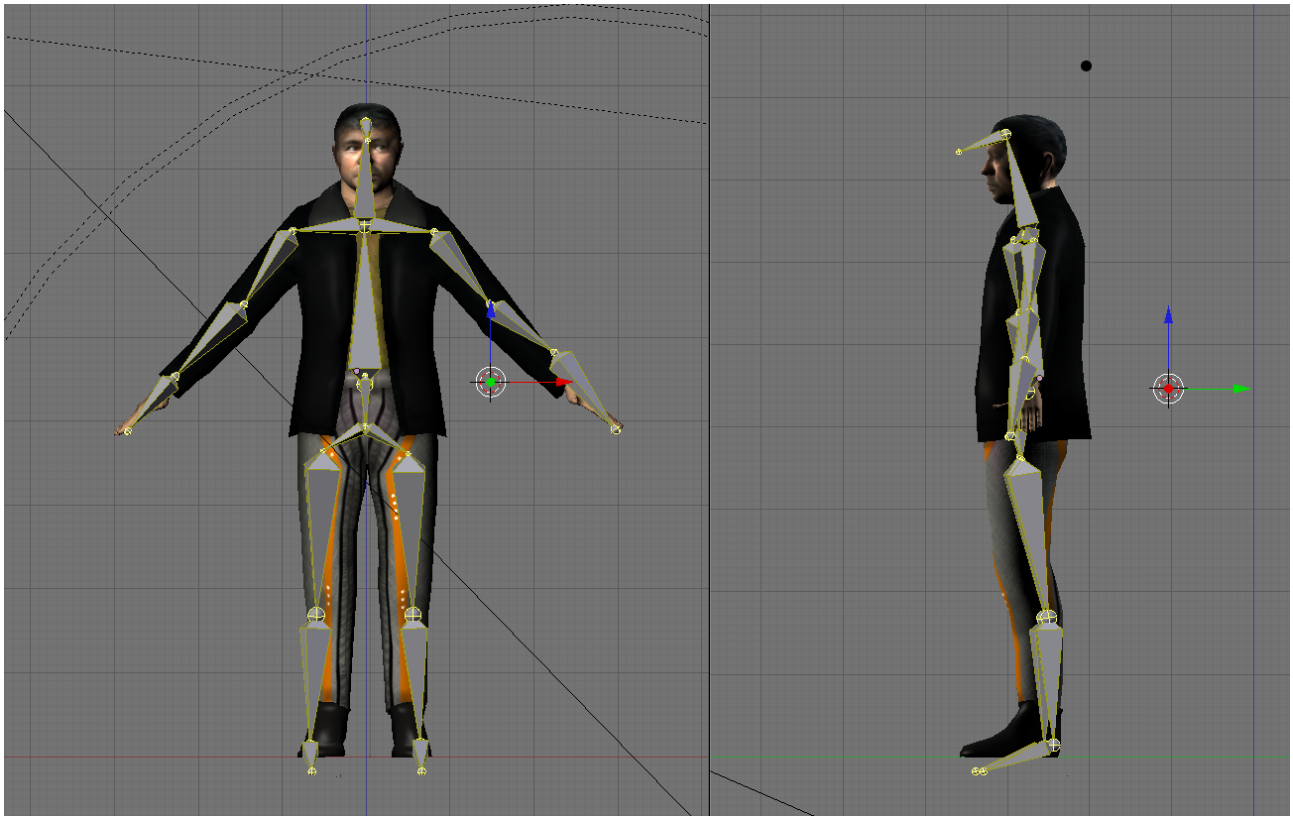

Figura 26

Hecho esto, volvemos al primer hueso. Desactivamos el botón "**X-Axis Mirro**r", y lo extrusionamos hacia arriba. Como puede verse en la "Figura 26", solo he utilizado un hueso para llegar, más o menos, hasta la parte superior del esternón. Si precisamos una espalda articulable, deberíamos dotar a esta región de más huesos. Pero mi esqueleto es un ejemplo simple, pensado deliberadamente para un personaje en baja. Además, es más útil de este modo, desde el punto de vista didáctico.

En la zona de la escápula, es donde se va a resolver finalmente este ejercicio. Como se ve en la imagen, extrusionaremos tres extremidades (los dos brazos y la cabeza), partiendo de este eje. Empezaremos por extrusionar los huesos de los brazos. Para ello, volvemos a activar la opción "**X-Axis Mirro**r", repitiendo exactamente el mismo proceso que empleamos para las piernas.

La cabeza no supone mayor problema. Desactivamos "**X-Axis Mirro**r", y extrusionamos una vez más, el hueso situado en la articulación superior. Dos tramos son suficientes, uno para el cuello, y el otro lanzado hacia el frente para la cabeza propiamente dicha.

El lector suspicaz, habrá observado que el esqueleto de la "Figura 26", no cumple escrupulosamente las indicaciones dadas en este manual. Todo tiene su explicación. Las irregularidades son dos. Por un lado, los huesos del pie y la rodilla están claramente desplazados hacia los extremos. Por otra parte, el hueso de la cabeza se encuentra, prácticamente fuera del personaje. Ambas licencias, se ha tomado por razones prácticas. Un esqueleto perfecto puede ser difícil de ver, sobre todo si se utiliza una visualización en modo "**Stick**" (linea). Y más difícil todavía, de seleccionar. Mediante este truco, resulta muy sencillo de ver y seleccionar.

13- MOVIENDO EL ESQUELETO.

O lo que es lo mismo, animando al personaje. En este capítulo, vamos a tratar del movimiento del esqueleto que acabamos de realizar en el apartado anterior, y de como trasferir este movimiento al personaje en el **Blender Game Engine** (**BGE**). Para empezar, como nos gustan los retos, vamos a intentar que nuestro actor (**Kris**) camine de una forma más o menos normal. Esto no es tan simple, como puede parecer a priori.

Como se comentó anteriormente, el movimiento del personaje viene heredado de su esqueleto. Pero como este influencia a la malla del personaje, es de lo que vamos a tratar ahora. El esqueleto, posee un campo de influencia llamado "**envelope**" (envolvente), dentro del cual, cualquier vértice del personaje, sera "capturado" por el primero, quedando obligado a seguirle en su movimiento. Este área de influencia, es mayor que el tamaño aparente de los huesos. En la "Figura 27", podemos ver la envolvente del esqueleto de **Kris**.

Si observamos la imagen con un poco de atención, veremos que la envolvente distingue, mediante colores, las rótulas (amarillo), de los tramos rectos de los huesos (magenta). También, veremos que no todos los huesos tienen una envolvente del mismo tamaño. Asimismo, existe un área, resaltada en blanco, alrededor de la envolvente de cada hueso.

La envolvente de cada hueso, debe adaptarse al cuerpo. Como el cuerpo no tiene una forma constante, las envolventes deben cubrir áreas mayores en unas partes del cuerpo que en otras.

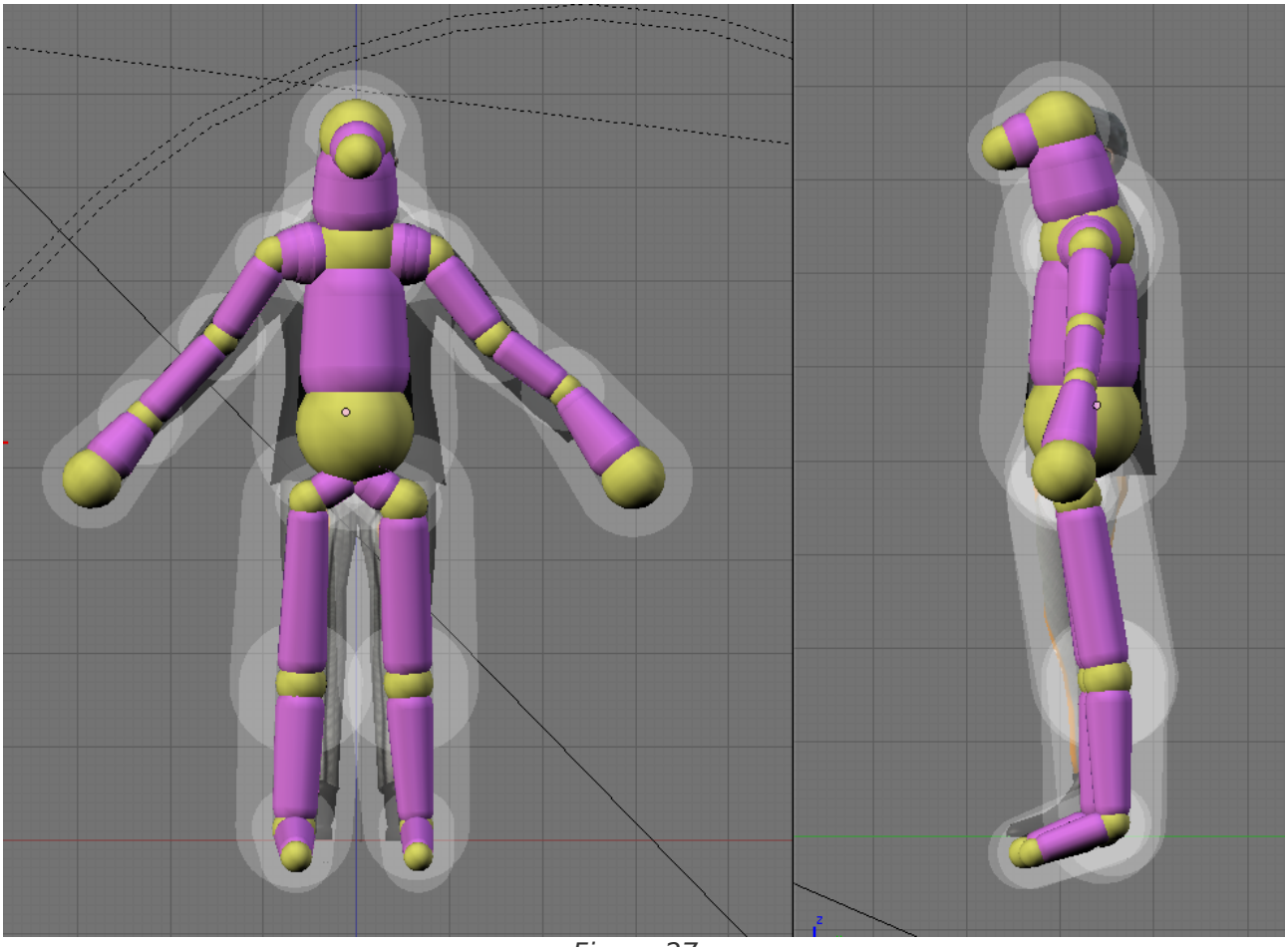

Figura 27

El área blanca, alrededor del esqueleto, es un campo de influencia decreciente de los huesos. La envolvente completa, es el resultado de la suma de ambas áreas de influencia. Para obtener buenos resultados, la envolvente completa debe cubrir totalmente al personaje. De no ser así, durante la animación, parte de los vértices del mismo se desplazarían junto al esqueleto, permaneciendo el resto inmóviles. Y esto, no es nada elegante.

