Fractals Interactive Analyzer

Windows / C++ 1.0

AUTOR: ANTONIO CARRILLO LEDESMA

México D.F. Marzo del 2000

Fractals Interactive Analyzer

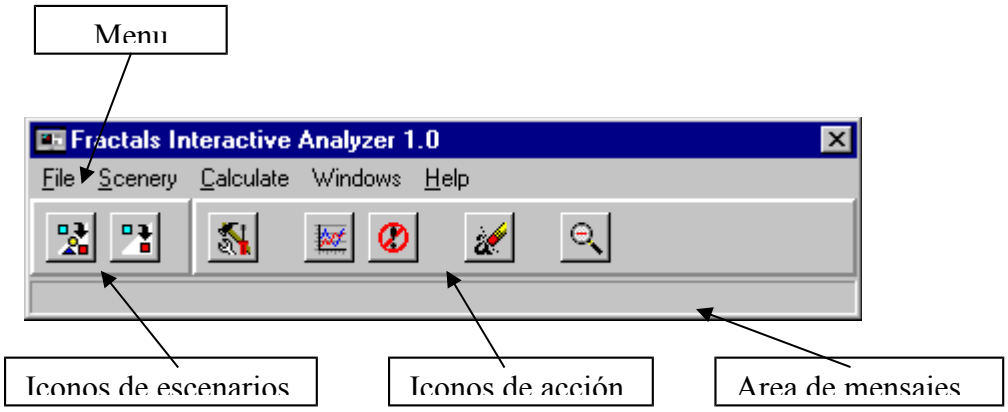
Windows/C++ 1.0

Autor: Antonio Carrillo Ledesma

Fractals Interactive Analyzer Windows/C++ 1.0 es un software interactivo para analizar varias familias de fractales de manera sencilla y completa, teniendo como características respecto a la interfaz gráfica del usuario:

- Su interfaz con el usuario está diseñada para que de una manera visual e intuitiva le permita hacer uso de todas las funciones del sistema con un mínimo de esfuerzo
- Tiene un sistema de menús sensitivos al contexto
- Tiene un conjunto de iconos de acceso directo a las funciones más utilizadas

El sistema Fractals Interactive Analyzer Windows/C++ 1.0 muestra la siguiente ventana principal:



En ella se destacan:

- Barra de menú principal
- Iconos de escenarios
- Iconos de acción directa
- Area de mensajes del Sistema

La barra de menú principal contiene todas las opciones que el sistema realiza, las cuales están duplicadas en los iconos que son accesos rápidos a las opciones del menú.

Los Iconos de escenarios son dos:

- Mandelbrot set
- Julia set

Con ellos se estudiará la dinámica de las familias de fractales que tiene el sistema Fractals Interactive Analyzer Windows/C++ 1.0.

Los iconos de acción directa tienen por objeto permitir acciones de una manera rápida a las acciones más socorridas que usa el sistema son:

- Configurar escenario activo
- Calcular (Mandelbrot o Julia según la ventana del escenario activo)
- Detener el cálculo del escenario activo
- Limpiar la ventana del escenario activo

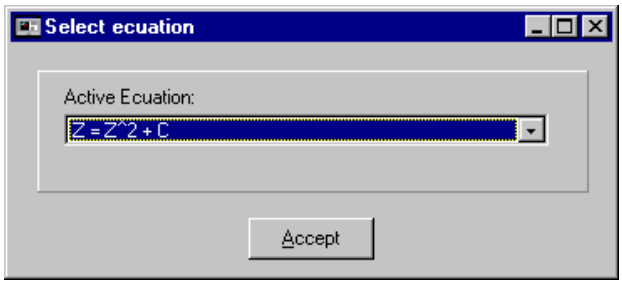
En el área de mensajes del sistema se muestra información sobre la acción que realizan los iconos y el estado de un proceso.

Las ventanas de los escenarios de Mandelbrot y Julia, tienen ecuación y parámetros independientes entre si, ya que es permitido lanzar múltiples ventanas de un mismo escenario y escoger en cada uno de ellos la ecuación activa.

El sistema Fractals Interactive Analyzer Windows/C++ 1.0 tiene las siguientes familias de fractales:

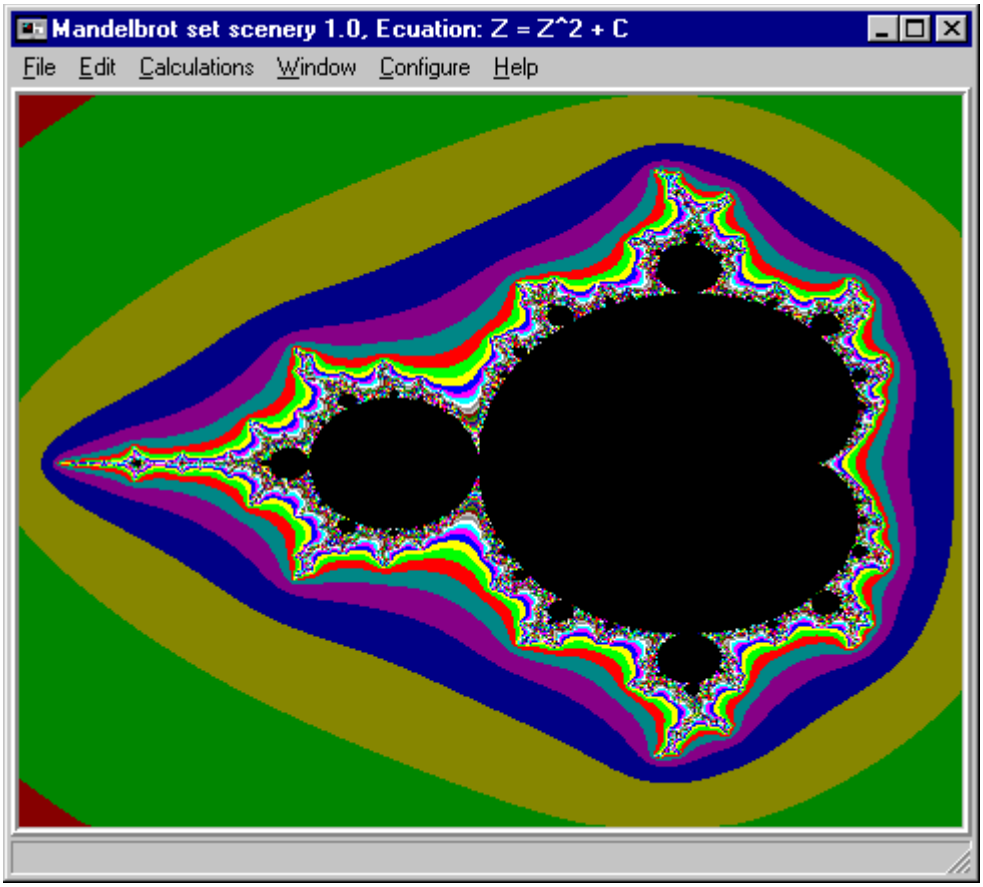
- Z = Z ^ 2 + C
- Z = A * sin(Z) + C
- Z = A * cos(Z) + C
- Z = A * exp(Z) + C
- Z = A * cosh(Z) + C
- Z = A * sinh(Z) + C

La ventana que se muestra para la selección de la ecuación activa es la siguiente:



En ella se selecciona la ecuación activa donde los parámetros de la ecuación serán las letras A,B,D, etc. La variable a iterar será la Z y la variable para generar el Mandelbrot set será la letra C.

