Escuela Universitaria de Ingeniería Técnica de Telecomunicación

Trabajo Fin de Carrera

Estación gráfica 3D bajo entorno PC

Secretario Presidente Vocal

Autor: Javier Morales Comalat
Tutor: Dionisio Rodriguez Esparragón
Fecha: Septiembre 1995

INDICE

1. Introducción 1
2. Especificaciones generales 2
3. Equipos de altas prestaciones 4
3.1. Silicon Graphics 4
3.1.1. Serie Indy XZ 4
3.1.2. Otras Series 5
3.2. Ardent Computer 5
3.3. Apollo Computer 6
3.4. Stellar Computer 6
3.5. Virtual Studio 3D 6
4. Equipos adicionales para componer una estación videográfica 8
4.1. Tarjetas de vídeo 9
4.1.1. Color 9
4.1.2. Resolución 10
4.1.3. Velocidad 11
4.1.4. Dossier de tarjetas gráficas 12
4.2. Monitores de alta resolución 14
4.2.1. Principales tecnologías en monitores 15
4.2.2. Dossier de monitores 16
4.3. Tabletas gráficas 17
4.3.1. Dossier de tabletas gráficas 18
4.4. Scanner 20
4.4.1. Dossier de Scanner 20
4.5. Dispositivos de almacenamiento masivo 22
4.5.1. Discos duros como sistemas de almacenamiento 23
4.5.1.1. Dossier de discos duros 23
4.5.2. Unidades CD grabables 25
4.5.2.1. Cuadro resumen de los principales equipos CD 28

4.6. Controladores de grabación..... **29**
4.6.1. Controladores disponibles..... **30**
5. Targa+..... **31**
5.1. Requerimientos..... **31**
5.2. Posibles configuraciones..... **32**
5.3. Cables y conectores..... **36**
5.4. Configuración de Switches..... **42**
5.5. Configuración del software..... **45**
5.5.1. Configuración de la Targa+ para diferentes programas..... **48**
6. Proceso de construcción de una animación tridimensional..... **51**
6.1. 3D Editor..... **52**
6.2. 2D Shaper..... **61**
6.3. 3D Lofter..... **64**
6.4. Materials Editor..... **73**
6.5. Keyframer..... **78**
7. Software de producción digital..... **89**
7.1. Requerimientos del programa..... **90**
7.2. Videomaker+..... **91**
7.2.1. Setup panel..... **92**
7.2.2. Preview panel..... **93**
7.2.3. Switcher panel..... **94**
7.2.4. Control panel..... **98**
7.2.5. Solarize panel..... **100**
7.2.6. Keyer panel..... **102**
7.2.6.1. Blue matte key..... **103**
7.2.6.2. Luminance key..... **103**
7.2.6.3. Overlay..... **104**
7.2.7. Sequencer panel..... **105**
Apendice A. El formato FLI..... **107**
1. Tipos de bloques..... **109**
2. Librerías de funciones para manejar los ficheros FLI..... **111**
3. Listado del programa y librerías..... **115**

Apendice B. Listado de fabricantes y distribuidores..... **136**

Bibliografía.

1. Introducción.

La información gráfica es un tema que indiscutiblemente está de moda. No es concebible hoy por hoy, que las empresas no utilicen los medios audiovisuales que se disponen actualmente. Son muy numerosas las aplicaciones y por tanto, también lo son los diferentes usos que se pueden hacer de ellos. Por tanto, es necesario, adaptarse a las ventajas que nos aportan estas nuevas tecnologías.

La realización de este proyecto se debe fundamentalmente a una necesidad del Laboratorio de Televisión II de la Escuela Universitaria de Ingeniería Técnica de Telecomunicación de la U.L.P.G.C. por adaptar la docencia a las nuevas tecnologías de *vídeo digital.*

Esta memoria no pretende ser un manual de uso de los equipos y materiales disponibles en el Laboratorio, sino una pequeña introducción al fascinante mundo de las animaciones por ordenador y a sus aplicaciones en el campo profesional.

Esta memoria esta dividida en tres partes claramente diferenciadas. En un primer bloque, analizaremos que equipos son ideales para este trabajo, que caracteristicas tienen y cuales son los más ventajosos. También se incluye un dossier de cada uno de los elementos analizados, para una mejor comparación de las características de cada uno.

Un segundo bloque se utiliza para definir el material del que disponemos, tanto en *hardware* como en *software*, sus caracteristicas, y sus posibles mejoras para aumentar su rendimiento.

En una ultima parte, estudiaremos el proceso de construcción de una animación, analizando todos los pasos, desde el diseño de los elementos hasta obtener el resultado deseado en video.

2. Especificaciones generales.

Antes de realizar una valoración sobre los posibles equipos que serían necesarios, debemos plantearnos cuales van a ser nuestras necesidades para así, posteriormente, poder realizar una elección mucho más certera.

En primer lugar, deberemos elegir una máquina que sea muy rápida. Para ello podremos elegir entre máquinas con un único procesador muy rápido, o máquinas multiprocesadoras (varios procesadores en paralelo). Hay que tener en cuenta que las segundas serán mucho más rápidas aunque cada uno de los procesadores por separado no lo sean, ya que al compartir los trabajos, se liberan mutuamente de tareas no especificas del calculo de imágenes. Este es un aspecto muy importante a tener en cuenta, ya que la velocidad del sistema va a depender en gran medida del tipo de procesador que tengamos instalado.

Otro factor a tener en cuenta y que es de especial relevancia, es la cantidad de memoria de nuestra estación ideal, ya que todos los cálculos que se realicen, así como la ejecución de una animación, va a ser un proceso que trabajará directamente con la memoria. Una máquina que tenga que cargar una animación para ejecutarla y que posea poca memoria, usará el disco duro como memoria virtual, con la consiguiente ralentización del sistema debido a que la velocidad de transferencia de los discos es siempre mucho menor que la de las memorias. Si elegimos como plataforma una estación de trabajo, ésta deberá tener 32 Mb. de RAM como mínimo y, en un entorno PC, 16 ó 32 Mb serán más que suficientes.

Un tercer aspecto serán los sistemas de almacenamiento masivo. Hay que tener en cuenta que una imagen o *frame* relativamente compleja a una resolución de 720x576 (

standard Pal) puede ocupar del orden de 2 Mb y si la animación es de 10 segundos, el espacio necesario en disco para almacenarla será de:

2 Mb* 25 frames/sg * 10 sg = 500 Mb

Como se podrá comprender, no se puede tener almacenada toda esta información permanentemente en un disco duro, ya que con unas pocas animaciones que realicemos, lo habremos llenado. Por tanto debemos buscar una solución a este problema. En las estaciones de trabajo del tipo *Silicon Graphics* se suele resolver con un disco duro de varios gigabytes donde se almacena temporalmente la información y una vez terminadas las animaciones se trasladan o bien a un *streamer* o bien a un magnetoscopio.

Otras consideraciones importantes, son las características de las tarjetas de digitalización y de volcado a vídeo. Estas tarjetas deben incorporar sistemas de compresión y descompresión de imágenes por *hardware* para así reducir al máximo el espacio de almacenamiento. Normalmente las tarjetas existentes en el mercado son capaces de realizar estas técnicas de compresión en tiempo real, pero para imágenes con resoluciones muy pequeñas (del orden de 300x200). A su vez, deben ser compatibles con los actuales sistemas de televisión (PAL, NTSC, etc..), y permitir el volcado sincronizado a vídeo. Para ello, la tarjeta debe cumplir la norma RS-422A del E.I.A. (Asociación de Industrias de Electrónica).Esto nos permitirá volcar una animación a vídeo *frame* a *frame* según un código de tiempos. Esto puede realizarse por control serie o paralelo.

Las características comentadas anteriormente son las más importantes a la hora de elegir la estación ideal, pero no podemos olvidar otras, como son:

- Tipo de monitor a emplear.
- Sistemas *hardware* para apoyar el diseño (tabletas gráficas, caja de diales, etc.)
- Cámaras de fotografía y de vídeo opcionales.

3. Equipos de altas prestaciones.

Los equipos que vamos a analizar han sido creados específicamente para el diseño en tres dimensiones y por tanto poseerán, sin duda alguna, las mejores características que se le pueden pedir a una máquina de este tipo. La mayor parte de ellas suelen estar basadas en tecnología RISC y suponen un avance considerable con respecto a sistemas desarrollados sobre PC's.

3.1 Silicon Graphics.

3.1.1. Serie Indy.XZ

Incorporan un microprocesador R4600PC. Se trata de máquinas de 24 bits a color que incorporan un Z-buffer de 24 bits. Actualmente existen dos versiones claramente diferenciadas. Por un lado, la primera posee un microprocesador R4600PC con una memoria de 32 Megabytes y un disco duro de 1 gigabyte. Incorpora también una videocámara digital IndyCam y el sistema operativo Irix 5.2., basado en el sistema operativo Unix, pero diseñado especialmente para este tipo de trabajos. Por lo que respecta a la segunda configuración, lleva un microprocesador R4600SC a 150 Mhz y, como características fundamentales, posee 256 Mb de memoria incorporados directamente de serie y un disco de 2 gigabytes de almacenamiento, así como siete ranuras tipo SCSI-2.

Asimismo, esta línea viene con la gama de productos *3D Graphics Acelerator*, que solo se incorporaban a los equipos grandes de esta misma marca. A este respecto, se ha incluido en estos modelos la utilidad *Geometry Engine*, para manipular objetos 3D dentro de imágenes de gran formato. De esta forma, el sistema está capacitado para procesar hasta 128 Mflops sobre un bus altamente integrado de 267 Mbps.

3.1.2. Otras series.

La serie Indy, anteriormente analizada, es la estación mas sencilla dentro de los equipos que posee Silicon Graphics. Existen otras estaciones que poseen mayores prestaciones con elevadas capacidades de proceso.

Las series "Indigo" e "Indigo 2" son estaciones de sobremesa con un increíble poder gráfico. La Indigo 2 es capaz de dibujar 630.000 polígonos sombreados y 415.000 con sombreados goraud. Contiene dos CPUs basadas en el microprocesador R4400 con una velocidad interna de reloj de 150 Mhz, dos canales SCSI con una transferencia de 10 Mbps y un bus con arquitectura de 64 bits. Contiene además un *frame buffer* y un *Z-buffer*, ambos de 24 bits.

Existen otras series que superan a la Indigo, como la "Crimson", con unas capacidades gráficas que pueden exceder del millón de triángulos. A ésta le sigue la serie Onix, que puede contener desde 2 hasta 24 procesadores R4400 con 126 MIPS por cada CPU. y puede dotarse de nuevas arquitecturas gráficas como la VTX, ofreciendo hasta un millón de polígonos y 80 millones de pixels texturados por segundo.

3.2 Ardent Computer.

El ordenador Titán es el máximo exponente de esta compañía. Esta estación puede estar configurada como una única unidad central o hasta cuatro iguales funcionando en paralelo.

El *hardware* de estas estaciones consiste en una CPU (puede llegar hasta cuatro), más de 128 Megas de memoria, dos tarjetas gráficas y una tarjeta de I/O.

Cada CPU se compone de una unidad vectorial, una unidad de enteros y 32 K de memoria caché. Todas las CPU's trabajan en paralelo y no es necesario que ninguna de ellas trabaje como servidor, con lo que se evitan los cuellos de botella al gestionar todo un único procesador. Todas estas características permiten que cada procesador trabaje a una velocidad máxima de 16 Mflops, por lo que con cuatro procesadores alcanzaremos hasta 64 Mflops.

El subsistema gráfico está compuesto por 10 procesadores, de los cuales 8 son procesadores de pixels y dos de polígonos, 52 planos de memoria a pantalla y 16 planos de z-buffer.

3.3 Apollo Computer.

Su estación gráfica principal es la DPS 10000 con una CPU RISC a 20 MIPS., cuya estructura interna permite instrucciones de 64 bits.y tiene diferentes caminos paralelos para instrucciones enteras y en coma flotante. La velocidad de cada CPU es de 20 MIPS y 10 MFLOPS pudiendo aceptar hasta 4 CPU's.

3.4. Stellar Computer.

Este ordenador gráfico emplea procesadores con arquitectura *pipeline* para incrementar la velocidad, llegándose a obtener hasta 20 MIPS y 40 MFLOPS. En términos gráficos puede representar 800.000 vectores/sg y 150.000 polígonos iluminados por segundo.

3.5 Virtual Studio 3D.

Este sistema está basado en un PC con un procesador *Pentium* a 60 ó 66 Mhz. Lleva incorporados 16 Mb de memoria RAM, pudiendo ser ampliados en placa hasta 128. Posee una tarjeta gráfica con una resolución virtual de 2048x870 ppp y 16 millones de colores. Incorpora un disco duro SCSI de 1080 Mb. como único sistema de almacenamiento, y un monitor de alta resolución de 17 pulgadas.

Para volcar a vídeo, el sistema incorpora una tarjeta Targa+ PAL con 2 Mb y un controlador de magnetoscopio por software SOFT-VTR.

Como opción puede incorporársele un magnetoscopio profesional que permita la grabación cuadro a cuadro.

4. Equipos adicionales para componer una estación videográfica.

Cualquier estación destinada a la producción de vídeo, no sólo debe ser capaz de generar animaciones de una forma eficiente, sino que debe incorporar todas las facilidades posibles para desarrollar este complejo proceso. Cualquier estación debe permitir la comunicación con el exterior para poder volcar a un magnetoscopio todas las animaciones que hayan sido realizadas.

Independientemente del *hardware* descrito anteriormente, una estación videográfica necesita:

- Monitor gráfico de alta resolución.
- Tarjeta de vídeo para el control del monitor.
- Codificador/decodificador de señales PAL y NTSC.
- Dispositivo de almacenamiento adicional para salvar las animaciones.
- Tableta gráfica para dibujar.
- Controlador de VTR.
- Sistemas para incorporar imágenes (Scanner y cámara de fotografía).

4.1 Tarjetas de vídeo.

Para entender mejor todo el conjunto que forma una estación videográfica, es necesario comprender los entresijos de los elementos que la forman.

Los programas muestran datos e imágenes que se visualizan en la pantalla del monitor. Este proceso es posible gracias al sistema de vídeo del ordenador, que consta de tres partes:

- Monitor.
- Programas controladores (*drivers*).
- Tarjeta gráfica.

Los tres tienen que estar coordinados entre sí. No tiene sentido una tarjeta capaz de trabajar a 1024x768 con 256 colores y un monitor monocromo. En el caso de que uno de los tres elementos anteriores tenga capacidades menores que los otros dos, será éste el que defina los valores máximos que se pueden conseguir.

El monitor visualiza la información que genera el programa, está controlado por la tarjeta gráfica y no influye en la velocidad del sistema. Los controladores son pequeños programas encargados de que el programa principal se pueda comunicar con la tarjeta gráfica, o dicho de otro modo, hace de interfaz entre el programa que estamos ejecutando y el *hardware* de la tarjeta. En cuanto a las tarjetas gráficas, los puntos más importantes a tener en cuenta son: color, resolución y velocidad.

4.1.1. Color.

El número de colores que una tarjeta puede mostrar en pantalla se denomina "paleta de colores". Mientras que la resolución determina las coordenadas X e Y de la pantalla, la profundidad de cada pixel (Z) describe el color que contiene. La

información de color de cada *pixel* de la pantalla se guarda en memoria y, por tanto, mientras más colores empleemos, más memoria será necesaria. En la siguiente tabla se muestran estos valores.

NECESIDADES DE MEMORIA DE VIDEO					
Modo	VGA	SVGA	SVGA	VGA Ext.	VGA Ampl.
Resolución	640x480	800x600	1.024X768	1.280x1.024	1.600x1.200
Nº de Pixels	307.200	480.000	768.432	1.310.720	1.920.000
16 Colores-4 bits					
Nº bytes	153.600	240.000	393.216	655.360	960.000
Mem.Video necesaria	256 Kb	256 Kb	512 Kb	1Mb	1Mb
256 Colores-8 Bits					
Nº bytes	307.200	480.000	786.432	1.310.720	1.920.000
Mem.Video necesaria	512 Kb	512 Kb	1 Mb	1.5 Mb	2 Mb
32.468 Col.-15 bits					
Nº bytes	576.000	900.000	1.474.560	2.457.600	3.600.000
Mem.Video necesaria	1 Mb	1 Mb	2 Mb	3Mb	4 Mb
65.536 Col-16 bits					
Nº bytes	614.400	960.000	1.572.864	2.621.440	3.840.000
Mem.Video necesaria	1Mb	1 Mb	2 Mb	3 Mb	4 Mb
16.777.216 Col-24 bits					
Nº bytes	921.600	1.440.000	2.359.296	3.932.160	5.760.000
Mem.Video necesaria	1 Mb	1.5 Mb	2.5 Mb	4 Mb	6 Mb
16.777.216 Col-32 bits					
Nº bytes	1.228.800	1.920.000	3.145.728	5.242.880	7.680.000
Mem.Video necesaria	1.5 Mb	2 Mb	3.5 Mb	5.5 Mb	8 Mb

El número de colores que puede tener un punto de la pantalla depende de los bits que tenga asignado (se calcula mediante color=2^n, siendo n el número de bits). Algunas tarjetas dedican más de 24 bits a cada punto, pero no proporcionan más de 16.7 millones de colores. Los restantes bits se dedican a otras tareas como son superposición, transparencia, *render*, etc..

4.1.2. Resolución.

Otro de los factores que determinan la calidad de un sistema de vídeo es la resolución, es decir, la densidad de puntos en pantalla. Mientras más puntos o *pixels*

posea la pantalla, más nítida se verá una imagen. Esta relación entre pixels también tiene gran importancia en lo que respecta a la proporción de las figuras. La gran mayoría de las pantallas del mercado actual mantienen una relación 4:3 entre sus medidas horizontal y vertical, independientemente del tamaño del monitor. Si no se mantienen estas proporciones, un cuadrado dibujado en pantalla tomará el aspecto de un rectángulo. No obstante, un buen monitor permite modificar la escala H:V efectiva para así poder variar la relación de aspecto.

Esta rejilla de puntos debe ser actualizada cada cierto tiempo, tanto para refrescar la imagen como para variar su contenido. Las frecuencias de refresco deben adecuarse a la resolución del sistema. No es lo mismo actualizar 480 líneas de 640 puntos, que 1024 líneas de 1280 puntos, ya que el proceso debe realizarse sin que se note el barrido de los haces de electrones.

4.1.3. Velocidad.

Cuando utilizamos programas de gestión el sistema de vídeo absorbe una parte del tiempo, que puede andar por el 10% del total, pero cuando trabajamos en entorno gráfico, el porcentaje aumenta como mínimo un 25%. Esto nos servirá para dar una idea de la importancia de la velocidad de un sistema de vídeo cuando se le exigen modos gráficos.

La velocidad de un sistema de vídeo depende de múltiples factores. En realidad depende de todos los elementos que intervienen en el funcionamiento de un programa, incluido el propio programa. Los parámetros de la tarjeta de video que influyen a este respecto son: El tipo de memoria, el procesador, el tipo de conexión y los controladores.

La memoria que normalmente se encuentra en una tarjeta de vídeo es del tipo DRAM (RAM Dinámica). Es monotarea y dispone de un único canal de comunicaciones. Si se encuentra enviando una imagen a pantalla, el sistema debe esperar a que termine

para poder escribir una nueva descripción de la imagen. Existe otro tipo de memoria, y se denomina VRAM (RAM de vídeo). Esta última es más veloz que la anterior, ya que dispone de dos canales de comunicaciones. No se suele incorporar debido a que es más cara que la primera.

Los procesadores que se incluyen en las tarjetas gráficas son de tres tipos diferentes: Controladores de imagen, coprocesadores aceleradores y coprocesadores programables.

Las tarjetas que incorporan un coprocesador acelerador o de funciones fijas incluyen en sus características funciones gráficas especificas grabadas en el propio chip, descargando al procesador principal de parte del trabajo.

Los procesadores gráficos son capaces de realizar cualquier tarea de forma similar al procesador central, dependiendo de las características del programa controlador que se instale. Suelen disponer de los dos tipos de memoria antes mencionados.

4.1.4. Dossier de tarjetas gráficas.

A continuación, se muestra un listado de algunas de las tarjetas gráficas que existen en el mercado. Las características expuestas en ellas son:

CARACTERÍSTICAS	DESCRIPCIÓN
Modelo	Nombre de la tarjeta
Conector	Tipo de ranura de expansión que necesita la tarjeta de vídeo
Memoria base/máxima	Cantidad de memoria que incorpora la tarjeta y memoria máxima que admite.
VESA	Indica si la tarjeta soporta el standard VESA ya sea por software (Sw) o por hardware (Hw)
Aceleradora	Especifica si se trata de una tarjeta aceleradora que trabaja a mayor velocidad
Resolución Gráfica	Resolución máxima en modo gráfico

CARACTERÍSTICAS	DESCRIPCIÓN
Resolución texto	Resolución máxima en modo texto
Colores	Numero máximo de colores que puede conseguir
Distribuidor	Nombre del distribuidor de la tarjeta.

Modelo Tarjeta	onector	Memoria Base/Máxima	VESA	Aceleradora	Resolución Gráfica	Resolució Texto	Colores	Distribuidor
Oti-077	ISA-16	512K/1M DRAM	No	Si	1280 x 1024 NE	N/D	256	Micro-Exe
VC 511/512 VGA	ISA-16	512K DRAM	No	No	1024 x 768 NE	N/D	256	Micro-Exe
Cirrus Logic CL20	ISA-16	1M/2M DRAM	No	Si	1024 x 768	N/D	16,7 M.	Lider Shop
Trident VGA ISA 1M	ISA-16	1M/1M VRAM	No	No	1024 x768	N/D	-	MCB Informática
UMC Tari VGA 1M	ISA-16	1M/1M VRAM	Hw	No	1024 x 768	80 x 25	256	Foxen Computer
VC 415 VGA	ISA-16	1M/1M DRAM	No	Si	1280 x 1024 NE	N/D	16,7 M.	Micro-Exe
VC 826	VESA	1M/1M DRAM	Hw	Si	1280 x 1024 NE	N/D	16,7 M.	Micro-Exe
Cirrus Logic CL28	VESA	1M/2M VRAM	Hw	Si	1280 x 1024 E	132 x 43	16,7 M.	Lider Shop
Trident VGA 9400	VESA	1M/2M DRAM	Sw	Si	1280 x 1024 E	132 x 60	16,7 M.	Think Company
Soyo GD 5428	VESA	1M/2M DRAM	Sw	Si	1280 x 1024 E	132 x 43	16,7 M.	Think Company
CirrusLogic VGA VES	VESA	1M/2M DRAM	No	Si	1280 x 1024	N/D	16,7 M.	MCB Informática
Micro Christal 10 SP	VESA	1M/2M DRAM	Sw	Si	1280 x 1024 E	N/D	16,7 M.	D.A.T.
ALI PCI	PCI	1M/2M VRAM	No	Si	1280 x 1024	N/D	16,7 M.	MCB Informática
T9200	VESA	1M/2M DRAM	Hw	Si	1280 x 1024 NE	N/D	16,7 M.	Micro-Exe
UMC Tari VGA VESA	VESA	1M/1M VRAM	Hw	Si	1280 x 1024	182 x 43	16,7 M.	Foxen Computer
S3/805	VESA	1M/2M DRAM	Hw	Si	1280 x 1024 NE	132 x 43	16,7 M.	Lider Shop
Taken Co. Fast View	VESA	1M/2M DRAM	Hw	Si	1280 x 1024 NE	N/D	16,7 M.	Venta
Acer S3P86C805	VESA	1M/2M DRAM	Sw	Si	1280 x 1024 E	132 x 43	16,7 M.	Think Company
Vermont Master VGA	VESA	1M/2M DRAM	Sw	Si	1024 x 768 NE	N/D	16,7 M.	CIOCE
GUI Accelerator	EISA	1M/1M VRAM	Sw	Si	1280 x 1024 NE	132 x 43	65536	Foxen Computer
IBM Radius XGA-2	ISA-16	1M/1M VRAM	Hw	Si	1024 x 768 NE	1118x400	65536	IBM
ATI Graphics Ultra Pro	ISA-16	2M/2M VRAM	Hw	Si	1280 x 1024 NE	N/D	16,7 M.	Ingram Micro
ATI Graphics Ultra Pro	VESA	2M/2M VRAM	Hw	Si	1280 x 1024 NE	N/D	16,7 M.	Ingram Micro
IBM Adaptador XGA-2	MCA	1M/1M VRAM	Hw	No	1360 x 1024 NE	1118x480	65536	IBM
ATI Graphics Ultra Pro	EISA	2M/2M VRAM	Hw	Si	1280 x 1024 NE	N/D	16,7 M.	Ingram Micro
ATI Graphics Star Mach	VESA	2M/2M VRAM	No	No	1280 x 1200	N/D	16 M.	MCB Informática
Matrox Graphic Star	PCI	2M/2M VRAM	No	Si	1280 x 1200	N/D	16 M.	MCB Informática
AMI Fast View VESA	VESA	2.2M/2.2M V-D RAM	Hw	Si	1280 x 1024 NE	N/D	16,7 M.	Venta Microinfor.
AMI Fast View VESA	PCI	3M/3M V-D RAM	Hw	Si	1280 x 1024 NE	N/D	16,7 M.	Venta Microinfor.
N° Nine GXE 64 PRO	VESA	4M/4M VRAM	Hw	Si	1800 x 1200 NE	N/D	16,7 M	Ingram Micro
N° Nine GXE 64 PRO	PCI	4M/4M VRAM	Hw	Si	1800 x 1200 NE	N/D	16,7 M.	Ingram Micro
N° Nine GXE 16 ISA	ISA-16	4M/4M VRAM	N/D	Si	1800 x 1200 NE	N/D	16,7 M.	Ingram Micro
Matrox MGA Impresion	ISA-16	3M/3M V-D RAM	Hw	Si	1600 x 1200 NE	N/D	16,7 M.	Ingram Micro
Matrox MGA Plus	PCI	4M/4M VRAM	Hw	N/D	1600 x 1200 NE	N/D	16,7 M.	Ingram Micro
Vermont Micro. Cobra	ISA-16	1M/2.5M DRAM	No	Si	1024 x 768 NE	N/D	16,7 M.	CIOCE
IBM Image Adapter	MCA	3M/3M	Hw	No	1600 x 1200 NE	720 x 480	65.536	IBM

Modelo Tarjeta	onector	Memoria Base/Máxima	VESA	Aceleradora	Resolución Gráfica	Resolució Texto	Colores	Distribuidor
N° Nine GXI 39 TC	ISA-16	5M/5M V-D RAM	Hw	Si	1800 x 1200 NE	N/D	16,7 M.	Ingram Micro
Matrox MGA Impres.	VESA	3M/3M V-D RAM	Hw	Si	1280 X 1024 NE	N/D	16,7 M.	Ingram Micro
Vermont Micro. X-ser.	ISA-16	1M/3.5M DRAM	No	Si	1280 x 1024 NE	N/D	16,7 M.	CIOCE
VGA - 16M	ISA-16	1M/1M DRAM	Sw	No	1024 x 768 E	N/D	256	Sintronic
VGA - ACC	ISA-16	1M/1M DRAM	Sw	Si	1024 x 768 E	N/D	256	Sintronic
VGA - TI	VESA	1M/2M DRAM	Sw	Si	1024 x 768 NE	N/D	16,7 M.	Sintronic

4.2 Monitores de alta resolución.

El monitor es uno de los dispositivos más importantes a la hora de trabajar con un ordenador. Dependiendo del tipo de trabajo que se va a realizar con él, es necesario una gran fidelidad en la gama de colores, o bien, un tamaño adecuado para una mejor visualización del trabajo.

A la hora de elegir un monitor es necesario tener en cuenta la combinación que debe existir entre la tarjeta gráfica y el monitor. Otro factor importante son los controles de pantalla, que deben ser: brillo, contraste, convergencia, efectos de almohadilla y la posibilidad de cambiar el tamaño de la pantalla.

El aspecto a tener en cuenta para trabajar con mayor comodidad es que el monitor posea una capa antirreflejos y antiestática, con lo cual evitaremos, por un lado, los brillos sobre la pantalla producidos por la incidencia de luz que perjudiquen la visibilidad de la imagen y, por otro lado, que ofrezca un aspecto engañoso sobre los colores mostrados.

La cantidad y el tipo de interfaces de conexión para la señal de vídeo también es un aspecto importante. La gran mayoría de los monitores de altas prestaciones incluyen únicamente conexión mediante BNC. En algunos casos las señales de sincronismo horizontal y vertical son portadas por dos cables diferentes, mientras que en otros, los dos sincronismos van en el mismo cable.

4.2.1 Principales tecnologías en monitores.

En la actualidad los principales fabricantes de monitores emplean dos tipos diferentes de tecnologías en la elaboración de los tubos de rayos. La calidad resultante del monitor dependerá en gran medida de la tecnología empleada.

En el caso del fabricante Hitachi, el tubo que se monta en sus modelos es el denominado *Invar Shadow Mask*. En él se emplea una máscara de sombra para asegurar el posicionamiento de los rayos relativos a los fósforos de forma precisa. En ciertos tubos puede producirse una distorsión de la máscara debida a una expansión térmica, lo cual deriva en una disminución de la pureza del color representado en pantalla. Para evitar este problema, Hitachi emplea un metal denominado *Invar*, que tiene un coeficiente mínimo de expansión térmica.

El cañón empleado por Hitachi es el FST de tres puntos, que proyecta cada uno de los rayos sobre una pantalla ligeramente esférica que, según el fabricante, proporciona al usuario un aspecto de punto redondo más conseguido que los puntos elípticos ofrecidos por los TRC que proyectan sobre una superficie cilíndrica.

La otra tecnología que se contrapone a la anterior es la empleada por SONY con sus tubos de tecnología trinitron, basada en el conjunto de cañón y lente única que ofrece un enfoque muy preciso. A diferencia del sistema de máscara, la tecnología trinitron se basa en una rejilla de aperturas verticales contínuas que posibilitan que los tres haces de color incidan sobre el punto de fósforo asociado. Para estabilizar dicha rejilla, los monitores disponen de dos hilos que la atraviesan horizontalmente.

