Escuela Superior Politécnica del Litoral

Facultad de Ingenieria en Electricidad

"GrafWin"

Graficador de Funciones de dos variables para Windows

Manual de Diseño

Proyecto de Tópico de Grado
Previa a la obtención del titulo de
Ingeniero en Computación

Presentado por:

Barahona Morales, Edison Ricardo
García Heras, Segundo Enrique
Mestanza Yépez, Carlos

Guayaquil - Ecuador
1994

Agradecimiento:

Queremos dar un especial agradecimiento al Ing. Sixto García A., Director del Tópico de Graduación por que gracias a este proyecto pudimos comprender que la preparación académica de nuestra área no termina, y más aún que ahora nos damos cuenta de que la carrera que hemos escogido es tan amplia y tan interesante que no se contenta consigo misma, sino que busca el conocimiento más allá de sus fronteras.

En este sistema, no sólo interactuamos con el área matemática, sino con áreas mucho más abstractas e interesantes como el de la lógica. Comprendimos que el Ingeniero de Sistemas es una esponja que aprende todo acerca del área con la que interactúa y se convierte en un experto en ella, pués se necesita más que el breve conocimiento de ésta para poder crear un sistema a la altura de las necesidades.

Estamos conscientes que aún queda mucho por aprender, pero confiamos en que lo aprendido sea como las bases firmes que ayudarán sirviendo de cimientos para construir un futuro.

Gracias también a nuestras familias que nos apoyaron en todo momento, que aunque no comprendían nada de lo que hablábamos, simulaban que sí para mantenérnos contentos y con ánimos para seguir.

Y principalmente, gracias a Dios que siempre fué la inspiración que necesitábamos.

Dedicatoria :

A nuestros padres que siempre nos apoyaron y sin escatimar sacrificio alguno nos brindaron su ayuda.

Declaración expresa

"La responsabilidad por los hechos, ideas y doctrinas expuestas en este proyecto, nos corresponde exclusivamente; y, el patrimonio intelectual de la misma a la Escuela Superior Politécnica del Litoral".

(Reglamento de Exámenes y Títulos Profesionales)

García Heras Segundo

Barahona Morales Edison

Mestanza Yépez Carlos

Armando Altamirano
Subdecano de la Facultad de
Ingeniería en Electricidad

Ing. Sixto García
Director del Tópico de Graduación

Ing. Guido Caicedo
Miembro del Tribunal

Ing. Mónica Villavicencio
Miembro del Tribunal

Introducción

Su diseño fue hecho Orientado a Objetos, pero cuidando de que sea lo más independiente posible para que sea útil para otras aplicaciones futuras. La plataforma que se escogió para su implementación fué WINDOWS, y se utilizó los recursos que ésta brinda. Se aprovechó de la herramienta utilizada como lenguaje de programación : Borland C++, pués se reusaron clases ya escritas además de las suyas propias.

WINDOWS es un ambiente que permite a GrafWin ser portable e independiente del hardware usado. Pués corre lo mismo en un monitor color como monocromático, o con monitores de diferentes resoluciones, o con distintas impresoras.

Para su implementación, se crearon clases para Validación y Evaluación de la función ingresada, para la Graficación y para la Impresión. El objetivo principal, fué hacer éstas clases lo más independientes posibles, para que sirvan como base a otras aplicaciones mucho más complejas. Otro objetivo fué que se usara en lo más posible los recursos disponibles, lo cual se logró usando las clases suministradas por Borland C++.

Diseño de GrafWin

GrafWin fué diseñado orientado a objetos. Este diseño se guió en los sgtes. requerimientos :

- Debe ser independiente.
- Debe ser portable.
- Debe utilizar las clases predefinidas en la herramienta de programación.

A continuación se presenta el diseño de GrafWin y después se explicará cada una de las partes, además de dar una justificación de porqué se hizo de esa manera.

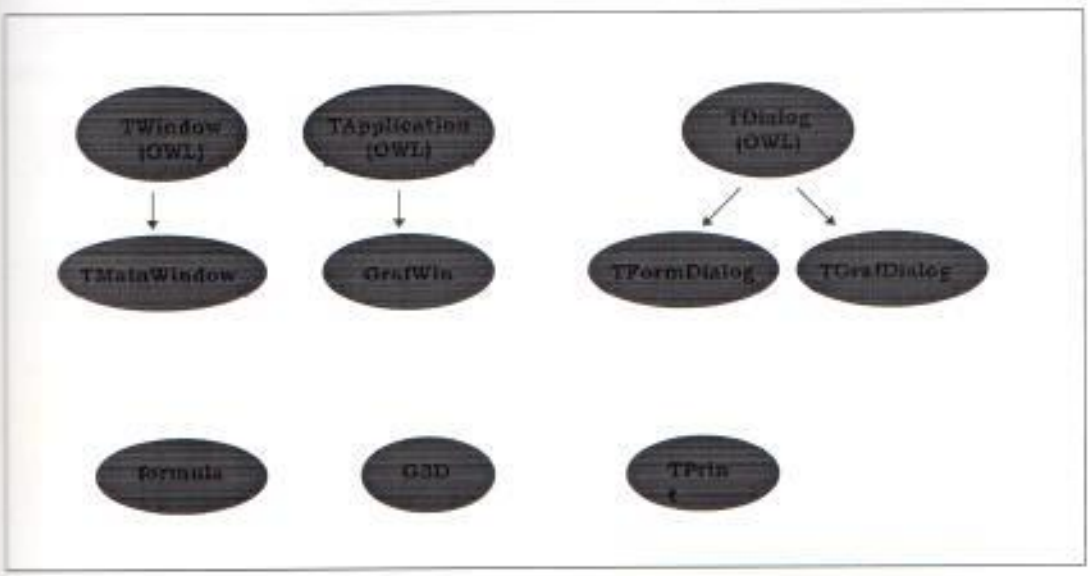

Figura 1. Diseño de GrafWin

Se utilizaron tres clases de predefinidas en la libreria de objetos de Borlandc C++ OWL (Object Windows Lybraries). Estas clase sirvieron de base para derivar las clases que usa GrafWin para interactuar con el ambiente de WINDOWS.

- TWindow es una clase que es usada para el manejo de una ventana de Windows. Como GrafWin sólo usa una ventana, sólo necesitamos crear una clase para esa ventana.
- TApplication sirve para el manejo de la aplicación, mensajes que llegan, interacción con las otras aplicaciones, etc.
- TDialog es una clase que permite la entrada de datos, por medio de una caja de diálogo. GrafWin usa dos de estas clases. Una para la entrada de la fórmula, la otra para la entrada de los parámetros de graficación.

Las otras tres clases independientes son para el manejo interno de la aplicación.

Fórmula es una clase que se encarga de la validación y evaluación de una función. Es independiente tanto así que se puede usar en cualquier otro problema parecido que tenga que validar y evaluar una función de dos variables de la forma z=f(x,y). Si alguien quiere modificar para que acepte funciones de la otra forma, podría hacerlo sin mucho esfuerzo. Sólo tendría que cuidar la misma interfase.

G3D es una clase que se encarga de la graficación de la fórmula. También es independiente y si alguien desea usarla deberá sobrescribir su función Evalúa para que le envíe los valores a graficar. Puede graficar en cualquier HDC, pués es accesible al usuario de la clase, para que él lo setee al HDC que estime conveniente.

Tprint es una clase que se encarga de la impresión. Es independiente y protable. Lee el driver que esta seteado en el archivo de inicialización de Windows WIN.INI y lo carga y usa sus facilidades para imprimir. Consta de funciones de Inicio y fin de trabajo de impresión, inicio y fin de página, etc.

Módulo de Validación y Evaluación de la Función ingresada

Para la validación de la función se usa un juego de reglas de producción que aceptan determinadas operaciones y funciones en notación INFIX. Se ha decidido usar ésta, por que es la notación más usada en nuestro medio. La naturaleza de esta validación es recursiva. La figura 1 muestra las reglas de validación que se usaron.

< *expresion* > = < *termino* > | < *termino* > <*operador aditivo* > <*expresion*>
< *termino* > = < *factor* > | < *factor* > < *operador multiplicativo* > < *termino* >
< *factor* > = (< *expresion* >) | < *constante* > | < *variable* > | < *funcion* >
< *constante* > = *"cualquier número real"*
< *variable* > = " *x* " | " *y* "
< *operador multiplicativo* > = " * " , " / " , " ^ "
< *operador aditivo* > = " + " , " - "
< *funcion* > = <*nombre de funcion*> (< *parametro* >)
< *nombre de funcion* > = *"sen", "cos", "tan", "asen","acos", "atan", "log" , "ln", "exp"*
< *parametro* > = < *expresion* >

Figura 2. Reglas de producción

Para que una expresión sea válida, debe cumplir con los siguientes requisitos :

- Debe estar escrita en notación Infix.
- Las únicas variables aceptadas son X y Y, en cualquier caso.
- Las funciones deben ser : sen, cos, tan, asen, acos, atan, exp, ln, log
- Las constantes pueden ser cualquier numero real. Este número real también puede estar escrito en notación decimal.
- Las operaciones matemáticas aceptadas son : Adición (+), Substracción (-), Multiplicación (*), División (/) y Potenciación (^).

Una vez ya validada la función, se procede a la evaluación. Para ésto se construye un árbol binario. Las hojas del árbol pueden ser constantes o variables, miéntras que las ramas operadores o funciones. La figura 2 muestra cómo quedaría el árbol de una expresión válida :

Sería bueno, en este momento, dar a conocer la estructura de un nodo del árbol, para que quede claro como el árbol puede guardar diferentes tipos de datos en el mismo nodo :

```
struct NODO
{
   OPERATOR op;      // Guarda el Operador,funcion, constante o variable de
                     // acuerdo al caso
   OP_TIPO tipo;     // Guarda el tipo : Operador,funcion,constante o variable
   NODO *hizq,*hder; // Punteros a los hijos izquierdos y derechos. Para una hoja
                     // ambos serian nulos
};
```

Figura 3. Estructura del nodo del árbol de evaluación

La estructura de un Nodo consiste en el dato en si (op), el tipo de dato (tipo), y los punteros a los hijos (hizq,hder). Como el dato puede ser un operador, una constante, una variable o una función, la variable op debe ser también una estructura (OPERATOR), que dependiendo del tipo, guarde el dato en el campo correcto. Como para cada nodo sólo se guarda un tipo de dato especifico, los otros campos de la estructura OPERATOR serían un gasto. Es por eso que OPERATOR ha sido definida como una unión :

```
union OPERATOR
 {
   OPE_MAT operador;                 // Guarda un operador : + , - , * , / , ^

   char funcion[MAXFORMULA+1];       // Guarda una función : sen, cos, etc..

   double constante;                 // Guarda una constante : 112, 24.12,
                                                          2.12e-10, etc...
   VAR variable;                     // Guarda una variable : X ó Y
   };
```

Figura 4. Estructura que guarda el operador

En esta unión, se puede guardar cualquiera de los 4 diferentes tipos de datos. En la variable operador se guarda el signo del operador matemático '+', '-', '*', '/', '^'. OPE_MAT es un tipo de dato enumerado que contiene los codigos ASCII de estos signos; asociados a palabras claves que son mas entendibles en la programación :

```
enum OPE_MAT { OP_ADICION=43, OP_SUBSTRACCION=45, OP_MULTIPLICACION=42,
OP_DIVISION=47, OP_POTENCIACION=94};
```

La variable funcion contiene el nombre de la funcion, tal como fué escrita por el usuario. así como constante, guarda el valor de la constante y variable el de la variable. El campo variable es un campo enumerado que puede tomar como valor cualquiera de las variables aceptadas como válidas en una función.

enum VAR {VARX,VARY};

Para que se pueda accesar al campo correcto, el nodo cuenta con una variable que guarda el tipo del dato. Según este tipo se sabe que es en realidad lo que guarda el nodo. El tipo de dato de este campo (OP_TIPO) se detalla a continuación :

Implementación de fórmula usando Orientación a Objetos OOP

Para implementar la evaluación y validación de la función, se utilizó la clase **fórmula**. Tiene construida dentro de ella, un ***"Árbol de Evaluación"*** que servirá para almacenar la expresión válida. En cada nodo del árbol, como se menciona arriba, se almacena un operador, una constante, una variable o una función dependiendo del caso.

```
class formula
{
  private:
     BOOL Ok;
     ERROR_TIPO error;
     NODO *raiz;
    char expresion[MAXFORMULA+1];
    int OpAditivo(int,int);   // Funciones para Validacion de la expresion
     int OpMultiplicativo(int,int);
     BOOL IsExpresion(NODO **,int,int);
     BOOL IsTermino(NODO **,int,int);
     BOOL IsFactor(NODO **,int,int);
     BOOL IsFuncion(NODO **,int,int);
     BOOL IsParam(NODO **,FUNCIONES,int,int);
     BOOL IsConstante(NODO **,int,int);
     BOOL IsVariable(NODO **,int,int);
     FUNCIONES IsNombreFuncion(char *);
     NODO *CrearNodo(void);          // Funciones que trabajan sobre el árbol
     void InsNodo(NODO** padre, NODO *nodo);
     void FreeTree(NODO **nodo);
     double Evaluar(NODO *,double,double);   // Funciones que evalúan el arbol

  public:
     formula(void);                  // Constructor default
     formula(char *string);          // Constructor desde una funcion almacenada
                                     // en un string
     ~formula();                     // Destructor
     BOOL GetStatus(void);
     ERROR_TIPO GetError(void);
     void Create(char *);
     double EvaluaZ(double,double);
};
```

Figura 5. Clase que implementa la evaluación y validación de una función

La clase posee dos constructores, el primero no recibe parámetros y sirve sólo para inicializar los datos miembros. Si se quiere construir el árbol de evaluación de alguna función, se usará la función miembro *Create* que será detallada posteriormente.
El segundo constructor recibe como parámetro la función en un string y si es válida construye el árbol de evaluación.

Se acordó hacerlo así y no cada vez que se evalúe, debido al retardo que la validación y la creación del árbol ameritan. Es por eso que se crea el árbol primero (en el constructor de la clase) y cada vez que se evalúe, se ahorrará tiempo del procesador, pués el árbol ya está creado y sólo se necesitará recorrerlo.

El destructor de la clase sólo libera el espacio ocupado por el árbol.
Existen 3 Tipos de Funciones :

- ***De Validación***, Las funciones que validan si la expresión esta bien escrita.
- ***De Árbol***, funciones que operan sobre el árbol de evaluación.
- ***De Evaluación***, funciones que reciben los valores de X ,Y y evalúan la expresión recorriendo el árbol.

Las funciones de Validación son una simple transcripción de las reglas de producción. Estas funciones són :

```
int OpAditivo(int,int);
int OpMultiplicativo(int,int);
BOOL IsExpresion(NODO **,int,int);
BOOL IsTermino(NODO **,int,int);
BOOL IsFactor(NODO **,int,int);
BOOL IsFuncion(NODO **,int,int);
BOOL IsParam(NODO **,FUNCIONES,int,int);
BOOL IsConstante(NODO **,int,int);
BOOL IsVariable(NODO **,int,int);
FUNCIONES IsNombreFuncion(char *);
```

Las funciones descritas anteriormente tienen incorporado la formación del árbol. Ésto lo hacen pasándose como parámetro el nodo donde insertarán la información. Si encuentran que algo no es válido, llaman a FreeTree para liberar todos los nodos creados y que son erróneos.FreeTree será hablada más en detalle posteriormente.

