UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO

FACULTAD DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA,INFORMÁTICA Y MECÁNICA

INGENIERÍA INFORMÁTICA Y DE SISTEMAS

Guía de Laboratorio 2 - Algoritmos de Generación de Circunferencias

Alumno:
Ian Logan Will Quispe Ventura
211359

Docente:
Hector Eduardo Ugarte Rojas

Curso:
Computación Gráfica

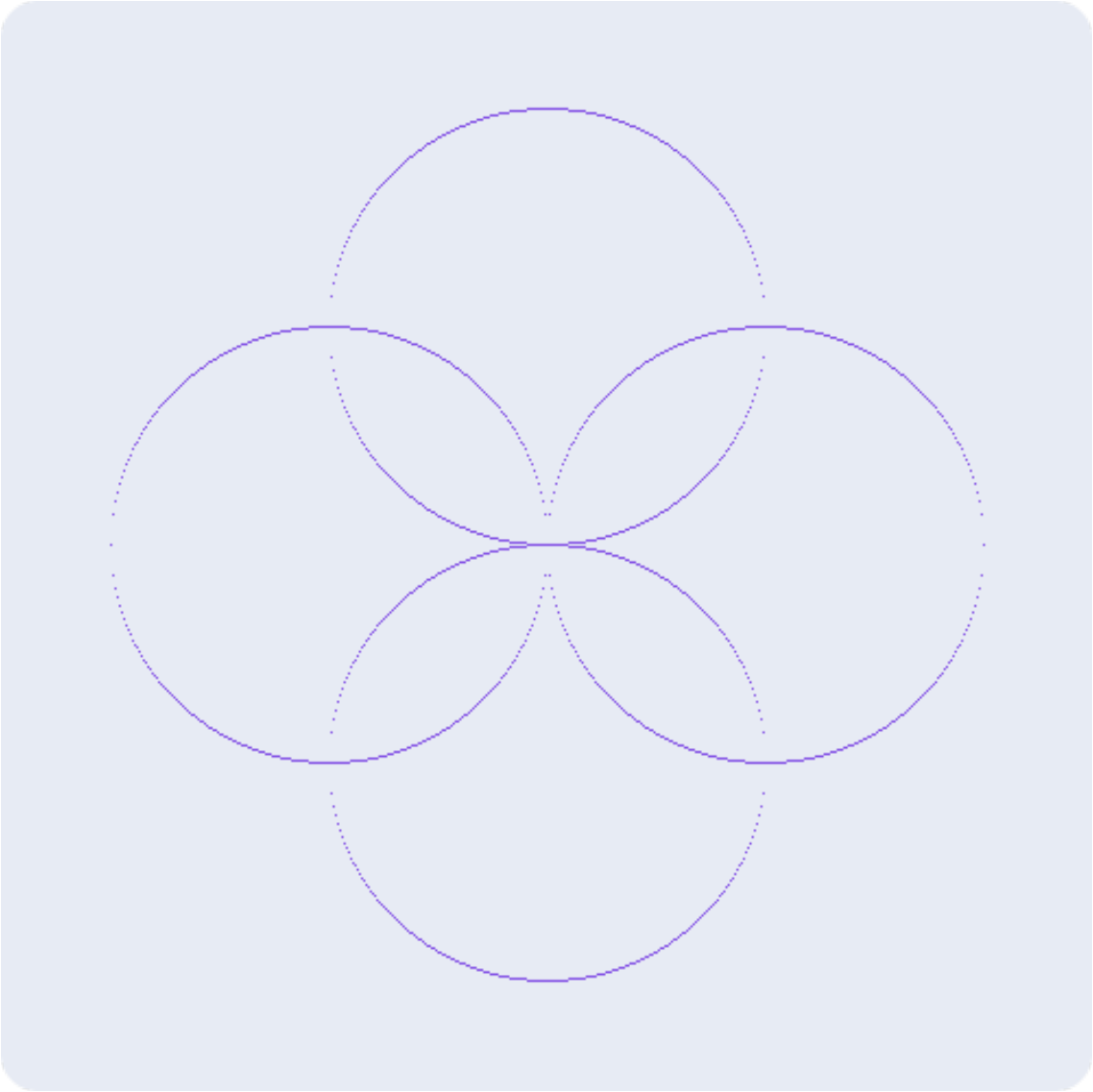
Cusco - Perú
2023 - II

Funcionamiento del algoritmo de 2 Vías

Modulo Display

```
def display_2vias():
    glClear(GL_COLOR_BUFFER_BIT)
    glPointSize (1)
    glColor3f(0.5, 0.3, 0.9)
    circulo2vias(150, 250, 100)
    circulo2vias(350, 250, 100)
    circulo2vias(250, 350, 100)
    circulo2vias(250, 150, 100)
    glFlush()
```

