Animación en una red de ordenadores

7.1. Introducción.

7.1.1. Objetivo del trabajo.

En la actualidad, en el campo de la animación generada por ordenador existen dos posibilidades básicas: la animación en tiempo real y la animación cuadro a cuadro.

El primero de los dos métodos consiste en generar las imágenes que componen la secuencia a una velocidad tal que puedan ser grabadas a la misma velocidad a que se reproducirán posteriormente, esto es a 25 imágenes por segundo (30 en EEUU)[1]. Dado que cada imagen requiere la realización de una gran cantidad de cálculos matemáticos - proyecciones, cálculos de iluminación, eliminación de superficies ocultas, etc. -, esta opción se limita a métodos de representación poco sofisticados realizada con ordenadores muy potentes con hardware específico para cálculos con geometrías 3D.

Por lo tanto, cuando se desea utilizar métodos de cálculo más sofisticados (más exactos) como pueda ser el trazado de rayos, se debe recurrir a la segunda solución.

En la grabación cuadro a cuadro se calcula cada imagen de la secuencia (frame) por separado, a continuación esta imagen se debe insertar en la secuencia final, en la posición exacta que le corresponde. Por lo general, es necesario disponer de un aparato de video cuyos motores puedan ser controlados por un ordenador con la suficiente precisión como para poder grabar una única imagen (con duración de 1/25 segundos) en un punto muy determinado de la cinta.

De esta manera se evita la limitación en cuanto a tiempo disponible para el cálculo y además, al separar el trabajo de cálculo del de representación de la animación, se facilita el empleo de una red de ordenadores de forma que el trabajo se reparta entre las máquinas disponibles.

---

[1] En lo que sigue se supondrá que el número de imágenes por segundo corresponde al estándar europeo a no ser que se diga explícitamente lo contrario.

Concretando, el objetivo de este trabajo ha sido la puesta a punto de un sistema de animación infográfica sobre una red de ordenadores utilizando la técnica del trazado de rayos. Para ello se trabajó sobre la red existente en el Centro Politécnico Superior de la Universidad de Zaragoza con la librería de trazado de rayos AlephTracer [§ALEPH1].

7.1.2. Planteamiento.

En una primera aproximación podemos dividir el sistema informático necesario en tres partes bien diferenciadas: un equipo de grabación, un sistema de cálculo y por último, el sistema de control.

El equipo de grabación está compuesto de:

- Un aparato de video con control cuadro a cuadro.
- Un ordenador.
- Dos (o más) monitores de video.

El ordenador debe ser capaz de:

- Recibir a través de la red las imágenes calculadas.
- Generar las señales de control oportunas para manejar el aparato de video.
- Generar una señal de video correspondiente a la imagen recibida.

El trabajo con dos monitores se debe a la necesidad de controlar el propio ordenador y de saber en todo momento qué imagen está recibiendo el aparato de video. Un tercer monitor resulta conveniente si se desea comparar la imagen generada con la registrada en la cinta de video.

Respecto al sistema de cálculo, se parte de una serie de programas escritos en C, utilizando una librería de trazado de rayos; que funcionarán sobre máquinas Unix. En principio, no existe ninguna limitación en cuanto a tipo de máquina o sistema operativo; las únicas que deben cumplir los programas de cálculo son:

- Debe poder recibir los parámetros que definen la imagen o bien leyéndolos de un archivo, o bien por comunicación con otro proceso.
- Debe poder generar ficheros de imagen en un formato predeterminado.
- Debe poder enviar dichos ficheros a otros ordenadores a través de la red.
- Conviene que pueda recibir señales de control para suspender o reanudar el trabajo.

Por último, el sistema de control debe encargarse de:

- Suministrar los datos que precisen los programas de cálculo.
- Controlar la ejecución de los cálculos. Dado que el trazado de rayos es una técnica que consume mucho tiempo de CPU, conviene realizar los cálculos en horas en las que no estén funcionando procesos en modo interactivo (por lo general, por la noche).
- En caso de que no lo haga el sistema de grabación, también deberá realizar el post-proceso de las imágenes.

7.2. El sistema de grabación.

7.2.1. Configuración.

En todo el sistema de animación, el único campo abierto en cuanto a hardware se refiere, es la parte correspondiente a la grabación de las imágenes; ya que el resto (red de ordenadores que realizarán los cálculos necesarios) ya viene dado por la configuración existente en el CPS.

Comenzando por el dispositivo físico que realiza la grabación sobre la cinta de video, se debe tener en cuenta que ya existe un formato estándar aceptado por la mayor parte de las empresas dedicadas a los medios audiovisuales, al que deberemos ajustarnos.

En este formato, se registran en la cinta cinco pistas: dos de audio, una de video, un código de tiempos y una pista de control. Es la existencia de esta pista de control la que hace posible un determinación precisa de la posición de las cabezas de grabación sobre la cinta.

En la actualidad existen dos tipos de aparatos de video (VTR) que utilizan este formato, unos basados en tecnología analógica y otros en tecnología digital. Para el trabajo que nos ocupa es suficiente con un aparato analógico, ya que no realizaremos labores de edición (recuperación y modificación de imágenes ya grabadas).

A continuación se expondrá el ciclo de trabajo del VTR; para comprenderlo es necesario entender el significado de los siguientes conceptos:

- Cuadro: Es la unidad fundamental de la animación. Como ya sabemos una animación consiste en la proyección de una sucesión de imágenes a suficiente velocidad como para que la apariencia sea de movimiento uniforme. A cada una de estas imágenes se le denomina ‘Cuadro’. En video, el número de cuadros proyectados por segundo es de 25.

- Puntos de Inserción y de Salida: El funcionamiento normal de un video betacam consiste en la grabación de un número determinado de cuadros consecutivos a partir de una posición determinada (punto de inserción). Cuando la cinta se usa por primera vez, se graban completamente las pistas de control y de código de tiempos; de forma que a partir de ese momento, cada posición de la cinta queda identificada biunívocamente por un código de tiempo (expresado en la forma hh:mm:ss:ff, donde ff representa el ordinal del cuadro, o lo que es igual, 1/25 de segundo). El código de tiempo correspondiente a la suma del número de cuadros grabados y el punto de inserción es lo que se conoce como punto de salida.
- Cuadros por Imagen: Expresa el número de cuadros consecutivos en los que se registra una misma imagen. Por lo general se utiliza un número de cuadros por imagen igual a la unidad; aunque en los casos en los que el movimiento de la imagen sea lento, puede aumentarse (a costa de una pérdida en la suavidad del movimiento; como contrapartida se puede obtener una duración mayor de la animación con el mismo tiempo de cálculo).

Con esto, al comienzo de la grabación se deberá inicializar el sistema con los datos correspondientes al nº de cuadros por imagen y al punto de inserción inicial (el punto de inserción del primer cuadro de la animación), ya que el punto de salida se puede determinar en función de estos dos datos.

Una vez inicializado el sistema, y suponiendo que la imagen correspondiente al cuadro que se va a grabar ya está disponible en las entradas de señal del VTR, el proceso es el siguiente:

Los motores de arrastre de la cinta se ponen en marcha a alta velocidad hasta alcanzar un punto determinado que precede al punto de inserción en una cantidad determinada de tiempo (lo que se conoce como Preroll Time). En este punto se detiene para a continuación ponerse otra vez en marcha a velocidad normal. Durante el tiempo que falta hasta llegar al punto de inserción la velocidad de los motores se estabiliza en su valor nominal, de forma que en el momento que el punto pasa sobre los cabezales la señal es registrada sin ruidos ni distorsiones. Una vez superado el punto de salida, los cabezales vuelven al modo de lectura y los motores comienzan a frenar.

