UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO

FACULTAD DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA,INFORMÁTICA Y MECÁNICA

INGENIERÍA INFORMÁTICA Y DE SISTEMAS

Guía de Laboratorio 1 - Algoritmos para trazo de lineas

Alumno:
Ian Logan Will Quispe Ventura
211359

Docente:
Hector Eduardo Ugarte Rojas

Curso:
Computación Gráfica

Cusco - Perú
2023 - II

Funcionamiento del algoritmo DDA

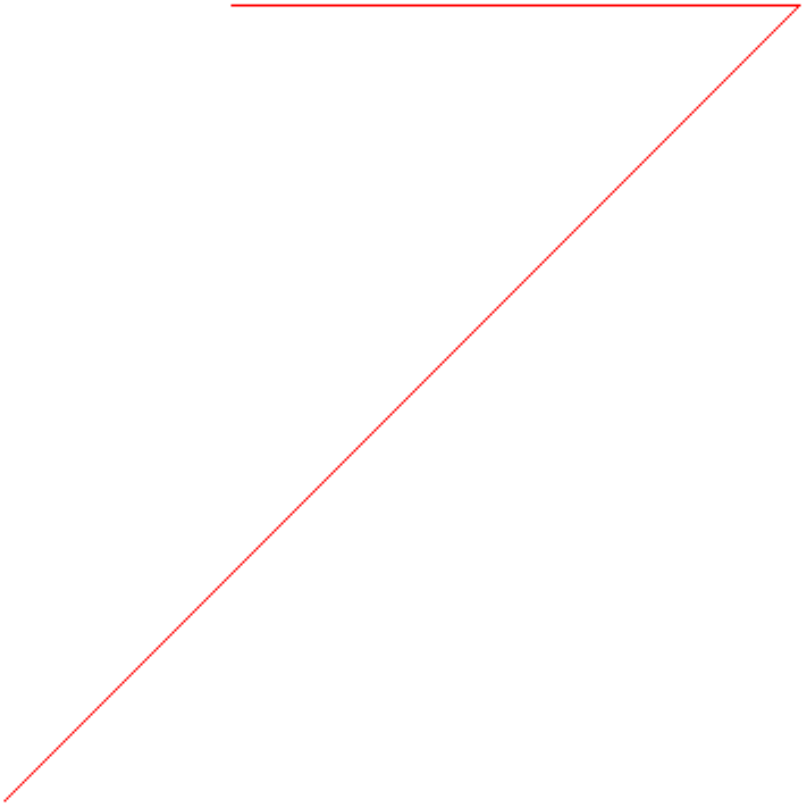
El código original impedía que ciertas lineas se dibujaran, estas lineas tenían a la pendiente mayor que 1 o menor que -1. Para solucionarlo usarmos esta versión mejorada.

```
def DDA(x0, y0, x1, y1):
    dx = x1 - x0
    dy = y1 - y0

    if abs(dx) >= abs(dy):
        steps = abs(dx)
    else:
        steps = abs(dy)

    x_increment = dx / steps
    y_increment = dy / steps

    x = x0
    y = y0

    for _ in range(steps + 1):
        plot(int(x), int(y))
        x += x_increment
        y += y_increment
```

Modulo Display para DDA

```
def display_DDA():
    glClear(GL_COLOR_BUFFER_BIT)
    glColor3f(1.0, 0.0, 0.0)
    DDA(100, 100, 450, 450)
    DDA(200, 450, 450, 450)
    glFlush()
```

Gráfico generado

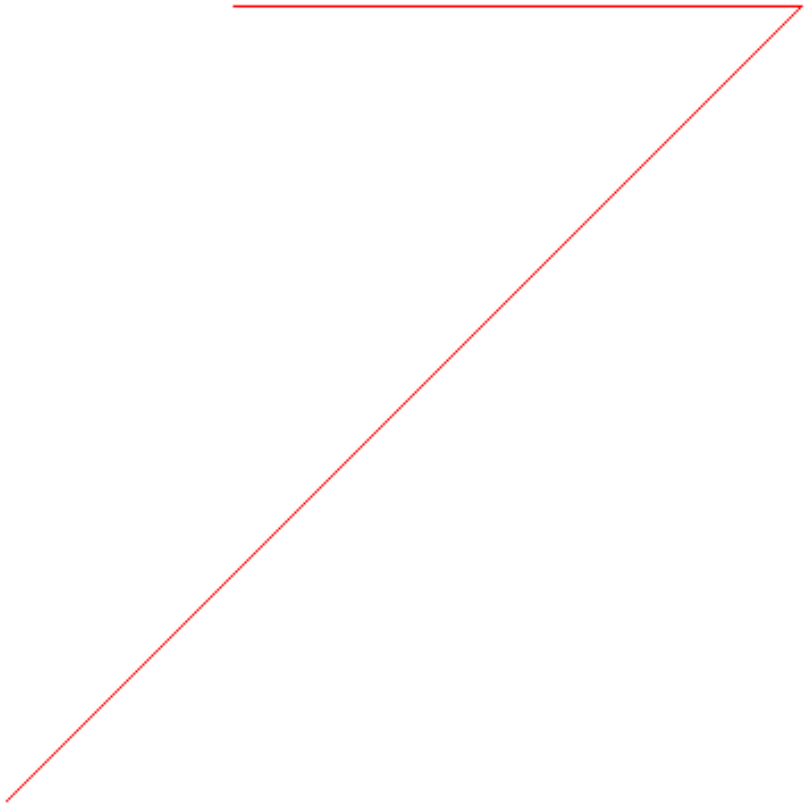


Funcionamiento del algoritmo Bresenham

Modulo Display para Bresenham

```
def display_Bresenham():
    glClear(GL_COLOR_BUFFER_BIT)
    glColor3f(1.0, 0.0, 0.0)
    Bresenham(100, 100, 450, 450)
    Bresenham(200, 450, 450, 450)
    glFlush()
```