4.2.2. Dossier de monitores.

A continuación se muestra un *dossier* de los principales monitores que se encuentran en el mercado, detallándose en cada uno de ellos sus características principales. Las características que se mostrarán, serán:

CARACTERÍSTICAS	DESCRIPCIÓN
Monitor	Nombre del fabricante y modelo del monitor
Tamaño	Tamaño del monitor en pulgadas
Barrido vertical	Rango de frecuencias que soporta para realizar el barrido vertical
Barrido horizontal	Rango de frecuencias que soporta para realizar el barrido vertical. Dependiendo de estas frecuencias, se definirán las diferentes resoluciones.
Resolución máxima	Resolución máxima que alcanza, indicando también el modo en el lo hace (E=Entrelazado y EN= No Entrelazado)
Punto	Tamaño del pixel medido en milímetros.
Distribuidor	Nombre del distribuidor

Monitor	Tamaño	B. Vertical	B. Horizontal	Res. máxima	Punto	Distribuidor
Monitores de 15 pulgadas						
Philips Brillance 1520	15"	50-100 Hz	35-58 Khz	1024 x 768 NE	0.28	Compumarket
AST Super VGA LR-15	15"	50-90 Hz	48-57 KHz	1024 x 768 NE	0.28	Cioce
Sampo Super VGA 15FS	15"	50-90 Hz	30-60 Khz	1280 x 1024 E	0.28	Dell España
Mitsubishi Scan 15FS	15"	50-90 Hz	30-63 KHz	1024 x 768 NE	0.28	Compumarket
Philips 4CM 8270	15"	50-100 Hz	30-58 Khz	1024 x 768 NE	0.28	Venta Microinformatica
Norgate MC SVGA 15	15"	50-120 Hz	30-64 Khz	1280 x 1024 NE	0.28	SCS
Brother BM-85 L	15"	50-100 Hz	N/D	1280 x 1024 E	0.28	Cia equipos oficina
Investrónica CM-1564	15"	50-90 Hz	50-90 KHz	1280 x 1024 NE	0.28	Investrónica
Fijitsu Color FCM-255L	15"	50-100Hz	30-64 KHz	1280 x 1024 NE	0.28	Fujitsu España
NEC Multisync 3V	15"	55-90 Hz	31-50 KHz	1024 x 768 NE	0.28	NEC Ibérica
Siemens MCM 1503	15"	55-90 Hz	30-62 KHz	1024 x 768 E	0.28	Siemens-Nixdorf
Metrologie	15"	55-100 Hz	30-58 KHz	1024 x 768 NE	0.28	Metrologie
ICL ErgoPro VE-15C	15"	48-100 Hz	30-62 KHz	1280 x 1024 E	0.28	ICL
IBM Color 9525X	15"	55-110 Hz	30-64 KHz	1280 x 1024 NE	0.28	IBM
NEC Multisync 4FGe	15"	55-90 Hz	31-62 KHz	1024 x 768 NE	0.28	NEC Iberica
AOC CM-536	15"	45-90 Hz	29.5-64 KHz	1280 x 1024 NE	0.28	Sintronic
Monitores de 17 pulgadas						
AST Super VGA LR-17	17"	55-90 Hz	48-57 KHz	1280 x 1024 NE	0.28	Cioce

Monitor	Tamaño	B. Vertical	B. Horizontal	Res. máxima	Punto	Distribuidor
Philips 1720	17"	50-120 Hz	30-82 KHz	1280 x 1024 NE	0.27	Compumarket
Mag InnoVision MX17F	17"	50-120 Hz	30-68 KHz	1280 x 1024 E	0.26	Lider Shop
Philips 4CM 4770	17"	50-100 Hz	30-58 KHz	1024 x 768 NE	0.31	Venta MicroInformatica
Siemens MCM 1701	17"	55-90 Hz	30-80 KHz	1280 x 1024 E	0.28	Siemens-Nixdorf
Chuntex 1760 LR	17"	50-100 Hz	30-65 KHz	1280 x 1024 NE	0.27	Comelta
Foxen EM-1760	17"	35-80 Hz	31.5-56 KHz	1024 x 768 E	0.31	Foxen Computer
Noman MD-17	17"	N/D	N/D	1280 x 1024 NE	0.28	Noman
Sony 1730	17"	55-110 Hz	28-58 KHz	1024 x 768 NE	0.25	Compumarket
Mitsubishi SVGA 17F	17"	50-130 Hz	30-64 KHz	1280 x 1024 NE	0.26	Dell España
IBM Color 17S	17"	43.5-88 Hz	31.5-63 KHz	1280 x 1024 NE	0.28	IBM España
ICL Ergo Pro 17C	17"	48-100 Hz	30-82 KHz	1280 x 1024 NE	0.26	ICL
NEC Multisync 5FGe	17"	55-90 Hz	31-62 KHz	1024 x 768 NE	0.28	NEC Iberica
Philips 4CM 6282	17"	50-120 Hz	30-82 KHz	1280 x 1024 NE	0.27	Venta MicroInformática
Visa MC 8750	17"	50-90 Hz	30-66 KHz	1280 x 1024 NE	0.28	Diode España
IBM Color 9527X	17"	50-110 Hz	30-82 KHz	1360 x 1024 NE	0.26	IBM España
NEC Multisync 5FG	17"	55-90 Hz	27-79 KHz	1280 x 1024 NE	0.28	NEC Iberica
AOC CM-735 LB	17"	50-90 Hz	30-64 KHz	1280 x 1024 NE	0.26	Sintronic
AOC CM-736 LB	17"	50-90 Hz	31.5-75 KHz	1280 x 1024 NE	0.26	Sintronic
Monitores de 20 pulgadas o más						
TyStar 2015	20"	48-90 Hz	30-65 KHz	1024 x 768 NE	0.26	Compumarket
Foxen 20C-35	20"	40-120 Hz	31-35.5 KHz	1024 x 768 E	0.31	Foxen Computer
Noman MS-7A	20"	N/D	29-60 KHz	1280 x 1024 E	0.31	Noman
Philips 4CM 2799	20"	50-120 Hz	30-64 KHz	1280 x 1024 NE	0.31	Venta Microinformática
Foxen 20C 3060	20"	40-120 Hz	30-60 KHz	1280 x 1024 NE	0.31	Foxen Computer
Sony 2036	20"	50-120 Hz	30-71 KHz	1280 x 1024 NE	0.30	Compumarket
Sony 2038	20"	50-160 Hz	28-85 KHz	1024 x 768 NE	0.30	Compumarket
Philips C 2082 DAS	20"	50-160 Hz	30-82 KHz	1280 x 1024 NE	0.31	Venta Microinformatica
Investrónica CM 2039M	20"	50-160 Hz	28-85 KHz	1280 x 1024 NE	0.30	Investrónica
Mitsubishi SVGA 21FS	21"	40-130 Hz	30-78 KHz	1280 x 1024 NE	0.28	Dell España
Siemens MCM 2102	21"	50-160 Hz	30-82 KHz	1600 x 1200 E	0.28	Siemens-Nixdorf
NEC Multisync 6FG	21"	55-90 Hz	27-79 KHz	1280 x 1024 NE	0.28	NEC Iberica
Philips C2182 DAS	21"	50-160 Hz	30-82 KHz	1280 x 1024 NE	0.27	Venta Microinformatica
IBM Color 9521X	21"	50-110 Hz	30-82 KHz	1600 x 1200 NE	0.31	IBM España
Hantarex CT 28 EV	28"	48-120 Hz	31-38 Khz	1024 x 768 E	0.80	Hantarex Iberica
Hantarex CT 34 EV	34"	48-120 Hz	31-38 KHz	1024 x 768 E	1.00	Hantarex Iberica

4.3 Tabletas gráficas.

Las tabletas gráficas son una herramienta muy útil para las personas encargadas de generar los dibujos. Es una especie de pizarra electrónica con un lápiz o puntero con

el que el dibujante realiza su tarea de forma más cómoda que como lo haría con un ratón.

Existen diferentes tipos de tabletas según el tipo de tecnología que utilicen. Estas pueden ser: electrostática, electromagnética, óptica y ultrasónica.

Las tabletas electromagnéticas permiten una alta resolución pero requieren un mayor consumo de corriente, además de tener problemas con la sensibilidad en función del ángulo existente entre la superficie de la tableta y el lápiz. En las electrostáticas existen debajo de la superficie trazas o líneas que generan pulsos. Estos pulsos son recogidos por el extremo del lápiz y así puede determinarse de una forma precisa su posición exacta.

Una mejora en las tabletas electromagnéticas consiste en la utilización del trazado permutado. Esto quiere decir que no se genera un pulso por traza, sino que se generan ocho pulsos solamente y se varía su orden secuencial entre las trazas. Con estas secuencias sabemos donde está el lápiz, sin embargo la resolución que se obtiene es muy mala. Para solucionar este problema, cada traza se divide en 256 regiones diferentes.

El lápiz se compone de varios condensadores y un pequeño amplificador que recoge los pulsos de las trazas.

4.3.1. Dossier de tabletas gráficas.

Las características generales de cualquier tableta gráfica pueden resumirse en las siguientes:

CARACTERÍSTICAS	DESCRIPCIÓN
Tamaño	Tamaño del arrea de trabajo en pulgadas de altura por pulgadas de anchura
Exactitud	Distancia mínima que puede reconocer la tableta medida en pulgadas

CARACTERÍSTICAS	DESCRIPCIÓN
Resolución	Resolución de la tableta medida el líneas por pulgada. Es preferible valores altos.
Cursor	Indica el número de botones que posee el cursor
Lápiz	Indica si posee lápiz (Stylus)
Emulación	Emulaciones soportadas por la tableta, interpretadas de la siguiente forma: MM (Summagraphics MM), MG (Summagraphics Microgrid), BP (Summagraphics Bit Pad Series), C (Calcomp 2000), K (Kurta IS/ONE) y G (GTCO Digipad)
Diodos	Indica si posee diodos LED de encendido (E) y de proximidad (P)

Tableta	Tamaño	Exactitud	Resol.	Cursor	Lápiz	Emulaciones	Diodo	Distribuidor
Potworld PT-3030	12 x 12	0.025	1.016	4	Si	MM	No	2I Ingeniería Informática
Genius HiSketch1212	12 x 12	0.01	1.000	4	Si	MM,MG,C,G	E,P	Power Computing
Wintime KD-5000	18 x 12	0.025	1.016	6	Si	MM	No	Power Computing
Calcomp Drawingslate A5	9 x 6	0.001	1.270	No	Si	N/D	E	Calcomp España
Calcomp Drawingslate A4	12 x 12	0.001	1.270	4	No	N/D	E	Calcomp España
Genius HiSketch 1812	18 x 12	0.01	1.000	4	Si	MM,MG,BP,C,G	E,P	Power Computing
Graphtec KD3220	8.7x11.7	0.02	1.016	4	Si	MM,G	E,P	Cioce
Graphtec KD3320	12 x 12	0.02	1.016	4	Si	MM,G	E,P	Cioce
Numonics Grid A4	12 x 12	0.01	5.000	4	No	MM,MG,BP,C,K	E	Roland
SummaSketch III	12 x 12	0.007	2.540	4	No	MM	No	Soft-Sell
Potworld PT-3050	12 x 18	0.025	1.016	4	Si	MM	No	2I Ingeniería Informática
Numonics Graphics A4	12 x 12	0.01	5.000	4	Si	MM,MG,BP,C,K	E	Roland
Calcomp Drawingboard	12 x 12	0.001	2.540	4	No	N/D	E	Calcomp España
OCE G6421	12 x 12	0.002	508	4	Si	MM,MP	E	OCE Graphics
Calcomp Drawingslate A3	12 x 18	0.001	1.270	4	No	N/D	E	Calcomp España
Graphtec KD 4320	11 x 15	0.02	1.016	4	Si	MM,G	E,P	Cioce
Graphtec KD 4620	12 x 18	0.02	1.016	4	Si	MM,G	E,P	Cioce
Graphtec KD 3820	15 x 15	0.02	1.016	4	Si	MM,G	E,P	Cioce
SummaSketch III Pro	12 x 18	0.007	2.032	4	No	MM	No	Soft-Sell
OCE G6461	12 x 12	0.001	1.000	4	Si	MM,BP	E,P	OCE Graphics
Numonics Graphics A3	12 x 18	0.01	5.000	4	Si	MM,MG,BP,C,K	E	Roland
Calcomp Drawingboard	12 x 18	0.001	2.540	4	No	N/D	E	Calcomp España
OCE G6422	12 x 18	0.001	1.000	4, 16	Si	MM,BP	E,P	OCE Graphics
OCE G6452	15 x 15	0.001	1.000	4	Si	MM,BP	E,P	OCE Graphics
OCE G6462	12 x 18	0.001	1.000	4	Si	MM,BP	E,P	OCE Graphics
Numonics Accugrid TA4	12 x 12	0.01	2.000	4	No	MM,MG,BP,C,G	E	Roland
Numonics Accugrid TA3	12 x 17	0.01	2.000	4	No	MM,MG,BP,C,G	E	Roland
Summagraphics IV 1824	18 x 24	0.1	2.540	4	No	MM,MG,C,G	No	Soft-Sell
Calcomp Drawingflex A1	28 x 36	N/D	2.540	4	No	N/D	E	Calcomp España
Calco. Drawingboard A2	18 x 24	0.001	2.540	4	No	N/D	E	Calcomp España

Tableta	Tamaño	Exactitud	Resol.	Cursor	Lapiz	Emulaciones	Diodo	Distribuidor
OCE G6464	18 x 25	0.001	1.000	4	Si	MM,BP	E,P	OCE Graphics
Numonics Accugrid A2	20 x 24	0.01	2.000	8, 16	No	MM,MG,BP,C,G	No	Roland
Numonics Accugrid XNT	36 x 48	0.01	2.000	4	No	MM,MG,BP,C,G	No	Roland
Calcomp Drawingflex A0	36 x 48	N/D	2.540	4	No	N/D	E	Calcomp España
Numonics Accugrid TA2	20 x 24	0.005	2.000	4	No	MM,MG,BP,C,G	E	Roland
Numonics Accugrid A1	24 x 36	0.01	2.000	8, 16	No	MM,MG,BP,C,G	No	Roland
Summagrid IV 2436	24 x 36	0.1	2.540	4	No	MM,MG,C,G	No	Soft-Sell
Microgrid III 1724	17 x 24	0.005	2.000	4	No	MM,MG,C,G	No	Soft-Sell
Microgrid III 2020	20 x 20	0.005	2.000	4	No	MM,MG,C,G	No	Soft-Sell
Summagrid IV 4460	44 x 60	0.1	2540	4	No	MM,MG,C,G	No	Soft-Sell
Numonics Accugrid A00	44 x 60	0.01	2.000	8, 16	No	MM,MG,BP,C,G	No	Roland
Calco. Drawingboard A1	24 x 36	0.008	2.540	16	N/D	N/D	E	Calcomp España
Calco. Drawingboard A0	36 x 48	0.008	2.540	16	N/D	N/D	E	Calcomp España
Numonics Accugrid TA0	36 x 48	0.005	2.000	4	No	MM,MG,BP,C,G	E	Roland
Microgrid III 2436	24 x 36	0.005	2.000	4	No	MM,MG,C,G	No	Soft-Sell
CalcompDrawingboard III	44 x 60	0.008	2.540	N/D	N/D	N/D	E	Calcomp España
Microgrid III 3648	34 x 48	0.005	2.000	4	No	MM,MG,C,G	No	Soft-Sell
Microgrid III 4460	44 x 60	0.005	2.000	4	No	MM,MG,C,G	No	Soft-Sell

4.4 Scanner.

A la hora de trabajar en procesos que requieran la utilización de imágenes, se hace indispensable disponer de todos los medios posibles para la captura de éstas. Es por este motivo por lo que se hace fundamental algún sistema que convierta las imágenes contenidas en fotografía o papel en cualquier formato compatible con los programas de imágenes que utilizamos. Esta es, sencillamente, la misión que debe cumplir un *scanner*.

4.4.1. Dossier de Scanner.

Los modelos disponibles en el mercado dependen, en gran medida, de las características que necesitemos. Para aclarar un poco este tema, comentaremos cómo se clasifican los *scanner* según sus características.

CARACTERÍSTICAS	DESCRIPCIÓN
Modelo.	Nombre del fabricante y modelo de scanner.
Tipo.	Indica si se trata de scanner de mano (M), sobremesa (S) o de planos (P)
Grises o colores.	Representa el número de grises o colores que puede gestionar el scanner. Si es un scanner monocromo, el número representa el número de tonos de grises que puede reconocer. Si es de color, representa el número de colores.
Resolución.	Resolución con la que puede digitalizar el scanner.
Ancho.	Representa el ancho en mm. De la ventana a escanear.
Altura.	Altura en mm. De la ventana a escanear. Junto con el ancho, indica el tamaño máximo del documento.
Puerto.	Tipo de interfaz que utiliza el scanner para conectarse al ordenador.
Diapositiva.	Indica si el scanner está preparado para aceptar un módulo especial para aceptar diapositivas.
Programa retoque.	Nombre y versión del programa para el tratamiento de imágenes.
Distribuidor.	Nombre del distribuidor nacional.

A continuación, se incluye un *dossier* con los principales modelos de *scanner* a color que existen actualmente en el mercado.

Modelo	Tipo	Num. Color	Reso-lución	Anch	Altura	Puerto	Diap	Programa retoque	Distribuidor
Vobis ColorScan18	M	65.536	N/D	105	N/D	Propio	No	Photo Plus	Vobis
Mustek Twain Scan Color	M	16.8 M	400	105	N/D	Propio	No	¡Foto Plus	Worlwide Sales
Mustek PrinScan Color	M	256144	400	105	N/D	Paral.	No	¡Foto Plus	Worlwide Sales
Suvil Colorbrush 12 bits	M	4096	200	105	558	Propio	No	¡Foto Plus	ACG Zaragoza
Suvil Colorbrush 18 bits	M	262164	200	105	558	Propio	No	Color-it	ACG Zaragoza
Mustek Color Artist Pro	M	16.8 M	800	105	N/D	Propio	No	Pict. Publisher	Worlwide Sales
Primax Color Mobile	M	16.8 M	200	105	300	Propio	No	Finish. Touch	SIP
Vobis ColorScan 24	M	16.8 M	N/D	105	N/D	Propio	No	¡Photo Plus	Vobis
Suvil Colorbrush 18 bits	M	262144	200	105	558	SCSI	No	Color-it	ACG Zaragoza
Suvil Colorbrush 24 bits	M	16.8 M	400	105	558	Propio	No	PhotoMagic	ACG Zaragoza
Logitech Scanman color	M	16.8 M	400	105	550	Propio	No	Photo Touch	Logitech
Suvil Colorbrush Laptop	M	262144	200	105	558	Paral.	No	¡Foto Plus	ACG Zaragoza
Suvil Colorbrush 24 bits	M	16.8 M	800	105	558	Propio	No	PhotoMagic	ACG Zaragoza
Sicos Colour Page	S	16.8 M	1200	N/D	N/D	Propio	No	¡Foto Plus	Crisfer
Plustek Scanplus Color600	S	16.8 M	600	216	355	Propio	No	¡Foto Plus	Worlwide Sales
Tamarack SCA6000C	S	16.8 M	1200	210	297	SCSI	Si	PhotoStyler	Data Logic
Mustek Paragon 600	S	16.8 M	600	210	355	SCSI	Si	¡Foto Plus	Worlwide Plus

Modelo	Tipo	Num. Color	Reso-lución	Anch	Altura	Puerto	Diap	Programa retoque	Distribuidor
Apple Color OneScanner	S	16.8 M	1200	210	350	SCSI	No	Ofoto Color	Apple
Epson GT-6500	S	16.8 M	600	216	297	Ser/Par	Si	No	Epson Ibérica
Canon IX-4015	S	16.8 M	N/D	216	296	SCSI	No	No	Canon España
Umax Vista	S	16.8 M	1800	216	297	SCSI	Si	PhotoShop	Disvent
Agfa Gevaer StudioScan II	S	65536	2400	216	355	SCSI	Si	PhotoShop	Agfa Gevaert
Mustek Paragon 1200	S	16.8 M	1200	210	355	SCSI	Si	¡Foto Plus	Worlwide Sales
Fujitsu AV 680	S	16.8 M	800	210	355	SCSI-2	Si	Image Pals	Fujitsu España
Epson GT-6500	S	16.8 M	600	216	297	SCSI	Si	No	Epson Ibérica
Microtek ScanMaker II Pc	S	16.8 M	1200	216	356	SCSI	Si	Image Star	Mensana
HP ScanJet II cx	S	16.8 M	400	216	279	SCSI	Si	Photo Styler	Hewlett-Packard
Microtek Scanmaker II SP	S	16.8 M	1200	216	356	SCSI	Si	Photoshop	Mensana
Epson GT-9000	S	16.8 M	1200	216	297	Paral.	Si	No	Epson Ibérica
Epson GT-9000	S	16.8 M	1200	216	297	SCSI	Si	No	Epson Ibérica
Howtek Color Scanner	S	16.8 M	600	216	356	Propio	No	Varias opcion.	Cioce
Microtek Scanmaker II XE	S	16.8 M	1200	216	356	SCSI	Si	PhotoShop	Mensana
AVR 880/CLX	S	16.8 M	1600	216	356	SCSI-2	Si	Microg. Draw	SCS
Microtek Scanmaker II HR	S	16.8 M	2400	216	330	SCSI	Si	Photoshop	Mensana
Dextra 2400	S	16.8 M	1200	210	355	SCSI	Si	¡Foto Plus	Worlwide Sales
UmaxPS-2400 PowerLook	S	16.8 M	2400	212	297	SCSI	Si	PhotoShop LE	Disvent
Agfa Gevaert Arcus II	S	16.8 M	2400	210	355	SCSI	Si	Photoshop	Agfa Gevaert
Microtek Scanmaker III	S	36.8 M.	2400	216	356	SCSI	Si	Photoshop	Mensana
Agfa Gevaert Vision 35	S	16.8 M	3175	.40	40	SCSI	Si	Agfa Fotolook	Agfa Gevaert
Howtek Scanmaster 3+	S	16.8 M	1200	297	432	GPIB	Si	varias opcion.	Cioce
Agfa Gevaert Horizon Plus	S	16.8 M.	1800	297	420	SCSI	Si	Agfa Fotolook	Agfa Gevaert
Agfa Gevaert SelectScan	S	16.8 M.	4000	210	297	SCSI	Si	Agfa Fotolook	Agfa Gevaert

4.5. Dispositivos de almacenamiento masivo.

Para el almacenamiento de una animación es necesario tener en cuenta el dispositivo que vamos a utilizar, ya que este tipo de información ocupa una gran cantidad de espacio y no será valido cualquiera. Para hacernos una idea, supongamos una imagen con una resolución de 320 x 200 con 8 bits por pixel. Realizando una simple multiplicación, obtendremos una cantidad considerable de datos. Si tenemos en cuenta que, para conseguir el efecto de movimiento, se necesitan 25 imágenes por segundo, nos daremos cuenta de la enorme cantidad de información que se produce en este tipo de trabajos.

Por todo lo anterior, debemos analizar diferentes tipos de dispositivos de almacenamiento para una posterior elección del más adecuado. Será necesario por un lado, un disco duro rápido, y por otro, un sistema de almacenamiento secundario que dé la posibilidad de utilizar las animaciones ya creadas sin necesidad de emplear técnicas de compresión.

Por tanto, haremos dos tipos de estudios: uno de discos duros y otro de sistemas de grabación de discos CD-ROM.

4.5.1 Discos duros como sistemas de almacenamiento.

Actualmente en el mercado existen dispositivos capaces de almacenar hasta 6 ó 7 gigabytes. Teniendo en cuenta que esto se puede ocupar en sus totalidad por varias animaciones, no podremos hacer uso de él como dispositivo de almacenamiento final. Tendremos, por tanto, que recurrir al CD-ROM y sacar partido a sus grandes prestaciones.

Aún así, será necesario utilizar un disco duro como soporte para guardar la animación que estemos utilizando en un determinado momento. Debido a la utilidad que le vamos a dar, necesitaremos un disco de gran capacidad que nos de la suficiente velocidad como para poder reproducir animaciones con el menor tiempo de lectura posible.

4.5.1.1. Dossier de discos duros.

A continuación se muestra una lista con gran parte de los discos que existen en el mercado. Debido a la gran variedad de éstos, solo se detallarán los que poseen más de 500 megabytes de espacio. Las características que se especifican para la elección del disco, son:

CARACTERÍSTICAS	DESCRIPCIÓN
Disco	Fabricante y modelo del disco.
Megabytes	Capacidad del disco una vez formateado.
Interfaz	Interfaz utilizado por el disco para conectarse al ordenador.
TMB	Tiempo medio de búsqueda en milisegundos. Corresponde al tiempo medio que tarda el disco en encontrar un sector.
Velocidad	Velocidad de transferencia en megabytes por segundo.
Altura	Altura del disco medida en centímetros.
Tamaño	Anchura del disco medida en pulgadas.
Distribuidor	Nombre del distribuidor.

Disco Duro	MB	Interfaz	TMB	Velocid.	Altura	Tamaño	Distribuidor
IBM 527AT	527	IDE	8.7	30.1	2.54	4.00	IBM
IBM 540MB Fast SCSI-2	540	SCSI-2	8.7	30.1	2.54	4.00	IBM
Conner CP-30544	546	IDE	10.0	6.0	2.54	3.50	Siscomp
Conner CP-30540	546	SCSI-2	10.0	10.0	2.54	3.50	Siscomp
Quantum LPS-540	540	SCSI-2	12.0	11.5	2.0	3.50	Computer 2000
Quantum LPS-540	540	IDE	12.0	11.5	2.0	3.50	Computer 2000
Western Digital AC-540	540	IDE	13.0	5.7	2.54	3.50	Diode
Seagate ST-3550A	550	IDE	12.0	5.5	2.50	3.50	Sintronic
Seagate ST-3550N	550	SCSI-2	12.0	5.0	3.50	3.50	Sintronic
Hitachi DK 515-78	673	ESDI	16.0	2.4	8.25	5.25	Ataio Interacción
Hitachi DK 515C-78	673	SCSI	16.0	2.4	8.25	5.25	Ataio Interacción
Hitachi DK 315C-10	1000	SCSI-2	10.4	5.0	4.13	3.50	Ataio Interacción
Toshiba MK-538 FB	1023	SCSI-2	12.0	10.0	4.13	3.50	Cioce
Hewlett-Packard C-2247	1024	SCSI-2	9.0	20.0	4.0	3.50	Computer 2000
IBM 1GB Fast SCSI-2	1024	SCSI-2	8.6	38.4	2.54	4.00	IBM
Micropolis MD1050 ext.	1050	SCSI-2	10.0	10.0	1.27	3.50	Diode
Conner CFP-1060S	1062	SCSI-2	9.0	10.0	2.54	3.50	Siscomp
Fujitsu M29287-SA	1080	SCSI-2	10.0	10.0	2.50	3.50	Fujitsu España
Fujitsu M29287-QA	1080	W.-SCSI	10.0	20.0	2.50	3.50	Fujitsu España
Quantum Empire-1080	1080	SCSI-2	9.0	20.0	2.0	3.50	Computer 2000
Hitachi DK 315C-11	1100	SCSI-2	10.4	5.0	4.13	3.50	Ataio Interacción
Seagate ST-11200N	1248	SCSI-2	10.5	10.0	4.10	3.50	Sintronic
Hitachi DK 516-15	1321	ESDI	14.0	2.7	8.25	5.25	Ataio Interacción
Hitachi DK 516C-16	1342	SCSI	13.5	5.0	8.25	5.25	Ataio Interacción
Conner CP-31370	1370	SCSI-2	10.0	10.0	4.13	3.50	Siscomp
Hitachi DK 315C-14	1400	SCSI-2	10.4	5.0	4.13	3.50	Ataio Interacción

Micropolis MD1760 ext.	1760	SCSI-2	10.0	N/D	1.27	3.50	Diode
Quantum PQ-1800	1800	SCSI-2	10.0	10.0	4.0	3.50	Computer 2000
Hewlett-Packard C-2490A	2048	SCSI-2	9.0	20.0	4.0	3.50	Computer 2000
Fujitsu M2654-SA	2055	SCSI-2	12.0	10.0	8.20	5.25	Fujitsu España
Micropolis RAID T2060	2060	SCSI-2	14.5	5.0	13.9	5.25	Diode
Fujitsu M2903-SA	2100	SCSI-2	9.0	10.0	4.10	3.50	Fujitsu España
Micropolis MD-2100 ext.	2100	SCSI-2	11.5	N/D	1.27	3.50	Diode
Fujitsu M2903QA	2100	W.-SCSI	9.0	20.0	4.10	3.50	Fujitsu España
Micropolis RAID T2100	2100	SCSI-2	12.0	10.0	13.9	5.25	Diode
Disco Duro	MB	Interfaz	TMB	Velocid.	Altura	Tamaño	Distribuidor
Conner CFP-2120S	2124	SCSI-2	10.0	10.0	4.13	3.50	Siscomp
Micropolis MIC-1926	2158	SCSI-2	13.0	10.0	8.26	5.25	Diode
Seagate ST-12400N	2537	SCSI-2	9.0	10.0	4.10	3.50	Sintronic
Seagate ST-12550N	2572	SCSI-2	8.0	10.0	4.10	3.50	Sintronic
Micropolis RAID 2680	2680	SCSI-2	14.5	5.0	13.9	5.25	Diode
Hitachi DK 577C-37	2870	SCSI-2	12.0	5.0	8.25	5.25	Ataio Interacción
Fujitsu M2909-SA	3000	SCSI-2	9.0	10.0	4.10	3.50	Fujitsu España
Fujitsu M2909-QA	3000	W.-SCSI	9.0	20.0	4.10	3.50	Fujitsu España
Micropolis MIC-1936	3022	SCSI-2	13.0	10.0	8.26	5.25	Diode
Micropolis MD3020 ext.	3022	SCSI-2	12.0	N/D	1.27	3.50	Diode
Micropolis RAID T3500	3500	SCSI	14.0	10.0	13.9	5.25	Diode
Micropolis RAID T4200	4200	SCSI-2	12.0	10.0	13.9	5.25	Diode
Micropolis RAID T6040	6040	SCSI-2	12.0	10.0	13.9	5.25	Diode

4.5.2 Unidades CD grabables.

Las unidades CD grabables son una solución cuando se requiere almacenar gran cantidad de información. Es necesario resaltar que solo será valido este recurso cuando se generen muchas animaciones, ya que infrautilizar este tipo de dispositivos puede suponer un gran coste económico. Por el contrario, si su utilización es frecuente, este equipo puede llegar a convertirse en el dispositivo de almacenamiento ideal, ya que no se necesita comprimir la información.

Antes de comentar la características de los principales equipos disponibles, es necesario conocer los principales estándares y conceptos que se utilizan en el mundo de los CD's.

• CD-DA: (*Compact Disc-Digital Audio*). Formato de los discos CD utilizados para la grabación de música. Las especificaciones de este formato se conocen como "Libro Rojo".

• CD-I: (*Compact Disc-Interactive*). Formato de discos CD desarrollado por Philips para almacenar datos, audio y vídeo que se puede reproducir en un televisor. La especificación de este formato se conoce como "Libro Verde".

• CD-R: (*Compact Disc-Recordable*). Especificación que define como han de trabajar las unidades que permiten grabar discos compactos. La especificación de este formato está recopilada en la segunda parte de lo que se conoce como "Libro Naranja".

• CD-ROM: (*Compact Disc- Read Only Memory*). Formato de los discos CD denominados estrictamente CD-ROM Modo 1. El formato del CD-ROM es un superconjunto del CD-DA y está diseñado para discos que pueden almacenar tanto audio como datos (texto y gráficos). La especificación de este formato se conoce como "Libro amarillo".

• CD-ROM XA: (*CD-ROM eXtended Architecture*). Es una variación del formato CD-ROM Modo 1 y también se le conoce como formato CD-ROM Modo 2. Poseen pistas entrelazadas de audio y vídeo con datos para crear efectos multimedia que simule vídeo o animación.

• HFS: (*Hierarchical File System*). Estructura de ficheros utilizada en los discos del sistema Apple Macintosh. Es posible crear discos CD-ROM y CD-ROM XA que utilicen la estructura de ficheros HFS.

• ISO 9660: Estructura de ficheros utilizada de forma mayoritaria en los discos CD-ROM y CD-ROM XA. Es independiente de la plataforma utilizada y se reconoce en los sistemas operativos DOS, Macintosh y Unix.

• MO: (*Magneto Optical*): Tecnología para unidades y discos magneto-ópticos que son regrabables y pueden almacenar en un tamaño de 3 ½'' hasta 256 Mb. de información. La especificación de este formato se recoge en la primera parte del denominado "Libro Naranja".

• Monosesión: Un disco monosesión es un disco que se ha grabado durante una única sesión. Los lectores CD-ROM monosesión solo pueden leer la primera sesión de un disco multisesión.

• Multisesión: Un disco multisesión es un disco cuyo contenido se ha impreso en el disco en dos o más sesiones de grabación. Un lector CD-ROM multisesión puede leer tanto discos monosesión como discos multisesión.

• *Photo CD*: Tecnología creada por Kodak cuyo objetivo es crear un disco CD-ROM cuyo contenido sean las fotografías contenidas en un carrete de fotos. Para poder leer estos discos se necesita una unidad CD-ROM XA, pues el formato *Photo CD* es una variante del formato CD-ROM XA. En un disco con este formato pueden almacenarse hasta 100 fotografías. En este formato existen también CD-ROM monosesión y multisesión, y para leer ambos se necesita un lector CD-ROM XA multisesión.

• Pista: Unidad mínima que se puede grabar en un disco CD grabable con una unidad CD-R. En un disco puede haber un máximo de 99 pistas, sumando todas las pistas de cada sesión.

• Sector: La cantidad de información que puede almacenar un disco CD se mide en minutos y segundos. Cada segundo contiene 75 sectores de 2K cada uno.

• Sesión: Segmento grabado del disco compacto que puede contener una o más pistas de cualquier tipo (datos o audio) y que ha de estar cerrado.

4.5.2.1 Cuadro resumen de los principales equipos CD-R.

CARACTERÍSTICAS	KODAK PCD 200	PHILIPS CDD-521	PINNACLE RCD-202	SONY CDW-900E
CD-Audio	Escribe	Escribe	Lee/ Escribe	Escribe
CD-ROM	Lee/Escribe	Lee/Escribe	Lee/Escribe	Escribe
CD-ROM XA	Lee/Escribe	Lee/Escribe	Lee	Escribe
CD-I	Escribe	Escribe	No	Escribe
Multisesión	Lee/Escribe[1]	Lee/Escribe[2]	Lee/Escribe[1]	No
HFS	No[3]	No[3]	No[3]	Escribe
Unix	No	No	No	Escribe
Botón volumen sonido	No	No	Si	No
Conector auriculares	No	No	Si	No
Grabación a doble velocidad	Si	Si	No	Si
Reconoce codigos Infoguard	Si	No	No	No
Alto x Ancho x Largo (mm)	110x420x334	110x420x334	55x245x265	95x424x412
Precio unidad	993.600	943.000	1.069.500	1.262.700
Precio Kit[4]	1.428.300	1.371.145	1.069.500	1.507.650

[1] Con el software incluido no se puede escribir multisesión, sino multivolumen.

[2] Para poder leer los discos multisesión se necesita el controlador suministrado para la unidad.

[3] Se puede leer y escribir en el formato HFS con el software adecuado para Macintosh, pero no desde el programa DOS/WIN.

[4] Se incluye unidad, programa, tarjeta y cable.

4.6. Controladores de grabación.

Una vez que tenemos terminada la animación, debemos volcarla a vídeo. Para ello podemos trabajar de dos maneras diferentes; Una de ellas consiste en poner la tarjeta digitalizadora como dispositivo de salida, con lo cual volcaremos la animación a vídeo. Pero esta forma se plantea un problema. Si la animación es muy grande y no cabe en memoria (cosa que es muy probable), necesitaremos utilizar el disco duro como memoria virtual. Esto supone que el volcado se realizará a la velocidad que el disco duro permita, que no se aproximará nunca a tiempo real. Como consecuencia de esto, la animación se verá con saltos.

