Simulación de una Escena con Python

Sebastián Cotes, Yefferson González, Lorann Peñuela

IES, INFOTEP Instituto Nacional de Formación Técnica Profesional "HVG"

Humberto Velazquez Garcia

Abstract

En esta actividad, se desarrollará un programa en Python que nos permita crear y manipular una escena con un punto superpuesto en una circunferencia. Fueron implementadas varias funciones para modificar la escena, desplazar el punto y simular su movimiento en distintas direcciones. Se utilizó la librería "matplotlib" para la representación gráfica y "numpy" para las operaciones vectoriales.

1 Introducción

Python ha tomado gran fuerza entre los programadores en los últimos años. Uno de los principales retos de esta actividad fue aprender a usar Python para generar gráficos interactivos y aplicar operaciones vectoriales en la manipulación de objetos. En este caso en particular, se creó la clase "Escena" que permitirá modificar y representar parámetros como el radio y el color de una circunferencia, así como desplazar, calcular la Norma vectoial, producto escalar y animar el punto dentro de la escena.

2 Objetivos

Los objetivos principales de esta actividad son:

- Reforzar el lenguaje Python y aprender a usarlo para generar gráficos.
- Implementar clases en Python para manejar escena gráfica con operaciones.
- Aplicar operaciones vectoriales para posicionar objetos.
- Calcular producto interno, producto escalar, Norma vectorial y rotación del punto.

3 Descripción de la actividad

Se creó la clase "Escena" con las siguientes funciones:

1. `__init__`: inicializa la escena con un radio, un color y la posición del punto en el centro.

2. `CrearEscena`: crea un plano con coordenadas X, Y.
3. `DibujarEscena`: dibuja una circunferencia y un punto superpuesto en el plano.
4. `DesplazarPunto`: mueve el punto en la dirección que se le indica con operaciones vectoriales.
5. `Escalar`: Calcula el producto escalar.
6. `Norma`: Calcula la norma vectorial.
7. `P. Interno`: Calcula el producto interno.
8. `Rotar`: Calcula los puntos a rotar por medio de matices.

Se utilizaron librerías como `matplotlib` y `numpy` para visualizar la representación gráfica y realizar los cálculos utilizando vectores.

