UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO

FACULTAD DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA,INFORMÁTICA Y MECÁNICA

INGENIERÍA INFORMÁTICA Y DE SISTEMAS

Guía de Laboratorio 5 - Rotación

Alumno:
Ian Logan Will Quispe Ventura
211359

Docente:
Hector Eduardo Ugarte Rojas

Curso:
Computación Gráfica

Cusco - Perú
2023 - II

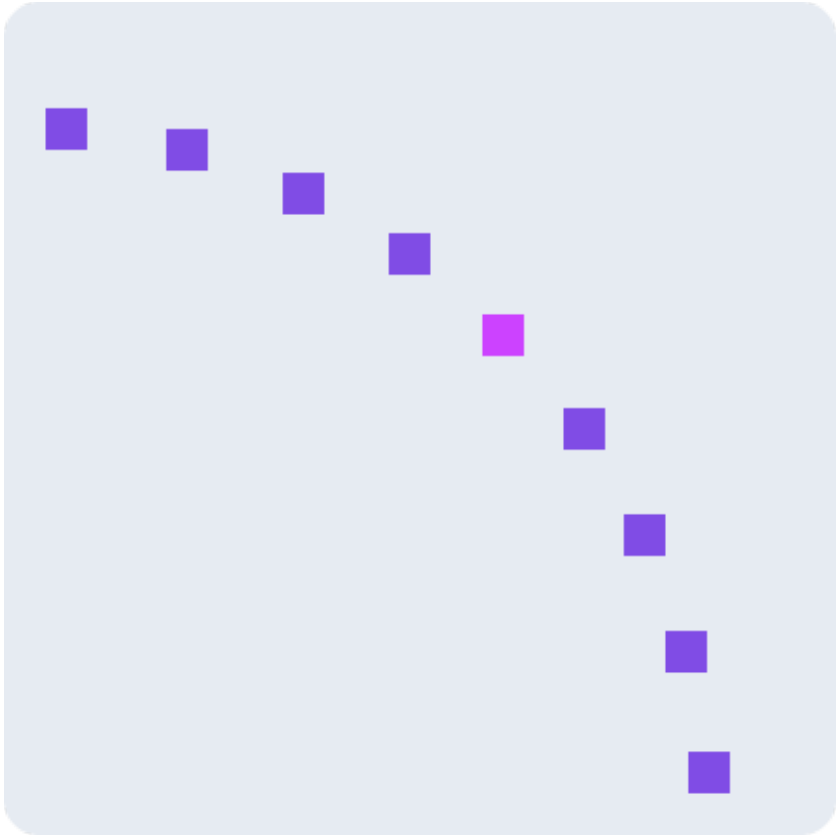
Para todos los 3 ejercicios de rotación que presento en este informe se usó el método de multiplicar la matriz de los puntos de origen con una matriz de rotación.

Algoritmo de Rotación de un Punto

```
from OpenGL.GL import *
from OpenGL.GLUT import *
from OpenGL.GLU import *
import numpy as np
import math

def punto():
    glColor3f(0.8, 0.26, 1.0)
    glPointSize(20)
    glBegin(GL_POINTS)
    glVertex2f(300, 300) # Coordenadas del punto
    glEnd()

def punto_rotacion(angulo):
    glColor3f (0.5 , 0.3 , 0.9)
    # Matriz de senos y cosenos
    matriz_a = np.array([[math.cos(math.radians(
      angulo)), -math.sin(math.radians(angulo))], [
      math.sin(math.radians(angulo)), math.cos(math.
      radians(angulo))]])
    # Matriz de punto inicial
    matriz_b = np.array([[300], [300]])
    resultado = np.dot(matriz_a, matriz_b) # Producto
       de matrices
    glBegin(GL_POINTS)
    glVertex2f(resultado[0], resultado[1])  #
      Coordenadas del punto rotado
    glEnd()
def display():
    glClear(GL_COLOR_BUFFER_BIT)
    punto()
    punto_rotacion(-40)
    punto_rotacion(-30)
    punto_rotacion(-20)
    punto_rotacion(-10)
```

```
    punto_rotacion(10)
    punto_rotacion(20)
    punto_rotacion(30)
    punto_rotacion(40)
    glutSwapBuffers()

def main():
    glutInit(sys.argv)
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB)
    glutInitWindowSize(400, 400)
    glutInitWindowPosition(100, 100)
    glutCreateWindow("Dibujar Punto rotado")
    glClearColor (0.9 ,0.92 , 0.95 , 1.0)
    gluOrtho2D(0.0, 499.0, 0.0, 499.0)
    glutDisplayFunc(display)
    glutMainLoop()

if __name__ == "__main__":
    main()
```