Las funciones de Árbol, son una implementación sencilla del TDA Árbol Binario. Las funciones de este grupo són :

```
NODO *CrearNodo(void);
void InsNodo(NODO** padre, NODO *nodo);
void FreeTree(NODO **nodo);
```

CrearNodo reserva un espacio de memoria para almacenar un nodo del árbol y retorna su dirección. InsNodo inserta un nodo en la posición que se le envía. Debido a que ésta, cambia el puntero que se le envía, este valor es pasado por referencia (NODO **). FreeTree es una función recursiva que libera todos los hijos del nodo especificado, así como al nodo mismo.
Las funciones de Evaluación, recorren el árbol en forma INFIX para ir evaluando la expresión. Es de naturaleza recursiva y valida los parámetros de evaluación para evitar los errores de excepción en el procesador. Estas funciones son :

```
double Evaluar(NODO *,double,double);
double EvaluaZ(double,double);
```

Existen otras funciones, que no han sido tomadas en cuenta. La razón es que sólo sirven como interfase entre la clase y el mundo exterior. Estas funciones són:

```
BOOL GetStatus(void);
ERROR_TIPO GetError(void);
void Create(char *);
```

GetStatus simplemente retorna la variable Ok que dice si la función es válida o no. GetError retorna, en cambio la variable error que dice que error ha ocurrido en la validación. Create es una función que se usa cuando se quiere validar otra función, pero el objeto ya se ha creado. Ésta limpia el árbol (llamando a FreeTree) y Valida la función que se le envía en un string (llamando al constructor).

Existen 4 datos privados dentro de la clase, los cuales le dan una mayor funcionalidad.

```
BOOL Ok;
ERROR_TIPO error;
NODO *raiz;
char expresion[MAXFORMULA+1];
```

Ok almacena la validez de la función. Si es FALSA, ha ocurrido un error y este valor se encuentra en la variable **error**. **raiz** guarda el puntero a la raiz del árbol, si es NULL el árbol no se ha podido

crear, ya sea por invalidez de la función o por falta de memoria. La razón en este caso, también se encuentra en la variable **error**. **expresion** almacena la función que dió forma al árbol.

Para dar una mejor idea de la validación y evaluación de función, se mostrará como quedará el árbol de evaluación después de haber de la validación.

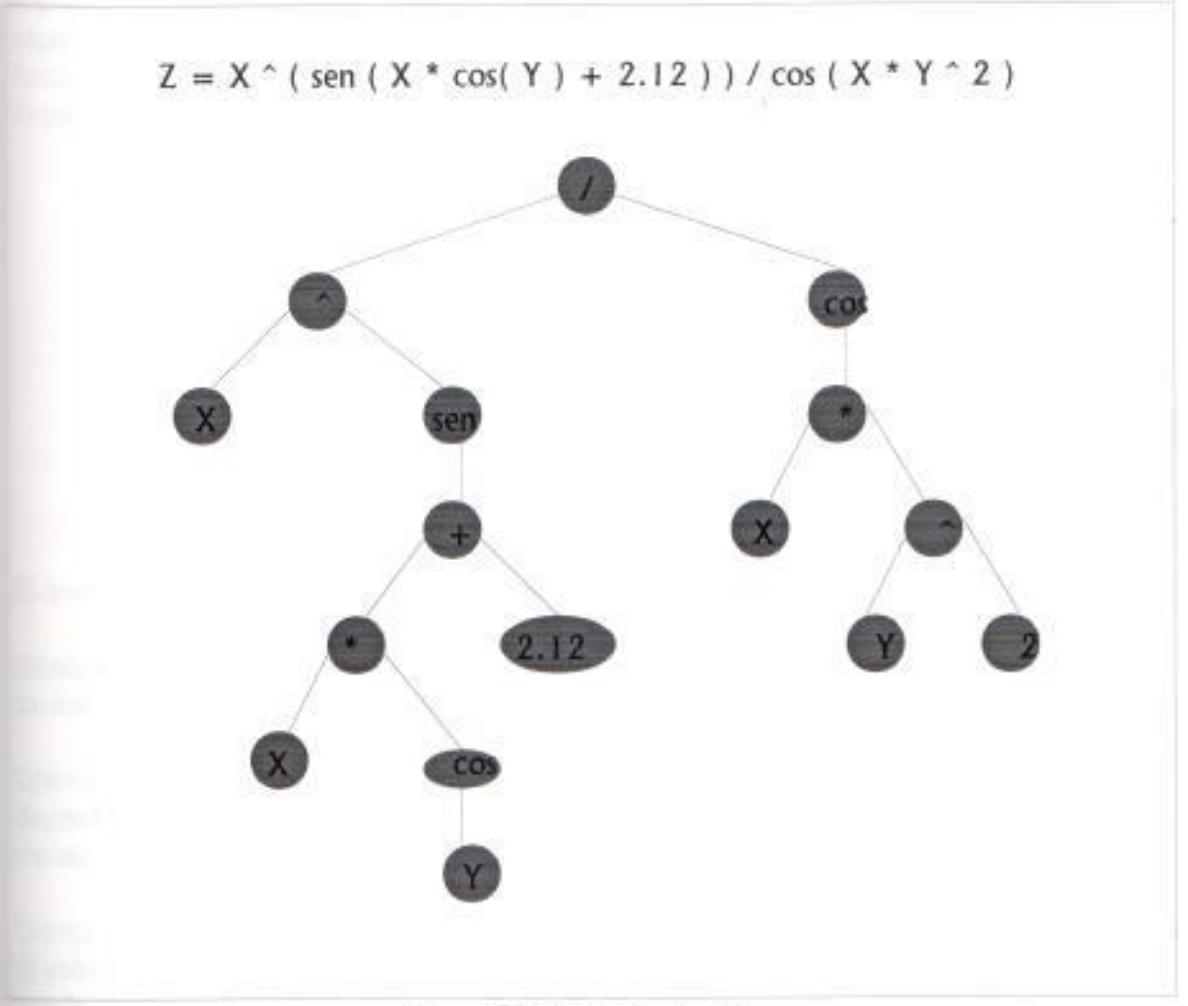

Figura 6. Árbol de Evaluación

Modulo Tprint

Tprint es un módulo que se encarga de obtener el handle de contexto de la impresora. Para aquellas personas que desconozcan el término de dispositivo de contexto, éste es la manera de como windows accesa a los recursos gráficos o d e impresión. Obteniendo el handle de este dispositivo, se puede imprimir en pantalla o e impresora, independiente del hardware instalado.

Para lograr ésto, Tprint lee el archivo de inicialización de Windows para encontrar el driver de impresora instalado, cargarlo y obtener el handle del dispositivo de contexto para la impresora especificada en Windows. Una vez obtenido este handle, se puede utilizar cualquiera de las funciones del GDI (Graphics Device Interfase) de Windows para mostrar texto, lineas, rectángulos, circulos, etc.

Figura 7. Interfase de Tprint con Windows

El driver de la impresora instalada en Windows se encuentra especificado de la siguiente manera :

[Windows]
device=Kyocera F-Series (USA),HPPCL,LPT1:

Obteniendo estas entradas en WIN.INI, cargamos el driver HPPCL y obtenemos el handle del dispositivo de contexto con los otros 2 parámetros. Tprint no sólo cuenta con ésto, sino que da ciertas facilidades usando internamente sentencias de escape que se envian a la impresora.

Cuenta con una cosntructora que encuentra el Handle del dispositivo de contexto de la impresora. Si esteocurrió un error o no se pudo cargar el handle, Tprint setea la variable que contiene este handle (hPr) a NULL. El usuario de la clase, deberá chequear si este valor es válido para poder usarla. Si el usuario de la clase no verifica ésto y usa un handle nulo, puede obtener resultados inesperados.

Cosnta además de una función para el seteo de la impresora PrinterSetup. Esta función carga el driver y obtiene la dirección del puntero a la función de seteo de la impresora default de Windows. Aparece el mismo diálogo que aparece en el la opción Printer Setup del Print Manager.

Posee funciones miembras de inicio y fin de trabajo de impresión, inicio y fin de página, cancelación del trabajo de impresión, etc. Si el usuario quiere mas facilidades, puede derivar la

clase e insertar nuevas funciones asociadas a sentencias de escape. Esto es muy fácil hacerlo. Sólo debe leerse el help de Borlandc C++ o de otro compilador compatible para comprender las diferentes sentencias de escape y entenderlas.

Posee una variable que dice si un error ha ocurrido. Fácilmente se puede preguntar por esa variable y saber si se ha cometido un error. Posee funciones de cálculo del tamaño de la línea para el tipo de letra escogido, útil para hacer una salida bien formateada.

Implementación de Tprint con orientación a objetos

Tprint es implementada en una clase que se detalla a continuación :

```
class TPrint
{
  BOOL Job;
  BOOL Page;
  int Error;
  POINT PageSize;
  char far PrintDriver[MAXPRINTER+1];
  char far PrintPort[MAXPRINTER+1];
  char far PrintType[MAXPRINTER+1];

  void GetPageSize(void);
  void ReleaseDC(void);

public:
  HDC hPr;
  int LineSpace,LinesPerPage;
  float SCALEX,SCALEY,MARGENX,MARGENY;

  TPrint(void);
 ~TPrint(void);
  virtual void AbortDoc(void);
  virtual void StartDoc(void);
  virtual void StartPage(void);
  virtual void EndPage(void);
  virtual void EndDoc(void);
  virtual void PrinterSetup(HWND);
   virtual HDC GetDC(void) { return hPr;};
};
```

Figura 8. Clase Tprint

Tprint ofrece al handle de la impresora como público, para que todos los usuarios de la clase lo puedan usar. Debido a ésto es que el usuario debe tener mucho cuidado en no perder este valor, pués el único que sirve para la labor. Es aconsejable usarlo como de lectura solamente.

Módulo G3D (Gráfico de la función).

General

Los puntos dentro de este algoritmo serán calculados considerando un sistema de ejes cartesianos, esto quiere decir que para definir un punto serán necesarios tres valores, valor para x, valor para y, y valor para z. (x, y, z).

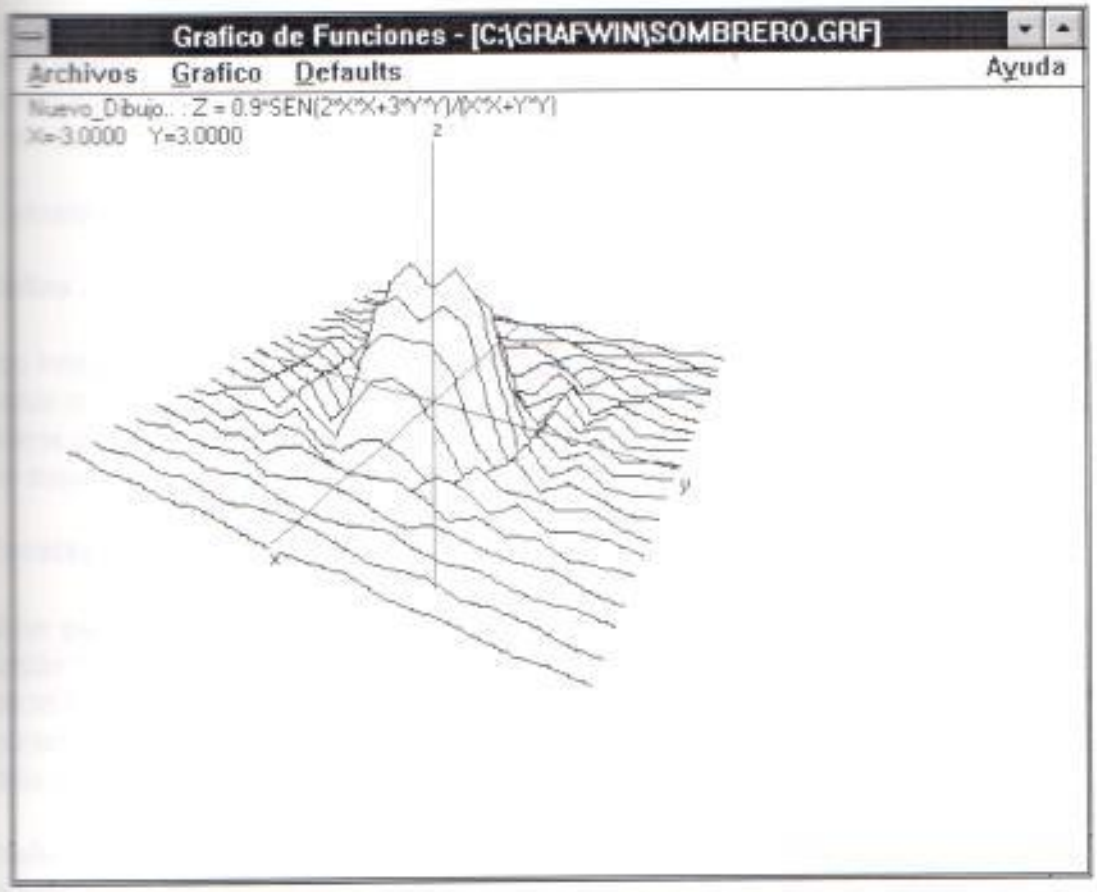

Figura 9.Gráfico obtenido con la clase G3D

Parámetros considerados.

Punto de vista :

El punto de vista o lugar desde donde el usuario desea ver la función, estará dado por un sistema de cooredenadas polares. Para esto es necesario especificar tres valores, ángulo entre el eje X y la proyección del vactor posición sobre el plano XY (theta), ángulo entre el eje Z y el vertor

posición (phi), la magnitud de el vector posición o distancia entre el origen (0,0,0) y el punto de vista (rho).

Se eligió el sistema de coordenadas polares porque nos pareció el más natural para esta función.

Para la interfase con el usuario, theta se leerá como ángulo X, phi como ángulo Z, y rho como distancia.

Dominio :

Los rangos para los valores de X y Y, que serán evaluados, estan especificacos por el usuario a través de los parametros de rango en X, Y y Z. Los valores de xdesde, xhasta, ydesde y yhasta definen el dominio sobre el cual la función será evaluada, los valores de zdesde y zhasta definen el tamaño del eje Z que se graficará.

El usuario verá estos rangos como "RANGO X", "RANGO Y", y "RANGO Z".

Deltas :

Esto indica la calidad o grado de resolución del gráfico, con un delta más pequeño el algoritmo evalua más puntos, los efecto de esto es : se demora más en dibujar la misma función, y se obtiene un gráfico más nítido sobre todo en funciones que presentan cambios muy bruscos (no son suaves).

Escalas :

Sirven para deformar el gráfico a voluntad sin necesidad de cambiar la formula. Por ejemplo la función 'SEN(X+Y)' para X de -5 a 5 y Y de -5 a 5 se ve en la pantalla como algo muy pequeño, que no se aprecia, pero si le ponemos la escala en X igual a 2 y la escala en Y igual a 2, el dibujo aparece tan grande como si elijieramos un dominio de -10 a 10 en X y en Y, y se ve en la pantalla como si la formula fuera 'SEN(2*X+2*Y)'.

Puede ser ademas que las ondulaciones no sean muy notadas o se quiere apreciar mejor las ondulaciones, esto se consigue agregandole un factor a la formula como por ejemplo : 5*SEN(X+Y), este mismo esfecto se consigue con la escala en Z igual a 5, de esta forma el efecto es el mismo y no se requiere cambiar la formula.

Desarrollo matemático :

El punto de vista se representa por las coordenadas : (rho, theta, phi).

Este mismo punto se puede representar en coordenadas cartesianas (x, y, z). Y la relación entre estos puntos es :

x = rho * SEN(theta) * COS(phi)
y = rho * SEN(theta) * SEN(phi)
z = rho * COS(phi)

Además se comprueba que : $rho^2 = x^2 + y^2 + z^2$.

Transformaciones.