Desde el punto de vista teórico, todo esto, está muy bien. Pero, ¿Como se hace en **Blender**? Veámoslo, paso a paso. Primero, seleccionamos nuestro esqueleto. Y en modo de edición (**F9**) activamos los botones "**X-Ray**", en la parte superior del panel "**Armature**", y "**envelope**" en la zona media del mismo panel. Ver "Figura 25". De forma inmediata, veremos la envolvente de nuestro esqueleto.

Para modificar el tamaño de la envolvente de un hueso, en modo de edición (**F9**), simplemente lo escalamos "**s**". Para hacer lo mismo con el área de influencia decreciente (color blanco) "**Alt s**". Ver la "Figura 28", a continuación.

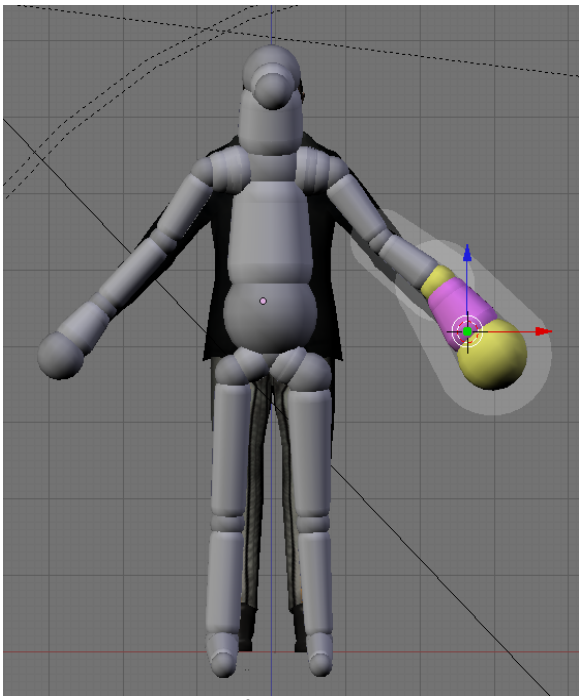

Figura 28

Para concluir este capítulo, ya solo queda explicar como se realiza, en la práctica, el vínculo jerárquico entre el esqueleto y el personaje. Es sencillo, basta con seleccionar, en modo de objetos, primero al personaje, y luego al esqueleto. Pulsamos "**Ctrl p**", y en el menú que aparece, seleccionamos "**Armature**". Esta acción, convertirá al esqueleto, en el objeto padre. Finalmente, seleccionamos en un nuevo menú emergente "**create from Envelopes**". O sea, crear el vínculo a partir de la envolvente del esqueleto.

14- ANIMACION DE PERSONAJES.

En el capítulo anterior, prometimos "mover el esqueleto", pero al final no lo hicimos. Pero parece que ha llegado ya el momento para ello.

Blender, incorpora un "**modo**" especializado en animación de personajes, llamado "**poser**". Se selecciona este modo, al igual que el resto de modos, desde el menú inferior de la ventana **3D**. Ver "Figura 29". La traducción al castellano de "**poser"** es maniquí. En realidad, un esqueleto **3D**, y un maniquí, son objetos muy similares.

En este modo, la **armature** o esqueleto se visualiza en color azul. Las selecciones, quedan resaltadas en azul claro. Como se comentó anteriormente, la animación consiste en hacer que el personaje conocido como **Kris**, camine. Para ello, vamos a seguir la siguiente estrategia. Vamos a animar, solamente la mitad del ciclo. Cuando caminamos. adelantamos primero un pie. Este primer paso, supone que el peso del cuerpo se desplaza hacia el otro lado, para poder mantener el equilibrio. Como consecuencia de ello, la espalda se curva también, hacia el lado en el que nos apoyamos. Después, cuando el pie ya esta bien asentado en el suelo, se invierte el proceso. Es decir, adelantamos el otro pie, desplazando de nuevo el peso del cuerpo hacia el lado opuesto.

Esto, que puede parecer muy complejo de animar, no lo es tanto. Pero es conveniente tener cierto sentido práctico, y simplificar las cosas al máximo. Lo que vamos a hacer es modificar la posición de nuestro esqueleto, de tal forma que las piernas formen una Lambda Griega (una V invertida), en la vista lateral. Se presupone, que el esqueleto esta emparentado con el personaje, y que gracias a un buen trabajo con su envolvente, su forma se anima correctamente.

Bien, la actitud de nuestro actor, es la de encontrarse en medio de un paso. Y a cubierto la mitad del ciclo, ya que todavía le falta el movimiento del otro pie. Ahora, escogemos uno de los pies, y fijándonos sobre todo, en la vista de perfil, rotamos la cadera, la rodilla y el propio pie, hasta lograr la postura del paso. En esta "**pose**", el pie se eleva un poco del suelo, a la vez que se adelanta. Para hacer posibles estas transformaciones, hacemos uso de los atajos de teclado conocidos.

Cuando estemos satisfechos con el resultado, seleccionamos todo el esqueleto (**a**), y situándonos en el fotograma **1**, pulsamos "**i**". En el menú emergente que aparece, seleccionamos "**LocRotScale**". Esto, crea un fotograma clave, con esta postura. A continuación, en el menú "**pose**" en la parte inferior de la vista **3D**, seleccionamos "**Copy Current pose**". Esto, copia la postura actual, ya que precisamos que al completar el ciclo completo, el personaje termine exactamente en la misma posición que empezó. De lo contrario, no obtendríamos un ciclo que pueda ser repetido hasta el infinito.

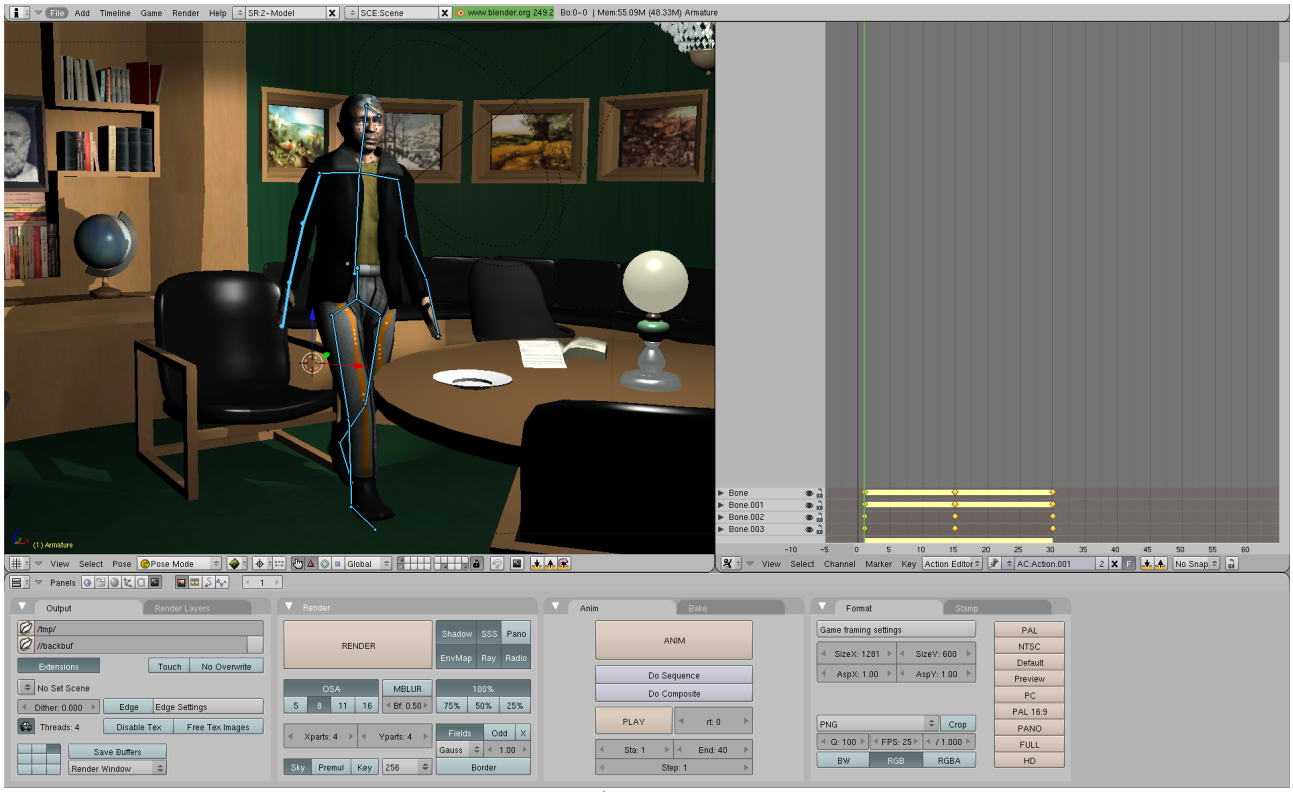

Figura 29

Ahora, nos situamos por ejemplo, en el fotograma **20**. Modificamos la postura, pero esta vez exactamente al revés. De tal manera, que el pie que de el paso sea el contrario. Hecho esto, seleccionamos el esqueleto completo, y mediante la tecla "**i**", creamos un nuevo fotograma clave (**20**). Después, nos desplazamos al fotograma **40**, y desde el menú "**pose**" seleccionamos "**Paste pose**". Finalmente, convertimos el fotograma **40**, también en clave. (este tipo de fotogramas se suelen denominar "**keyframes"**). De este modo, pegamos la pose inicial en el último fotograma (**40**).