Gráfico generado

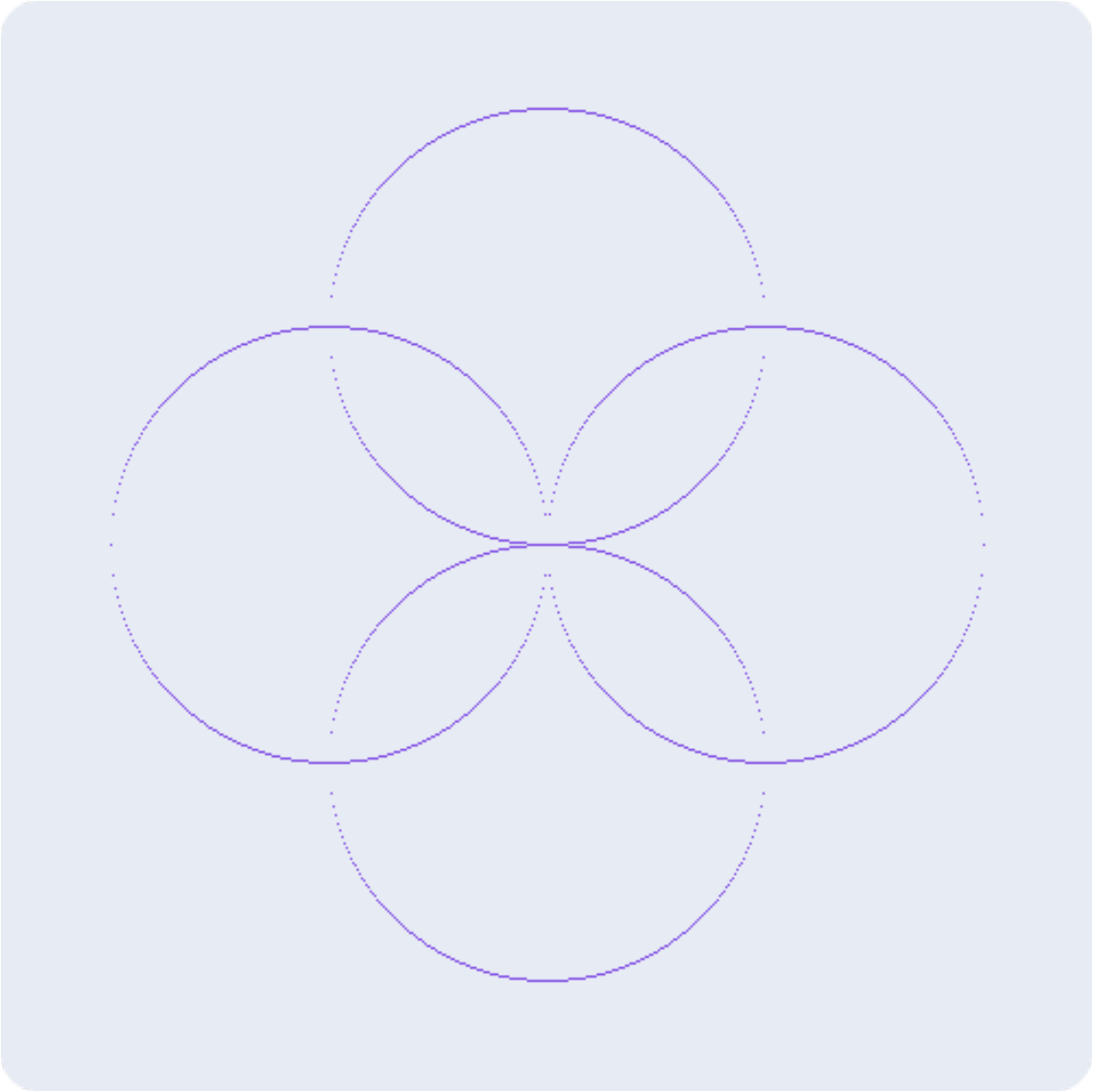


Funcionamiento del algoritmo de 4 Vías

Modulo Display

```
def display_4vias():
    glClear(GL_COLOR_BUFFER_BIT)
    glPointSize (1)
    glColor3f(0.5, 0.3, 0.9)
    circulo4vias(150, 250, 100)
    circulo4vias(350, 250, 100)
    circulo4vias(250, 350, 100)
    circulo4vias(250, 150, 100)
    glFlush()
```