Hecho esto, el sistema debe calcular los nuevos puntos de inserción y de salida (obviamente el nuevo punto de inserción coincidirá con el antiguo punto de salida, a no ser que las imágenes no sean calculadas secuencialmente).

Como se puede comprender fácilmente todo este juego de motores lleva su tiempo, teniéndose por lo general que emplear alrededor de 25 segundos para cada inserción en la cinta. Un rápido cálculo nos lleva a deducir que para grabar medio minuto de animación serían necesarias algo más de 5 horas de espera. Ahora bien, "por suerte" este tiempo es despreciable frente a lo que cuesta calcular cada imagen mediante el trazado de rayos, con lo que estos 25 segundos por imagen no constituyen ninguna limitación.

En la descripción anterior se ha hecho referencia a ciertas consideraciones como la necesidad de ajustar los puntos de inserción y de salida y la suposición de que la señal de video está disponible para ser registrada. Está claro que en la imagen calculada en forma de números no llega por si sola a la entrada de video del VTR. Además el aparato no tiene forma de saber cuando la señal de video que le llega es la que debe registrar. Debido a todo esto es bastante evidente la necesidad de un segundo aparato capaz de realizar todas estas labores; siendo la solución más sencilla la de emplear un ordenador.

Como hemos visto, el ordenador deberá generar una señal de video, y además deberá generar las señales necesarias para controlar el VTR. Comenzando por la señal de video, será necesario disponer de una tarjeta de video capaz de generar una señal conforme al estándar que utilice nuestro aparato, por lo general PAL. Esta señal vendrá dada en forma de 4 cables coaxiales, 3 correspondientes a la imagen propiamente dicha y 1 correspondiente a lo que se denomina señal de sincronismo, que indica el instante en que comienza cada imagen.

La utilización de tres canales separados para la transmisión de la imagen se debe a la separación en componentes (rojo, verde y azul -RGB) con objeto de conseguir una mayor calidad. Esta separación en tres componentes aditivas es la que mejor se adapta a la forma de trabajo del ordenador (y de el monitor), no obstante el sistema betacam utiliza una separación diferente (denominada YBR), por lo que además de los aparatos ya nombrados deberá intercalarse un conversor de señal (de RGB a YBR).

La señal de sincronismo tiene una gran importancia en todos los sistemas profesionales de video; por lo general se debe disponer de lo que se denomina un generador de sincronismo a partir del cual se distribuye la señal a todos los aparatos que toman parte en la grabación del video, aunque en instalaciones pequeñas, puede utilizarse el propio generador de la señal de video.

En segundo lugar, tenemos la necesidad de hacer saber al VTR lo que debe hacer en cada momento, por lo que necesitaremos un tercer canal de comunicación entre el VTR y el ordenador. Para generar estas señales se utilizará una segunda tarjeta, especialmente diseñada para comunicarse con una amplia variedad de marcas y modelos de aparatos de video.

Por último, teniendo en cuenta la necesidad de un segundo monitor para el control del propio ordenador, la configuración final quedará como se puede apreciar en la figura 7.1

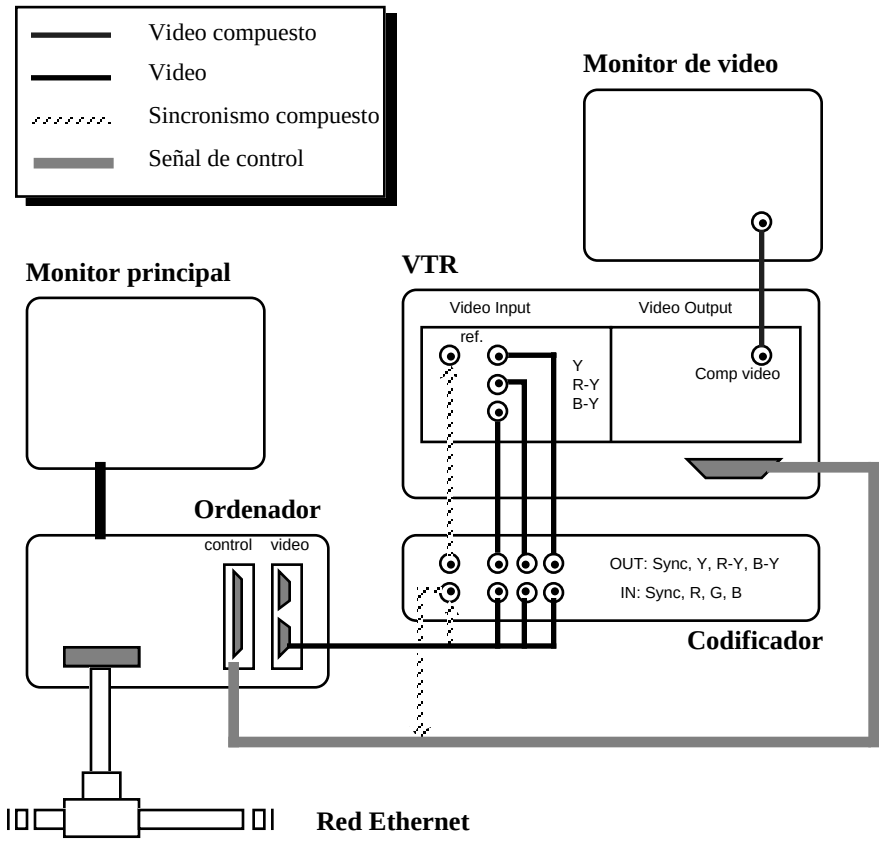

Figura 7.1: Configuración del sistema de grabación

Como se puede apreciar en este esquema, la señal de sincronismo es generada por la tarjeta de video. Esta señal se distribuye al codificador, a la tarjeta de control mediante una derivación y por último al VTR a través de un puente en el propio codificador.

Es importante notar que esta señal se deteriora rápidamente al viajar de un aparato a otro, especialmente en el caso de la existencia de derivaciones; y dada la importancia de que la señal llegue idéntica a todos los equipos, puede ser necesario, dependiendo de factores como la longitud de los cables o la existencia de interferencias, utilizar un generador de sincronismo aparte y uno o varios distribuidores de señal.

7.2.2. Hardware.

La elección de los aparatos que componen la estación de grabación viene condicionada por la necesidad de aprovechar el material disponible y por las limitaciones que imponga el presupuesto.

En cuanto del ordenador, debido a la primera de las dos razones anteriores, se utilizó un Macintosh IIvi con las siguientes características:

- Procesador: 68030
- Frecuencia del reloj: 16 MHz
- RAM: 5 Mb
- Disco interno: 40 Mb
- Sistema Operativo: MacOS 7.1
- Nº de expansiones NuBus: 3
- Tarjeta de video 8•24
- Tarjeta Truevision NuVista+™ 2M, 1.0
- Tarjeta DQ M.A.C. animation controller, Alpha 1.2
- EtherNet SCSI Interface DaynaPort Link

Como se puede observar, las 3 ranuras NuBus están ocupadas por las dos tarjetas de video (8•24 y NuVista) y por la de control del VTR (DQ M.A.C.), por lo que la conexión a la red se deberá realizar a través de un interface vía SCSI.

La utilización del sistema MacOS presenta ciertas ventajas e inconvenientes frente a un sistema Unix; principalmente en el campo de las comunicaciones y en el de la modularidad de la programación; estos problemas se comentarán más adelante, en el capítulo dedicado al software.

Respecto a la señal de sincronismo, cabe destacar la existencia de dos modelos de tarjeta DQ M.A.C, uno de ellos puede generar sincronismos y el otro no. Por lo general no es necesario que los genere, bastando con la señal que proporciona la tarjeta de video (NuVista).