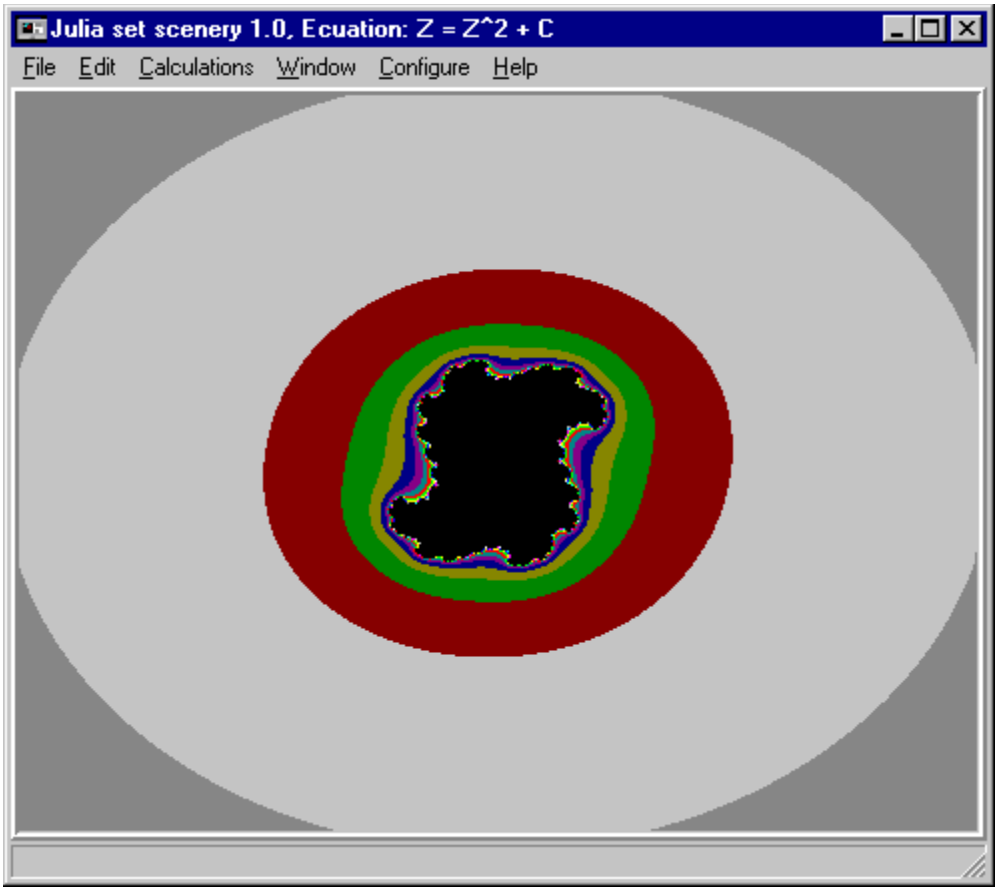
Una ejemplo del escenario Mandelbrot set es mostrado en la ventana siguiente:



En este escenario se destacan:

- Barra de menú
- Area de visualización del conjunto de Mandelbrot
- Area de mensajes del Sistema

Una ejemplo del escenario Julia set es mostrado en la ventana siguiente:



En este escenario se destacan:

- Barra de menú
- Area de visualización del conjunto de Julia
- Area de mensajes del Sistema