Gráfico generado

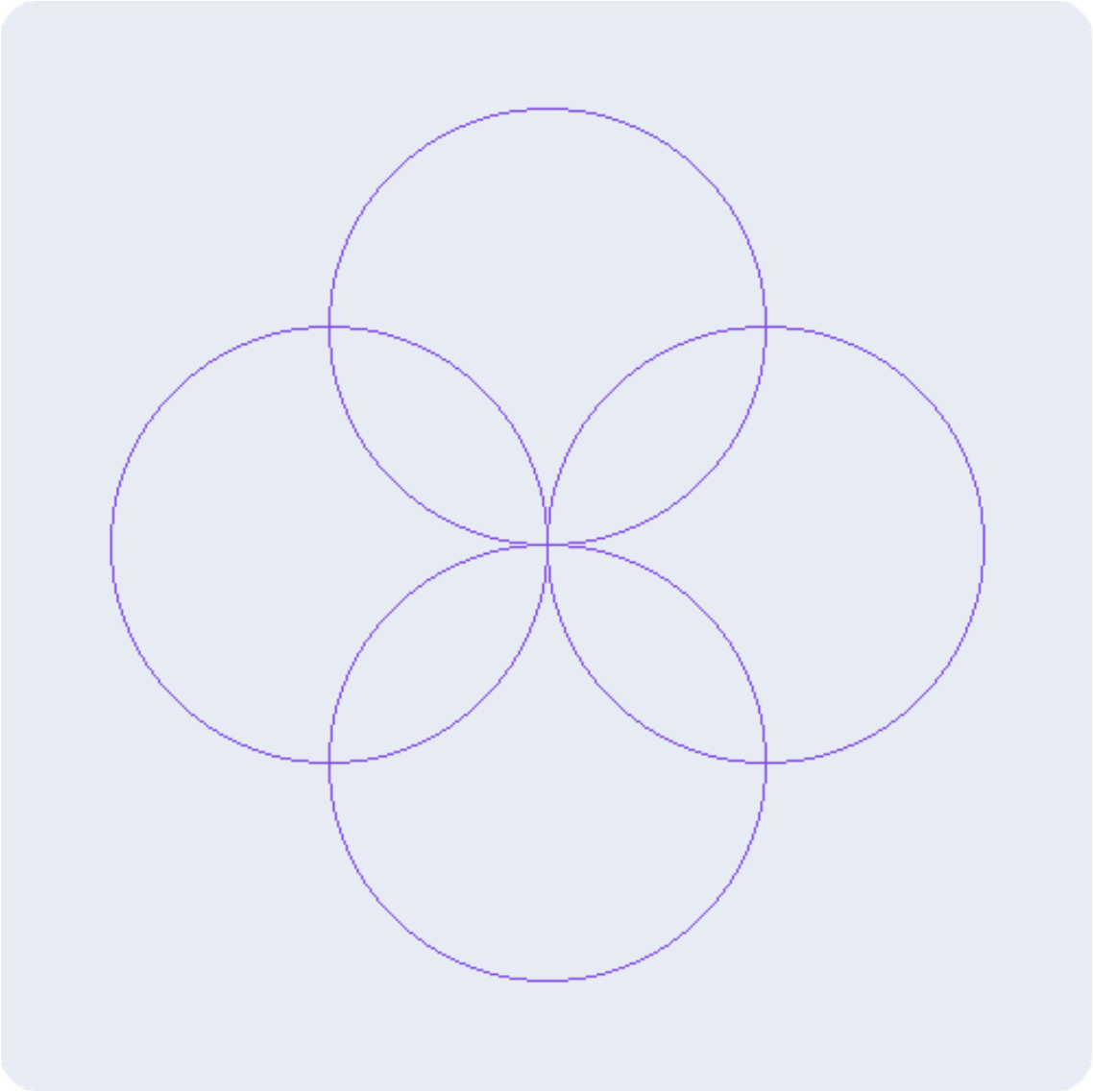


Funcionamiento del algoritmo de 8 vías

Modulo Display

```
def display_8vias():
    glClear(GL_COLOR_BUFFER_BIT)
    glPointSize (1)
    glColor3f(0.5, 0.3, 0.9)
    circulo8vias(150, 250, 100)
    circulo8vias(350, 250, 100)
    circulo8vias(250, 350, 100)
    circulo8vias(250, 150, 100)
    glFlush()
```