Por último, el VTR elegido fue un VideoCasette PVW-2800P Betacam SP Sony, junto con un codificador RGB-YRB AXON.

7.2.3. Software.

7.2.3.1. Comunicaciones

En primer lugar se considerará la comunicación con el resto del sistema. Por una parte, lo fundamental es la recepción de los archivos de imagen generados por el sistema de cálculo. Dado que el sistema de cálculo está formado por máquinas Unix, el sistema obvio de comunicación es el FTP; con lo que será necesario utilizar un servidor de FTP para Macintosh.

Existen diferentes servidores de FTP disponibles en el mercado. Se han considerado los siguientes:

• NCSA Telnet

Este es un programa de dominio público que permite la conexión con diferentes sistemas operativos a través de emulación de terminal de texto; además, lleva incorporado un servidor de FTP completo. Este servidor presenta los siguientes inconvenientes:

- Requiere que toda la aplicación esté cargada en RAM.
- No permite varias conexiones simultaneas.
- No utiliza los usuarios y palabras clave definidos en el sistema.
- Aparecen dificultades a la hora de especificar nombres de directorios y archivos que contengan espacios y/o caracteres especiales del castellano.

Otras características de interés son la posibilidad de crear un fichero de usuarios autorizados a conectar, utilizando palabras clave encriptadas y la posibilidad de monitorizar la actividad del servidor a través de una ventana de texto.

Esta opción fue rechazada debida principalmente a los dos inconvenientes mencionados en primer lugar. La necesidad de tener varias conexiones simultaneas aparece debido al sistema paralelo de cálculo; efectivamente, mientras un ordenador está enviando una imagen al sistema de grabación, otro ordenador puede terminar de calcular su imagen, y si el servidor de FTP sólo permite una conexión, el segundo ordenador quedará bloqueado esperando a poder efectuar la conexión.

• FTPd v2.1.0 © 1992-93 Peter Lewis.

FTPd es uno de los servidores de FTP (y Gopher) más completos para Macintosh. Es Shareware, y se puede encontrar en mac.archive.umich.edu y en otros servidores de “ftp anonymous”.

Presenta, entre otras, las siguientes ventajas:

- Permite conexiones FTP y Gopher.
- Permite múltiples conexiones simultaneas.
- Utiliza los Usuarios y Grupos del sistema 7.
- Permite utilizar diferentes formatos en la transmisión de ficheros.
- Puede funcionar como aplicación de fondo.
- Permite definición automática del tipo de fichero en el sistema Macintosh según la extensión del nombre.

Esta parece ser la mejor opción para todos los Macintosh que tengan suficiente memoria RAM; ya que el máximo número de conexiones simultaneas depende de la particición de memoria reservada para el servidor.

• VersaTerm FTP Server.

Este servidor es muy similar al incorporado en NCSA Telnet; las principales diferencias son que no requiere que una aplicación ‘grande’ quede cargada en memoria y que admite solamente la definición de un máximo de 4 usuarios.

También presenta el problema de la imposibilidad de varias conexiones simultaneas. Esta limitación puede superarse mediante la separación de los procesos de grabación y postproceso. En efecto, si todas las máquinas que calculan imágenes envían los ficheros a una máquina Unix intermedia (que obviamente no tiene ningún problema en aceptar múltiples conexiones), esta puede almacenarlos, realizar las posibles operaciones de postproceso y enviar los ficheros resultantes, de uno en uno, al sistema de grabación.

En cualquier caso, para ordenadores de gama media-baja, la limitación a una sola conexión simultanea aumenta la fiabilidad del proceso.

Una solución más elegante pasaría por la utilización del sistema operativo Unix en el sistema de grabación. Es interesante notar que, además, resulta mucho más sencilla la programación de las operaciones de postproceso en forma modular en un sistema Unix que en un MacOS.

7.2.3.2. Salida de video

La señal de video es generada por la tarjeta NuVista+, esta tarjeta, de cara al usuario, aparece como un segundo monitor, cuya utilización es totalmente transparente (ambos monitores configuran un espacio de trabajo común, de manera que no es necesario manejar soft específico para utilizar esta tarjeta). Esta es una ventaja del sistema Macintosh frente a otros; de la misma manera, el programador puede utilizar las propias librerías del sistema (QuickDraw) para acceder al buffer de pantalla de la tarjeta NuVista+.

Por otra parte, existen opciones propias de la tarjeta, como pueda ser la magnificación[2] por hardware, a las que se puede acceder mediante una librería que proporciona la empresa fabricante (Truevisión) a través de la red de Apple de apoyo a los desarrolladores (AppleLink)

Existen aplicaciones comerciales que son capaces de manejar el VTR y grabar cuadro a cuadro una secuencia de imágenes previamente almacenadas en algún dispositivo al que el ordenador tenga acceso. El problema de estas aplicaciones es que una secuencia larga ocupa una gran cantidad de espacio en disco, con lo que para utilizar estas aplicaciones sería necesario calcular un determinado número de cuadros, enviarlos al disco duro del ordenador del sistema de grabación, lanzar el proceso de grabación y borrar los archivos del disco. Dado que una imagen puede ocupar un tamaño medio de 1Mb (a la resolución que dicta el formato PAL, formato de fichero PICT2 con compresión RLE sin pérdidas), este proceso resultaría demasiado lento, además de dificilmente automatizable.

Por lo tanto aparece la necesidad de programar una aplicación que realice la siguiente tarea:

- Esperar la aparición de una nueva imagen en el disco duro.
- Volcar esa imagen a pantalla
- Poner en marcha la grabación en el VTR
- Esperar a que termine la grabación
- Borrar la imagen del disco y repetir el proceso

Dentro del ámbito de prográmación para Macintosh, aparte de los diálogos, menús y ventanas típicos del MacOS, se deben resolver los siguientes problemas:

---

[2] Aumento por un factor n de las dimensiones de la imagen.

- En la espera de la aparición en el disco de una nueva imagen el programa deberá chequear periódicamente el directorio seleccionado. Para reconocer la llegada de un nuevo archivo (cuyo nombre se supone que estará compuesto por una raíz común a todos los cuadros, un número de índice y una sufijo opcional) pueden utilizarse dos sistemas: En primer lugar se puede intentar obtener información sobre el archivo que se espera; si se obtiene un error (archivo no encontrado) se debe seguir esperando. La otra opción consiste en intentar abrir directamente el archivo.

El primer sistema tiene el inconveniente de que se obtiene un resultado no erroneo aún cuando el archivo ya ha sido creado pero no se ha terminado de recibir. Una posible solución a este problema consiste en intentar abrir, en lugar de la imagen que se espera, la siguiente a ésta (se supone que los cuadros llegan ordenadamente); de modo que si está empezando a llegar la imagen n+1, ello implica que la imagen n ya ha sido totalmente recibida. Este método se puede generalizar cuando las imágenes llegan de m máquinas distintas siempre y cuando cada una genere cuadros en sucesión n, n+m, n+2m...

El segundo método resulta más adecuado en general (nótese que en el sistema anterior las últimas m imágenes quedan sin grabar, por lo que deben grabarse manualmente). Este método funcionará siempre y cuando la aplicación que escribe la imágen (en nuestro caso, el servidor de FTP) mantenga el fichero abierto con acceso exclusivo mientras dure todo el proceso de transmisión. Efectivamente, en este caso, al intentar abrir el archivo a medio escribir, se obtendrá un error de acceso. Por otra parte este sistema puede fallar en algunos casos (por ejemplo, cuando se graban archivos a través del servidor AppleShare), por lo que se han dejado ambas opciones en la aplicación final.