Gráfico generado



Algoritmo de Rotación de una Linea

```
from OpenGL.GL import *
from OpenGL.GLUT import *
from OpenGL.GLU import *
import numpy as np
import math

# Variables globales para los puntos
x1 = 350
y1 = 350
x2 = 50
y2 = 50
def punto():
    glColor3f(0.8, 0.26, 1.0)
    glLineWidth(3)
    glBegin(GL_LINES)
    glVertex2f(x1, y1)  # Coordenadas del punto
      inicial
    glVertex2f(x2, y2)  # Coordenadas del punto final
    glEnd()
def punto_rotacion(x1, y1,x2, y2, angulo):
    glColor3f(0.5, 0.3, 0.9)
    matriz_a = np.array([[math.cos(math.radians(
      angulo)), -math.sin(math.radians(angulo))],[
      math.sin(math.radians(angulo)), math.cos(math.
      radians(angulo))]])
    matriz_b = np.array([[x1], [y1]])
    matriz_c = np.array([[x2], [y2]])
    # Multiplicamos las matrices
    resultado1 = np.dot(matriz_a, matriz_b)
    resultado2 = np.dot(matriz_a, matriz_c)
    glBegin(GL_LINES)

    # Coordenadas del punto inicial rotado
    glVertex2f(resultado1[0], resultado1[1])
    # Coordenadas del punto final rotado
    glVertex2f(resultado2[0], resultado2[1])
    glEnd()
```

```
def display():
    glClear(GL_COLOR_BUFFER_BIT)
    punto()
    punto_rotacion(x1, y1, x2, y2, -40)
    punto_rotacion(x1, y1, x2, y2, -30)
    punto_rotacion(x1, y1, x2, y2, -20)
    punto_rotacion(x1, y1, x2, y2, -10)
    punto_rotacion(x1, y1, x2, y2, 10)
    punto_rotacion(x1, y1, x2, y2, 20)
    punto_rotacion(x1, y1, x2, y2, 30)
    punto_rotacion(x1, y1, x2, y2, 40)
    glutSwapBuffers()

def main():
    # Declarar x y y como globales para poder
       modificarlas
    global x, y
    glutInit(sys.argv)
    glutInitDisplayMode(GLUT_DOUBLE |
      GLUT_RGB)
    glutInitWindowSize(500, 400)
    glutInitWindowPosition(100, 100)
    glutCreateWindow("Dibujar Punto rotada")
    glClearColor(0.9, 0.92, 0.95, 1.0)
    gluOrtho2D(0.0, 499.0, 0.0, 499.0)
    glutDisplayFunc(display)
    glutMainLoop()

if __name__ == "__main__":
    main()
```

