Despliegue de Imágenes en Paralelo a Través de Dos Monitores, Utilizando Dos Tarjetas de Video Modificadas

Héctor Samuel García Salas
[hsalas@vmredipn.ipn.mx]
Profesor e Investigador del CIDETEC-IPN.
Marco Antonio Ramirez Salinas
[mramirez@vmredipn.ipn.mx]
Profesor e Investigador del CIC-IPN.
Aquilino Cervantes Avila
[a@vmredipn.ipn.mx]
Alumno del CIC-IPN.

Esta investigación forma parte de un proyecto para el desarrollo del laboratorio de realidad virtual en el CIDETEC. El objetivo de este trabajo es el desarrollo de un método de presentación de imágenes sincronizadas, a través de dos dispositivos de despliegue (en este caso, dos monitores). En este artículo se habla de la circuitería y la programación modificadas y desarrolladas para llevar a cabo esta tarea.

Ya que las computadoras personales son la herramienta principal para los profesores y estudiantes del CINTEC, y desde luego en muchos otros lugares, el desarrollo de un laboratorio de realidad virtual permitará que las simulaciones se desarrollen más facilmente por estas personas. De esta forma, para el desarrollo de un "motor de realidad virtual" es necesario el tener una herramienta de visualización para realizar experiencias.

Para conseguir lo anterior se comenzó por adecuar un segundo video a una computadora personal.

Antecedentes

La utilización de dispositivos estereoscópicos de visualización es, hoy en día, una de las más importantes herramientas para las aplicaciones de simulación en cualquiera de las disciplinas del conocimiento humano, además de ser, desde luego, uno de los medios utilizados por los juegos en computadora.

Las técnicas de tratamiento de imágenes y el desarrollo de las nuevas tecnologías en circuitería electrónica (hardware), que permiten altos grados de integración (lo cual influye en la disminución de tiempos de procesamiento y aumento en la potencia de manipulación de datos), permiten actualmente el desarrollo de aplicaciones como la realidad virtual, que puede ser adaptada a una gran cantidad de situaciones de simulación de eventos.

En nuestro país, el desarrollo de tecnologías dedicadas a aplicar las técnicas de la realidad virtual no solo no es relevante sino que no existe. Por ello, este proyecto (del cual se presentan a continuación los resultados) es una de las partes que conforman la línea de investigación sobre realidad virtual que se llevó a cabo en su momento el CINTEC y ahora es retomada en el CIDETEC.

Introducción

Con el fin de obtener efectos 3D de los ambientes virtuales, se han desarrollado dispositivos que permiten de una o de otra forma, el presentar al usuario imágenes que en conjunto proporcionan la sensación de encontrarse en un ambiente tridimensional, donde existe la profundidad, la altura y lo ancho; además, si a esto se auna un efecto de movimiento, el usuario puede confundir facilmente un ambiente virtual (creación abstracta, matemática, etc.), con una situación real (aunque todavía la potencia necesaria de la computadoras y los algoritmos no permite simular a un grado tal que sea la representación exacta de la realidad).

Ahora bien, uno de los métodos para producir un efecto tridimensional en el ser humano es la utilización de imágenes estereoscópicas, es decir, imágenes que son vistas con ángulos diferentes correspondientes a los puntos de vista de los ojos de un ser humano. Esto implica una renderización* para cada uno de los ojos y una presentación independiente para cada uno, por lo cual, una de las técnicas es la de presentar simultáneamente y en pantallas separadas, a una cierta distancia (desde luego con una óptica adecuada), las imágenes correspondientes a cada ojo.

***Nota:** Para efectos de este artículo, se maneja el término "renderizar"como la obtención de una imagen a partir de un conjunto de datos, usando un algoritmo específico.