- En lo que respecta a la generación de la señal de video, se pueden utilizar las funciones del “picture utilities package” (standar de Macintosh). Existe un interface para la utilización de estas funciones disponible en los CD’s que suministra Apple con codigo fuente en C para desarrolladores registrados; en concreto se puede encontrar en:

Dev.CD Jan 93:Periodicals:develop:develop 10 code:Picture Utilities

Con esto queda resuelto el problema de la interpretación del archivo PICT2; todavía quedan por resolver dos problemas: el de encontrar las coordenadas correspondientes a la pantalla en la que se va a volcar la imagen y el de conseguir llenar dicha pantalla con la imagen, sin que aparezcan elementos como marcos de ventanas y/o menús.

El problema de encontrar la localización de los distintos monitores disponibles se puede resolver mediante la función GetDeviceList(). (ver figura 7.2)

```
Rect                PackORects[7];
char                LastMonitorID=0;
GDHandle GDHdl;

        GDHdl = GetDeviceList();

        do
        {
                BlockMove(&(**GDHdl).gdRect, &PackORects[LastMonitorID], sizeof(Rect));
                        LastMonitorID++;
        }
        while((GDHdl = GetNextDevice(GDHdl))!=NIL);
```

Figura 7.2: Localización de los monitores.

Para conseguir llenar toda la pantalla con una imagen se deben seguir los siguientes pasos:

1: Abrir una ventana cuyas coordenadas se ajusten a las del monitor seleccionado.

2: Forzar el redibujado de la ventana mediante la función *InvalRect()*.

3: Forzar el valor de la variable global *MBarHeight* a cero con objeto de impedir que otras aplicaciónes intenten escribir en la barra de menú. (Estos dos últimos pasos no son necesarios si el monitor en cuestión no es el principal -el que contiene la barra de menús-)

- La comunicación con la tarjeta de control del VTR resulta ser el punto más conflictivo; principalmente debido a la dificultad existente para conseguir las librerías necesarias. La única herramienta de programación que suministra la empresa DiaQuest consiste en un módulo de extensión para el programa MacroMind Director; este programa dispone de un lenguaje de programación (Lingo) especialmente diseñado para aplicaciones interactivas multimedia; pero a nivel de sistema operativo tiene serias deficiencias que impiden que todo el sistema de grabación pueda programarse en este lenguaje.

Ante la falta de las librerías no queda otra solución que dividir el sistema de grabación en un programa de volcado de imágenes a pantalla (escrito en C) y otro de control del VTR escrito en Lingo.

La solución adoptada se basa en la creación de archivos de intercambio de información. Cuando la primera aplicación termina el volcado a pantalla de una imagen (esto es, la señal de video ha sido generada y está lista para ser grabada), crea un archivo con un nombre prefijado en un directorio al cual tiene acceso la segunda aplicación, que está bloqueada esperando la aparición del mismo.

El fichero contiene una linea de texto con el nombre del fichero de imagen cargado y la fecha y hora actual. La creación de este fichero equivale a la ejecución de una instrucción de grabado; y la desaparición del archivo despues de ser creado es equivalente a una señal de 'proceso de grabación terminado'. En efecto, cuando el segundo programa detecta la aparición del fichero, lee su contenido (con objeto de crear posteriormente un fichero 'histórico'), pone en marcha la grabación en el VTR y cuando ésta termina, borra el fichero de intercambio, repitiéndose el proceso hasta que es interrumpido por el usuario.

Por lo tanto, cuando el primer programa ve desaparecer el fichero, cerrará la ventana de imagen, borrará -si así lo había seleccionado el usuario- la imagen del disco duro, y pasará al modo de espera.

Este sistema de dos aplicaciones funcionando simultáneamente plantea dos problemas:

En primer lugar, la instrucción correpondiente a 'inicializar la tarjeta de control' se convierte en una instrucción de lanzar una aplicación (la de control) desde el interior de otra (la de representación). Una solución a este problema se encuentra en Inside Macintosh,Technical Note 126. En segundo lugar, debido a la propio sistema de programación de MacroMind, una aplicación generada con este programa no puede funcionar de fondo, de modo que el ordenador queda confinado a la tarea de grabación exclusivamente.

- Por último, el código necesario para poder guardar los parámetros de la aplicación (retardo en la espera de archivos, tarjeta de video a utilizar, magnificación de la imagen, borrado de archivos una vez grabados, etc...) se puede encontrar en

Dev.CD Jan 93:Technical Documentation:Sample Code:Snippets:Toolbox

En este directorio hay un ejemplo completo bajo el nombre de *'Prefs'* .

Además de esto, para tener la opción de suspender el trabajo y reanudarlo posteriormente, debemos saber en que directorios están los ficheros con los que estamos trabajando (los ficheros de imagen y la aplicación de comunicación con el VTR). Esto se puede conseguir mediante la función que devuelve el nombre completo de un archivo (incluyendo los directorios 'padres') que aparece descrita en Inside Macintosh Vol VI, Files, pág 2-44.

7.2.3.3. Control del VTR

Como se ha dicho, el control del VTR se realizará mediante una aplicación generada por Macromind Director. Para realizar esta aplicación, se deberán utilzar dos extensiones del Lenguaje: la extensión de acceso a archivos y la extensión de control de la tarjeta.

Para el manejo del VTR se deberán definir una serie de funciones en el campo denominado *"Movie Script"* :

```
on initVTR
  global VTR

   -- close it if it was previously left open
  if objectP(VTR) then VTR(mDispose)

  openXlib "DiaQuestXObj"
  set VTR = DiaQuestXObj(mNew)

   -- test that it was successfully opened
  if not objectP(VTR) then
     alert "Diaquest Object not made!"
    exit
  end if

end initVTR

on waitVTR
  global VTR

  repeat while VTR(mBusy) = 1
     put "Wait..." into field "Busy"
  end repeat
   put "Free" into field "Busy"
end waitVTR

on checkBusy
  global VTR

  if VTR(mBusy) = 1 then
     put "Wait..." into field "Busy"
  else
     put "Free" into field "Busy"
  end if
end checkBusy
```

Figura 7.3: Funciones globales

La función initVTR deberá llamarse al comienzo de la ejecución. Una vez hecho esto, aparte de diferentes opciones de interactividad y de modos de utilización, sólo queda la definición de las condiciones iniciales y el bucle principal de trabajo.

Los valores del número de cuadros por imagen y del punto de entrada se establecen mediante la instrucción

VTR(mSetUp,value(field "FPI"),field "inpoint",1)

en la que los respectivos valores de los campos*"FPI" e "inpoint"* se supone que han sido previamente establecidos, o bien interactivamente o bien mediante recuperación de datos de un fichero de configuración.

El ciclo principal, ya descrito en el apartado anterior, se realizará mediante el siguiente código:

```
-- This frame is labelled "ThisFrame"

global sockObj
global logObj

put VTR(mInPoint) into field "ActInPoint"
put "Esperando Imagen..." into field "status"

put FileIO(mNew,"read","x.sock") into sockObj

if objectP(sockObj) then

  -- Existe el fichero, lo que significa que la imagen
  -- está disponible

  put "Esperando VTR..." into field "status"
 waitVTR

  put "Grabando..." into field "status"
 -- Aquí se manda la señal de control al VTR
 VTR(mAtFrame,1,1)

  put sockObj(mReadLine) & " " & field "ActInPoint" into field "done"

 sockObj(mDelete)
 if the result < 0 then alert "¡¡¡Delete: Error espantoso!!!!."
 -- Al borrarse el fichero, se le comunica a la aplicación
 -- de video que la grabación ha terminado

 put FileIO(mNew,"append","VTR.log") into logObj
 if not objectP(logObj) then alert "No log"
 logObj(mWriteString,"Recorded " & field "done" & " " & the long time & RETURN)
 logObj(mDispose)

end if

go to frame "ThisFrame"
```

Figura 7.4: Bucle de grabación

7.3. El sistema de cálculo.

7.3.1. Configuración.

El sistema de cálculo está basado en la división de la animación en cuadros independientes (obviamente puede existir una coherencia entre cuadros consecutivos, pero esto no puede ser aprovechado por las librarías de trazado de rayos de que se dispone actualmente).

