UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO

FACULTAD DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA,INFORMÁTICA Y MECÁNICA

INGENIERÍA INFORMÁTICA Y DE SISTEMAS

Guía de Laboratorio 4 - Traslación

Alumno:
Ian Logan Will Quispe Ventura
211359

Docente:
Hector Eduardo Ugarte Rojas

Curso:
Computación Gráfica

Cusco - Perú
2023 - II

Algoritmo de Traslación de una Linea en Python

```
from OpenGL.GL import *
from OpenGL.GLUT import *
from OpenGL.GLU import *

def trasladaLinea(P, T):
    # Dibuja la línea original
    glColor3f(0.8, 0.26, 1.0)
    glBegin(GL_LINES)
    glVertex2f(P[0][0], P[0][1])
    glVertex2f(P[1][0], P[1][1])
    glEnd()
    # Calcula las coordenadas de traslación
    P[0][0] += T[0]
    P[0][1] += T[1]
    P[1][0] += T[0]
    P[1][1] += T[1]
    glColor3f (0.5 , 0.3 , 0.9)
    # Dibuja la línea trasladada
    glBegin(GL_LINES)
    glVertex2f(P[0][0], P[0][1])
    glVertex2f(P[1][0], P[1][1])
    glEnd()
    glFlush()

def display():
    P = [[150, 80], [320, 380]]
    T = [100, 60]
    glClear(GL_COLOR_BUFFER_BIT)
    trasladaLinea(P, T)
    glFlush()

def myinit():
    glClearColor (0.9, 0.92, 0.95, 1.0)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    gluOrtho2D(0.0, 499.0, 0.0, 499.0)
```

```
# Función principal
def main():
    # Inicialización de GLUT estándar
    glutInit(sys.argv)
    glutInitDisplayMode(GLUT_SINGLE |
      GLUT_RGB)
    glutInitWindowSize(500, 500)
    glutInitWindowPosition(0, 0)
    glutCreateWindow("Traslación de líneas")
    glutDisplayFunc(display)
    myinit()
    glutMainLoop()

# Llama a la función principal
if __name__ == "__main__":
    main()
```