Gráfico generado

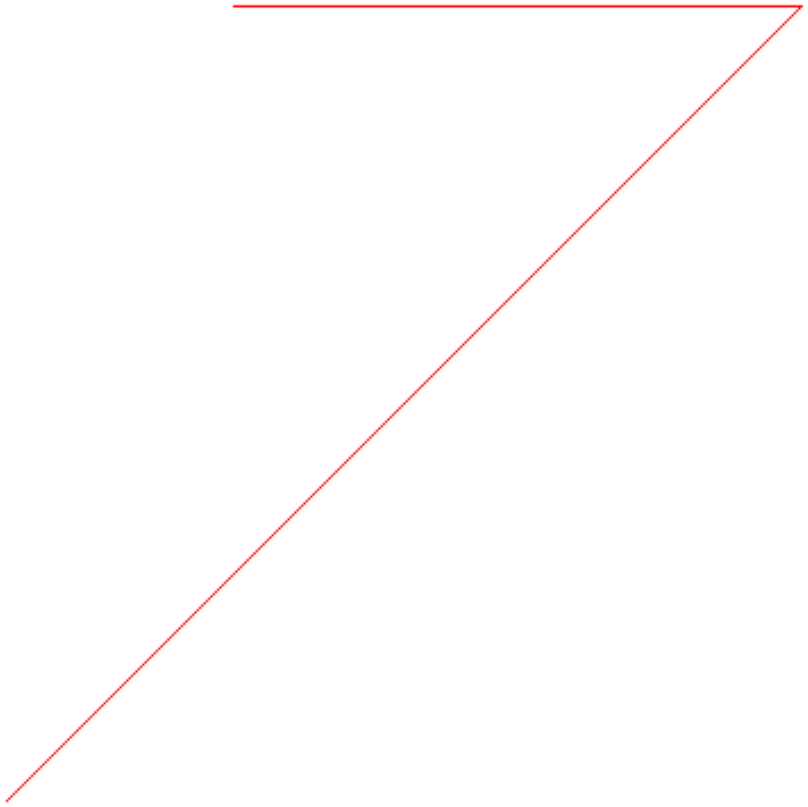


Funcionamiento del algoritmo Punto Medio

Modulo Display para Punto Medio

```
def display_PtoMedio():
    glClear(GL_COLOR_BUFFER_BIT)
    glColor3f(1.0, 0.0, 0.0)
    PtoMedio(100, 100, 450, 450)
    PtoMedio(200, 450, 450, 450)
    glFlush()
```

Gráfico generado



Algoritmo para dibujar 5 lineas y mostrar la intersección

Primero creamos una lista y guardamos en ella los puntos de las lineas.

```
# Declarar lista de todos los puntos que
   conforman las lineas
lista_puntos = []

def DDA(x0, y0, x1, y1):
    m = (y1 - y0) / (x1 - x0)
    y = y0
    for x in range(x0, x1 + 1):
        plot(x, int(y))
        # Anadir los puntos a la lista
        lista_puntos.append((x, int(y)))
        y = y + m
```

En la función display dibujaremos las 5 lineas requeridas, con valores aleatorios en los puntos.

```
def display():
    glClear(GL_COLOR_BUFFER_BIT)
    glColor3f(1.0, 0.0, 0.0)

    # Limpiar la lista de puntos
    lista_puntos.clear()

    # Dibujar 5 líneas aleatorias
    for _ in range(5):
        x0 = random.randint(10, 150)
        y0 = random.randint(10, 150)
        x1 = random.randint(250, 499)
        y1 = random.randint(250, 499)
        DDA(x0, y0, x1, y1)
```

Los puntos se guardan en una lista, en esta verificaremos los puntos repetidos que serán las intersecciones.
Dibujaremos puntos de otro color con esas coordenadas para resaltar las intersecciones.

```
# Lista para los puntos duplicado
interseccion = []

# Crear un conjunto para llevar un registro de
  los puntos únicos que hemos visto
puntos_unicos = set()

# Recorremos la lista de puntos
for punto in lista_puntos:
    if punto in puntos_unicos:
        # Si los puntos se encuentran en el
          conjunto, se añaden a la lista
          intersección. Inicialmente no hay
          elementos en el conjunto
        interseccion.append(punto)
    else:
        # Si no se encuentra el punto en el
          conjunto, entonces este se añade al
          mismo conjunto que se utilicará en la
          próxima iteración
        puntos_unicos.add(punto)
print("Puntos de intersección de las lineas:",
  interseccion)

# Pintar de negro los puntos de intersección
glColor3f(0.0, 0.0, 0.0)
glPointSize(3)

# Recorrer la lista de duplicados para dibujarlos
for punto in interseccion:
    x, y = punto
    plot(x, y)
glPointSize(1)
glFlush()
```

Capturas de la ventana de gráficos