Suponiendo que las diferencias entre distintos cuadros pueden definirse mediante un número limitado de parámetros numéricos (por ejemplo seis coordenadas para definir la posición de la cámara), el sistema de calculo se implementará mediante la ejecución de un proceso en cada máquina disponible; proceso que deberá obtener de alguna manera un juego de parámetros único, calcular la imagen correspondiente a estos parámetros y enviar el fichero de imagen al sistema de postproceso.

Además de este trabajo básico, resulta conveniente que los procesos puedan suspender el trabajo en cualquier momento y reanudarlo posteriormente; también puede resultar interesante, cuando las comunicaciones no son muy fiables, disponer de algún sistema de seguridad que evite que el proceso se quede bloqueado intentando enviar la imagen.

Todas estas cuestiones pueden resolverse con relativa sencillez utilizando los medios disponibles dentro del sistema UNIX. La forma de trabajo consistirá en generar en primer lugar un fichero de texto cuyas lineas contendrán en primer lugar el número de cuadro y a continuación los parámetros que definen el cuadro; a continuación se deberán lanzar los procesos de cálculo, teniendo en cuentra que los parámetros comunes para toda la animación (resolución, precisión de los cálculos y calidad de la imagen, etc.) se fijarán como argumentos de la linea de comandos. Hecho esto, el sistema de cálculo ya está en marcha; sólo resta controlar la carga de trabajo de las máquinas y vigilar la aparición de posibles fallos.

7.3.2. Software.

7.3.2.1. Entrada de datos

Como se ha dicho anteriormente, cada proceso de cálculo recibe los datos necesarios para calcular cada cuadro de un mismo fichero (o de una copia). Se han considerado dos sistemas de trabajo posibles: acceso directo a archivo y utilización de un servidor de datos.

En el modo de acceso directo, debe existir una copia del fichero de datos en cada máquina en que se vaya a trabajar. En la linea de comandos, a la hora de lanzar el proceso, se indicará el nombre del fichero y tres parámetros numéricos:

Número identificador del cuadro inicial a calcular
Número identificador del cuadro final a calcular
Distancia entre cuadros calculados

De ésta manera, el número total de cuadros se divide entre el número de máquinas disponibles; y todas ellas calculan el mismo número de cuadros. Lógicamente a cada proceso le corresponde un parámetro *n* correlativo, siendo el parámetro *s* común a todos ellos.

Este sistema tiene varios inconvenientes:

- Falta de flexibilidad: Una vez iniciado el proceso, resulta sumamente dificil añadir o eliminar máquinas del sistema.
- Si una máquina, debido a su menor carga de trabajo o a su mayor potencia de cálculo termina antes que las demás, queda inactiva, desaprovechándose su capacidad de trabajo.
- La suspensión y reanudación del trabajo implica un cambio en la forma de la linea de comandos (al cambiar el parámetro *n* , lo que dificulta la automatización.
- Diferentes velocidades de cálculo para las diferentes máquinas pueden conducir a una acumulación de ficheros en alguna de las máquinas, lo que puede dar lugar a un colapso del sistema por agotamiento de la cuota de disco. En efecto, si suponemos que las imagenes son procesadas secuencialmente y a continuación borradas del disco, todas las imágenes generadas por la máquina rápida quedan ocupando sitio en disco hasta que las máquinas lentas llenan los huecos existentes.

No obstante este sistema es adecuado cuando sólo se desea calcular un sólo cuadro o un número reducido de cuadros repartido uniformemente por toda la animación en una sóla máquina con objeto de verificar que la secuencia corresponde con lo esperado.

Con el segundo método, la utilización de un servidor de datos, se resuelven todos los problemas anteriormente mencionados. Este sistema consiste en la utilización de un proceso que queda funcionando en una de las máquinas como proceso de fondo, ocupando muy poco tiempo de CPU. Este proceso espera a recibir una petición a través de la red procedente de alguno de los procesos de cálculo, y cuando esto ocurre, lee una linea del fichero de datos y se la envía al proceso que efectuó la petición.

La librería de trazado de rayos utilizada incluye este tipo de servidor para máquinas UNIX, así como de las llamadas necesarias para efectuar las peticiones de datos.

7.3.2.2. Control del proceso

Debido a que las máquinas van a ser compartidas con otros, se requiere que cada uno de los procesos de cálculo pueda ser interrumpido en un momento dado sin perder el trabajo en curso. La librería de trazado de rayos no contempla la posibilidad de interrumpir el trabajo en mitad del cálculo de un cuadro; de modo que la única forma ordenada de realizar la interrupción consiste en parar la ejecución cuando se termina una imagen.

Esto se lleva a cabo mediante un esquema de captura de señales. En cualquier momento, el programa puede recibir una señal que es capturada inmediatamente y provoca el marcado de una variable global de interrupción. Cuando el cálculo de la imagen termina, al encontrar marcada ésta variable, en lugar de continuar calculando la siguiente imagen pasa a modo *sleep* durante un tiempo considerablemente largo (por ejemplo, 36 horas). En este modo de funcionamiento (standar de UNIX), el proceso no consume tiempo de CPU, pero el espacio en disco correspondiente a la memoria que necesita sigue ocupado. En caso de que esto suponga un problema, puede "matarse" el proceso en el momento en que entra en la fase de inactividad; al no haberse interrumpido el cálculo de un cuadro, será sencillo volver a poner en marcha el proceso.

En caso de que se desee volver del modo *sleep* al funcionamiento normal, basta con enviar otra vez la misma señal al proceso. En la figura 7.5 se muestra un ejemplo de este esquema.

```
#include  <stdio.h>
#include  <signal.h>
#include  <unistd.h>

void siesta( int i );
extern  void  make_it(void);
int  sleeping=0;

void     siesta( int i )
{
       signal(_SIGUSR1,siesta);
       sleeping=1;
}

main ( int argc, char **argv )
{

int  done=0;

       signal(_SIGUSR1,siesta);

       while(!done)
       {
              make_it();

              if(sleeping)
             {
                     sleep(129600);
                    sleeping=0;
             }
       }
}
```

Figura 7.5: Tratamiento de señales

7.3.2.3. Transferencia de ficheros

El sistema más sencillo disponible para realizar la transmisión de los ficheros de imagen es el *File Transfer Protocol* . La forma de acceder a este protocolo es el comando *ftp* de UNIX.

Para poder acceder a este comando desde el interior del programa de cálculo se pueden utilizar las llamadas de UNIX de gestión de procesos (*fork, exec*). Una solución sencilla consiste en crear un programa utilizando comandos del *shell*. En este programa se incluirían las llamadas al comando *ftp* , las ordenes que requiere este comando y las posibles actuaciones en caso de fallo (por ejemplo, si todo va bien, borrar el archivo).

De esta forma, el programa de cálculo sólamente deberá ejecutar este fichero de comandos, al que le pasará como parámetro el nombre del fichero a transmitir. Como medida de precaución se puede añadir un sistema de vigilancia, que en caso de que el proceso de transmisión quede bloqueado, lo “mate” para poder seguir calculando la siguiente imagen.