La otra solución consiste en utilizar un controlador de VTR, que nos permitirá grabar una animación cuadro a cuadro. Su misión es la de tener un control exacto de cada cuadro de vídeo, lo que permite una grabación exacta de la señal de vídeo en cualquier momento.

Para almacenar una animación cuadro a cuadro, se graba primero en la cinta en tiempo real un código de tiempo. Cuando el ordenador ha generado una imagen, le dice al controlador que hay que grabarla, y éste, a su vez, le ordena al VTR que grabe justo donde le ha indicado el controlador. Cuando se genera una nueva imagen, se repite el proceso y así, hasta concluir el volcado de la animación entera.

La grabación mediante este segundo método tiene el inconveniente de que es perjudicial para los magnetoscopios, ya que al trabajar con elementos mecánicos, éstos pueden desregularse debido al rebobinado continuo de las cintas. También existe el inconveniente del desgaste de las cabezas del vídeo.

Aún con estos inconvenientes, es el método más efectivo para la grabación de animaciones y así lo demuestran los equipos de altas prestaciones, que utilizan este tipo de mecanismos para el volcado.

4.6.1 Controladores disponibles.

COMPAÑIA	MODELO	OBSERVACIONES
Diaquest	DQ-422	Para PC. Control serie.
	DQ-50	Control paralelo.
VideoMedia	V-LAN	Control de 31 VTR's
BCD Assoc.	BCD-5000	Control de 2 VTR's
Adv. Dig. Im.	Mac-Vac	Solo para Macintosh
Lion Lamb	Minivas-2	Generación interna de codigo de tiempo
	ProVas	Incluye codificador PAL
Chase Tech.	SOFT-VTR	Compatible con PC's

5. Targa+.

Como tarjeta digitalizadora, ha sido elegida la Targa 32+, por ser ésta el standard en digitalizadoras de vídeo. Esta tarjeta nos permitirá capturar cuadros de vídeo, así como volcar secuencias de imágenes generadas en el PC. De forma genérica, las características de esta tarjeta son:

- Soporte de formato PAL.
- Más de 30 tipos de resoluciones diferentes.
- Posibilidad de efectuar un *Chroma Key*.
- Posibilidad de *Overlay*.
- *Genlock* avanzado.

Todas estas características serán expuestas con más detalles más adelante.

5.1 Requerimientos.

En primer lugar, es necesario analizar si se cumplen los requisitos mínimos respecto a *hardware* para el correcto funcionamiento de la tarjeta digitalizadora. Cabe decir que estos son bastante pocos, por lo que actualmente la mayoría de los PC's cumplen con creces estas necesidades básicas.

Los requerimientos mínimos son:

- Un PC con 640 K de memoria RAM.
- Un *slot* ISA para la inserción de la tarjeta.

- Espacio suficiente en disco duro para la correcta instalación del *software*.
- Un monitor para usarse como dispositivo de salida de la tarjeta, aunque dependiendo del trabajo que se realice, esta salida podrá ser conectada a una mesa de mezclas.

Los requerimientos anteriormente exigidos son superados con creces. Para imprimirle mayor velocidad al sistema, esta tarjeta ha sido instalada sobre un equipo con las siguientes características:

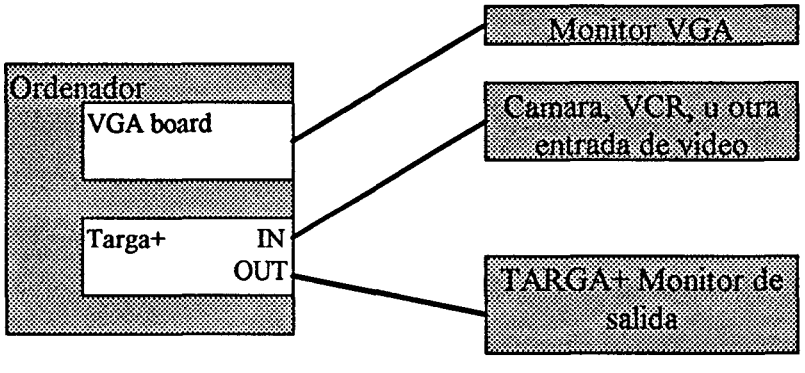
- PC 486-DX33
- 8 Mb-RAM en módulos SIMM de 1 Mb.
- FD 3 ½ 1.44 Mb.
- FD 5 ¼ 1.2 Mb.
- HD 250 Mb. Conner 12 msg.
- Tarjeta de vídeo Trident.
- Unidad lectora CD-ROM Mitsumi 150 Kb/sg.
- Tableta digitalizadora Genius 12x12
- 8 Slot tipo ISA 16 bits.

5.2 Posibles configuraciones.

Antes de comenzar el proceso de instalación de la Targa+, es necesario conocer las posibles configuraciones para elegir la que mejor se adapte a nuestras necesidades y a los equipos disponibles. Todas estas configuraciones hacen referencia a los equipos que podemos conectar, así como a la forma de hacerlo. A continuación se exponen las posibles configuraciones, así como un pequeño análisis de cada una de ellas.

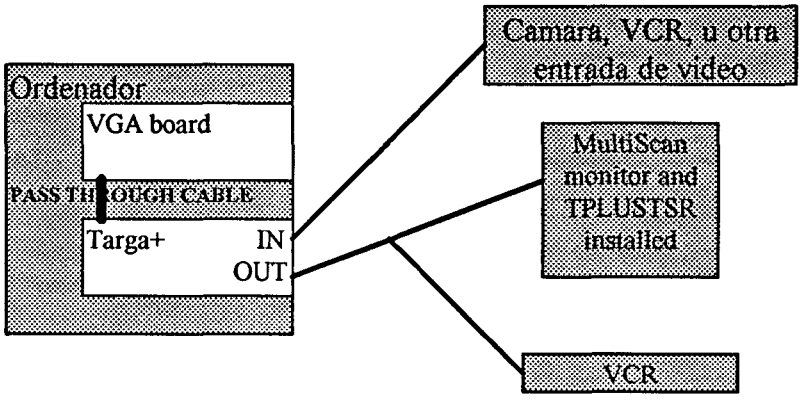
Configuración 1. *(Two-monitors solution).*

Esta primera elección, se basa en conectar el monitor del ordenador a su propia salida de vídeo, es decir, a la salida VGA de la controladora.



Las conexiones realizamos en la targa+, son de la siguiente forma: Por un lado, conectamos una cámara, VTR, o cualquier otro dispositivo generador de vídeo a la entrada de la tarjeta, con lo cual podremos digitalizar imágenes. Y por otro lado, conectamos un monitor a la salida de la tarjeta para poder visualizar el contenido de su memoria.

Configuración 2. *(Single-monitor solution).*

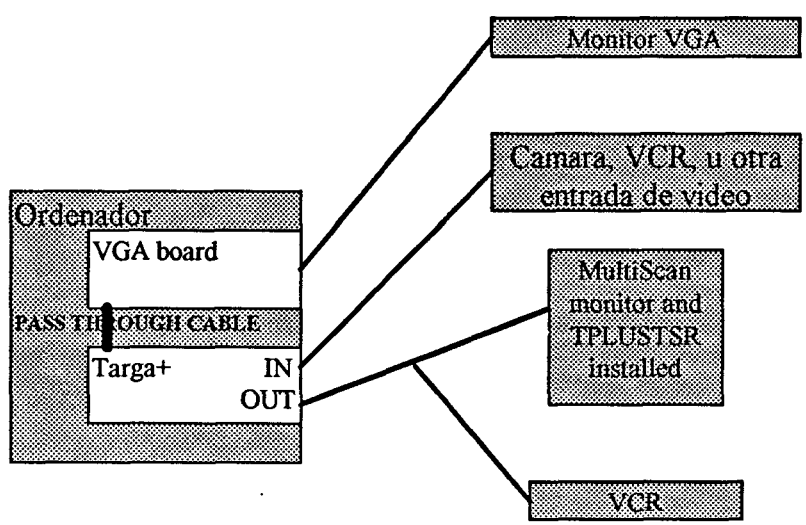


Como se puede comprobar, esta segunda configuración no utiliza el monitor VGA típico del ordenador. Esto se debe a que en esta opción usaremos la salida de vídeo de la Targa+ como dispositivo de visualización.

Para poder llevar a cabo este montaje, es necesario disponer de algún dispositivo como fuente de vídeo y, como salida, un monitor para la visualización de la memoria de la tarjeta y un vídeo o controlador de vídeo para la grabación de las secuencias de animaciones.

Esta opción la utilizaremos cuando se necesite volcar una animación a un magnetoscopio. Para ello, deberemos cargar el *driver* de targa del programa que estemos utilizando y a continuación visualizar la animación siguiendo el proceso normal del programa. Esto cobra gran importancia, ya que en programas como el *3D Studio*, se puede utilizar la tarjeta como dispositivo de *render* y visualización, con lo cual ganaremos en velocidad a la hora de hacer el *render* y en comodidad, ya que no tendremos que realizar ningún proceso para la grabación a vídeo de la animación.

Configuración 3.

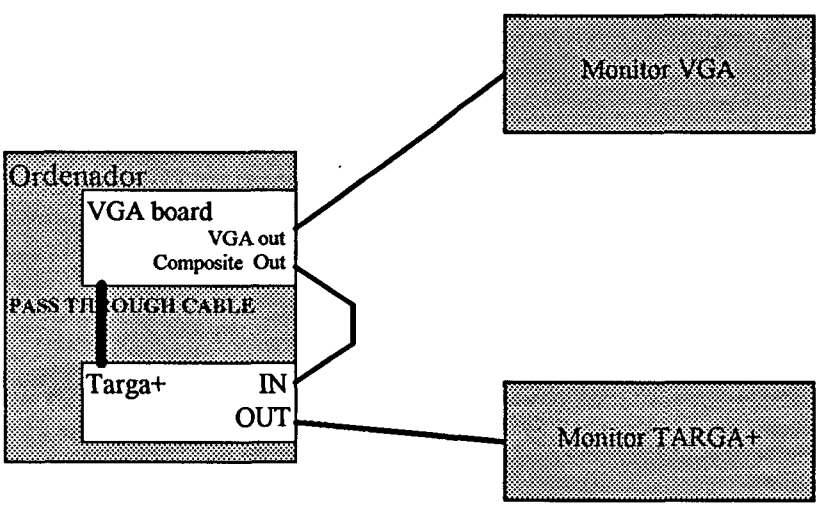


Esta es otra de las posibles configuraciones que se ofertan con la tarjeta. Analizándola junto con las anteriores, se puede observar que es una mezcla entre las

configuraciones 1 y 2. De esta manera se permite usar la Targa+ como dispositivo de salida y a la vez podremos usar el monitor VGA del ordenador para la realización de otras tareas. Esto puede resultar una gran ventaja y para comprobarlo estudiemos un ejemplo:

Supongamos que trabajamos con el *software* de animación *3D Studio*. Con el monitor VGA realizaremos toda la animación siguiendo los pasos normales de construcción de la misma y, una vez realizada, utilizaremos la targa+ para realizar el *render* y visualizar los resultados. Estos últimos serán visualizados en el monitor que tengamos conectado a la salida de la targa+ y no en el monitor VGA del ordenador.

Configuración 4.



Esta es la última configuración que analizaremos. Como se puede comprobar en la ilustración anterior, ésta solo será valida cuando poseamos una tarjeta de vídeo con dos tipos de salida: la salida VGA normal del ordenador, y otra salida de vídeo compuesto. De esta forma, la salida compuesta de la tarjeta, podrá ser conectada a la entrada de la targa+. Esta conexión debe ser realizada desde el exterior, ya que este tipo

de tarjetas no permite la unión de ambas desde el interior del ordenador como se realiza con el *pass-through cable*.

5.3 Cables y conectores.

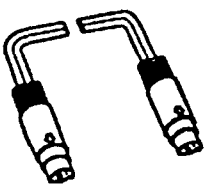
Para poder llevar a cabo todas las configuraciones anteriormente analizadas, es obligatorio disponer de todos los materiales necesarios. Obviamente entre ellos están los cables. Para saber cuales son los que vamos a necesitar, es necesario primero conocer cuales son los tipos de conectores que tenemos y cual es su significado.

La tarjeta targa+ dispone de dos conectores de 9 pines tanto para la entrada como para la salida. El significado de cada pin, se muestra en la tabla siguiente:

ENTRADA		SALIDA	
N° de Pin	**Señal**	**N° de Pin**	**Señal**
1	Ground	1	Ground
2	Ground	2	Ground
3	Red Input	3	Red Output
4	Green Input	4	Green Output
5	Blue Input	5	Blue Output
6	Composite Video Input	6	Composite Video Input
7	Composite Sync Input	7	Composite Sync Input
8	Y Input (S-Video)	8	Y/H Sync Output
9	C Input (S- Video)	9	C/V Sync Output

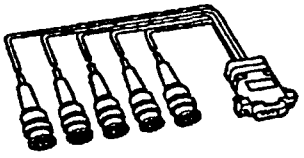
Como se podrá comprobar, disponemos de entradas y salidas en diferentes formatos: RGB, Video Compuesto y S-Video. Por tanto, los cables que empleemos deberán tener diferentes salidas para poder seleccionar cada uno de los formatos posibles.

Aunque la targa+ se distribuye con un único cable, se puede disponer de 10 tipos de cables posibles. A continuación, se muestra una lista de ellos con sus características principales:

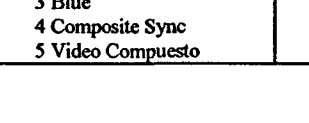


PART#	DESCRIPTION	PINOUTS & SIGNALS	PRODUCT (S)
CA-1	De 4 Pin Mini-DIN a 4 Pines Mini-DIN	4 Pin Mini-DIN End 1 Gnd 2 "Y" Luminancia 3 "C" Chroma 4 Gnd	Vid I/O Box, Cualquier dispositivo S-VHS compatible
		4 Pin Mini-DIN End 1 Gnd 2 "Y" Luminancia 3 "C" Chroma 4 Gnd	Vid I/O Box, Cualquier dispositivo S-VHS compatible

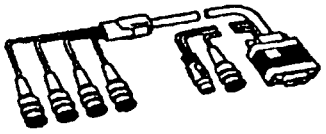
PART#	DESCRIPTION	PINOUTS & SIGNALS	PRODUCT (S)
CA-2	9 Pines a 4 BNC	9 pines 1 Gnd 2 Gnd 3 Red 4 Green 5 Blue 6 7 Composite Sync 8 9	Targa+
		4 BNC 1 Red 2 Green 3 Blue 4 Composite Sync	Cualquier dispositivo RGB con soporte de sincronismo.



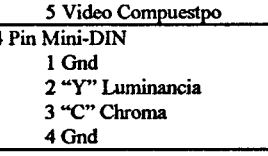
PART#	DESCRIPTION	PINOUTS & SIGNALS	PRODUCT (S)
CA-3	9 Pines a 5 BNC	9 pines 1 Gnd 2 Gnd 3 Red 4 Green 5 Blue 6 Video Compuesto 7 Composite Sync 8 9	Targa+
		4 BNC 1 Red 2 Green 3 Blue 4 Composite Sync 5 Video Compuesto	Cualquier dispositivo RGB con soporte de sincronismo.



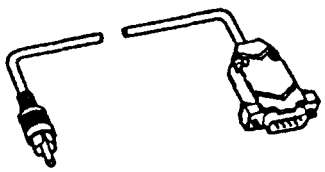
PART#	DESCRIPTION	PINOUTS & SIGNALS	PRODUCT (S)
CA-4	9 Pines a 9 Pines	9 pines 1 Gnd 2 Gnd 3 Red 4 Green 5 Blue 6 Video Compuesto 7 Composite Sync 8 "Y" o H-Sync 9 "C" o V-Sync	ATVista, NuVista, TARGA y TARGA+
		9 pines 1 Gnd 2 Gnd 3 Red 4 Green 5 Blue 6 Video Compuesto 7 Composite Sync 8 "Y" o H-Sync 9 "C" o V-Sync	Monitor Sony 1302, monitores analogicos RGB, dispositivos con salida compuesta.



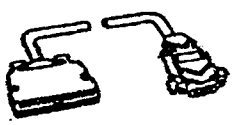
PART#	DESCRIPTION	PINOUTS & SIGNALS	PRODUCT (S)
CA-6	9 Pines a 5 BNC con un 4 Pines Mini-DIN	9 pines 1 Gnd 2 Gnd 3 Red 4 Green 5 Blue 6 Video Compuesto 7 Composite Sync 8 "Y" Luminancia 9 "C" Chroma	ATVista, NuVista, TARGA y TARGA+
		5 BNC 1 Red 2 Green 3 Blue 4 Sincronismo compuesto 5 Video Compuestpo	Cualquier dispositivo de video RGB con conector BNC
		4 Pin Mini-DIN 1 Gnd 2 "Y" Luminancia 3 "C" Chroma 4 Gnd	Cualquier dispositivo S-Video



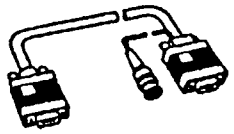
PART#	DESCRIPTION	PINOUTS & SIGNALS	PRODUCT (S)
CA-7	9 Pines a 9 Pines con un 4 Pines mini-DIN y un BNC	9 pines 1 Gnd 2 Gnd 3 Red 4 Green 5 Blue 6 Video Compuesto 7 Composite Sync 8 "Y" Luminancia 9 "C" Chroma	ATVista, NuVista, TARGA y TARGA+
		9 pines 1 Gnd 2 Gnd 3 Red 4 Green 5 Blue 6 7 8 "Y" o H-Sync 9	Monitor Sony 1302, monitores analogicos RGB, dispositivos con salida compuesta.
		4 Pin Mini-DIN 1 Gnd 2 "Y" Luminancia 3 "C" Chroma 4 Gnd	Cualquier dispositivo S-Video.
		1 BNC 1 Compuesto	Cualquier dispositivo con entrada de video compuesto.



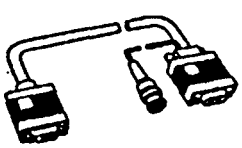
PART#	DESCRIPTION	PINOUTS & SIGNALS	PRODUCT (S)
CA-10	9 Pines a RCA Phono	9 pines 1 2 3 4 5 6 Video Compuesto 7 8 9	ATVista, NuVista, TARGA y TARGA+
		RCA Phono 1 Compuesto	



PART#	DESCRIPTION	PINOUTS & SIGNALS	PRODUCT (S)
CA-12	9 Pines a 34 pines	9 pines 1 Gnd 2 Gnd 3 Red 4 Green 5 Blue 6 7 sincronismo compuesto 8 9	ATVista, NuVista, TARGA y TARGA+
		34 Pines Sony 1 -7 8 Gnd 9 Gnd 10 Gnd 11 13 Gnd 14 -24 25 Red 26 Green 27 Blue 28 -29 30 Sincronismo Compuesto 31 -34	Monitores Sony

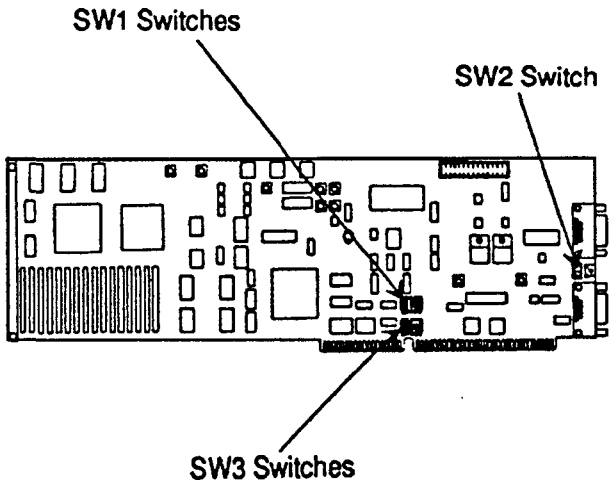


PART#	DESCRIPTION	PINOUTS & SIGNALS	PRODUCT (S)
CA-16	9 Pines a 9 pines y un BNC	9 pines 1 Gnd 2 Gnd 3 Red 4 Green 5 Blue 6 Video Compuesto 7 8 H Sync 9 V Sync	ATVista, NuVista, TARGA y TARGA+
		9 Pines 1 Gnd 2 Gnd 3 Red 4 Green 5 Blue 6 7 8 H Sync 9 V Sync	Monitores Sony 1302 o monitores analogicos RGB
		1 BNC 1 Compuesto	Culaquier dispositivo con entrada de video compuesto



PART#	DESCRIPTION	PINOUTS & SIGNALS	PRODUCT (S)
CA-17	9 Pines a 18 pines y un BNC	9 pines 1 Gnd 2 Gnd 3 Red 4 Green 5 Blue 6 Video Compuesto 7 8 H Sync 9 V Sync	ATVista, NuVista, TARGA y TARGA+
		15 Pines 1 Red 2 Green 3 Blue 4 Gnd 5 6 Red Gnd 7 Green Gnd 8 Blue Gnd 9 10 Gnd 11 Gnd 12 13 H Sync 14 V Sync 15	NEC multisincronos o monitores RGB analogicos que usen un conector de 15 pines
		1 BNC 1 Compuesto	Culaquier dispositivo con entrada de video compuesto

5.4. Configuración de Switches.



Como se puede comprobar, la tarjeta dispone de tres conmutadores: SW1, SW2 y SW3. Los significados de cada uno de los *jumper* se explican a continuación.

SW1 controla la dirección base de I/O. Esto representa la dirección donde el ordenador puede encontrar a la targa+. En el caso de que dispusiéramos de varias tarjetas (no tienen por qué ser iguales), cada una de ellas deberá llevar direcciones diferentes, ya que de lo contrario, se produciría un conflicto de direcciones y el sistema completo no funcionaría correctamente. Obviamente, esta dirección se puede cambiar para solucionar posibles conflictos. La forma de modificarla se muestra en la tabla de la pagina siguiente.

Se puede observar que para modificar la dirección solo es necesario tocar los tres primeros *jumpers*. Esto se debe a que de los cinco disponibles, solo los tres primeros están dedicados a esta tarea. La labor del cuarto jumper es asegurar el modo de trabajo. Esto se usa cuando el software utilizado en el manejo de la tarjeta no ha sido diseñado específicamente para el trabajo con la targa+. Es en este caso el *jumper* debe estar a cero. El quinto, es de uso exclusivo de la tarjeta y no debe ser nunca modificado.

1	2	3	4	5
ON	ON	ON		
ON	ON	OFF		
ON	**OFF**	**ON**		
ON	OFF	OFF		
OFF	ON	ON		
OFF	ON	OFF		
OFF	OFF	ON		
OFF	OFF	OFF		

Switches			BASE
1	2	3	I/O ADDRESS
0	0	0	0x200
0	0	1	0x210
0	**1**	**0**	**0x220**
0	1	1	0x230
1	0	0	0x240
1	0	1	0x250
1	1	0	0x260
1	1	1	0x270

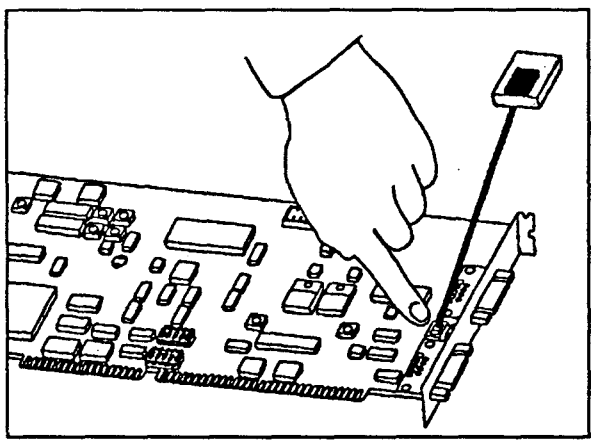
La línea marcada en negrita corresponde a la configuración que tiene por defecto la tarjeta.

Con el conmutador SW2 podremos seleccionar el tipo de sincronismo de salida (horizontal o vertical). De esta forma, y con los cables adecuados, podremos conectar múltiples dispositivos diferentes.

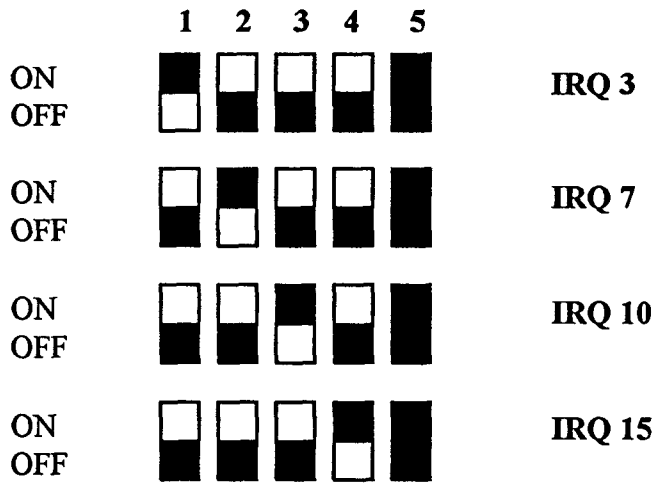
En la ilustración siguiente se muestra la localización del switch SW2. Como su posición depende del tipo de dispositivo que vaya a ser conectado, es necesario tener en cuenta que, cuando cambiemos el dispositivo externo, deberemos revisar su valor, para adaptarlo a las nuevas necesidades. No podremos nunca conectar varios dispositivos a la salida que no tengan las mismas características.

El conmutador SW3 está diseñado para el control de las interrupciones. Una interrupción se puede definir como una llamada de atención que realiza un dispositivo para que la Unidad Central (C.P.U.) le preste atención. Cuando ocurre un evento y la Unidad Central está dedicada a otras tareas, la tarjeta, en este caso la targa+, genera una interrupción y en ese momento, la CPU abandona los trabajos que estaba realizando para atender esa petición. Por último, hay que decir que una interrupción puede ser

compartida o exclusiva. Esto significa que puede ser usada por una única tarjeta o por varias. A esto se le conoce con el nombre de *Interrupt Shared or non-shared.*



Aunque el conmutador consta de cinco elementos, solo se van a utilizar cuatro combinaciones de las treinta y dos posibles. Estas son:



Se puede comprobar que el switch número 5, está desactivado en todas las opciones. Esto se debe a que es usado para definir si la interrupción elegida será compartida o no, tal y como se ha explicado anteriormente.

5.5. Configuración del Software.

Una vez que hemos completado el procedimiento de instalación y configuración del *hardware*, es necesario completar el proceso con la correcta configuración del *software*, que instalará los *drivers* adecuados y necesarios para el perfecto funcionamiento del conjunto del equipo.

Para comenzar con este proceso es necesario arrancar el programa TConfig. Este programa consta de tres pantallas principales, cada una de las cuales nos ayudará a definir determinadas características de la tarjeta. Estas pantallas son:

- *Main Screen.*
- *Tunning Screen.*
- *Hardware setup Screen.*

Main Screen.

Será en esta pantalla donde configuraremos los parámetros principales de la targa+. Esta pantalla consta de tres ventanas:

1. *System Settings.*

Esta es la ventana en la que se debe definir el tipo de tarjeta de que disponemos y el número de éstas. Es posible instalar varias tarjetas y por tanto el software debe conocer perfectamente el número exacto de ellas.

2. *Input Settings.*

En esta ventana definiremos los parámetros de los equipos que vamos a conectar a la entrada. Estos serán:

En esta ventana definiremos los parámetros de los equipos que vamos a conectar a la entrada. Estos serán:

- Tipo de entrada de Vídeo: Podremos elegir entre RGB, compuesto, S-Video o ninguno.
- Señal de Vídeo: Se puede seleccionar entre vivo y grabada. Un ejemplo de señal en vivo es la que procede una cámara de Video y una grabada de un VCR es mostrada con más frecuencia. Este parámetro es utilizado por la tarjeta para su modo de trabajo.
- Genlock: Puede estar en ON o en OFF dependiendo de con quien se sincroniza. Cuando está a OFF, la targa+ se sincroniza con el dispositivo externo de Video. En caso contrario (cuando está a ON) se sincroniza el dispositivo externo con la targa+.

3. *Output Settings.*

Aquí se definirán los parámetros de salida que tendrá a tarjeta. Estos son:

- Resolución: Podremos elegir entre una amplia gamma de resoluciones dependiendo del modo de operación que hallamos seleccionado en la ventana anterior. Podremos elegir entre diferentes resoluciones con diferentes formatos: PAL, NTSC, etc.

- Profundidad de pixel: Esto hace referencia al número de bits por pixel que se utilizarán para almacenar una imagen. Podremos elegir entre 8, 16, 24 y 32.

- Entrelazado: Define la manera de refrescar la imagen en el monitor. En el caso de trabajar con Video en vivo, este parámetro debe estar a ON. En caso contrario debe estar a OFF.

• *Gamma Correction*: Permite llevar a cabo una recoloración de la imagen.

Tunning Screen.

Esta ventana contiene todos los controles para el ajuste de las señales de vídeo, tanto de entrada como de salida. Para jugar con ellos, es recomendable tener conectado algún monitor para poder así ver las modificaciones que estamos realizando.

Hardware setup screen.

Es en esta ventana donde le indicaremos al programa los valores de la configuración hardware que hemos establecido anteriormente. Los parámetros que podemos modificar son:

• *Memory Base Address*: Aquí le indicaremos al programa cual será la zona de la memoria del ordenador donde podrá trabajar. Por defecto, estará en la D000, aunque puede ser modificable. Un aspecto muy importante a tener en cuenta en este apartado, es que cometer un error en la selección de la dirección de memoria, puede llevar a conflictos del hardware que será muy difícil de descubrir.

• I/O Base address: Le indicaremos al programa la dirección donde está localizada la tarjeta. Este valor se debe corresponder con el valor establecido en el *hardware*.

• *Data Transfer Size*: Aquí seleccionaremos el tamaño que queremos usar del bus de datos. Podemos seleccionar entre 8 y 16. Con el ordenador que disponemos, elegiremos 16, ya que el bus es un tipo ISA de 16 bits.

- *Wait States*: Los estado de espera representan el tiempo que debe esperar el ordenador por la tarjeta cada vez que se realice una operación de lectura o escritura, o una operación de entrada y salida.

5.5.1. Configuración de la targa+ para diferentes programas.

Con Microsoft Windows.

El primer paso para la utilización de la tarjeta con el *software* Microsoft Windows, es tener este último instalado y configurado correctamente. Este es un tema ajeno al que estamos tratando y no vamos a entrar en él, ya que su estudio necesita de un profundo desarrollo.

Para poder utilizar la targa+, una vez realizado este paso, es necesario poner como *driver* de vídeo el que incorpora el software de la tarjeta. Para ello podemos seguir dos caminos diferentes.

El primer método consiste en ejecutar desde Windows, un pequeño programa que viene incorporado en el *software* de la targa+, cuya misión consiste en realizar estos pasos por nosotros. Este programa instalará como *driver* de pantalla, el que necesita la tarjeta para su correcto funcionamiento, pudiendo seleccionarlo en dos versiones: *Small Font* y *Large Font*.

El segundo método consiste en realizar nosotros todas las tareas que realiza el programa de instalación del driver. Para ello es necesario conocer el nombre de este pequeño programa, que una vez identificado, lo colocaremos como controlador de vídeo. Estas tareas se realizan en la ventana principal de Windows, en el icono de instalar componentes de Windows.

Ambos métodos alcanzan los mismos objetivos, con lo que podemos utilizar cualquiera de las dos opciones anteriores para realizar esta tarea.

Hay que tener en cuenta que, aunque sigamos fielmente el proceso de instalación, podremos tener problemas. Si tenemos configurada la tarjeta en una zona de memoria diferente a la 0xA000 podrán surgir conflictos de memoria. La forma de solucionar este problema es eliminando esta zona de la zona de trabajo normal de Windows. Para ello pondremos en el fichero SYSTEM.INI, en la sección [386Enh], una línea como la siguiente:

EMMExclude=x000-xFFF

con lo cual excluiremos esta zona de memoria y resolveremos el conflicto.

Con productos de Autodesk.

El proceso de configuración de la targa+ para su uso con productos de Autodesk, tales como Autocad, AutoShade, 3D Studio, etc.., es más sencillo que el que realizamos con Windows.

Como el proceso de configuración es similar para todos estos programas, comentaré en líneas generales los pasos a seguir y, a continuación, comentaré las peculiaridades de cada uno de ellos.

En primer lugar, debemos identificar el *driver* adecuado para colocarlo en el subdirectorio *..\driver* del programa en cuestión. Una vez hecho esto, es necesario ejecutar una función desde la línea de comando o *prompt*, aunque si el programa lo vamos a utilizar con mucha frecuencia, esta línea puede escribirse en el fichero de arranque AUTOEXEXC.BAT. Esta línea instalará el *driver* anterior como *driver* de

vídeo, con lo cual solo nos quedaría entrar en nuestro programa y especificarle el que debe usar.

