Manejo de Objetos en un Editor Gráfico en Tres Dimensiones

Ing. Amadeo Argüelles Cruz
Alumno de la Maestría del CINTEC-IPN.
Ing. Rubén García Duana
Alumno de la Maestría del CINTEC-IPN.

El presente artículo describe un programa para el manejo de imágenes de objetos en 3 dimensiones. Como el tema es bastante extenso, se dan solo los principios del manejo de éstos y se ejemplificará con algunas figuras de fácil comprensión. Se consideran en este trabajo, los siguientes puntos:

Generación de un numero restringido de objetos tridimensionales en memoria.

Representación gráfica de los objetos en la pantalla.

Implantación de operaciones básicas como son **Mover** y **Rotar** objetos en función de las coordenadas X, Y & Z.

Manejo de un ambiente amigable mediante la aplicación de pantallas gráficas de interacción con el usuario, como lo son menús, cajas de dialogo, etc.

Lectura y escritura de archivos de almacenamiento de los gráficos elaborados.

Impresión de las imágenes en una impresora de matriz de puntos.

Filosofía

Para la elaboración del programa, se optó por implementar un ambiente gráfico utilizando lenguaje C++ orientado a objetos. Es importante hacer notar que el programa fue elaborado en un ambiente DOS y no para Windows, debido a que la generación de un ambiente gráfico propio nos permitirá dominar la técnica de manejo de ventanas, menús, etc.; elementos que por estar definidos para Windows, restringen su utilización solo a hacer un llamado a sus funciones. Al ser independiente de la plataforma Windows, este paquete es fácilmente exportable ; además, al ser escrito para DOS resulta muy compacto, lo que permite ser ejecutado con características muy ventajosas.

La solución al problema está dividida en cuatro áreas:

1) Creación del ambiente gráfico.

2) Generación y representación de objetos tridimensionales.

3) Tratamiento de los objetos en tres dimensiones.

4) Manejo de E/S de la información generada.

A continuación se desglosarán las soluciones para cada una de las áreas:

Creación del Ambiente Gráfico

Dibujar la pantalla principal:

Se inicializa el modo gráfico y se dibuja el marco de la pantalla, se colocan los encabezados y la barra de menús.

Acceso a los menús:

Los menús se invocan presionando la tecla «Alt» y la tecla asociada para cada uno de ellos, dicha tecla es la letra mayúscula de las opciones de los menús; la alternativa <Alt>X para salir también esta habilitada.

Generación de una caja de dialogo general para todos los casos:

Para las opciones del menú se ideó una caja de dialogo general, mediante una función que lleva como parámetro un apuntador al titulo de la caja; a esta se añadió una función general que permitiera la selección de cualquier opción del menú por medio de las flechas, desde dos hasta nueve selecciones como máximo.

Generación y representación de objetos tridimensionales.

Generar objetos tridimensionales:

El cubo se genera dando la posición espacial de uno de los vértices, y la longitud de los lados del cubo en

cada una de las coordenadas; cuando proporcionamos estos datos, se ejecuta una función que genera los siete vértices restantes del cubo, y estos quedan grabados para cada objeto.

El cilindro lee las coordenadas de la base en las tres dimensiones, su longitud, su radio y la dirección del objeto sobre uno de los ejes. La función que dibuja al cilindro necesita generar dos circunferencias, una para la tapa y la otra para la base; se desarrollaron dos métodos para dibujar las circunferencias: uno es dibujándola punto a punto, y el otro generando un polígono de 20 vértices. La creación del objeto incluye los 40 vértices que generan al cilindro (20 para cada circunferencia).

La pirámide se crea con la misma filosofía que los objetos anteriores: la rutina lee las coordenadas de la punta, la altura y la dirección de la pirámide, el número de lados de la base (máximo 20) y el radio de la circunferencia en la que se encontrará inscrito el polígono de la base. Dicho programa calcula las posiciones de los vértices de la pirámide.

Dibujar objetos tridimensionales en pantalla:

Para el problema de dibujar los objetos tridimensionales en pantalla, se consideraron las coordenadas X & Y perpendiculares con el eje Z a 135 grados de ambos. La forma de conversión de coordenadas espaciales a planas se hace de la siguiente manera: Las coordenadas X & Y en 3 dimensiones, corresponden unitariamente a las coordenadas X & Y planas; la coordenada Z espacial se descompone en 2: una proyección en X, dada por su componente a 135 grados (usando el coseno), y una proyección en Y dada por su proyección en Y (usando el seno). Hay que recordar que en las coordenadas en pantalla, se tiene el origen (0,0) en la esquina superior izquierda y que Y crece hacia abajo, es decir, en sentido inverso al plano cartesiano propio; es por esto que en la conversión de coordenadas planas a pantalla, se debe mover el origen e invertir el sentido de las componentes Y.