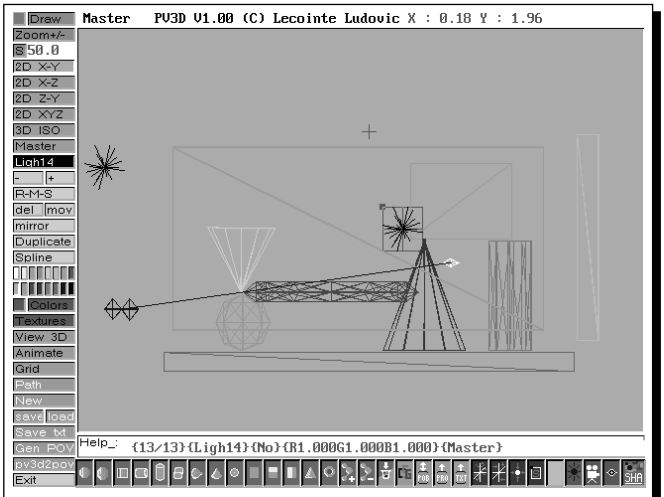

Figura 1. Vista lateral del escenario virtual.

Lo anterior implica un desarrollo de hardware capaz de utilizar simultáneamente dos dispositivos de salida para imagen, en este caso, se utilizan dos tarjetas de vídeo VGA modificadas, y un software que permite mapear la información en la memoria extendida de una computadora personal (PC) y presentar las imágenes en dos monitores.

Imágenes

Las imágenes utilizadas se calcularon con el programa POVRAY, y la paquetería 3DVG.

En un principio los escenarios se diseñaron utilizando un paquete constructor de objetos 3D. La **figura 1** muestra el plano X-Y de la construcción del escenario.

Las **figuras 2** y **3** muestran, respectivamente, el plano X-Z de los escenarios construidos para el ojo Izquierdo y Derecho (esto se puede notar en el punto donde se encuentra situado el observador, el doble diamante en la esquina inferior izquierda)

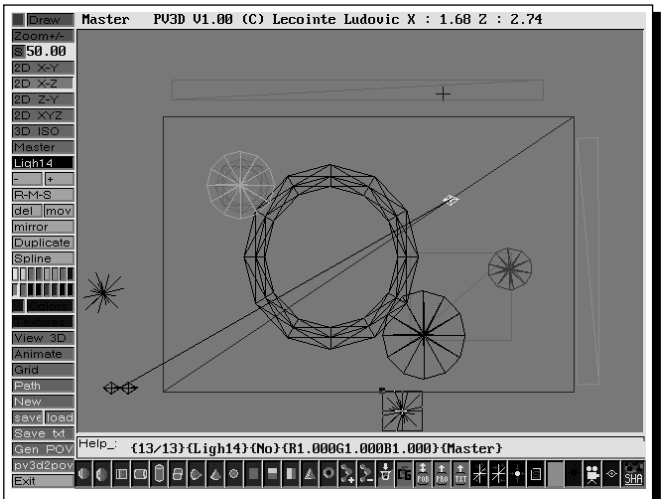

Figura 2. Vista del plano X-Z del escenario virtual. La posición del observador corresponde al ojo izquierdo.

La **figura 4** muestra las imágenes "renderizadas" del escenario 3D para el ojo izquierdo. El modelo que sirvió para calcular esta imagen es el mostrado en la **figura 2**.

La **figura 5** muestra la imagen renderizada del escenario 3D para el ojo derecho. El modelo que sirvió para calcular esta imagen es el de la **figura 3**. Es posible que no se alcance a apreciar la diferencia entre la imagen 4 y la 5, ya que el ángulo entre los ojos es muy pequeño; sin embargo, es el necesario para que el cerebro humano perciba las características 3D.