Gráfico generado

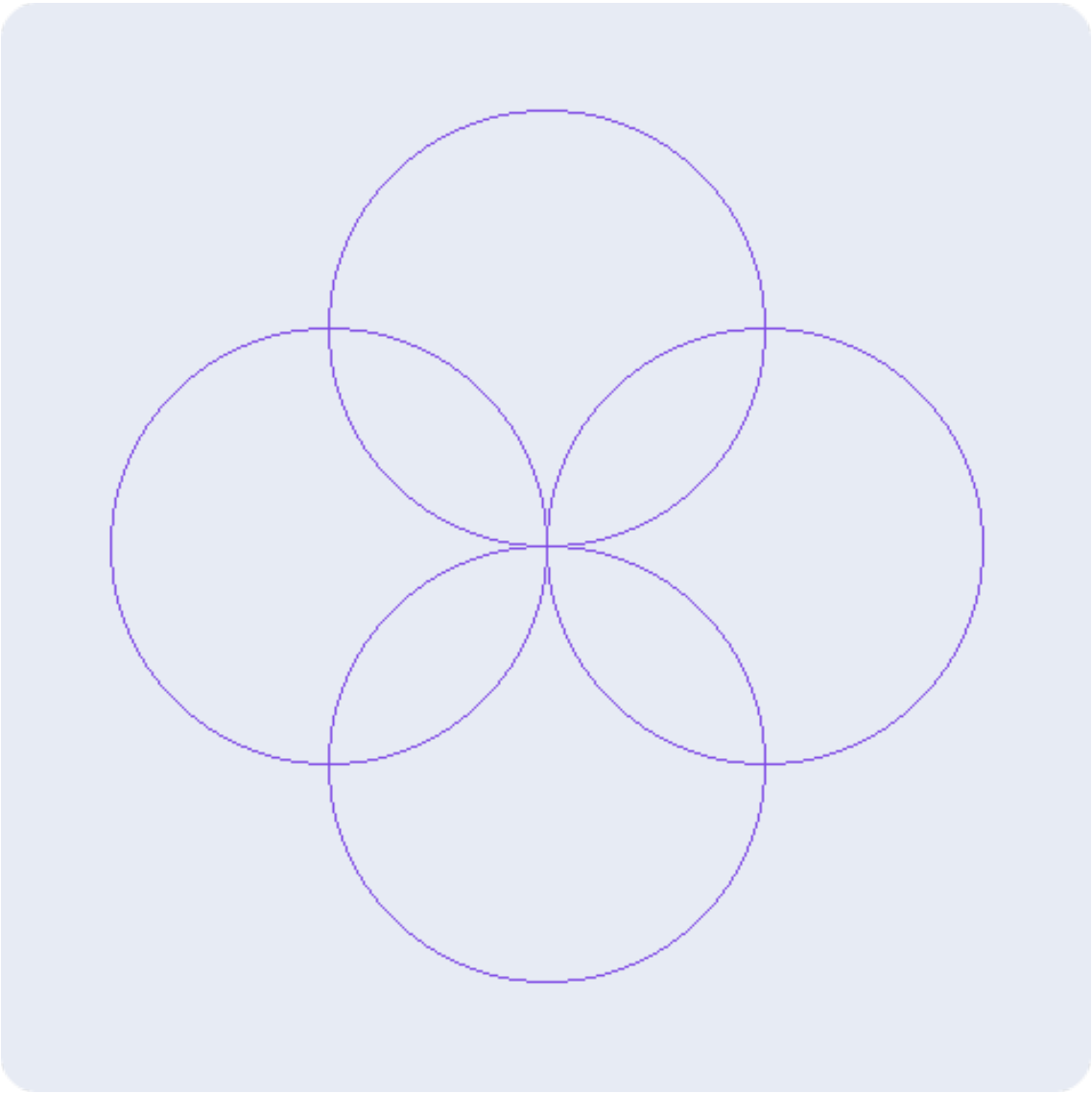


Funcionamiento del algoritmo de Punto Medio

Modulo Display

```
def display_PtoMedio():
    glClear(GL_COLOR_BUFFER_BIT)
    glPointSize (1)
    glColor3f(0.5, 0.3, 0.9)
    circuloPtoMEdio(150, 250, 100)
    circuloPtoMEdio(350, 250, 100)
    circuloPtoMEdio(250, 350, 100)
    circuloPtoMEdio(250, 150, 100)
    glFlush()
```

Gráfico generado



Funcionamiento del algoritmo de trazados de lineas y círculo

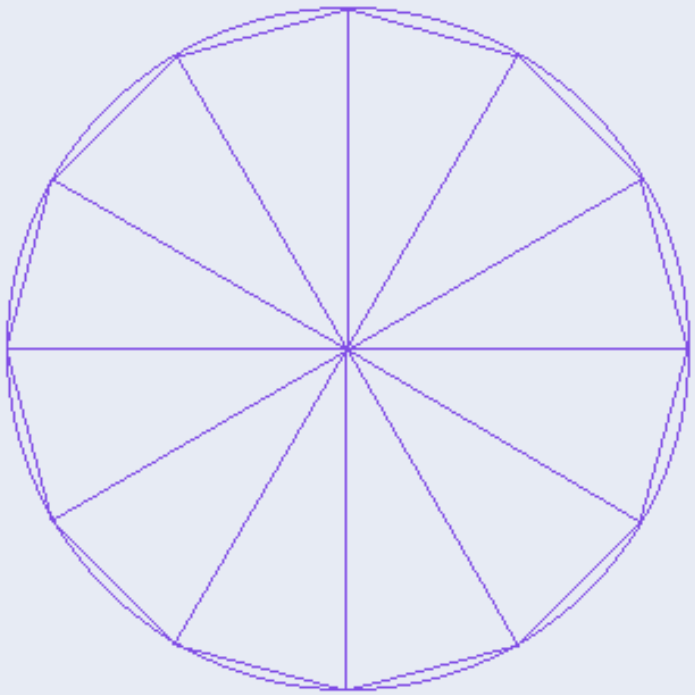
Necesitamos un módulo que dibuje y ejecute las lineas mediante el módulo DDA

```
def lineas(x0, y0, r, n_puntos):
    # Calcularemos el ángulo inicial de la primera
      linea
    angulo_inicial = 360 / n_puntos
    puntos_interseccion = []

    for i in range(n_puntos):
        # El ángulo incrementará a cada linea que se
          quiera dibujar
        angulo = i * angulo_inicial
        # Calcularemos los puntos que intersectan con
           la circunferencia
        angulo_radianes = math.radians(angulo)
        x = x0 + r * math.cos(angulo_radianes)
        y = y0 + r * math.sin(angulo_radianes)

        #Dibujar las lineas desde el origen hasta un
          punto de la circunferencia
        DDA(x0, y0, int(x), int(y))
        # Los puntos se duardan para luego unirlos
          formando un polígono
        puntos_interseccion.append((x, y))

    for a in range(n_puntos):
        # Conenctar las anteriores lineas para formar
           un polígono regular (dodecágono)
        # (a + 1) % n_puntos Permite unir el punto
          final con el inicial
        DDA(int(puntos_interseccion[a][0]), int(
          puntos_interseccion[a][1]), int(
          puntos_interseccion[(a + 1) % n_puntos][0])
          , int(puntos_interseccion[(a + 1) %
          n_puntos][1]))
```