Tratamiento de los objetos en tres dimensiones.

Girar objetos «base» en 3 dimensiones:

Para rotar un punto, ésto se efectúa en el espacio tridimensional y sobre los ejes X, Y & Z. Primero se calcula el vector del punto, es decir, su magnitud tomada desde el origen y su ángulo proyectado con respecto a los ejes; los ángulos obtenidos se incrementan con los ángulos de rotación del punto. Se calculan las nuevas proyecciones del vector en cada uno de los ejes, obteniendo con esto las nuevas coordenadas del punto rotado. Una vez girado el punto, se procesa como cualquier otro, para ser desplegado en pantalla.

Generación de círculos en 3 dimensiones:

Las circunferencias se calculan por medio de las ecuaciones trigonométricas del circulo, es decir, la función seno en cada coordenada (solo 2 de ellas) defasadas 90 grados entre si. La generación de los circunferencias utilizando este método proporciona una gran resolución, pero tiene como desventaja el ser demasiado lento el cálculo de todos los puntos. Este detalle es muy notorio en máquinas lentas; para corregir éste inconveniente, se ideó la forma de dibujar una circunferencia de baja resolución, ésto se logra produciendo un polígono regular con número alto de lados, lo cual a la vista da la sensación de una circunferencia; el número de lados utilizados es de 20, dado como valor fijo. La resolución utilizada para las circunferencias es conmutable por el usuario.

Manejo de Entrada/Salida (E/S) de la información generada.

Lectura y escritura de archivos:

Para la escritura del archivo, se graba un entero que contiene el número de cubos, e inmediatamente todos los objetos cubos; después se graba otro entero, pero ahora con el número de cilindros, seguido de todos los objetos cilindro; finalmente para las pirámides se realizo el mismo procedimiento. Para la lectura, se lee el primer carácter del archivo que es el número de cubos, y se realiza un ciclo de creación y lectura del todos los objetos cubos; se repite este procedimiento para los cilindros y las pirámides.

Impresión de archivos:

Para lograr habilitar esta función, se utiliza de la interrupción 05h del Bios, la cual realiza una impresión de la vista actual de la pantalla. Para ejecutar dicha interrupción, se debe limpiar la pantalla y solo dibujar los objetos, con el fin de no imprimir el marco y la barra de menús. Hay que recordar que para que la interrupción 05h pueda ejecutar la impresión en modo gráfico, es necesario cargar previamente el programa del sistema operativo llamado GRAPHICS.COM.

Algoritmo

Tomando en cuenta los elementos de este análisis , el paquete desarrollado efectúa las operaciones de acuerdo al siguiente algoritmo:

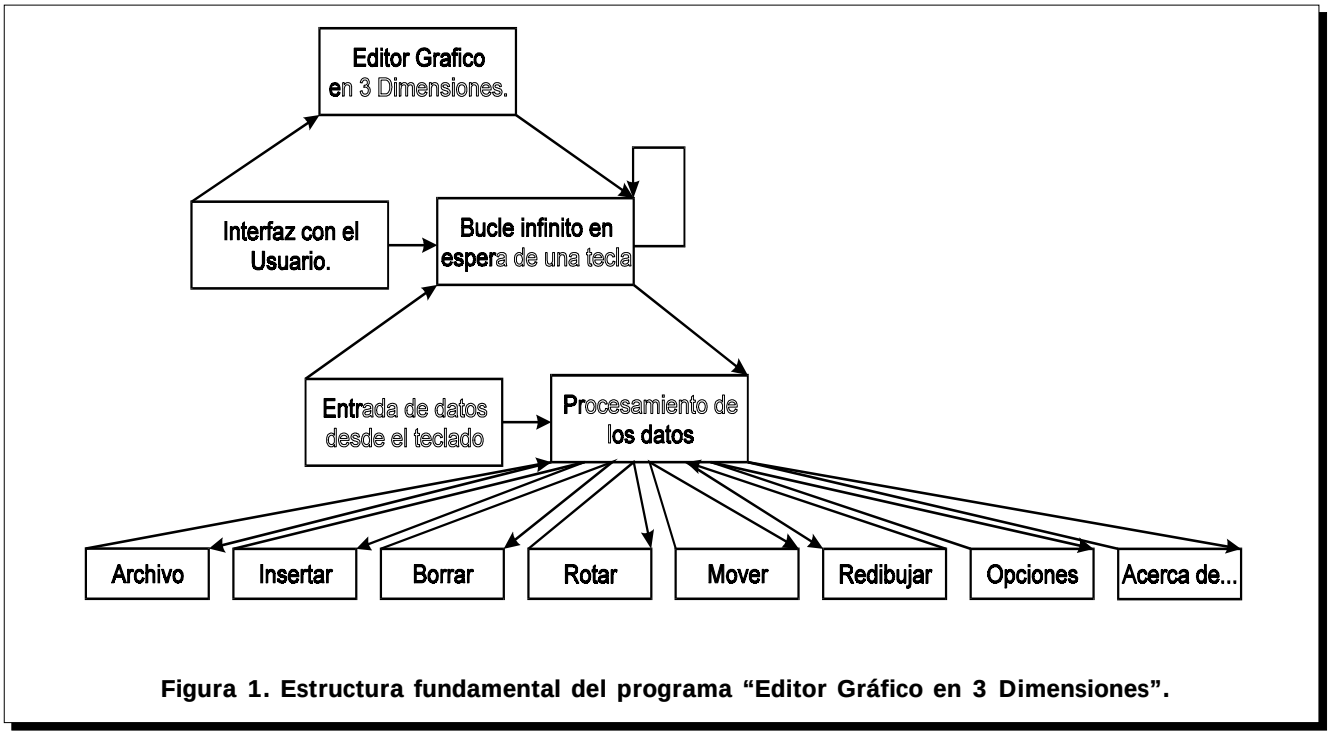

Figura 1. Estructura fundamental del programa "Editor Gráfico en 3 Dimensiones".

Inicialización del modo gráfico.
Limpieza de Pantalla.
Presentación de la interface gráfica.
Trabajo en ciclo infinito en la espera de acontecimientos (cualquier tecla o juego de teclas presionadas).

Una vez sucedido un evento, se efectúa su procesamiento ejecutando la función correspondiente.

De acuerdo a la opción escogida dentro de los menús de la interface gráfica, se lleva a cabo el proceso apropiado para cada una de las funciones a desarrollar.

Método Gráfico

A continuación se presenta el método gráfico empleado para llegar a la especialización de las tareas. En primer lugar se inicializa el ambiente gráfico, a continuación el proceso de ciclo infinito se desarrolla en la espera de mensajes. Esto se muestra en la **figura 1**.

Descripción Particular de los Elementos Fundamentales del Programa

Como siguiente paso se describen las partes principales del programa, siendo consideradas de acuerdo a la solución mostrada anteriormente en las siguientes clases:

Para el manejo de los objetos en tres dimensiones y su conversión a la pantalla gráfica, existen dos funciones muy importantes que son: *conv_coor* y *rotador*. El código de estas y otras funciones se muestra en los recuadros siguientes.

❶La función *conv_coor*, hace la conversión de las coordenadas en tres dimensiones de los objetos a dos dimensiones, que son las correspondientes a las de la pantalla.

❷La función «rotador», efectúa el cálculo de coordenadas de la nueva posición, utilizando para ello las antiguas coordenadas y los ángulos de rotación definidos para cada uno de los ejes X, Y & Z.

❸La lectura de cualquiera de los objetos solo consiste en leer cada uno de los elementos del objeto; una vez que se ha leído el objeto, este se genera y se dibuja. Véase el código de la generación y dibujo de un cubo.

❹Una rutina interesante es la que dibuja el cilindro, debido a que se deben generar las circunferencias a cualquier rotación, en alta y baja resolución.