Para obtener el efecto estereoscópico se debe presentar a cada ojo su respectiva imagen. Con la respectiva óptica, el cerebro se encarga de mezclar la imágenes para formar una

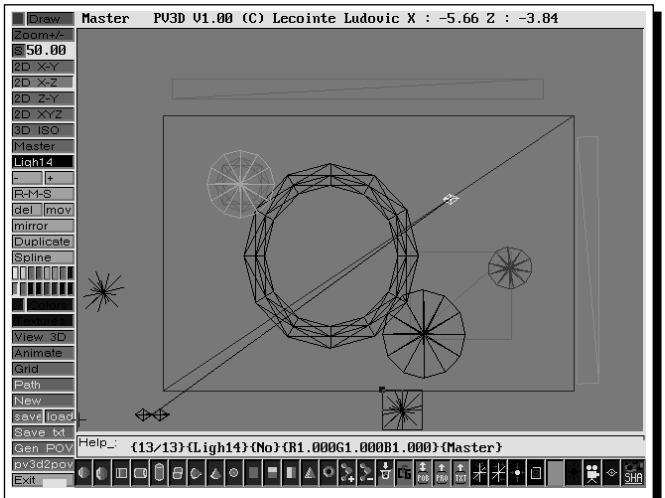

Figura 3. Vista del plano X-Z del escenario virtual, la posición del observador corresponde al ojo derecho.

sola imagen tridimensional. Algunas técnicas de visualización (como son los estereogramas) mezclan dos imágenes (una para cada ojo) poniendo columnas de cada imágen una al lado de la otra, de esta forma el observador haciendo un esfuerzo puede logra ver la escena en tres dimensiones. Hay que hacer notar que, según la literatura al respecto, existe un porcentaje de personas que son incapaces de observar este tipo de escenas.

En la **figura 6** se han colocado las imágenes (**figuras 4** y **5**) de tal

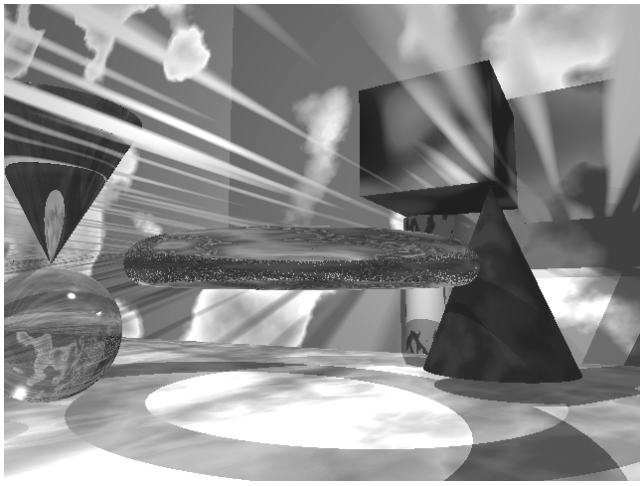

Figura 4. Imagen renderizada a partir del escenario de la figura 2.

manera que, con un poco de práctica, el observador puede apreciar la escena tridimensional. Desde luego, esto sería más fácil si se utilizara un casco de realidad virtual.

Este tipo de imágenes fueron utilizadas para realizar pruebas con el hardware y software que hemos desarrollado y utilizado en nuestra investigación.

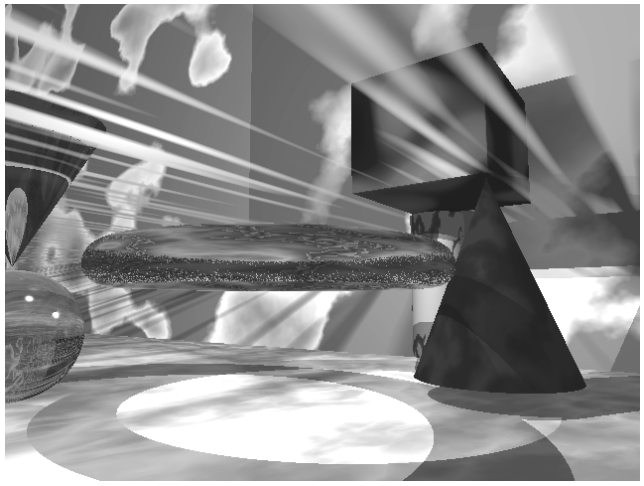

Figura 5. Imagen renderizada a partir del escenario de la figura 3.