Diez Primeras Páginas

```
// Autor:     Antonio Carrillo Ledesma.
// R.F.C.:    CAAN-691229-TV7
// Dirección: Amsterdam 312 col. Hipódromo Condesa
// Teléfono:  55-74-43-53

// Propiedad intelectual, todos los derechos reservados conforme a la
ley, registro en trámite 1999-2000
// Revisión  1.1-A

//---------------------------------------------------------------------
----
#include <vcl.h>
#pragma hdrstop
USERES("Fractal.res");
USEFORM("Principal.cpp", VentanaPrincipal);
USEFORM("Ayuda.cpp", AyudaForm);
USEFORM("Acecade.cpp", VAcercaDe);
USEFORM("VentanaMandelbrot.cpp", FormaMandelbrot);
USEFORM("ParamMandelbrot.cpp", VCap_Mandelbrot);
USEUNIT("V_pixel.cpp");
USEUNIT("DefiniciónEcuación.cpp");
USEFORM("ParamJulia.cpp", VCap_Julia);
USEFORM("VentanaJulia.cpp", FormaJulia);
USEFORM("SelectEcuacion.cpp", FormSeleccionEcuacion);
//---------------------------------------------------------------------
----
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
        try
        {
                 Application->Initialize();
                 Application->CreateForm(__classid(TVentanaPrincipal),
&VentanaPrincipal);
                 Application->Run();
        }
        catch (Exception &exception)
        {
                 Application->ShowException(&exception);
        }
        return 0;
}
//---------------------------------------------------------------------
----
```

```
// Autor:     Antonio Carrillo Ledesma.
// R.F.C.:    CAAN-691229-TV7
// Dirección: Amsterdam 312 col. Hipódromo Condesa
// Teléfono:  55-74-43-53

// Propiedad intelectual, todos los derechos reservados conforme a la
ley, registro en trámite 1999-2000
// Revisión  1.1-A


//---------------------------------------------------------------------
----
#ifndef PrincipalH
#define PrincipalH
//---------------------------------------------------------------------
----
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ComCtrls.hpp>
#include <ExtCtrls.hpp>
#include <Menus.hpp>
#include <Buttons.hpp>
//---------------------------------------------------------------------
----
class TVentanaPrincipal : public TForm
{
__published:      // IDE-managed Components
        TStatusBar *BarraEstadoPrincipal;
        TMainMenu *Menu;
        TMenuItem *MenuFile;
        TMenuItem *MenuFileExit;
        TMenuItem *MenuHelp;
        TPanel *Panel1;
        TPanel *Panel2;
        TMenuItem *Separador1;
        TMenuItem *MenuHelpAboutOf;
        TMenuItem *MenuHelpGeneral;
        TSpeedButton *IconoMandelbrut;
        TSpeedButton *IconoJulia;
        TMenuItem *Separador2;
        TMenuItem *MenuFilePrintSetup;
        TMenuItem *MenuScenery;
        TMenuItem *MenuSceneryMandelbrot;
        TMenuItem *MenuSceneryJulia;
        TMenuItem *MenuCalculate;
        TMenuItem *MenuCalculateMandelbrot;
        TMenuItem *MenuCalculateJulia;
        void __fastcall FormCloseQuery(TObject *Sender, bool &CanClose);
        void __fastcall FormCreate(TObject *Sender);
        void __fastcall MenuFileExitClick(TObject *Sender);
        void __fastcall MenuHelpAboutOfClick(TObject *Sender);
        void __fastcall MenuHelpGeneralClick(TObject *Sender);
        void __fastcall IconoMandelbrutClick(TObject *Sender);
        void __fastcall IconoJuliaClick(TObject *Sender);
private:    // User declarations
```

```
        // Controla la visualizacion de Hint en la barra de estado
        void __fastcall OnHint(TObject *Sender);
public:           // User declarations
        __fastcall TVentanaPrincipal(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TVentanaPrincipal *VentanaPrincipal;
//---------------------------------------------------------------------------
#endif




// Autor:     Antonio Carrillo Ledesma.
// R.F.C.:    CAAN-691229-TV7
// Dirección: Amsterdam 312 col. Hipódromo Condesa
// Teléfono:  55-74-43-53

// Propiedad intelectual, todos los derechos reservados conforme a la
ley, registro en trámite 1999-2000
// Revisión  1.1-A


//---------------------------------------------------------------------------
#include <vcl.h>
#include <stdio.h>
#include "Acecade.h"
#include "Ayuda.h"
#include "DefinicionEcuacion.hpp"
#pragma hdrstop

#include "Principal.h"
#include "VentanaMandelbrot.h"
#include "VentanaJulia.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TVentanaPrincipal *VentanaPrincipal;

char TituloAplicacion[300];
C2D Posicion;
//---------------------------------------------------------------------------


__fastcall TVentanaPrincipal::TVentanaPrincipal(TComponent* Owner) :
TForm(Owner)
{
   Caption = "Fractal Systems 1.0";
   Application->Title = Caption;
   try {
      Application->Icon->LoadFromFile("FRACTAL.ICO");
```

```
   } catch (...) {};
   // Tiempo maximo de muestra de Hits
   Application->HintHidePause = 10000;
   Posicion.X = 0.0;
   Posicion.Y = 0.0;
}

// Al crear la forma ...
void __fastcall TVentanaPrincipal::FormCreate(TObject *Sender)
{
   // Asigna la rutina de visualizacion de la barra de estado
   Application->OnHint = &OnHint;
}

// Controla la visualizacion de Hint en la barra de estado
void __fastcall TVentanaPrincipal::OnHint(TObject *Sender)
{
   BarraEstadoPrincipal->SimpleText = Application->Hint;
}

// Controla la solicitud de cerrar la forma
void __fastcall TVentanaPrincipal::FormCloseQuery(TObject *Sender,bool
&CanClose)
{
   if (MessageBox(Handle,"Do you wish to end the
program?",TituloAplicacion,MB_YESNO + MB_ICONQUESTION)  == IDYES)
CanClose = true;
    else CanClose = false;
}


// MenuPrincipal->Archivo->Salir
void __fastcall TVentanaPrincipal::MenuFileExitClick(TObject *Sender)
{
   Close();
}

// Acerca de ...
void __fastcall TVentanaPrincipal::MenuHelpAboutOfClick(TObject *Sender)
{
   TVAcercaDe *Acercade = new TVAcercaDe(this);
   if (Acercade) {
      Acercade->ShowModal();
      delete Acercade;
   }
}

// Ayuda General
void __fastcall TVentanaPrincipal::MenuHelpGeneralClick(TObject *Sender)
{
   TAyudaForm *Ayuda = new TAyudaForm(this);
   if (Ayuda) {
      Ayuda->Abrir_archivo("General help","General.hlp");
      Ayuda->ShowModal();
      delete Ayuda;
   }
}
```

```
// Icono de ventana de Mandelbrot
void __fastcall TVentanaPrincipal::IconoMandelbrutClick(TObject *Sender)
{
   TFormaMandelbrot *vent_man = new TFormaMandelbrot(this);
   if(vent_man) {
      vent_man->Show();
   }
}

// Icono de ventana de Julia
void __fastcall TVentanaPrincipal::IconoJuliaClick(TObject *Sender)
{
   TFormaJulia *vent_jul = new TFormaJulia(this);
   if(vent_jul) {
      vent_jul->Show();
   }
}
```

```
// Autor:     Antonio Carrillo Ledesma.
// R.F.C.:    CAAN-691229-TV7
// Dirección: Amsterdam 312 col. Hipódromo Condesa
// Teléfono:  5-74-43-53

// Propiedad intelectual, todos los derechos reservados conforme a la
ley, registro en trámite 1999-2000
// Revisión  1.1-A


#ifndef __V_PIXEL_HPP__
#define __V_PIXEL_HPP__

#include "definic.hpp"

#define REVISAR

// Definición del pixel
struct Definicion_pixel {
   bool pixel;
};


// Clase que manipila y controla un array de pixeles
class Arreglo_pixeles
{
   private:

      bool               Activo;        // Indica si esta activo el
objeto
      int                Num_Y;         // Longitud de la matriz en Y
      int                Num_X;         // Longitud de la matrix en X
```

```
      Definicion_pixel **Arreglo;       // Puntero al arreglo de pixeles
      Definicion_pixel  *prtArreglo;       // Puntero al arreglo de
pixeles

      C2D                Escala;        // Escala usada dentro de la
