Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación

GRAFICACIÓN

TAREA 2: GRAFICA DE UNA FUNCIÓN POLAR (BRESENHAM)

Docente:
Prof. Iván Olmos Pineda

Alumno:
Jesús Huerta Aguilar

Matricula:
202041509

NRC: 10592
Sección: 001

CUARTO SEMESTRE

Puebla, Pue. Fecha de entrega: 29/08/2022

INTRODUCCIÓN

Un algoritmo preciso y efectivo para la generación de líneas de rastreo, desarrollado por Bresenham, convierte mediante rastreo las líneas al utilizar sólo cálculos incrementales con enteros que se pueden adoptar para desplegar circunferencias y otras curvas.
El algoritmo de línea de Bresenham se basa en probar el signo de un parámetro entero, cuyo valor es proporcional a la diferencia entre las separaciones de las dos posiciones de píxel de la trayectoria real de la línea.
En este trabajo se implementara el algoritmo de Bresenham para ahorrar costos computacionales al momento de imprimir una gráfica de una función polar, además de reajustar la resolución de la grafica para tener puntos pivote, los cuales serán la entrada de el algoritmo.

CONCEPTOS DESARROLLADOS

Es un algoritmo creado para dibujar rectas en los dispositivos de gráficos rasterizados, como por ejemplo un monitor de computadora, que determina que pixeles se rellenaran, en función de la inclinación del ángulo de la recta a dibujar.
Es un algoritmo preciso para la generación de líneas de rastreo que convierte mediante rastreo las líneas al utilizar solo cálculos incrementales con enteros que se pueden adaptar para desplegar circunferencias y curvas. Los ejes verticales muestran las posiciones de rastreo y los ejes horizontales identifican las columnas de píxel.

ANALISIS EMPIRICO

Implementamos las librerías para poder graficar y operar las funciones trigonométricas en diferentes apartados.

```
#ifdef __APPLE__
#include <GLUT/glut.h>
#else
#include <GL/glut.h>
#endif

#include <stdlib.h>
#include <cmath>
GLfloat sizep;
int i;
```

Función para crear el espacio virtual.
Establecemos el color de la ventana de visualización donde los tres primeros parámetros corresponden al RGB y el cuarto parámetro corresponde al valor alfa que permite el efecto de transparencia (0 = totalmente transparente y 1 = totalmente opaco)
Establecemos los parámetros de proyección ortogonal y se visualizara una proyección ortogonal de dimensiones 400x400.

```
void init(void){

    glClearColor(0.0, 0.0, 0.0, 1.0);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-200,200,-200,200);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}
```

Función algoritmo de Bresenham.
En esta función se establece el agoritmo de Bresenham el cual con ayuda de dos puntos pivote determina otros puntos enteros a imprimir con base a los puntos de entrada, si usar operaciones flotantes para así tener un ahorro computacional, dentro de este algoritmo se consideran las situaciones cuando la pendiente de los puntos es menor o mayor a 45°.

```
void Bresenham(int x0, int y0, int x1, int y1){
    int x, y, dx, dy, p, incE, incNE, stepx, stepy;
    dx = (x1 - x0);
    dy = (y1 - y0);
    if (dy < 0) {/* determinar que punto usar para empezar, cual para terminar */
        dy = -dy; stepy = -1;
      }
    else
      stepy = 1;
    if (dx < 0) {
        dx = -dx; stepx = -1;
      }
    else
      stepx = 1;
    x = x0;
    y = y0;
    glBegin(GL_POINTS);
        glVertex2i(x0,y0);
    glEnd();
      if(dx>dy){ /* se cicla hasta llegar al extremo de la línea */
        p = 2*dy - dx;
        incE = 2*dy;
        incNE = 2*(dy-dx);
        while (x != x1){
          x = x + stepx;
          if (p < 0){
            p = p + incE;
          }
          else {
            y = y + stepy;
            p = p + incNE;
          }
        glBegin(GL_POINTS);
            glVertex2i(x,y);
        glEnd();
        }
      }
      else{
        p = 2*dx - dy;
        incE = 2*dx;
        incNE = 2*(dx-dy);
        while (y != y1){
          y = y + stepy;
          if (p < 0){
            p = p + incE;
          }
          else {
            x = x + stepx;
            p = p + incNE;
          }
        glBegin(GL_POINTS);
            glVertex2i(x,y);
        glEnd();
        }
      }
 }
```