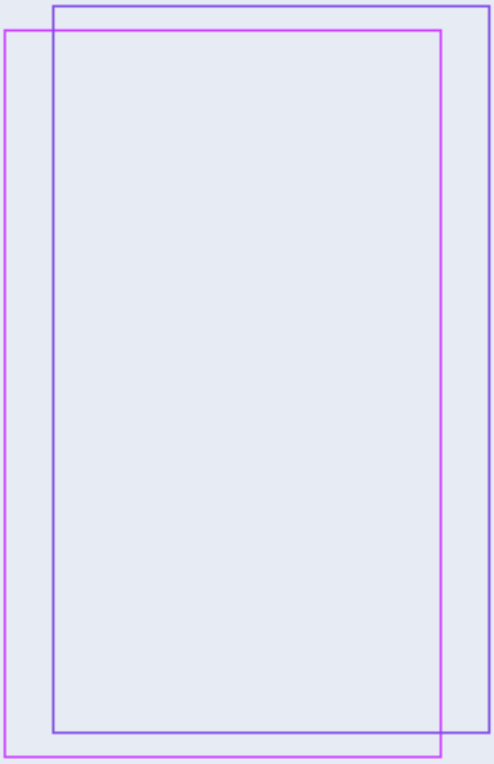
Gráfico generado

Algoritmo de Traslación de un Rectángulo en Python

```
from OpenGL.GL import *
from OpenGL.GLUT import *
from OpenGL.GLU import *

def trasladaRectangulo(P, T):
    glColor3f(0.8, 0.26, 1.0)
    glBegin(GL_LINES)
    # -- IZQUIERDA
    glVertex2f(P[0][0], P[0][1])
    glVertex2f(P[0][0], P[1][1])
    # -- ARRIBA
    glVertex2f(P[0][0], P[1][1])
    glVertex2f(P[1][0], P[1][1])
    # -- DERECHA
    glVertex2f(P[1][0], P[1][1])
    glVertex2f(P[1][0], P[0][1])
    # -- ABAJO
    glVertex2f(P[1][0], P[0][1])
    glVertex2f(P[0][0], P[0][1])
    glEnd()

    # Calcula las coordenadas de traslación
    P[0][0] += T[0]
    P[0][1] += T[1]
    P[1][0] += T[0]
    P[1][1] += T[1]

    glColor3f (0.5 , 0.3 , 0.9)
    # Dibuja el rectángulo trasladado
    glBegin(GL_LINES)
    glVertex2f(P[0][0], P[0][1])
    glVertex2f(P[0][0], P[1][1])
    # -- ARRIBA
    glVertex2f(P[0][0], P[1][1])
    glVertex2f(P[1][0], P[1][1])
```

```
    # -- DERECHA
    glVertex2f(P[1][0], P[1][1])
    glVertex2f(P[1][0], P[0][1])
    # -- ABAJO
    glVertex2f(P[1][0], P[0][1])
    glVertex2f(P[0][0], P[0][1])
    glEnd()
```