ventana de pixeles
      Dimencion_Ventana  Dimenciones;   // Dimenciones de la ventana de
trabajo
      int                px, py;        // Valores temporales
      int                i1,i2;
      bool               st;

   public:
                     // Constructor de la clase
                     Arreglo_pixeles(void)
                     { Activo = false; }
                     // Destructor de la clase
                    ~Arreglo_pixeles(void)
                     {Destruye();}
                     // Calcula la posición de un punto dentro de la
ventana de pixeles
               bool  Calcula_punto(const long double x, const long double
y, int &px, int &py)
                     {
                        st = true;
#ifdef REVISAR
                        if ((y >= Dimenciones.Yi && y <= Dimenciones.Yf)
&& (x >= Dimenciones.Xi && x <= Dimenciones.Xf)) {
#endif
                           py = Num_Y - (((y - Dimenciones.Yi) *
Escala.Y) + 1.0);
                           px = (((x - Dimenciones.Xi) * Escala.X));
#ifdef REVISAR
                        } else st = false;
#endif
                        return st;
                     }
                     // Inicializa la clase
               void  Inicializa(const int pix_x, const int pix_y, const
bool val, const Dimencion_Ventana v_dim);
                     // Asigna a un pixel del arreglo el valor indicado
               void  Asigna_valor(const int pix_x, const int pix_y, const
bool val)
                     {
                        if (Activo && pix_x >= 0 && pix_x < Num_X &&
pix_y >= 0 && pix_y < Num_Y) Arreglo[pix_x][pix_y].pixel = val;
                     }
                     // Asigna a un pixel del arreglo el valor indicado
               void  Asigna_valor(const long double x, const long double
y, const bool val)
                     {
                        Calcula_punto(x, y, px, py);
                        Asigna_valor(px, py, val);
                     }
                     // Destruye el contenido del arreglo
               void  Destruye(void);
```

```
                    // Retorna el valor del pixel indicado dentro del
arreglo
              bool  Retorna_valor(const int pix_x, const int pix_y)
const
                    {
                       if (Activo) {
                          if (pix_x < Num_X && pix_y < Num_Y)
                             return (Arreglo[pix_x][pix_y].pixel);
                       }
                       return false;
                    }
                    // Asigna a todo el array el valor especificado
              void  Asigna_todo_array(const bool val)
                    {
                       if (Activo) {
                          for (i1 = 0; i1 <  Num_X; i1++) {
                             prtArreglo = &Arreglo[i1][0];
                             for (i2 = 0; i2 <  Num_Y; i2++) {
                                prtArreglo->pixel = val;
                                prtArreglo++;
                             }
                          }
                       }
                    }
              void  Asigna_linea(const int xi, const int yi, const int
xf, const int yf);
              void  Asigna_linea(const long double xi, const long double
yi, const long double xf, const long double yf);
              void  Cambia_dimension(const Dimencion_Ventana v_dim,
const bool val = false);
              void  Cambia_cantidad_pixeles(const int pix_x, const int
pix_y, const bool val, const Dimencion_Ventana v_dim);
};


// Notas:
//   (1) El array empieza en cero para los pixeles Y y X
//   (2) El numero maximo de elemetos depende de la memoria de la maquina

#endif
```

```
// Autor:     Antonio Carrillo Ledesma.
// R.F.C.:    CAAN-691229-TV7
// Dirección: Amsterdam 312 col. Hipódromo Condesa
// Teléfono:  5-74-43-53

// Propiedad intelectual, todos los derechos reservados conforme a la
ley, registro en trámite 1999-2000
// Revisión  1.1-A


#include "V_pixel.hpp"

// Inicializa el arreglo de pixeles con el valor VAL
void Arreglo_pixeles::Inicializa(const int pix_x, const int pix_y, const
bool val, const Dimencion_Ventana v_dim)
{
   if (Activo) return;
   // Declara el arreglo para los renglones
   Arreglo = new Definicion_pixel *[pix_x];
   // Declara el arreglo para las columnas dentro de cada renglon
   for (i1 = 0; i1 < pix_x; i1++) Arreglo[i1] = new Definicion_pixel
[pix_y];
   Num_X = pix_x, Num_Y = pix_y;
   Activo = true;
   Cambia_dimension(v_dim, val);
}

// Cambia la cantidad de pixeles en la ventana de pixeles
void  Arreglo_pixeles::Cambia_cantidad_pixeles(const int pix_x, const int
pix_y, const bool val, const Dimencion_Ventana v_dim)
{
   Destruye();
   Inicializa(pix_x, pix_y, val, v_dim);
}

// Cambia la dimension de la ventana de trabajo
void Arreglo_pixeles::Cambia_dimension(const Dimencion_Ventana v_dim,
const bool val)
{
   // Dimenciones de la ventana de trabajo
   Dimenciones = v_dim;
   // Escala para trabajar en la ventana de pixeles
   Escala.X = (Num_X -1) / (Dimenciones.Xf - Dimenciones.Xi);
   Escala.Y = (Num_Y -1) / (Dimenciones.Yf - Dimenciones.Yi);
   Asigna_todo_array(val);
}


// Destruye el contenido del arreglo
void Arreglo_pixeles::Destruye(void)
{
   if (!Activo) return;
   // Destruye el arreglo
   for (i1 = 0; i1 < Num_X; i1++) delete []Arreglo[i1];
   delete []Arreglo;
   Activo = false;
}
```

```
// Asigna linea
void Arreglo_pixeles::Asigna_linea(const int xi, const int yi, const int
xf, const int yf)
{
     int min_x, max_x, min_y, max_y, xxi, xyi, xxf, xyf, py;
     long double m;
     int i;

     xxi = xi, xxf = xf, xyi = yi, xyf = yf;
     min_x = xxi < xxf ? xxi: xxf;
     max_x = xxf > xxi ? xxf: xxi;
     min_y = xyi < xyf ? xyi: xyf;
     max_y = xyf > xyi ? xyf: xyi;
     // Linea vertical
     if (xxi == xxf) {
        for (i = min_y; i <= max_y; i++) Asigna_valor(xi, i, true);
     }
     // Linea horizontal
     if (xyi == xyf) {
        for (i = min_x; i <= max_x; i++) Asigna_valor(i, yi, true);
     }
     // Linea en general
     if (xxi != xxf && xyi != xyf) {
        m = (xyf - xyi) / (long double) (xxf - xxi);
        for (i = min_x; i <= max_x; i++) {
            py = m * (i - xxf) + xyf;
            Asigna_valor(i, py, true);
        }
     }
}

// Asigna linea
void Arreglo_pixeles::Asigna_linea(const long double xi, const long
double yi, const long double xf, const long double yf)
{
   int p1, p2, p3, p4;
////////////////////////////////////////////////////////////////////////
//////
//#error Ajustar los valores para generar una linea truncada a la ventana
   if (!Calcula_punto(xi, yi, p1, p2)) return;
   if (!Calcula_punto(xf, yf, p3, p4)) return;
////////////////////////////////////////////////////////////////////////
//////
   Asigna_linea(p1, p2, p3, p4);
}
```

```
// Autor:     Antonio Carrillo Ledesma.
// R.F.C.:    CAAN-691229-TV7
// Dirección: Amsterdam 312 col. Hipódromo Condesa
// Teléfono:  5-74-43-53

// Propiedad intelectual, todos los derechos reservados conforme a la
ley, registro en tr mite
// Revisi¢n  1.1-A


#ifndef __DEFINIC_HPP__
#define __DEFINIC_HPP__

struct Dimencion_Ventana {
    long double Xi;
    long double Yi;
    long double Xf;
    long double Yf;
};

struct Definicion_Ventana {
    int Xi;
    int Yi;
    int Xf;
    int Yf;
};


struct C2D {
    long double X;
    long double Y;
};

struct C3D {
    long double X;
    long double Y;
    long double Z;


};

#endif
```