Para poder ver la animación de una forma cíclica, debemos indicarle a **Blender**, que visualice la animación desde el marco **1** al **40,** de forma continua. Esto, lo hacemos desde el apartado **Scene**, (**F10**), en el panel "**Anim**". Ver "Figura 30".

Figura 30

Las animaciones, se muestran en **Blender** en la vista **3D**, mediante "**Alt** a". Si hemos seguido este tutorial, nuestro personaje caminará. O mejor dicho, sus piernas lo harán, pues por razones didácticas, no hemos animado, ni los brazos, ni la espalda, ni la cabeza. Si este primer ejercicio te ha resultado sencillo, es que estas preparado para continuar. Si no es así, intentalo de nuevo.

Este primer ejercicio, se convierte en uno más avanzado, animando el resto del cuerpo. El objetivo principal es animar la espalda, que es lo que presenta mayor dificultad. No es fácil, ni posiblemente práctico, describir con palabras estos movimientos. La observación de la realidad, el trabajo con imágenes y vídeos da mejores resultados. En esta fase del trabajo, la experimentación y la práctica son fundamentales. La falta de movimiento, genera animaciones muy mecánicas, pero el exceso no resulta nada natural. Así que, hay que dar con el punto exacto. Como con otras actividades artísticas, algunas personas, tienen facilidad para estas cosas y otras no.

El tema de la animación es tan complejo, que para las producciones cinematográficas en **3D**, se hace uso de una técnica llamada "**Motion Capture**", o sea, captura de movimiento. Consiste, en colocar sensores en las articulaciones de un actor real, y recoger los datos de su movimiento. Estos, se le suministran, más tarde, al personaje **3D**. Como se puede imaginar, los resultados presentan una complejidad y naturalidad mucho mayor.

Cuando hayamos logrado, que nuestro personaje camine, tendremos que hacer que su movimiento funcione también en el **Blender Game Engine** (**BGE**). Pulsamos "**p**", para ejecutar el motor del juego. Pero, nuestra figura no se anima. Es imprescindible, vincular el objeto "**armature**", con su animación, desde el editor de juegos de **Blender**. Seleccionamos el esqueleto, y mediante **F4**, vamos al editor de juegos. Creamos un sensor, un controlador, y un actuador para el esqueleto. Los unimos mediante hilos. Hacemos una ventana nueva, y ejecutamos en ella el "**Action Editor**". Como consecuencia de haber animado a nuestro personaje previamente, dispondremos de forma automática, de una acción para el mismo. Copiamos su nombre, seleccionándolo de la parte inferior de la ventana. Escogemos el actuador "**Action**", y pegamos el nombre de la acción o movimiento que acabamos de copiar.

Si ahora ejecutamos el modo de juego de **Blender** (**p**) la animación del personaje se reproducirá.

Pero todavía hay algo más, muy interesante, que podríamos hacer. Lograr que el personaje se desplace a la vez que camina, ya que en su configuración actual, camina sin moverse del mismo lugar. Para ello, creamos un nuevo actuador de tipo "**Motion**", y le damos un valor de **0.03** en el eje **Y**. También, nos podríamos divertir, creando muchas copias de nuestro personaje preferido y observar como desfila para nosotros, nuestro ejército clon. Ver Figura 31.

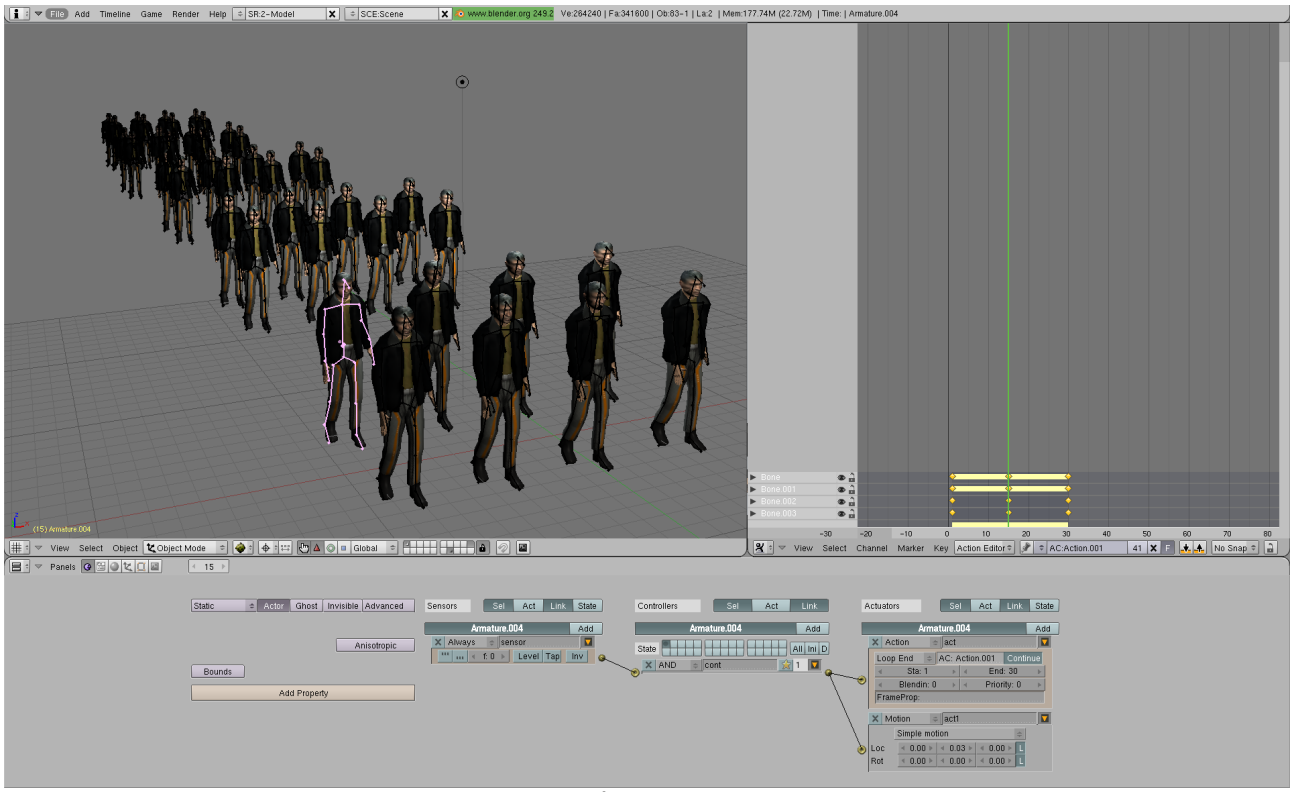

Figura 31

15- CLAVES DE VÉRTICES.

Si has llegado hasta aquí, deberías estar entusiasmado. Ya puedes crear tus propios mundos en **3D**. Poblarlos con tus personajes, y vivir la experiencia de sumergirte en ellos, en tiempo real. Pero quizás seas muy observador, y hayas advertido, que a los personajes les falta algo. Se mueven, es cierto, pero con movimientos bastante simples, y solamente de las partes más básicas del cuerpo. Necesitamos más expresión. Esta se logra, animando las manos y el rostro de nuestros personajes. La expresión, los vuelve vivos e interesantes.

Como puede observarse, las cosas se complican cada vez más, en este taller. Es el precio que se paga, por seguir avanzando. A continuación, vamos a animar unas manos, y más tarde veremos una técnica, que se puede utilizar para un rostro humano.

La animación de una mano, no implica ningún cambio importante, en relación con las técnicas empleadas en la animación de personajes, hasta ahora. Si acaso, supone un cambio de mentalidad acerca de la escala. Antes, trabajábamos sobre una figura completa, y ahora lo hacemos solamente, sobre una fracción de la misma. Las dos partes principales de la mano son, la palma, y los dedos. Los dedos, pueden ser entendidos como extremidades de la palma, y esta, como el cuerpo de la propia mano.

Lo que tenemos que hacer, es una especie de esqueleto para la mano, partiendo del esqueleto general. El hueso, que antes llegaba hasta el extremo final de la mano, no desaparece. Simplemente

extrusionamos nuevos huesos, a partir de él. Podemos observar la configuración de estos, en la “Figura 32”, a continuación.