Gráfico generado

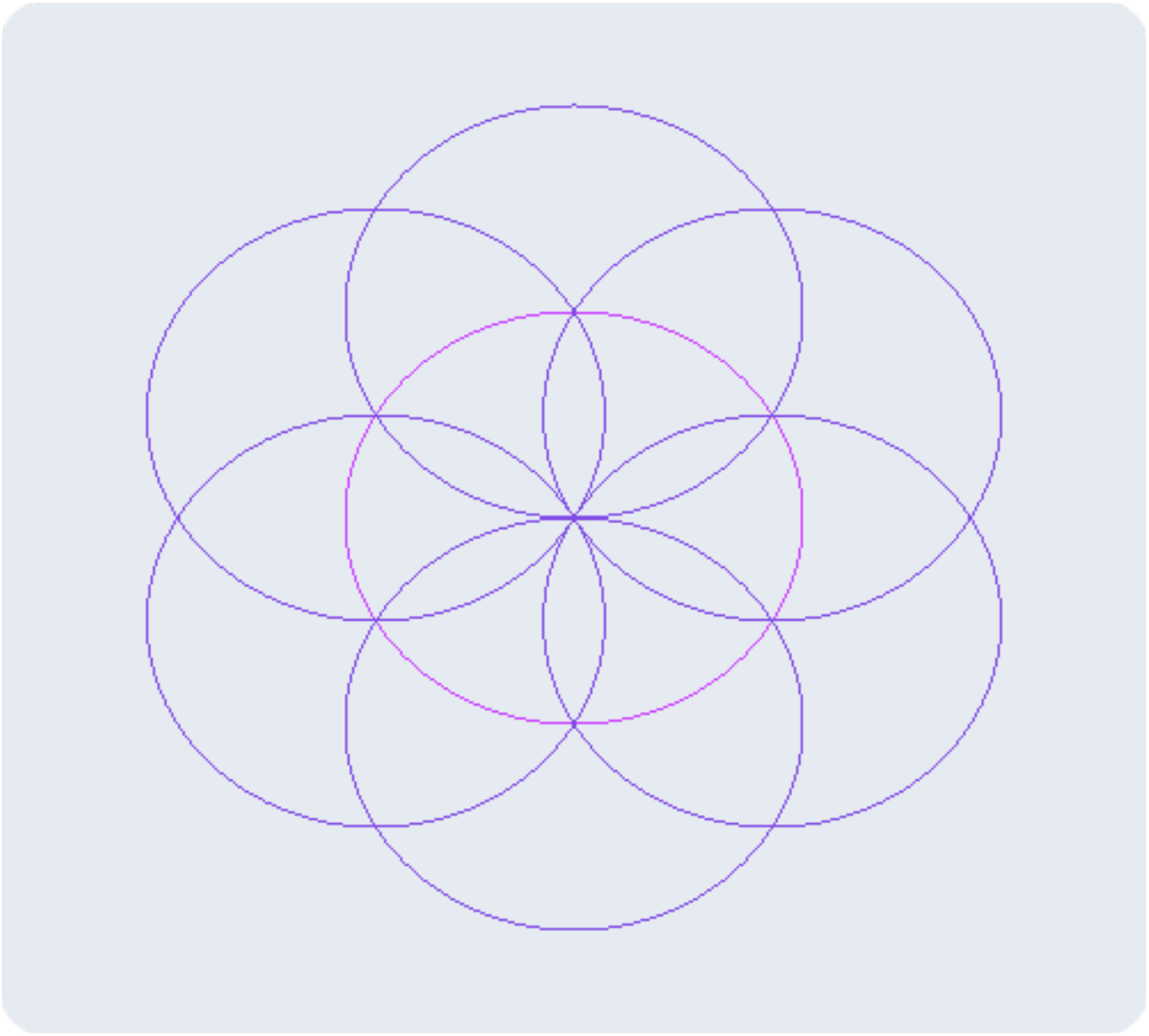


Algoritmo de Traslación de un circulo en Python

```
from OpenGL.GL import *
from OpenGL.GLUT import *
from OpenGL.GLU import *
import math

# Configuración inicial
width, height = 500, 450
r = 0.4
change = 0
p = [[0, 0] for _ in range(6)]

# Función para dibujar círculos
def draw(tx, ty):
    global r
    glBegin(GL_LINE_LOOP)
    for i in range(1200):
        theta = 2 * math.pi * i / 1200
        x1 = r * math.cos(theta) + tx
        y1 = r * math.sin(theta) + ty
        glVertex2f(x1, y1)
    glEnd()

# Función de visualización
def display():
    global width, height, r, change, p
    glClear(GL_COLOR_BUFFER_BIT |
      GL_DEPTH_BUFFER_BIT)
    glClearColor(0.9, 0.92, 0.95, 1.0)
    glColor3f(0.8, 0.26, 1.0)
    glMatrixMode(GL_MODELVIEW)
    j = 0
    change = 1 - change
    glBegin(GL_LINE_LOOP)
    for i in range(1200):
        theta = 2 * math.pi * i / 1200
        x1 = r * math.cos(theta)
        y1 = r * math.sin(theta)
```

```
            glVertex2f(x1, y1)
            if i in [100, 300, 500, 700, 900,
              1100] and change == 0:
                p[j] = [x1, y1]
                j += 1
    glEnd()
    if change == 0:
        glColor3f(0.5, 0.3, 0.9)
        for i in range(6):
            draw(p[i][0], p[i][1])
    glutSwapBuffers()

# Función principal
def main():
    glutInit()
    glutInitDisplayMode(GLUT_DOUBLE |
      GLUT_RGB | GLUT_DEPTH)
    glutInitWindowSize(width, height)
    glutCreateWindow("circles".encode("ascii"
      ))
    glutDisplayFunc(display)
    glutIdleFunc(display)
    glutMainLoop()

main()
```