Este sistema de seguridad consiste en lanzar dos procesos en paralelo: uno de ellos será el de transferencia de ficheros mientras que el otro será el comando *sleep* de UNIX. El parámetro del comando *sleep* vendrá dado por el tiempo máximo estimado de transmisión (en segundos); en caso de que este comando termine antes que el de transferencia, se supondrá que éste está bloqueado y se procederá a eliminarlo.

```
/ *
* Ejemplo de control de subprocesos:
*           primer parametro:  Num. maximo de segundos
*           segundo y sucesivos: proceso a ejecutar
* /

#include  <stdio.h>
#include  <errno.h>

extern void syserr( char *men );    /* escribe error & exits */
int launch( char *cmd[], int time );

main(int argc, char **argv )
{
          char  *excmd[2]={NULL,NULL};

          excmd[0]=argv[2];
          if(1==launch(  &argv[2],  atoi(argv[1])))
                    puts("ok");
          else
                    puts("aborted");
}

int launch( char *cmd[], int time )
{
          int  res=1;
          int sync=1, pid, wdog_pid, wpid;
          char   *wdog[3]={"sleep","0000000000",NULL};

          sprintf( wdog[1], "%d", time);
          switch ( pid = fork() )
          {
          case -1:   /*  error  */
                              syserr  (  "fork"  );
          case 0:             if  (  sync  )  execvp  (  cmd[0],  &cmd[0]  );
                              syserr  (  "execvp  sleep"  );
          default:    if  (  sync  )
                              {
                                        if((wdog_pid=fork())==0)
                                                  execvp  (  wdog[0],  &wdog[0]  );
                                        else  if(wdog_pid  ==  -1)
                                                  syserr("fork  wdog");

                                        /*  esperando  un  final  */

                                        while  (  pid  !=(wpid=  wait  (  NULL  ))  )
                                        if  (  wpid  ==  wdog_pid  )
                                        {
                                                  /*  sleep  acaba  antes  */
                                                  res=0;
                                                  kill(  pid,  9  );
                                        }
                                        else
                                                  kill(wdog_pid,9);

                                        wait(NULL);
                                        /* en caso de no esperar, puede
                                        agotarse el numero de procesos
                                        permitidos  */
                              }
          }   /* Fin del    switch */

          return  res;
}
```

Figura 7.6: Ejemplo de utilización de un proceso de control de tiempo

Una posible alternativa a este enfoque sería la de llamar directamente al servidor de ftp desde dentro del programa principal utilizando para ello alguna de las librerías de dominio público existentes (por ejemplo, la de Oleg); sin embargo esta solución tendría mucha menos flexibilidad, por lo que se ha desechado, aunque en algún caso en concreto podría ser preferible (por ejemplo, si se desea realizar un programa comercial completo en si mismo).

Un ejemplo de como prodría ser el subprograma de transferencia sería el que se muestra a continuación:

```
# script para transferencia de ficheros
#VERSION HP-UX, ksh
# Se intenta transferir el fichero pasado como parametro
# $HOST debe tener la direccion del destino

FTPL=`(ftp $HOST << @EOF
bin
put $1
close
bye
@EOF
) `

if [ "$FTPL" = "" ]
then
        echo ok transfer $1
        rm $1 $1.i
else
        echo `date`: ftp failed. file: $1
fi
```

Figura 7.7: Programa en ksh para transferir ficheros

Conviene destacar que la verificación de que la transmisión ha sido correcta en el ejemplo anterior no se realiza mediante la verificación de la variable *$?* , como suele hacerse en *ksh* , si no mediante la verificación de que la salida textual del comando sea nula. Esto debe hacerse así debido a que el comando *ftp* , al ser interactivo, siempre deja dicha variable con valor nulo.

7.3.2.4. Operaciones de postproceso

Hasta ahora hemos visto los sistemas de cálculo y de grabación. Entenderemos como fase de postproceso todas las operaciones que se realicen con las imágenes entre ambos sistemas. En el caso más simple, en el que las imágenes se graben directamente tal y como son calculadas, seguiremos necesitando este sistema, simplemente para tener la posiblidad de recibir varias conexiones simultaneas procedentes de los sistemas de cálculo y mandar ordenadamente, de uno en uno, los cuadros al sistema de grabación. (Lógicamente esto no sería necesario si el sistema de grabación tuviera un servidor de ftp más completo).

No obstante, por lo general será necesario el realizar ciertas operaciones con las imágenes calculadas, principalmente debido a los efectos de discontinuidad en los movimientos y al entrelazado en la señal de video final.

7.2.4.1. Supermuestreo temporal

En la librería de trazado de rayos utilizada, así como en gran parte de los programas de síntesis de imágenes por ordenador, cada cuadro se obtiene a partir de la situación de la escena en un determinado instante de tiempo, al contrario de lo que ocurriría en caso de que la animación fuese filmada con una cámara real, donde cada cuadro se formaría a partir de una integración de dicha situación a lo largo de un intervalo de tiempo no infinitésimal.

La consecuencia de esto es que, a no ser que la accion se desarrolle muy despacio (respecto al periodo de muestreo), la secuencia de imágenes no dará sensación de movimiento.

La solución de este problema consiste en obtener cada cuadro como la media de *n* imágenes (a las que denominaremos *subcuadros*), cada una calculada en un instante determinado t_i

$$t_i = t_0 + \frac{i\,\Delta}{n}\ , \qquad i = 0, 1, \ldots, n\text{-}1$$

donde t_0 es el instante inicial del cuadro y Δ es el intervalo durante el cual se calcula la media, análogo al tiempo durante el cual el obturador está abierto en una cámara. Por lo general se consiguen buenos resultados haciendo Δ igual al intervalo completo entre dos cuadros consecutivos. Si *I* representa a una imagen, se calculará cada cuadro como:

$$I = \frac{1}{n}\sum_{i=0}^{n-1} I(t_i)$$

siendo $I(t_i)$ el subcuadro calculado en el instante t_i.

Esta técnica, denominada muestreo uniforme, puede causar efectos extraños cuando n es muy bajo (o la frecuencia temporal muy alta). Estos efectos pueden reducirse sin aumentar el número de muestras por medio del muestreo aleatorio. Si t_i se determina como:

$$t_i = t_0 + \frac{(i + \delta)\,\Delta}{n}$$

donde δ es una variable aleatoria con distribución uniforme entre [0,1]. De esta manera el efecto de "aliasing" se convierte en una señal de ruido superpuesta, que por lo general es mejor aceptada por los sentidos humanos.

Para realizar éste cálculo, se desarrolló el programa *mergep*.

7.3.2.4.2. Cálculo por campos

En las pantallas de video, todas las lineas de un cuadro no se presentan simultaneamente. En lugar de esto, las lineas se muestran de forma entrelazada: en primer lugar aparecen las lineas 1, 3, 5 … (denominadas el primer campo) y a continuación las lineas 0, 2, 4 … (denominadas segundo campo). Por lo tanto, se puede conseguir un efecto supermuestreo (similar a duplicar la tasa de muestreo) calculando por campos, esto es, de los n subcuadros calculados se toman los primeros n/2 para promediar el campo impar y los siguientes n/2 para calcular el campo par. Si denominamos *F* a los campos:

$$F_1 = \frac{2}{n}\sum_{i=0}^{n/2-1} I\left(t_0 + \frac{(i + \delta)\,\Delta}{n}\right)$$

$$F_2 = \frac{2}{n}\sum_{i=n/2}^{n-1} I\left(t_0 + \frac{(i + \delta)\,\Delta}{n}\right)$$

con lo que la linea k de la imagen I vendrá dada por:

$$I_k = \begin{cases} F_{1k} & \text{si k es impar} \\ F_{2k} & \text{si k es par} \end{cases}$$

Dado que ni las lineas pares de F_1 ni las lineas impares de F_2 se utilizan, no es preciso calcularlas con lo que el tiempo de cálculo no aumentará.