Función para dibujar la gráfica.
Definimos las variables a usar y después borramos la ventana de visualización. El primer punto debe de estar en 0° pero para calcular el diferencial tomamos la resolución la cual es de 120 y se divide sobre 360° y el resultado lo multiplicamos por PI/180, $diff = \left(\frac{360°}{120°} = 3° \Longrightarrow 3°\left(\frac{\pi rad}{180°}\right) = \frac{\pi rad}{60}\right)$,y este será el incremento que tomara el angulo en cada paso del ciclo for, además, dentro del ciclo se ejecutaran las operaciones necesarias para pasar los valores polares a cartesianos y además de asignar los colores por punto a imprimir.
Mediante un sistema de bandera, se asignaran los primeros 2 puntos para mandarlos al algoritmo de Bresenham, una vez asignados estos primeros, la bandera cambiara para asignar más fácilmente los nuevos puntos, asignando los (x_1, y_1) a (x_0, y_0) para que así, (x_1, y_1) tomen los valores de los nuevos puntos (x_t, y_t) (generados por cada iteración) para después llamar al algoritmo de Bresenham.

```
void dibujaGrafica(){
    glClear(GL_COLOR_BUFFER_BIT);  //borra la ventana de visualizacion
    //glColor3f(0.0, 0.0, 0.0);      //establece que lo que se dibuja sera rojo
    GLfloat rpolar,angulo,diff;
    GLint xt,yt,x0,y0,x1,y1,first;
    //MUESTRA GRAFICA
    bandera = 0;
    first = 1;
    glPointSize(1);
    diff = 360/120;
    for (i=0;i<240;i++){
        rpolar = 6*cos(4*rad);  //donde a = longitud del petalo y n = 4 (r =
a*cos(n*theta))
        xt = rpolar*cos(rad)*(25);
        yt = rpolar*sin(rad)*(25);

        glColor3f(0.255, 0, 0.700);

        //IMPRESION DE RECTAS
        if(first == 1){//PRIMER PUNTO
            x0 = xt;
            y0 = xt;
            x1 = xt;
            y1 = xt;
            first = 0;
        }
        else{//DEMAS PUNTOS
            x0 = x1;
            y0 = y1;

            x1 = xt;
            y1 = yt;
            if(i > 1){
                Bresenham(x0,y0,x1,y1);
            }
        }
        angulo = angulo + diff;
        rad = angulo*(M_PI/180);
    }
    //se obliga a procesar todas las instrucciones de openGL tan rapidamente como sea
posible
    glFlush();
}
```

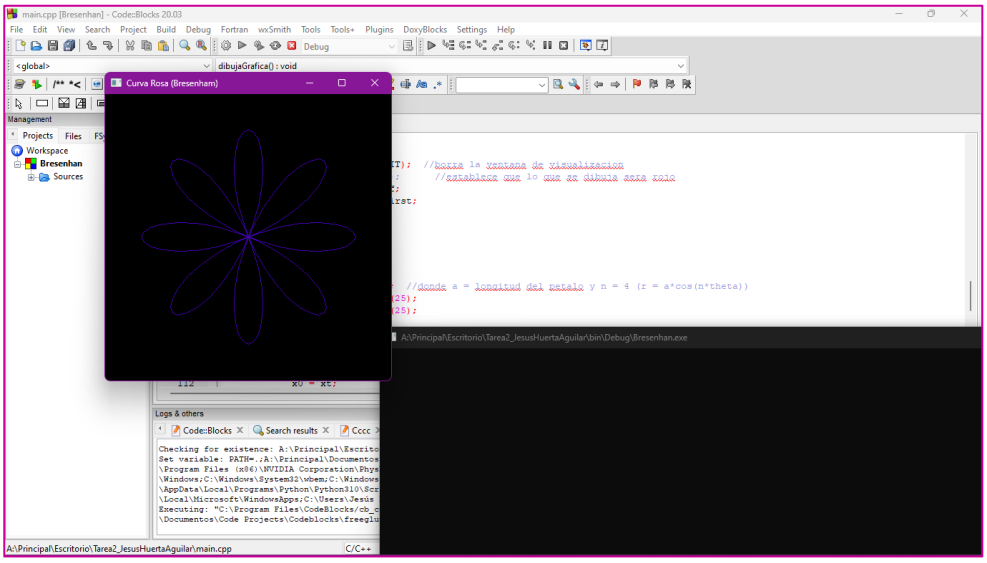
Función main.
Llamamos a las funciones e instrucciones necesarias para inicializar la pantalla grafica y establecer el modo de visualización, establecemos la posición en la esquina superior izquierda y el ancho y altura de la ventana de visualización además de crearla, ejecutamos la función de iniciación de parámetros y enviamos los gráficos a pantalla.