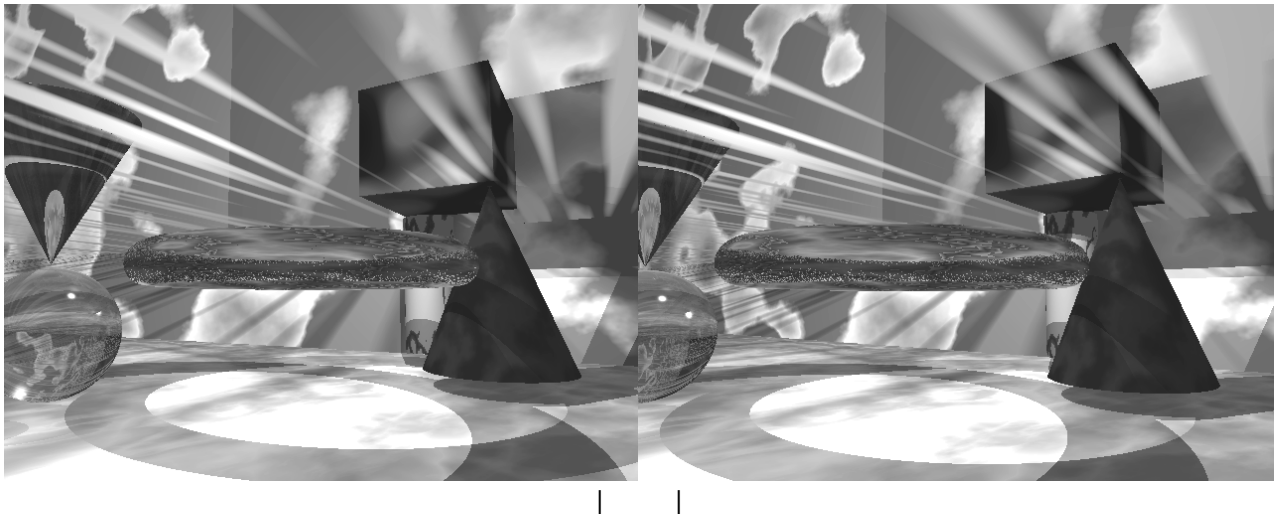

Figura 6. Efecto estereoscópico. Fijar cada imagen respectivamente con un ojo, tratar de formar una sola imagen, entonces se puede apreciar el escenario en 3D. (utilizar las guias verticales).

Desarrollo de Circuitería

Adecuación de un segundo subsistema de video en formato vga para cualquier pc/at compatible

Se tiene la necesidad de despliegue de dos imágenes diferentes al mismo tiempo. Como ya se mencionó, esto es con el fin de desarrollar la herramienta y el software necesarios para integrar un motor de realidad virtual, el cual forma parte de un laboratorio de realidad virtual. Los dos depligues de video son para simular un "casco". El casco de realidad virtual es un sistema que permite obtener el efecto de imágenes en tercera dimensión, y de esta forma apreciar la simulación de los ambientes virtuales.

Estudio del caso

Dentro de la arquitectura de las PC´s compatibles se ha utilizado el primer Megabyte para direccionar subsistemas, tales como: 640 KByte como memoria base, 128 KByte como RAM de Video, 128 KByte como ROM reservada para tarjetas de periféricos y 128 Kbytes para bios de video y sistema (**Figura 7**).