Hasta aquí son los pasos a seguir de forma general. Ahora lo que debemos hacer es identificar los *drivers* y las líneas de comando para cada uno de los programas.

	AUTOCAD	3D STUDIO
DRIVER	RCPTARGP.EXP	RCPTARGP.EXP
LÍNEA DE COMANDO	SET DSPADI=C:\ACAD\RCPTARGP.EXP	SET RCPADI=C:\3DS\DRIVERS\RCPTARGP.EXP

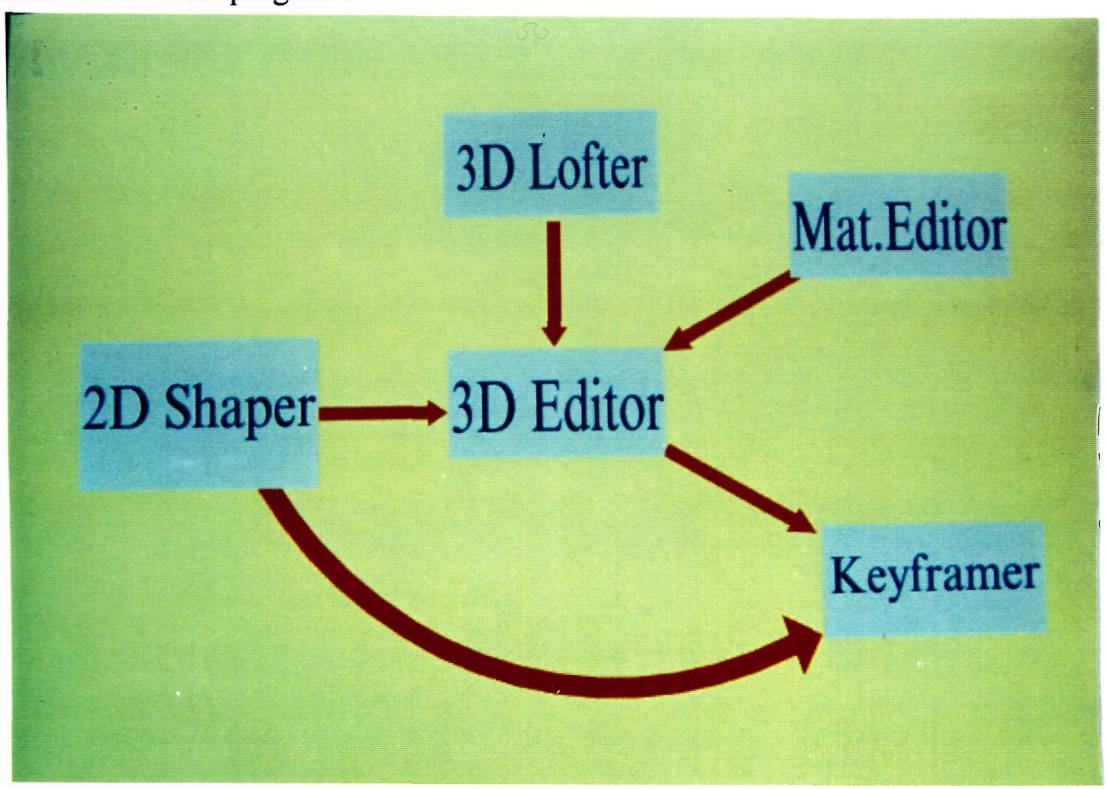
En programas como el ANIPLAY no es necesario realizar este proceso. Solo hace falta seleccionar el *driver* en el programa que vayamos a utilizar.

6. Proceso de construcción de una animación tridimensional.

Para explicar el proceso de construcción de una animación tridimensional, no podemos hacer referencia a programas de dibujo que existan en el mercado, ya que la filosofía de trabajo es completamente diferente al resto y por tanto no existe ningún tipo de semejanzas.

Hay que tener en cuenta que el *3D Studio* trabaja en tres dimensiones y que, por tanto, no dibujamos, ya que para este proceso solo es necesario un plano. Lo que vamos a crear son objetos tridimensionales (mallados).

La siguiente ilustración nos ayudará a entender un poco mejor todo el funcionamiento del programa.



Ilust. 1. *Esquema de bloques del 3D Studio*

El proceso que se lleva a cabo cuando queremos construir una animación, lo podemos resumir de la forma siguiente. Primero, debemos crear los elementos u objetos que queremos poner en escena. Para ello usaremos el *3D Editor*, que nos dará las herramientas necesarias para diseñar los objetos. Una vez creados, debemos poner luces en la escena a animar, con lo cual se obtiene un efecto mucho más fotorrealista. El siguiente proceso sería aplicar materiales o texturas a los elementos que tenemos. El concepto de material o textura se explicará más adelante.

Lo único que nos falta sería aplicar movimiento a los objetos. Para este proceso usaremos el *Keyframer*. Por último debemos poner cámaras para poder observar los movimientos desde los puntos de vista deseados.

Una vez realizada toda la animación, solo nos faltaría hacer el render, que se puede definir como el proceso por el cual se muestra el resultado definitivo a partir de todas las operaciones realizadas anteriormente.

Se puede observar la gran semejanza que existe entre el proceso de construcción de una animación y el proceso de construcción de una secuencia de vídeo real. Para entender un poco mejor todo este complejo procedimiento, pasamos a analizar cada uno de los bloques del programa con un poco más de detenimiento.

6.1. 3D Editor.

El *3D Editor* es el módulo al que accedemos cuando cargamos el programa y nos va a permitir crear y editar objetos complejos en 3D a partir de figuras tridimensionales sencillas (cubos, conos, cilindros, toros, etc.). Este será el elemento desde el cual controlaremos el total de la animación.

Permite crear y editar objetos en 3D (coordenadas X,Y,Z), aplicar texturas a las superficies, incorporar luces a la escena y efectuar el render del modelo. Se pueden crear objetos con las herramientas de diseño incorporadas en el mismo programa, o bien usar los módulos *2D Shaper* y *3D Lofter*. Posee suficientes funciones de dibujo y edición como para conseguir que cualquier objeto pueda definirse a base de mallas tridimensionales.

Para conseguir figuras complejas nos basamos en la creación de figuras sencillas, que poco a poco iremos modificando y uniendo para formar la figura deseada.

Vamos a hacer un estudio por opciones para intentar clarificar el funcionamiento de este módulo. Las opciones más relevantes que incorpora este módulo del programa son:

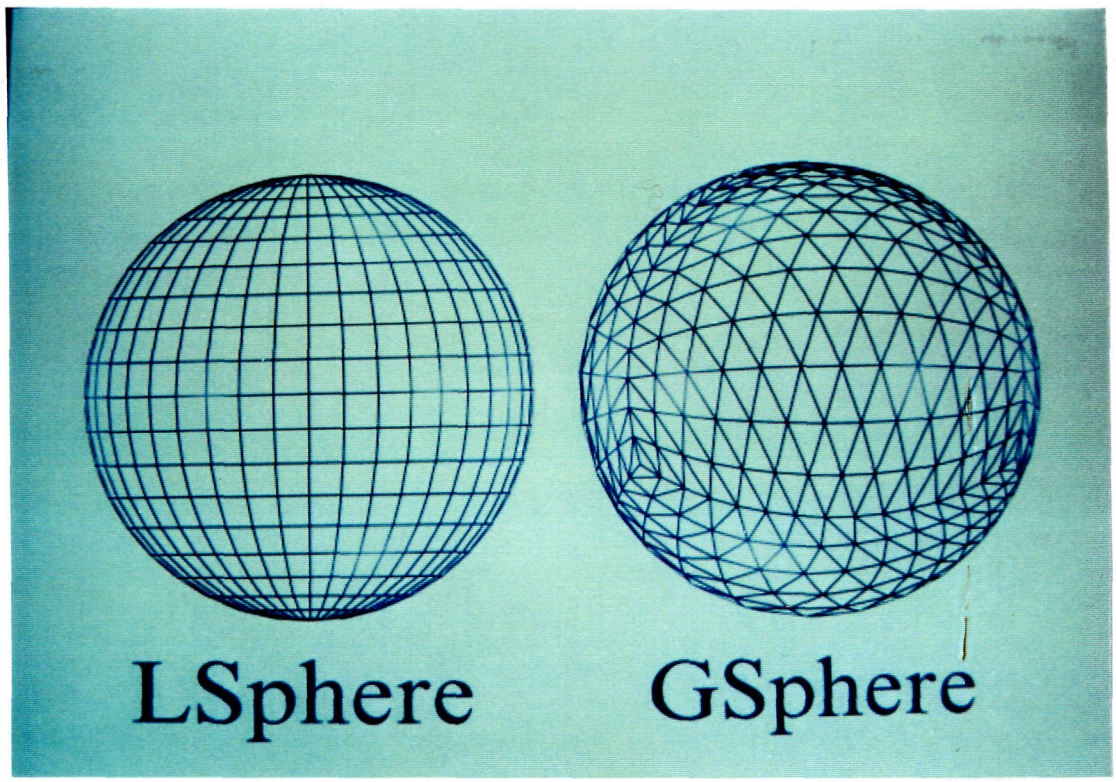
- Create.
- Modify.
- Surface.
- Lights.
- Cameras.

Create.

Esta opción nos permitirá crear distintos tipos de elementos, como son primitivas, caras y elementos extraídos de otros módulos del programa y de distintos programas.

En la opción de creación de primitivas, podremos crear directamente cajas, semiesferas, tubos, toros, cilindros, conos y esferas. Estas ultimas podrán ser de dos tipos diferentes: *LSphere* y *Gsphere*. Las diferencias entre ambas se muestran en la figura siguiente.

Para cada una de las primitivas que se pueden crear directamente habrá que indicar una serie de parámetros como son: diámetro de las esferas, alturas y radios de los conos y cilindros, diámetro mayor y menor de los tubos, etc. . Una vez indicados los paramentos necesarios podremos observar diferentes vistas de la primitiva creada. Nos basaremos en estos objetos elementales para crear otros mas complejos. Podemos comprobar en cualquiera de las vistas que el objeto creado es una malla formada por múltiples vértices y múltiples caras, las cuales podremos modificar posteriormente.



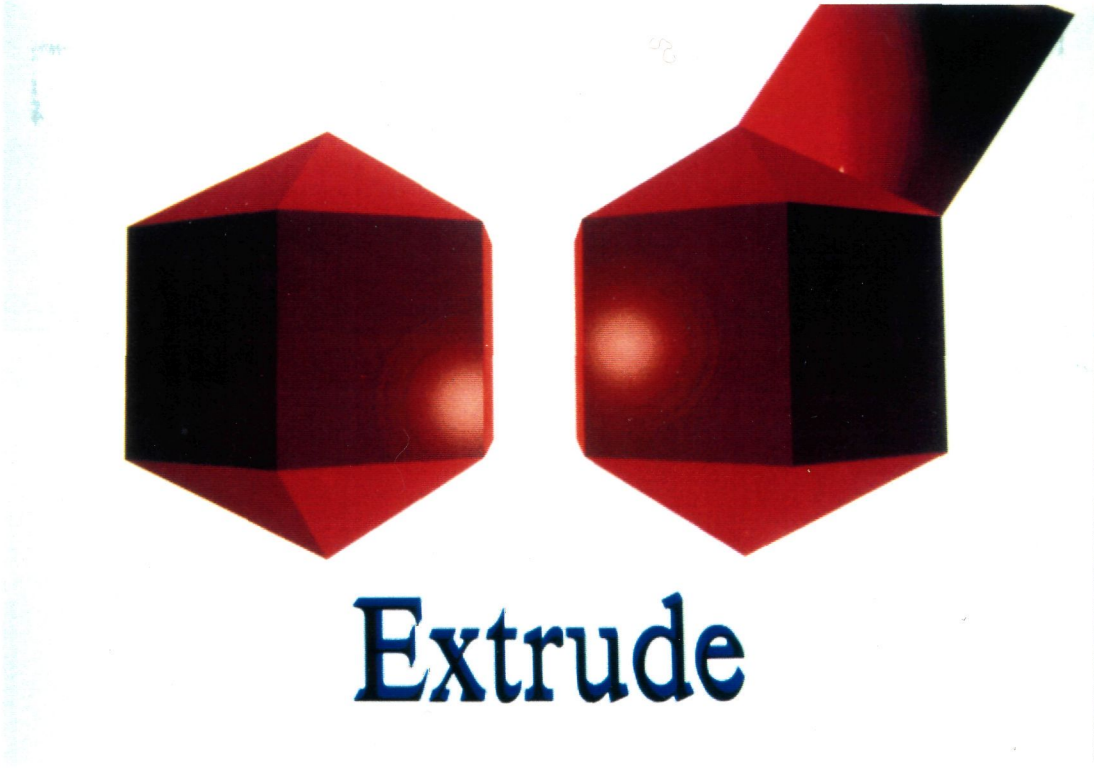
Ilust. 2. *Diferencia entre Lsphere y GSphere.*

Otra de las opciones posibles es la creación de caras (*faces*), que nos permitirá realizar múltiples operaciones con las caras de los objetos creados anteriormente. De entre estas operaciones, las mas importantes pueden ser:

- *Build.* Construye una cara a partir de la definición de tres vértices.

- *Extrude*. Da altura a una cara. Para ello debemos seleccionar la cara a modificar e indicar la altura que queremos darle. E1 resultado obtenido sigue siendo parte del mismo objeto.
- *Tessellate*. A partir de una cara seleccionada, realiza una división de la misma.

La ultima opción que comentaremos dentro de *Create*, es la que hace referencia a objetos. Haré especial hincapié en este menú debido a la gran versatilidad que proporcionan estas herramientas en la creación de objetos. Disponemos de las siguientes posibilidades:



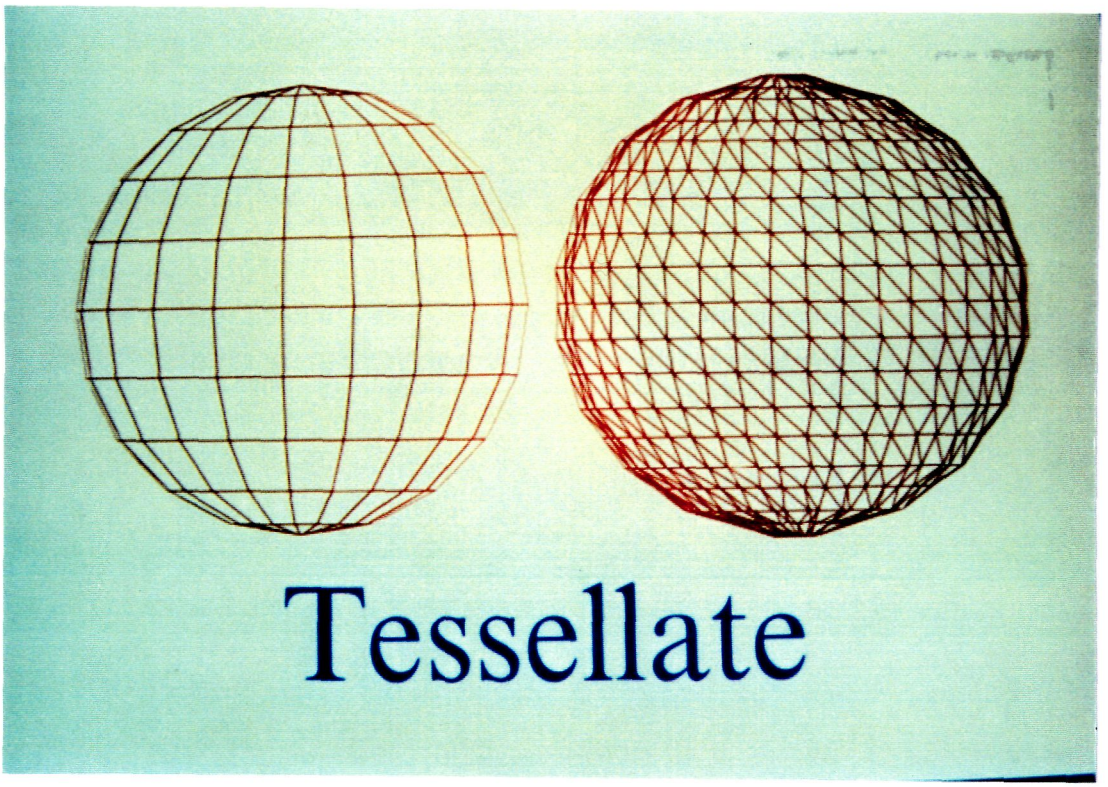
Ilust. 3. *Ejemplo de Extrude.*

• Copiar: Nos permitirá duplicar cualquier objeto que ya tengamos hecho.

• *Attach*: Une varios objetos en uno solo sin variar los parámetros de cada uno de ellos. Esto tiene especial importancia, cuando queremos realizar algún

movimiento con un grupo de objetos, duplicarlos, o simplemente unirlos para crear uno único.

• *Tesellate*: Realiza la misma operación que la realizada antes con las caras, pero ahora con los objetos. Veamos un ejemplo:



Ilust. 4. *Ejemplo de Tesellate.*

• *Get Shape*: Proporciona la posibilidad de incluir elementos diseñados con el *2D Shaper* en el módulo *3D Editor*. Esta opción es importante, porque es la única posibilidad que da el programa de incluir elementos en dos dimensiones para utilizarlos en el espacio. El elemento que deseemos utilizar debe haber sido seleccionado con la opción *Shape/Assign* en el otro módulo.

• Ya la ultima opción a comentar es la *Booleana*, que permite realizar este tipo de operaciones con dos objetos. Las opciones que permite son unión, sustracción

y diferencia. Esta opción ofrece una gran versatilidad en la creación de elementos, ya que podemos generar complejos objetos a partir de elementos sencillos.

Modify.

Este menú nos permitirá la modificación de los distintos componentes de una animación, como son objetos, elementos, caras, vértices, etc...

Debido a la gran similitud entre las opciones de modificación para los distintos componentes, solo haré una breve descripción de las herramientas de modificación que incorpora el programa, pero sin entrar en los detalles característicos destinados a la modificación de cada elemento.

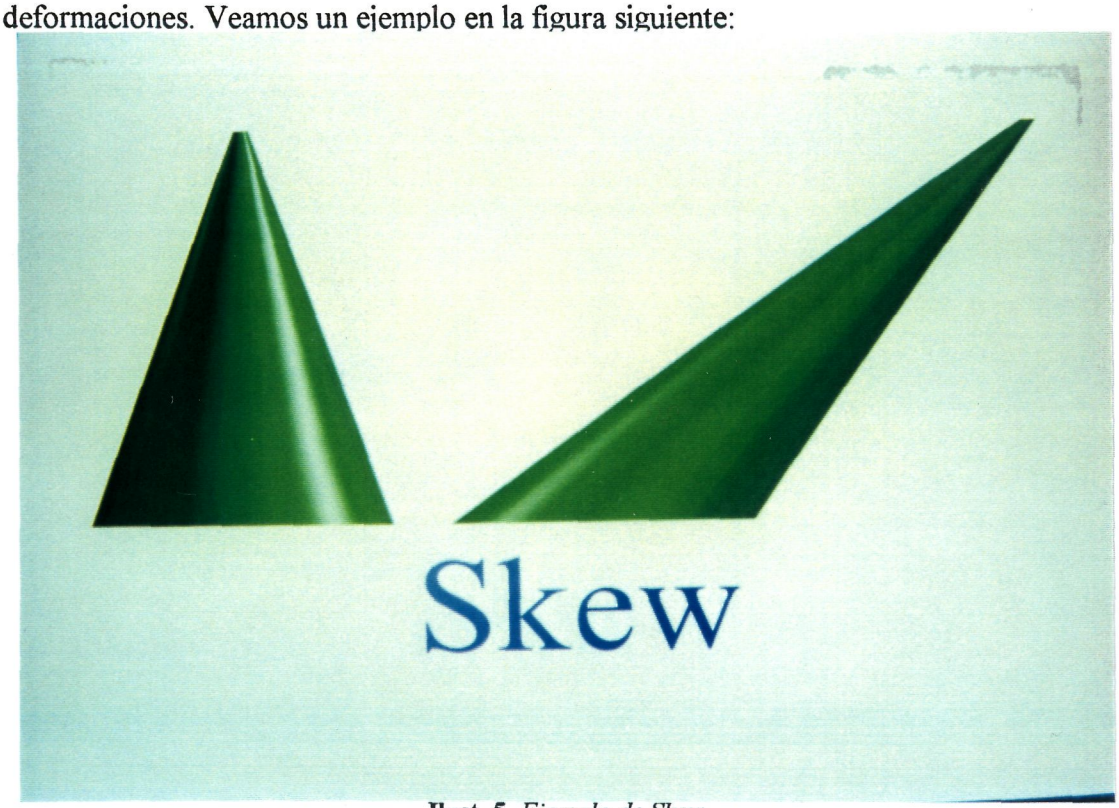
La opción *Move* permite mover los objetos en el espacio. Hay que tener en cuenta que excepto en la ventana de usuario, los movimientos que realicemos serán en dos dimensiones y no en el espacio, así que si queremos mover un objeto en el espacio deberemos reliazarlo en varias ventanas a la vez, con lo cual conseguiremos el efecto deseado.

La opción *Rotate*, realiza la rotación de un elemento en el espacio. Podemos escoger entre dos opciones posibles: rotación en el plano y rotación absoluta. La primera de ellas se realiza en dos dimensiones y la segunda en tres. De esta forma podremos mostrar las caras ocultas de los objetos.

Por último, aparece la opción *Scale*, la cual nos permitirá aplicar un escalado a un objeto en dos y tres dimensiones.

Hasta aquí, hemos comentado las opciones de modificación mas básicas. Las siguientes nos van a permitir obtener objetos complejos a partir de primitivas sencillas.

Una de las más importantes es *Skew*, las cual nos permitirá estirar un objeto "agarrándolo" por dos extremos. Esta opción es muy importante en la realización de deformaciones. Veamos un ejemplo en la figura siguiente:

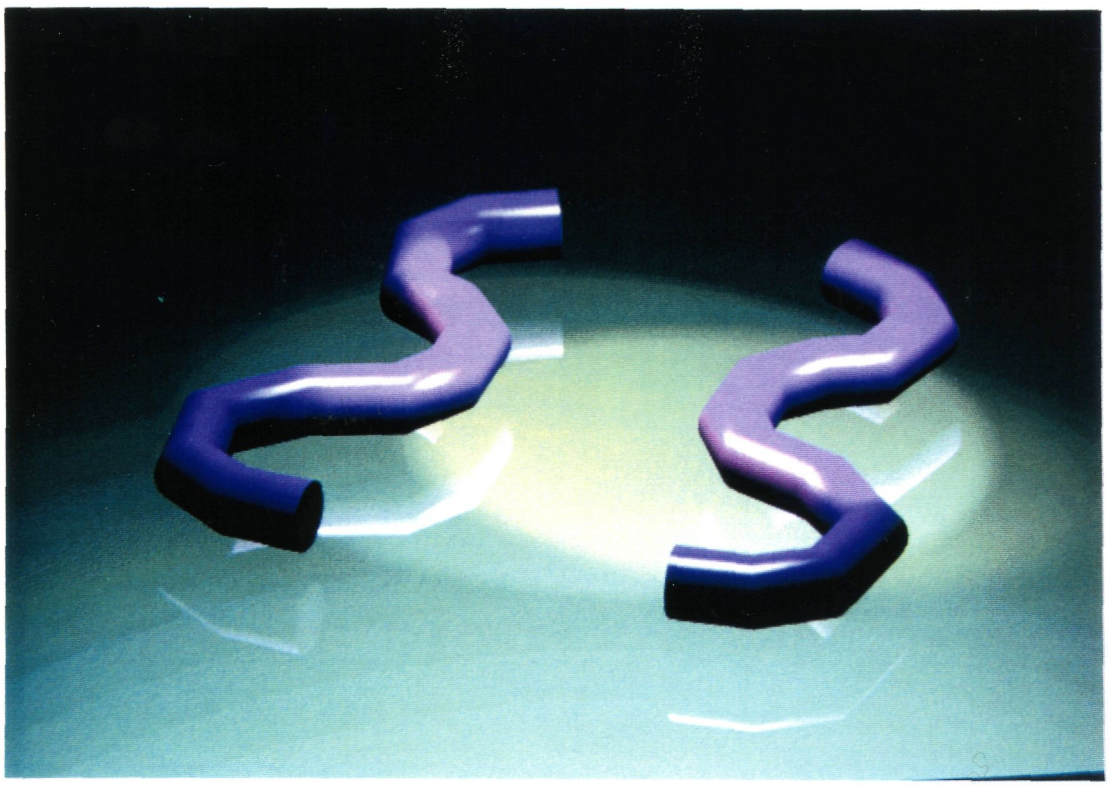


Ilust. 5. *Ejemplo de Skew.*

Otra de las opciones es *Mirror*, con la que podemos obtener una imagen espejo de la primera según el eje indicado. Se puede realizar de dos formas: la primera de ellas consiste en reflejar simplemente el objeto y la segunda consiste en obtener un segundo objeto a partir del primero.

Las otras dos opciones más relevantes de este submenu son *Taper* y *Bend*, cuyo efecto consiste en doblar los objetos seleccionados.

Hay que tener en cuenta que con estos ejemplos se pretende visualizar de una forma sencilla y entendible el efecto que producen, pero ambos podrían haber sido realizados en otra dirección, con lo que el efecto conseguido hubiese sido diferente.



Ilust. 6. *Ejemplo de la opción Mirror.*

Ilust. 7. *Otro ejemplo de Mirror.*

Surface.

Una vez llegados a este punto, vamos a pasar a asignar a cada objeto un material con unas características especificas. Podemos definir material, como aquel color que, junto con una serie de propiedades, da al objeto en cuestión la apariencia deseada. En este apartado no estudiaremos como conseguir los materiales deseados, ya que eso es un tema perteneciente al *materials editor*, sino como aplicarlos y las posibles variaciones que podemos hacer en ellos.

La opcion principal que presenta este menú de superficies es la de escoger un material entre un conjunto de ellos, que conforman una librería de materiales. Podemos escoger entre varias librerías diferentes, pudiendo crear, si queremos, la nuestra propia. Cuando se escoge un material de la librería, se nos muestra para cada material, los procesos que lleva asociados. Estos procesos hacen referencia a transparencia, textura, opacidad, *bump mapping*, etc... Todos estos términos se aclararan en el apartado *Materials Editor*.

Esta información la usaremos principalmente para saber si debemos mapear el objeto, lo cual haremos en una segunda opción (*Surface/mapping*). Mapear un objeto significa decirle al render en que parte de la geometría del objeto queremos aplicar la textura del material elegido.

Se puede demostrar que dos figuras aparentemente distintas, pueden haber sido diseñadas con el mismo material. La única diferencia entre ambos es que el mapeado de uno abarca el objeto completo, mientras que el del puede ser mucho mas pequeño, con lo cual al tener que abarcar la figura completa tiene que aplicar el material muchas veces.

El software permite aplicar el mapping de tres formas diferentes: plano, circular y elíptico. Esto se hace para poder conseguir resultados diferentes según el objeto.

Lights.

Disponemos de dos tipos diferentes de luces: omnidireccionales y tipo *Spot*. La primera de ellas ilumina igual en todos los sentidos, mientras que la segunda es similar a un foco. En ambas podemos seleccionar el color con el que deseamos que ilumine. Podemos aplicar a la escena tanta luces como queramos, pero hay que tener en cuenta que un exceso de luz puede saturar una imagen.

Cameras.

Con ellas estableceremos los puntos de vista desde los cuales queremos observar los objetos anteriormente diseñados. Podemos definir tantas cámaras como queramos, aunque normalmente con dos será suficiente.

En una cámara podemos variar diferentes parámetros, como son: apertura del foco, distancia focal, giro, etc. Para poder comprobar que es lo que esta viendo la cámara, definimos uno de los viewports para la vista de cámara.

Llegados a este punto, conviene hacer un breve resumen de lo que llevamos hecho hasta ahora. Hemos creado los objetos con las características deseadas, les hemos aplicado materiales, hemos puesto luces y por ultimo hemos añadido las cámaras. Hemos terminado la escena y por tanto debemos pasar a hacer el render para obtener el producto final. Después, bastará con unas pequeñas modificaciones para alcanzar el resultado definitivo.

6.2. 2D Shaper.

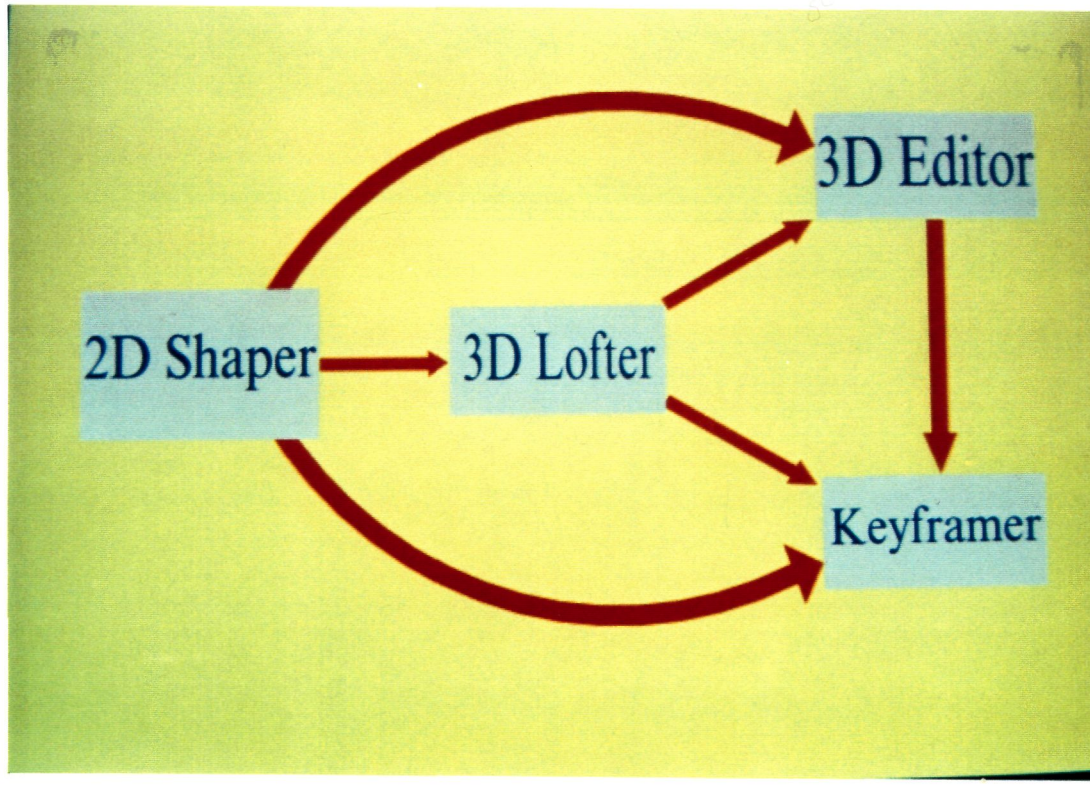
El módulo *2D Shaper* no trabaja en tres dimensiones como el anterior, sino en dos. Es parecido a un programa convencional de dibujo por ordenador. Para ver que podemos realizar con él, analizaremos el esquema de la figura siguiente:

Las tareas que podemos realizar son las siguientes:

• Crear un polígono en 2 dimensiones para ser utilizado en el *3D Editor* como un objeto plano.

• Crear un polígono en dos dimensiones para pasarlo por el *3D Lofter* y así obtener un objeto en tres dimensiones. El único requisito es que el polígono debe estar cerrado.

• Utilizar una línea abierta o un polígono cerrado para ser utilizada por el *3D Lofter* como eje.



Ilust.8. *Esquema de bloques del programa a partir del 2D Shaper.*

• Utilizar una línea abierta o un polígono cerrado para ser utilizado por el *Keyframer* como path o línea de movimientos.

Como se puede apreciar, las utilidades del *2D Shaper* son muchas y por tanto solo vamos a comentar las más relevantes, que son:

Create.

Esta opción o menú despliega un submenú que nos permitirá crear directamente elementos típicos de un programa de dos dimensiones, como son: arcos, círculos, polígonos de n lados, elipses, y como no, dibujar a mano y crear texto con un amplio surtido de fuentes.

Otra de las posibilidades que ofrece es conectar dos vértices o cerrar polígonos. En esta ultima acción hay que tener en cuenta que si desconocemos la situación de los vértices abiertos, al ejecutar la acción puede aparecer una línea en un lugar imprevisto.

Modify.

Esta opción es muy parecida a la opción de modificar objetos en el *3D Editor*. Asi como antes realizábamos un análisis de posibles modificaciones respecto a vértices, caras, objetos, etc.., ahora el análisis hará referencia a vértices, segmentos y polígonos.

Las opciones de modificación más importantes son:

- *Move*: Desplaza un vértice o polígono a una posición diferente.
- *Rotate*: Rota un vértice o polígono según sea la posición del eje de rotación.
- *Scale*: Escala un vértice y los segmentos asociados a él.
- *Mirror*: Realiza la función de espejo solo con polígonos.
- *Adjust*: Realiza pequeñas modificaciones de los segmentos asociados a un vértice.
- *Linear*: Alinea segmentos curvados.