Diez Ultimas Páginas

```
// Menu->Archivo->Grabar BMP
void __fastcall TFormaMandelbrot::MenuArchivoGrabarBMPClick(TObject
*Sender)
{
   TRect   xRect = Rect(0,0,ClientWidth,ClientHeight-20);
   Graphics::TBitmap *Bitmap = new Graphics::TBitmap;
   Bitmap->Width = ClientWidth;
   Bitmap->Height = ClientHeight-20;
   Bitmap->Canvas->CopyRect(xRect, Canvas, xRect);
   SaveDialog->Title = "Save as ...";
   if (SaveDialog->Execute()) {
      Bitmap->SaveToFile(SaveDialog->FileName);
   }
   delete Bitmap;
}

// Menu->Archivo->Imprimir
void __fastcall TFormaMandelbrot::MenuArchivoImprimirClick(TObject
*Sender)
{
   TRect   xRect = Rect(0,0,ClientWidth,ClientHeight-20);
   Graphics::TBitmap *Bitmap = new Graphics::TBitmap;
   Bitmap->Width = ClientWidth;
   Bitmap->Height = ClientHeight-20;
   Bitmap->Canvas->CopyRect(xRect, Canvas, xRect);
   Printer()->BeginDoc();
   Printer()->Canvas->StretchDraw(Rect(50,50,Printer()->PageWidth-
50,Printer()->PageWidth-50),Bitmap);
   Printer()->EndDoc();
   delete Bitmap;
}

// Menu->Archivo->Cerrar
void __fastcall TFormaMandelbrot::MenuArchivoCerrarClick(TObject *Sender)
{
   Close();
}

// Menu->Copiar->Copiar la posición del mouse
void __fastcall TFormaMandelbrot::MenuEdicionCopiarClick(TObject *Sender)
{
   Posicion.X = Vs.Xi;
   Posicion.Y = Vs.Yi;
}


// Menu->Calcular->Bifurcaciones
void __fastcall TFormaMandelbrot::MenuCalculaBifurcacionesClick(TObject
*Sender)
{
   Calcula_Mandelbrot();
}

// Menu->Calcular->Numero de Rotación// Menu->Calcular->Sincronización//
Menu->Calcular->Exponente de Lyapunov// Menu->Ventana->Detener el calculo
void __fastcall
TFormaMandelbrot::MenuCalcularDetenerelcalculoClick(TObject *Sender)
```

```
{
   Sw_cancela_calculo  = true;
}


// Menu->Ventana->Limpiar
void __fastcall TFormaMandelbrot::MenuVentanaLimpiarClick(TObject
*Sender)
{
   Sw_se_grafico = false;
   Limpia_matriz_pixeles();
   Limpiar_vantana();
   Grafica();
}

// Menu->Ventana->Zoom Out
void __fastcall TFormaMandelbrot::MenuVentanaZoomOutClick(TObject
*Sender)
{
   Sw_se_grafico = false;
   // Almacena las dimensiones actuales
   if (Ind_zoom > 0) {
      Ind_zoom --;
      Dim_Vtn = Dim_zoom[Ind_zoom];
      for (int i = 0; i < 16; i++ ) pix[i].Cambia_dimension(Dim_Vtn);
      // Calcula la escala de la ventana de visualizacion
      Escala1.X = (VTMandelbrot->Width -1) / (Dim_Vtn.Xf - Dim_Vtn.Xi);
      Escala1.Y = (VTMandelbrot->Height -1) / (Dim_Vtn.Yf - Dim_Vtn.Yi);
      FormPaint(this);
      Calcula_Mandelbrot();
      if (Ind_zoom < 1) MenuVentanaZoomOut->Enabled = false;
   }
}

// Menu->Ventana->Dimensiones originales
void __fastcall
TFormaMandelbrot::MenuVentanaDimensionesOriginalesClick(TObject *Sender)
{
   Sw_se_grafico = false;
   Dim_Vtn = Dim_orig;
   Ind_zoom = 0;
   MenuVentanaZoomOut->Enabled = false;
   for (int i = 0; i < 16; i++ ) pix[i].Cambia_dimension(Dim_Vtn);
   // Calcula la escala de la ventana de visualizacion
   Escala1.X = (VTMandelbrot->Width -1) / (Dim_Vtn.Xf - Dim_Vtn.Xi);
   Escala1.Y = (VTMandelbrot->Height -1) / (Dim_Vtn.Yf - Dim_Vtn.Yi);
   Calcula_Mandelbrot();
}

// Menu->Configura->Ecuacion
void __fastcall TFormaMandelbrot::MenuConfigurarEcuationClick(TObject
*Sender)
{
   TFormSeleccionEcuacion *configura = new TFormSeleccionEcuacion(this);
   if (configura) {
      // Sistema activo
      configura->ComboBoxSeleccionEcuacion->Items->Clear();
```

```
      for( int i = 0; i < NUM_MAX_ECUACIONES; i++) {
         configura->ComboBoxSeleccionEcuacion->Items-
>Add(def_ecu.Ecuacion_texto[i]);
      }
      configura->ComboBoxSeleccionEcuacion->ItemIndex =
def_ecu.Sistema_activo;
      configura->ShowModal();
      if (configura->Aceptar) {
         // Sistema activo
         def_ecu.Sistema_activo = configura->ComboBoxSeleccionEcuacion-
>ItemIndex;
         // Asigna parámetros
         def_ecu.AsignaParametros();
         sprintf(xcad,"Mandelbrot scenery 1.0, Ecuation:
%s",def_ecu.Ecuacion_texto[def_ecu.Sistema_activo]);
         Caption = xcad;
         Sw_se_grafico = false;
      }
      delete configura;
   }
   Limpiar_vantana();
}


// Menu->Configura->Parametros
void __fastcall TFormaMandelbrot::MenuConfigurarParametrosClick(TObject
*Sender)
{
   configura = new TVCap_Mandelbrot(this);
   if (configura) {
      PasarValoresConfiguracion();
      configura->TabbedNotebook1->PageIndex = 0;
      configura->ShowModal();
      RetornarValoresConfiguracion();
      delete configura;
   }
}


// Menu->Configura->Dimensiones
void __fastcall TFormaMandelbrot::MenuConfigurarDimensionsClick(TObject
*Sender)
{
   configura = new TVCap_Mandelbrot(this);
   if (configura) {
      PasarValoresConfiguracion();
      configura->TabbedNotebook1->PageIndex = 1;
      configura->ShowModal();
      RetornarValoresConfiguracion();
      delete configura;
   }
}

// Menu->Configura->Calcular
void __fastcall TFormaMandelbrot::MenuConfigurarCalculateClick(TObject
*Sender)
{
```

```
   configura = new TVCap_Mandelbrot(this);
   if (configura) {
      PasarValoresConfiguracion();
      configura->TabbedNotebook1->PageIndex = 2;
      configura->ShowModal();
      RetornarValoresConfiguracion();
      delete configura;
   }
}



// Menu->Ayuda->Acerca de ...
void __fastcall TFormaMandelbrot::MenuAyudaAcercadeClick(TObject *Sender)
{
   TVAcercaDe *Acercade = new TVAcercaDe(this);
   if (Acercade) {
      Acercade->ShowModal();
      delete Acercade;
   }
}

// Menu->Ayuda->Bifurcaciones
void __fastcall TFormaMandelbrot::MenuAyudaBifurcacionesClick(TObject
*Sender)
{
   TAyudaForm *Ayuda = new TAyudaForm(this);
   if (Ayuda) {
      Ayuda->Abrir_archivo("Bifurcations scenery","Bifurcaciones.hlp");
      Ayuda->ShowModal();
      delete Ayuda;
   }
}

//////////////////////////////////////////////////////////////////////////
//////
//  Definición de Comportamientos de bifurcaciones
//////////////////////////////////////////////////////////////////////////
//////

TColor ColoresM[] = {
   clWhite,
   clGray,
   clSilver,
   clMaroon,
   clGreen,
   clOlive,
   clNavy,
   clPurple,
   clTeal,
   clRed,
   clLime,
   clYellow,
   clBlue,
   clFuchsia,
   clAqua,
   clBlack
```