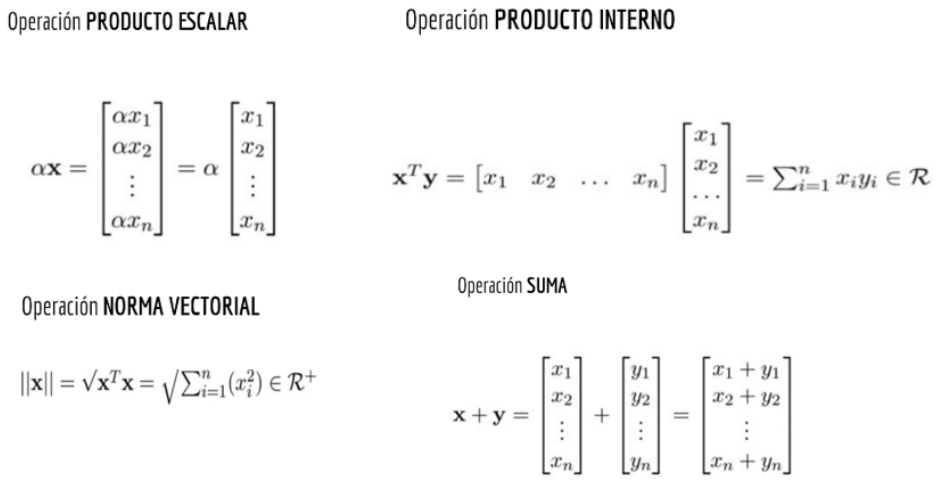

Figure 1: **Ecuaciones utilizadas**

3.1 Codificación

Listing 1: Escena en Python

```
import numpy as np
import matplotlib.pyplot as plt
```

```
class Escena:
  def __init__(self, radio, color):
    self.radio = radio
    self.color = color
    self.punto = np.array([0, 0])
    self.puntorotar = self.punto

#Calculamos Escalar, Mostramos y retornamos su valor
#"return self.punto"
  def escalar(self, escalada):
    self.punto *= escalada
    print("escalar:", escalada)
    return self.punto
    self.dibujar_escena()

#Calculamos Norma con el vector del punto (self.punto),
#Mostramos en pantalla su valor
  def norma(self):
    resultado_norma = np.linalg.norm(self.punto)
    print("norma:", resultado_norma)

#Calculamos Producto Interno con el vector [3, 3] y el
#vector del Punto, Mostramos en pantalla su resultado
  def prod_int(self):
    self.punto2 = np.array([3, 3])
    producto_punto = np.dot(self.punto, self.punto2)
    print("producto interno:", producto_punto)

#Para hacer rotar el punto, utilizamos la matrix de rotacion
#y convertimos el angulo en radianes, se imprime los puntos a
#rotar y se retorna el valor para ser usado por "dibujar_escena"
  def rotar(self, angulo):
    angulo_rad = np.radians(angulo)
    matriz_rotacion = np.array([[np.cos(angulo_rad),
    -np.sin(angulo_rad)],[np.sin(angulo_rad),
    np.cos(angulo_rad)]])
    self.puntorotar = matriz_rotacion @ self.punto
    print("rotar:", angulo,"  ", self.puntorotar)
    self.dibujar_escena(mostrar_rotacion=True)

#Crea la escena, el circulo y el punto, ademas, recibe los
#parametros para dibujar la ubicacion del punto
  def dibujar_escena(self, mostrar_rotacion=False):
```

```
    plt.clf()
    fig, ax = plt.subplots()
    ax.add_patch(plt.Circle((0, 0), self.radio,
    color=self.color, fill=False))
    plt.plot(*self.punto, 'ro')

    if mostrar_rotacion:
      plt.plot(*self.puntorotar, 'o', color='purple')

    ax.set_aspect('equal')
    ax.set_xlim(-self.radio-1, self.radio+1)
    ax.set_ylim(-self.radio-1, self.radio+1)
    plt.grid(True)
    plt.show()

#Mueve el punto segun datos que le demos, como direccion y distancia.
  def desplazar_punto(self, distancia, direccion):
    direcciones = {
        'arriba': np.array([0, distancia]),
        'abajo': np.array([0, -distancia]),
        'izquierda': np.array([-distancia, 0]),
        'derecha': np.array([distancia, 0])
    }
    if direccion in direcciones:
      self.punto +=direcciones[direccion]
      self.dibujar_escena()

escena = Escena(15, 'blue')
escena.desplazar_punto(2, 'arriba')
#escena.escalar toma el valor con el que realizara las operciones
escena.escalar(2)
escena.norma()
escena.prod_int()
#escena.rota toma el angulo en el se quiere rotar el punto
escena.rotar(90)
```

4 Gráficas de Resultados

Terminada la **codificación**[1], y teniendo en cuenta los requerimientos iniciales, obtenemos como resultado las siguientes.

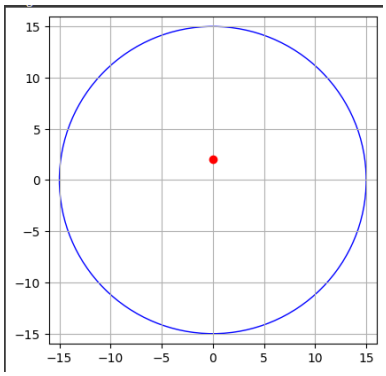

Figure 2: **Escena inicial con el punto desplazado**

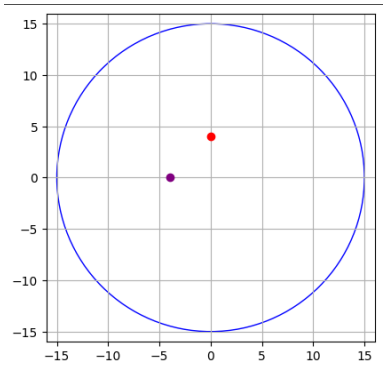

Figure 3: **Escena con la norma vectorial, producto escalar, producto interno y rotación según angulo dado.**

5 Conclusiones

Esta actividad nos ayudó a tener una ligera idea de cómo usar la librería matplotlib para representar gráficos y de cómo aplicar operaciones vectoriales en la manipulación de posiciones de objetos. También fue posible observar de manera clara los cambios en la escena a medida que se daban las operaciones. En general, este proceso fue una experiencia útil para reforzar conceptos de programación orientada a objetos, uso de operaciones vectoriales y representaciones gráficas en Python.

References

[1] Google colab, simulación [online], available from: `https://colab.research.google.com/drive/1h0Tr2HTVS7MuHLpgFsa4eV6wTZHEJdD_?usp=sharing`.