Las transformaciones que podemos realizar sobre un espacio tridimensional son : ESCALAMIENTO, ROTACION, TRASLACION y REFLECCION.

Para poder trabajar mas comodamente pasamos los puntos (x, y, z) que corresponden a un espacio vectorial de 3 dimensiones a un espacio vetorial homogeneo de 4 dimensiones, esto se consigue agregando un 1 como cuarto elemento, el punto (x, y, z) será el punto (x, y, z, 1).

Escalamiento.

Esto nos permite hacer más grande o más pequeño la función sin cambiar la formula en sí. La matriz representativa de esta operación es :

$$(x, y, z, 1) \begin{pmatrix} A & 0 & 0 & 0 \\ 0 & B & 0 & 0 \\ 0 & 0 & C & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (Ax, By, Cz, 1)$$

Rotación.

La rotación se realizará en el sentido de las manecillas del reloj. Se requiere de una matriz diferente para la rotación sobre cada uno de los ejes.

La rotación sobre el eje Z tiene la siguiente matriz caracteristica.

$$\begin{pmatrix} \cos f & \text{sen } f & 0 & 0 \\ -\text{sen } f & \cos f & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotación sobre el eje X :

1	0	0	0
0	cos f	sen f	0
0	-sen f	cos f	0
0	0	0	1

Rotación sobre el eje Y :

cos f	0	-sen f	0
0	1	0	0
sen f	0	cos f	0
0	0	0	1

Traslación.

Es el desplazamiento de un punto sobre el espacio :

	1	0	0	0	
(x, y , z, 1)	0	1	0	0	= (x+H, y+K, z+L, 1)
	0	0	1	0	
	H	K	L	1	

Reflección

Es la obtención de una imagen invertida o con efecto de 'espejo', para esto necesitamos tres matrices, una para cada plano.

Para el plano XY.

1	0	0	0
0	1	0	0
0	0	-1	0
0	0	0	1

Para el plano XZ.

1	0	0	0
0	-1	0	0
0	0	1	0
0	0	0	1

Tras el plano YZ

$$\begin{matrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

Con el objetivo de obtener una imagen de dos dimensiones (sx, sy) de una imagen de tres dimensiones (x, y, z) es necesario el siguiente proceso :

Si consideramos el punto de vista como el punto P y queremos realizar la transformación de un segmento de recta AB que esta en un espacio de tres dimensiones, la pantalla es un plano sobre el cual la figura es proyectado. El plano de proyección es perpendicular a la linea OP, y a una distancia fija D (distancia entre P y el plano). Todos los puntos del esplacio (x, y, z) son proyectados sobre el plano (sx, sy), esto significa que uno o más puntos (x, y, z) pueden corresponder a un punto (sx, sy).

Los valores de sx y sy se obtienen de la relación entre triangulos semejantes, el desarrollo es puramente geometrica y la esplicación detallada la podemos ver en cualquier texto de geometria.

Implementación de la clase G3D usando OOP.

Descripción de la clase.

La clase tiene de nombre 'g3d' y tiene la siguiente definición :

```
class g3d
{
        float   d,s1,c1,s2,c2,x,y,z;
        float   theta, phi, rho;
        float   sx,sy,ox,oy, ex, ey, ez;
        short   fl,f,mx,my, cx, cy;
        float   MARGENX,MARGENY,SCALEX,SCALEY;
        float   yn[1024],yx[1024];
        void    transforma();
        void    esconde();
        void    evalua();
        float   redondea (float);

        public:

        HDC     hdc;
        void    init_g3d( short tx,short ty,
                          short centx,short centy,
                          float dist,float angx,float angz,
                          float vect,
                          float,float,float,
                          float,float,float,float);
        void    grafica(formula *, float xdesde, float xhasta, float xdelta,
                          float ydesde, float yhasta, float ydelta,
                          float zdesde, float zhasta);
};
```

Figura 10: Descripción de la clase G3D

Privados :

Son variables auxiliares y variables que sirven para almacenar los valores de los parametros.

Publicas :

La función 'init_g3d' que sirve para inicializar los valores de los parametros.
La función 'grafica' que realiza todo el proceso de graficación.

variable HDC, que apunta a la ventana a usar.

Todo el proceso de la tranformación que debe seguir un punto (x, y, z) esta escrita en la funcion 'transforma' dentro de la clase 'g3d'.

El proceso de esconder lineas esta escrito en la funcion 'esconde'.

Diseñado orientado a objetos, es por eso que la herramienta que se debía usar para implementación también debería soportar Orientación a Objetos OOP. BorlandC ++ 3.1 fué la herramienta escogida debido a las facilidades que brinda y a la experiencia anteriormente adquirida en otros compiladores similares de la misma compañía.

Implementación fué modular, con cada módulo del diseño en un módulo de la implementación. También se hizo uso de recursos que Windows necesita para interactuar con su ambiente, cosa que se logró utilizando una herramienta de Borland llamada Resource Workshop, que permite adicionar al programa, recursos como iconos, bitmaps, cajas de diálogos, cursores, etc. Todos estos recursos se encuentran en un sólo módulo de GrafWin.

	Clases contenidas
[illegible].CPP	Aplicación, Ventanas, Cajas de diálogo
[illegible].CPP, GRAFICO.HPP	G3D
[illegible].CPP, PRINTER.HPP	Tprint
[illegible].CPP, FORMULA.HPP	Formula
[illegible].RC	Recursos usados por GrafWin
[illegible].HPJ	Archivo de definición del Help
[illegible].RTF	Archivo fuente del Help
[illegible].HLP	Archivo de Ayuda (HELP)

Figura 11: Módulos de GrafWin

Descripción de Módulos

GRAFWIN.CPP

Este módulo que contiene toda la información que necesita la aplicación para interactuar con el ambiente operativo, es decir con Windows 3.1. Contiene las clases:

- **Aplicación .-** Encargada del manejo de la aplicación y su interacción con Windows.
- **Ventana.-** Encargada del manejo de las ventanas de la aplicación y su coexistencia con las de otras aplicaciones.
- **Cajas de Diálogo.-** Encargada del manejo de todas las cajas de diálogo que la aplicación usa para entrada de datos.

GRAFICO.CPP

Este módulo que contiene la clase G3D, que es la grafica la función, haciendo uso de la as facilidades que brinda el GDI (Graphics Device Interfase) de Windows. Esta clase es independiente a las otras, pués no depende de ninguna, más bien puede usársela en cualquier otro problema similar. Contiene la clase :

- **G3D.-** Encargada de graficar la función que se desea en la pantalla que se le especifique por medio de su dispositivo de contexto, que debe ser seteado antes de la graficación. Debido a que G3D grafica en cualquier dispositivo de contexto, ésta es tan flexible que puede imprimir lo mismo en pantalla, como en una impresora, plotter o cualquier otro dispositivo similar.

PRINT.CPP

Este módulo que contiene la clase de impresión:

- **TPrint.-** Encargada de obtener el dispositivo de contexto de la impresora seteada como default en Windows y de brindar facilidades de Inicio y Fin de trabajo, Inicio y Fin de página, cancelación del trabajo, etc..

FORMULA.CPP

Contiene la clase que se encarga de validar y evaluar la expresión:

- **Formula.-** Encargada de validar y evaluar la expresión, creando internamente el árbol de evaluación que le permite validar y al mismo tiempo evaluar dicha función. Una limitación de la clase es que sólo acepta expresiones en Notación INFIX.

GRAFWIN.RC

Contiene todos los recursos que GrafWin utiliza para su interacción con el usuario. Estos recursos son:

Recurso	Tipo	Descripción
[illegible]Dialog	Caja de Diálogo	Entrada de los parámetros de graficación
[illegible]Dialog	Caja de Diálogo	Entrada de la función
[illegible]	Bitmap	Gráfico de la portada
[illegible]	Icono	Icono de la aplicación
[illegible]MsgBox	Caja de Diálogo	Reseña de la autoría de la aplicación
[illegible]	Menú	Menú Principal de la Aplicación
[illegible]	Caja de Diálogo	Ventana de Portada

Figura 12: Recursos de GrafWin

GRAFWIN.HPJ

Archivo de definición de la Ayuda de GrafWin. Contiene la información necesaria para que el compilador de Help (HELP COMPILER para Windows 3.1) HC31 provisto por Borland compile sin ningún problema el archivo RTF.

GRAFWIN.RTF

Archivo fuente de la ayuda de GrafWin. Es un archivo editado en Microsoft Word 6.0, que contiene toda la ayuda de GrafWin.

GRAFWIN.HLP

Es el archivo RTF compilado y listo para ser usado por la aplicación. Puede también ser leído desde cualquier otra parte, haciendo uso del WINHELP.EXE, que es el programa que provee Windows para poder accesar a los archivos HLP.

Listados

A continuación se muestran los listados de los módulos de GrafWin.