```
};



void TFormaMandelbrot::Calcula_Mandelbrot(void)
{
   Sw_se_grafico = true;

   long double Incx, Incy;
   int px, py;
   int i;
   C2D pos;

   Incx = (Dim_Vtn.Xf - Dim_Vtn.Xi) / (long double) (Pix_x);
   Incy = (Dim_Vtn.Yf - Dim_Vtn.Yi) / (long double) (Pix_y);

   // Loop vertical
   for (pos.Y = Dim_Vtn.Yf; pos.Y >= Dim_Vtn.Yi; pos.Y -= Incy) {
       // Controla la cancelacion del calculo
       Application->ProcessMessages();
       if (Sw_cancela_calculo) break;
       // Loop horizontal
       for (pos.X = Dim_Vtn.Xi; pos.X < Dim_Vtn.Xf; pos.X += Incx) {
           i = def_ecu.Calcula_Mandelbrot(pos.X, pos.Y);
           // dibuja el punto
           pix[i].Asigna_valor(pos.X,pos.Y,true);
           px = (pos.X - Dim_Vtn.Xi) * Escala1.X;
           py = VTMandelbrot->Height - (((pos.Y - Dim_Vtn.Yi) *
Escala1.Y) + 1.0);
           if (px >= 0 && px < VTMandelbrot->Width && py >= 0 && py <
VTMandelbrot->Height) VTMandelbrot->Canvas->Pixels[px][py] = ColoresM[i];
       }
   }
}


// Grafica las curvas calculadas
void TFormaMandelbrot::Grafica(void)
{
   if (!Sw_se_grafico) return;
   unsigned int x, y;
   C2D escala;

   VTMandelbrot->Canvas->Brush->Color = clBlack;
   VTMandelbrot->Canvas->Rectangle(0, 0, VTMandelbrot->Width,
VTMandelbrot->Height);
   // Calcula la escala de la ventana de visualizacion con respecto a la
ventana de pixeles
   escala.X = VTMandelbrot->Width / (long double) Pix_x;
   escala.Y = VTMandelbrot->Height / (long double) Pix_y;

   // Visualiza el diagramna de bifurcaciones
   for (int i = 0; i < 15; i++ ) {
      for (y = 0; y < Pix_y; y++) {
         for (x = 0; x < Pix_x; x++) {
             if (pix[i].Retorna_valor(x, y)) VTMandelbrot->Canvas-
>Pixels[x * escala.X][y * escala.Y] = ColoresM[i];
```

```
          }
       }
   }
}

// Limpia la ventana de graficacion
void TFormaMandelbrot::Limpiar_vantana(void)
{
   VTMandelbrot->Canvas->Pen->Color = Color_fondo;
   VTMandelbrot->Canvas->Brush->Color = Color_fondo;
   VTMandelbrot->Canvas->Rectangle(0, 0, VTMandelbrot->Width,
VTMandelbrot->Height);
}

////////////////////////////////////////////////////////////////////////
//////
// Controla el movimiento del mouse y genereación del recuadro para el
zoom del
// la ventana de bifurcaciones
////////////////////////////////////////////////////////////////////////
//////



// Pasa los valores de configuración de la ventana de captura
void TFormaMandelbrot::PasarValoresConfiguracion(void)
{
   ////////////////////////////////////////////////////
   // Parámetros
   ////////////////////////////////////////////////////
   // Parámetros
   if (def_ecu.Numero_parametros[def_ecu.Sistema_activo]) {
      configura->StaticText1->Visible = true;
      configura->ListBoxParametros->Visible = true;
      configura->ListBoxParametros->Items->Clear();
//      for(int i = 0; i <
def_ecu.Numero_parametros[def_ecu.Sistema_activo]; i++) {
         sprintf(xcad,"Re(A)  = %1.9Lf",def_ecu.P[def_ecu.Sistema_activo]
[0]);
         configura->ListBoxParametros->Items->Add(xcad);
         sprintf(xcad,"Im(A)  = %1.9Lf",def_ecu.P[def_ecu.Sistema_activo]
[1]);
         configura->ListBoxParametros->Items->Add(xcad);
//      }
   }
   ////////////////////////////////////////////////////
   // Dimensiones
   ////////////////////////////////////////////////////
   sprintf(xcad,"%Lf",Dim_Vtn.Xi);
   configura->EditHMin->Text = (AnsiString) xcad;
   sprintf(xcad,"%Lf",Dim_Vtn.Xf);
   configura->EditHMax->Text = (AnsiString) xcad;
   sprintf(xcad,"%Lf",Dim_Vtn.Yi);
   configura->EditVMin->Text = (AnsiString) xcad;
   sprintf(xcad,"%Lf",Dim_Vtn.Yf);
   configura->EditVMax->Text = (AnsiString) xcad;
   ////////////////////////////////////////////////////
```

```
   // Escenario
   ////////////////////////////////////////////////////
   sprintf(xcad,"%Lf",def_ecu.Norma_Maxima_Mandelbrot);
   configura->Edit8->Text = (AnsiString) xcad;
}

// Retorna los valores de configuración de la ventana de captura
void TFormaMandelbrot::RetornarValoresConfiguracion(void)
{
   if (!configura->Aceptar) return;
   ////////////////////////////////////////////////////
   // Parámetros
   ////////////////////////////////////////////////////
//   for (int i = 0; i <
def_ecu.Numero_parametros[def_ecu.Sistema_activo]; i++) {
       int i = 0;
       unsigned int i1, i2;
       char xcad[100], xcad1[100];
       strcpy(xcad1,configura->ListBoxParametros->Items-
>Strings[i].c_str());
       // Valor del parámetro
       for (i2 = 0, i1 = 9; i1 < strlen(xcad1); i2++, i1++) xcad[i2] =
xcad1[i1];
       xcad[i2] = 0;
       def_ecu.P[def_ecu.Sistema_activo][0] = _atold(xcad);

       i++;
       strcpy(xcad1,configura->ListBoxParametros->Items-
>Strings[i].c_str());
       // Valor del parámetro
       for (i2 = 0, i1 = 9; i1 < strlen(xcad1); i2++, i1++) xcad[i2] =
xcad1[i1];
       xcad[i2] = 0;
       def_ecu.P[def_ecu.Sistema_activo][1] = _atold(xcad);
//   }
   def_ecu.AsignaParametros();
   ////////////////////////////////////////////////////
   // Dimensiones
   ////////////////////////////////////////////////////
   Dim_Vtn.Xi = _atold(configura->EditHMin->Text.c_str());
   Dim_Vtn.Xf = _atold(configura->EditHMax->Text.c_str());
   Dim_Vtn.Yi = _atold(configura->EditVMin->Text.c_str());
   Dim_Vtn.Yf = _atold(configura->EditVMax->Text.c_str());
   ////////////////////////////////////////////////////
   // Escenario
   ////////////////////////////////////////////////////
   def_ecu.Norma_Maxima_Mandelbrot = _atold(configura->Edit8-
>Text.c_str());
   ////////////////////////////////////////////////////
   // Al terminar
   ////////////////////////////////////////////////////
   for (int i = 0; i < 16; i++ ) pix[i].Cambia_dimension(Dim_Vtn);
   // Calcula la escala de la ventana de visualizacion
   Escala1.X = (VTMandelbrot->Width -1) / (Dim_Vtn.Xf - Dim_Vtn.Xi);
   Escala1.Y = (VTMandelbrot->Height -1) / (Dim_Vtn.Yf - Dim_Vtn.Yi);
   Calcula_Mandelbrot();
}
```

```
//////////////////////////////////////////////////////////////////////////
//////
// Controla el movimiento del mouse y genereación del recuadro para el
zoom del
// la ventana de bifurcaciones