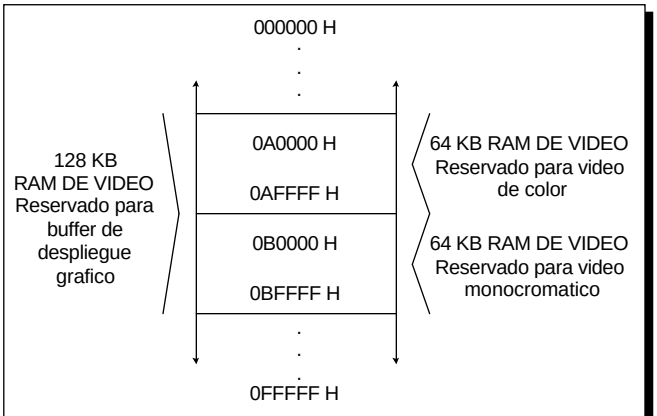

Figura 7. Ubicación del video dentro del 1er. Mbyte de memoria del sistema.

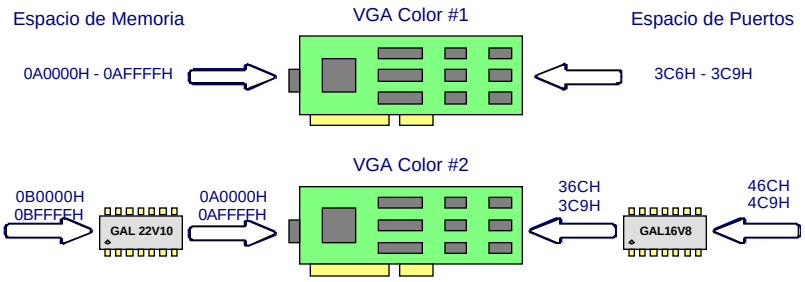

Figura 8. Como son mapeadas las direcciones para cada tarjeta.

Una tarjeta de video es direccionada en 64 KBytes de los 128 KBytes del bufer de despliegue gráfico, dependiendo del tipo de video detectado por el bios de la máquina.

Dado que el bios detecta únicamente una sola tarjeta de video a color, los 64 KBytes restantes quedan libres; entonces es posible direccionar otra tarjeta de video a color en ese espacio de memoria mediante un "arreglo sencillo".

Como requisito primordial se requieren tarjetas de la misma marca y modelos idénticos, esto es para que al momento de encender la maquina no existan conflictos en la inicialización de ambas tarjetas.

El "Arreglo" mencionado anteriormente consiste en permitir la inicialización simultánea de ambas tarjetas, pero separar las imágenes que se escriben a cada subsistema de video.

Los gráficos en el subsistema de video VGA a color se escriben en bloques de 64 KBytes a partir de la dirección 0A0000H hasta la dirección 0AFFFFH. En el 2º video VGA a Color los gráficos también se tienen que escribir en bloques de 64 Kbytes a partir de la dirección 0B0000H hasta la dirección 0BFFFFH, pero en este caso hay que hacer "creer" a la tarjeta del 2º video que se esta escribiendo al bufer natural (0A0000H-0AFFFFH), esto se logró utilizando una GAL22V10 con un programa que hace la conversión de las direcciones (**Figura 8**).

Por otro lado, el DAC de video esta atado regularmente a los puertos 3C6H a 3C9H, mientras que en la 2a tarjeta; tuvo que aislarse del DAC de la primera, utilizando un arreglo en una GAL16V8. En seguida se muestran los programas para los PLD´s.