GRAFWIN.CPP

```
# define WIN31
# define MAXDATOS 10
# define MAXDES 80

#include <owl.h>
#include <math.h>
#include <stdio.h>
#include <windows.h>
#include <string.h>
#include <dialog.h>
#include <commdlg.h>
#include <filewnd.h>
#include "graf3d.h"
#include "grafhelp.h"
#include "formula.hpp"
#include "grafico.hpp"
#include "printer.hpp"

char *IniFileName="grafwin.ini";
//edison: poner path completo para que cargue default.
//char *IniFileName="grafwin.ini";


char far *ProgramName="Grafico de Funciones";
char FileName[MAXFORMULA+1];

BOOL SaveDefaults;

struct ParaGrafic
{
        char Rangos_x[2][MAXDATOS+1];
        char Rangos_y[2][MAXDATOS+1];
        char Rangos_z[2][MAXDATOS+1];
        char Escala_x[MAXDATOS+1];
        char Escala_y[MAXDATOS+1];
        char Escala_z[MAXDATOS+1];
        char Delta_x[MAXDATOS+1];
        char Delta_y[MAXDATOS+1];
        char PtoVista_x[MAXDATOS+1];
        char PtoVista_z[MAXDATOS+1];
        char PtoVista_d[MAXDATOS+1];
  char Descripcion[MAXDES+1];
};

void SaveParametros(ParaGrafic&,char *);
```

```
void GetParametros(ParaGrafic&,LPSTR);

formula          Form;
g3d                    gf,graf;

ParaGrafic ParamGraf,Default;


static char  szIconName[]="Grafwin";
static char  szIApplName[]="GrafMenu";
static char  szCursorName[]="GrafCursor";

int  xClientView,yClientView;

class TgrafDialog:public TDialog
{
 public :
        TgrafDialog(PTWindowsObject  AParent,LPSTR AName);
        virtual void WMCommand(TMessage& Message)=[WM_FIRST+WM_COMMAND];
        virtual void SetupWindow();
};

class TFormDialog:public TDialog
{
       public :
        TFormDialog(PTWindowsObject  AParent,LPSTR AName);
        virtual void WMCommand(TMessage& Message)=[WM_FIRST+WM_COMMAND];
        virtual void SetupWindow();
};


TgrafDialog::TgrafDialog(PTWindowsObject  AParent,LPSTR AName):TDialog(AParent, AName)
{
}

TFormDialog::TFormDialog(PTWindowsObject  AParent,LPSTR AName):TDialog(AParent, AName)
{
}

void  TgrafDialog::SetupWindow()
{
        if (SaveDefaults)
        {
                SetDlgItemText(HWindow,CM_RANGO_X1,Default.Rangos_x[0]);
                SetDlgItemText(HWindow,CM_RANGO_X2,Default.Rangos_x[1]);
                SetDlgItemText(HWindow,CM_RANGO_Y1,Default.Rangos_y[0]);
                SetDlgItemText(HWindow,CM_RANGO_Y2,Default.Rangos_y[1]);
                SetDlgItemText(HWindow,CM_RANGO_Z1,Default.Rangos_z[0]);
                SetDlgItemText(HWindow,CM_RANGO_Z2,Default.Rangos_z[1]);
                SetDlgItemText(HWindow,CM_DOMINIO_X,Default.Delta_x);
                SetDlgItemText(HWindow,CM_DOMINIO_Y,Default.Delta_y);
                SetDlgItemText(HWindow,CM_ESCALA_X,Default.Escala_x);
```

```
                SetDlgItemText(HWindow,CM_ESCALA_Y,Default.Escala_y);
                SetDlgItemText(HWindow,CM_ESCALA_Z,Default.Escala_z);
                SetDlgItemText(HWindow,CM_PUNTVIST_X,Default.PtoVista_x);
                SetDlgItemText(HWindow,CM_PUNTVIST_Y,Default.PtoVista_z);
                SetDlgItemText(HWindow,CM_PUNTVIST_Z,Default.PtoVista_d);
                SetDlgItemText(HWindow,CM_DESCRIPCION,Default.Descripcion);
        }
        else
        {
           SetDlgItemText(HWindow,CM_RANGO_X1,ParamGraf.Rangos_x[0]);
    SetDlgItemText(HWindow,CM_RANGO_X2,ParamGraf.Rangos_x[1]);
    SetDlgItemText(HWindow,CM_RANGO_Y1,ParamGraf.Rangos_y[0]);
    SetDlgItemText(HWindow,CM_RANGO_Y2,ParamGraf.Rangos_y[1]);
    SetDlgItemText(HWindow,CM_RANGO_Z1,ParamGraf.Rangos_z[0]);
    SetDlgItemText(HWindow,CM_RANGO_Z2,ParamGraf.Rangos_z[1]);
    SetDlgItemText(HWindow,CM_DOMINIO_X,ParamGraf.Delta_x);
    SetDlgItemText(HWindow,CM_DOMINIO_Y,ParamGraf.Delta_y);
    SetDlgItemText(HWindow,CM_ESCALA_X,ParamGraf.Escala_x);
    SetDlgItemText(HWindow,CM_ESCALA_Y,ParamGraf.Escala_y);
    SetDlgItemText(HWindow,CM_ESCALA_Z,ParamGraf.Escala_z);
    SetDlgItemText(HWindow,CM_PUNTVIST_X,ParamGraf.PtoVista_x);
    SetDlgItemText(HWindow,CM_PUNTVIST_Y,ParamGraf.PtoVista_z);
                SetDlgItemText(HWindow,CM_PUNTVIST_Z,ParamGraf.PtoVista_d);
                SetDlgItemText(HWindow,CM_DESCRIPCION,ParamGraf.Descripcion);
        }
  return;
}


void  TFormDialog::SetupWindow()
{
 SetDlgItemText(HWindow,IDD_FORMULA,Form.expresion);
}

class TMainWindow:public TWindow
{
 public:
        TMainWindow(PTWindowsObject Parent, LPSTR ATitle);
   virtual void SetupWindow(void);
        virtual void Grafhelp(TMessage& Message)=[CM_FIRST + CM_HELP_CONTENIDO];
        virtual void About(TMessage& Message)=[CM_FIRST + CM_ACERCA_DE];
        virtual void GrafInput(TMessage& ) = [CM_FIRST  + CM_PARAMETROS];
        virtual void IngFormula(TMessage&)=[CM_FIRST + CM_FORMULA];
        virtual void Paint(HDC hdc,PAINTSTRUCT&  Paintinfo);
        virtual void WMSize(TMessage& Message ) = [WM_FIRST + WM_SIZE];
        virtual void GetWindowClass(WNDCLASS& Wndclass);
        virtual void Salir(void)=[CM_FIRST+CM_SALIR];
        virtual void GrabarPar(void)=[CM_FIRST+CM_RANGOS];
        virtual void Abrir(TMessage&)=[CM_FIRST + CM_ABRIR];
        virtual void Grabar(void)=[CM_FIRST + CM_GRABAR];
        virtual void GrabarComo(void)=[CM_FIRST + CM_GRABAR_COMO];
        virtual void GrabarBMP(void)=[CM_FIRST + CM_GRABAR_BMP];
        virtual void Nuevo(void)=[CM_FIRST + CM_NUEVO];
        void GrabarFormula(void);
```

```
        char szName[256];

};

typedef TMainWindow * PMainwindow;

class Grafwin:public TApplication
{
 public:
        Grafwin(LPSTR AName,HINSTANCE hInstance,HINSTANCE hPrevInstance,LPSTR lpCmd,int
nCmdShow):TApplication(AName,hInstance,hPrevInstance,lpCmd,nCmdShow){};
        virtual void InitMainWindow(void);
};


void TgrafDialog::WMCommand(TMessage& Message)
{
        char valor[MAXDATOS+1];
        switch(Message.WParam)
        {
                case  998:
                    if(SaveDefaults)
                            WinHelp( HWindow,
"GRAFWIN.HLP",HELP_CONTEXT,HELP_DEFAULT);
                    else
                            WinHelp( HWindow,
"GRAFWIN.HLP",HELP_CONTEXT,HELP_PARAMETROS);

                break;
                    case  IDOK:
      if(SaveDefaults)
                            {

        GetDlgItemText(HWindow,CM_RANGO_X1,Default.Rangos_x[0],MAXDATOS);

        GetDlgItemText(HWindow,CM_RANGO_X2,Default.Rangos_x[1],MAXDATOS);

        GetDlgItemText(HWindow,CM_RANGO_Y1,Default.Rangos_y[0],MAXDATOS);

        GetDlgItemText(HWindow,CM_RANGO_Y2,Default.Rangos_y[1],MAXDATOS);

        GetDlgItemText(HWindow,CM_RANGO_Z1,Default.Rangos_z[0],MAXDATOS);

        GetDlgItemText(HWindow,CM_RANGO_Z2,Default.Rangos_z[1],MAXDATOS);

        GetDlgItemText(HWindow,CM_ESCALA_X,Default.Escala_x,MAXDATOS);

        GetDlgItemText(HWindow,CM_ESCALA_Y,Default.Escala_y,MAXDATOS);

        GetDlgItemText(HWindow,CM_ESCALA_Z,Default.Escala_z,MAXDATOS);

        GetDlgItemText(HWindow,CM_DOMINIO_X,Default.Delta_x,MAXDATOS);

        GetDlgItemText(HWindow,CM_DOMINIO_Y,Default.Delta_y,MAXDATOS);
```

```
GetDlgItemText(HWindow,CM_PUNTVIST_X,Default.PtoVista_x,MAXDATOS);
GetDlgItemText(HWindow,CM_PUNTVIST_Y,Default.PtoVista_z,MAXDATOS);
GetDlgItemText(HWindow,CM_PUNTVIST_Z,Default.PtoVista_d,MAXDATOS);
GetDlgItemText(HWindow,CM_DESCRIPCION,Default.Descripcion,MAXDES);
                    SaveParametros(Default,IniFileName);
                }
                else
                {
GetDlgItemText(HWindow,CM_RANGO_X1,ParamGraf.Rangos_x[0],MAXDATOS);
GetDlgItemText(HWindow,CM_RANGO_X2,ParamGraf.Rangos_x[1],MAXDATOS);
GetDlgItemText(HWindow,CM_RANGO_Y1,ParamGraf.Rangos_y[0],MAXDATOS);
GetDlgItemText(HWindow,CM_RANGO_Y2,ParamGraf.Rangos_y[1],MAXDATOS);
GetDlgItemText(HWindow,CM_RANGO_Z1,ParamGraf.Rangos_z[0],MAXDATOS);
GetDlgItemText(HWindow,CM_RANGO_Z2,ParamGraf.Rangos_z[1],MAXDATOS);
GetDlgItemText(HWindow,CM_ESCALA_X,ParamGraf.Escala_x,MAXDATOS);
GetDlgItemText(HWindow,CM_ESCALA_Y,ParamGraf.Escala_y,MAXDATOS);
GetDlgItemText(HWindow,CM_ESCALA_Z,ParamGraf.Escala_z,MAXDATOS);
GetDlgItemText(HWindow,CM_DOMINIO_X,ParamGraf.Delta_x,MAXDATOS);
GetDlgItemText(HWindow,CM_DOMINIO_Y,ParamGraf.Delta_y,MAXDATOS);
GetDlgItemText(HWindow,CM_PUNTVIST_X,ParamGraf.PtoVista_x,MAXDATOS);
GetDlgItemText(HWindow,CM_PUNTVIST_Y,ParamGraf.PtoVista_z,MAXDATOS);
GetDlgItemText(HWindow,CM_PUNTVIST_Z,ParamGraf.PtoVista_d,MAXDATOS);
GetDlgItemText(HWindow,CM_DESCRIPCION,ParamGraf.Descripcion,MAXDES);
                }
                EndDialog(HWindow,TRUE);
                break;
        case  IDCANCEL:
                EndDialog(HWindow,FALSE);
                break;
        case CM_DEFAULT:
                SetDlgItemText(HWindow,CM_RANGO_X1,Default.Rangos_x[0]);
                SetDlgItemText(HWindow,CM_RANGO_X2,Default.Rangos_x[1]);
                SetDlgItemText(HWindow,CM_RANGO_Y1,Default.Rangos_y[0]);
                SetDlgItemText(HWindow,CM_RANGO_Y2,Default.Rangos_y[1]);
                SetDlgItemText(HWindow,CM_RANGO_Z1,Default.Rangos_z[0]);
```

```
                        SetDlgItemText(HWindow,CM_RANGO_Z2,Default.Rangos_z[1]);
                        SetDlgItemText(HWindow,CM_DOMINIO_X,Default.Delta_x);
                        SetDlgItemText(HWindow,CM_DOMINIO_Y,Default.Delta_y);
                        SetDlgItemText(HWindow,CM_ESCALA_X,Default.Escala_x);
                        SetDlgItemText(HWindow,CM_ESCALA_Y,Default.Escala_y);
                        SetDlgItemText(HWindow,CM_ESCALA_Z,Default.Escala_z);
                        SetDlgItemText(HWindow,CM_PUNTVIST_X,Default.PtoVista_x);
                        SetDlgItemText(HWindow,CM_PUNTVIST_Y,Default.PtoVista_z);
                        SetDlgItemText(HWindow,CM_PUNTVIST_Z,Default.PtoVista_d);
                        SetDlgItemText(HWindow,CM_DESCRIPCION,Default.Descripcion);

        }
}


void TFormDialog::WMCommand(TMessage& Message)
{
        switch(Message.WParam)
        {
                case 998:
                        WinHelp( HWindow,
"GRAFWIN.HLP",HELP_CONTEXT,HELP_FORMULA);
                        break;

                case IDOK:
                GetDlgItemText(HWindow,IDD_FORMULA,Form.expresion,MAXFORMULA);
                        Form.Create(Form.expresion);
              if (!Form.GetStatus())
                        {

        MessageBox(HWindow,"InValida",ProgramName,MB_OK|MB_ICONASTERISK);
                                return;
                        }
                        EndDialog(HWindow,TRUE);
                        break;
                case IDCANCEL:
                        EndDialog(HWindow,FALSE);
                        break;
        }
}


void TMainWindow::Grafhelp(TMessage&)
{

  WinHelp( HWindow, "GRAFWIN.HLP",HELP_INDEX,0L );


}

void TMainWindow::Nuevo(void)
{
        char Name[MAXFORMULA+1];
        strcpy(FileName,"Sin titulo");
```

```
        sprintf(Name,"%s - [%s]",ProgramName,FileName);
        Form.Create("");
        ParamGraf=Default;
        InvalidateRect(HWindow,NULL,TRUE);
        UpdateWindow(HWindow);
        SetCaption(Name);
    return;
}


void TMainWindow::Grabar(void)
{
        if (!stricmp(FileName,"Sin titulo"))
                GrabarComo();
    else
                SaveParametros(ParamGraf,(LPSTR)szName);
    return;
}


void TMainWindow::GrabarFormula(void)
{
        char Name[MAXFORMULA+1];
        WritePrivateProfileString("Formula","Expresion",Form.expresion,FileName);
        sprintf(Name,"%s - [%s]",ProgramName,FileName);
        SetCaption(Name);
    return;
}

void TMainWindow::GrabarComo(void)
{
        OPENFILENAME ofnTemp;
        char szTemp[] = "Archivos GrafWin (*.grf)\0*.grf\0Todos los archivos (*.*)\0*.*\0";
        ofnTemp.lStructSize = sizeof( OPENFILENAME );
        ofnTemp.hwndOwner = HWindow;                            // An invalid hWnd causes non-
modality
        ofnTemp.hInstance = 0;
        ofnTemp.lpstrFilter = (LPSTR)szTemp;      // See previous note concerning string
        ofnTemp.lpstrCustomFilter = NULL;
        ofnTemp.nMaxCustFilter = 0;
        ofnTemp.nFilterIndex = 1;
        ofnTemp.lpstrFile = (LPSTR)szName;                  // Stores the result in this variable
        ofnTemp.nMaxFile = sizeof( szName );
        ofnTemp.lpstrFileTitle = NULL;
        ofnTemp.nMaxFileTitle = 0;
        ofnTemp.lpstrInitialDir = NULL;
        ofnTemp.lpstrTitle = Title;                             // Title for dialog
        ofnTemp.Flags = OFN_OVERWRITEPROMPT;
        ofnTemp.nFileOffset = 0;
        ofnTemp.nFileExtension = 0;
        ofnTemp.lpstrDefExt = "*";
        ofnTemp.lCustData = NULL;
        ofnTemp.lpfnHook = NULL;
        ofnTemp.lpTemplateName = NULL;
```

```
        if(GetSaveFileName( &ofnTemp ) != TRUE)
        {
                DWORD Errval; // Error value
                char Errstr[50]="GetOpenFileName retornó Error #";
                char buf[5];        // Error buffer

                Errval=CommDlgExtendedError();
                if(Errval!=0)  // 0 value means user selected Cancel
                {
                        sprintf(buf,"%ld",Errval);
                        strcat(Errstr,buf);

        MessageBox(HWindow,Errstr,ProgramName,MB_OK|MB_ICONEXCLAMATION);
              return;
                }
                else
                        return;
        }
        InvalidateRect( HWindow, NULL, TRUE ); // Repaint to display the new name
        SavaParametros(ParamGraf,(LPSTR)szName);
        char Name[MAXFORMULA+1];
        strcpy(FileName,szName);
        sprintf(Name,"%s - [%s]",ProgramName,FileName);
        UpdateWindow(HWindow);
        SetCaption(Name);
}

void TMainWindow::GrabarBMP(void)
{
        char str_o[161];
        TPrint Print;

        graf.hdc=Print.GetDC();

        if (!graf.hdc)
                return;

//      SetMapMode( graf.hdc, MM_ISOTROPIC);
        graf.init_g3d(xClientView, yClientView,
                                        -1, -1,
                                        400,
                                        atof(ParamGraf.PtoVista_x),
                                        atof(ParamGraf.PtoVista_z),
                                        atof(ParamGraf.PtoVista_d),
                                        atof(ParamGraf.Escala_x),
                                        atof(ParamGraf.Escala_y),
                                        atof(ParamGraf.Escala_z),
                                        Print.MARGENX,
                                        Print.MARGENY,Print.SCALEX,Print.SCALEY);


        Print.StartDoc();
        Print.StartPage();

        sprintf(str_o,"%s : Z = %s",ParamGraf.Descripcion,Form.expresion);
```

```
        TextOut(graf.hdc,10,0,str_o,strlen(str_o));

        graf.grafica(&Form, atof(ParamGraf.Rangos_x[0]),
                                                        atof(ParamGraf.Rangos_x[1]),
                                                        atof(ParamGraf.Delta_x),
                                                        atof(ParamGraf.Rangos_y[0]),
                                                        atof(ParamGraf.Rangos_y[1]),
                                                        atof(ParamGraf.Delta_y),
                                                        atof(ParamGraf.Rangos_z[0]),
                                                        atof(ParamGraf.Rangos_z[1]));

        Print.EndPage();
        Print.EndDoc();
        return;
}

void TMainWindow::Abrir(TMessage&)
{
        OPENFILENAME ofnTemp;
        char szTemp[] = "Archivos GrafWin (*.grf)\0*.grf\0Todos los archivos (*.*)\0*.*\0";
        ofnTemp.lStructSize = sizeof( OPENFILENAME );
        ofnTemp.hwndOwner = HWindow;                    // An invalid hWnd causes non-
modality
        ofnTemp.hInstance = 0;
        ofnTemp.lpstrFilter = (LPSTR)szTemp;    // See previous note concerning string
        ofnTemp.lpstrCustomFilter = NULL;
        ofnTemp.nMaxCustFilter = 0;
        ofnTemp.nFilterIndex = 1;
        ofnTemp.lpstrFile = (LPSTR)szName;              // Stores the result in this variable
        ofnTemp.nMaxFile = sizeof( szName );
        ofnTemp.lpstrFileTitle = NULL;
        ofnTemp.nMaxFileTitle = 0;
        ofnTemp.lpstrInitialDir = NULL;
        ofnTemp.lpstrTitle = Title;                     // Title for dialog
        ofnTemp.Flags = OFN_FILEMUSTEXIST | OFN_HIDEREADONLY |
OFN_PATHMUSTEXIST;
        ofnTemp.nFileOffset = 0;
        ofnTemp.nFileExtension = 0;
        ofnTemp.lpstrDefExt = "grf";
        ofnTemp.lCustData = NULL;
        ofnTemp.lpfnHook = NULL;
        ofnTemp.lpTemplateName = NULL;
        if(GetOpenFileName( &ofnTemp ) != TRUE)
        {
                DWORD Errval; // Error value
                char Errstr[50]="GetOpenFileName retornó Error #";
                char buf[5];        // Error buffer

                Errval=CommDlgExtendedError();
                if(Errval!=0)  // 0 value means user selected Cancel
                {
                        sprintf(buf,"%ld",Errval);
                        strcat(Errstr,buf);
```

```
        MessageBox(HWindow,Errstr,ProgramName,MB_OK|MB_ICONEXCLAMATION);

                }
                return;
        }
        InvalidateRect( HWindow, NULL, TRUE ); // Repaint to display the new name
        GetParametros(ParamGraf,(LPSTR)szName);
        Form.Create(Form.expresion);
        if (!Form.GetStatus())
           {
                MessageBox(HWindow,"Función
InValida",ProgramName,MB_OK|MB_ICONASTERISK);
                return;
        }
        char Name[MAXFORMULA+1];
        strcpy(FileName,szName);
        sprintf(Name,"%s - [%s]",ProgramName,FileName);
        SetCaption(Name);
}

void  TMainWindow::GrafInput(TMessage&)
{
        SaveDefaults=FALSE;
        GetModule()->ExecDialog(new TgrafDialog(this,"GrafDiabox"));
        InvalidateRect(HWindow,NULL,TRUE);
        UpdateWindow(HWindow);
}

void  TMainWindow::IngFormula(TMessage&)
{
        GetModule()->ExecDialog(new TFormDialog(this,"DlgFormula"));
        InvalidateRect(HWindow,NULL,TRUE);
        UpdateWindow(HWindow);
}

void TMainWindow::GrabarPar(void)

{
        SaveDefaults=TRUE;
        GetModule()->ExecDialog(new TgrafDialog(this,"GrafDiaBox"));
        InvalidateRect(HWindow,NULL,TRUE);
        UpdateWindow(HWindow);
        return;
}

void TMainWindow::Salir(void)
{
        PostQuitMessage(0);
        return;
}

void TMainWindow::WMSize(TMessage&  Message)
{
```

```
        xClientView=LOWORD(Message.LParam);
        yClientView=HIWORD(Message.LParam);
}

void TMainWindow::About(TMessage&)
{
        GetApplication()->ExecDialog(new TgrafDialog(this,"AboutDiabox"));
    return;
}

void TMainWindow::GetWindowClass(WNDCLASS & WndClass)
{
        TWindow::GetWindowClass(WndClass);
        WndClass.hbrBackground=GetStockObject(WHITE_BRUSH);
        WndClass.hIcon = LoadIcon( GetApplication()->hInstance,
MAKEINTRESOURCE(GRAFWIN));
}

void TMainWindow::SetupWindow(void)
{
        TWindow::SetupWindow();
        GetApplication()->ExecDialog(new TDialog(this,INICIO));
}

void TMainWindow::Paint(HDC hdc , PAINTSTRUCT&)
{
        char str_o[161];
        HMENU HMenu;
        HBRUSH          hNBrush;
        HFONT hNFont;
        static LOGFONT lf;
//      unsigned int iTotalWedge[maxnumwedge+1];
        int i,y1,y2,iNWedges;

        if( !Form.GetStatus() )
                return;
        SetMapMode( hdc, MM_ISOTROPIC);
        SetWindowExtEx(hdc, 500,500,NULL);
//      SetViewportExtEx(hdc, xClientView, -yClientView, NULL);
//      SetViewportOrgEx(hdc, xClientView/2, yClientView/2,NULL);
        SetViewportExtEx(hdc, xClientView, yClientView, NULL);
        SetViewportOrgEx(hdc, 0, 0,NULL);
        lf.lfCharSet=0;
        lf.lfPitchAndFamily=34;
        lf.lfHeight=16;
        hNFont=CreateFontIndirect(&lf);
        SelectObject(hdc,hNFont);

// gf.hdc=BeginPaint(hWnd,&ps);

        gf.hdc=hdc;

//      gf.init_g3d(640, 440, -1, -1, 400, 0, 0, 60);
        gf.init_g3d(xClientView, yClientView,
```

```
                                        -1, -1,
                                        400,
                                        atof(ParamGraf.PtoVista_x),
                                        atof(ParamGraf.PtoVista_z),
                                        atof(ParamGraf.PtoVista_d),
                                        atof(ParamGraf.Escala_x),
                                        atof(ParamGraf.Escala_y),
                                        atof(ParamGraf.Escala_z),
                                        0.0,0.0,1.0,1.0);

        sprintf(str_o,"%s : Z = %s",ParamGraf.Descripcion,Form.expresion);

        TextOut(gf.hdc,10,0,str_o,strlen(str_o));

        gf.grafica(&Form, atof(ParamGraf.Rangos_x[0]),
                                                        atof(ParamGraf.Rangos_x[1]),
                                                        atof(ParamGraf.Delta_x),
                                                        atof(ParamGraf.Rangos_y[0]),
                                                        atof(ParamGraf.Rangos_y[1]),
                                                        atof(ParamGraf.Delta_y),
                                                        atof(ParamGraf.Rangos_z[0]),
                                                        atof(ParamGraf.Rangos_z[1]));

//                      ValidateRect(hWnd, NULL);
//                      EndPaint(hWnd, NULL);

/*      iNWedges=0;
        for(i=0; i<maxnumwedge; i++)
        {
        if( iWedgesize[i]!=0 ) iNWedges++;
        }

        iTotalWedge[0]=0;
        for( i=0; i<iNWedges; i++)
                iTotalWedge[i+1]=iTotalWedge[i]+iWedgesize[i];

        SetMapMode( hdc, MM_ISOTROPIC);
        SetWindowExtEx(hdc, 500,500,NULL);
        SetViewportExtEx(hdc, xClientView, -yClientView, NULL);
        SetViewportOrgEx(hdc, xClientView/2, yClientView/2,NULL);
        lf.lfCharSet=0;
        lf.lfPitchAndFamily=34;
        lf.lfHeight=xClientView/50;
        hNFont=CreateFontIndirect(&lf);
        SelectObject(hdc,hNFont);
        TextOut(hdc,(-150-(strlen(szTString)*lf.lfWidth/2)),
                240, szTString, strlen(szTString));

                y1=-100;
                y2=y1+15;
                for( i=0; i<iNWedges; i++)
                {
                        hNBrush=CreateSolidBrush(lColor[i]);
                        SelectObject(hdc, hNBrush);
```

```
                        Pie(hdc, -300, 200, 100, -200,
                                (int) (radius*cos(2*M_PI*iTotalWedge[i]/
                                iTotalWedge[iNWedges])),
                                (int) (radius*sin(2*M_PI*iTotalWedge[i]/
                                iTotalWedge[iNWedges])),
                                (int) (radius*cos(2*M_PI*iTotalWedge[i+1]/
                                iTotalWedge[iNWedges])),
                                (int) (radius*sin(2*M_PI*iTotalWedge[i+1]/
                                iTotalWedge[iNWedges])));
                        Rectangle(hdc, 130, y1, 145, y2);
                        TextOut(hdc, 155, y2+2, szTLabel[i], strlen(szTLabel[i]));
                        y1=y2+5;
    y2+=20;
                } */
}

TMainWindow::TMainWindow(PTWindowsObject AParent, LPSTR ATitle)
                        :TWindow(AParent, ATitle )
{
        AssignMenu("GrafMenu");
}


void Grafwin::InitMainWindow(void)
{
        char Name[MAXFORMULA+1];
 sprintf(Name,"%s - [%s]",ProgramName,FileName);
        MainWindow=new TMainWindow(NULL, Name);

}

void GetParametros(ParaGrafic& ParamG,char *IniFileName)
{
        GetPrivateProfileString("Parametros","RanMinX","-
10",ParamG.Rangos_x[0],MAXDATOS,IniFileName);
        GetPrivateProfileString("Parametros","RanMaxX","10",ParamG.Rangos_x[1],MAXDATOS,IniFi
leName);
        GetPrivateProfileString("Parametros","RanMinY","-
10",ParamG.Rangos_y[0],MAXDATOS,IniFileName);
        GetPrivateProfileString("Parametros","RanMaxY","10",ParamG.Rangos_y[1],MAXDATOS,IniFi
leName);
        GetPrivateProfileString("Parametros","RanMinZ","-
10",ParamG.Rangos_z[0],MAXDATOS,IniFileName);
        GetPrivateProfileString("Parametros","RanMaxZ","10",ParamG.Rangos_z[1],MAXDATOS,IniFil
eName);
        GetPrivateProfileString("Parametros","DeltaX","1",ParamG.Delta_x,MAXDATOS,IniFileName);
        GetPrivateProfileString("Parametros","DeltaY","1",ParamG.Delta_y,MAXDATOS,IniFileName);
        GetPrivateProfileString("Parametros","AnguloX","0",ParamG.PtoVista_x,MAXDATOS,IniFileN
ame);
        GetPrivateProfileString("Parametros","AnguloY","0",ParamG.PtoVista_z,MAXDATOS,IniFileN
ame);
        GetPrivateProfileString("Parametros","DistanciaZ","100",ParamG.PtoVista_d,MAXDATOS,IniFi
leName);
```

```
        GetPrivateProfileString("Parametros","EscalaX","1",ParamG.Escala_x,MAXDATOS,IniFileNam
e);
        GetPrivateProfileString("Parametros","EscalaY","1",ParamG.Escala_y,MAXDATOS,IniFileNam
e);
        GetPrivateProfileString("Parametros","EscalaZ","1",ParamG.Escala_z,MAXDATOS,IniFileName
);
        GetPrivateProfileString("Parametros","Formula","",Form.expresion,MAXFORMULA,IniFileNam
e);
        GetPrivateProfileString("Parametros","Descripcion","Sin
nombre",ParamG.Descripcion,MAXDES,IniFileName);
        return;
}

void SaveParametros(ParaGrafic& ParamG,LPSTR IniFileName)
{
        WritePrivateProfileString("Parametros","RanMinX",ParamG.Rangos_x[0],IniFileName);
        WritePrivateProfileString("Parametros","RanMaxX",ParamG.Rangos_x[1],IniFileName);
        WritePrivateProfileString("Parametros","RanMinY",ParamG.Rangos_y[0],IniFileName);
        WritePrivateProfileString("Parametros","RanMaxY",ParamG.Rangos_y[1],IniFileName);
        WritePrivateProfileString("Parametros","RanMinZ",ParamG.Rangos_z[0],IniFileName);
        WritePrivateProfileString("Parametros","RanMaxZ",ParamG.Rangos_z[1],IniFileName);
        WritePrivateProfileString("Parametros","DeltaX",ParamG.Delta_x,IniFileName);
        WritePrivateProfileString("Parametros","DeltaY",ParamG.Delta_y,IniFileName);
        WritePrivateProfileString("Parametros","AnguloX",ParamG.PtoVista_x,IniFileName);
        WritePrivateProfileString("Parametros","AnguloY",ParamG.PtoVista_z,IniFileName);
        WritePrivateProfileString("Parametros","DistanciaZ",ParamG.PtoVista_d,IniFileName);
        WritePrivateProfileString("Parametros","EscalaX",ParamG.Escala_x,IniFileName);
        WritePrivateProfileString("Parametros","EscalaY",ParamG.Escala_y,IniFileName);
        WritePrivateProfileString("Parametros","EscalaZ",ParamG.Escala_z,IniFileName);
        WritePrivateProfileString("Parametros","Formula",Form.expresion,IniFileName);
        WritePrivateProfileString("Parametros","Descripcion",ParamG.Descripcion,IniFileName);
        return;
}

int PASCAL WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
                                LPSTR lpCmd, int nCmdShow)

{
   /*   HINSTANCE HLibrary;
        HLibrary=LoadLibrary ("BWCC.DLL");
        if ((UINT)HLibrary <= 32){
                MessageBox(NULL, "No se Inicializo BWCC.DLL", "Error", MB_OK);
                return (UINT)HLibrary;
        }  */
        strcpy(FileName,"Sin titulo");
        Grafwin grafwin("GRAFWIN", hInstance, hPrevInstance,lpCmd, nCmdShow);
        grafwin.nCmdShow=SW_SHOWMAXIMIZED;
        GetParametros(Default,IniFileName);
        ParamGraf=Default;
        memset(Form.expresion,'\0',MAXFORMULA);
        grafwin.Run();
   /*   FreeLibrary (HLibrary);*/
        return(grafwin.Status);

}
```