Gráfico generado



Algoritmo de Traslación del Dibujo E

```
from OpenGL.GL import *
from OpenGL.GLU import *
from OpenGL.GLUT import *
# Matriz de coordenadas del dibujo original
matriz_origen = [[100, 300, 200, 300, 200, 280, 100,
  280],[100, 290, 130, 290, 130, 180, 100, 180],[100,
   250, 160, 250, 160, 230, 100, 230],[100, 200, 200,
   200, 200, 180, 100, 180]]
# Los componentes de esta matriz se sumarán a las
  cordenadas de la matriz de origen
matriz_traslacion = [200,0]

def dibujar_E():
    glBegin(GL_QUADS)
    glColor3f(0.8, 0.26, 1.0)
    glVertex2f(matriz_origen[0][0],matriz_origen
      [0][1])
    glVertex2f(matriz_origen[0][2],matriz_origen
      [0][3])
    glVertex2f(matriz_origen[0][4],matriz_origen
      [0][5])
    glVertex2f(matriz_origen[0][6],matriz_origen
      [0][7])

    glVertex2f(matriz_origen[1][0],matriz_origen
      [1][1])
    glVertex2f(matriz_origen[1][2],matriz_origen
      [1][3])
    glVertex2f(matriz_origen[1][4],matriz_origen
      [1][5])
    glVertex2f(matriz_origen[1][6],matriz_origen
      [1][7])
```

```
    glVertex2f(matriz_origen[2][0],matriz_origen
      [2][1])
    glVertex2f(matriz_origen[2][2],matriz_origen
      [2][3])
    glVertex2f(matriz_origen[2][4],matriz_origen
      [2][5])
    glVertex2f(matriz_origen[2][6],matriz_origen
      [2][7])

    glVertex2f(matriz_origen[3][0],matriz_origen
      [3][1])
    glVertex2f(matriz_origen[3][2],matriz_origen
      [3][3])
    glVertex2f(matriz_origen[3][4],matriz_origen
      [3][5])
    glVertex2f(matriz_origen[3][6],matriz_origen
      [3][7])
    glEnd()

def dibujar_E_trasladado():
    glBegin(GL_QUADS)
    glColor3f (0.5 , 0.3 , 0.9)
    glVertex2f(matriz_origen[0][0] +
      matriz_traslacion[0],matriz_origen[0][1] +
      matriz_traslacion[1])
    glVertex2f(matriz_origen[0][2] +
      matriz_traslacion[0],matriz_origen[0][3] +
      matriz_traslacion[1])
    glVertex2f(matriz_origen[0][4] +
      matriz_traslacion[0],matriz_origen[0][5] +
      matriz_traslacion[1])
    glVertex2f(matriz_origen[0][6] +
      matriz_traslacion[0],matriz_origen[0][7] +
      matriz_traslacion[1])
```

```
glVertex2f(matriz_origen[1][0] +
  matriz_traslacion[0],matriz_origen[1][1] +
  matriz_traslacion[1])
glVertex2f(matriz_origen[1][2] +
  matriz_traslacion[0],matriz_origen[1][3] +
  matriz_traslacion[1])
glVertex2f(matriz_origen[1][4] +
  matriz_traslacion[0],matriz_origen[1][5] +
  matriz_traslacion[1])
glVertex2f(matriz_origen[1][6] +
  matriz_traslacion[0],matriz_origen[1][7] +
  matriz_traslacion[1])

glVertex2f(matriz_origen[2][0] +
  matriz_traslacion[0],matriz_origen[2][1] +
  matriz_traslacion[1])
glVertex2f(matriz_origen[2][2] +
  matriz_traslacion[0],matriz_origen[2][3] +
  matriz_traslacion[1])
glVertex2f(matriz_origen[2][4] +
  matriz_traslacion[0],matriz_origen[2][5] +
  matriz_traslacion[1])
glVertex2f(matriz_origen[2][6] +
  matriz_traslacion[0],matriz_origen[2][7] +
  matriz_traslacion[1])

glVertex2f(matriz_origen[3][0] +
  matriz_traslacion[0],matriz_origen[3][1] +
  matriz_traslacion[1])
glVertex2f(matriz_origen[3][2] +
  matriz_traslacion[0],matriz_origen[3][3] +
  matriz_traslacion[1])
glVertex2f(matriz_origen[3][4] +
  matriz_traslacion[0],matriz_origen[3][5] +
  matriz_traslacion[1])
glVertex2f(matriz_origen[3][6] +
  matriz_traslacion[0],matriz_origen[3][7] +
  matriz_traslacion[1])
glEnd()
```

Gráfico generado