Módulo display que ejecutará el algoritmo Punto Medio y el de las lineas

```
def display():
    glClear(GL_COLOR_BUFFER_BIT)
    circuloPtoMEdio(250, 250, 150)
    glColor3f(0.5, 0.3, 0.9)
    lineas(250, 250, 150, 12)
    glFlush()
```

Gráfico generado



Algoritmo para encontrar las intersecciones de 5 círculos con centros y radios aleatorios

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

```
# Declarar lista de todos los puntos que conforman
  las lineas
puntos8=[]
def circulo8vias(x0, y0, r):
    plot (x0, y0 + r)
    puntos8.append((x0, y0 + r))
    plot (x0, y0 - r)
    puntos8.append((x0, y0 - r))
    plot (x0 + r, y0)
    puntos8.append((x0 + r, y0))
    plot (x0 - r, y0)
    puntos8.append((x0 - r, y0))
    x=1
    y = math.floor(math.sqrt(r * r - x * x) + 0.5)
    while x < y:
        plot(x0 + x, y0 + y)
        puntos8.append((x0 + x, y0 + y))
        plot(x0 + x, y0 - y)
        puntos8.append((x0 + x, y0 - y))
        plot(x0 - x, y0 + y)
        puntos8.append((x0 - x, y0 + y))
        plot(x0 - x, y0 - y)
        puntos8.append((x0 - x, y0 - y))
        plot(x0 + y, y0 + x)
        puntos8.append((x0 + y, y0 + x))
        plot(x0 + y, y0 - x)
        puntos8.append((x0 + y, y0 - x))
        plot(x0 - y, y0 + x)
        puntos8.append((x0 - y, y0 + x))
        plot(x0 - y, y0 - x)
        puntos8.append((x0 - y, y0 - x))
        x = x + 1
        y = math.floor(math.sqrt(r * r - x * x) +
          0.5)
```

```
    if x == y:
        plot(x0 + x, y0 + y)
        puntos8.append((x0 + x, y0 + y))
        plot(x0 + x, y0 - y)
        puntos8.append((x0 + x, y0 - y))
        plot(x0 - x, y0 + y)
        puntos8.append((x0 - x, y0 + y))
        plot(x0 - x, y0 - y)
        puntos8.append((x0 - x, y0 - y))
```

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

```
def display():
    glClear(GL_COLOR_BUFFER_BIT)
    glPointSize (1)
    puntos8.clear()
    glColor3f(0.5, 0.3, 0.9)

    for _ in range(5):
        r = random.randint(50, 100)
        # Para asegurar que los círculos se
          dibujen dentro de los límites de la
           ventana
        x0 = random.randint(r, 499-r)
        y0 = random.randint(r, 499-r)
        circulo8vias(x0, y0, r)
```

Verificaremos los puntos repetidos, en la lista punros8, que serán las intersecciones. Dibujaremos puntos de otro color con esas coordenadas para resaltar las intersecciones.

```
    # Lista para los puntos duplicado (intersección)
    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 puntos8:
        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 otro color los puntos de intersección
    glColor3f(0.05, 0.08, 0.1)
    glPointSize(4)

    # 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