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. Y 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, por lo tanto, uno de los principales retos de esta actividad fue aprender a usar Python para generar gráficos interactivos y aplicando operaciones vectoriales en la manipulación de objetos. En este caso en particular, fue creada la clase "Escena" que permitirá modificar y representar parámetros como el radio y el color que tiene una circunferencia, como también desplazar 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 una clase en Python para manejar una escena grafica.
- Aplicar operaciones vectoriales para posicionar objetos.
- Simular el movimiento de un punto dentro de una circunferencia.

3 Descripción de la actividad

Se creo 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. `CambiarEscena`: permite modificar el radio de la circunferencia, el color y la ubicación del punto.

4. `DibujarEscena`: Dibuja una circunferencia y un punto superpuesto en el plano.

5. `DesplazarPunto`: mueve el punto en la dirección que se le indica con operaciones vectoriales.

6. `simular`: repite el desplazamiento varias veces para que se genere una animación.

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

3.1 Codificación

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

class Escena:
    def __init__(self):
        self.figure = None
        self.circulo = None
        self.punto = np.array([0, 0])
        self.punto_plot = None

    def CrearEscena(self):
        plt.figure(figsize=[5,5])
        plt.axis([0, 10, 0, 10])

    def DibujarEscena(self):
        escena.CrearEscena()
        circulo = plt.Circle([5, 5], radius=4, color='gray')
        plt.gca().add_patch(circulo)
        punto = plt.Circle([5, 5], radius=0.1, color='purple')
        plt.gca().add_patch(punto)
        plt.show()

    def CambiarEscena(self, radio, color, punto):
        escena.CrearEscena()
        circulo = plt.Circle((5, 5), radius=radio, color=color)
        plt.gca().add_patch(circulo)
        self.punto = np.array(punto)
        self.punto_plot, = plt.plot(self.punto[0], self.punto[1], 'mo')
```

Figure 1: **Codificación parte uno**

```python
    def DesplazarPunto(self, x, direccion):
        escena.CrearEscena()
        desplazamiento = np.array([0, 0])
        if direccion == 'derecha':
            desplazamiento = np.array([x, 0])
        elif direccion == 'izquierda':
            desplazamiento = np.array([-x, 0])
        elif direccion == 'arriba':
            desplazamiento = np.array([0, x])
        elif direccion == 'abajo':
            desplazamiento = np.array([0, -x])
        self.punto += desplazamiento
        self.punto_plot, = plt.plot(self.punto[0], self.punto[1], 'mo')
        self.circulo = plt.Circle((5, 5), 4, color='gray')
        plt.gca().add_patch(self.circulo)

        plt.draw()

    def simular(self, velocidad, direccion, pasos):
        for _ in range(pasos):
            escena.DesplazarPunto(velocidad, direccion)
            plt.pause(0.1)