7.3.2.4.3. Filtrado de la imagen

En el video entralazado es frecuente la aparición de un fenómeno de parpadeo ("flicker") cuando existen lineas de una anchura de un pixel de color diferente a las adyacentes. Si una linea de estas características existe sólamente en uno de los dos campos, aparecerá y desaparecerá a una frecuencia de 50 Hz.

Este problema puede resolverse filtrando cada campo según la dirección vertical antes de desechar las lineas no utilizadas. Se pueden conseguir resultados aceptables con un filtro de cinco lineas de ancho con los coeficientes ([§SNYDER])

$$-\frac{1}{8}, \frac{2}{8}, \frac{6}{8}, \frac{2}{8}, -\frac{1}{8}$$

Esto es, cada linea F'_k se calcula a partir de las cinco lineas adyacentes de la imagen original,F_{k-2} a F_{k+2} , según la expresión:

$$F'_k = -\frac{1}{8} F_{k-2} + \frac{2}{8} F_{k-1} + \frac{6}{8} F_k + \frac{2}{8} F_{k+1} - \frac{1}{8} F_{k+2}$$

Los dos subcuadros así calculados se entralazan para formar el cuadro final. En caso de que se utilice este filtro no se consigue el ahorro de tiempo mencionado en el apartado anterior, ya que deben calcularse todas las lineas de cada subcuadro.

7.3.2.4.4. Problemas debidos al trazado de rayos

Cuando el sistema de cálculo se basa en la técnica del trazado de rayos aparece un problema adicional: si el modelo geométrico que se utiliza contiene primitivas cuyas dimensiones relativas a la distancia respecto al punto de vista son reducidas o si se utilizan texturas que contienen lineas de anchura relativamente escasa.

Efectivamente, dado que ésta técnica se basa en el seguimiento de un número finito de rayos de luz desde el punto de vista hasta las primitivas con las que intersectan, la resolución máxima que podremos conseguir (entendiendo por resolución la capacidad de distinguir entre dos puntos diferentes de una determinada superficie) vendrá dada por la relación entre el número de rayos trazados y la distancia existente entre la superficie y el punto de vista.

Dado que en una escena genérica no hay, en principio, limitación en cuanto a la distancia; y que el número de rayos trazados es proporcional al tiempo de cáculo de la imagen, y por lo tanto, limitado, en caso de que se trabaje con objetos como los descritos, fácilmente ocurrirá que algunas de las lineas existentes en las texturas aparecerán en la imagen en algunos casos y en otros no, dependiendo de la posición de la cámara.

Este fenómeno es impredecible en el sentido de que mínimas variaciones en la posición de la cámara dan lugar a que unas lineas aparezcan y otras desaparezcan.

Existen diferentes posibles soluciones a este problema:

- Utilizar una librería de trazado de rayos que realice algún tipo de comprobación de coherencia temporal.
- Utilizar el sistema de interpolación descrito en los apartados anteriores. Esto no resuelve el problema en todos los casos; pero, en algunos casos se consigue mejorar la apariencia de la animación. Los tiempos de cálculo por imagen actuales hacen casi inviable la utlización de más de 4 subcuadros por imagen, con lo que, para que con éste método no aparezcan fenómenos extraños, la velocidad de movimiento de la cámara deberá ser relativamente baja.
- Otra posibilidad es utilizar la misma técnica de interpolación pero con muestreo aleatorio y con un parámetro Δ menor que el intervalo completo entre dos cuadros consecutivos. Con esto se consigue que una mayor coherencia entre las imágenes utilizadas en la interpolación, aunque no mejorará significativamente el fenómeno de discontinuidad del movimiento.
- Por último, puede aumentarse el número de rayos trazados. Se han hecho pruebas multiplicándolo por cuatro; pero la mejora no compensa el aumento en el tiempo de cálculo.

7.4. Notas sobre el ejemplo utilizado

7.4.1. Modelo geométrico

La animación presentada corresponde a un vuelo alrededor del edificio del Centro de Investigaciones Aplicadas, situado en la calle María de Luna, nº 5 (Zaragoza, España).

Los datos se obtuvieron a partir de mediciones realizadas sobre copias de los planos originales del edificio, realizados por el arquitecto Joaquin Magrazo.

Para codificar el modelo geométrico se realizó una librería con las primitivas geométricas y transformaciones necesarias; librería que, implementada de tres formas distintas, actua como interface entre el modelo y tres sistemas de representación: el formato RIB [§PIXAR], el formato Starbase y el formato utilizado por la librería de trazado de rayos.

El formato Starbase permite una visualización rápida del modelo geométrico, con una calidad baja. Utilizando una serie de programas desarrollados dentro del Grupo y un ordenador HP 9000/385 dotado de acelerador gráfico fue posible manipular el modelo geométrico cási en tiempo real, verificar la corrección del mismo y seleccionar posiciones de cámara clave, a partir de las cuales, y por interpolación, se obtuvieron los ficheros con los datos necesarios para realizar la animación.

El formato RIB, permite utilizar software comercial para obtener imágenes de calidad, utilizando librerías de texturas y apariencias superficiales ya existentes, así como traducir el modelo a otros formatos.

El modelo consta de 1957 objetos (en la definición en formato del trazador de rayos, en el que una paralepípedo es considerado como un único objeto). Para realizar la codificación fue necesario, salvo algunos elementos repetitivos, como las filas de columnas, que pueden ser definidos algorítmicamente, escribir en forma de programa en C las transformaciones necesarias para posicionar cada objeto, las propias definiciones de los objetos y la descripción de las características de cada superficie. Esto supone del orden de 3400 lineas de programa, sólo para la descripción del edificio.

7.4.2. Modelo óptico

La descripción de las características de las superficies es la parte que más varía de un sistema de *rendering* a otro. Para el sistema de trazado de rayos, se consideran de distinta forma las superficies "lisas" (de propiedades superficiales uniformes) y el resto de las superficies (de propiedades variables, por ejemplo, la piedra gris de Calatorao, que presenta una superficie gris no uniforme veteada con un color más claro).

En el primer caso se puede describir la superficie mediante la especificación de un número limitado de parámetros relacionados con el color, la reflectividad a distintos tipos de luz y otras propiedades ópticas. En el segundo caso se debe añadir a esta información, toda aquella que dependa de la posición. Esto se puede hacer o bien de forma algorítmica, lo que implica una modificación de la librería de trazado de rayos, o bien mediante lo que se denomina "texturas".

Este sistema consiste en obtener una imagen real del tipo de superficie que se desea conseguir y proyectar esta imagen sobre las primitivas que componen la escena. Para obtener estas texturas se realizaron una serie de fotografías de los materiales realmente utilizados en el edificio. Estas fotografías fueron digitalizadas mediante un scanner de 24 bits de color (lo que significa una capacidad de distinción entre unos 16 millones de colores). Los archivos de imagen así obtenidos fueron retocados con la ayuda de un programa de tratamiento digital de imágenes (para corregir los errores debidos a la perspectiva y las diferencias de iluminación); consiguiéndose así los ficheros definitivo que se utilizarían como texturas.

Por último, se debe describir la iluminación de la escena. Aprovechando las posibilidades ofrecidas por la librería empleada, se definió una fuente de luz con un espectro de radiación similar a la luz del sol; simulando, debido a su posición, la luz del atardecer.