❺Otra rutina digna de atención es la opción ARCHIVO; esta función permite leer y escribir archivos, así como imprimir e inclusive terminar el programa.

```
//    Convertidor de Coordenadas de 3 dimensiones a coordenadas
//    de pantalla.
void conv_coor(float X1,float Y1,float Z1,int *x,int *y)
{
     int aux;
     aux=xc+escala*(X1-(0.4*Z1));          // Obtiene coordenada X
     *x=aux;
     aux=yc+escala*(-Y1+(0.4*Z1));         // Obtiene coordenada Y
     *y=aux;
}
```

Función *conv_coor* ❶

```
// Genera los vértices del cubo.
void cubo::gen_cubo(void)
{
  float x,y,z;              // Variables intermedias              // Vértice

 conv_coor(xor,yor,zor,&vertice[0][0],&vertice[1][0]);       // 1ero
   rotador(xlong,0,0,rx,ry,rz,&x,&y,&z);
 conv_coor(xor+x,yor+y,zor+z,&vertice[0][1],&vertice[1][1]);   // 2o
   rotador(xlong,ylong,0,rx,ry,rz,&x,&y,&z);
 conv_coor(xor+x,yor+y,zor+z,&vertice[0][2],&vertice[1][2]);   // 3er
   rotador(0,ylong,0,rx,ry,rz,&x,&y,&z);
 conv_coor(xor+x,yor+y,zor+z,&vertice[0][3],&vertice[1][3]);   // 4o
   rotador(0,0,zlong,rx,ry,rz,&x,&y,&z);
 conv_coor(xor+x,yor+y,zor+z,&vertice[0][4],&vertice[1][4]);   // 5o
   rotador(xlong,0,zlong,rx,ry,rz,&x,&y,&z);
 conv_coor(xor+x,yor+y,zor+z,&vertice[0][5],&vertice[1][5]);   // 6o
   rotador(xlong,ylong,zlong,rx,ry,rz,&x,&y,&z);
 conv_coor(xor+x,yor+y,zor+z,&vertice[0][6],&vertice[1][6]);   // 7o
   rotador(0,ylong,zlong,rx,ry,rz,&x,&y,&z);
 conv_coor(xor+x,yor+y,zor+z,&vertice[0][7],&vertice[1][7]);   // 8o
}

// Dibuja el cubo en pantalla.
void cubo::dib_cubo(void)
{
   // Se dibujan las líneas entre los vértices para generar el cubo.

   line(vertice[0][0],vertice[1][0],vertice[0][1],vertice[1][1]);
   line(vertice[0][0],vertice[1][0],vertice[0][3],vertice[1][3]);
   line(vertice[0][0],vertice[1][0],vertice[0][4],vertice[1][4]);
   line(vertice[0][2],vertice[1][2],vertice[0][1],vertice[1][1]);
   line(vertice[0][2],vertice[1][2],vertice[0][3],vertice[1][3]);
   line(vertice[0][2],vertice[1][2],vertice[0][6],vertice[1][6]);
   line(vertice[0][7],vertice[1][7],vertice[0][3],vertice[1][3]);
   line(vertice[0][7],vertice[1][7],vertice[0][4],vertice[1][4]);
   line(vertice[0][7],vertice[1][7],vertice[0][6],vertice[1][6]);
   line(vertice[0][5],vertice[1][5],vertice[0][1],vertice[1][1]);
   line(vertice[0][5],vertice[1][5],vertice[0][4],vertice[1][4]);
   line(vertice[0][5],vertice[1][5],vertice[0][6],vertice[1][6]);
}
```

Generación y Dibujo de un Cubo ❸

```
// Obtiene las nuevas coordenadas de un punto rotado.
void rotador(float xe,float ye,float ze,float ax,float ay,float az,float *xs,float
*ys,float *zs)


{
  float d,xm,ym,zm,alfa;
  int kerror;

  if(xe<0)kerror=-1;
  else kerror=1;
  d=sqrt(xe*xe+ye*ye);          // Calcula la magnitud y el angulo
  if(xe!=0)alfa=atan(ye/xe);   // de la proyeccion en el plano XY
  else
  {
    if(ye<0)alfa=-PI/2;
    else alfa=PI/2;             // Obtiene el signo p/angulo de 90
                                // grados
  }

  if(kerror==-1)alfa=PI+alfa;
  alfa=alfa+az;
  xm=d*cos(alfa);               // Obtiene las nuevas
  ym=d*sin(alfa);               // coordenadas del punto

  if(ze<0)kerror=-1;
  else kerror=1;
  d=sqrt(ze*ze+xm*xm);          // Calcula la magnitud y el angulo
  if(ze!=0)alfa=atan(xm/ze);   // de la proyeccion en el plano ZX
  else
  {
    if(xm<0)alfa=-PI/2;         // Obtiene el signo p/angulo de 90 grados
    else alfa=PI/2;
  }
  if(kerror==-1)alfa=PI+alfa;
  alfa=alfa+ay;
  zm=d*cos(alfa);               // Obtiene las nuevas coordenadas
  xm=d*sin(alfa);               // del punto
  if(ym<0)kerror=-1;
  else kerror=1;
  d=sqrt(ym*ym+zm*zm);          // Calcula la magnitud y el ángulo
  if(ym!=0)alfa=atan(zm/ym);   // de la proyeccion en el plano YZ
  else
  {
    if(zm<0)alfa=-PI/2;        // Obtiene el signo p/angulo de 90 grados
    else alfa=PI/2;
  }
  if(kerror==-1)alfa=PI+alfa;
  alfa=alfa+ax;

  ym=d*cos(alfa);               // Obtiene las nuevas coordenadas
  zm=d*sin(alfa);               // del punto
  *xs=xm;                       // Regresa las nuevas coordenadas
  *ys=ym;
  *zs=zm;
}
```