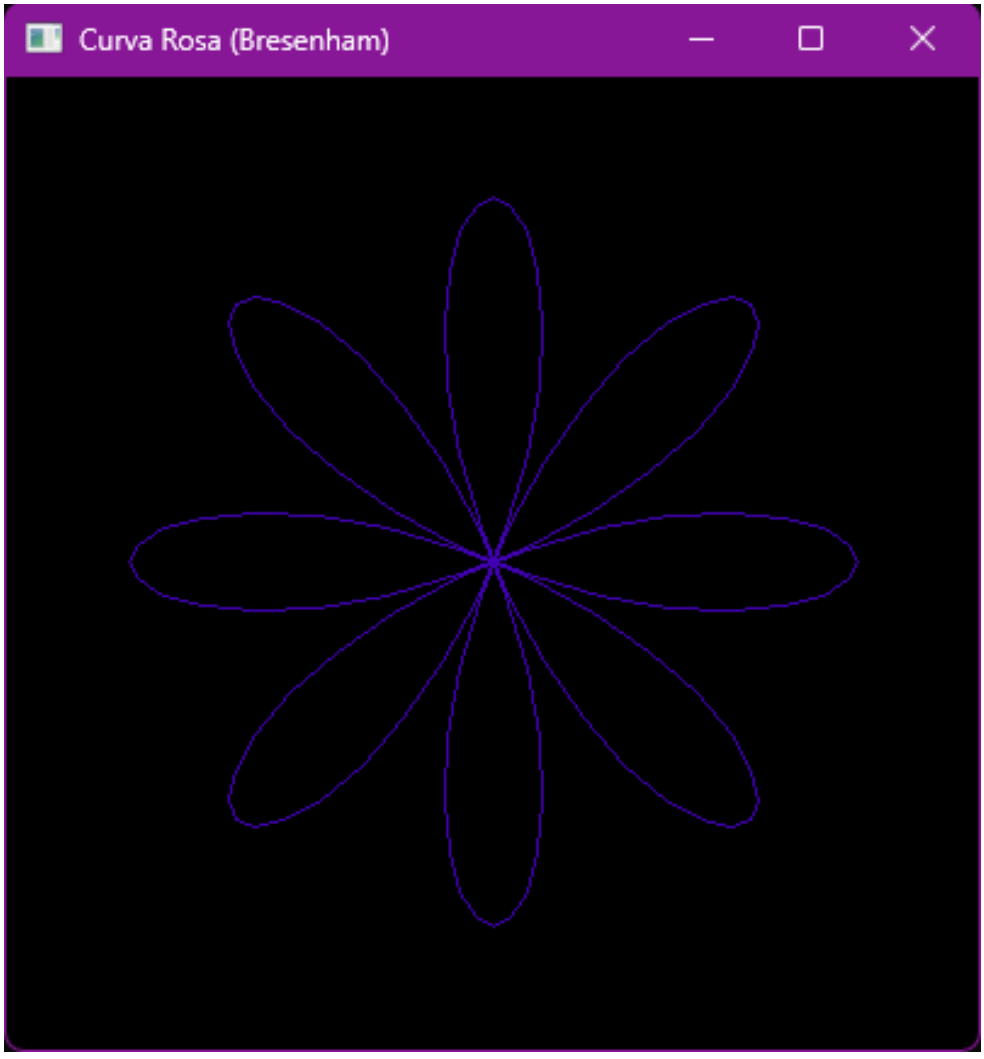
```
int main(int argc, char** argv){
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowPosition(100,100);
    glutInitWindowSize(400,400);
    glutCreateWindow("Curva Rosa (Bresenham)");
    init();
    glutDisplayFunc(dibujaGrafica);
    glutMainLoop();

    return 0;
}
```

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA
FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

EJECUCIONES





Jesús Huerta Aguilar

CONCLUSIONES

En conclusión, los pasos a seguir para la correcta implementación del algoritmo son: se almacena como primer punto el que se encuentra más a la izquierda y se pinta. Luego, se calculan los valores de incremento de x y de y, junto al parámetro de decisión, anotado p. Para cada x se comprueba la distancia genérica. Si ésta es menor o igual a cero, el siguiente punto a trazar será (x+1, y) y el parámetro de decisión del siguiente punto será igual al anterior más dos veces el incremento de y. Pero, si es mayor de cero, el siguiente punto a trazar será (x+1, y+1) y el parámetro de decisión se verá incrementado en 2dy-2dx.

BIBLIOGRAFIA

- Olmos Pineda, I. (2022b, agosto 25). Reunión en «General». sharepoint. Recuperado 28 de agosto de 2022, de https://correobuap.sharepoint.com/sites/Section_202235-CCOS261-10592-001/Shared%20Documents/Forms/AllItems.aspx?FolderCTID=0x01200042EC72C454FA1E45805A79EC6BAB0F21&id=%2Fsites%2FSection_202235-CCOS261-10592-001%2FShared%20Documents%2FGeneral%2FRecordings%2FReunión%20en%20_General_-20220825_070814-Grabación%20de%20la%20reunión%2Emp4&parent=%2Fsites%2FSection_202235-CCOS261-10592-001%2FShared%20Documents%2FGeneral%2FRecordings
- Universidad Autónoma Del Estado De Hidalgo. (s. f.). 2.3.3 Algoritmo de Punto Medio - Bresenham. cidecame. Recuperado 28 de agosto de 2022, de http://cidecame.uaeh.edu.mx/lcc/mapa/PROYECTO/libro23/233_algoritmo_de_punto_medio__bresenham.html
- Delgado, O. (2014, 11 abril). Algoritmo de Bresenham para trazar líneas. Garcia Delgado Oscar. Recuperado 28 de agosto de 2022, de http://garciaoscar10110795.blogspot.com/p/algoritmo-de-bresenham-para-trazar.html