GRAFICO.CPP

```
# include <windows.h>
# include <math.h>
# include <stdio.h>
# include <string.h>
# include "formula.hpp"
# include "grafico.hpp"
# include "3dwin.h"

# define BLANCO   RGB(255,255,255)
# define ROJO     RGB(255,0,0)
# define AMARILLO RGB(255,255,40)
# define NEGRO    RGB(0,0,0)

extern formula Form;
extern int xClientView;

void g3d::init_g3d( short tx,short ty,
                                                short centx,short centy,
                                                float dist,float angx,float angz,
                                                float vect,
                                                float eex, float eey, float eez,
                                                float MX,float MY,
                                                float SX,float SY)
{
        short i;
  if( tx>=MXMIN && tx<=MXMAX)
        mx=tx;
  else
                mx=MXMAX;
        if( ty>=MYMIN && ty<=MYMAX)
                my=ty;
        else
                my=MYMAX;
        if( centx>=CXMIN && centx<=CXMAX)
                cx=centx;
        else
                cx=mx/2;
        if( centy>=CYMIN && centy<=CYMAX)
                cy=centy;
        else
                cy=my/2;
        rho=vect;
        d=dist;
        theta=redondea(angx);
        phi=angz;
        s1=sin(theta * A_RAD);
        s2=sin(phi * A_RAD);
        c1=cos(theta * A_RAD);
        c2=cos(phi * A_RAD);
        ex=eex;
```

```
        ey=eey;
 ez=eez;
        for(i=0; i<=mx; i++){ yn[i]=(float)my; yx[i]=(float)0; }
        MARGENX=MX;
        MARGENY=MY;
        SCALEX=SX;
 SCALEY=SY;
}


void g3d::transforma()
{
        float xe,ye,ze;
        float ax,ay,az;
        ax=ex*x; ay=ey*y; az=ez*z;
        xe=-ax*s1+ay*c1;
        ye=-ax*c1*c2-ay*s1*c2+az*s2;
        ze=-ax*s2*c1-ay*s2*s1-az*c2+rho;
        sx=cx+d*xe/ze;
        sy=cy-d*ye/ze;
}



void g3d::esconde(void)
{
short fg=0,xp;
float dx,sl,yp;


if(fl==0){ fl=1; f=0; ox=sx; oy=sy; }
dx=ox-sx;
if(dx==0) dx=1;
sl=(oy-sy)/dx;
yp=oy;
if( sx<0 && ox<0 )
{
        ox=sx;oy=sy;
        return;
}
if( sx>mx && ox >mx)
{
        ox=sx;oy=sy;
        return;
}

for(xp=ox+1; xp<=sx && xp<=mx; xp+=1)
{
        fg=1;
        yp=yp+sl;
        if(xp<0 || xp>mx)
        {
                fg=0;
```

```
f=0;
        }
        else
        {
                if(yp>my || yp<0)
                {       fg=0;
                        f=0;
                }
                else
 {
                        if(yp<=yn[xp] || yp>=yx[xp])
                        {
                                if(fg!=0)
        {
                                        if(f==0) {

        MoveToEx(hdc,xp*SCALEX+MARGENX,yp*SCALEY+MARGENY,NULL);
                                                f=1;
                                        }
                                        LineTo(hdc,
xp*SCALEX+MARGENX,yp*SCALEY+MARGENY);

        SetPixel(hdc,xp*SCALEX+MARGENX,yp*SCALEY+MARGENY,NEGRO);
                                }

                        } else f=0; // else TextOut(hdc,xp,yp,"o",1);
        }
                if(yp<=yn[xp]) yn[xp]=yp;
                if(yp>=yx[xp]) yx[xp]=yp;
        }
}
ox=sx;oy=sy;

}


void g3d::grafica(formula *fobj,float xdesde, float xhasta, float xdelta,
                                                float ydesde, float yhasta, float ydelta,
                                                float zdesde, float zhasta)

{
        HPEN hPen;

        char    str_o[81];
        hPen=CreatePen(PS_SOLID,1,NEGRO);

        SelectObject(hdc,hPen);

        SetBkColor(hdc,BLANCO);

        SetTextColor(hdc,NEGRO);

        if (( theta>=0.0 && theta<45.0) || (theta>=315.0 && theta<=360.0))
        {
```

```
            for( x=xhasta; x>=xdesde; x-=xdelta)
            {
                    fl=0;
                    for( y=ydesde; y<=yhasta; y+=ydelta)
                    {
                            if(MARGENX==0)
        {
                                    sprintf(str_o, "X=%-10.4f Y=%-10.4f", x, y);
                                    TextOut(hdc,10,18,str_o,strlen(str_o));
        }
                            z=(float)fobj->EvaluaZ((double)x,(double)y);
                            if(fobj->error!=NOERROR)
                                    continue;
                            transforma();
                            esconde();
                    }
            }
}
    if (theta>=45.0 && theta<135.0)
    {
            for( y=yhasta; y>=ydesde; y-=ydelta)
            {
                    fl=0;
                    for( x=xhasta; x>=xdesde; x-=xdelta)
                    {
                            if(MARGENX==0)
        {
                                    sprintf(str_o, "X=%-10.4f Y=%-10.4f", x, y);
                                    TextOut(hdc,10,18,str_o,strlen(str_o));

        }
                            z=(float)fobj->EvaluaZ((double)x,(double)y);
                            if(fobj->error!=NOERROR)
                                    continue;
                            transforma();
                            esconde();
                    }
            }
}
    if (theta>=135.0 && theta<225.0)
    {
            for( x=xdesde; x<=xhasta; x+=xdelta)
            {
                    fl=0;
                    for( y=yhasta; y>=ydesde; y-=ydelta)
                    {
                            if(MARGENX==0)
        {
                                    sprintf(str_o, "X=%-10.4f Y=%-10.4f", x, y);
                                    TextOut(hdc,10,18,str_o,strlen(str_o));

        }
                            z=(float)fobj->EvaluaZ((double)x,(double)y);
                            if(fobj->error!=NOERROR)
                                    continue;
                            transforma();
```

```
                        esconde();
                    }
                }
}
    if (theta>=225.0 && theta<315.0)
    {
        for( y=ydesde; y<=yhasta; y+=ydelta)
        {
            fl=0;
            for( x=xdesde; x<=xhasta; x+=xdelta)
            {
                if(MARGENX==0)
    {
                    sprintf(str_o, "X=%-10.4f Y=%-10.4f", x, y);
                    TextOut(hdc,10,18,str_o,strlen(str_o));
    }
                z=(float)fobj->EvaluaZ((double)x,(double)y);
                if(fobj->error!=NOERROR)
                    continue;
                transforma();
                esconde();
            }
        }
}
    DeleteObject(hPen);
    hPen=CreatePen(PS_SOLID,1,ROJO);

    SelectObject(hdc,hPen);

    SetTextColor(hdc,NEGRO);

    x=0;z=0;fl=0;
    for( y=ydesde;y<=yhasta;y+=ydelta)
    {
        transforma();
        if( sx<0 || sx>mx || sy<0 || sy>my )
{
            fl=0;
            break;
        }
        if( fl==0 )
        {
            fl=1;
            MoveToEx(hdc,sx*SCALEX+MARGENX,sy*SCALEY+MARGENY,NULL);
        }
        LineTo(hdc, sx*SCALEX+MARGENX,sy*SCALEY+MARGENY);
    }
    TextOut(hdc,sx*SCALEX+MARGENX,sy*SCALEY+MARGENY,"y",1);

    y=0;z=0;fl=0;
    for( x=xdesde;x<=xhasta;x+=xdelta)
    {
        transforma();
        if( sx<0 || sx>mx || sy<0 || sy>my )
```

```
            {
                    fl=0;
                    break;
            }
            if( fl==0 )
            {
                    fl=1;
                    MoveToEx(hdc,sx*SCALEX+MARGENX,sy*SCALEY+MARGENY,NULL);
            }
            LineTo(hdc,sx*SCALEX+MARGENX,sy*SCALEY+MARGENY);
        }
        TextOut(hdc,sx*SCALEX+MARGENX,sy*SCALEY+MARGENY,"x",1);
        y=0; x=0; fl=0;
        for( z=zdesde; z<=zhasta; z++)
        {
                transforma();
                if( sx<0 || sx>mx || sy<0 || sy>my )
                {
                        fl=0;
                        break;
                }
                if( fl==0 )
                {
                        fl=1;
                        MoveToEx(hdc,sx*SCALEX+MARGENX,sy*SCALEY+MARGENY,NULL);
                }
                LineTo(hdc, sx*SCALEX+MARGENX,sy*SCALEY+MARGENY);
        }
        TextOut(hdc,sx*SCALEX+MARGENX,sy*SCALEY+MARGENY,"z",1);

        DeleteObject(hPen);
}

float g3d::redondea (float theta)
{
        if(theta>360.0)
                for(;theta>360.0;theta-=360.0);
        else if(theta<0.0)
        for(;theta<0.0;theta+=360.0);
        return theta;
 }
```

```
# include <windows.h>
# include <string.h>
# include "printer.hpp"

# define and &&
# define or ||


extern int xClientView,yClientView;

typedef VOID (FAR PASCAL * DEVMODEPROC)(HWND,HANDLE,LPSTR,LPSTR);

extern char far *ProgramName;

TPrint::TPrint(void)
{
        hPr=(HDC)NULL;
        Job=FALSE;
  Error=FALSE;
        Page=FALSE;

        PageSize.x=0;
        PageSize.y=0;

        SCALEX=1;
        SCALEY=1;

        MARGENX=10;
        MARGENY=10;

        char Printer[MAXPRINTER+1];
   TEXTMETRIC tm;

        // Obteniendo el HDC de la impresora

        GetProfileString("windows","device",",,,",Printer,MAXPRINTER);
        strcpy(PrintType,strtok(Printer,","));
        strcpy(PrintDriver,strtok(NULL,","));
        strcpy(PrintPort,strtok(NULL,","));
        if(stricmp(PrintType,"")!=0 and stricmp(PrintDriver,"")!=0 and  stricmp(PrintPort,"")!=0)
                hPr=CreateDC(PrintDriver,PrintType,PrintPort,NULL);
        else
        {
                ReleaseDC();
    hPr=(HDC)NULL;
                Error=-1;
    return;
        }
        GetPageSize(); // Obtiene el tamaño de la página
```

```
        GetTextMetrics(hPr,&tm);
        LineSpace=tm.tmHeight+tm.tmExternalLeading;

        LinesPerPage=(PageSize.y/LineSpace) - 6;
        if(LinesPerPage<0)
                LinesPerPage=0;

        if(xClientView>0)
                SCALEX=(PageSize.x-2*MARGENX)/xClientView;
        else
                SCALEX=(PageSize.x-2*MARGENX)/500;

/*      if(yClientView>0)
                SCALEY=(PageSize.y-2*MARGENY)/yClientView;
        else
                SCALEY=(PageSize.y-2*MARGENY)/500;*/

        SCALEY=SCALEX;

        if(SCALEX==0.0)
        SCALEX=0.1;
        if(SCALEY==0.0)
                SCALEY=0.1;

        return;
}

TPrint::~TPrint(void)
{
        if(Page)
        {
   EndPage();
                EndDoc();
                ReleaseDC();
  }
        if(Job)
        {
                EndDoc();
                ReleaseDC();
        }
        if(hPr)
        ReleaseDC();
        return;
}

void TPrint::StartDoc(void)
{
        if(hPr==(HDC)NULL)
                return;
        if(Page)
        {
                EndPage();
                EndDoc();
```

```
}
        if(Job)
        EndDoc();
        Error=Escape(hPr,STARTDOC,strlen(ProgramName),ProgramName,NULL);
        if(Error>0)
                Job=TRUE;
        return;
}

void TPrint::StartPage(void)
{
        if(!hPr or !Job)
        return;
        if(Page)
                EndPage();
        Page=TRUE;
        return;
}

void TPrint::EndPage(void)
{
        if(!Job or !hPr)
        return;
        Error=Escape(hPr,NEWFRAME,NULL,NULL,NULL);
        Page=FALSE;
}

void TPrint::EndDoc(void)
{
        if(hPr==(HDC)NULL or !Job)
                return;
        Error=Escape(hPr,ENDDOC,NULL,NULL,NULL);
        Page=FALSE;
        Job=FALSE;
  return;
}

void TPrint::AbortDoc(void)
{
        if(hPr==(HDC)NULL or !Job)
                return;
        Error=::AbortDoc(hPr);
        Page=FALSE;
        Job=FALSE;
  return;
}

void TPrint::GetPageSize(void)
{
        if(!hPr)
        return;
        Escape(hPr,GETPHYSPAGESIZE,NULL,NULL,(LPSTR)&PageSize);
        return;
}
```

```
void TPrint::PrinterSetup(HWND hwnd)
{
        DEVMODEPROC lp;
  HANDLE hlib;
        char DriverFile[17];
        strcat(strcpy(DriverFile,PrintDriver),".DRV");
        hlib=LoadLibrary(DriverFile);
        if(hlib<32)
        {
                Error=-1;
                return;
        }
        (FARPROC)lp=GetProcAddress(hlib,"DEVICEMODE");
        if(lp==NULL)
        return;
        (*lp)(hwnd,hlib,(LPSTR)PrintType,(LPSTR)PrintPort);   // Llama a la funcion de Setup
   FreeLibrary(hlib);
}

 void TPrint::ReleaseDC(void)
 {
          if(!hPr)
                  return;
          if(Page)
          {
                   EndPage();
                   EndDoc();
          }
          if(Job)
                   EndDoc();
          DeleteDC(hPr);
          hPr=(HDC)NULL;
          return;
  }
```