Función «rotador» ❷

```
// Dibuja el cilindro en la pantalla
void cilindro::dib_cil(void)
{
   float i,xa=0,ya=0,za=0,cont=0;
   int x,y,xx,yy;
   float d,alfa,xxa,yya,zza;

   if(resol==1)          // Pregunta por la resolución
   {                              // Si es alta resolución:
      for(i=0;i<2*PI;i=i+.01)  // Dibuja punto por punto los círculos
      {
         xa=Kx*radio*sin(i+(Kx*Ky*PI/2));
         ya=Ky*radio*sin(i+(Ky*Kz*PI/2));
         za=Kz*radio*sin(i+(Kz*Kx*PI/2));
         rotador(xa,ya,za,rx,ry,rz,&xxa,&yya,&zza);
         xxa=basex+xxa;
         yya=basey+yya;
         zza=basez+zza;
         conv_coor(xxa,yya,zza,&x,&y);
         putpixel(x,y,WHITE);
         if(Kx==0)xa=clong;
         if(Ky==0)ya=clong;
         if(Kz==0)za=clong;
         rotador(xa,ya,za,rx,ry,rz,&xa,&ya,&za);
         xa=basex+xa;
         ya=basey+ya;
         za=basez+za;
         conv_coor(xa,ya,za,&xx,&yy);
         putpixel(xx,yy,WHITE);
         if(cont>=PI/(2*sqrt(radio)))
         {
            line(x,y,xx,yy);
            cont=0;
         }
         cont=cont+.01;
      }
   }
   else           // Si es baja resolucion:
   {                              // Dibuja poligonos de 20 lados
      for(i=0;i<19;i++)
      {
         line(verticeb[0][i],verticeb[1][i],verticeb[0][i+1],verticeb[1][i+1]);
         line(verticet[0][i],verticet[1][i],verticet[0][i+1],verticet[1][i+1]);
         line(verticeb[0][i],verticeb[1][i],verticet[0][i],verticet[1][i]);
      }
      line(verticeb[0][0],verticeb[1][0],verticeb[0][19],verticeb[1][19]);
      line(verticet[0][0],verticet[1][0],verticet[0][19],verticet[1][19]);
   }
}
```