Programas

A continuación se incluye el programa para mapear la memoria de la computadora personal; este programa escribe, en la dirección de memoria correspondiente a cada tarjeta de video, las líneas correspondientes a cada imagen.

```
BEGIN HEADER
 Este diseño mapea una tarjeta de video VGA color (0A0000-0AFFFF)
en el área
 de video monocromático (0B0000-0BFFFF), pero el BIOS VGA es
mapeado en la
 misma dirección (E0000-EFFFF), usando tabla de verdad.
END HEADER

BEGIN DEFINITION
  DEVICE        G22V10;
  INPUTS        A16=2,A17=3,A18=4,A19=5;
 OUTPUTS (COM)  /B19=20,/B18=21,/B17=22,/B16=23;
END DEFINITION

BEGIN TRUTH_TABLE

  TTIN  A19,A18,A17,A16;
  TTOUT /B19, /B18, /B17, /B16;
  1011 1010
  1110 1110

END TRUTH_TABLE


Programa para mapear  una tarjeta de videa en la segunda área de
memoria.

BEGIN HEADER

 Este diseño mapea el DAC de una tarjeta de video VGA color (03XXX)
en el área de puertos(04XXX), usando tabla de verdad.

END HEADER

BEGIN DEFINITION

  DEVICE        G22V10;
  INPUTS        A11=2,A10=3,A9=4,A8=5,IOW=6,SMW=7;
  OUTPUTS   (COM) /V11=18,/V10=19,/V9=20,/V8=21;
  FEEDBACK  (REG)  BIOS=17;

END DEFINITION

BEGIN TRUTH_TABLE

  TTIN  A11,A10,A9,A8,BIOS,IOW,SMW;
  TTOUT /V11,/V10,/V9,/V8,BIOS;

  0000010  00000
  0001010  00010
  0010010  00100
  0011010  00110
  0100010  01000
  0101010  01010
  0110010  01100
  0111010  01110
  1000010  10000
  1001010  10010
  1010010  10100
  1011010  10110
  1100010  11000
  1101010  11010
  1110010  11100
  1111010  11110
  0000001  00000
  0001001  00010
  0010001  00100
  0011001  00110
  0100001  01001
  0100101  00111
  0000101  00000
  0001101  00010
  0010101  00100
  0011101  00110

END TRUTH_TABLE
```

A continuación se presenta el listado del programa que se usó para su demostración:

NFRG.CPP

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <alloc.h>
#include <mem.h>
#include <string.h>
#include <dos.h>

#include «imagepcx.h»
#include «archivo.h»

void setvgapalette(char *, int, int);

unsigned char *screentable[481];
unsigned char *table[481];

void main(int argc,char **argv)
{
 IMAGEHEAD ih,im;
 FILE *source,*FileFrg;

 unsigned int i,j;
 char _b[20];

 if(argc<=2) {
   printf(«\n Error se necesita un archivo FRG «);
   exit(1);
  }

 strcpy(_b,argv[1]);
 strcat(_b,».frg»);
 AbreArchBinLec(source,_b);
 LeeArch((char *)&ih,sizeof(IMAGEHEAD),source,_b);

 if(memcmp(ih.sig,IMAGESIG,4)) {
     printf(«\n Error no es archivo FRG «);
     exit(1);
     }

 fseek(source,(unsigned long)ih.palette,SEEK_SET);
   if(fread(palette0,RGB_SIZE,1<<ih.bits,source) != (1<<ih.bits)) {
     printf(«\n Error no se puede leer la paleta «);
     exit(1);
       }

 fseek(source,(unsigned long)ih.image+4,SEEK_SET);

 strcpy(_b,argv[2]);
 strcat(_b,».frg»);
```

```
AbreArchBinLec(FileFrg,_b);
LeeArch((char *)&im,sizeof(IMAGEHEAD),FileFrg,_b);

if(memcmp(im.sig,IMAGESIG,4)) {
    printf(«\n Error no es archivo FRG «);
    exit(1);
    }

fseek(FileFrg,(unsigned long)im.palette,SEEK_SET);

  if(fread(palette1,RGB_SIZE,1<<im.bits,FileFrg) != (1<<im.bits)) {
   printf(«\n Error no se puede leer la paleta «);
   exit(1);
     }

fseek(FileFrg,(unsigned long)im.image+4,SEEK_SET);