FORMULA.CPP

```
# include "formula.hpp"

char *Funciones[]={"exp","sen","cos","tan","asen","acos","atan","log","ln"};

/********************* Funciones de la clase formula ****************/

/******************* Constructora default ***********************/

formula::formula()
{
        raiz=(NODO *)NULL;
        error=NOFORMULA;
        Ok=FALSE;
        return;
}

/****************** Constructora 1 ***********************/

formula::formula(char *form)
{
        raiz = CrearNodo();
        if(raiz==NULL)
        {
                Ok=FALSE;
                error=NOMEMORIA;
                return;
        }
        strcpy(expresion,form);
        if(IsExpresion(&(raiz->hizq),0,strlen(form)-1))
                Ok=TRUE;
        else
  {
                Ok=FALSE;
                FreeTree(&(raiz->hizq));
        }
        return;
}

/******************** Crea el arbol de evaluacion *************/

void formula::Create(char *form)
{
        if(raiz!=NULL)
                FreeTree(&raiz->hizq);
        raiz = CrearNodo();
  if(raiz==NULL)
        {
                Ok=FALSE;
                error=NOMEMORIA;
                return;
```

```
        }
        if(!stricmp(form,""))
        {
                Ok=FALSE;
                return;
}
        strcpy(expresion,form);
        if(IsExpresion(&(raiz->hizq),0,strlen(form)-1))
                Ok=TRUE;
        else
{
                Ok=FALSE;
                FreeTree(&(raiz->hizq));
        }
        return;
}

/******************** Destructora **********************/

formula::~formula()
{
        FreeTree(&raiz);
        return;
}

/******************** Operador Aditivo **********************/

int formula::OpAditivo(int ini,int fin)
{
        int i,par=0;
        if(ini>fin)
                return FALSE;
        for(i=ini;i<=fin;i++)
                if(expresion[i]=='(')
                        par--;
                else if(expresion[i]==')')
                        par++;
                else if(expresion[i]==OP_ADICION or expresion[i]==OP_SUBSTRACCION)
                        if(par==0)
                                return i;
        return ERRORNUM;
}

/******************** Operador Multiplicativo **********************/

int formula::OpMultiplicativo(int ini,int fin)
{
        int i,par=0;
        if(ini>fin)
                return FALSE;
        for(i=ini;i<=fin;i++)
                if(expresion[i]=='(')
                        par--;
                else if(expresion[i]==')')
```

```
                par++;
            else if(expresion[i]==OP_MULTIPLICACION or expresion[i]==OP_DIVISION
                or expresion[i]==OP_POTENCIACION)
                    if(par==0)
                            return i;
        return ERRORNUM;
}

/****************** Verifica si es una expresion ************************/

BOOL formula::IsExpresion(NODO **nodo,int ini,int fin)
{
        int k;
        NODO *hijo;
        if(ini>fin)
                return FALSE;
        k=OpAditivo(ini,fin);
        if(k==ERRORNUM)
        {
                if(IsTermino(nodo,ini,fin))
                        return TRUE;
                else
                {
                        FreeTree(nodo);
                        return FALSE;
                }
        }
        else
        {
                hijo=CrearNodo();
                if(hijo==(NODO*)NULL)
                {
                        error=NOMEMORIA;
                        return FALSE;
                }
                hijo->op.operador=expresion[k];
                hijo->tipo=OPERADOR;
                InsNodo(nodo,hijo);
                if(IsTermino(&(hijo->hizq),ini,k-1))
                        if(IsExpresion(&(hijo->hder),k+1,fin))
                                return TRUE;
                        else
                                FreeTree(nodo);
                else
                        FreeTree(nodo);
                return FALSE;
        }
}

/********************** Verifica si es un termino ********************/

BOOL formula::IsTermino(NODO **nodo,int ini,int fin)
{
        int k;
```

```
    NODO *hijo;
    if(ini>fin)
            return FALSE;
    k=OpMultiplicativo(ini,fin);
    if(k==ERRORNUM)
    {
            if(IsFactor(nodo,ini,fin))
                    return TRUE;
            else
                    FreeTree(nodo);
    }
    else
    {
            hijo=CrearNodo();
            if(hijo==(NODO*)NULL)
            {
                    error=NOMEMORIA;
                    return FALSE;
            }
            hijo->op.operador=expresion[k];
            hijo->tipo=OPERADOR;
            InsNodo(nodo,hijo);

            if(IsFactor(&(hijo->hizq),ini,k-1))
            {
                    if(IsTermino(&(hijo->hder),k+1,fin))
                            return TRUE;
                    else
                    {
                            FreeTree(nodo);
                            return FALSE;
                    }
            }
            else
            {
                    FreeTree(nodo);
                    return FALSE;
            }
    }
    FreeTree(nodo);
    return FALSE;
}

/***************** Verifica si es factor ***************************/

BOOL formula::IsFactor(NODO **nodo,int ini,int fin)
{
        if(ini>fin)
                return FALSE;
        else if(expresion[ini]=='(' and expresion[fin]==')')
                return IsExpresion(nodo,ini+1,fin-1);
        else if(IsFuncion(nodo,ini,fin))
                return TRUE;
        else if(IsConstante(nodo,ini,fin))
```

```
                return TRUE;
        else
                return IsVariable(nodo,ini,fin);
}

/****************** Verifica si es funcion ************************/

BOOL formula::IsFuncion(NODO **nodo,int ini,int fin)
{
        int i;
        FUNCIONES Index;
        NODO *hijo;
        char funcion[MAXFORMULA+1];
        if(ini>fin)
        {
                FreeTree(nodo);
                return FALSE;
        }
        if(expresion[fin]!=')')
        {
                FreeTree(nodo);
                return FALSE;
        }
        for(i=ini;i<=fin;i++)
        {
                strncpy(funcion,expresion+ini,i-ini+1);
                funcion[i-ini+1]=0;
                Index=IsNombreFuncion(funcion);
                if(Index>ERRORFUNCION)
                {
                        hijo=CrearNodo();
                        if(hijo==(NODO*)NULL)
                        {
                                error=NOMEMORIA;
                                FreeTree(nodo);
                                return FALSE;
                        }
                        strcpy(hijo->op.funcion,Funciones[Index]);
                        hijo->tipo=FUNCION;
                        InsNodo(nodo,hijo);

                        if(i<fin and expresion[i+1]=='(')
                                if(IsParam(&(hijo->hizq),Index,i+2,fin-1))
                                        return TRUE;

                }
        }
        FreeTree(nodo);
        return FALSE;
}

/******* Verifica si los parametros de una funcion estan  correctos *****/

BOOL formula::IsParam(NODO **nodo,FUNCIONES tipo,int ini,int fin)
{
```

```
        if(ini>fin)
                return FALSE;
        switch(tipo)
        {
                case EXP:
                case SEN:
                case COS:
                case TAN:
                case ASEN:
                case ACOS:
                case ATAN:
                case LOG:
                case LN:
                        return IsExpresion(nodo,ini,fin);
                default:
                        FreeTree(nodo);
                        return FALSE;
        }
}

/****************** Verifica si es una constante **********************/

BOOL formula::IsConstante(NODO **nodo,int ini,int fin)
{
        char num[MAXFORMULA+1];
        int i,digitos_enteros=0,digitos_decimales=0;
        BOOL punto=FALSE,FlagE=FALSE;
        NODO *hijo;
        if(ini>fin)
        {
                FreeTree(nodo);
                return FALSE;
        }
        for(i=ini;i<=fin;i++)
        {
                if(expresion[i]==MARCAPUNTO)
                {
                        if(punto)
                                return FALSE;
                        else
                                punto=TRUE;
                }
                else if(expresion[i]=='E' or expresion[i]=='e')
                {
                        if(FlagE)
                                return FALSE;
                        else
                                FlagE=TRUE;
                }
                else if(!isdigit(expresion[i]))
                {
                        FreeTree(nodo);
                        return FALSE;
                }
```

```
                else if (punto)
                        digitos_decimales++;
                else
                        digitos_enteros++;
        }
        if(digitos_enteros>MAXNUMDIGITOS or digitos_decimales>MAXNUMDECIMALES)
        {
                FreeTree(nodo);
                return FALSE;
        }
        hijo=CrearNodo();
        if(hijo==(NODO*)NULL)
        {
                error=NOMEMORIA;
                return FALSE;
        }
        strncpy(num,expresion+ini,fin-ini+1);
        hijo->op.constante=atof(num);
        hijo->tipo=CONSTANTE;
        InsNodo(nodo,hijo);
        return TRUE;
}

/********************* Verifica si es variable ************************/

BOOL formula::IsVariable(NODO **nodo,int ini,int fin)
{
        NODO *hijo;
        if(ini>fin or ini!=fin)
        {
                FreeTree(nodo);
                return FALSE;
        }
        if(expresion[ini]=='X' or expresion[ini]=='x')
        {
                hijo=CrearNodo();
                if(hijo==(NODO*)NULL)
                {
                        error=NOMEMORIA;
                        return FALSE;
                }
                hijo->op.variable=VARX;
                hijo->tipo=VARIABLE;
                InsNodo(nodo,hijo);
                return TRUE;
        }
        else if(expresion[ini]=='Y' or expresion[ini]=='y')
        {
                hijo=CrearNodo();
                if(hijo==(NODO*)NULL)
                {
                        error=NOMEMORIA;
                        return FALSE;
                }
```

```
                hijo->op.variable=VARY;
                hijo->tipo=VARIABLE;
                InsNodo(nodo,hijo);
                return TRUE;
        }
        FreeTree(nodo);
        return FALSE;
}

/*************** Verifica si es el nombre de una funcion ****************/

FUNCIONES formula::IsNombreFuncion(char * funcion)
{
        int i;
        for(i=0;i<NUMFUNCIONES;i++)
                if(stricmp(Funciones[i],funcion)==0)
                        return ((FUNCIONES)i);
        return ERRORFUNCION;
}

/************************ Crea un nodo del arbol ***********************/

NODO * formula::CrearNodo(void)
{
        NODO *nodo;
        nodo=(NODO *)malloc(sizeof(NODO));
        if(nodo==(NODO*)NULL)
        {
                error=NOMEMORIA;
                return NULL;
        }
        nodo->hizq=NULL;
        nodo->hder=NULL;
        return nodo;
}

/************************ Inserta un nodo en el arbol ***************/

void formula::InsNodo(NODO **padre,NODO *nodo)
{
        *padre=nodo;
        return;
}

/******************* Liberar Nodo *************************/

void formula::FreeTree(NODO **nodo)
{
        if(*nodo==(NODO*)NULL)
                return;
        FreeTree(&((*nodo)->hizq));
        FreeTree(&((*nodo)->hder));
        free( *nodo);
        *nodo=(NODO *)NULL;
```

```
        return;
}

/*int matherr(struct exception *e)
{
        e->retval=0.0;
  return 1;
}*/

/****************** Evalua el arbol ************************/

double formula::EvaluaZ(double x, double y)
{
        double rrr;
        rrr=Evaluar(raiz->hizq, x,y);
  return rrr;
}

double formula::Evaluar(NODO *nodo,double x,double y)
{
        double res,base,exponente;
  error=NOERROR;
        if(nodo==NULL)
        return 0.0;
  switch(nodo->tipo)
  {
        case OPERADOR:
        switch(nodo->op.operador)
      {
        case OP_ADICION:
        return Evaluar(nodo->hizq,x,y)+Evaluar(nodo->hder,x,y);
        case OP_SUBSTRACCION:
        return Evaluar(nodo->hizq,x,y)-Evaluar(nodo->hder,x,y);
        case OP_MULTIPLICACION:
        return Evaluar(nodo->hizq,x,y)*Evaluar(nodo->hder,x,y);
        case OP_DIVISION:
        res=Evaluar(nodo->hder,x,y);
         if(res==0.0)
         {
                error=DIVXCERO;
                return 0.0;
         }
         else                                    return Evaluar(nodo->hizq,x,y)/res;

        case OP_POTENCIACION:
          base=Evaluar(nodo->hizq,x,y);
          exponente=Evaluar(nodo->hder,x,y);
          return pow(base,exponente);
        default:
          return 0;
      }
      case FUNCION:
        if(stricmp(nodo->op.funcion,"sen")==0)
          return sin(Evaluar(nodo->hizq,x,y));
```

```
else if(stricmp(nodo->op.funcion,"exp")==0)
    return exp(Evaluar(nodo->hizq,x,y));
else if(stricmp(nodo->op.funcion,"cos")==0)
    return cos(Evaluar(nodo->hizq,x,y));
else if(stricmp(nodo->op.funcion,"tan")==0)
    return tan(Evaluar(nodo->hizq,x,y));
else if(stricmp(nodo->op.funcion,"asen")==0)
{
    res=Evaluar(nodo->hizq,x,y);
  if(res<-1.0 or res>1.0)
  {
    error=DOMINIO_ASEN;
     return 0.0;
  }
  else
                        return asin(res);
}
    else if(stricmp(nodo->op.funcion,"acos")==0)
{
  res=Evaluar(nodo->hizq,x,y);
  if(res<-1.0 or res>1.0)
  {
    error=DOMINIO_ACOS;
     return 0.0;
  }
  else
                                        return acos(res);
}
else if(stricmp(nodo->op.funcion,"atan")==0)
    return atan(Evaluar(nodo->hizq,x,y));
else if(stricmp(nodo->op.funcion,"log")==0)
{
                                    res=Evaluar(nodo->hizq,x,y);
   if(res>0.0)
              return log10(res);
   else
   {
     error=DOMINIO_LOG;
     return 0.0;
   }
}
else if(stricmp(nodo->op.funcion,"ln")==0)
{
     res=Evaluar(nodo->hizq,x,y);
   if(res>0.0)
              return log(res);
   else
   {
     error=DOMINIO_LN;
     return 0.0;
   }
}
     else
{
```

```
        error=FUNCION_DESCONOCIDA;
        return 0.0;
      }
    case CONSTANTE:
        return nodo->op.constante;
    case VARIABLE:
        if(nodo->op.variable==VARX)
        return x;
      else
        return y;
    default:
        error=TIPO_DESCONOCIDO;
        return 0.0;
  }
}
```