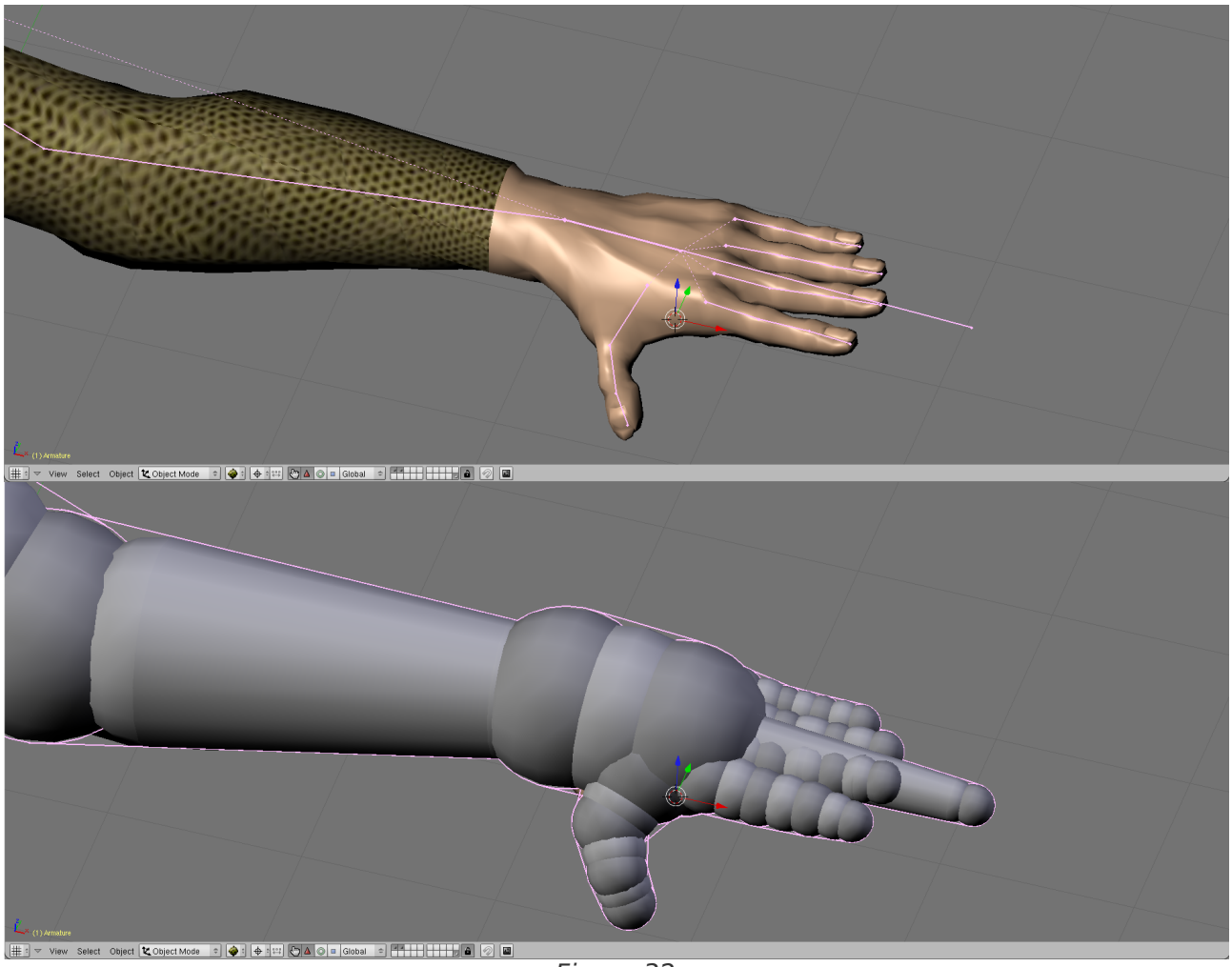

Figura 32

La verdad, es que es un trabajo bastante delicado. Cada dedo, tiene dos articulaciones, sin contar los nudillos. El dedo pulgar es el más complejo, ya que tiene un movimiento lateral, que permite hacer una pinza, con cualquiera de los otros cuatro. Probablemente, lo más laborioso sean las envolventes. El cuerpo general de la mano utiliza grandes envolventes, para poder cubrir totalmente la palma. Pero los dedos, requieren de envolventes que los recubran correctamente, pero sin llegar a solaparse entre si.

Como acabamos de ver, la animación de las manos, no requiere ninguna técnica especial. Ahora, vamos a tratar el tema de la animación facial, es decir la animación del rostro humano. El rostro es, sin duda, la parte más expresiva del cuerpo humano. Las técnicas de animación, que hemos visto hasta ahora no resultan demasiado útiles para la animación facial, ya que los elementos que debemos modificar no son extremidades, que puedan animarse independientemente. Por otro parte, los movimientos de los elementos implicados: labios, ojos, orejas, e incluso de la propia cara, son extremadamente complejos.

Un enfoque práctico, frente a este nuevo problema, puede consistir en realizar modificaciones, sobre los vértices de la propia malla. Es decir, en el fotograma **X**, la malla presenta una forma dada, y en el fotograma **Y**, la modificamos. Y que el programa, calcule los fotogramas intermedios. Esta técnica, que fue conocida como “**Morphing**”, actualmente se la suele conocer como “**Blending**”. Lamentablemente, el motor de juegos de **Blender**, no es capaz de reproducir estas manipulaciones de las mallas. Por lo menos, en las versiones estables del programa actualmente.

He podido comprobar, que la versión "**Blender-2.5 Alpha2**", si que lo hace. Pero esta es una versión experimental, que carece por el momento, de un **BGE** independiente. No obstante, vamos a abordar el uso del **morphing** sobre grupos de vértices, por dos motivos. Primero, porque es una técnica eficaz, muy utilizada en las animaciones convencionales, con salida de imagen o vídeo. Y segundo, porque hay que suponer, que el **BGE** lo soportará completamente en futuras versiones.

Lo más sencillo, para comprender esta técnica, es realizar un primer ejercicio con un cuerpo simple. Así que, hacemos una nueva sesión de trabajo en **Blender**, y seleccionamos el cubo que este proporciona por defecto. Vamos al modo de edición de vértices, mediante "**Tab**". Y al modo de edición pulsando "**F9**". Subdividimos varias veces para obtener una malla, sirviéndonos del botón "**Subdivide**" del panel "**Mesh Tools**". Hecho esto, seleccionamos un grupo de vértices. Entonces, los desplazamos un poco hacia el interior del cubo, creando de este modo un hueco en él.

Esta forma modificada del cubo, será una de nuestras posiciones para el **Morphing** o **Blending**. Para poder continuar el ejercicio, necesitamos abrir una nueva ventana y ejecutar en ella el editor de curvas **Ipo**. En el menú inferior de esta ventana, escogemos como tipo de **Ipo**, "**Shape**", que significa curva. Y buscamos entre los paneles de edición, precisamente el panel "**Shapes**". Ver la siguiente captura de pantalla:

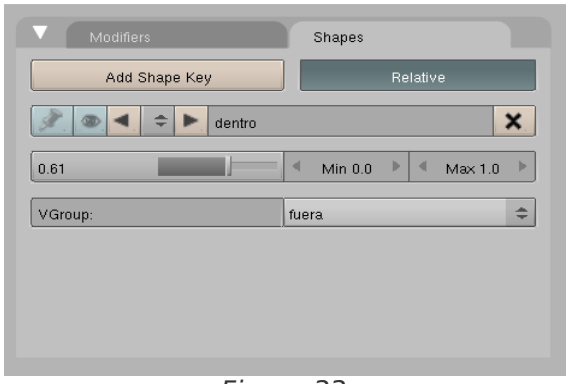

Figura 33

Mediante el botón "**Add Shape key**", hacemos una **clave de vértices**. Es lo mismo, que se hace con un fotograma clave, pero utilizando un grupo de vértices. Bien, a continuación nombramos esta clave como "**dentro"**, por ejemplo. Tomamos de nuevo nuestro cubo, modificamos su forma desplazando los vértices anteriores hacia afuera del cubo. Y creamos una nueva clave de vértices, al que denominaremos "**fuera**". Para desplazarnos entre una y otra clave, utilizamos las flechas. El botón "**Relative**", tiene que estar activado. Es el que apunta hacia otra clave. El nombre de la otra clave, lo escribimos en la casilla, abajo a la derecha. De este modo, relacionamos ambas claves.

Ahora solo faltan las curvas en sí. Para ello, tenemos dispuesto al editor **Ipo**. Veremos, que en la parte superior derecha, aparecen nuestras claves con los nombres que les hemos dado, (dentro y fuera). Están esperándonos. Seleccionamos la clave "**dentro**", y nos desplazamos al fotograma **0**, con la linea de tiempo. Esta, es una linea vertical de color verde, que podemos mover con el botón izquierdo del ratón. La linea horizontal, representa el movimiento de nuestro grupo de vértices. Realmente, movimiento quiere decir, posición relativa a una de las dos formas del cubo, en función del valor vertical. Este valor, que oscila entre **0.00** y **1.00** se selecciona en le botón "**Key Value**" del panel "**Shapes**". Es el que marca **0.61** en la "Figura 33".

La técnica, consiste en situar la linea de tiempo en la posición deseada, y con el ratón, dar al botón "**Key Value**", el valor que queramos. Cada una de estas operaciones, dará lugar a un punto de nuevo que, lógicamente modificará la curva **Ipo**, de la clave de vértices que estemos editando. Objetivos más ambiciosos, como la animación de unos labios, se logran de este modo. Editando varias claves, para las posiciones de los labios, y fundiéndolas entre sí de una forma sincronizada, hasta lograr el efecto de movimiento deseado.

16- ANIMACIÓN MECÁNICA.