/*Termina de verificar si el archivo es FRG*/

 for(i=0; i<480; ++i){
   screentable[i] = (char *)MK_FP(0xa000,i*80);
    table[i] = (char *)MK_FP(0xb000,i*80);
    }


 asm mov ax,0x012;
 asm int 0x10

setvgapalette(palette0,ih.bits,0);
setvgapalette(palette1,im.bits,1);

 for(i=0; i<ih.depth; ++i){
   if(i>=480)
     break;

   EGAWRITEPLANE(8);
    if(fread(screentable[i],1,ih.planebytes,source) != ih.planebytes)
       break;

   EGAWRITE(8);
    if(fread(table[i],1,im.planebytes,FileFrg) != im.planebytes)
       break;

   EGAWRITEPLANE(4);
    if(fread(screentable[i],1,ih.planebytes,source) != ih.planebytes)
       break;

   EGAWRITE(4);
    if(fread(table[i],1,im.planebytes,FileFrg) != im.planebytes)
       break;



EGAWRITEPLANE(2);
    if(fread(screentable[i],1,ih.planebytes,source) != ih.planebytes)
       break;

   EGAWRITE(2);
    if(fread(table[i],1,im.planebytes,FileFrg) != im.planebytes)
       break;

   EGAWRITEPLANE(1);
    if(fread(screentable[i],1,ih.planebytes,source) != ih.planebytes)
       break;


   EGAWRITE(1);
    if(fread(table[i],1,im.planebytes,FileFrg) != im.planebytes)
       break;

    }

   EGAWRITE(15);
   EGAWRITEPLANE(15);

 getch(); getch();

fclose(source);
fclose(FileFrg);

for(i=0;i<480;i++)
 memset(table[i],0,80);

 asm mov ax,0x03
 asm int 0x10


}


void setvgapalette(char *p , int n, int k)
{
 union REGS r;
 int i,j;

 j=1<<n;
 if(k==0){

  outp(0x3c6,0xff);
   for(i=0; i<j; ++i){
   outp(0x3c8,i);
   outp(0x3c9,(*p++)>>2);
   outp(0x3c9,(*p++)>>2);
   outp(0x3c9,(*p++)>>2);
     }
   if(n<=4){
   r.x.bx = 0x0000;
   for(i=0; i<j; ++i){
      r.x.ax =0x1000;
      int86(0x10,&r,&r);
      r.x.bx += 0x0101;
    }
  }
  }
else{
  outp(0x4c6,0xff);
   for(i=0; i<j; ++i){
   outp(0x4c8,i);
   outp(0x4c9,(*p++)>>2);
   outp(0x4c9,(*p++)>>2);
   outp(0x4c9,(*p++)>>2);
     }
   if(n<=4){
   r.x.bx = 0x0000;
   for(i=0; i<j; ++i){
      r.x.ax =0x1000;
      int86(0x10,&r,&r);
      r.x.bx += 0x0101;
    }
  }
 }
}
```

Se incluyen las siguiente librerias;

IMAGEPCX.H

```
#define RGB_RED   0
#define RGB_GREEN 1
#define RGB_BLUE  2
#define RGB_SIZE  3

#define IMAGESIG  «IFRG»

#define HIGHRED(c)   ((((c) >> 10) & 0x1f) << 3)
#define HIGHGREEN(c) ((((c) >> 5) & 0x1f) << 3)
#define HIGHBLUE(c)   (((c)  & 0x1f) << 3)


#define PIXELS2BYTES(n) ((n+7)/8)
#define EGAWRITEPLANE(n) { outp(0x3c4,2); outp(0x3c5,n); }
#define EGAWRITE(n) { outp(0x400,0); outp(0x4c4,2); outp(0x4c5,n);
outp(0x500,0);}
#define  GREYVALUE(r,g,b)  (((r*30)/100) + ((g*59)/100) + ((b*11)/
100))