7.4.3. Tiempo de cálculo

La animación presentada como ejemplo consta de 1655 cuadros, correspondientes a algo más de un minuto de video en tiempo real. Los cálculos se realizaron sin emplear ninguna operación de postproceso; con un modelo de iluminación de Hall a la resolución de 768 por 576 pixels.

Se utilizaron 4 ordenadores HP 9000/710 y un HP 9000/735

El tiempo total de cálculo (incluyendo paradas del sistema durante las horas diurnas) fué de aproximadamente 10 días.

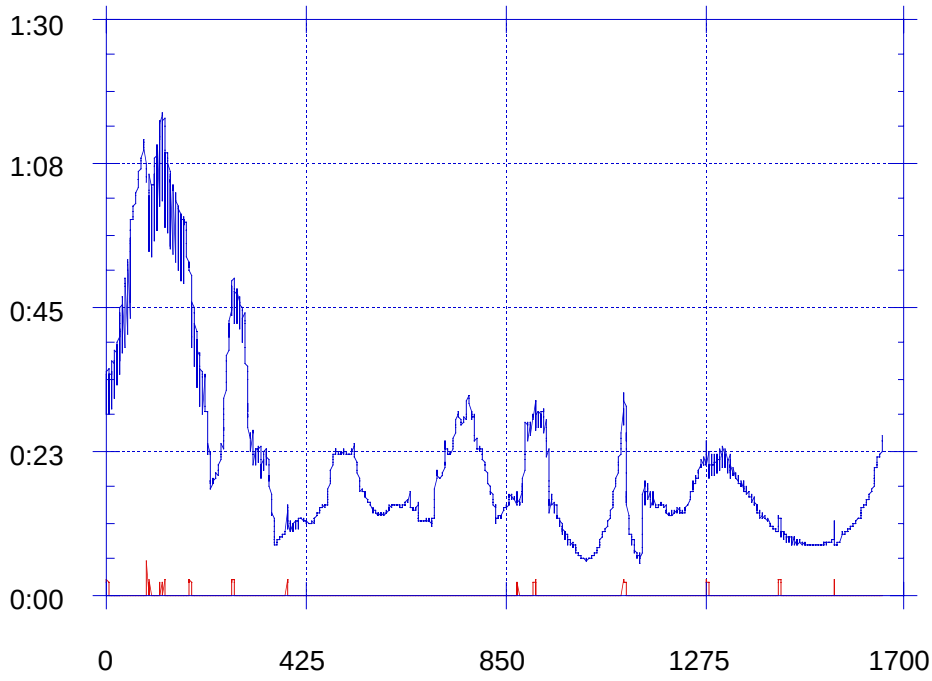

Figura 7.8: Tiempo de cálculo

En ésta gráfica se muestra el tiempo de cálculo por cuadro (horas:minutos), detallándose (trazado inferior) los tiempos de preproceso. Estos tiempos de preproceso corresponden a la interpretación geométrica de los datos que componen la escena; al ser estos constantes para toda la animación, sólo deberán interpretarse al comienzo de cada proceso (esto es, los tiempos de preproceso no nulos corresponden a las paradas del sistema de cálculo en alguna de las máquinas).

Como se puede observar, los tiempos totales varían en un cierto margen entre cuadros contiguos; esto es debido a la existencia de dos tipos de máquinas diferentes y a las diferentes cargas de trabajo que soportan.

Por otra parte las variaciones ‘macroscopicas’ en el tiempo de cálculo se deben a las diferencias en los contenidos de las distintas imágenes. Debido al sistema de cálculo, una imagen en la que predomine un fondo vacío se requiere menos cálculos de intersección que una en la que una gran variedad de objetos cubra la zona visible.

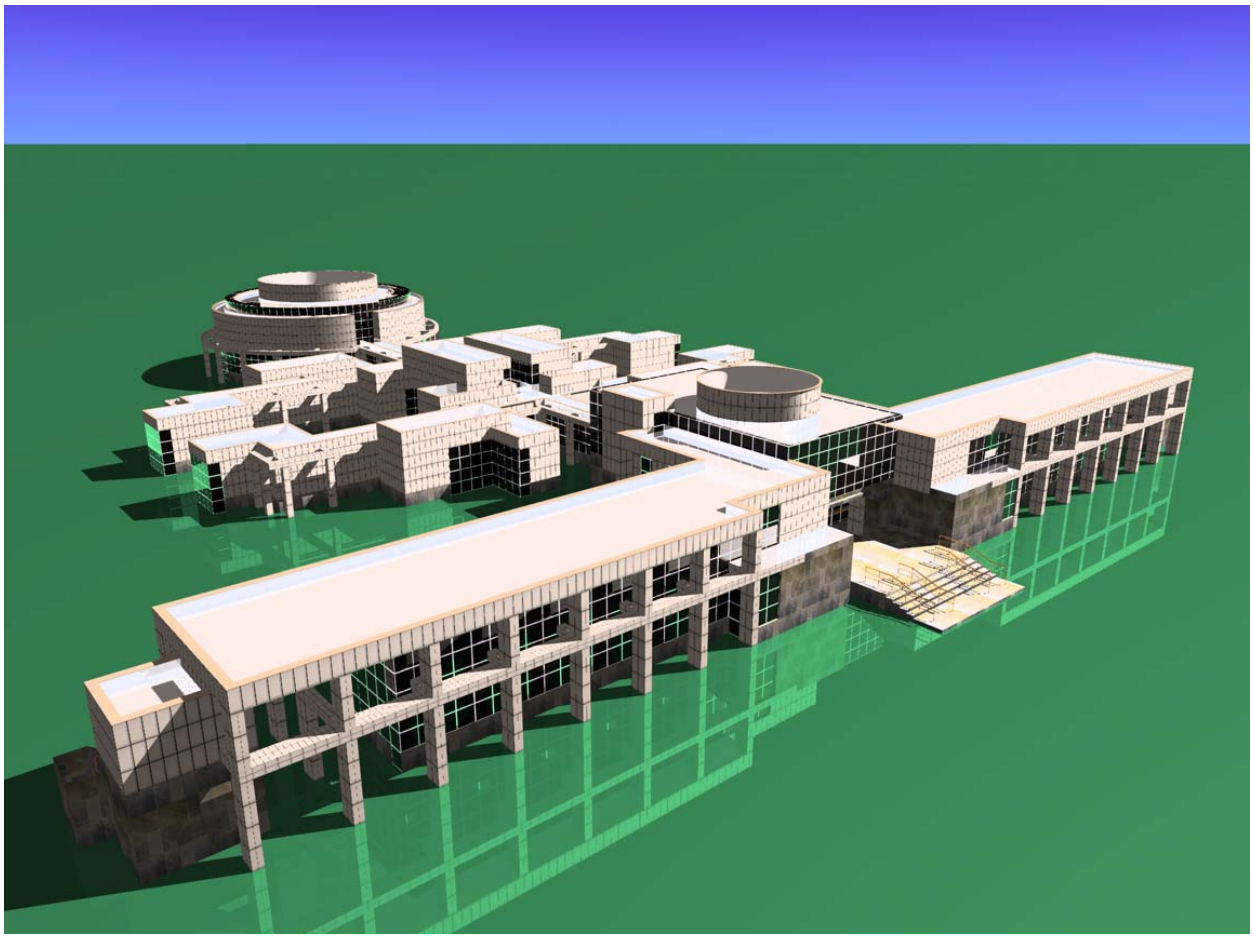

Figura 7.9: Una de las imágenes generadas