- *Curve*: Curva segmentos lineales creando líneas tipo Spline.
- *Delete*: Borra vértices, segmentos o polígonos.

Por último, si queremos utilizar el módulo *Modify* para realizar algunas de las tareas propias del *2D Shaper*, como por ejemplo añadir al *3D Editor* un objeto de dos dimensiones, tendremos que asignar o seleccionar los poligonos en cuestión a través de la opción *Shape/Assign*.

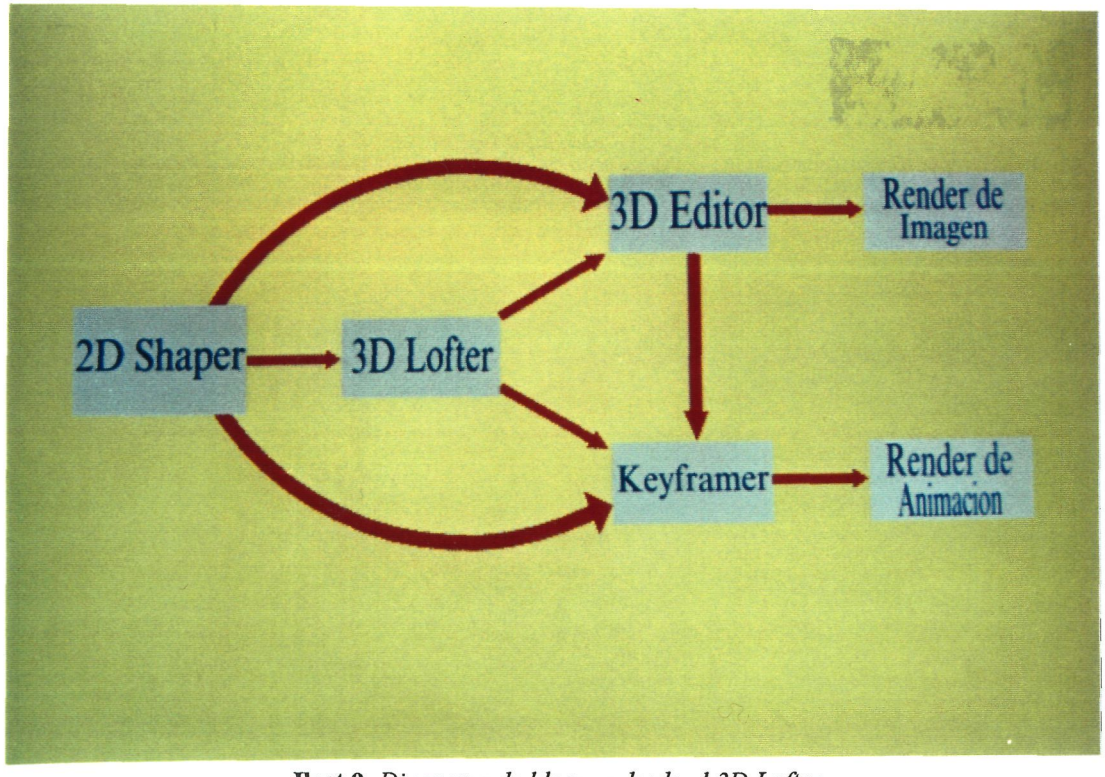
6.3. 3D Lofter.

Una vez que hemos diseñado un polígono en el *2D Shaper*, podremos usar el módulo *3D Lofter* para:

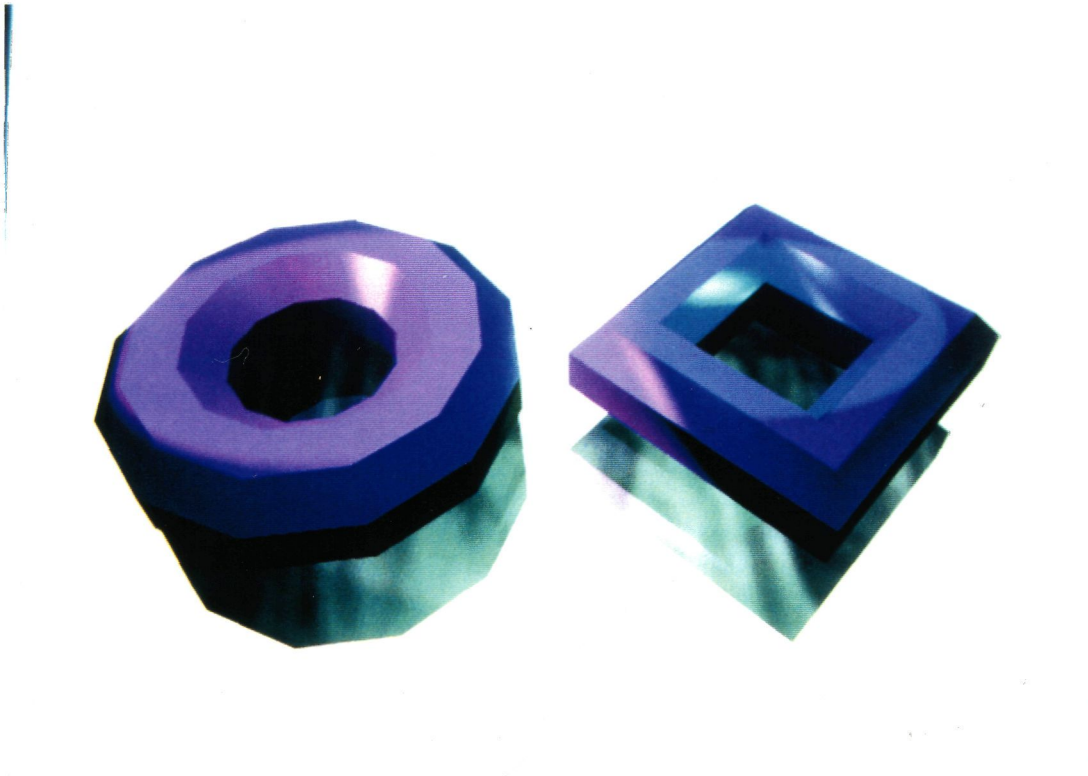
- Importar un polígono desde el *2D Shaper*, para convertirlo en un objeto mallado en tres dimensiones.
- Importar una línea del *2D Shaper* para sustituirlo por el path que existe por defecto.

Cuando creamos un objeto mallado en tres dimensiones a partir del 3D Lofter, el programa envía el objeto al *3D Editor* para que sea tratado como un objeto más, es decir, le aplicamos texturas, luces, etc..

El funcionamiento general de esta parte del programa se basa en construir objetos mallados de tres dimensiones a partir de objetos planos. Esto se consigue estirando el objeto generado por el *2D Shaper* a partir de un eje. Aparentemente, el eje debería ser una línea recta sobre la cual colocamos repetidas veces el elemento plano para conseguir una figura en 3D, pero no es así. El eje sobre el que prolongaremos el plano puede tener distintas formas y puede ser modificado posteriormente, con lo cual se obtendrán diferentes figuras a partir de un mismo plano. Este es el tema del que trata el *3D Lofter* y a continuación comentaremos algunas de las propiedades más importantes.



Ilust 9. *Diagrama de bloques desde el 3D Lofter.*



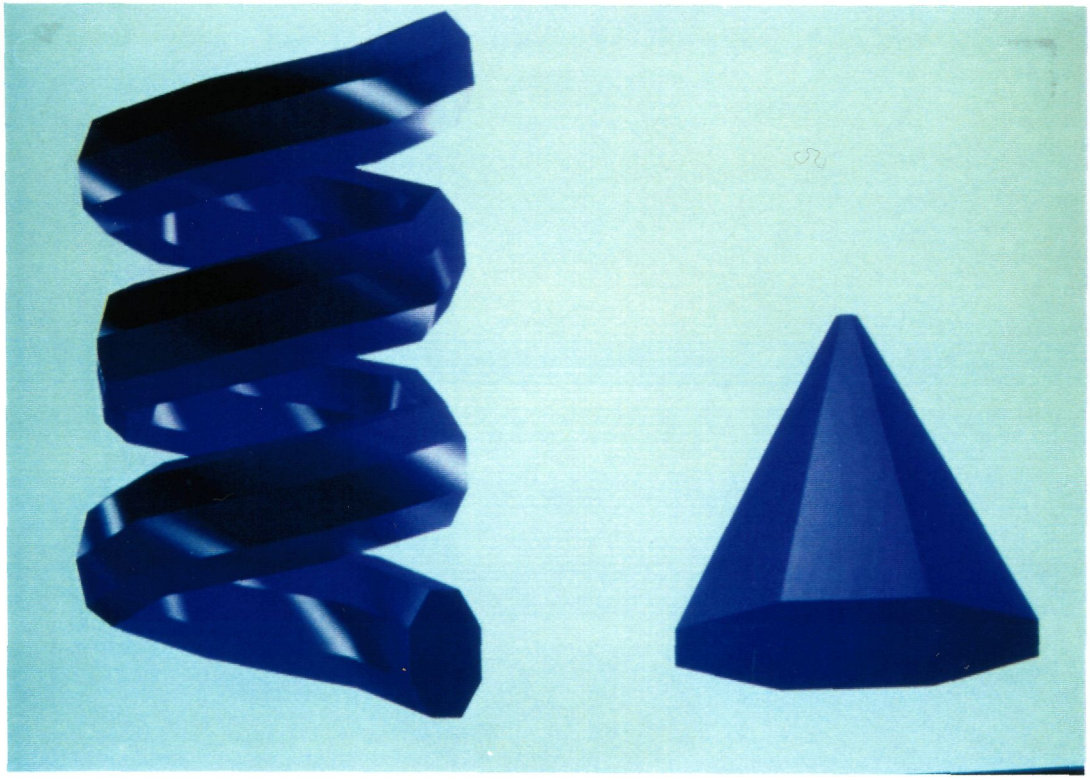
Ilust. 10. *Objetos con el mismo shape y distinto path.*

El eje de estiramiento o *path*, se compone de una línea con múltiples vértices. A cada uno de los vértices podemos asignarle un *Shape* diferente, con lo cual podremos

conseguir objetos muy complejos de dibujar a partir de herramientas que simplifican enormemente esta tarea.

Las opciones más importantes en este módulo del programa, son:

- Shapes.
- Paths.
- Deform.



Ilust. 11. *Objetos con el mismo shape y distinto path.*

Shapes.

Este menú hace referencia a la obtención y modificación de objetos planos destinados a crear de objetos 3D. En primer lugar, lo que debemos hacer es obtener una figura plana. Para ello tenemos dos posibilidades: importarla del *2D Shaper* o cargar una directamente de disco (extensión *.shp).

Una vez que la tenemos, podremos centrarla con respecto al eje, moverla, rotarla, escalarla, etc.., hasta conseguir que la figura tenga la forma y posición deseada. No es excesivamente importante lograr una gran precisión en este apartado, ya que una vez diseñada la figura en 3D podremos modificarla con todas las herramientas del *3D Editor*.

Path.

El *path* es el eje sobre el que "estiraremos" un objeto plano para crear uno en 3D. Se compone de una línea recta (2 vértices) por defecto, aunque podremos variar su forma y número de vértices, convirtiéndola en una recta tipo *Spline*.

Para variar su forma, podemos proceder de dos maneras bien diferenciadas: utilizar las herramientas que posee el módulo en cuestión o bien exportar el *path* al *2D Shaper*.

Utilizando las herramientas que nos aporta podremos insertar el número de vértices necesarios, para posteriormente moverlos y modificar así la curvatura, convirtiendo la recta en una *Spline* con la forma deseada.

Una de las opciones más importantes de que disponemos en la modificación de *paths*, es la posibilidad de crear espirales y revoluciones, con lo cual podremos obtener un objeto en 3D simétrico a partir de una revolución del mismo.(Ver figura siguiente)

Deform.

Esta será la última opción que comentaremos dentro del *3D Lofter* que, debido a su importancia en la creación de objetos, revisaremos más detenidamente. Existen cuatro tipos de deformaciones básicas, que son: *Scale, Twist, Bevel* y *Fit*.

Para todos ellos, existe un *grid* común, que nos facilitará la tarea de realizar las deformaciones. Las escalas y valores por defecto del mismo, variarán en función del tipo de deformación a emplear.

Ilust. 12. *Ejemplo de revolución.*

Deform/Scale.

Este tipo de deformación consiste en escalar los distintos *shapes* que vamos colocando en los vértices. Para entender un poco mejor este concepto, supongamos un reloj de arena. Está formado por circunferencias que se van escalando a medida que nos

movemos por su eje de simetría. Conseguir este efecto con este tipo de deformación es muy sencillo, ya que simplemente emplearemos el *grid* para indicar el tanto por ciento de escalado que deseamos tener en cada vértice del *path*.

Deform/Twist.

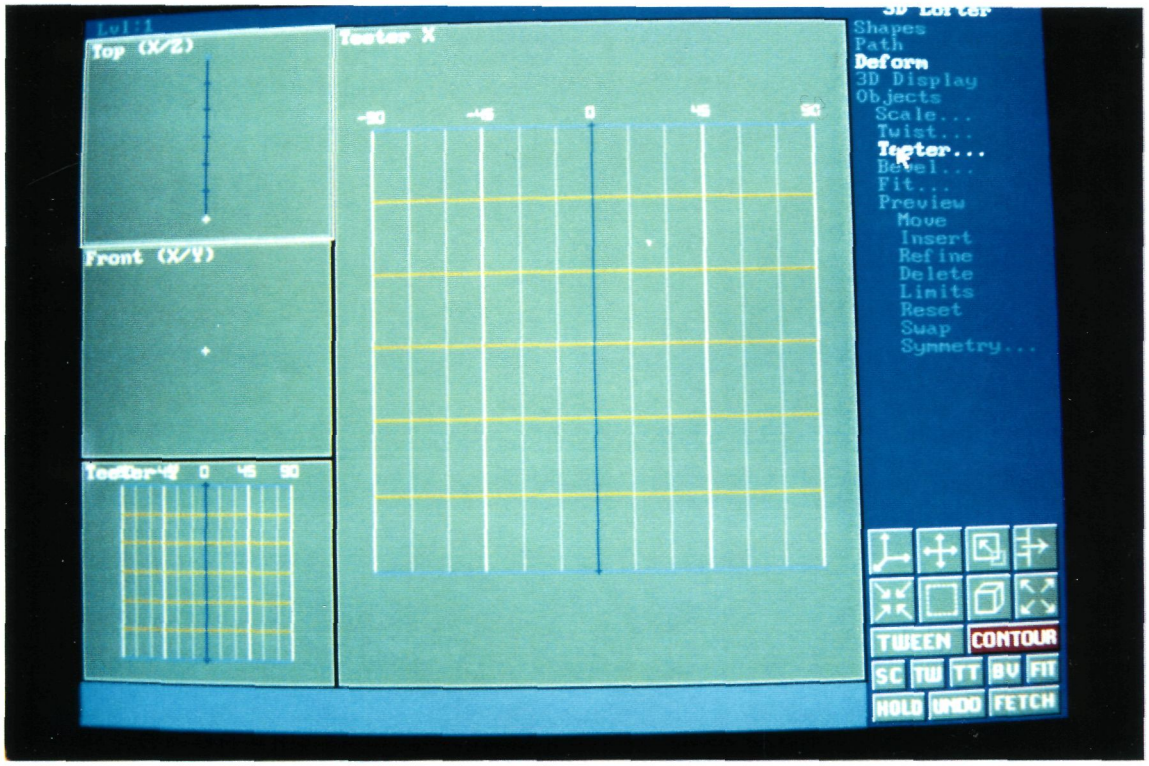
La filosofía de esta deformación es similar a la anterior, con la particularidad de que no hace referencia al escalado, sino a la torsión. Consiste en coger un elemento 3D por sus tapas (supongamos un cilindro) y retorcerlo a modo de un envoltorio de caramelo. Por tanto, el parámetro necesario que debemos indicar para esta deformación será el ángulo de giro del eje.

Ilust. 12. *Ejemplo de deformación.*

Deform/Teeter.

Permite realizar una deformación por ejes. Para entenderlo, veamos la figura siguiente (Ilustración 13).:

Podemos seleccionar el eje sobre el cual queremos realizar la deformación, o bien, realizarlo sobre todos a la vez, a través de la opción de simetría. El resto de las opciones que incorpora son las mismas que para las deformaciones anteriores.

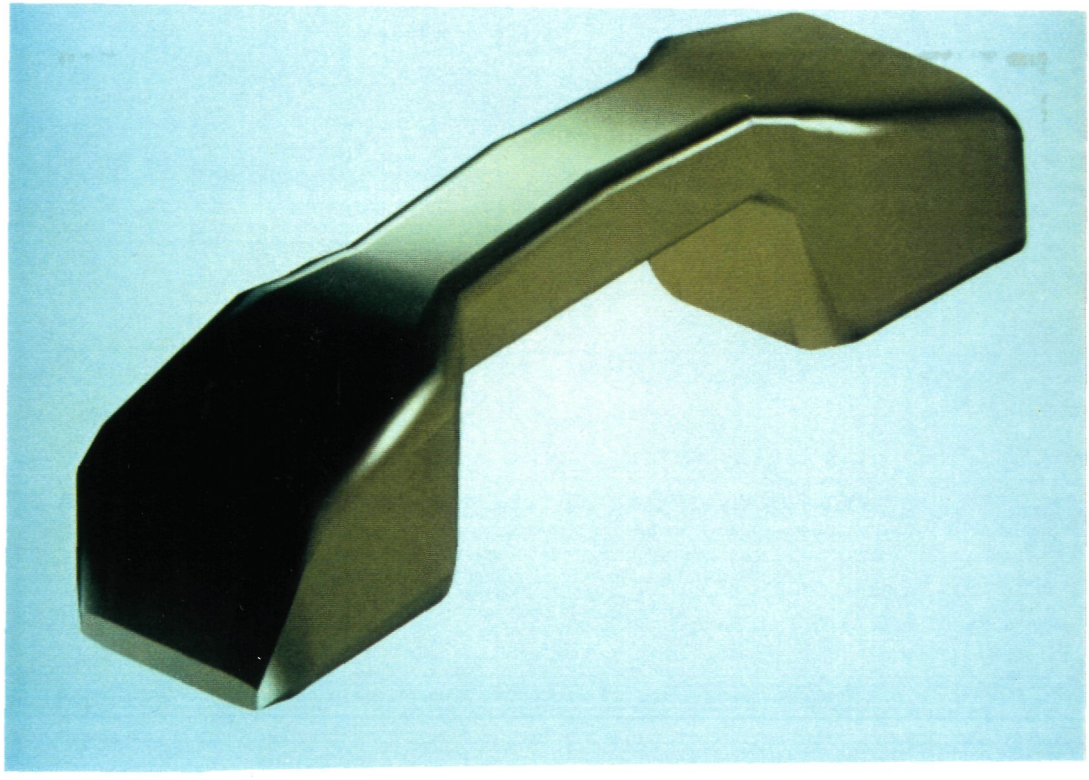


Ilust. 13. *Pantalla principal de la opción Deform/teeter*

Deform/Fit.

Este tipo de deformación es una de las más importantes que incorpora el *3D Studio*, debido a la gran facilidad que se obtiene para la generación de objetos complejos. Permite generar objetos 3D a partir de su planta y alzado.

Para ello, usaremos el *2D Shaper*, en el cual crearemos los polígonos necesarios, tanto de planta como de perfil. Veamos esta opción con un ejemplo:



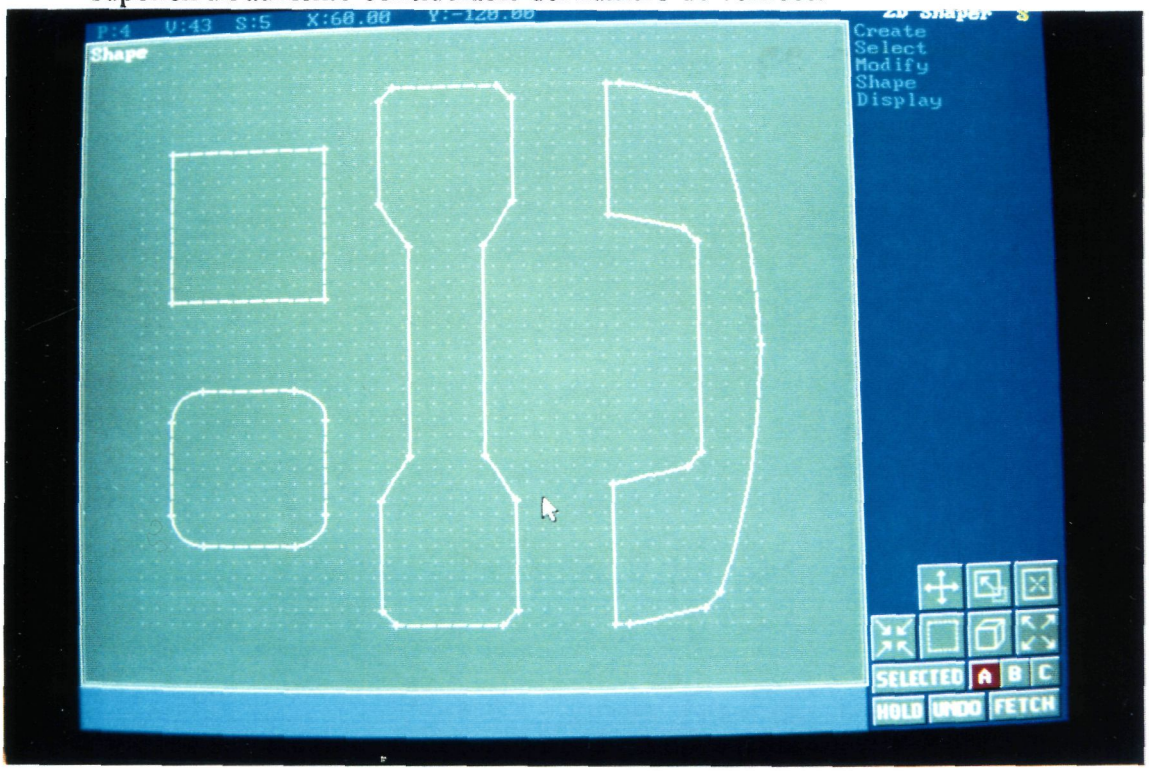
Ilust. 14. *Imagen generada con la deformación Fit.*

Supongamos que queremos generar la figura anterior. Para ello, crearemos dos polígonos cerrados en el *2D Shaper*. El primero de ellos será un perfil del objeto y el segundo una vista de planta del mismo. Una vez generados, colocaremos cada una de los polígonos en las vistas X e Y del *3D Lofter* y a continuación haremos la generación automática del *path*, con lo cual siguiendo a partir de aquí los pasos normales para generar un objeto, obtendremos una figura completamente tridimensional a partir de dos polígonos sencillos. Podremos realizar un *preview* para ver como va a quedar la figura antes de generarla

Hay que saber que, cuando realizamos este tipo de deformaciones, vamos a generar un objeto mallado muy sofisticado y que, por tanto, hay que tener en cuenta una

serie de advertencias a tener en cuenta para evitar generar objetos excesivamente complicados. Estas son:

- Es preferible utilizar líneas rectas en lugar de curvas, ya que estas ultimas suponen un aumento considerable del numero de vértices.



Ilust. 15. *Planta y perfil del teléfono anterior.*

- Los polígonos que utilicemos como planta y perfil deben estar alineados, ya que de lo contrario supondría, al igual que antes, un aumento del número de vértices.

- Siempre que el objeto a realizar lo permita, deberemos usar el mismo polígono como planta y como perfil, ya que esto simplifica mucho el calculo.

- Hay que tener en cuenta que todo lo anterior repercute en la complejidad del objeto, y por tanto, en el tiempo de generación de la malla y en el tiempo de render, más aún si tenemos otros objetos en la escena.

6.4. Materials Editor.

Ilust. 16. *Aspecto general del Materials Editor.*

Este es el módulo del programa que generará o modificará los materiales, colores o texturas que emplearemos en los objetos mallados. Antes de entrar en el modo de funcionamiento del mismo, es conveniente entender una serie de conceptos, como son:

- *Color*. El programa permite seleccionar un color mediante dos métodos diferentes: RGB y HLS. El primero de ellos consigue un color a partir de la mezcla entre los tres colores primarios: Rojo, azul y amarillo. Con la mezcla de estos tres en diferentes proporciones podremos obtener cualquier color. El segundo método (HLS) se basa en lo mismo de antes excepto que ahora en vez de mezclar colores, mezclamos luminancia y saturación.

- *Surface Color*. En este apartado controlaremos la manera en que un objeto reflejará un color en función de la luz directa que le afecte.

• *Mapping*. Un "mapeado" sobre un objeto, es el resultado de aplicar una imagen sobre la superficie de un objeto. Para entender mejor este concepto, supongamos una botella con una etiqueta en su barriga. La etiqueta sería el resultado obtenido al aplicarle un mapeado a la botella.

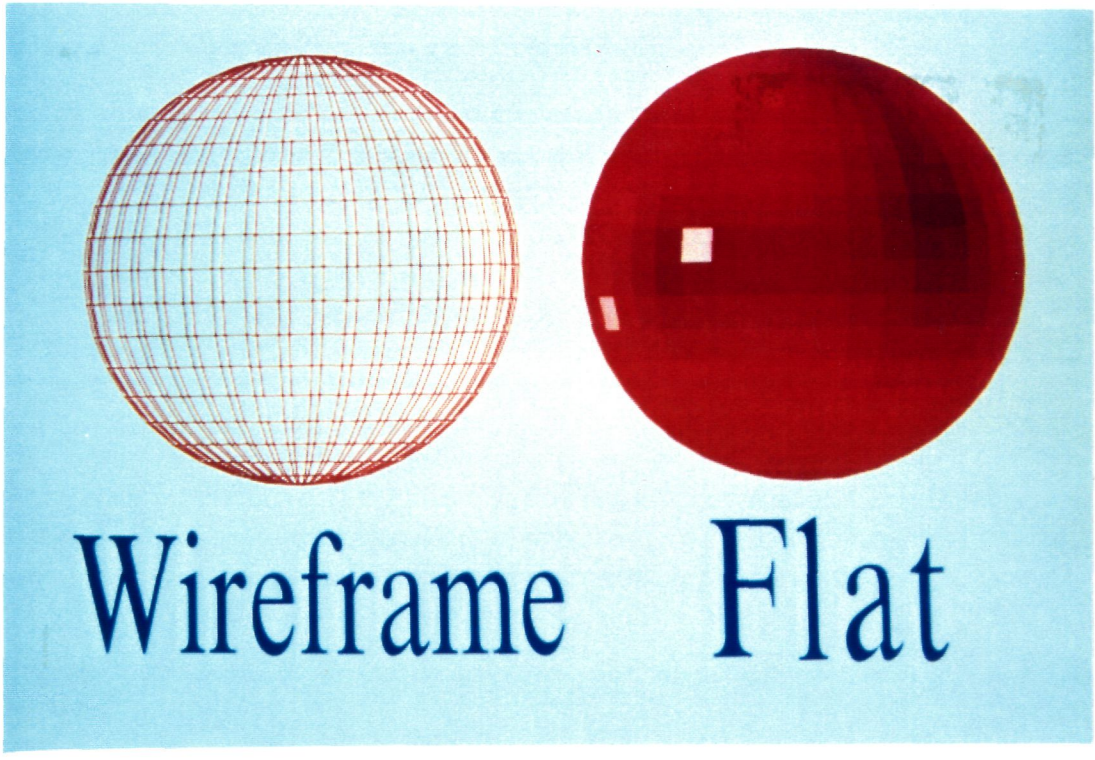
• *Shading Modes*. En este apartado, definiremos como van a reaccionar los objetos ante una fuente de luz. El *3D Studio* define cuatro formas diferentes: *Wireframe, goraud, flat* y *phong*.

El primero (Wireframe) es el más simple de todos, y el resultado que se obtiene, una vez realizado el *render* de la imagen, es el mismo que estamos viendo en el diseño: un objeto mallado.

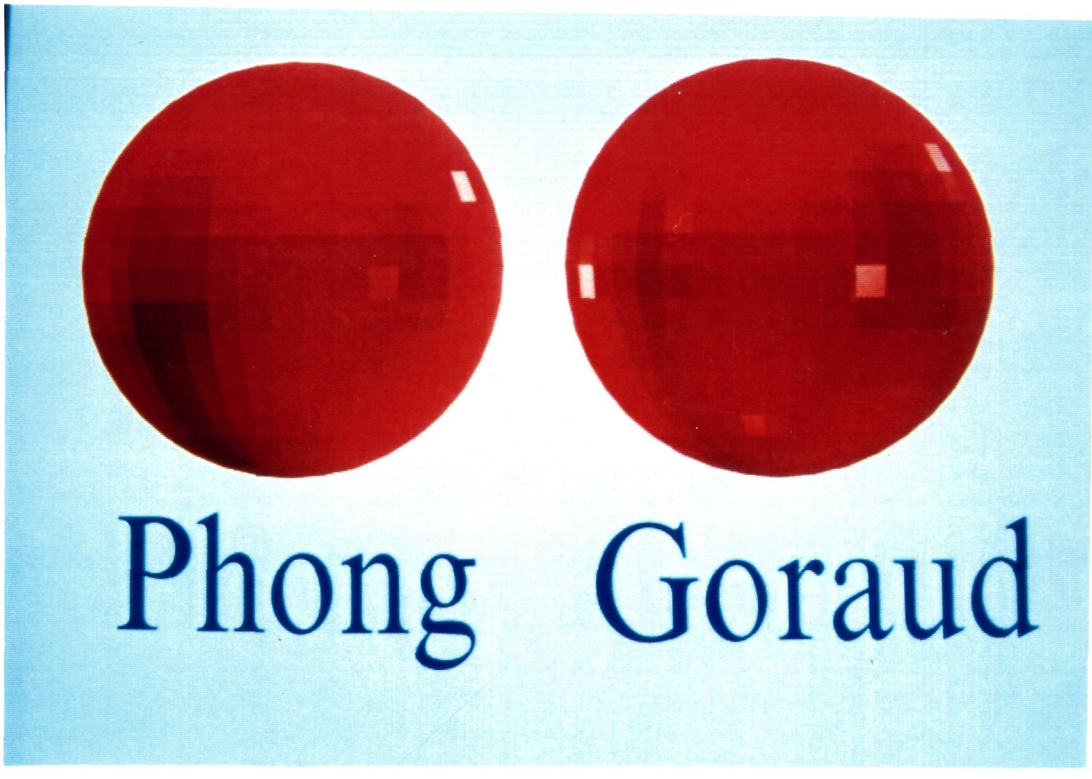
Con la opción *flat* lo que conseguimos, es ver al objeto como si estuviera formado por una superficie poligonal, es decir, con múltiples caras.

Con las otras dos opciones *phong* y *goraud* , se consigue un efecto más realista. *Goraud* realiza una interpolación del color a lo largo de todas las caras, basándose en el color de cada uno de los tres vértices. La opción *Phong* realiza un paso más, calculando también las normales.

El *materials editor* permite generar colores y materiales con una serie de herramientas que comentaré posteriormente. Estos colores pueden ser salvados en lo que se llama "Librería de materiales". Estas librerías están formadas por un conjunto de materiales que ya han sido diseñados y que están listos para ser utilizados. Podemos utilizar tantas librerías como deseemos.



Ilust. 17. *Ejemplo de Wireframe y Flat*



Ilust. 18. *Ejemplo de Phong y Goraud.*

Las herramientas que disponemos para la generación de materiales son:

• *Color Buttons and Swatches.* Está formado por tres botones que definirán como va a afectar un color a un objeto. Los botones son:

- *Ambient.* Controla la porción del objeto que no está sometida a luz directa.

- *Diffuse.* Controla el color de la porción del objeto iluminado.

- *Specular.* Controla el color en la parte que brilla del objeto.

• *Color Sliders.* Son barras de desplazamiento que se encargan de definir la mezcla de colores para la obtención de un color final. Está formado por dos tipos de barras que ya mencionamos antes: *RGB Sliders* y *HLS Sliders.*

• *Shininess Slider.* Es una barra de desplazamiento que determinará el brillo que le vamos a aplicar a un objeto.

• *Transparency.* La transparencia de un objeto la podemos ajustar de varias maneras diferentes:

- *Transparency Slider.* Determinará el grado de transparencia de un objeto. En la escala de 0 a 100, 0 es opaco y 100 es totalmente transparente.

- *Sub and Add button.* Con esta opción seleccionaremos si el color del objeto transparente se le suma al color de fondo (*color background*) o se le resta.

Hay otras formas para provocar transparencias que comentaremos posteriormente.

• *Tile and Decal buttons.* Estas dos opciones determinarán si la imagen que usemos para el *mapping* se repetirá a lo largo de toda la geometría del objeto (*tile*) o no (*Decal*).

• *Mapping.* Anteriormente, definimos *mapping* como el proceso por el cual incrustábamos una imagen en un objeto. En este apartado hablaremos de los cuatro tipos de *mapping* que el programa nos permite.