Generación y dibujo de un cilindro ❹

```
// Opcion Archivo
void ARCHIVO(void)
{
   char op,inst='n',arc;
   int i,j,k,n,l=0,sel=0;
   FILE *arch;                // Apuntador al archivo

   n=0;
   for(i=0;i<2;i++)           // Pone los elementos de menú
   {
      for(j=0;j<3;j++)
      {
         outtextxy(200+(i*100),80+(j*20),&Archivo[n][0]);
         n++;
      }
   }
   while(sel==0)
   {
      op=getch();
      if(op=='\r')op=inst;
      switch(op)              // Selecciona
      {
         case 'n':            // Nuevo
         case 'N':
            res_vent=0;
            putimage(170,50,mem,COPY_PUT);
            n=n_cubos;                    // Borra todos los elementos
            for(i=0;i<n;i++)borra_elem('c',0);
            n=n_cil;
            for(i=0;i<n;i++)borra_elem('i',0);
            n=n_pir;
            for(i=0;i<n;i++)borra_elem('p',0);
            REDIBUJAR();
            sel=1;
            break;
         case 'a':            // Abrir
         case 'A':
            line(170,140,470,140);
            outtextxy(200,150,»Nombre del Archivo: «);
            for(k=0;k<20;k++)archivo[k]=NULL;
            for(k=0;k<20;k++)          // Lee el nombre del nuevo archivo
            {
               archivo[k]=getch();
               if(archivo[k]=='\r')
               {
                  archivo[k]=NULL;
                  k=20;
               }
               gotoxy(30,12);
               cout<<archivo;
            }
            res_vent=0;
            putimage(170,50,mem,COPY_PUT);
            n=n_cubos;                    // Borra todos los elementos
            for(i=0;i<n;i++)borra_elem('c',0);
            n=n_cil;
            for(i=0;i<n;i++)borra_elem('i',0);
            n=n_pir;
            for(i=0;i<n;i++)borra_elem('p',0);
            arch=fopen(&archivo[0],»rb»);          // Abre el archivo
            fread(&n_cubos,sizeof(int),1,arch);    // Lee el archivo
            for(i=0;i<n_cubos;i++)                 // Cubos
            {
               ap_cubo[i]=new cubo;
               if(ap_cubo[i]==NULL)LM();
```

❺...

```
            fread(ap_cubo[i],sizeof(cubo),1,arch);
          }
          fread(&n_cil,sizeof(int),1,arch);
          for(i=0;i<n_cil;i++) // Cilindros
          {
            ap_cil[i]=new cilindro;
            if(ap_cil[i]==NULL)LM();
            fread(ap_cil[i],sizeof(cilindro),1,arch);
          }
          fread(&n_pir,sizeof(int),1,arch);
          for(i=0;i<n_pir;i++)      // Piramides
          {
            ap_pir[i]=new piramide;
            if(ap_pir[i]==NULL)LM();
            fread(ap_pir[i],sizeof(piramide),1,arch);
          }
          fclose(arch);             // Cierra el archivo
          REDIBUJAR();
          sel=1;
          break;
        case 'c':                   // Salvar como...
        case 'C':
          line(170,140,470,140);
          outtextxy(200,150,»Nombre del Archivo: «);
          for(k=0;k<20;k++)archivo[k]=NULL;
          for(k=0;k<20;k++)    // Lee el nombre del archivo
          {
            archivo[k]=getch();
            if(archivo[k]=='\r')
            {
              archivo[k]=NULL;
              k=20;
            }
            gotoxy(30,12);
            cout<<archivo;
          }                         // No termina
        case 's':                   // Salvar
        case 'S':
          arch=fopen(&archivo[0],»wb+»);
          fwrite(&n_cubos,sizeof(int),1,arch);      // Salva cubos

for(i=0;i<n_cubos;i++)fwrite(ap_cubo[i],sizeof(cubo),1,arch);
          fwrite(&n_cil,sizeof(int),1,arch);        // Salva cilindros
          for(i=0;i<n_cil;i++)fwrite(ap_cil[i],sizeof(cilindro),1,arch);
          fwrite(&n_pir,sizeof(int),1,arch);        // Salva piramides
          for(i=0;i<n_pir;i++)fwrite(ap_pir[i],sizeof(piramide),1,arch);
          fclose(arch);
          sel=1;
                  break;
        case 'i':
        case 'I':
          RED_FIG();
          int86(0x5,NULL,NULL);     // Imprime
          sel=1;
          REDIBUJAR();
          break;
        case 'l':                   // Salir

        case 'L':
          exit(0);
          break;

        case '\x0':                 // Flechas
          l=flechas(l,6);
          switch(l)
```

... ❺ ...

```
          {
            case 0:                 // Nuevo
              inst='n';
              break;
            case 1:    // Abrir
              inst='a';
              break;
            case 2:    // Salvar
              inst='s';
              break;
            case 3:    // Salvar como...
              inst='c';
              break;
            case 4:    // Imprimir
              inst='i';
              break;
            case 5:    // Salir
              inst='l';
              break;
          }
          break;
      }
    }
}
```

Función Archivo ... ❺

En la **figura 2** se muestra el ambiente de trabajo, en el cual ya se han insertado tres figuras: un cubo, un cilindro y una pirámide de cinco lados.

Como ya se ha mencionado, el programa permite Mover, Rotar, Borrar e Insertar objetos; partiendo de la pantalla anterior se girará el cilindro 90 grados con respecto al eje X, desplazando además al cubo de su posición original, se borrará la pirámide y se insertará una nueva más alta y con ocho lados. La pantalla queda como se muestra en la **figura 3**.