En los videojuegos, puede haber otros objetos (aparte de los personajes y el escenario), como coches, aviones o tanques. Animar un vehículo de este tipo, sin tener en cuenta la física del **BGE**, puede ser muy sencillo. Pero también muy limitado y poco realista. En este capítulo, vamos a animar un vehículo, concretamente un tanque **T-34**, de la segunda guerra mundial, modificado para moverse sobre ruedas, en lugar de cadenas. La animación de un tanque, con todos los mecanismos de la oruga en tiempo real, es un ejercicio muy avanzado, totalmente inadecuado, desde un punto de vista didáctico.

Para este ejercicio, primero necesitamos un tanque. No vamos a modelar uno ahora. Yo he utilizado un **T-34**, pero puedes utilizar cualquier vehículo de cuatro ruedas. Los **scripts** de **Python**, que vamos a utilizar se crearon, originalmente para un coche. Cuando terminemos, nuestro tanque sera capáz de desplazarse, adaptándose a las irregularidades del terreno. Presentará además, propiedades físicas, complejas, como la suspensión del chasis sobre las ruedas, sensación de peso, inercia, etc.

Podemos utilizar, cualquier cosa como chasis. El secreto, está en las propiedades de las ruedas. Y casi todo depende de tres **scripts** de **Python**. Aunque el **Blender Game Engine**, presume de permitir la edición, sin necesidad de saber programación, lo cierto es que, algunas cosas muy complejas precisan todavía de código. Pero veremos también, que ello no es un problema tan grande. Yo no he escrito los guiones de **Python** que vamos a utilizar en esta práctica. No obstante, los utilizo. Lo mejor, es perder el miedo a la programación cuanto antes. Con un mínimo de ingenio, cualquier persona es capaz de comprender, y aprovecharse de las grandes posibilidades, que unos pocos fragmentos de código le aportan.

Vamos a empezar, buscando un vehículo adecuado y editando el terreno. Para ello, creamos una escena nueva en **Blender**, eliminamos el cubo que aparece por defecto, y cargamos un plano nuevo. Escalamos el plano, para que pueda ser la superficie de nuestro mundo, y en modo de edición (**F9**) utilizamos el botón "**Subdivide**" del panel "**Mesh tools**".

Figura 33

Entonces nuestro plano, originalmente formado por una sola cara, se dividirá en las dos dimensiones, manteniendo la misma forma, pero esta vez, conteniendo cuatro planos. Cuatro planos, son muy pocos para hacer deformaciones, así que subdividimos varias veces, hasta obtener una malla más fina. Subdividiremos tantas veces como sea necesario, en función del tipo de paisaje que queramos modelar. Cuanta más altura, y detalles precisemos, más tendremos que subdividir. Es conveniente no excederse, al fin y al cabo estamos editando videojuegos, y cada vértice es un tesoro. Si el terreno estuviese destinado a un fin diferente del tiempo real, no tendríamos que preocuparnos tanto de estas cosas.

Para dar forma a nuestro plano, hay dos métodos. Uno, es utilizar un modo especial para escultura llamado "**sculpt mode**". El segundo método, consiste en obtener las alturas (eje **Z**) partiendo de una imagen apropiada. En este ejercicio, vamos a hacerlo de esta segunda forma. Para ello, mapeamos el plano en **UV,** con una imagen de una vista aérea de un terreno. Este tipo de imágenes, se pueden conseguir en internet. También, se pueden generar en programas de edición de gráficos, como **The Gimp**, aunque ello exige cierto dominio de las técnicas de retoque fotográfico.

Una vez tenemos mapeado nuestro plano, en modo de edición "**Tab"**, abrimos una ventana con una vista de perfil "**3"**. Seleccionamos todo "**a"**, editamos mediante "**F9**", y en el panel "**Mesh tools"** escogemos la opción "**Noise**". La malla, se modificará en el eje **Z**, en función de la imagen que hemos utilizado como material. Normalmente, es necesario repetir esta operación dos o tres veces para lograr el resultado deseado. Ver "Figura 34".

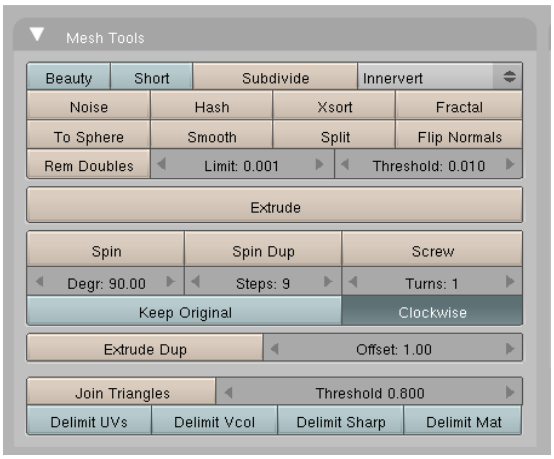

Figura 34

Como se puede observar, hay muchas opciones que es posible utilizar en este panel. Pero unas son más importantes que otras. "**Smooth**" es muy recomendable, ya que suaviza un poco el resultado. Se aplica, las veces que sea necesario. Si el nivel de suavizado no nos parece suficiente, siempre podemos utilizar "**Set Smooth",** en modo de edición "**F9**", que nos permitirá un suavizado completo.

El botón "**Fractal**", genera relieve mediante patrones matemáticos de tipo fractal. Se puede utilizar sobre una malla cualquiera, aunque no tenga ningún material. "**To Sphere**" transforma la malla en una forma lo más esférica posible, en función del valor que escojamos entre **1** y **100**. Puede llegar a ser útil, en algunos casos.

Una tercera forma de modelar un paisaje, es editando los vértices manualmente. O sea, del mismo modo, que modelaríamos un edificio o un personaje. Pero suele ser el método más lento y laborioso.

En cuanto hayamos terminado con el suelo, y suponiendo que disponemos de algún tipo de vehículo, pasamos a los contenidos, propiamente dichos de este capítulo. Si no es así, podríamos utilizar un simple cubo. No supone ninguna diferencia, desde el punto de vista didáctico. Lo importante, es que todo el chasis del vehículo sea un solo objeto, y que las ruedas sean objetos independientes.

En realidad, casi todo el trabajo más complejo recae sobre tres **scripts** de **Python**. Yo no he escrito estos programas, solo me sirvo de ellos. Aprovecho la oportunidad, para agradecer a las personas que si los escribieron, que estos **scripts** sean públicos. Y también estamos en deuda, con la persona que se preocupó, de realizar la traducción de los comentarios de estos programas al castellano.

Para no tener que ocupar innecesariamente, varias páginas con estos **scripts**, puedes descargarlos desde este enlace:

http://es.gnu.org/~littledog/3d/4scripts.tar.gz

Ahora, vamos a ver como se hace uso de estos programas, desde **Blender**. Partimos del hecho, de que continuamos trabajando, sobre nuestra escena del paisaje y el tanque. Hacemos una ventana nueva, o reutilizamos una de las existentes, y ejecutamos en ella el editor de textos de **Blender**. Lo hacemos igual, que con cualquier otro subprograma, como el editor de curvas **IPO**, o el editor **UV**. Ver "Figura 35".

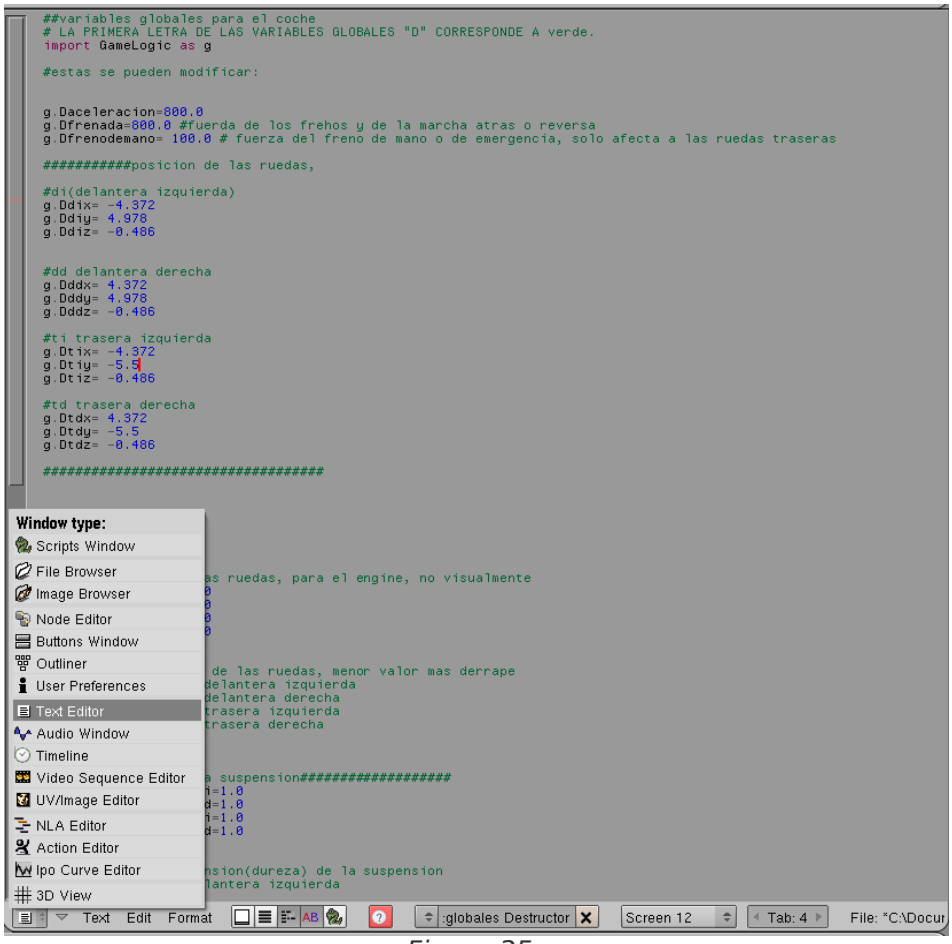

Figura 35

Para abrir un archivo de texto, en la barra de la parte inferior de la ventana hacemos: "**Text** / **open**", o también "**Text** / **new**", pegando en el nuevo archivo, el contenido del texto que queramos. Podemos tener varios ficheros de texto en el editor, y alternar entre ellos, seleccionándolos desde la barra inferior. Como puede observarse, en la captura de pantalla (Figura 35), se utiliza para ello un menú emergente, que esta situado justo a la derecha del pequeño ícono rojo. Actualmente, esta cargado un guión de **Python** (o **script**, es lo mismo) llamado "**globales Destructor**". Lo que tenemos que hacer a continuación, es abrir los guiones de **Python** (cuatro en total), para poderlos utilizar más adelante.