- *Animated Map.* Consiste en utilizar como imagen de *mapping* un fichero *.fli o una secuencia de imágenes, para lo cual deberemos crear un fichero de extensión *.ifl que contendrá el conjunto de imágenes que queremos usar como animación.

- *Texture map.* Viene a definir lo que es el mapeado en general. Consiste en definir el color de superficie de un objeto con una imagen estática ya creada.

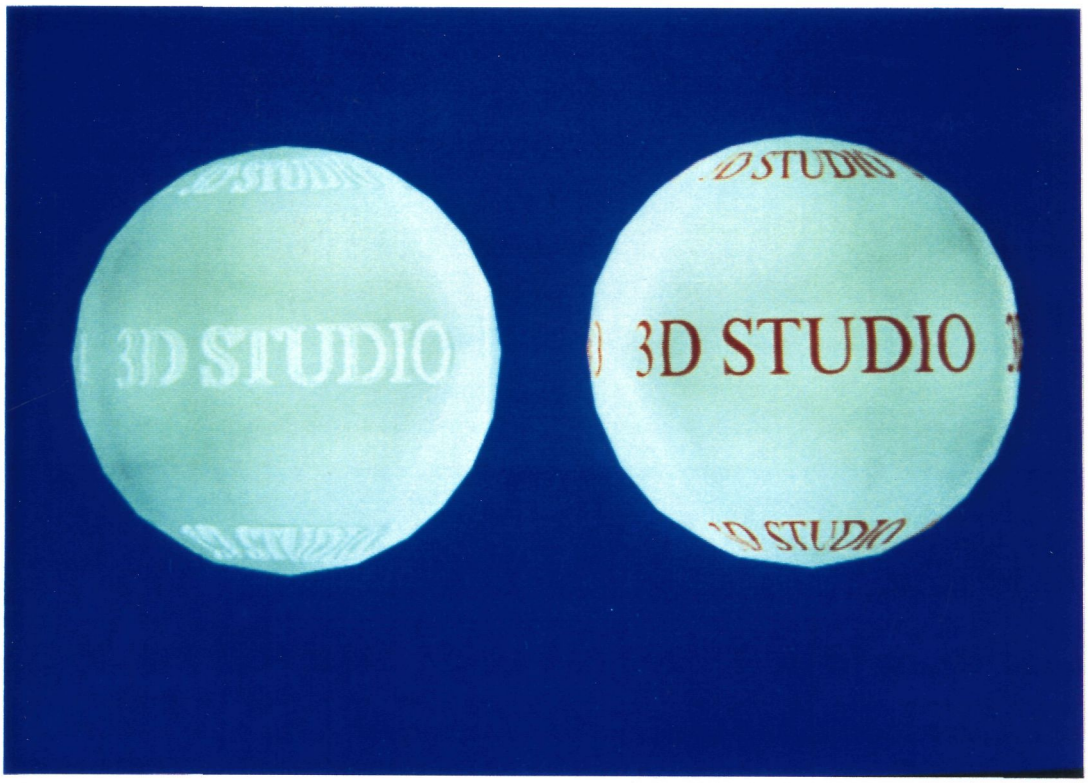
- *Opacity map.* Define, a través de una imagen estática, el grado de transparencia de un objeto. Podremos usarlo junto con la barra de desplazamiento de opacidad.

- *Reflection map.* Determinará, mediante una imagen estática, el grado de reflexión de un objeto. Tiene la posibilidad de realizar un reflexión automática, que consiste en utilizar, como imágenes reflectantes, los objetos que están alrededor de éste.

- *Bump map.* Altera las normales en todos los puntos de la superficie para conseguir un efecto de "inflado" en algunas partes del objeto o en todas. Su efecto se observa mejor en la ilustración siguiente.

- *Procedural Maps (SXP).* Se utiliza para realizar un mapeado. La diferencia con los anteriores es que, en vez de usar una imagen *bitmap*, usaremos un patrón sólido. Posee la ventaja de que no es necesario definir las coordenadas de mapeado, imprescindible para todo este tipo de objetos. Esto es muy importante,

sobre todo en los objetos obtenidos a partir de operaciones booleanas, ya que los tipos de mapeado que se permiten no se suelen adaptar bien a la geometría de los mismos.



Ilust. 19. *Ejemplo de Bump mapping.*

Una vez realizadas todas las elecciones con respecto al color y material a usar, deberemos hacer un *render* para ver una muestra en una esfera, o bien en un cubo, del resultado obtenido. Para conseguir el material deseado, deberemos ahora hacer modificaciones de todo lo anterior y continuos *render* hasta obtener el resultado deseado.

6.5. Keyframer.

Hemos llegado a un punto en el que ya tenemos diseñada toda la escena: hemos puesto los objetos, las luces, los materiales, etc.. Ahora, en este último módulo del programa, llega el movimiento. El efecto del movimiento no es más que una colección de imágenes estáticas que, al reproducirlas rápidamente, dan la sensación de

desplazamiento. A la imagen estática que forma parte de una colección, le denominaremos *frame* y es un conjunto de ellos, lo que denominaremos animación. La calidad de imagen de una animación se define en el número de *frames* por segundo. Así, se tiene:

Dibujos animados	Entre 12 y 24 frames/sg.
película	24 frames/sg
NTSC TV	30 frames/sg
PAL TV	25 frames/sg

Por tanto, a la hora de volcar a vídeo una animación, es necesario conocer el dispositivo de salida antes de aplicar la tabla anterior.

Objetos animables.

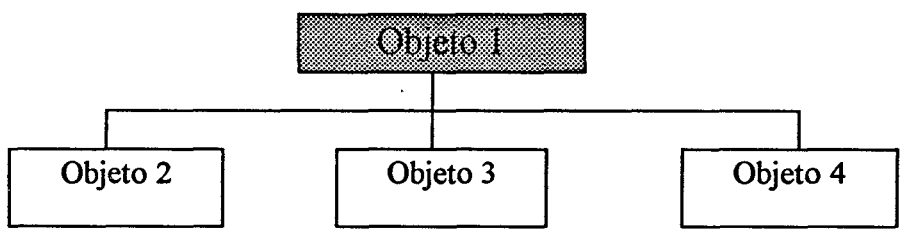
Dependiendo del tipo de objeto con el que tratemos, las transformaciones que podemos realizar serán diferentes. En la siguiente tabla se definen los tipos de objetos, y las alteraciones que podemos hacer:

Tipo de Objeto	Posibles Transformaciones
	Posición
Objetos mallados	Rotación
	Escalado
	Morphing (metamorfosis entre dos objetos)
Luces omnidireccionales	Posición
	Color
	Posición de la fuente
Spotlights	Posición del target.
	Color

Tipo de Objeto	Posibles Transformaciones
	HotSpot
	Falloff
	Posicion de la cámara
Camaras	Posición del target
	Roll
	Punto de vista

Estructuras jerárquicas.

Tomando elementos de la escena, podremos crear estructuras jerárquicas con el fin de asociar movimientos. Esto tiene gran importancia a la hora de animar objetos que pretendan simular articulaciones, o bien, objetos que tengan que moverse juntos.



La estructura en árbol, mostrada en la figura anterior, puede ser tan grande como se quiera. Si realizamos una transformación, según la tabla anterior, con el objeto 1, se aplicarán automáticamente las mismas transformaciones a todos los objetos dependientes de él.

Track Info Dialog Box.

Esta caja de dialogo es la que nos va a permitir controlar en todo momento, la secuencia de la animación, pudiendo controlar todos los movimientos de los objetos en escena.

Cuando deseamos mover un objeto, basta con situarnos en el *frame* que queramos, y definir la posición que tendrá el objeto en ese nuevo instante de tiempo. El programa generará con esa información la secuencia de movimiento desde la posición inicial hasta la final. Una vez realizada esta operación, aparecerá en el *Track info dialog box* una barra de desplazamiento, cuya longitud será igual al numero de *frames* de la escena y en el instante final del movimiento aparecerá un punto indicando que en ese instante el objeto termina un movimiento.

Toda esa información nos valdrá para ajustar los valores de posición, rotación, etc.. que tendrán los elementos. Si por un casual, al ejecutar la escena, observamos que no se ajustan los instantes de tiempo de los objetos tal y como habíamos diseñado, no tendremos que colocar el objeto de nuevo en su posición, sino que iremos al *Track Info dialog Box* y modificaremos su posición en el instante deseado.

Por tanto, este cuadro nos va a permitir tener un control total sobre todos los elementos de la escena sin necesidad de retoques.

A continuación comentaré los tipos de movimientos que podrán ser utilizados, pero sin entrar en detalles del tipo de objeto al que puede ser aplicado.

Tipos de movimiento.

• *Move*. Para aplicar un movimiento a un objeto, bastará con posicionar el objeto en su posición inicial (*Frame* inicial), ir al frame final, y definir nuevamente la posición. La secuencia será generada automáticamente. Las posiciones vendrán definidas en sus tres ejes (X,Y,Z). Pulsando la tecla *Shift* crearemos un nuevo objeto, igual que el anterior, pero con la denominación *Copy.ObjectX.*

• *Rotate*. Cambia la rotación de un objeto a partir de la definición de un eje de rotación (*Pivot Point*). Al igual que antes, habrá que indicar en el *frame* destino, el valor de la

rotación en grados. Puede indicarse sobre que eje de rotación trabajaremos: X, Y, Z, o completa. Si no indicamos cual será el punto de pivote, el programa seleccionara el que tenga por defecto en ese momento, con lo cual, puede que no se adapte a nuestras necesidades. Por ejemplo supongamos la rotación de una puerta. Si no especificamos la posición del punto de rotación, puede que el movimiento resultante no sea de nuestro agrado.

• *Scale*. Escala en el tiempo el tamaño de un objeto. Su funcionamiento y especificaciones son análogas a las funciones anteriores.

• *Morph*. Realiza una transformación (metamorfosis) entre dos objetos. Para ello, los objetos deben cumplir una serie de propiedades comunes, sin las cuales, no se podría aplicar este procedimiento. Para aplicar este procedimiento deberemos ir obligatoriamente a un *frame* distinto del primero y, a continuación, aplicar el método. Es condición indispensable que los dos objetos (fuente y destino) tengan exactamente el mismo numero de vértices. De todas formas, cuando apliquemos el método, aparecerá una ventana para indicarnos los objetos con los que se puede hacer la transformación.

Hay que tener en cuenta que el tipo de metamorfosis que se realiza es exclusivamente con respecto a la geometría y no al color o materiales que tengan aplicados los objetos.

Todos los movimientos comentados anteriormente y algunos más, a los cuales no hemos hecho referencia, pueden ser aplicados conjuntamente, es decir, podemos aplicar varios movimientos simultáneamente.

Comandos de *Path.*

El *path* es la trayectoria que mantiene el objeto durante una secuencia de movimiento. Debido a la gran importancia que posee en este apartado, debemos tener

con él especial consideración.

Normalmente, cuando en el *Keyframer* animamos un objeto, estamos generando el *path* automáticamente, aunque puede ser generado de múltiples formas distintas. Utilizando la opción *Show Path*, podemos comprobar como es la secuencia de movimiento de un objeto, e incluso, podemos modificarla.

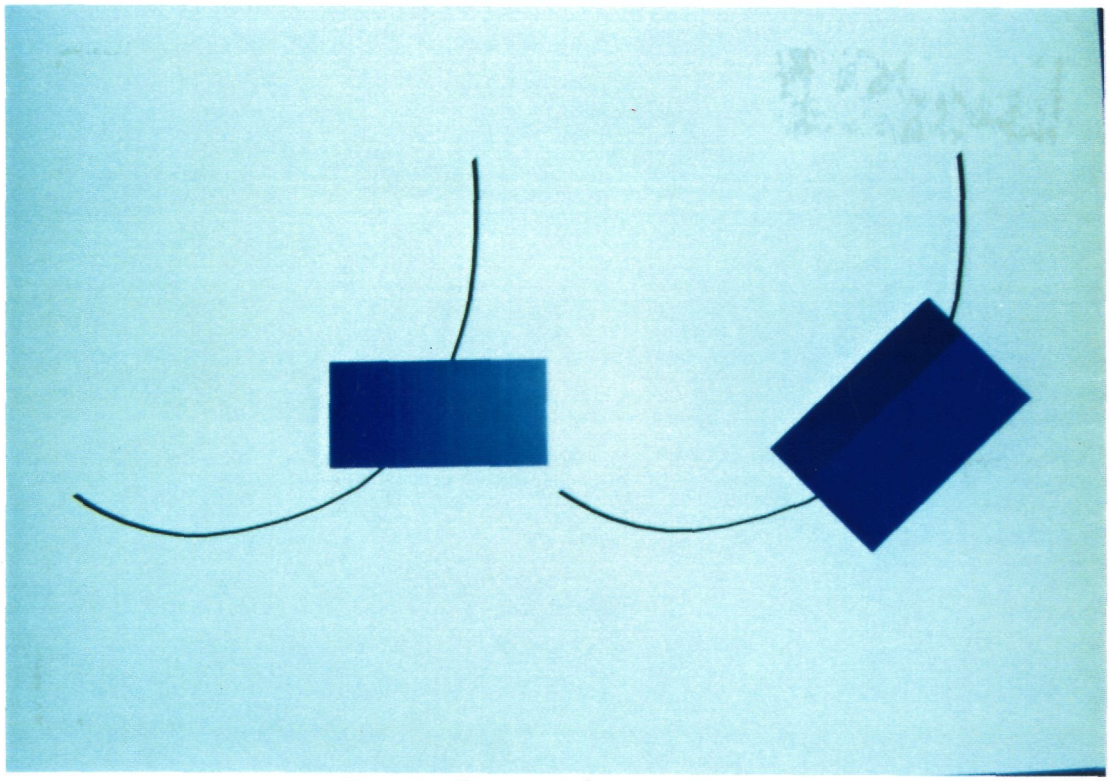
Si todavía no hemos animado un objeto, tenemos dos opciones para hacerlo: La primera consistirá en aplicar uno de los tipos de movimiento explicados anteriormente y, la segunda, es importar la trayectoria de movimiento que deseamos que tengan los objetos desde algún sitio. Podemos utilizar como trayectorias lo siguiente:

- Un polígono o línea abierta importada desde el *2D Shaper*.
- Un eje importado desde el *3D Lofter*.
- Cualquier trayectoria contenida en un fichero con extensión *.dxf o *.lft

Hay que tener en cuenta que las trayectorias cogidas de otra parte tiene que ser traducidas por el programa y que, por tanto, se convertirán los vértices de las líneas en instantes de tiempo (*Keys*) por el cual pasarán los objetos.

Otra opción muy importante dentro de los comandos relacionados con el *path* es la opción de seguimiento o de *Follow*. Su funcionamiento se analiza en la siguiente figura:

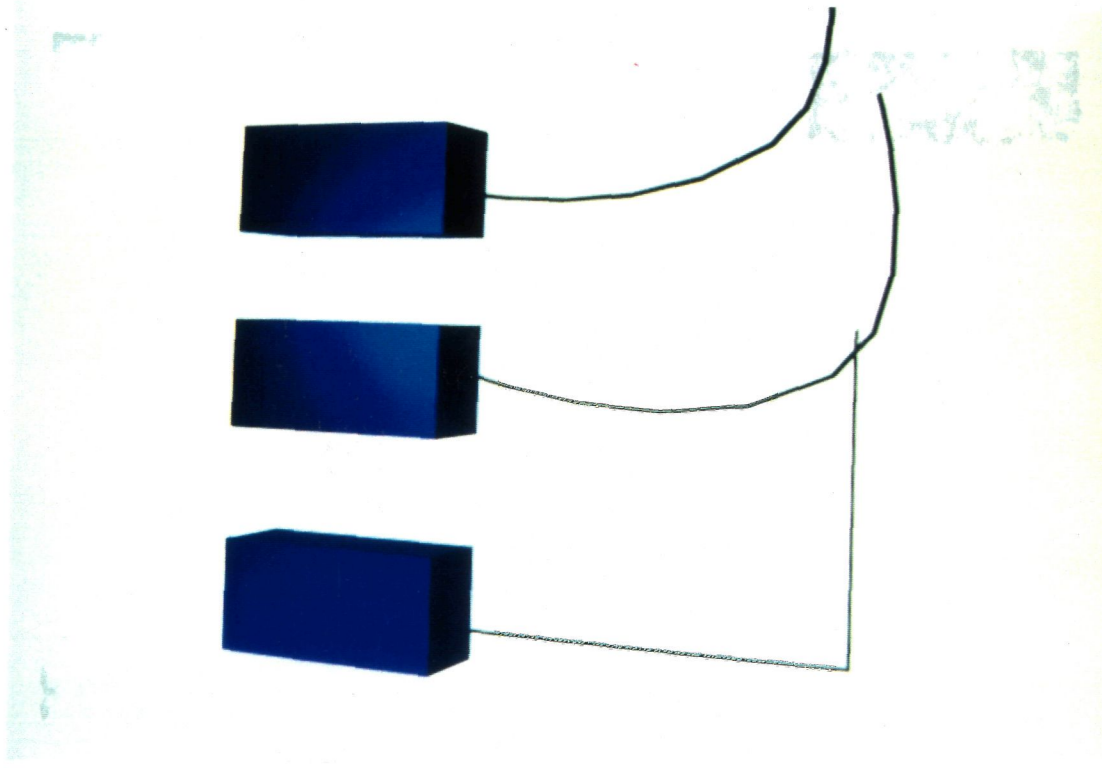
Se puede observar que, cuando activamos esta opción, el objeto no solamente sigue al *path*, sino que realiza pequeñas rotaciones para ajustar perfectamente los movimientos del objeto con la trayectoria.



Ilust. 20. *Ejemplo de la opción Follow.*

Existen otras muchas herramientas referentes a las trayectorias de los objetos, las cuales comentaremos más brevemente:

- *Add Key*. Define una nueva posición en la trayectoria de un objeto.
- *Move Key*. Traslada en tiempo una posición.
- *Delete Key*. Elimina una posición, uniendo el punto anterior con el siguiente.
- *Adjust Continuity*. Ajusta la continuidad de la trayectoria, ya que pueden existir trayectorias con las mismas posiciones pero con distintas formas.



Ilust. 21. *Ejemplo de Continuidad.*

Render.

Ya hemos concluido la animación (¡ Por fin !). Ahora el siguiente paso es realizar el proceso final: el *render*. Con él, obtendremos el resultado final que tanto habiamos esperado. Para poder realizarlo perfectamente, es necesario conocer unos parámetros que estarán ligados directamente con la calidad del resultado final.

• *Shading Limit*. Se seleccionará la técnica que se empleará para el render. Podremos elegir entre cuatro modalidades diferentes:

- *Wire*: realiza un render de los bordes. Este tipo de *shading* se realiza para ver de una forma rápida la geometría de los objetos.

- *Flat*: Cada cara es mostrada como única, con un color sólido. El color es calculado usando las propiedades de la superficie, su localización y su posición relativa respecto a la fuente de luz.

- *Goraud*: Da a cada cara un aspecto de gradiente. Los objetos tendrán la apariencia de una superficie suavizada

- *Phong*: Calculará el color de cada pixel en cada cara.

• *Anti-aliasing*. El programa dará a elegir el nivel de *anti-aliasing* de entre cuatro opciones: Ninguno, bajo, medio y alto. Con la opción de alto, el calculo del *render* durará menos tiempo.

• *Shadows*. Estará a ON/OFF dependiendo de si queremos que se realice el cálculo de sombras entre objetos. Esta opción estará puesta a ON cuando calculemos la animación final, ya que ralentiza el tiempo de *render*.

• *Force 2-sided*. Si está a ON, realizará el *render* de las dos partes de una cara. En caso contrario, solo lo realizará de las partes que estén mas cerca del punto de vista.

• *Mapping*. Permite desactivar el mapeado, para una mayor velocidad de calculo. Esta opción no afecta al *automatic reflection map*.

• *Auto Reflect Maps*. Es similar a la anterior. Se utiliza para acelerar el tiempo de calculo del *render*.

• *Hidden geometry*. Da la opción de calcular o no los elementos que estarán ocultos en la escena. Por tanto, su posición por defecto será *hide*.

• *Background.* Cuando utilizamos una imagen de fondo, ésta puede que no tenga la misma resolución que el render que vamos a realizar. Por tanto, aquí podremos elegir entre reescalar el fondo, en caso de que las resoluciones fuesen diferentes, o bien mantenerlo como está.

• *Frames.* Aquí seleccionaremos el número de *frames* de los cuales haremos el *render.* Podremos elegir entre cuatro opciones diferentes: Un único *frame*, un conjunto de ellos, todos, y por último, un segmento de tiempo.

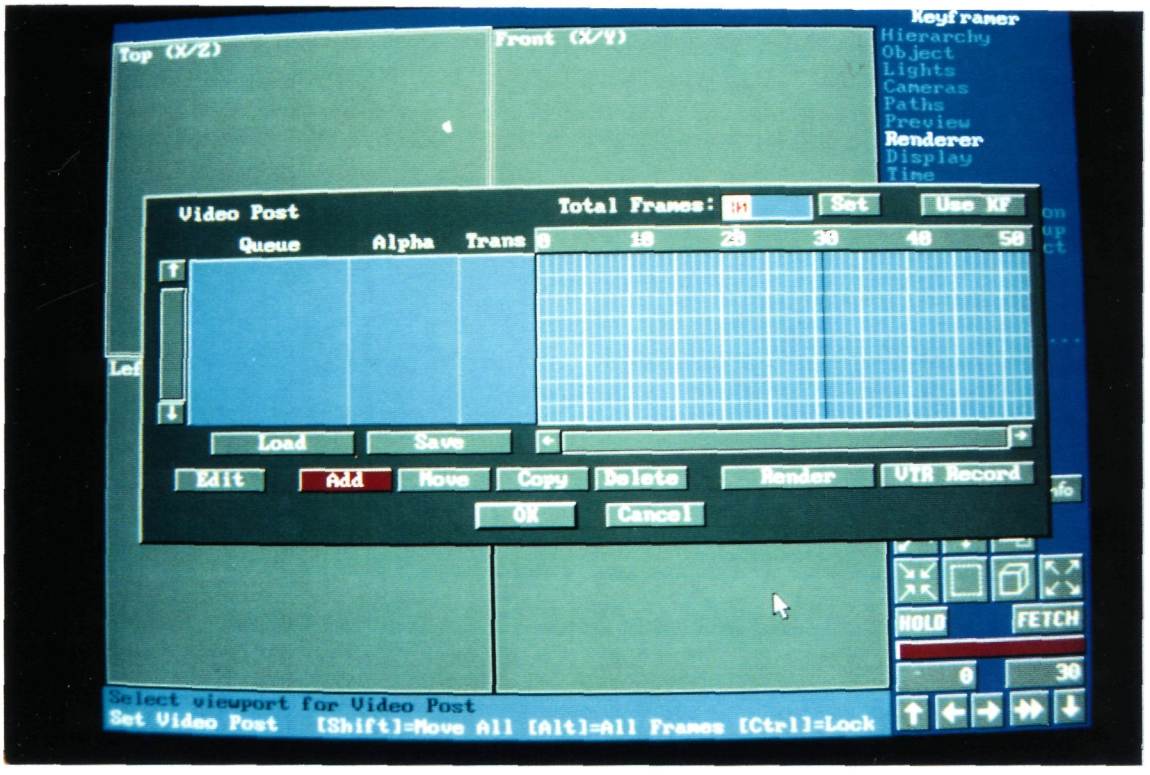
Render. Video Post.

El último apartado que estudiaremos en el *3D Studio* será el de *Video Post*, debido a la importancia que tiene la postproducción en un centro de producción de televisión. No será un tema de especial relevancia ya que para el tema de postproducción utilizaremos un software diferente.

Llegados a este punto podremos trabajar con animaciones que ya estén terminadas, con imágenes estáticas, con la escena actual, etc.. y generar varios efectos de transición entre ellos.

En primer lugar, debemos definir cuales son las escenas que vamos a utilizar. Estas escenas estarán definidas en lo que se denomina *Queue Column*, que contendrán los nombres de las imágenes utilizadas. En segundo lugar, debemos indicar el periodo de duración de dichas imágenes, lo cual haremos en lo que se denomina como *Frames Grid.* Una vez establecido lo anterior, hay que determinar como será el paso entre las imágenes, tal y como se realizaría en un estudio normal. En este sentido el programa viene muy limitado, ya que solo permite unos efectos muy básicos.

Este será el procedimiento general de realizar la postproducción y ahora solo nos faltaría realizar el render. Una posibilidad que incorpora el programa y que no usaremos es la de volcar a video todo lo anterior directamente. No se usará porque para ello es necesario una tarjeta compatible con RS-422 y que en este momento no está disponible en el laboratorio.



Ilust. 22. *Pantalla del Video-Post.*

7. *Software* de producción digital.

Estamos terminando ya nuestra tarea, y por ello, debemos hacer un pequeño resumen de lo que llevamos hecho hasta ahora. Con los programas disponibles, hemos diseñado nuestras animaciones, con el único fin de sacarlas a vídeo. La misión que debemos cumplir ahora es realizar la postproducción de estas animaciones. Para realizar este trabajo podemos proceder de dos maneras diferentes: La primera consistirá en volcar a vídeo todas las animaciones, para una vez allí, montarlas tal y como se haría en la edición de un programa normal. La otra opción consistirá en utilizar algún software especifico que nos permita realizar esta tarea desde un PC.

Este capítulo estará destinado a explicar la segunda opción. Si hemos llegado hasta aquí sin utilizar ningún tipo de máquina que no sea un ordenador, creo que debemos tratar de terminar este trabajo utilizando la misma plataforma.

Para ello utilizaremos un *software* de postproducción que nos permitirá utilizar un ordenador como una pequeña y simple mesa editora. En este punto es necesario comentar que, aunque para este trabajo utilizaremos un programa relativamente sencillo, existen en el mercado programas excelentes destinados a la misma tarea.

El programa que utilizaremos es el denominado *VideoMaker+*, que posee, en líneas generales, las siguientes características:

- Es capaz de trabajar tanto en PAL como en NTSC con las tarjetas digitalizadoras TARGA+16, TARGA+16/32 y TARGA+64.

- Permite mezclar y hacer transiciones de cortina entre: vivo/gráfico, gráfico/vivo y gráfico/gráfico.

- Permite determinados efectos especiales tales como *blurs*, *strobes*, *feedback*, etc..

- Permite realizar un *key* de azul y un *key* de luminancia.

- Permite la generación automática de secuencias usando imágenes y transiciones.

Todas estas características serán comentadas posteriormente junto con un pequeño análisis de cada una de ellas.

7.1 Requerimientos del programa.

Este programa necesita para su correcto funcionamiento de unos requisitos mínimos tanto en *hardware* como en *software*. Éstos serán comentados a continuación:

Requisitos *hardware*.

- Cualquier modelo de tarjeta Targa+.

- Dos o más monitores. Uno de ellos será el del ordenador, para tener control sobre el programa, y otro será utilizado para ir viendo las secuencias que iremos generando, es decir, la salida de la Targa+.

- Un ordenador con Coprocesador matemático y al menos 640K de RAM.

- Un dispositivo de apuntamiento, tal como un ratón o una tableta gráfica.

- Opcionalmente también puede ser conectado un VCR para la grabación de todas las secuencias que generemos.

Requisitos *software*.

El único requisito que se debe cumplir para que el programa funcione es tener el software de la Targa+ instalado en el modo apropiado, ya que el programa solo funciona con determinados modos de vídeo. Los modos aceptados son:

512x400 NTSC	16 bpp	TMODE 1
512x476 PAL	16 bpp	TMODE 5
512x486 NTSC	16 bpp	TMODE 9
512x576 PAL	16 bpp	TMODE 13
756x486	16 bpp	TMODE 25
768x576	16 bpp	TMODE 27

Modos utilizados por VideoMaker+.

7.2 VideoMaker+.

En líneas generales, el propósito del programa es crear una serie de imágenes de vídeo con efectos especiales que puedan ser almacenadas en secuencias para una posterior ejecución. Esta ejecución podrá utilizarse para ver la secuencia que hemos generado y para volcarla a vídeo.

Para poder llevar a cabo todo lo comentado en el párrafo anterior, el programa ha sido diseñado formando paneles desde los cuales se podrá tener un control completo del programa.

Los paneles que lo conforman son:

- *Setup panel.*
- *Preview panel.*
- *Switcher panel.*
- *Transition/wipes panel.*
- *Control panel.*
- *Solarization panel.*
- *Keyer panel.*
- *Sequencer panel.*

En los siguientes párrafos, se comentarán todas y cada una de las opciones que incorporan los paneles.

7.2.1 Setup panel.

Este panel nos permitirá establecer los parámetros que definirán el modo de trabajo del programa. Este panel dispone de ocho áreas, que son: *grab*, *input*, *blur*, *feedback*, *live border*, *video levels*, *strobe* y *background*. Cada una de estas áreas se utilizan para especificar parámetros diferentes.

• Grab area. Cuando desde el panel principal seleccionamos la opción *grab*, le indicamos al programa que digitalice una señal de vídeo y la coloque en cualquiera de los dos *buffers* (G1 y G2). La forma en que se realiza esta digitalización depende del valor que tengamos definido en este área.

Podremos establecer tres valores: *frame*, *field1* y *field2*. Si ponemos por defecto la primera opción, cada vez que capturemos una imagen, capturaremos un frame completo de vídeo. Seleccionando cualquiera de las otras dos opciones, solo capturaremos las líneas pares o las impares dependiendo de cual tengamos seleccionado.

• Input area. Este parámetro especificará el tipo de fuente de vídeo que estará conectado a la entrada de la Targa+. Podremos elegir entre RGB, Compuesto y S-Vídeo.

• Blur area. Cuando realizamos un *blur*, cuyo efecto se explicará posteriormente, el programa realiza la transición hasta concluir en uno de los dos *buffers* posibles (G1 ó G2). Es, en este bloque, donde indicaremos en cual de ellos debe concluir. Solo uno de ellos puede ser seleccionado.

• Feedback area. Al igual que antes, este opción permite elegir que *buffer* se utilizará como final cuando se produzca este efecto. Solo uno de ellos puede ser seleccionado.

• Live border area. Cuando volcamos una secuencia de imágenes cuya resolución es menor que la que permite el monitor de visualización, podremos elegir entre adaptar los bordes con las imágenes en vivo que tenemos, o bien, dejarlo como aparezca. Esta elección se escoge en este menú, cuyos valores serán ON u OFF.

• Video levels area. Aquí controlaremos los niveles (H,S,C) de la imagen. Estos valores afectarán a todas la secuencias que generemos.

• Strobe area. Al igual que en anteriores ocasiones, esta opción nos permitirá elegir en qué *buffer* terminará cuando se ejecute este efecto. Aquí aparece una nueva opción, que es la barra de desplazamiento de *strobe* (*Strobe slider bar*) que indicará el número de *frames* por segundo que se visualizarán cuando se genere este efecto.

• Background area. Aquí estableceremos el color de fondo que queremos usar. Este valor se indica mediante tres barras de desplazamiento: *Red*, *Green* y *Blue*.

7.2.2 Preview panel.

Este panel nos permitirá observar las imágenes que han sido grabadas en los *buffer* de la Targa+. Cuando cargamos una imagen, ésta se depositará en uno de los dos posibles (G1 ó G2), y se podrán visualizar en cualquiera de las dos áreas dispuestas para tal efecto. Hay que notar que cuando grabamos una imagen, ésta solo aparecerá en su área cuando esté completamente volcada en la memoria de la tarjeta.

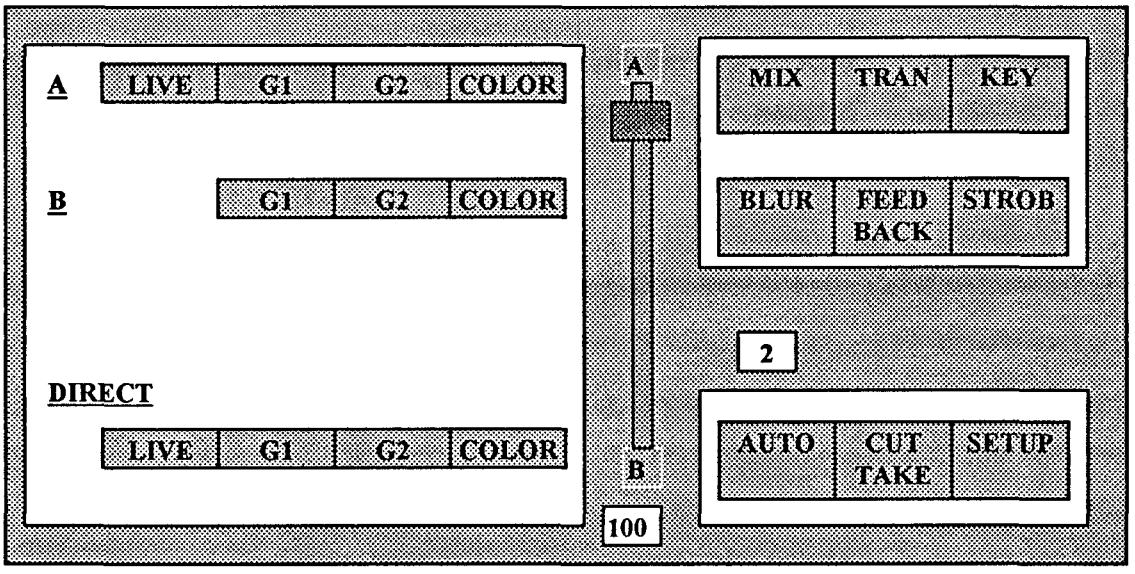
Para poder grabar una imagen en cualquiera de las dos áreas posibles, podemos proceder de dos maneras diferentes: La primera consistirá en cargar una imagen de disco indicando en que *buffer* se depositará, y la segunda, consistirá en capturar una imagen de la fuente de vídeo que tengamos conectada. Para ello tenemos que tener en el *Switcher area* del panel principal, la opción *direct* habilitada.

