

FES ACATLÁN
Graficación por Computadora Laboratorio 2
Maestra Ma. del Carmen Villar Patiño

1. Migrar el código LabMandel.c que usa graphics.h a OpenGL

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

#define MAX_COLORES 15

/* -----
,          CalcularFactoresEscala()          ,
,  Calcula el rectángulo más grande que cabe en pantalla semejan- ,
,  te a la región compleja de interés, para evitar deformaciones, ,
,  y devuelve el factor de escala apropiado.          ,
----- */

float CalcularFactoresEscala(float x0,float y0,float x1,float y1,int *x,int *y)
{
    float s,sx,sy;

    sx=(x1-x0) / *x;
    sy=(y1-y0) / *y;
    if (sx>sy)
        {
            s=sx;
            *y=(int) ( (y1-y0)/s );
        }
    else
        {
            s=sy;
            *x=(int) ( (x1-x0)/s );
        }
    return s;
} // CalcularFactoresEscala()

/* -----
,          AsignarColor()          ,
,  Asigna color a un pixel en función del número de iteracio- ,
,  nes, de forma cíclica.          ,
----- */

#define AsignarColor(i) ( (i)%MAX_COLORES )

/* -----
,          Mandel()          ,
,  Calcula el conjunto de Mandelbrot, permitiendo zoom y ,
,  coloreando según la velocidad de escape.          ,
----- */

Mandel(int iter,float x0,float y0,float x1,float y1)
{
    int xMax=640,yMax=480;          // valores máximos de la pantalla
    float x,y;          // (x,y) -> punto del plano complejo que se va obteniendo
    float x2,y2;          // punto del plano candidato a ser del conjunto de Mandelbrot
    float xx,yy;          // complejo auxiliar
    int px; register int py; // punto de pantalla que se corresponde al (x,y)
    float s;          // factor de escala para convertir de (px,py) a (x,y)
    register int iaux;          // entero auxiliar
    s=CalcularFactoresEscala(x0,y0,x1,y1,&xMax,&yMax);
    for (px=0;px<xMax;px++)          // para cada columna de pantalla
        {
```

FES ACATLÁN
Graficación por Computadora Laboratorio 2
Maestra Ma. del Carmen Villar Patiño

```
for (py=0;py<yMax;py++)          // para cada fila de pantalla
{
    // Conversion del pixel (px,py) al punto complejo (x2,y2).
    x2=x0+px*s;
    y2=y0+py*s;
    xx=yy=0.;                    // estudiamos la órbita de cero
    for(iaux=0; iaux<iter; iaux++) // comprobar si tiende a infinito
    {
        // Iterar: (x,y)=(xx,yy)ý+(x2,y2)
        x=xx*xx-yy*yy+x2;
        y=2*xx*yy+y2;
        // hacer que (x,y) sea el próximo punto iterado
        xx=x;
        yy=y;
        if (x*x+y*y>10000)      // si el módulo es mayor que 100,tiende a infinito
        {
            putpixel(px,yMax-1-py,AsignarColor(iaux));
            break; // fin del bucle iaux, no hace falta seguir
        }
    }
}
return 0;
} // Mandel()
main()
{
    int  gdriver=DETECT, gmode, errorcode; //para inicializar el modo gráfico
    float x0,y0,x1,y1;                    // región del plano complejo a representar
    int nIter;

    clrscr();
    printf("\n Conjunto de Mandelbrot \n");
    printf("\n Introduzca el numero de iteraciones: ");
    scanf("%d",&nIter);
    printf("\n Introduzca la region del plano complejo:\n");
    printf("     esquina inferior izquierda  x0 (-2.25): ");
    scanf("%f",&x0);
    printf("     esquina inferior izquierda  y0 (-1.8): ");
    scanf("%f",&y0);
    printf("     esquina superior derecha     x1 (+0.75): ");
    scanf("%f",&x1);
    printf("     esquina superior derecha     y1 (+1.5): ");
    scanf("%f",&y1);

    initgraph(&gdriver, &gmode, "c:\\tc\\bgi");
    errorcode = graphresult();
    if (errorcode != grOk)      /* an error occurred */
    {
        printf("Graphics error: %s\n", grapherrormsg(errorcode));
        printf("Press any key to halt:");
        getch();
        exit(1);                /* return with error code */
    }
    Mandel(nIter,x0,y0,x1,y1);
    getch();
    closegraph();
    return 0;
}
```