escena = Escena()
escena.DibujarEscena()
escena.CambiarEscena(3,'blue', [5, 5])
escena.simular(velocidad=1, direccion='derecha', pasos=4)
```

Figure 2: **Codificación parte dos**

4 Gráficas de Resultados

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

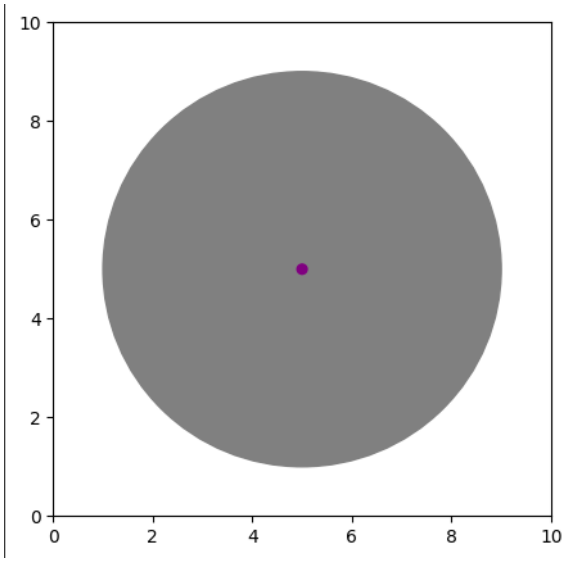

Figure 3: **Escena inicial**

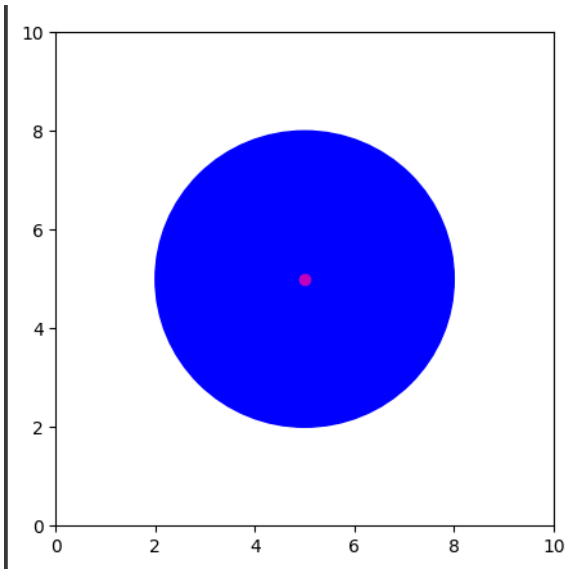

Figure 4: **Escena modificada**

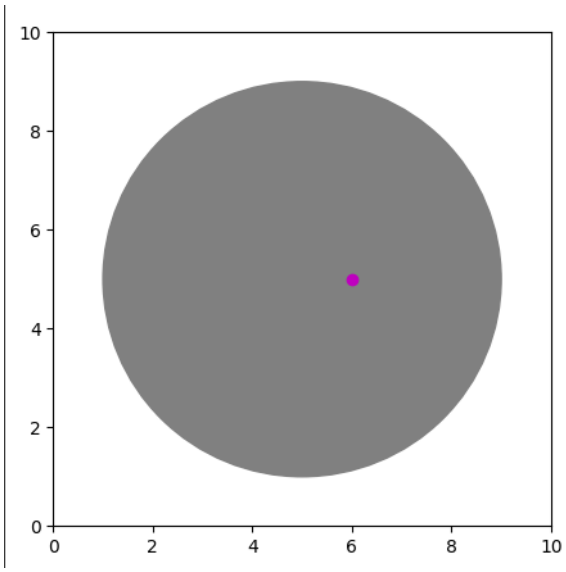

Figure 5: **Escena con movimiento del punto, pasando de [5,5] a [6,5]**

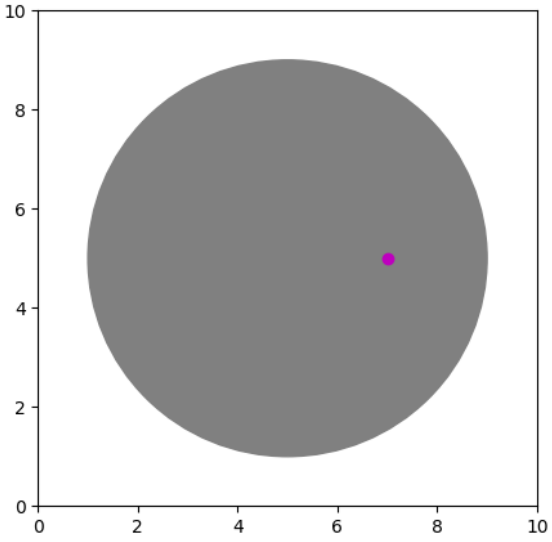

Figure 6: **Simulación del movimiento en eje Y**

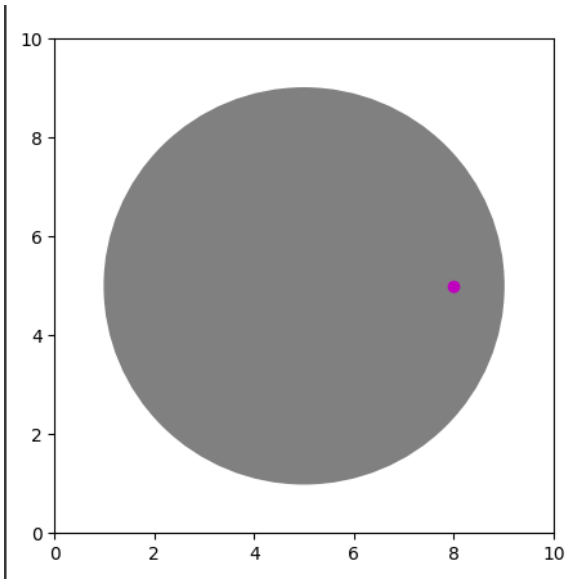

Figure 7: **Simulación del movimiento en eje Y**

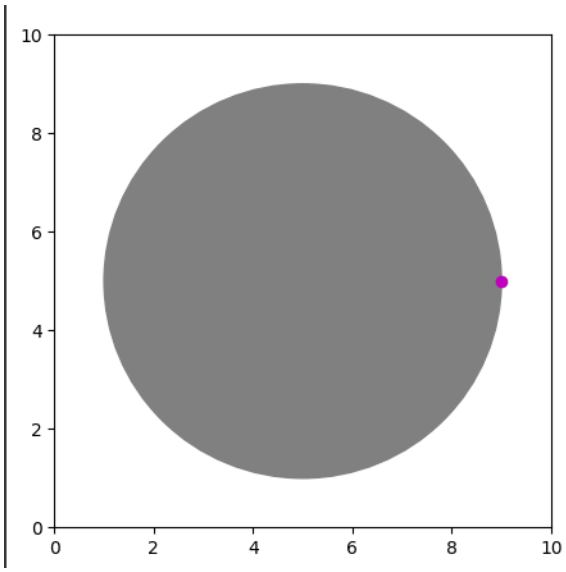

Figure 8: **Simulación del movimiento en eje Y**

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 los desplazamientos. En general, este proceso fue una experiencia útil para reforzar conceptos de programación orientada a objetos y representaciones gráficas en Python.

References

[1] Google colab, simulación [online], available from: `https://colab.research.google.com/drive/1onSWs5gkRaDiIT-dlCY8cSFXHvgXhMkW?usp=sharing`.