En este panel existe otro botón, denominado *field copy* que sirve para lo siguiente: Cuando con el botón *grab* capturamos una imagen muy rápida, como por ejemplo el movimiento de un coche, etc.., observaremos que puede aparecernos un parpadeo como consecuencia de la captura (el mismo efecto que una fotografía movida). En este caso, pulsando el botón *field copy* podremos reducir este efecto, ya que este botón lo que hace es eliminar uno de los dos campos de la imagen.

7.2.3. Switcher panel.

Este panel es el más importante de todos los que conforman el programa. Se usa para cambiar de una fuente de vídeo a otra, mientras se aplican varios efectos especiales. Tiene tres funciones principales:

- Indicar que fuente de vídeo va a ser usada.
- Realizar la transición entre dos imágenes con el efecto elegido.
- Definir el tipo de efecto que se usará para la transición.



Aspecto del Switcher Panel.

Para usar este panel seleccionaremos una fuente de vídeo A, otra fuente B, un efecto de transición y con la barra de desplazamiento efectuaremos este efecto. Como acabamos de comentar, disponemos de dos posibles fuentes: A y B. En la primera de ellas (A) podremos seleccionar entre una fuente en vivo, cualquiera de los dos *buffer* de la tarjeta o un color de fondo. Sin embargo, en la segunda fuente solo podremos seleccionar las dos últimas opciones, es decir, no podremos seleccionar una fuente en vivo. Esto se debe a que la tarjeta solo incorpora una entrada y por tanto no podemos disponer de dos fuentes en vivo.

Barra de desplazamiento A/B.

La barra de desplazamiento se utiliza para cambiar entre la fuente seleccionada en A y la seleccionada en B. Ésta solo estará habilitada cuando se haya escogido una efecto que utilice las dos fuentes. En caso contrario estará deshabilitada.

Para mover la barra desde la posición inicial (A) hasta la final (B), podremos utilizar dos métodos. El primero de ellos consiste en realizar esta operación manualmente, para lo cual utilizaremos el ratón para moverla, y el segundo método

consiste en establecer un tiempo de transición en segundos y utilizar la opción de automático. En este caso, el programa realiza la transición por nosotros en el tiempo establecido.

Esta barra puede ser utilizada conjuntamente con otras opciones tal como la solarización. Esta opción será comentada posteriormente.

Efectos.

Podremos elegir entre seis opciones posibles para seleccionar el efecto deseado. Las opciones son Mix, Tran, Key, Blur, Feedback y Strob. A continuación se comentará brevemente cada una de ellas.

• MIX. Esta opción permite realizar un *fade* entre las fuentes A y B. Cuando se selecciona esta opción, todos los demás botones de este panel se desactivan, ya que no se puede conjugar con otros efectos. Si la solarización está activada, cuando se realice la mezcla también se incluirá. La mezcla se puede realizar manual o automáticamente tal y como se indicó anteriormente.

• TRAN. Realiza un efecto parecido al anterior, pero con la diferencia que para realizar una transición se van a utilizar cortinillas. Podremos seleccionar entre ocho tipos de cortinillas diferentes mostradas en el *transition/wipe area* de panel principal. Al igual que antes, la solarización es un efecto que puede ser usada conjuntamente con éste. La forma de ejecutar la transición es similar a la anterior.

• KEY. Cuando seleccionamos este efecto, el *keyer panel* es mostrado para elegir entre varias opciones. Este efecto se explicará en el apartado 7.2.7.

• BLUR. Este efecto solo puede usarse con la fuente de vídeo en vivo, es decir, no se puede usar con los *buffer* de la tarjeta. El resultado que se consigue al utilizar este efecto es mostrar las imágenes que se van reproduciendo en el vídeo como si estuvieran

borrosas. Esto se consigue capturando cuadros de vídeo continuamente y colocándolos en los *buffer* G1 y G2. A continuación se realiza una mezcla de estos cuadros para conseguir este resultado.

Cuando se selecciona esta opción, todos los demás botones de efectos son desactivados automáticamente, ya que no se permite la utilización conjunta de éste con otros. El único permitido es el de solarización, al igual que en anteriores opciones.

Este efecto puede ser salvado a disco, con lo cual podemos utilizarlo para generar una secuencia. Esta opción se comentará posteriormente.

Cuando usamos esta opción, los contenidos de las memorias son eliminados y, cuando terminamos este efecto, el contenido de los *buffers* es actualizado a las dos últimas imágenes que se hayan capturado. Es por ello, por lo que es importante almacenar las imágenes que tengamos antes de utilizar el *blur*.

• FEEDBACK. Éste, al igual que antes, solo puede ser usado con una fuente de vídeo y, cuando se activa, el resto de las opciones son desactivadas automáticamente, ya que no pueden ser usadas conjuntamente.

El resultado que produce es como una especie de estela que va dejando el objeto o los objetos en movimiento sobre un fondo azul. Por tanto es condición indispensable utilizar un fondo azul para usar esta opción.

Al igual que en el caso anterior, este efecto puede ser grabado a disco y también puede utilizarse conjuntamente con la solarización.

• STROB. Esta opción solamente funciona con una fuente de vídeo y, el resultado que produce son saltos de imagen. Para ello se suprimen cuadros de vídeo a intervalos regulares.

Esta opción elimina las imágenes que estén contenidas en los *buffers* y las sustituye por las que vaya capturando. Al terminar, en G1 y G2 quedarán los dos últimos cuadros capturados.

Su configuración puede ser grabada a disco y se puede utilizar conjuntamente con la solarización.

Un último aspecto a tener en cuenta es que en la opción *Strob setup*, se puede modificar el intervalo de captura de cuadros.

• CUT TAKE. Este es el último efecto que comentaremos. Es el más sencillo de todos y consiste únicamente en conmutar entre el contenido de G1 y el de G2.

7.2.4 Control panel.

Este panel permite salir del programa totalmente y temporalmente, así como cargar y almacenar imágenes, solarizaciones, etc.. Para ello dispone de cuatro botones, que se comentan a continuación.

• QUIT. Abandona el programa y vuelve al DOS. Hay que tener en cuenta que no nos va a preguntar si queremos salvar todas las opciones, así que deberemos hacerlas nosotros mismos antes de abandonar el programa.

• SHELL. Sale temporalmente del programa, aunque no definitivamente. Para volver solo es necesario realizar un EXIT.

• LOAD & SAVE. Estos dos últimos botones serán comentados conjuntamente ya que su metodología es exactamente igual.

Se manejan cuatro tipos de fichero: imágenes, solarizaciones, secuencias y parámetros. A continuación se explican cada uno de ellos.

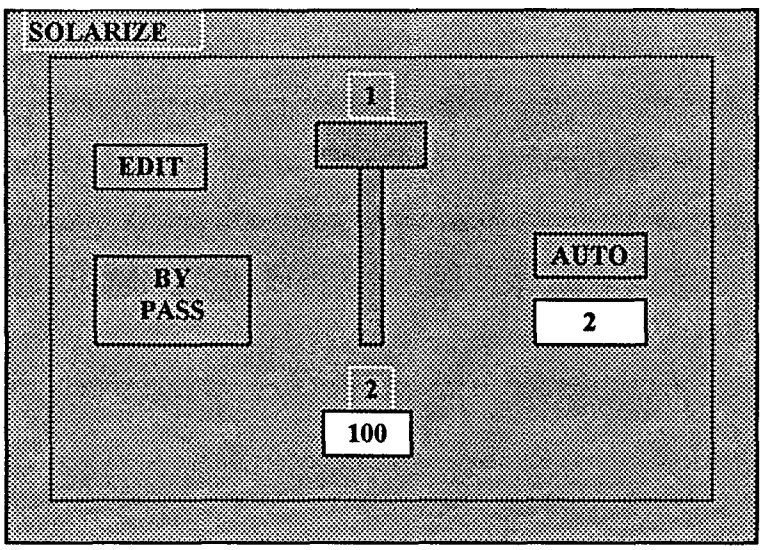
- Imágenes (TGA). El tipo de imágenes que se pueden cargar/guardar son exclusivamente TGA. Esto se debe a que es el formato utilizado por la Targa+. Podremos utilizar cualquier imagen siempre que esté en este formato. Cuando deseamos cargar una, no solo es necesario indicar el nombre del fichero sino también hay que decirle en que *buffer* de los dos posibles queremos colocarla.

- Solarizaciones (SOL). Cuando realizamos una solarización, los parámetros establecidos pueden ser guardados en el disco para podérselos aplicar posteriormente a otras secuencias de imágenes. Para ello se utilizan los ficheros de solarización, que tendrán extensión *.SOL,

- Parámetros de memoria (MEM). Cuando realizamos algún efecto tal como el *blur*, *Key*, etc., podemos a continuación guardarlos a disco para su posterior utilización. Todos los parámetros de los efectos son almacenados en ficheros de extensión *.MEM.

- Secuencias (SEQ). Este es el tipo de ficheros que más importancia cobra en este programa. El *software* nos permite realizar secuencias de transiciones con la inclusión de efectos, para poderlas ejecutar automáticamente posteriormente. Estas secuencias son almacenadas en ficheros *.SEQ.

7.2.5 Solarize panel.

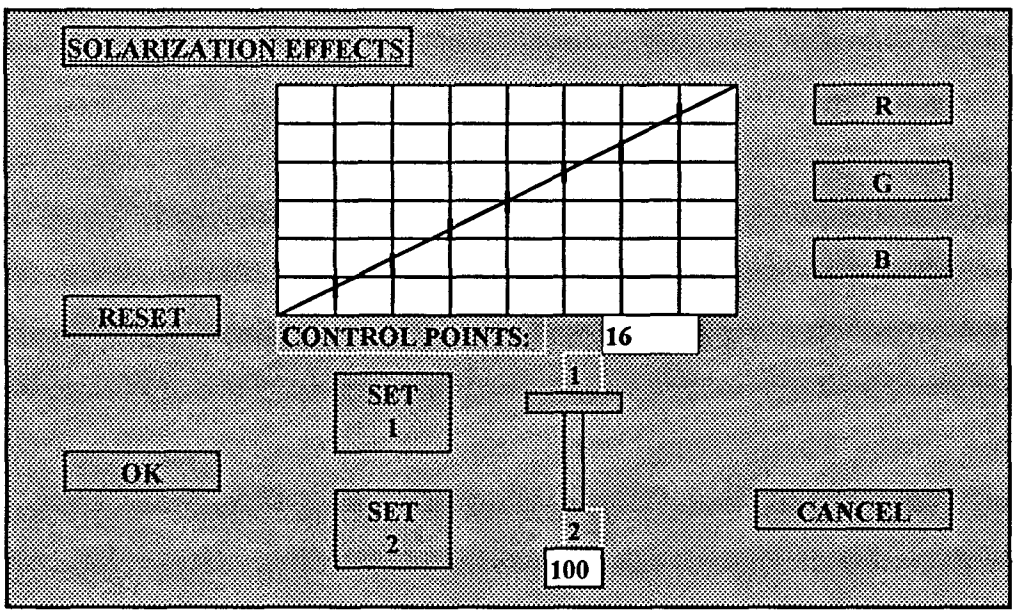


Aspecto del panel de solarización.

Solarización significa la sustitución de unos colores por otros, siguiendo unas reglas determinadas. Cuando activamos este efecto, toda la paleta de colores es sustituida por otra según nuestras preferencias.

Cuando seleccionamos la opción *edit* del panel principal, accedemos al panel de edición, el cual nos permitirá realizar todas las modificaciones de la paleta de colores que creamos oportunas. La forma de hacerlas, será explicada posteriormente.

Con la opción *By Pass*, conseguiremos activar o desactivar este efecto. Cuando el botón está habilitado, los parámetros establecidos en el menú de edición no serán tenidos en cuenta.



Subpanel de solarización.

El resto de las opciones se utilizan de la misma forma que en el *Switcher panel*, por lo que no serán comentadas.

Cuando accedemos al subpanel de solarización, a través de la opción *edit*, se nos muestra una ventana similar a la ilustración anterior. En ella, podremos modificar todos los parámetros necesarios para cambiar la paleta de colores. Para aclarar el funcionamiento comentaremos independientemente cada opción.

• RESET. Cuando lo seleccionamos, todos los parámetros de colores que hayan sido modificados previamente, serán puestos otra vez a su valor inicial.

• PANEL GRÁFICO. Este será nuestro lugar de trabajo para realizar los efectos de solarización. El eje X representa una tabla de 256 colores y el eje Y representa el valor de cada uno de ellos en una escala de 0 a 255. Con la línea que aparece en el gráfico podremos cambiar los valores simplemente pinchando con el ratón en el punto deseado. Desplazando verticalmente cada uno de estos puntos se formará una curva que nos

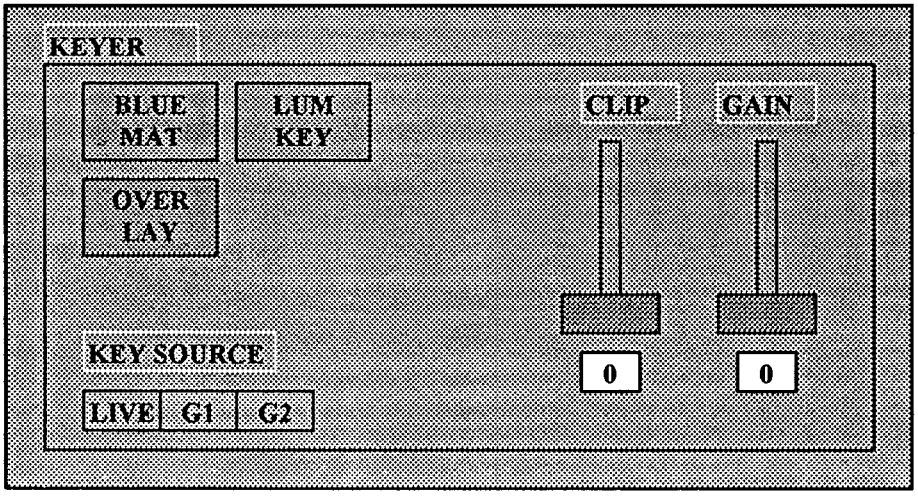
indicará los valores que tendrán los nuevos colores. Para un control más exacto de la paleta podremos cambiar el número de divisiones de la línea. Esto se realiza en la opción de "Puntos de Control".

• SET 1/SET 2. Si activamos la opción SET 1 y realizamos una solarización y después activamos SET 2 y repetimos el paso anterior, cuando ejecutemos la secuencia, podremos comprobar la transformación que se produce desde el gráfico del *buffer* 1 al gráfico del *buffer* 2.

• R,G,B. Estos botones pueden ser usados independiente o conjuntamente. Cuando seleccionamos alguno de ellos, la línea que nos aparece en el *graph panel* modificará solamente la componente que esté seleccionada.

7.2.6 Keyer panel.

Este panel define efectos de *Key* tales como un *Key* azul, un *Key* de luminancia o un *Overlay*. Un *Key* puede ser definido como la habilidad de seleccionar un color o gama de colores de un área de una imagen de vídeo y manipular esas áreas como si no existiera vídeo en ellas, con lo cual podremos hacer mezclas de imágenes. Este panel solo está habilitado cuando la opción *Key* del *Switcher panel* está activada.



Aspecto general del *Keyer Panel.*

Como desde este panel podemos utilizar tres efectos diferentes, comentaremos cada uno de ellos por separado.

7.2.6.1. Blue matte Key.

Este efecto, también definido como *Chroma Key*, se basa en superponer dos imágenes, de las cuales una de ellas tiene un fondo azul. El método consiste en detectar cada pixel de la pantalla y sustituir los azules por los pixels equivalentes de la otra imagen.

Cuando se selecciona este efecto, automáticamente las opciones *Lum Key* y *Overlay* son desactivadas, así como el *Key source* pasa automáticamente a *Live*.

En esta opción, la barra de desplazamiento denominada *Clip Slider* se utiliza para controlar el nivel de intensidad del azul. De la misma forma, la barra de desplazamiento de ganancia denominada *Gain Slider*, se utiliza para controlar el nivel de amplificación de la señal.

7.2.6.2 Luminance Key.

Este procedimiento es similar al anterior, salvo que antes eliminábamos un pixel por su característica de color y ahora es eliminado por su característica de luminancia. Para ello será necesario definir un valor de luminancia, el cual puede ser elegido usando dos métodos diferentes. El primero de ellos se realiza utilizando la opción *Set Lum*, la cual nos permitirá seleccionar un pixel en la imagen que se está mostrando en el monitor que esté conectado a la salida de la Targa+. El segundo método consiste en realizar la misma operación que antes, excepto que ahora pulsaremos directamente en el color y no en el botón.

Una vez establecido el valor, el efecto se hará visible tan pronto como se haya desactivado el ratón del monitor de visualización de la Targa+.

Hay que tener en cuenta que cuando utilizamos este efecto, solo podremos utilizar una fuente de vivo y un único *buffer* (G1 ó G2). Esto se debe a que el no utilizado, es empleado por el programa para guardar el valor de luminancia seleccionado. Por lo tanto si tenemos una imagen en el buffer que no vamos a emplear, es necesario guardarla, ya que en caso de no hacerlo, se perderá al utilizar esta opción.

7.2.6.3 Overlay.

El efecto *Overlay Key* es un procedimiento que nos permite seleccionar un color específico de una imagen para ser utilizado como *key*, de tal forma que, cuando el *Overlay* es inicializado, todos los *pixels* que tengan ese color serán hechos transparentes para mezclar la imagen con otra contenida en el buffer apropiado.

El procedimiento que tendremos que seguir consistirá en la elección de dos colores. El primero de ellos será el que se convertirá a transparente, es decir, el que utilizaremos como *Key* y, el segundo, será uno de la otra imagen.

Una vez que se hayan seleccionado los colores adecuados, pulsaremos el botón *Set Over*, el cual activará el efecto con los valores que hayan sido seleccionados. Hay que tener en cuenta que si tenemos este botón activado antes de realizar la selección de colores, el efecto será reproducido automáticamente y los buffer serán actualizados a los nuevos valores.

Otra opción que se nos presenta es la posibilidad de realizar mezclas cambiando las imágenes, es decir, si hasta ahora hemos hecho un *Overlay* entre dos fuentes A y B, ahora podemos invertirla y hacerla entre B y A, con lo cual el efecto resultante será completamente diferente.

7.2.7 Sequencer panel.

Este es último panel del programa y quizás el más importante. Desde aquí se nos permitirá generar secuencias de efectos para, posteriormente, poderlas ejecutar sin retardos de tiempo y con los valores adecuados.

El panel se compone de una tabla que posee cuatro columnas, cada una de ellas usadas para definir un parámetro determinado. Los nombres de las columnas son:

• ITEM. El contenido de este campo será siempre un número entero que nos indicará el orden de aparición del evento dentro de la secuencia.

• EVENT. Aquí aparecerá la acción a ejecutar. Ésta puede ser de diferentes tipos y, por tanto, se comentan a continuación:

• *Image*. Permitirá cargar una imagen TGA en el buffer indicado.

• *Solarize*. Carga un fichero *.SOL que contendrá parámetros de solarización.

• *Memory Setting*. Carga un grupo de efectos almacenados en ficheros del tipo *.MEM

• *Command*. Nos permite incluir una serie de comandos con el fin de controlar perfectamente la ejecución del fichero. Los comandos disponibles son:

1. PAUSE: Realiza una pausa hasta que se presione el botón del ratón.
2. MANUAL: Para temporalmente la ejecución de la secuencia hasta que una serie de acciones son ejecutadas.
3. LOOP: Hace que la secuencia vuelva otra vez a su inicio (Ítem nº 1).
4. GRAB 1: Realiza una digitalización de la fuente de vivo y el resultado es colocado en el *buffer* 1.

4. GRAB 1: Realiza una digitalización de la fuente de vivo y el resultado es colocado en el *buffer* 1.
5. GRAB 2: Realiza la misma acción que la anterior pero con el *buffer* 2.

Utilizando todos estos comandos, podremos generar una secuencia que se adapte a nuestros gustos con el único fin de depositar posteriormente nuestras animaciones a vídeo, objetivo que estabamos persiguiendo desde el inicio de esta memoria.

APÉNDICE A. El formato FLI.

El fin de todas las animaciones que realizamos es volcarlas a vídeo para usarlas en cualquier programa. Para llevar a cabo este proceso, el *software* de animación genera un fichero cuya extensión es *.FLI. Debido al uso continuado que haremos de estos ficheros, es necesario comprender su formato para de esta manera, poder generar aplicaciones de *software* que puedan trabajar con este tipo de información.

Lo primero que vamos a analizar es exactamente el formato que tiene el fichero, para posteriormente, pasar a comentar unas librerías que nos permitirán ejecutar este tipo de ficheros.

Unsigned long FileSize	--------	Longitud el fichero
Unsigned short Magic	--------	Numero mágico
Unsigned short FileFrames	--------	Numero de frames
Unsigned short FileWidth	--------	Anchura en pixels
Unsigned short FileHeight	--------	Altura en pixels
Unsigned short FileDepth	--------	bits por pixels
Unsigned short FileSpeed	--------	Velocidad reproducción
Unsigned short FileNext	--------	Vale 0
Unsigned short FileFrit	--------	Vale 0
Unsigned char filler[102]	--------	Bytes sin usar

Tabla 1. Formato del fichero FLI.

Lo primero que aparece en un fichero FLI es la cabecera con una longitud de 128 bytes. Dicha cabecera contiene mucha información, como por ejemplo, el número total de *frames*, el tamaño en pixels de los cuadros o la velocidad de reproducción.

Dentro de esta cabecera, aparece lo que se denomina número mágico, que está en la posición cuatro. Este número es un dato de 16 bits, que se utiliza para averiguar si el fichero que estamos leyendo es un fichero con formato FLI. Su valor en hexadecimal es AF11. Si este campo no contiene exactamente este valor, no podremos asegurar que se trate de este tipo de ficheros y, por tanto, es preferible no intentar leerlo.

A continuación de esta cabecera, se encuentra la información de la animación propiamente dicha. Ésta viene comprimida para cada *frame* o cuadro, por lo que cada cuadro dispone de una cabecera de 16 bytes (Ver tabla 2). Esta segunda cabecera nos proporciona un nivel adicional de control sobre la adaptación del fichero que estamos tratando, ya que dispone de un segundo número mágico cuyo valor en hexadecimal debe ser F1FA. Además de esta información, la cabecera nos proporciona información sobre el número de bytes que ocupa cada frame y del número de bloques que componen este cuadro.

Unsigned long framesize	--------	Longitud del cuadro
unsigned short FrMagic	--------	Número mágico
unsigned short NChunks	--------	Número de bloques en el cuadro
unsigned char Expand[8]	--------	8 bytes de relleno

Tabla 2. Cabecera de 16 bytes de cada *frame*.

Llegados a este punto nos encontramos con la información que contiene la animación comprimida. Los datos correspondientes a un cuadro de animación se encuentran separados en bloques de distintos tipos para estructurar la información que se encuentra comprimida en ellos. Para identificar el tipo de bloque que vamos a tratar, a la cabecera de cada cuadro le sigue otra correspondiente al bloque. Esta cabecera está compuesta únicamente por dos campos: la primera de ellas indica la longitud en bytes del bloque y la segunda indica el tipo de bloque. A continuación de la cabecera del bloque, se encuentran los datos y, a estos, le puede seguir otro bloque o el siguiente cuadro de la animación si es que existe.

1. Tipos de bloques.

En los ficheros FLI existen, tal y como vimos anteriormente, diferentes tipos de bloques. En total, hay 5 bloques diferentes:

- Bloque *FliColor*.
- Bloque *FliLc*.
- Bloque *FliBlack*.
- Bloque *FliBrun*.
- Bloque *FliCopy*.

A continuación se explican más detalladamente cada uno de ellos.

1.1 Bloque *FliColor*.

Este tipo se identifica en la cabecera del bloque con un valor 11 en hexadecimal (11h). La información que contiene es una paleta de color comprimida cuya estructura es la siguiente: a la cabecera del bloque le sigue un valor 16 bits que indica el número de paquetes que contiene el bloque, encontrándose dichos paquetes a continuación del campo que indica su número.

Los paquetes a su vez están formados por una cabecera de dos bytes con la siguiente información: el primer byte indica el número de colores que no deben ser modificados de la paleta actual, es decir, indica el número del primer color a modificar y el segundo byte indica el número de colores de la paleta a modificar. Si este último campo tiene un valor de cero indica que se deben modificar los 256 colores de la paleta. A continuación de esta cabecera viene la paleta que, para cada color, está formado por tres bytes con la información para el rojo, verde y azul.

1.2 Bloque *FliLc.*

Es el más complejo de todos los bloques que conforman el fichero *.FLI. Se identifica en la cabecera del bloque por llevar un valor 12h. Este bloque contiene información sobre los cambios que se han producido en la pantalla entre el cuadro anterior y el actual. Mediante este truco solo almacenamos en el fichero las zonas de la pantalla que han sufrido modificaciones entre dos cuadros consecutivos de la animación. Lo primero que nos encontramos en este tipo de bloque es la cabecera del bloque, que tiene un tamaño de 4 bytes. Los dos primeros indican el número de líneas que no deben modificarse y los otros dos, las líneas a modificar.

Después aparece un byte para indicarnos el número de paquetes que conforman este bloque. Este valor puede ser conflictivo, ya que puede tener el valor cero. Los paquetes de este bloque disponen de su propia cabecera de dos bytes. El primero de ellos indica el número de bytes que no debemos modificar de la línea de la pantalla que estamos descomprimiendo y, el segundo byte, indica el número de bytes que debemos imprimir en pantalla. Si el valor de este último campo es mayor que cero (positivo), quiere decir que, a continuación, le siguen ese número de bytes que deben ser impresos en pantalla. Si por el contrario, el valor es negativo, quiere decir que le sigue un único byte que debe ser copiado menos N veces a pantalla.

1.3 Bloque *FliBlack.*

Se identifica por tener en su cabecera el valor decimal 13 (13d). Este bloque indica que la pantalla deberá ser borrada completamente, para lo cual se usará el valor cero. Este bloque no dispone de información adicional que sigue a la cabecera de dicho bloque.

1.4 Bloque *FliBrun.*

Este bloque contiene la información de una pantalla completa comprimida mediante un algoritmo "run-length". Este bloque se identifica en su cabecera con el valor decimal 15 y su estructura y función es muy similar a la del bloque FliLc.

Directamente después de la cabecera del bloque, viene un byte que indica el número de paquetes que forman el bloque FliBrun. Los paquetes están formados por un único byte cuyo contenido se interpreta de la siguiente forma: Si su valor es positivo, mayor que cero, le sigue un único byte que debe ser copiado N veces a pantalla y si, por el contrario, su valor es negativo, le siguen menos N bytes que deben ser copiados a la pantalla.

1.5 Bloque *FliCopy.*

Este bloque se identifica en su cabecera con el valor decimal 16 y su función es muy sencilla: a la cabecera del bloque le siguen 64000 bytes que deben ser copiados directamente a la pantalla, es decir, contiene información de una pantalla VGA completa sin comprimir. Este tipo de bloque es creado por el programa que ha generado la animación cuando la longitud del cuadro comprimido es mayor de 64000 bytes, con lo que resulta más aconsejable guardar el cuadro completo sin utilizar técnicas de compresión.

2. Librerías de funciones para manejar los ficheros con formato FLI.

Para la lectura de ficheros con este formato se utiliza un programa en C que realiza llamadas externas a unas rutinas en ensamblador. A continuación, paso a comentar las rutinas en C que se mostrarán posteriormente.

2.1 Función *GetFliInfo.*

Como su propio nombre indica, esta función devuelve información sobre el fichero, del tipo de número de frames que conforman la animación, etc.., aunque la función más importante de esta rutina es comprobar la validez de todo el FLI, ya que comprueba los números mágicos de la cabecera y de todos los cuadros que forman la animación. Como datos de entrada se le debe pasar un puntero a una estructura de comunicación con la librería y otro puntero al nombre del fichero que contiene la animación.

2.2 Función *OpenFli.*

Esta rutina abre el fichero FLI y devuelve la información completa de la cabecera del mismo. Se debe utilizar aunque previamente se haya llamado a *GetFliInfo*, ya que este último cierra el fichero cuando ha concluido su comprobación. Los parámetros de entrada son idénticos a los de *GetFliInfo* y el handle del fichero es devuelto en el campo correspondiente a la estructura de comunicación con la librería.

2.3 Función *CloseFli.*

Cierra el fichero FLI y el único parámetro que se le debe pasar es un puntero a la estructura de comunicación con la librería.

2.4 Función *BuildFrame.*

Esta es la función más importante de la librería, ya que es la encargada de cargar de disco un cuadro y descomprimirlo. Como datos de entrada, necesita un puntero a la estructura de comunicación de la librería y el valor de un segmento donde realizar la descompresión del cuadro. La descompresión no se realiza por defecto a pantalla, ya que

para realizar la reproducción de animaciones sobre *hardware* SuperVga se ha optado por realizar la descompresión en un *buffer* y luego realizar el volcado a pantalla.

Aunque este mecanismo es menos eficiente que la descompresión directa sobre pantalla, tiene la ventaja de proporcionar independencia a la librería respecto del *hardware* gráfico que se desee usar para la reproducción de animaciones.

2.5 Función *PlayFli.*

Reproduce una animación completa y los parámetros de entrada son idénticos a los que precisa la función *BuildFrame* comentada anteriormente. Ya que es capaz de reproducir toda la animación, la función *PlayFli* está orientada a reproducir animaciones sobre pantalla directamente en el modo VGA standard de 320 por 200 pixels con 256 colores.

2.6. Función *Copy80x86.*

Se trata de una rutina de copia de bloques de memoria. Como entrada, necesita de tres parámetros: el primero de ellos es un puntero de 32 bits a la dirección origen, el segundo es otro a la dirección destino y el tercero es un valor de 16 bits (2 bytes) que indica el número de bytes a copiar.

Esta función realiza su trabajo usando la función MOVSW en lugar de MOVSB, ya que la primera usa el mismo número de ciclos pero copiando dos bytes en lugar de uno, con lo que se gana tiempo.

2.7. Función *DosMalloc.*

Llama a la función *DosMalloc* (*Memory Allocation*) del DOS para reservar memoria. Como entrada se le pasa el número de párrafos requeridos y devuelve en el

acumulador (registro AX) el segmento de comienzo del bloque solicitado.

2.8. Función *DosFree.*

Llama a la función *Free* de DOS (Función para liberar memoria) para devolver al sistema operativo un segmento de memoria que había sido reservado previamente utilizando la función *Malloc*. Como entrada se le pasa la dirección de comienzo del bloque de memoria que queremos que devuelva al sistema.

2.9. Función *WaitVbl.*

Esta función sincroniza el barrido del haz de electrones del monitor, no devolviendo el control al programa en C hasta que se produce el intervalo de blanqueo vertical.