Gráfico generado

Algoritmo de Rotación de la figura E

Se usará la matriz de coordenadas de la figura E presentada en el anterior informe

```
from OpenGL.GL import *
from OpenGL.GLUT import *
from OpenGL.GLU import *
import numpy as np
import math

# Matriz de coordenadas del dibujo original
matriz_origen = [[200, 300, 300, 300, 300, 280, 200,
  280],
[200, 290, 230, 290, 230, 180, 200, 180],
[200, 250, 260, 250, 260, 230, 200, 230],
[200, 200, 300, 200, 300, 180, 200, 180]]

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 punto_rotacion(angulo):
    glColor3f(0.5, 0.3, 0.9)
    # Matriz de rotación
    matriz_a = np.array([[math.cos(math.radians(
      angulo)), -math.sin(math.radians(angulo))], [
      math.sin(math.radians(angulo)), math.cos(math.
      radians(angulo))]])
    # Ubicando cada punto como matriz
    matriz_b = np.array([[matriz_origen[0][0]], [
      matriz_origen[0][1]]])
    matriz_c = np.array([[matriz_origen[0][2]], [
      matriz_origen[0][3]]])
    matriz_d = np.array([[matriz_origen[0][4]], [
      matriz_origen[0][5]]])
    matriz_e = np.array([[matriz_origen[0][6]], [
      matriz_origen[0][7]]])
```

```
matriz_f = np.array([[matriz_origen[1][0]],[
  matriz_origen[1][1]]])
matriz_g = np.array([[matriz_origen[1][2]],[
  matriz_origen[1][3]]])
matriz_h = np.array([[matriz_origen[1][4]],[
  matriz_origen[1][5]]])
matriz_i = np.array([[matriz_origen[1][6]],[
  matriz_origen[1][7]]])

matriz_j = np.array([[matriz_origen[2][0]],[
  matriz_origen[2][1]]])
matriz_k = np.array([[matriz_origen[2][2]],[
  matriz_origen[2][3]]])
matriz_l = np.array([[matriz_origen[2][4]],[
  matriz_origen[2][5]]])
matriz_m = np.array([[matriz_origen[2][6]],[
  matriz_origen[2][7]]])

matriz_n = np.array([[matriz_origen[3][0]],[
  matriz_origen[3][1]]])
matriz_o = np.array([[matriz_origen[3][2]],[
  matriz_origen[3][3]]])
matriz_p = np.array([[matriz_origen[3][4]],[
  matriz_origen[3][5]]])
matriz_q = np.array([[matriz_origen[3][6]],[
  matriz_origen[3][7]]])

# Multiplicando cada matriz punto con la matriz
  de rotación
resultado1 = np.dot(matriz_a, matriz_b)
resultado2 = np.dot(matriz_a, matriz_c)
resultado3 = np.dot(matriz_a, matriz_d)
resultado4 = np.dot(matriz_a, matriz_e)

resultado5 = np.dot(matriz_a, matriz_f)
resultado6 = np.dot(matriz_a, matriz_g)
resultado7 = np.dot(matriz_a, matriz_h)
resultado8 = np.dot(matriz_a, matriz_i)
```

```
resultado9 = np.dot(matriz_a, matriz_j)
resultado10 = np.dot(matriz_a, matriz_k)
resultado11 = np.dot(matriz_a, matriz_l)
resultado12 = np.dot(matriz_a, matriz_m)

resultado13 = np.dot(matriz_a, matriz_n)
resultado14 = np.dot(matriz_a, matriz_o)
resultado15 = np.dot(matriz_a, matriz_p)
resultado16 = np.dot(matriz_a, matriz_q)

# Dibujando la figura rotada
glBegin(GL_QUADS)
glVertex2f(resultado1[0], resultado1[1])
glVertex2f(resultado2[0], resultado2[1])
glVertex2f(resultado3[0], resultado3[1])
glVertex2f(resultado4[0], resultado4[1])

glVertex2f(resultado5[0], resultado5[1])
glVertex2f(resultado6[0], resultado6[1])
glVertex2f(resultado7[0], resultado7[1])
glVertex2f(resultado8[0], resultado8[1])

glVertex2f(resultado9[0], resultado9[1])
glVertex2f(resultado10[0], resultado10[1])
glVertex2f(resultado11[0], resultado11[1])
glVertex2f(resultado12[0], resultado12[1])

glVertex2f(resultado13[0], resultado13[1])
glVertex2f(resultado14[0], resultado14[1])
glVertex2f(resultado15[0], resultado15[1])
glVertex2f(resultado16[0], resultado16[1])

glEnd()
```

```
def display():
    glClear(GL_COLOR_BUFFER_BIT)
    dibujar_E()
    punto_rotacion(-20)
    glutSwapBuffers()

def main():
    glutInit(sys.argv)
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB)
    glutInitWindowSize(500, 400)  # Tamaño de la
      ventana
    glutInitWindowPosition(100, 100)
    glutCreateWindow("Dibujar figura E rotada")
    glClearColor(0.9, 0.92, 0.95, 1.0)
    gluOrtho2D(0.0, 499.0, 0.0, 399.0)
    glutDisplayFunc(display)
    glutMainLoop()

if __name__ == "__main__":
    main()
```

Gráfico generado