Y ya podemos empezar a trabajar con nuestro vehículo. Lo primero que hacemos, es seleccionar el chasis del mismo, es decir, el tanque sin las ruedas. Si, desde el apartado de edición (**F9**), en el panel "**Mesh**", pulsamos "**Center**", el centro geométrico del objeto se convertirá en su centro de masa. Pero no es interesante, en este caso. Los **scripts**, solo funcionan bien, con un centro de masas, que coincida con la linea de base del chasis. Para cambiar el centro de masa, desplazamos el puntero del ratón, aproximadamente a la parte inferior del chasis del tanque. Lo situamos en el centro del mismo, tanto en una vista de perfil, como en la vista superior. Finalmente, hacemos uso de la opción "**Center Cursor**" del panel "**Mesh**".

En las ruedas, hay que situar su centro de masa, en su centro geométrico exacto. Por ello, lo más sencillo es valerse del botón "**Center**". Hecho esto en las cuatro ruedas, las desplazamos a la posición que consideremos que les corresponde. Es decir, bajo el chasis del tanque, pero sin llegar a tocarlo. El **BGE**, simula un comportamiento físico real. Si una pieza móvil entra en contacto, por error, con otra, se pueden producir resultados inesperados. Lo mas típico, es que el objeto entero salte por los aires, o que alguna pieza salga despedida. Por ello, hay que tener un poco de cuidado.

La situación espacial de las ruedas, y su comportamiento, están determinadas por los **scripts** de **Python**. No por **Blender**. Por ello, que las ruedas se visualicen en la ventana **3D**, a una cierta distancia del chasis, no supone ningún problema. Lo importante, es que estén en su posición correcta en el **BGE** (**Blender Game Engine**). Es muy conveniente, que el vehículo este, claramente por encima del plano del suelo. En caso contrario, al iniciar la simulación, el tanque saldría despedido por el aire.

Para asegurarse, de que el resultado es correcto, ejecutamos el modo de videojuegos, cada vez que realizamos algún cambio. Es, también una buena medida de seguridad, realizar frecuentes copias de seguridad. Muchas veces, es más práctico retomar el trabajo, desde una versión de comportamiento impecable, que afrontar problemas de programación, en una versión problemática más avanzada.

Para ajustar la posición de las ruedas en el **script**, primero tenemos que conocer su posición en **Blender**. Para ello, seleccionamos una de las ruedas, y pulsamos "**n**". Aparece un panel flotante, con datos numéricos, acerca de la posición del objeto. Este panel, es una herramienta muy útil, pues puede utilizarse para modificar la posición, rotación y escala de cualquier objeto en **Blender**.

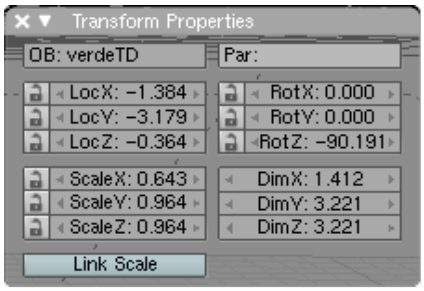

Figura 36

Los valores, que pueden observarse en la parte superior izquierda de la "Figura 36", son la posición de la rueda trasera derecha (**TD**), en **Blender**. Los tres valores, empiezan por "**Loc**", y representan posiciones en los tres ejes cartesianos (**X,Y,Z**). Estos valores, tenemos que trasladarlos al **script** "**Globales d,estructor**". La parte de este programa, que nos interesa ahora esta al principio del mismo:

```
#td trasera derecha
g.Dtdx= 0.616
g.Dtdy= -3.179
g.Dtdz=-0.364
```

Bien:

g.Dtdx es LocX.

g.Dtdy es LocY

g.Dtdz es LocZ

Por tanto, sustituimos los valores del **script**, por los de nuestra rueda en **Blender**:

g.Dtdx= -1.384

g.Dtdy= -3.179

g.Dtdz= -0.364

Como se ve, trabajar con programas en modo texto, no es tan complicado como a veces se dice. Para continuar, tenemos que hacer lo mismo, con las otras tres ruedas. Y para que la simulación sea realista, tenemos que hacer algunos cambios en el editor de juegos (**F4**).

Estos cambios, son "**Rigid body**", "**Actor**" y "**Bounds**" de tipo "**box**" solamente en el chasis del tanque. El primero es indicar al **BGE** que el objeto es rígido. El segundo, que sea detectable, sobre todo para las colisiones. Y el tercero, indica precisamente que haya colisiones, y que estas sean de tipo "**box**", o sea con forma de caja.

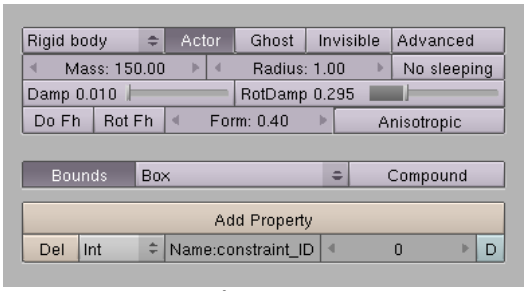

Figura 37

En cuanto a las ruedas, hay que asegurarse de que este seleccionado exclusivamente "**No collision**", en lugar de "**Rigid body**". Si ahora ejecutamos el modo de videojuegos "**p**", el vehículo debería, simplemente caer hasta el plano del suelo, y quedarse en reposo sobre él. Las ruedas, deberían estar en posiciones correctas. Son fallos típicos, los rebotes muy fuertes, vibraciones, etc. Casi siempre, la causa esta en una posición incorrecta de las ruedas. Se suele solucionar, separándolas un poco del chasis. Otro tema importante es el peso del vehículo. El valor que viene por defecto, es inadecuado. Yo he utilizado un valor de **150.00,** y el tanque se maneja con facilidad. Quizás demasiado fácilmente. Un valor más alto, podría dar la sensación de conducir un objeto más pesado, pero también se podría quedar atascado en desniveles pronunciados. Al fin y al cabo, nuestro tanque no tiene orugas.

Una buena costumbre, consiste, en nombrar a los objetos, desde el principio de forma lógica. Si así lo hacemos, nos ahorraremos muchos esfuerzos innecesarios. Si decidimos poner cualquier nombre a las partes de nuestro vehículo, tendremos que modificar los **scripts**, incluyendo estas nuevas denominaciones. Y es un proceso bastante tedioso. Mantener la nomenclatura original es lo más sencillo. Esta es la siguiente:

verde = chasis.
torreta = cañon.
VerdeDD = rueda delantera derecha
VerdeDI = rueda delantera izquierda.
VerdeTD = rueda trasera derecha.
VerdeTI = rueda trasera derecha.

En realidad, el objeto torreta no es llamado por los **scripts** en ningún momento. Es una división, que he realizado por mi cuenta, para poder animar la torreta y el cañón del tanque. Es divertido y sencillo dotarla de capacidad de giro. En el futuro, se puede dotar al cañón de capacidad de elevación y disparo. Todo ello, parece muy apropiado para un buen tanque.

De momento, tenemos un vehículo sin capacidad de movimiento. La verdad, es que si hemos nombrado adecuadamente las ruedas, estas deben ser capaces, de realizar todos aquellos movimientos programados en los **scripts** de **Python**. Pero un vehículo, no son solo sus ruedas, así que vamos a tener que trabajar un poco más. Seleccionamos el chasis, y mediante "**F4"** vamos al editor de juegos. Tenemos que hacer ocho sensores y cuatro controladores nuevos. Los tenemos que nombrar y relacionar entre si, exactamente como puede verse en la "Figura 38" .

Figura 38

Asusta un poco, pero no es para tanto. Como se puede imaginar, los cuatro controladores son los **scripts** de **Python**. Los sensores son tres "**Always**", o sea que se ejecuten siempre, y cinco "**Keyboard**" con las teclas que se van a utilizar en el juego para mover el tanque.

W = Adelante.

S = Atrás.

A = Izquierda.

D = Derecha.

Space = Frenar.

Los **scripts** se cargan creando un controlador, escogiendo como tipo de datos "**Python**", y escribiendo el nombre exacto del **script**. Estos, tienen que estar cargados en el editor de textos, y se guardan, junto con todo lo demás, en el fichero de extensión **.blend,** nativo de **Blender**. Así que, no tenemos que preocuparnos por ellos. Siempre podremos recuperar un **script,** desde un fichero de Blender, en el que hayamos hecho uso de él.