//////////////////////////////////////////////////////////////////////////
//////

// Al presionar el botón del mouse
void __fastcall TFormaMandelbrot::VTMandelbrotMouseDown(TObject
*Sender,TMouseButton Button, TShiftState Shift, int X, int Y)
{
   if (Sw_Recuadro_activo) {
      if (Button == mbLeft) {
         Vt2.Xf = Vt1.Xf = Vt2.Xi = Vt1.Xi = X;
         Vt2.Yf = Vt1.Yf = Vt2.Yi = Vt1.Yi = Y;
         Sw_Dibuja_rectangulo = true;
         Sw_Dibuja_rectangulo_ant = false;
         Ventana_seleccionada = false;
      }
   }
}


// Al mover el mouse
void __fastcall TFormaMandelbrot::VTMandelbrotMouseMove(TObject
*Sender,TShiftState Shift, int X, int Y)
{
   if (Sw_Recuadro_activo) {
      if (Sw_Dibuja_rectangulo) {
         TPenMode mode;

         mode = VTMandelbrot->Canvas->Pen->Mode;
         VTMandelbrot->Canvas->Pen->Mode = pmNot;

         if (Sw_Dibuja_rectangulo_ant) {
            VTMandelbrot->Canvas->MoveTo(Vt2.Xi,Vt2.Yi);
            VTMandelbrot->Canvas->LineTo(Vt2.Xf,Vt2.Yi);
            VTMandelbrot->Canvas->MoveTo(Vt2.Xi,Vt2.Yi);
            VTMandelbrot->Canvas->LineTo(Vt2.Xi,Vt2.Yf);
            VTMandelbrot->Canvas->MoveTo(Vt2.Xf,Vt2.Yf);
            VTMandelbrot->Canvas->LineTo(Vt2.Xi,Vt2.Yf);
            VTMandelbrot->Canvas->MoveTo(Vt2.Xf,Vt2.Yf);
            VTMandelbrot->Canvas->LineTo(Vt2.Xf,Vt2.Yi);
         }
         VTMandelbrot->Canvas->MoveTo(Vt1.Xi,Vt1.Yi);
         VTMandelbrot->Canvas->LineTo(X,Vt1.Yi);
         VTMandelbrot->Canvas->MoveTo(Vt1.Xi,Vt1.Yi);
         VTMandelbrot->Canvas->LineTo(Vt1.Xi,Y);
         VTMandelbrot->Canvas->MoveTo(X,Y);
         VTMandelbrot->Canvas->LineTo(Vt1.Xi,Y);
         VTMandelbrot->Canvas->MoveTo(X,Y);
         VTMandelbrot->Canvas->LineTo(X,Vt1.Yi);

         Vt2.Xf = X;
```

```
         Vt2.Yf = Y;
         VTMandelbrot->Canvas->Pen->Mode = mode;
         Sw_Dibuja_rectangulo_ant = true;

         // Ajusta a que xVt tenga el inicio y el final de la ventana
         xVt.Xi = Vt1.Xi, xVt.Xf = Vt2.Xf, xVt.Yi = Vt1.Yi, xVt.Yf =
Vt2.Yf;
         int  xtmp;
         if (xVt.Xi > xVt.Xf) xtmp = xVt.Xi, xVt.Xi = xVt.Xf, xVt.Xf =
xtmp;
         if (xVt.Yi > xVt.Yf) xtmp = xVt.Yi, xVt.Yi = xVt.Yf, xVt.Yf =
xtmp;
         // Ajusta los valores a la longitud de la ventana
         if (xVt.Xi < 0) xVt.Xi = 0;
         if (xVt.Yi < 0) xVt.Yi = 0;
         if (xVt.Xf > VTMandelbrot->Width) xVt.Xf = VTMandelbrot->Width;
         if (xVt.Yf > VTMandelbrot->Height) xVt.Yf = VTMandelbrot-
>Height;
         // Visualiza los valores de la region seleccionada segun la
dimención de la ventana
         Vs.Xi = xVt.Xi / Escala1.X + Dim_Vtn.Xi;
         Vs.Yi = (-(xVt.Yi - VTMandelbrot->Height) / Escala1.Y) +
Dim_Vtn.Yi;
         Vs.Xf = xVt.Xf / Escala1.X + Dim_Vtn.Xi;
         Vs.Yf = (-(xVt.Yf - VTMandelbrot->Height) / Escala1.Y) +
Dim_Vtn.Yi;
         sprintf(xcad,"%s: (%3.5Lf, %3.5Lf, %3.5Lf,
%3.5Lf)",VGM_TXT01,Vs.Xi,Vs.Yf,Vs.Xf,Vs.Yi);
         BarraDeEstadosBifurcaciones->SimpleText = (AnsiString) xcad;
      } else {
         Vs.Xf = X, Vs.Yf = Y;
         Vs.Xi = Vs.Xf / Escala1.X + Dim_Vtn.Xi;
         Vs.Yi = (-(Vs.Yf - VTMandelbrot->Height) / Escala1.Y) +
Dim_Vtn.Yi;

         sprintf(Msg,"Mouse position: (%3.8Lf, %3.8Lf)", Vs.Xi, Vs.Yi);
         BarraDeEstadosBifurcaciones->SimpleText = (AnsiString) Msg;
      }
   }
}

// Al soltar el botón del mouse
void __fastcall TFormaMandelbrot::VTMandelbrotMouseUp(TObject
*Sender,TMouseButton Button, TShiftState Shift, int X, int Y)
{
   if (Sw_Recuadro_activo) {
      if (Button == mbLeft) {
         TPenMode mode;
         mode = VTMandelbrot->Canvas->Pen->Mode;
         VTMandelbrot->Canvas->Pen->Mode = pmNot;
         VTMandelbrot->Canvas->MoveTo(Vt2.Xi,Vt2.Yi);
         VTMandelbrot->Canvas->LineTo(Vt2.Xf,Vt2.Yi);
         VTMandelbrot->Canvas->MoveTo(Vt2.Xi,Vt2.Yi);
         VTMandelbrot->Canvas->LineTo(Vt2.Xi,Vt2.Yf);
         VTMandelbrot->Canvas->MoveTo(Vt2.Xf,Vt2.Yf);
         VTMandelbrot->Canvas->LineTo(Vt2.Xi,Vt2.Yf);
         VTMandelbrot->Canvas->MoveTo(Vt2.Xf,Vt2.Yf);
```

```
         VTMandelbrot->Canvas->LineTo(Vt2.Xf,Vt2.Yi);
         VTMandelbrot->Canvas->Pen->Mode = mode;
         Vt1.Xf = X;
         Vt1.Xf = Y;
         Sw_Dibuja_rectangulo = false;
         Sw_Dibuja_rectangulo_ant = false;
         BarraDeEstadosBifurcaciones->SimpleText = (AnsiString) " ";
         Ventana_seleccionada = true;
         // Revisa si realmente se solicito el zoom
         if ((Vt2.Xf - Vt2.Xi) > 5 && (Vt2.Yf - Vt2.Yi) > 5) {
            // Pregunta si se desea hacer el zoom
            if (MessageBox(Handle,"Do you wish to zoom?","Bifurcations
scenery",MB_YESNO + MB_ICONQUESTION)  == IDYES) {
               // Almacena las dimensiones actuales
               Dim_zoom[Ind_zoom] = Dim_Vtn;
               if ((Ind_zoom + 1) < NUM_MAX_DIM_ZOOM) Ind_zoom ++;
               MenuVentanaZoomOut->Enabled = true;
               long double incx = fabsl(Vs.Xf - Vs.Xi);
               long double incy = fabsl(Vs.Yf - Vs.Yi);
               long double inc  = (incx > incy ? incx : incy);
               // Actualiza las dimensiones de las ventanas de trabajo
               Dim_Vtn.Xi = Vs.Xi;
               Dim_Vtn.Yi = Vs.Yf;
               Dim_Vtn.Xf = Vs.Xi + inc;
               Dim_Vtn.Yf = Vs.Yf + inc;
               for (int i = 0; i < 16; i++)
pix[i].Cambia_dimension(Dim_Vtn);
               // Calcula la escala de la ventana de visualizacion
               Escala1.X = (VTMandelbrot->Width -1) / (Dim_Vtn.Xf -
Dim_Vtn.Xi);
               Escala1.Y = (VTMandelbrot->Height -1) / (Dim_Vtn.Yf -
Dim_Vtn.Yi);
               Calcula_Mandelbrot();
            }
         }
      }
   }
}

// Solicitu de abrir ventana de Julia con la posición del mouse
void __fastcall TFormaMandelbrot::OpenJuliaWindow1Click(TObject *Sender)
{
   TFormaJulia *vent_jul = new TFormaJulia(NULL);
   if(vent_jul) {
      vent_jul->def_ecu.P[0][0] = Vs.Xi;
      vent_jul->def_ecu.P[0][1] = Vs.Yi;
      vent_jul->Show();
   }
}


// Al solicitar repintar la forma
void __fastcall TFormaMandelbrot::VTMandelbrotPaint(TObject *Sender)
{
   Grafica();
}
```