3. Listado del Programa y Librerías.

3.1 Programa.

```
/*
* Demo de la librer¡a de funciones para manejar FLIs
*
* En esta demo se reproduce la animación llamando a una función que descomprime un único cuadro
cada vez que se la llama
*/

#include "fli.h"
#include "dos.h"
#define VideoBios 0x10

main()
{
        union REGS Registros;
        struct FliData AniData;
        unsigned short int VideoSeg=0xA000;
        unsigned short int AniCont;
        char Animacion[]="ideas.fli";
//
// Modo 320x200 con 256 colores
//
        Registros.x.ax=0x13;
        int86( VideoBios, &Registros, &Registros );
//
// Chequea todo el fichero FLI
//
        GetFliInfo( &AniData, &Animacion );
//
// Buscamos info de la animaci¢n
//
        OpenFli( &AniData, &Animacion );
//
// Si no hay errores, reproducimos la animaci¢n cuadro a cuadro
//
        if( !AniData.FliError )
        {
         AniCont=AniData.Frames;
         while( AniCont )
         {
//
// Aqu¡ llamamos a la funci¢n de descompresi¢n de cuadros
//
          BuildFrame( &AniData, VideoSeg );
          AniCont--;
         }
        }
```

```
//
// Cerramos el FLI
        CloseFli( &AniData );
}
```

3.2 Librerías en ensamblador.

```
;*************************************************
;Funciones de acceso a ficheros en formato FLI
;*************************************************

        locals @@
        .286P


VideoBios     equ   10h
KeybBios      equ   16h
Dos   equ   21h

;
;Puertos del chip VGA
;

Status1 equ   3DAh

;
;Puertos del DAC de la VGA
;

PelWrite      equ   3C8h
PelRead equ   3C7h
PelData equ   3C9h

;
;Números "mágicos" usados en los FLI
;

FileMagic     equ   0AF11h
FramesMagic   equ   0F1FAh

;
;Estructura de la  cabecera de los ficheros FLI
;

FliHeader     struc
FliSize       dd    ?     ;longitud del FLI
FliMagic      dw    ?     ;para identificar los FLI
FliFrames     dw    ?     ;número de "frames" del FLI
FliWidth      dw    ?     ;anchura pantalla (320)
FliHeight     dw    ?     ;altura pantalla (200)
FliDepth      dw    ?     ;bits por pixel (8)
FliSpeed      dw    ?     ;número de "VBLs" entre "frames"
```

```
FliNext dw    ?     ;vale 0
FliFrit dw    ?     ;vale 0
FliFiller     db    102 dup(?)    ;relleno
        ends

;
;Además, cada "frame" tiene su propia cabecera
;

FrameHeader   struc
FrameSize     dd    ?     ;longitud del "frame"
FrameMagic    dw    ?     ;para saber que es un FLI
FrameChunks   dw    ?     ;número de bloques en el frame
FrameExpand   db    8 dup(?)      ;relleno
        ends

;
; los "chunks" también tienen su propia cabecera
;

ChunkHeader   struc
ChunkSize     dd    ?     ;longitud del bloque (chunk)
ChunkType     dw    ?     ;tipo del bloque
        ends

;
;Los tipos de bloques que pueden existir en un FLI son los siguientes
;

FliColor      equ   11    ;una paleta comprimida
FliLc         equ   12    ;una línea de pixels comprimida
FliBlack      equ   13    ;pone la pantalla con el color 0
FliBrun       equ   15    ;compresión "run-length"
FliCopy       equ   16    ;copiar 64000 bytes sin comprimir

;
;Estructura del "chunk" de paleta comprimida
;

FliColorData  struc
Packets dw    ?     ;número de paquetes en este bloque
        ends

;
;Estructura de los paquetes de paleta comprimida
;

ColorPacket   struc
SkipColors    db    ?     ;n£mero de colores a saltar
ChangeColors  db    ?     ;n£mero de colores a modificar
        ends

;
```

```
;Estructura del "chunk" de línea comprimida
;

FliLcData     struc
SkipLines     dw    ?     ;número de líneas que no cambian
ChangeLines   dw    ?     ;número de líneas a modificar
              ends

;
; Después de la estructura anterior viene 1 byte que indica el
; número de paquetes que forman la l¡nea comprimida.
;
;
;Estructura de los paquetes que definen una línea
;

LinePackets   struc
LineSkip      db    ?     ;pixels a ignorar
LineCount     db    ?     ;pixels a modificar
              ends

;
;Después de esta estructura vienen los datos si LineCount!=0.
;Si LineCount>0->le siguen el número de bytes que indica, los
;cuales deben ser copiados a pantalla.
;Si LineCount<0->le sigue un único byte que hay que repetir el
;número de veces que indica.
;

FliBrunData   struc
BrunNumPack   db    ?     ;número de paquetes FliBrun
              ends

;
;Estructura de los paquetes FliBrun
;

BrunPackets   struc
BrunCount     db    ?     ;longitud del paquete
BrunData      db    ?
              ends

;
;Si BrunCount>0->le sigue un sólo byte que hay que repetir el
;número de veces indicado.
;Si BrunCount<0->le siguen el número de bytes indicado, los
;cuales deben ser copiados a pantalla.
;

;
;Funciones usadas del DOS
;
```

```
SetVector     equ   25h
GetVector     equ   35h
Open  equ   3Dh
Close equ   3Eh
Read  equ   3Fh
Lseek equ   42h
Malloc equ   48h
Free  equ   49h
Realloc equ   4Ah
Exit  equ   4Ch

;
;Formato de la estructura para comunicarnos con el C
;

FliData struc
FlSize  dd    ?     ;longitud del fichero FLI
Frames  dw    ?     ;número de frames del FLI
FlWidth dw    ?     ;anchura FLI
FlHeight      dw    ?     ;altura FLI
FlDepth dw    ?     ;bits por pixel del FLI
FlSpeed dw    ?     ;velocidad reproducción
FliHandle     dw    ?     ;handle del fichero FLI
FliError      dw    ?     ;devuelve errores
        ends

_FliCode      segment byte public "CODE"

        assume  cs:_FliCode,ds:_FliCode

;*********************************
;Chequea y devuelve info sobre un fichero FLI
;Entradas:
;       puntero al nombre del fichero (en el stack)
;       puntero a estructura FliData (en el stack)
;Salidas:
;       llena los campos de info de FliData
;*********************************

FliDataPtr    equ   bp+6
FliName equ   bp+0Ah

        public _GetFliInfo

_GetFliInfo   proc  far
        push  bp
        mov   bp,sp
        pusha
        push  ds
        push  es
        cld
;
;Preparamos un buffer de 128 bytes
```

```
;
        mov   bx,128/16
        mov   ah,Malloc
        int   Dos
        jc    @@Error
        mov   word ptr cs:[Header],ax
        les   di,dword ptr [FliDataPtr]
        mov   word ptr es:[di.FliError],0
        lds   dx,dword ptr [FliName]
        mov   ah,Open
        xor   al,al
        int   Dos
        jc    @@Error
        mov   bx,ax
        mov   word ptr es:[di.FliHandle],ax
;
;Cargamos la cabecera del FLI
;
        push  si
        push  ds
        mov   ds,word ptr cs:[Header]
        xor   dx,dx
        mov   si,dx
        mov   cx,128
        mov   ah,Read
        int   Dos
;
;Comprobar el número mágico de la cabecera
;
        cmp   word ptr [si.FliMagic],FileMagic
        jz    @@HeaderOk
        pop   ds
        pop   si
@@Error:    mov   word ptr es:[di.FliError],0FFFFh
        pop   es
        pop   ds
        popa
        pop   bp
        ret
;
;Ahora comprobamos las cabeceras de todos los frames del FLI
;
@@HeaderOk:   movsw       ;devuelve longitud del FLI
        movsw
        sub   si,4
        sub   di,4
        mov   cx,word ptr [si.FliFrames]
        mov   word ptr es:[di.Frames],cx
        mov   dx,128
        xor   ax,ax
@@ChkFrames:  push  cx
        push  ax
        push  dx
```

```
        xor    dx,dx
        mov    cx,16
        mov    ah,Read
        int    Dos
        cmp    word ptr [si.FrameMagic],FramesMagic
        jz     @@FrameOk
        mov    word ptr es:[di.FliError],0FFFFh
;
;Apuntamos al siguiente frame en el fichero
;
@@FrameOk:    pop    dx
        pop    ax
        add    dx,word ptr [si]
        adc    ax,word ptr [si+2]
        mov    cx,ax
        push   ax
        push   dx
        mov    ah,Lseek
        xor    al,al
        int    Dos
        pop    dx
        pop    ax
        pop    cx
        loop   @@ChkFrames
        mov    ax,ds
        mov    es,ax
        mov    ah,Free
        int    Dos
;
;Cerramos el fichero
;
        mov    ah,Close
        int    Dos
        pop    ds
        pop    si
        pop    es
        pop    ds
        popa
        pop    bp
        ret
_GetFliInfo    endp

;**********************************
;Abre un fichero FLI
;Entradas:
;      puntero al nombre del fichero (en el stack)
;      puntero a estructura de comunicación (en el stack)
;Salidas:
;      devuelve info en la estructura
;**********************************

        public _OpenFli
```

```
_OpenFli      proc   far
        push   bp
        mov    bp,sp
        pusha
        push   ds
        push   es
        cld
        les    di,dword ptr [FliDataPtr]
        mov    word ptr es:[di.FliError],0
;
;Crea un buffer para la cabecera y la carga
;
        mov    bx,128/16
        mov    ah,Malloc
        int    Dos
        jnc    @@Ok
@@Error:      mov    word ptr es:[di.FliError],0FFFFh
        jmp    short @@Exit
@@Ok:  push    ax
        lds    dx,dword ptr [FliName]
        mov    ah,Open
        xor    al,al
        int    Dos
        mov    bx,ax
        mov    word ptr es:[di.FliHandle],ax
        pop    ax
        jc     @@Error
        mov    ds,ax
        xor    dx,dx
        mov    si,dx
        mov    cx,128
        mov    ah,Read
        int    Dos
        cmp    word ptr [si.FliMagic],FileMagic
        jnz    @@Error
        movsw
        movsw
        add    si,2
        mov    cx,5
        rep    movsw
        mov    ax,ds
        mov    es,ax
        mov    ah,Free
        int    Dos
@@Exit: pop    es
        pop    ds
        popa
        pop    bp
        ret
_OpenFli      endp

;**********************************
;Cierra un fichero FLI
```

```
;Entradas:
;       puntero a estructura de comunicación (en el stack)
;Salidas:
;       cierra el fichero FLI
;**********************************

        public _CloseFli

_CloseFli       proc    far
        push    bp
        mov     bp,sp
        push    ax
        push    bx
        push    si
        push    ds
        lds     si,dword ptr [FliDataPtr]
        mov     bx,word ptr [si.FliHandle]
        mov     ah,Close
        int     Dos
        pop     ds
        pop     si
        pop     bx
        pop     ax
        pop     bp
        ret
_CloseFli       endp

;*********************************************
;Lee un frame de disco y lo descomprime a un buffer
;Entradas:
;       puntero a estructura de comunicación (en el stack)
;       segmento donde descomprimir el frame
;Salidas:
;       se descomprime el frame en el segmento solicitado
;*********************************************

OutputSeg       equ     bp+0Ah

        public _BuildFrame

_BuildFrame     proc    far
        push    bp
        mov     bp,sp
        pusha
        push    ds
        push    es
        lds     si,dword ptr [FliDataPtr]
        mov     word ptr [si.FliError],0
;
;Creamos un buffer de disco
;
        mov     bx,65536/16
        mov     ah,Malloc
```

```
        int   Dos
        jnc   @@Ok
@@Error:    mov   word ptr [si.FliError],0FFFFh
        jmp   short @@Exit
@@Ok:  push   ax
        mov   bx,word ptr [si.FliHandle]
        mov   ax,cs
        mov   ds,ax
        mov   dx,offset CuadroHeader
        mov   si,dx
@@ReadHead:    mov   cx,16
        mov   ah,Read
        int   Dos
        and   ax,ax  ;EOF
        jnz   @@NotEof
;
;Colocamos el FLI en el primer frame...
;
        xor   cx,cx
        mov   dx,128
        mov   ah,Lseek
        xor   al,al
        int   Dos
;
;...y volvemos a intentar leer la cabecera del frame 1
;
        mov   dx,si
        jmp   short @@ReadHead
@@NotEof:    pop   ax
        cmp   word ptr [si.FrameMagic],FramesMagic
        jnz   @@Error
        mov   cx,word ptr [si.FrameChunks]
        push  cx
        mov   cx,word ptr [si]      ;longitud del frame
        sub   cx,16
        mov   ds,ax
        xor   dx,dx
        mov   ah,Read
        int   Dos    ;lee de disco el frame completo
        pop   cx     ;recupera número de chunks
        mov   es,word ptr [OutputSeg]
        mov   di,dx
        mov   si,dx
        call  DoFrame ;descomprime el cuadro
        mov   ax,ds
        mov   es,ax
        mov   ah,Free
        int   Dos
@@Exit: pop   es
        pop   ds
        popa
        pop   bp
        ret
```

```
_BuildFrame    endp

;***********************************************
;Reproduce un fichero FLI completo
;Entradas:
;      puntero a estructura de comunicación (en el stack)
;      segmento donde descomprimir
;Salidas:
;      reproduce la animaci¢n completa
;***********************************

        public _PlayFli

_PlayFli      proc   far
        push   bp
        mov    bp,sp
        pusha
        push   ds
        push   es
        lds    si,dword ptr [FliDataPtr]
        mov    word ptr [si.FliError],0
        mov    bx,word ptr [si.FliHandle]
        mov    cx,word ptr [si.Frames]
@@Play: push   cx
;
;Ponemos los parámetros en el stack como si la llamada se hubiese
;efectuado desde un programa escrito en C...
;
        push   word ptr [OutputSeg]
        push   ds
        push   si
;
;...y llamamos a la función de la librería que presenta 1 cuadro
;
        call   far ptr _BuildFrame
;
;Quitamos los parámetros del stack
;
        add    sp,6
        pop    cx
        loop   @@Play  ;hasta al fin de la animaci¢n !
        pop    es
        pop    ds
        popa
        pop    bp
        ret
_PlayFli      endp

;***********************************
;Procesa los "frames" del fichero FLI
;Entradas:
;      cx->número de "chunks" del cuadro
;      es:di->buffer donde descomprimir
```

```
;       ds:si->buffer con el cuadro comprimido
;Salidas:
;       descomprime un frame en es:di
;*************************************

DoFrame proc    near
@@DoFrame:      push    cx
        mov     ax,word ptr [si.ChunkType]
        cmp     ax,FliColor
        jnz     @@NextType1
        call    DoFliColor
        jmp     short @@NextType
@@NextType1:    cmp     ax,FliLc
        jnz     @@NextType2
        call    DoFliLc
        jmp     short @@NextType
@@NextType2:    cmp     ax,FliBlack
        jnz     @@NextType3
        call    DoFliBlack
        jmp     short @@NextType
@@NextType3:    cmp     ax,FliBrun
        jnz     @@NextType4
        call    DoFliBrun
        jmp     short @@NextType
@@NextType4:    cmp     ax,FliCopy
        jnz     @@NextType
        call    DoFliCopy
@@NextType:     pop     cx
        loop    @@DoFrame
        ret
DoFrame endp

;*************************************
;Procesa un "chunk" del tipo "FliLc"
;Entradas:
;       ds:si->apunta al "chunk" a procesar
;Salidas:
;       ds:si->siguiente "chunk"
;*************************************

DoFliLc proc    near
        push    si
        add     si,6
        xor     di,di
        mov     cx,word ptr [si.SkipLines]
        jcxz    @@DoLines
        mov     ax,cx
        mov     ch,cl
        xor     cl,cl   ;*256
        mov     di,cx
        add     ax,ax   ;*2
        add     ax,ax   ;*4
        add     ax,ax   ;*8
```

```
        add     ax,ax  ;*16
        add     ax,ax  ;*32
        add     ax,ax  ;*64
        add     di,ax
@@DoLines:      mov     cx,word ptr [si.ChangeLines]
        add     si,4
        xor     ah,ah
@@FrameLoop:    push    cx
        push    di
        mov     ch,ah
        lodsb
        mov     cl,al  ;cx->número de paquetes
        jcxz    @@NextLine
@@DoLine:       push    cx
        lodsb           ;número de bytes a saltar
        add     di,ax  ;los salta
        lodsb           ;número de bytes a copiar
        mov     cl,al
        jcxz    @@NextPack
        and     cl,cl
        js      @@Negativo
        push    cx
        shr     cl,1    ;para poder usar MOVSW
        rep     movsw
        pop     cx
        test    cl,1
        jz      @@NextPack
        movsb           ;por si el número era impar
        jmp     short @@NextPack
@@Negativo:     lodsb           ;byte a repetir
        mov     ah,al
;
;Pasamos la cantidad a valor absoluto
;
        xor     cl,0FFh
        inc     cl
        push    cx
        shr     cl,1
        rep     stosw
        pop     cx
        test    cl,1
        jz      @@NextPack
        stosb
@@NextPack:     pop     cx
        mov     ah,ch
        loop    @@DoLine
@@NextLine:     pop     di
        pop     cx
        add     di,320
        loop    @@FrameLoop
        pop     si
        add     si,word ptr [si.ChunkSize]
        ret
```

```
DoFliLc endp

;***********************************
;Procesa un "chunk" del tipo "FliBlack"
;Entradas:
;       ds:si->apunta al "chunk" a procesar
;Salidas:
;       ds:si->siguiente "chunk"
;***********************************

DoFliBlack      proc    near
        push    si
        add     si,6
        xor     ax,ax
        mov     di,ax
        mov     cx,32000
        rep     stosw   ;más rápido STOSW que STOSB
        pop     si
        add     si,word ptr [si.ChunkSize]
        ret
DoFliBlack      endp

;***********************************
;Procesa un "chunk" del tipo "FliCopy"
;Entradas:
;       ds:si->apunta al "chunk" a procesar
;Salidas:
;       ds:si->siguiente "chunk"
;***********************************

DoFliCopy       proc    near
        push    si
        add     si,6
        xor     di,di
        mov     cx,32000
        rep     movsw   ;más rápido MOVSW que MOVSB
        pop     si
        add     si,word ptr [si.ChunkSize]
        ret
DoFliCopy       endp

;***********************************
;Procesa un "chunk" del tipo "FliColor"
;Entradas:
;       ds:si->cabecera del chunk
;Salidas:
;       ds:si->apuntando al siguiente chunk
;***********************************

DoFliColor      proc    near
        push    si
        add     si,6
        lodsw
```

```
        mov    cx,ax  ;cx->número de paquetes
@@DoChunk:    push   cx
        mov    dx,PelWrite
        outsb
        mov    dx,PelData
        lodsb          ;colores a modificar
        mov    cl,al
        xor    ch,ch
        and    cx,cx
        jnz    @@Normal
        inc    ch     ;cx=256
@@Normal:     outsb          ;Red
        outsb          ;Green
        outsb          ;Blue
        loop   @@Normal
        pop    cx
        loop   @@DoChunk
        pop    si
        add    si,word ptr [si.ChunkSize]
        ret
DoFliColor    endp

;************************************
;Descomprime a pantalla un cuadro con compresión "run-length"
;Entradas:
;      ds:si->apunta al "chunk" a procesar
;Salidas:
;      ds:si->apunta al siguiente "chunk"
;************************************

DoFliBrun     proc   near
        push   si
        add    si,6
        xor    di,di
        mov    cx,200  ;líneas de la pantalla
@@DoScreen:    push   cx
        lodsb
        mov    cl,al
        xor    ch,ch  ;cx->número de paquetes
@@DoChunk:    push   cx
        lodsb
        mov    cl,al
        and    cl,cl
        js     @@Negativo
        lodsb          ;al->dato a repetir
        rep    stosb
        jmp    short @@NextPack
@@Negativo:    movsb
        inc    cl
        jnz    @@Negativo
@@NextPack:    pop    cx
        loop   @@DoChunk
        pop    cx
```

```
        loop    @@DoScreen
        pop     si
        add     si,word ptr [si.ChunkSize]
        ret
DoFliBrun       endp

;*********************************
;Reserva bloques de memoria
;Entradas:
;       número de p rrafos a reservar
;Salidas:
;       ax->segmento del bloque de memoria
;*********************************

        public _DosMalloc

_DosMalloc      proc    far
        push    bp
        mov     bp,sp
        push    bx
        mov     bx,word ptr [bp+6]
        mov     ah,Malloc
        int     Dos
        pop     bx
        pop     bp
        ret
_DosMalloc      endp

;*********************************
;Libera un bloque de memoria
;Entradas:
;       segmento del bloque de memoria
;Salidas:
;       ninguna
;*********************************

        public _DosFree

_DosFree        proc    far
        push    bp
        mov     bp,sp
        push    ax
        push    es
        mov     es,word ptr [bp+6]
        mov     ah,Free
        int     Dos
        pop     es
        pop     ax
        pop     bp
        ret
_DosFree        endp

;*********************************
```

```
;Copia una serie de bytes de una posición de memoria a otra
;Entradas:
;       puntero de 32 bits al origen
;       puntero de 32 bits al destino
;       longitud en bytes de la cadena
;Salidas:
;       copia los bytes usando MOVSW
;*********************************

        public _Copy80x86

_Copy80x86      proc    far
        push    bp
        mov     bp,sp
        push    cx
        push    si
        push    di
        push    ds
        push    es
        lds     si,dword ptr [bp+6]
        les     di,dword ptr [bp+0Ah]
        mov     cx,word ptr [bp+0Eh]
        cld
        push    cx
        shr     cx,1
        cld
        rep     movsw
        pop     cx
        test    cl,1
        jz      @@Exit
        movsb
@@Exit: pop     es
        pop     ds
        pop     di
        pop     si
        pop     cx
        pop     bp
        ret
_Copy80x86      endp

;*********************************
;Espera hasta que se produce el intervalo de blanqueo vertical
;Entradas:
;       ninguna
;Salidas:
;       retorna al principio del blanqueo vertical de la VGA
;*********************************

        public _WaitVbl

_WaitVbl        proc    far
        push    ax
        push    dx
```

```
        mov    dx,Status1
@@Wait: in     al,dx
        test   al,8
        jz     @@Wait
        pop    dx
        pop    ax
        ret
_WaitVbl      endp

Header dw     ?
CuadroHeader   dw     8 dup(?)

_FliCode      ends

        end
```

3.3 Librería usada por el programa en C. (Fli.h)

```
/*
* Estructuras y constantes usadas en los ficheros FLI
*
*/

//
//Estructura de comunicación con las rutinas en Ensamblador
//

struct FliData
        {
         unsigned long FliSize;    //Longitud del FLI
         unsigned short Frames;    //Número de frames
         unsigned short FliSpeed; //Velocidad reproducción
         unsigned short FliWidth; //Anchura en pixels
         unsigned short FliHeight;          //Altura en pixels
         unsigned short FliDepth; //Bits por pixel
         unsigned short FliHandle;          //Handle del fichero
         unsigned short FliError;  //Devolución errores
        };

//
// Las siguientes definiciones sólo serán útiles para manejar
// los FLI desde C, sin necesidad de usar las rutinas en
// ensamblador
//
//Aquí va la estructura de la cabecera de los ficheros FLI.
//Esta cabecera es lo primero que nos encontramos en un FLI.
//

struct FliHeader
        {
```

```
        unsigned long FileSize;   //Longitud fichero
        unsigned short Magic;     //Número mágico
        unsigned short FileFrames;        //Número de frames
        unsigned short FileWidth;         //Anchura en pixels
        unsigned short FileHeight;        //Altura en pixels
        unsigned short FileDepth;         //Bits por pixel
        unsigned short FileSpeed;         //Velocidad reproducción
        unsigned short FileNext; //Vale 0
        unsigned short FileFrit;  //Vale 0
        unsigned char Filler[102];        //Bytes sin usar
        };

//
// El campo comentado como "Número mágico" se usa para comprobar
// que es un FLI y su contenido debe ser AF11 en hexadecimal, por
// lo tanto, definimos
//

#define FileMagic 0xAF11

//
// A continuación vienen los cuadros o "frames" que forman la
// animación. Llevan una cabecera cuya estructura es la siguiente

struct FrameHeader
        {
        unsigned long FrameSize;          //Longitud del cuadro
        unsigned short FrMagic; //N£mero m gico
        unsigned short NChunks;           //Chunks en el cuadro
        unsigned char Expand[8];          //8 bytes de relleno
        };

//
// Aquí también hay un "número mágico" de comprobación cuyo valor
// debe ser F1FA en hexadecimal, por lo tanto, definimos

#define FrameMagic 0xF1FA

//
// Hay que tener cuidado cuando se usa la función LSEEK del DOS
// para colocar el puntero del DOS en un cuadro determinado del
// FLI, ya que el campo longitud incluye los 16 bytes que forman
// la cabecera de cada cuadro. Por lo tanto, si ya hemos leído
// del disco la cabecera del cuadro, deberemos restar 16 a la
// longitud que se nos indica en dicha cabecera para colocar el
// puntero del fichero en el siguiente cuadro.
//
// Cada cuadro está  formado por un número de bloques o "chunks"
// que contiene información comprimida. Hay varios tipos de
// "chunks", que van inmendiatamente después de la cabecera de
// cada cuadro. Cada "chunk" tiene una cabecera de 6 bytes con
// la siguiente estructura:
//
```

```
struct Chunk
        {
         unsigned long ChunkSize;        //Longitud del "chunk"
         unsigned short ChunkType;       //Tipo del "chunk"
        };

//
// Actualmente hay 5 tipos de "chunks" que se identifican con
// los siguientes valores:
//

#define FliColor 11       //Una paleta comprimida
#define FliLc 12          //Una línea de pixels comprimida
#define FliBlack 13       //Llenar la pantalla con el color 0
#define FliBrun 15        //Pantalla con compresión run-length
#define FliCopy 16        //64000 bytes sin comprimir

// El "chunk" del tipo FliColor está  formado por una palabra
// que indica el número de paquetes que hay en el "chunk". A
// continuación de esta palabra vienen los paquetes, cuya
// estructura es la siguiente:

struct FliColorPacket
        {
         unsigned char SkipColors;       //Colores a saltar
         unsigned char ChangeColor;      //Colores a modificar
        };

// Si el campo ChangeColor vale 0 quiere decir que hay que
// modificar los 256 colores de la paleta.
// A continuación vienen los datos de la paleta que, para cada
// color, est n formados por 3 bytes con la información para el
// rojo, el verde y el azul.
//
// Los "chunks" del tipo FliLc contienen los datos comprimidos
// sobre los cambios que se han producido en la pantalla entre
// 2 cuadros consecutivos, siempre y cuando la longitud compri-
// mida no exceda 64000 bytes en cuyo caso es preferible gene-
// rar un "chunk" del tipo FliCopy.
// Los  "chunks" del tipo FliLc tienen una cabecera con los si-
// guientes campos:
//

struct FliLcHead
        {
         unsigned short SkipLines;       //Líneas sin modificar
         unsigned short ChangeLines; //Líneas a modificar
        };

//
// A continuación de esta cabecera viene 1 byte que indica el
// número de paquetes del "chunk". Este byte puede valer 0, lo
```

```
// cual indicará que la línea no ha cambiado desde el cuadro
// anterior. Esto es útil ya que incluso dentro de un bloque de
// líneas donde se hayan producido cambios puede existir alguna
// que no haya sido modificada.
//

struct FliLcPackets
        {
         unsigned char SkipBytes;//Nº bytes sin modificar
         char ChangeBytes;       //Nº bytes a copiar a pantalla
        };

//
// El número de bytes a copiar puede ser 0, lo cual quiere decir
// que el número de bytes que se deseaban saltar era mayor de
// 255 bytes.
// Si el número de bytes a copiar es mayor que 0 quiere decir que
// le siguen ese número de bytes, los cuales deben ser copiados a
// la pantalla.
// Si el número de bytes a copiar es menor que 0 quiere decir que
// le sigue un único byte que debe ser repetido -N veces.
//
// El "chunk" del tipo FliBlack sólo indica que toda la pantalla
// debe colocarse con el color 0.
//
// El "chunk" del tipo FliBrun es muy similar al del tipo FliLc.
// Directamente después de la cabecera que identifica el tipo de
// "chunk" viene un byte que indica el número de paquetes que
// hay en el "chunk".
// Los paquetes est n formados por un único byte: si su valor es
// mayor que 0 le sigue un único byte que debe ser repetido N
// veces. Si es menor que 0 le siguen -N bytes que deben ser copiados a pantalla.
//
// El "chunk" del tipo FliCopy es muy sencillo. Después de la cabecera
// que identifica el tipo de "chunk", van 64000 bytes que
// deben ser copiados a la pantalla. Este tipo de "chunk" se genera
// cuando al intentar comprimir un cuadro su longitud excede de 64000 bytes.
//
```

APENDICE B. Listado de Fabricantes y Distribuidores.

ATAIO INTERACCION
Crta. Fuencarral-Alcobendas Km.12
28049 Madrid
Tlf: (91) 358.90.48
Fax: (91) 358.94.60

BROTHER
Enric Granados, 65
08008 Barcelona
Tlf: (93) 323.60.15
Fax: (93) 453.88.96

CIOCE
Numancia 117,121
08029 Barcelona
Tlf: (93) 419.34.37
Fax: (93) 321.52.10

COMPUMARKET
Avda. de Baviera, 1
28028 Madrid
Tlf: (91) 361.31.52
Fax: (91) 355.89.61

COMPUTER 2000
Constitución, 1
08960 Sant Just Devern
Tlf: (93) 499.91.11
Fax: (93) 499.91.12

DELL ESPAÑA
Barajas Park. San Severo s/n
28042 Madrid
Tlf: (91) 329.10.80
Fax: (91) 329.23.99

DEVELOPMENT ADVANCED TECHNOLOGY
Roger Lluria 53
08009 Barcelona
Tlf: (93) 487.02.87

DIODE
Orense, 34
28020 Madrid
Tlf: (91) 555.36.86
Fax: (91) 556.71.59

FOXEN COMPUTER
Marqués de Monteagudo 24
28028 Madrid
Tlf: (91) 361.06.66
Fax: (91) 356.63.13

FUJITSU ESPAÑA
Paseo de la Castellana, 95
28046 Madrid
Tlf: (91) 581.80.00
Fax: (91) 581.81.25

HANTAREX IBERICA
Aragón, 210
08011 Barcelona
Tlf: (93) 323.29.41
Fax: (93) 453.81.63

IBM ESPAÑA
Santa Hortensia, 26
28002 Madrid
Tlf: (91) 397.60.00
Fax: (91) 519.39.82

ICL
Arturo Soria, 245
28033 Madrid
Tlf: (91) 384.81.00
Fax: (91) 384.82.00

INGRAM MICRO-KEYLAN
Ciencias 149. Poligono Industrial Gran Via Sur
08908 Barcelona
Tlf: (93) 263.05.75
Fax: (93) 263.27.74

INVESTRONICA
Tomás Bretón, 62
28045 Madrid
Tlf: (91) 467.82.10
Fax: (91) 467.87.23

LIDER SHOP
Valencia, 176
08011 Barcelona
Tlf: (93) 454.81.00
Fax: (93) 454.55.62

MCB INFORMATICA
Palencia, 31
28020 Madrid
Tlf: (91) 554.29.92
Fax: (91) 554.02.89

METROLOGIE
Avda. de la Industria, 32
28100 Madrid
Tlf: (91) 661.11.42
Fax: (91)661.57.55

MICRO-EXE
Bruc, 60
08009 Barcelona
Tlf: (93) 301.37.77
Fax: (93) 301.12.29

NEC IBERICA
Avda. de Burgos, 16
28030 Madrid
Tlf: (91) 362.28.46
Fax: (91) 766.83.91

SINTRONIC
Real, 54
43004 Tarragona
Tlf: (977) 24.44.74
Fax: (977) 21.25.66

SISCOMP
Rosellón, 154
08008 Barcelona
Tlf: (93) 323.45.65
Fax: (93) 323.46.54

THINK COMPANY
Vizconde de Matamala, 15
28028 Madrid
Tlf: (91) 356.88.25
Fax: (91) 355.64.29

VENTA MICRO-INFORMATICA
Varsovia, 154
08026 Barcelona
Tlf: (93) 450.06.20
Fax: (93) 456.21.08

BIBLIOGRAFÍA.

- Internet News

- "Autodesk 3D Studio Release 2. Reference manual."
Autodesk

- "Targa+. Reference guide."
TrueVision

- "Targa+. Installation Guide."
TrueVision

- "VideoMaker+."
Truevision

- "Generación de Imágenes 3D: Software, hardware y estudio de mercado."
Carlos Alvarez Santana.

- "Organización de Computadoras: Un enfoque estructurado."
Andrew S. Tanenbaum.

- "Discos duros digitales."
Rev: Byte nº 193

- "Sistema CD-ROM con grabación de datos."
Rev: Byte nº 193

- "Avance en los sistemas de almacenamiento masivo."
Rev: Byte nº 193

- "Aceleradores graficos para windows."
Rev: Byte nº 1812

- "Avance en los sistemas magneto-opcticos."
Rev: Byte nº 193

- "3D Studio: Animaciones profesionales."
PcWorld.98

- "Aceleradores graficos para windows: Buses PCI y tarjetas de 64 bits."
PcWorld (USA).123

- "Bus Local: Funcionamiento y standares."
PcWorld.92

- " Capacidad de los dispositivos multimedia en Windows: Audio y video."
PcWorld.89

- "Captura de video para windows."
PcWorld.91

- "Discos duros de gran capacidad."
PcWorld(USA).126

- "Kit de herramientas IPAS para 3D Studio."
PcWorld.84

- "Tarjetas de vídeo."
PcWorld.91

- "Microsoft e Intel integran vídeo al PC."
PcWorld.84

- "Monitores de altas prestaciones."
PcWorld.104

- "Normas VESA para VGA."
PcWorld.92

- " Producción de peliculas digitales en PC."
PcWorld.90

- "Sistemas de vídeo digital para windows."
PcWorld.104

- "Situación actual de la televisión digital."
R.E. Electrónica.469

- "Software de digitalización de imágenes."
PcWorld.86

- "Tarjeta digitalizadora de video en tiempo real."
PcWorld.92

- "Tarjeta Intel para la grabación de video en una sola etapa."
PcWorld.90

- "Tarjetas de vídeo."
PcWorld.93

- "Tarjetas digitalizadoras de video."
PcWorld.96

- "Unidades CD-ROM grabables."
PcWorld.96

- "Unidades CD-ROM para multimedia."
PcWorld(USA).124