Para finalizar con este tema, seleccionamos la torreta del tanque. Se supone, que tiene que ser un objeto independiente, desde el principio de este ejercicio. En el editor de juegos (**F4**) hacemos lo siguiente:

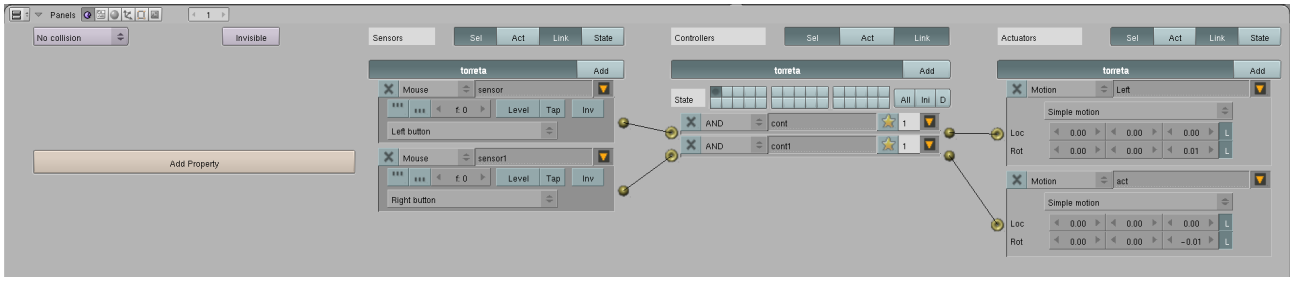

Figura 39

La torreta es más sencilla que el chasis. No hace uso de ningún guión de **Python**. No tiene colisiones, ni es influenciada por la física del juego. Simplemente, hace lo que le decimos. El botón izquierdo y derecho del ratón son sus sensores. Actuadores de tipo "**Motion**", proporcionan la rotación necesaria. Esta unida al chasis, mediante una simple operación padre-hijo. El chasis es el padre, y la torreta le sigue a todas partes. Como hemos visto anteriormente, para ello se utiliza "**Crtl p**".

Explicar con detenimiento, como hacen cada cosa los **scripts** de **Python**, excede los objetivos de este tutorial. De momento, puedes hacer uso de ellos. La forma más sencilla de aprender más, consiste en estudiarlos a fondo, y realizar pruebas, para examinar luego los resultados. Están muy bien comentados en castellano, lo cual es de una ayuda enorme. Son bastante sofisticados, ya que permiten modificar muchos aspectos de un vehículo. Entre ellos, la dirección, suspensión, dureza de los amortiguadores, resistencia al volcado, y muchas más cosas, que un experto en automoción comprenderá mucho mejor que yo.

También puedes ampliar el ejercicio, haciendo todo tipo de mejoras en la escena. Puedes hacer un mapa de normales para el terreno. Texturar el tanque de forma óptima (actualmente tiene un material básico), terminar la animación de la torreta. Dotarla de colisiones y demás. Y cualquier otra cosa que se te ocurra. Incluso puedes crear tanques enemigos, y acabar por crear un juego de tanques.

17- EL ARTE DE LA PELUQUERIA.

En capítulos anteriores, hemos creado nuestros propios personajes. Hemos aprendido a animarlos y a dotarlos de expresión. Pero, si nos hemos fijado con un poco de atención, habremos observado que en ningún momento, se ha hecho ningún comentario acerca del pelo. Como nuestros actores se modelaron deliberadamente en baja resolución, este "adorno natural", no era necesario. Pero lo cierto es, que el cabello humano es una parte muy expresiva, de nuestro cuerpo. Y que puede marcar la diferencia, en cuanto a la calidad de nuestros personajes.

Para el mundo del **3D** en tiempo real, este tipo de efectos supone todo un reto. Como siempre, resulta muy costoso en recursos hardware, durante la simulación. Aparte del trabajo adicional, que supone el realizarlo. Por suerte, **Blender** es un programa muy avanzado, y dispone de herramientas que nos ayudan en esta tarea. Para ello, existe un módulo, o apartado de edición de partículas, muy indicado en este tema. También puede utilizarse para crear hierva, humo, nubes, etc.

Pero, a día de hoy, el **BGE** no soporta partículas. No obstante, resulta útil generar pelo, mediante el editor de partículas. Para empezar, es bastante sencillo, obteniéndose unos resultados muy buenos. Para continuar, los resultados se pueden renderizar en forma de imágenes y vídeos. Estos, se pueden utilizar, por ejemplo, en las cinemáticas del juego. Las cinemáticas, son los vídeos de alta calidad, que sirven como introducción, o ayudan a desarrollar la historia entre nivel y nivel del mismo. Por último, la forma obtenida mediante el editor de partículas, puede tomarse como modelo para crear una cabellera convencional. Es decir, un objeto de forma similar a la primera, pero con una geometría simplificada.

Bien. Pasemos a la práctica. En este ejercicio vamos a dotar a un personaje de cabellera, con el editor de partículas. Después, modelaremos la misma, para que pueda visualizarse en le **BGE**. Para ello, cargamos a nuestro personaje preferido, en **Blender**. Yo voy a dotar de pelo a **Hari**, mi personaje femenino favorito. La verdad, es que encuentro poco atractiva su calvicie actual.

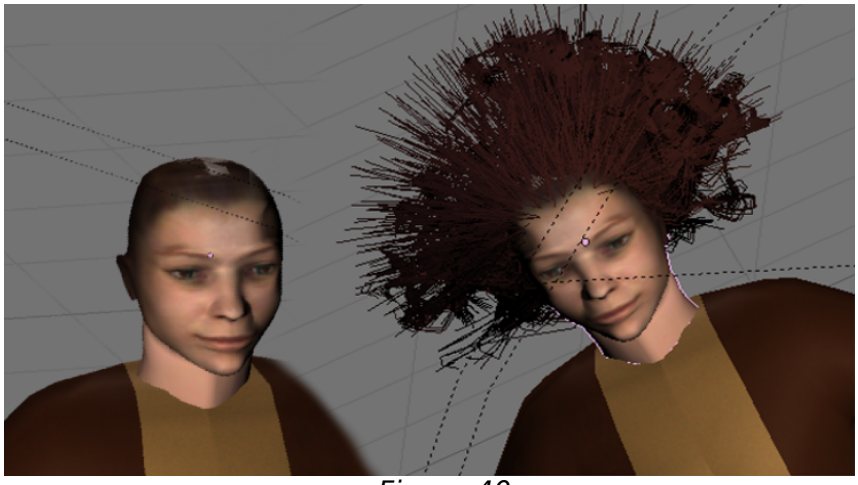

Figura 40

Lo primero que hacemos, es seleccionar al personaje. Después, en el modo "**Weight Paint**" (pintura de pesos), pintamos las partes desde las cuales, va a nacer el pelo. Por defecto, todo la figura será azul. Pintamos con cuidado, estas zonas en rojo. Hecho esto, mediante "**F7**" vamos al modo de objetos. Entonces, aparecerán dos nuevos íconos, en la barra inferior de la ventana de paneles. Estos son "**Physics buttons**" y "**Particle buttons**". Ambos nos interesan, pero de momento, vamos a utilizar el segundo. Este, muestra el panel "**Particle System**". Hacemos un nuevo sistema de partículas, y se pone a nuestro disposición, un completo editor con seis paneles independientes. Ver "Figura 41".

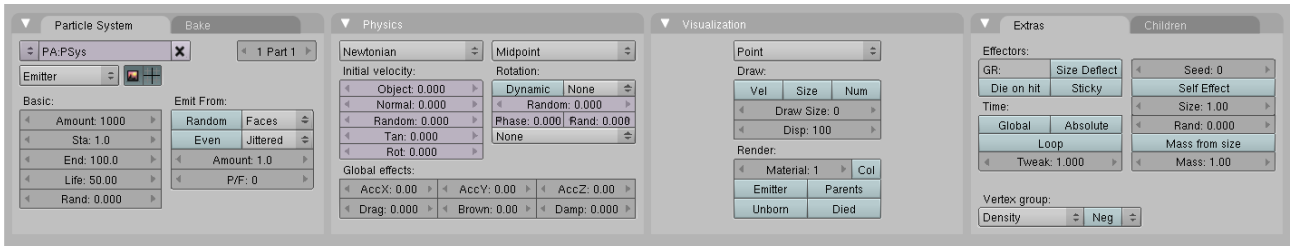

Figura 41

En el panel "**Particle System**", escogemos "**Hair**", en lugar de "**Emiter**". Ello, indica a **Blender,** que el sistema de partículas se va a utilizar para simular pelo. Por ello, algunos de los paneles de este apartado cambian, ofreciendo todo aquello, que pueda ser de utilidad para este cometido. Pero por el momento, no observaremos ningún cambio en el modelo.

En el mismo panel, se encuentra el botón "**Amount**". Este, controla la cantidad de pelo. Su valor por defecto es **1000**, pero lo podemos cambiar con toda tranquilidad a **1500** o **2000**. El panel "**Physics**", habitualmente a la derecha del primero, es clave a la hora de empezar a ver resultados. Si incrementamos el valor del botón "**Normal**", veremos que empieza a nacer el pelo. Este valor, es la altura del mismo. Pero también observaremos dos cosas que seguramente nos sorprendan un poco. La primera, es que el pelo parece crecer en todas partes del personaje. La segunda, que surge con un ángulo perpendicular al mismo, como si fuera de alambre.