Por su parte, se muestra en la **figura 4** una caja de dialogo referente al menú ARCHIVO.

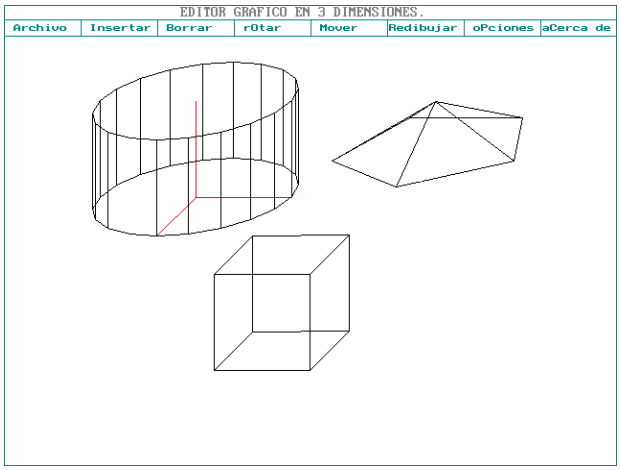

Figura 2

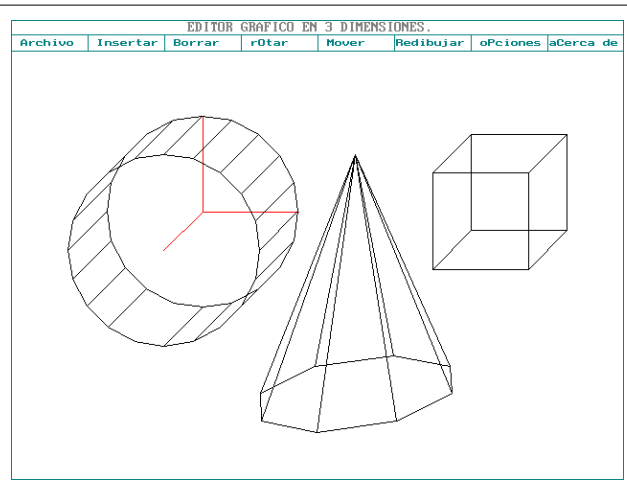

Figura 3

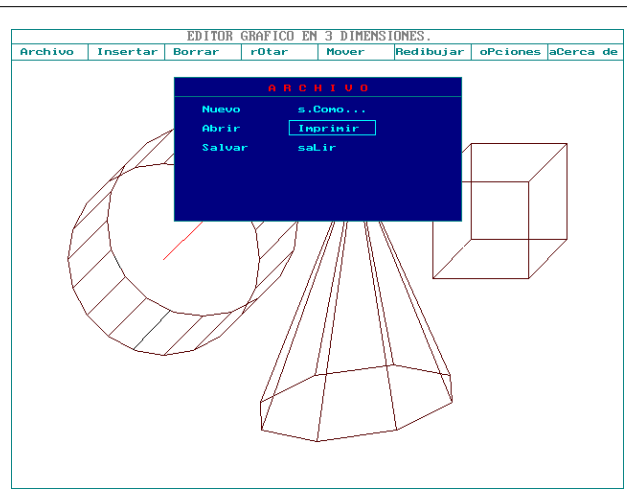

Figura 4

Conclusiones

El crear programas de ambiente gráfico, para el manejo elemental de figuras en tres dimensiones con una programación orientada a objetos, no solo facilita la realización de esta tarea, sino que además permite la generación y expansión de nuevas figuras tridimensionales a partir de las rutinas previamente diseñadas. Con esto se logra una reutilización del código y una reducción significativa en el consumo de tiempo de desarrollo de software, lo cual lo hace muy atractivo si tomamos en cuenta las necesidades de tener herramientas que nos permitan llevar a cabo nuestras tareas de una forma rápida y concisa.

Elaborar técnicas de desarrollo de software propias, para adecuarlas a nuestras necesidades de cómputo, nos hace ser menos dependientes de los cambios que sufren constantemente los actuales terrenos del software.

Bibliografía

[1] Herbert Schildt. *"Turbo C/C++: Manual de referencia"*. McGraw-Hill

[2] *"Funciones del DOS y BIOS"*. Addison-Wesley Iberoamericana.