Archivos de Cabecera

A continuación se procederá a mostrar los listados de los headers de los módulos anteriormente descritos.

GRAFICO.HPP

```
class g3d
{
        float           d,s1,c1,s2,c2,x,y,z;
        float           theta, phi, rho;
        float           sx,sy,ox,oy, ex, ey, ez;
        short           fl,f,mx,my, cx, cy;
  float         MARGENX,MARGENY,SCALEX,SCALEY;
        float           yn[1024],yx[1024];
        void            transforma();
        void            esconde();
        void            evalua();
        float     redondea (float);

        public:

        HDC             hdc;
        void            init_g3d( short tx,short ty,
                                        short centx,short centy,
                                        float dist,float angx,float angz,
                                        float vect,
                                        float,float,float,
                                        float,float,float,float);
        void            grafica(formula *, float xdesde, float xhasta, float xdelta,
                                        float ydesde, float yhasta, float ydelta,
                                        float zdesde, float zhasta);

};
```

PRINTER.HPP

```
# define MAXPRINTER 80

class TPrint
{
        BOOL Job;
        BOOL Page;
        int Error;
        POINT PageSize;
        char far
 PrintDriver[MAXPRINTER+1],PrintPort[MAXPRINTER+1],PrintType[MAXPRINTER+1];
```

```
        void GetPageSize(void);
void ReleaseDC(void);

public:
                HDC hPr;
                int LineSpace,LinesPerPage;
                float SCALEX,SCALEY,MARGENX,MARGENY;

                TPrint(void);
                ~TPrint(void);
                virtual void AbortDoc(void);
                virtual void StartDoc(void);
                virtual void StartPage(void);
                virtual void EndPage(void);
                virtual void EndDoc(void);
                virtual void PrinterSetup(HWND);
   virtual HDC GetDC(void) { return hPr;};
};
```

FORMULA.HPP

```
# include <windows.h>
# include <string.h>
# include <math.h>
# include <ctype.h>
# include <stdlib.h>

# define or ||
# define and &&
# define MAXFORMULA 80    // Longitud maxima del nombre de una funcion
# define NUMFUNCIONES 9   // Numero de funciones consideradas
# define MAXNUMDIGITOS 15  // Maximo numero de digitos enteros soportados
# define MAXNUMDECIMALES 6 // Max numero de digitos decimales soportados
# define ERRORNUM -1
# define MARCAPUNTO '.'


 enum VAR {VARX,VARY};
 enum OP_TIPO {OPERADOR,FUNCION,CONSTANTE,VARIABLE};
 enum FUNCIONES {ERRORFUNCION=-1,EXP=0,SEN,COS,TAN,ASEN,ACOS,ATAN,LOG,LN};
 enum OPE_MAT
 {OP_ADICION=43,OP_SUBSTRACCION=45,OP_MULTIPLICACION=42,OP_DIVISION=47,OP_PO
 TENCIACION=94};
 enum ERROR_TIPO
 {NOERROR,OVER_ENTEROS,OVER_DECIMALES,NOVARX,NOVARY,NOCONST,NOFORMULA,
        DOMINIO_ASEN,DOMINIO_ACOS,DOMINIO_LOG,DOMINIO_LN,FUNCION_DESCONO
 CIDA,
          TIPO_DESCONOCIDO,DIVXCERO,NOMEMORIA,INVALIDA};
```

```
union OPERATOR
{
        char operador;
        char funcion[MAXFORMULA+1];
        double constante;
        VAR variable;
};

struct NODO
{
        OPERATOR op;
        OP_TIPO tipo;
        NODO *hizq,*hder;
};

class formula
{
        NODO *raiz;

        public:

        char expresion[MAXFORMULA+1];
        BOOL Ok;
        ERROR_TIPO error;
        formula(void);          // Constructor default
        formula(char *string);  // Constructor 1
        ~formula();             // Destructor

        int OpAditivo(int,int); // Funciones para Validacion de la expresion
        int OpMultiplicativo(int,int);
        BOOL IsExpresion(NODO **,int,int);
        BOOL IsTermino(NODO **,int,int);
        BOOL IsFactor(NODO **,int,int);
        BOOL IsFuncion(NODO **,int,int);
        BOOL IsParam(NODO **,FUNCIONES,int,int);
        BOOL IsConstante(NODO **,int,int);
        BOOL IsVariable(NODO **,int,int);
        FUNCIONES IsNombreFuncion(char *);

        BOOL GetStatus(void){return Ok;}// Funciones que trabajan sobre el arbol
        NODO *CrearNodo(void);
        void InsNodo(NODO** padre, NODO *nodo);
        void FreeTree(NODO **nodo);
        void Create(char *);

  double Evaluar(NODO *,double,double); // Funciones que evaluan el arbol
//      void Graficar(HWND);
  double EvaluaZ(double,double);
};
```

Archivo de Recursos

A continuación se muestra el archivo de recursos de GrafWin :