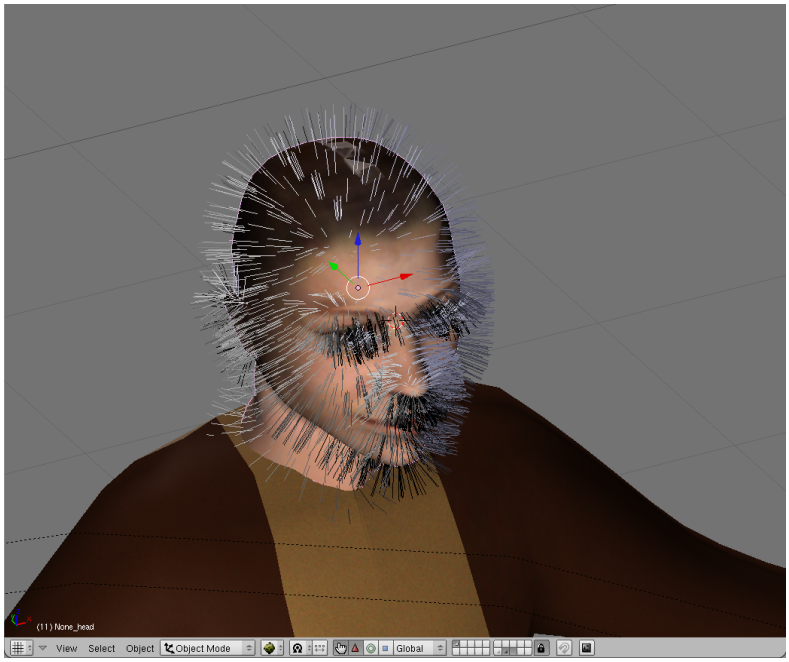

Figura 42

Todo ello es normal. Pero vamos a corregir esta "desagradable situación", lo antes posible. Para ello, nos desplazamos al apartado de edición (**F9**), y en el panel "**Link and Materials**", creamos un "**Vertex group**", o sea un grupo de vértices, y le ponemos un nombre. Yo lo he denominado "cabeza". Ver Figura 43. Después, regresamos al grupo de paneles de edición de partículas, En el panel "**Vertex group**" escogemos:

Density / Neg / Cabeza

La opción "**Neg**", hace que el pelo, se muestre en la zona que pintamos al principio de este tutorial de rojo, o que lo haga en las zonas originales azules. "**Cabeza**", es simplemente el nombre del grupo de vértices, que acabamos de crear.

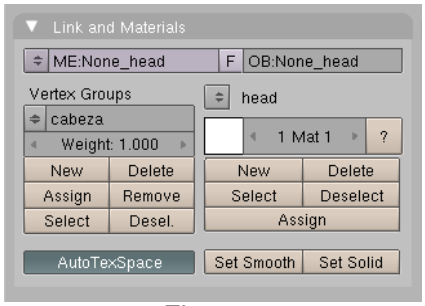

Figura 43

Gracias a esta operación, nuestro personaje ya dispone de algo parecido al pelo. Pero, sin duda es muy mejorable.

Pienso, que mejoraría mucho si lo peinásemos. Para ello, en el panel "**Particle System**", hacemos uso del botón "**Set editable**", que se transformara en "**Free edit**". O sea, que ya tenemos permiso para editar el pelo. Ahora, mediante la tecla "**n**", abrimos el panel emergente "**Particle edit Properties**". Escogemos "**Comb**", y ya estamos preparados para peinar. Ver Figura 44. El peine, es simplemente una circunferencia blanca.

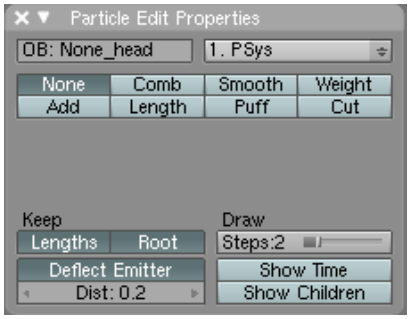

Figura 44

Para peinar la figura, tendremos que alternar sobre todo, entre la vista frontal y lateral. Habitualmente, requiere pasar varias veces, por el mismo lugar, como también ocurre en el mundo real. Cuando consideremos que hemos terminado, nos vamos a encontrar con otra situación inesperada. El cabello peinado, parece tener un espesor mínimo. De hecho, apenas es capaz de separarse de la forma del cráneo. No hemos cometido ningún error. Lo que ocurre es, que todavía no hemos terminado. Esta es, solamente una fase del trabajo. Basta, con cambiar del modo de partículas al modo de objetos, para que el peinado aumente su volumen considerablemente. A veces, incluso demasiado.

En modo de objetos, podemos experimentar con las posibilidades que nos brinda el programa. Los paneles "**Children**" y "**Extras**", son especialmente ricos en opciones. Observa las diferencias, entre "**Particles**" y "**Faces**", en el panel "**Extras**". Cualquiera de los botones del panel "**Children**", afecta de forma muy importante a la textura del pelo. Este, se puede volver revuelto y erizado, en lugar de suave y plano como hasta ahora. Lo mejor, es que experimentes por ti mismo todas las opciones.

Con un poco de práctica, podremos realizar peinados "de fantasía", con bastante facilidad. Los peinados más comunes, quizás se nos resistan un poco más. Pero al final, todo se logra. Y parece que este tema esta llegando a su fin. Pero si así pensásemos, nos equivocaríamos. El cabello humano, no esta constituido solo por forma, sino también por color.

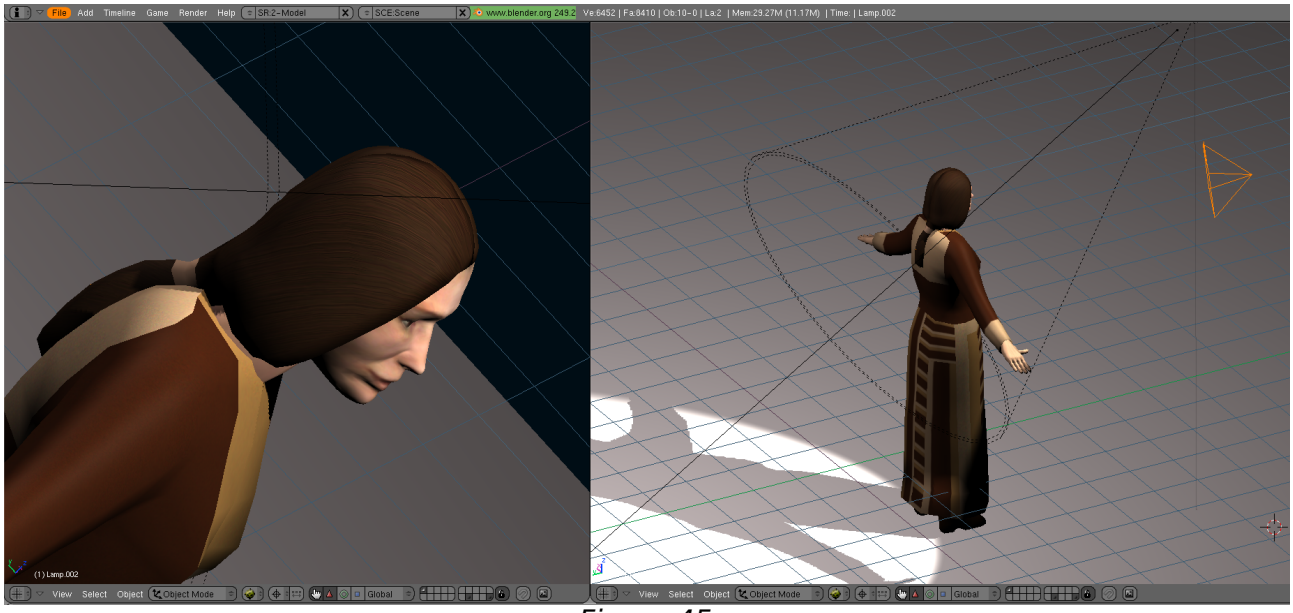

Figura 45

Así que, debemos dotar a nuestro cabellera, de un material apropiado, como con el resto del personaje. Para ello, seleccionamos el botón "**Col**", del panel "**Visualization**". Gracias a este, el pelo mostrará el color del material, que vamos a editar a continuación. Entonces, en el panel "**Material"** del editor de materiales (**F5**), escogemos los colores que queramos, para las casillas "**Col**", "**Spe**" y "**Mir**". De este modo, determinamos el color del pelo.

Si ahora pulsamos "**p**", veremos nuestro personaje en el **BGE**. Lamentablemente, **Hari** vuelve a padecer alopecia... El motor de juegos de **Blender**, no visualiza el pelo. Para poder hacer frente a esta contrariedad, solo podemos hacer una cosa. Modelar un objeto, que tenga la forma de la cabellera, y texturarla adecuadamente. El peinado creado, mediante el editor de partículas, será un modelo perfecto para esta tarea. La técnica empleada para modelar el pelo, no difiere para nada, de la empleada, en el proceso del modelado del propio personaje. Recordemos, que partiendo de un solo cubo, y mediante cortes y extrusiones, se logra generar la forma final. Ver "Figura 45".

El resultado, no es tan perfecto como podría alcanzarse con el editor de partículas, pero resulta convincente, para un personaje en baja resolución. Y algún día, el **BGE** visualizará los efectos de partículas, y entonces, podremos rentabilizar el esfuerzo que hemos dedicado a este tema.

18- FIN.

Todo lo que se empieza tiene un final. Y así, este tutorial llega a su conclusión. En el disco duro han quedado muchos temas. Por ejemplo, los menús y pantallas de configuración del juego. El uso de tipografías. Un sistema de guardar y cargar. La programación de la **IA** (inteligencia artificial), y varios asuntos más. Las cuales, justifican una continuación de este tutorial. Este nuevo tutorial, nos ayudaría a lograr dos objetivos adicionales. El primero, que la primera parte no tenga demasiada extensión. En mi opinión, ya empieza a ser demasiado pesada. Y el segundo objetivo, disponer de un poco más de tiempo, para adaptarme a la última versión del programa. Siempre es bueno estar actualizado. Como siempre, cuando esté disponible lo haré público.

Hasta entonces, como siempre, espero que este tutorial pueda ser útil para alguien. Para mí ya lo es.

Antonio Becerro Martínez.

mailto:littledog@es.gnu.org

Madrid. 14 de Junio de 2010.