#define HIGHCOLOUR(r,g,b) (((((unsigned int)r >>3) & 0x1f) >> 10)+\
          ((((unsigned int)g >>3) & 0x1f) <<  5)+\
           (((unsigned int)b >>3) & 0x1f))
typedef unsigned char BYTE;
typedef unsigned int  WORD;

typedef struct{

 char manufacturer;
 char version;
 char encoding;
 char bits_per_pixel;
 int  xmin,ymin;
 int xmax,ymax;
  int  hres;
  int  vres;
 char palette[48];
 char reserved;
 char colour_planes;
 int  bytes_per_line;
 int  palette_type;
 char filler[58];

 }PCXHEAD;

typedef struct{

 char sig[4];
 unsigned int width,depth,bits,bytes,planebytes;
 unsigned long palette;
 unsigned long image;

 }IMAGEHEAD;

typedef struct {
   unsigned char Red;
   unsigned char Green;
   unsigned char Blue;
}ColorRegister;
```

```
char pcxpalette[48] = {
        0x00,0x00,0x0e,0x00,0x52,0x07,0x2c,0x00,
        0x0e,0x00,0x00,0x00,0xf8,0x01,0x2c,0x00,
        0x85,0x0f,0x42,0x00,0x21,0x00,0x00,0x00,
        0x00,0x00,0x6a,0x24,0x9b,0x49,0xa1,0x5e,
        0x90,0x5e,0x18,0x5e,0x84,0x14,0xd9,0x95,
        0xa0,0x14,0x12,0x00,0x06,0x00,0x68,0x1f
           };

char palette0[768];
char palette1[768];
char masktable[8] = {0x80,0x40,0x20,0x10,0x08,0x04,0x02,0x01};
char bittable[8] = {0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80};
```

ARCHIVO.H

```
#include<stdio.h>
#include<stdlib.h>
#include <conio.h>

/*****************************************************************
      Macros de manejo de archivos
*****************************************************************/

/* AbreArchBinEsc(FILE *Archivo, Char *NombreArchivo)
// AbreArchBinLec(FILE *Archivo, Char *NombreArchivo)
// CierraArch(FILE *)
// LeeArch(char *BufferLectura, int BytesPorLeer, FILE *Archivo, char
* Tipo de error)
// EscribeArch(char *BufferLectura, int BytesPorLeer, FILE *Archivo,
char * Tipo de error)
// VerSiExiste(FILE *Archivo, Char *NombreArchivo);
*/

#define AbreArchBinEsc(a,b) if((a=fopen(b,»wb»))==NULL)
   { printf(«\n No se puede abrir archivo %s «,b); exit(1); }
#define AbreArchBinLec(a,b) if((a=fopen(b,»rb»))==NULL)
   { printf(«\n No se puede abrir archivo %s «,b); exit(1); }
#define CierraArch(a) fclose(a);
#define LeeArch(a,b,c,d)  if(fread(a,1,b,c) != b)
   { printf(«\n Error %s «,d); exit(1); }
#define EscribeArch(a,b,c,d)  if(fwrite(a,1,b,c) != b )
    { printf(«\n %s «,d); exit(1); }
/* #define VerSiExiste(a,b) if((a=fopen(b,»rb»))==NULL)
    { return(0) }*/

/*****************************************************************/
```

Bibliografía

[1] Joseph Graderick, *"Réalité Virtuelle"*, SYBEX, Paris, 1994.

[2] Ron Wodaski, *"La réalité Virtuelle"*, SYBEX, Paris, 1994.

[3] Robert Sedgewick, *"Algorithms in C"*, ADDISON-WESLEY, New York, 1990.

[4] Ludovic Lecoint, *"PV3D un modeleur d'images de synthèse"*, Paris, 1993.

[5] Jean-Pierre Couwenberg,*"La synthèse d'images, du réel au virtuelle"*, MARABOUT, Paris, 1995

[6] Richard F. Ferraro, *"Programmer's Guide to the EGA, VGA and Super VGA cards"*, ADDISON-WESLEY, New York, 1995