Listado de Archivos fuentes

```
02/10/00  06:37p                847 Acecade.cpp
02/10/00  06:42p             34,734 Acecade.dfm
02/09/00  05:50p              1,290 Acecade.h
02/09/00  05:50p              2,564 Ayuda.cpp
01/31/00  05:45p              1,183 Ayuda.dfm
02/09/00  05:50p              1,839 Ayuda.h
02/09/00  06:42p                683 DEFINIC.HPP
03/03/00  10:42p              5,586 DefinicionEcuacion.hpp
03/06/00  11:25p              3,478 DefinicionEcuacion.cpp
03/01/00  09:36p              6,231 Fractal.bpr
02/09/00  09:21p              1,395 Fractal.cpp
02/09/00  05:45p              2,862 ParamJulia.cpp
03/03/00  10:58p              3,200 ParamJulia.dfm
03/03/00  10:58p              2,242 ParamJulia.h
02/09/00  05:50p              2,894 ParamMandelbrot.cpp
03/03/00  10:58p              3,195 ParamMandelbrot.dfm
03/03/00  10:34p              2,192 ParamMandelbrot.h
03/03/00  10:45p              4,036 Principal.cpp
03/03/00  10:23p              5,801 Principal.dfm
02/09/00  09:45p              3,203 Principal.h
02/08/00  11:46p                648 SelectEcuacion.cpp
03/03/00  10:39p                678 SelectEcuacion.dfm
02/08/00  11:45p              1,073 SelectEcuacion.h
03/06/00  11:25p             26,077 VentanaJulia.cpp
02/21/00  10:08p              3,817 VentanaJulia.dfm
03/01/00  09:35p              6,756 VentanaJulia.h
03/06/00  11:25p             27,715 VentanaMandelbrot.cpp
03/03/00  11:07p              4,197 VentanaMandelbrot.dfm
03/03/00  11:07p              6,667 VentanaMandelbrot.h
02/09/00  05:51p              3,211 V_pixel.cpp
02/17/00  10:41p              5,109 V_pixel.hpp
```