GRAFWIN.RC

```
#include "graf3d.h"

ABOUTDIABOX DIALOG 47, 17, 180, 121
STYLE WS_POPUP | WS_CAPTION | WS_SYSMENU | WS_THICKFRAME
CLASS "bordlg"
CAPTION "Acerca De"
FONT 8, "MS Sans Serif"
BEGIN
        CONTROL "Grafico de Funciones", -1, "STATIC", SS_CENTER | WS_CHILD | WS_VISIBLE,
17, 4, 145, 10
        CONTROL "Button", IDOK, "BorBtn", BS_PUSHBUTTON | WS_CHILD | WS_VISIBLE |
WS_TABSTOP, 74, 91, 32, 20
        CTEXT "Carlos Mestanza Y.", -1, 49, 32, 82, 8, WS_CHILD | WS_VISIBLE | WS_GROUP
        CTEXT "Edison Barahona M.", -1, 49, 40, 82, 8, WS_CHILD | WS_VISIBLE | WS_GROUP
        CTEXT "Segundo García H.", -1, 49, 47, 82, 8, WS_CHILD | WS_VISIBLE | WS_GROUP
        CTEXT "Tópico de Graduación II. Año 1994", -1, 25, 14, 129, 7, WS_CHILD | WS_VISIBLE |
WS_GROUP
        LTEXT "Profesor : Ing. Sixto García", -1, 45, 63, 90, 8, WS_CHILD | WS_VISIBLE |
WS_GROUP
        LTEXT "ESPOL", -1, 78, 75, 24, 11, WS_CHILD | WS_VISIBLE | WS_GROUP
END


GrafMenu MENU
BEGIN
        POPUP "&Archivos"
        BEGIN
                MENUITEM "&Nuevo ...", CM_NUEVO
                MENUITEM "&Abrir  ...", CM_ABRIR
                MENUITEM "&Grabar ...", CM_GRABAR
                MENUITEM "Grabar &Como...", CM_GRABAR_COMO
                MENUITEM "&Imprimir", CM_GRABAR_BMP
                MENUITEM SEPARATOR
                MENUITEM "&Salir", CM_SALIR
        END

        POPUP "&Grafico"
        BEGIN
                MENUITEM "&Formula ...", CM_FORMULA
                MENUITEM "Para&metros ...", CM_PARAMETROS
        END

        MENUITEM "&Defaults", CM_RANGOS
```

```
        POPUP "A&yuda", HELP
        BEGIN
                MENUITEM "Contenid&o", CM_HELP_CONTENIDO
                MENUITEM "Acerca de GRAFWIN ..", CM_ACERCA_DE
        END

END


GRAFDIABOX DIALOG 5, 17, 218, 183
STYLE DS_MODALFRAME | DS_NOIDLEMSG | WS_OVERLAPPED | WS_VISIBLE |
WS_CAPTION | WS_SYSMENU | WS_THICKFRAME
CLASS "Bordlg"
CAPTION "Ingreso de Parametros "
FONT 9, "Arial"
BEGIN
        LTEXT "Descripción :", -1, 6, 6, 46, 8, WS_CHILD | WS_VISIBLE | WS_GROUP
        CONTROL "", CM_DESCRIPCION, "EDIT", ES_LEFT | ES_AUTOHSCROLL | WS_CHILD |
WS_VISIBLE | WS_BORDER | WS_TABSTOP, 60, 4, 154, 12
        CONTROL "Dominio", 102, "button", BS_GROUPBOX | BS_LEFTTEXT | WS_CHILD |
WS_VISIBLE | WS_GROUP, 3, 23, 120, 60
        LTEXT "Rango X :", -1, 9, 36, 36, 10, WS_CHILD | WS_VISIBLE | WS_GROUP
        CONTROL "", CM_RANGO_X1, "EDIT", ES_LEFT | ES_NOHIDESEL | WS_CHILD |
WS_VISIBLE | WS_BORDER | WS_GROUP | WS_TABSTOP, 52, 35, 28, 12
        EDITTEXT CM_RANGO_X2, 88, 35, 28, 12, ES_LEFT | WS_CHILD | WS_VISIBLE |
WS_BORDER | WS_TABSTOP
        LTEXT "Rango Y :", -1, 9, 51, 35, 11, WS_CHILD | WS_VISIBLE | WS_GROUP
        CONTROL "", CM_RANGO_Y1, "EDIT", ES_LEFT | ES_AUTOHSCROLL | ES_NOHIDESEL
| WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP, 52, 50, 28, 12
        CONTROL "", CM_RANGO_Y2, "EDIT", ES_LEFT | ES_NOHIDESEL | WS_CHILD |
WS_VISIBLE | WS_BORDER | WS_TABSTOP, 88, 50, 28, 13
        LTEXT "Rango Z :", -1, 9, 67, 35, 11, WS_CHILD | WS_VISIBLE | WS_GROUP
        EDITTEXT CM_RANGO_Z1, 52, 66, 28, 13, ES_LEFT | WS_CHILD | WS_VISIBLE |
WS_BORDER | WS_TABSTOP
        EDITTEXT CM_RANGO_Z2, 88, 66, 28, 13, ES_LEFT | WS_CHILD | WS_VISIBLE |
WS_BORDER | WS_TABSTOP
        LTEXT "Escala X :", -1, 134, 36, 36, 9, WS_CHILD | WS_VISIBLE | WS_GROUP
        EDITTEXT CM_ESCALA_X, 175, 35, 29, 12, ES_LEFT | WS_CHILD | WS_VISIBLE |
WS_BORDER | WS_TABSTOP
        LTEXT "Escala Y :", -1, 133, 52, 36, 9, WS_CHILD | WS_VISIBLE | WS_GROUP
        EDITTEXT CM_ESCALA_Y, 175, 50, 29, 12, ES_LEFT | WS_CHILD | WS_VISIBLE |
WS_BORDER | WS_TABSTOP
        LTEXT "Escala Z :", -1, 133, 67, 36, 10, WS_CHILD | WS_VISIBLE | WS_GROUP
        EDITTEXT CM_ESCALA_Z, 175, 65, 29, 12, ES_LEFT | WS_CHILD | WS_VISIBLE |
WS_BORDER | WS_TABSTOP
        LTEXT "Delta  X :", -1, 7, 113, 34, 10, WS_CHILD | WS_VISIBLE | WS_GROUP
        CONTROL "", CM_DOMINIO_X, "EDIT", ES_LEFT | ES_AUTOHSCROLL | WS_CHILD |
WS_VISIBLE | WS_BORDER | WS_TABSTOP, 51, 112, 30, 12
        LTEXT "Delta Y :", -1, 7, 129, 34, 12, WS_CHILD | WS_VISIBLE | WS_GROUP
        EDITTEXT CM_DOMINIO_Y, 51, 129, 30, 12, ES_LEFT | WS_CHILD | WS_VISIBLE |
WS_BORDER | WS_TABSTOP
        CONTROL "Perspectiva", 107, "button", BS_GROUPBOX | WS_CHILD | WS_VISIBLE, 128,
97, 83, 59
```

```
        EDITTEXT CM_PUNTVIST_X, 175, 108, 29, 12, ES_LEFT | WS_CHILD | WS_VISIBLE |
WS_BORDER | WS_TABSTOP
        EDITTEXT CM_PUNTVIST_Y, 175, 123, 29, 12, ES_LEFT | WS_CHILD | WS_VISIBLE |
WS_BORDER | WS_TABSTOP
        EDITTEXT CM_PUNTVIST_Z, 175, 139, 29, 12, ES_LEFT | WS_CHILD | WS_VISIBLE |
WS_BORDER | WS_TABSTOP
        LTEXT "Angulo Z :", -1, 134, 125, 36, 9, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "Distancia :", -1, 134, 141, 36, 10, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "Angulo X :", -1, 133, 110, 36, 9, WS_CHILD | WS_VISIBLE | WS_GROUP
        CONTROL "OK", IDOK, "BorBtn", BS_DEFPUSHBUTTON | WS_CHILD | WS_VISIBLE |
WS_TABSTOP, 10, 161, 35, 19
        CONTROL "&DEFAULT", CM_DEFAULT, "BorBtn", BS_PUSHBUTTON | WS_CHILD |
WS_VISIBLE | WS_TABSTOP, 62, 161, 35, 19
        CONTROL "", 2, "BorBtn", BS_PUSHBUTTON | WS_CHILD | WS_VISIBLE | WS_TABSTOP,
113, 161, 36, 19
        CONTROL "", 998, "BorBtn", BS_PUSHBUTTON | WS_CHILD | WS_VISIBLE |
WS_TABSTOP, 166, 161, 36, 19
        CONTROL "", 103, "BorShade", 2 | WS_CHILD | WS_VISIBLE, 5, 89, 205, 1
        CONTROL "Pasos", 105, "button", BS_GROUPBOX | WS_CHILD | WS_VISIBLE, 3, 99, 86, 49
        CONTROL "Escalas", 107, "button", BS_GROUPBOX | WS_CHILD | WS_VISIBLE, 130, 23,
83, 60
END

Grabar DIALOG 20, 20, 220, 146
STYLE DS_MODALFRAME | WS_POPUP | WS_DLGFRAME | WS_SYSMENU
CLASS "BorDlg"
FONT 8, "Helv"
BEGIN
        CONTROL "", -1, "borshade", 3 | WS_CHILD | WS_VISIBLE | WS_GROUP, 168, 0, 2, 146
        CONTROL "", -1, "borshade", 1 | WS_CHILD | WS_VISIBLE | WS_GROUP, 8, 8, 152, 20
        LTEXT "Nombre del &archivo", -1, 12, 14, 64, 9, WS_CHILD | WS_VISIBLE | WS_GROUP
        CONTROL "", 32, "COMBOBOX", CBS_DROPDOWN | CBS_AUTOHSCROLL |
CBS_OEMCONVERT | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_VSCROLL | WS_GROUP |
WS_TABSTOP, 79, 12, 73, 12
        LTEXT " Ruta:", -1, 8, 36, 22, 8, WS_CHILD | WS_VISIBLE | WS_GROUP
        LTEXT "", 34, 30, 36, 130, 8
        LTEXT " &Archivos", -1, 8, 51, 72, 9, WS_CHILD | WS_VISIBLE | WS_GROUP
        CONTROL "", -1, "borshade", 1 | WS_CHILD | WS_VISIBLE | WS_GROUP, 8, 60, 72, 78
        CONTROL "", 33, "LISTBOX", LBS_STANDARD | WS_CHILD | WS_VISIBLE | WS_GROUP
| WS_TABSTOP, 12, 66, 64, 70
        LTEXT " &Directorios", -1, 88, 51, 72, 9, WS_CHILD | WS_VISIBLE | WS_GROUP
        CONTROL "", -1, "borshade", 1 | WS_CHILD | WS_VISIBLE | WS_GROUP, 88, 60, 72, 78
        CONTROL "", 35, "LISTBOX", LBS_STANDARD | WS_CHILD | WS_VISIBLE | WS_GROUP
| WS_TABSTOP, 92, 66, 64, 70
        CONTROL "", 1, "borbtn", BS_DEFPUSHBUTTON | BBS_PARENTNOTIFY | WS_CHILD |
WS_VISIBLE | WS_GROUP | WS_TABSTOP, 176, 18, 37, 25
        CONTROL "", 2, "borbtn", 8192 | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 176, 60, 37, 25
        CONTROL "", 15, "borbtn", 8192 | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 176, 102, 37,
25
END
DlgFormula DIALOG 18, 18, 211, 55
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CLASS "bordlg"
CAPTION "Ingreso de Formula"
```

```
FONT 8, "MS Sans Serif"
BEGIN
        CONTROL "", IDD_FORMULA, "EDIT", ES_LEFT | ES_AUTOHSCROLL | ES_UPPERCASE
| WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP, 59, 5, 144, 12
        LTEXT "Formula :   Z = ", -1, 7, 7, 52, 9, WS_CHILD | WS_VISIBLE | WS_GROUP
        CONTROL "Button", IDOK, "BorBtn", BS_DEFPUSHBUTTON | WS_CHILD | WS_VISIBLE |
WS_TABSTOP, 38, 26, 36, 24
        CONTROL "Button", IDCANCEL, "BorBtn", BS_PUSHBUTTON | WS_CHILD | WS_VISIBLE
| WS_TABSTOP, 85, 26, 36, 24
        CONTROL "Button", 998, "BorBtn", BS_PUSHBUTTON | WS_CHILD | WS_VISIBLE |
WS_TABSTOP, 136, 25, 36, 24
END

PORTADA BITMAP "portada.bmp"

INICIO DIALOG 18, 18, 199, 178
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Graficador de Funciones"
BEGIN
        CONTROL "Button", 501, "BorBtn", BBS_BITMAP | WS_CHILD | WS_VISIBLE, 6, 4, 30, 18
        CONTROL "Button", 1, "BorBtn", BS_PUSHBUTTON | WS_CHILD | WS_VISIBLE |
WS_TABSTOP, 30, 150, 32, 20
END


GRAFWIN ICON "grafwin.ico"
```

Archivos de Ayuda

Se muestran posteriormente, todos los archivos usados para la construcción de la ayuda de GrafWin GRAFWIN.HLP.

GRAFWIN.HPJ

```
[OPTIONS]
TITLE=GrafWin
CONTENTS=CONTENIDO

[FILES]
grafwin.rtf

[CONFIG]
BrowseButtons()


 [MAP]
#include <grafhelp.h>
```

#{bmc grafwin.bmp} Contenido

{bmc grafname.bmp}

Es un graficador de funciones matemáticas de la forma ***Z= f(x,y).*** Permite al usuario cambiar la perspectiva con la cual desea observar el gráfico, cambiar escalas en todos los ejes, variar la resolución, imprimir y grabar un gráfico con todos sus parámetros.

#$K+Fórmula

GrafWin presenta un diálogo de entrada de la función, cuando se escoja la opción fórmula desde el menú principal. El diálogo luce de la siguiente manera:

{bmc formula.bmp}

Usted podrá ingresar desde aquí la función que desee graficar. Si la función es válida, GrafWin la graficará.
Si el gráfico de la función que usted ingresó no resulta como lo esperaba, podría ser que no lo esta viendo desde la perspectiva deseada.
Si el gráfico resulta muy pequeño, muy grande o simplemente no se ve; podría ser problema de los parámetros de graficación.

Vea también :
Abriendo un archivo Grabando un gráfico

#HELP_FORMULA
$ Formula
K Formula
+ BOWLHELP:015

#$K+Parámetros

GrafWin deja que el usuario decida como va a ver el gráfico. Para eso cuenta con un juego de parámetros de graficación que le permitan realizar ésto.

Existen 4 tipos de Parámetros :

1. Escalas

Este grupo de parámetros controla el tamaño del gráfico en todos sus ejes :

2. Punto de Vista

Este grupo de parámetros controla el punto desde donde el usuario ve el gráfico, es decir la perspectiva.

3. Rangos

Estos parámetros controlan el rango de valores permitidos para las coordenadas X ,Y y Z.

4. Pasos

Son los deltas que van tomando X y X al momento de la evaluación.

GrafWin permite que el usuario pueda cambiar los parámetros de graficación, por medio del siguiente diálogo:

{bmc param.bmp}

Vea Tambien:
Parámetros Default

HELP_PARAMETROS
$ Parametros
K Parametros
+ BOWLHELP:020

#$K+Defaults

Como una manera de personalizar el ambiente, GrafWin permite al usuario poder grabar los parámetros de graficación que más utilice.El usuario podrá en cualquier momento tener acceso a estos datos y graficar haciendo uso de éstos o de los parámetros que decida usar.

Modificando los Parámetros Default

Escogiendo la opción Defaults desde el menú principal, GrafWin presenta un diálogo como el siguiente:

{bmc default.bmp}

El usuario puede editar cualquier campo y cuando esté todo correcto, dar un click en el botón de ***Ok***. Con ésto los parámetros defaults quedan grabados.

Graficando con los Parámetros Default

Escogiendo la opción Parámetros en el menú de Gráfico, GrafWin muestra el diálogo de especificación de Parámetros de graficación:

{bmc param.bmp}

El usuario puede editar cualquier campo, pero además puede setear estos campos con los parámetros default. Esto lo hace dando un click en el botón ***Default***.

Vea tambien:
Parámetros de Graficación

HELP_DEFAULT
$ Defaults
K Defaults
+ BOWLHELP:010

#$K+Abriendo un gráfico

Para abrir un gráfico ya grabado, se selecciona la opción ***Abrir*** en el menú de Archivo. Esto mostrará un diálogo que le permite escoger el archivo que contiene el gráfico. Estos archivos tienen el mismo formato de los archivos de inicialización de Windows (*.INI). Los archivos de GrafWin son de extensión *.GRF.

Si no hay errores en los datos grabados, GrafWin le mostrará el gráfico con todos los parámetros seteados al momento de grabación.

HELP_ABRIR
$ Abrir
K Abrir
+ BOWLHELP:025

#$K+Grabando un gráfico

Para grabar un gráfico, se selecciona la opción ***Grabar*** en el menú de Archivo. Si el gráfico es nuevo y no ha sido grabado anteriormente, GrafWin le pedirá el nombre del archivo donde quiere almacenar su gráfico. Si el gráfico ya ha sido grabado anteriormente y usted desea grabar los cambios hechos, GrafWin grabará los cambios directamente.

Si Usted no desea alterar el archivo abierto, pero si los nuevos cambios, puede escoger la opción ***Grabar como*** en el menú de Archivo para que GrafWin le pida el nombre del archivo que almacenará los nuevos cambios.

El gráfico se graba junto con todos los parámetros de graficación corrientes. Si desea ponerle algún nombre descriptivo, lo puede hacer en el diálogo de Especificación de Parámetros.

Ver también
Parámetros de Graficación

HELP_GRABAR
$ Grabar
K Grabar
+ BOWLHELP:030

#$K+Perspectiva

La Perspectiva o Punto de Vista viene dada en Coordenadas Polares, especificándose el ángulo con respecto al eje Z, el ángulo con respecto al eje X y la distancia desde el origen hasta el punto deseado.

{bmc puntv6.bmp}

El Punto de Vista es un parámetro de graficación, y como tal, puede ser modificado con el diálogo de Especificación de Parámetros.

Vea también:
Cambiando los parámetros de graficación

HELP_PERSPECTIVA
$ Cambiando la Perspectiva del gráfico
K Perspectiva
+ BOWLHELP:035

#$K+Formato de los Archivos .GRF

GrafWin graba los datos de un gráfico en un archivo con extensión GRF. Este archivo tiene el mismo formato que los archivos de inicialización de Windows (*.INI).

Formato

```
[Parametros]

RanMinX = -7          // Rango menor en X
RanMaxX = 7           // Rango mayor en X
RanMinY = -8          // Rango menor en Y
RanMaxY = 7           // Rango menor en Y
RanMinZ = -10         // Rango menor en Z
RanMaxZ = 10          // Rango menor en Z
DeltaX = 0.5          // Pasos en X
DeltaY = 0.5          // Pasos en Y
AnguloX = 30          // Ángulo con respecto a X
AnguloZ = 10          // Ángulo con respecto a Z
DistanciaZ = 100      // Distancia del origen al punto de vista
EscalaX = 2           // Escala en X
EscalaY = 2           // Escala en Y
EscalaZ = 2           // Escala en Z
Formula = sen(x + 2*(cos(y)-1.2))     // Función
Descripcion = Sin nombre              // Descripción de la función
```

HELP_ARCHIVO
$ Formato de los archivos GRF
K Formato de los archivos GRF
+ BOWLHELP:040

INDICE