Rotación y Traslación 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 nos pide realizar un código que evalué el efecto del orden de las rotaciones y traslaciones ya que son operaciones básicas para modelar el movimiento, se utilizó para este caso varias librerías que nos ayudaran a realizar los cálculos correspondientes y las presentaciones gráficas.

1 Introducción

Python ha tomado gran fuerza entre los programadores en los últimos años. Uno de los principales retos de esta actividad fue usar Python para entender los conceptos de rotación y traslación, estas son operaciones básicas para modelar el movimiento y la orientación de objetos en un espacio tridimensional (3D) o bidimensional (2D). Estos conceptos son esenciales en campos como la física, la robótica, la ingeniería, etc.

2 Objetivos

Los objetivos principales de esta actividad son:

- Reforzar el lenguaje Python y aprender a usarlo para generar gráficos y operaciones con matrices.
- Comprender conceptos de traslación y rotación de un punto en un plano.
- Evaluar y comprender el efecto del orden de las operaciones vectoriales para el calculo de rotación

3 Descripción de la actividad

Se nos pide realizar operaciones con matrices para la rotación y traslación de un objeto o punto, utilizando como base un código desarrollado en clase que explica detalladamente estos conceptos y en diferentes dimensiones, ya sea 2D o 3D.Ademas, Evaluar y comprender el efecto del orden de las operaciones vectoriales para el calculo de rotación

Entender los conceptos de rotación y traslación es fundamental en la simulación, ya que son operaciones básicas para modelar el movimiento y la orientación de objetos en un espacio tridimensional o bidimensional. Estos conceptos son esenciales en campos como la física, la robótica, los gráficos por computadora, la realidad virtual, los videojuegos y la ingeniería.

- Rotación: Permite girar un objeto alrededor de un eje o punto fijo, lo que es esencial para cambiar la orientación de un objeto en el espacio. Por ejemplo, en un videojuego, un personaje puede rotar para mirar en una dirección específica. Para la rotación es muy importante el angulo.
- Traslación: Desplaza un objeto de una posición a otra en el espacio, sin alterar su orientación. Esto es crucial para simular movimientos lineales, como el desplazamiento de un vehículo en una carretera, y no depende de ningún angulo.

3.1 Codificación

Se utilizaron varias librerías para realizar esta actividad. Se resaltaran las funciones más importantes y su propósito.

1. `def dibujar_punto()`: Dibuja un punto en un espacio tridimensional y lo etiqueta.
2. `R_x = Matrix()`: Define una matriz de rotación simbólica alrededor del eje X
3. `df . to_csv` : Guardar un nuevo archivo CSV Modificado.
4. `def trasladar_punto()`: Realiza las operaciones para trasladar el punto.

Listing 1: Importando librerias

```
import numpy as np
from sympy import*
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
init_printing()
```

Listing 2: Función dibujar un vector punto en un espacio tridimencional

```
def dibujar_punto(punto, label, x_lim = [0, 1], y_lim=[0, 1],
    z_lim=[0, 1]):
  """
  Dibuja un punto en un espacio tridimensional.

  Parametros:
    punto: Un vector NumPy de tres elementos representando las
        coordenadas (x, y, z) del punto.
    label: Una cadena de texto para etiquetar el punto en el
        gr fico.
    x_lim: Una lista de dos elementos que define los l mites
        del eje x.
    y_lim: Una lista de dos elementos que define los l mites
        del eje y.
    z_lim: Una lista de dos elementos que define los l mites
        del eje z.
  """

  fig = plt.figure()
  ax = fig.add_subplot(111, projection='3d')

  ax.scatter(punto[0], punto[1], punto[2], color='b', label=
      label)

  ax.set_xlabel('X')
  ax.set_ylabel('Y')
  ax.set_zlabel('Z')
  ax.set_title('Espacio 3D')

  ax.quiver(0, 0, 0, punto[0], punto[1], punto[2], color='b')

  ax.set_xlim(x_lim)
  ax.set_ylim(y_lim)
  ax.set_zlim(z_lim)
  ax.legend()
  plt.show()
```

Listing 3: Efecto del orden de las rotaciones

```
theta_val = np.pi # 180 grados en radianes
R_x_p0 = np.array(R_x.subs({theta: theta_val})).astype(np.
    float64) #Se convierte theta simbolico a un valor concreto
    Rx
R_y_p0 = np.array(R_y.subs({theta: theta_val})).astype(np.
    float64) #Se convierte theta simbolico a un valor concreto
    Ry
R_z_p0 = np.array(R_z.subs({theta: theta_val})).astype(np.
    float64) #Se convierte theta simbolico a un valor concreto
    Rz

P0_rotado_x = np.dot(R_x_p0, P0) #Producto punto entre la
    matrix de rotaci n y el vector de posici n del punto
P0_rotado_y = np.dot(R_y_p0, P0_rotado_x) #Producto punto
    entre la matrix de rotaci n y el vector de posici n del
    punto
P0_rotado_z = np.dot(R_z_p0, P0_rotado_y) #Producto punto
    entre la matrix de rotaci n y el vector de posici n del
    punto

print(P0)
print(P0_rotado_x)
print(P0_rotado_y)
print(P0_rotado_z)

dibujar_punto(P0_rotado_z, label = 'Rotacion XYZ', x_lim=[-1,
    1], y_lim=[-1, 1])

P0_rotado_z = np.dot(R_z_p0, P0) #Producto punto entre la
    matrix de rotaci n y el vector de posici n del punto
P0_rotado_y = np.dot(R_y_p0, P0_rotado_z) #Producto punto
    entre la matrix de rotaci n y el vector de posici n del
    punto
P0_rotado_x = np.dot(R_x_p0, P0_rotado_y ) #Producto punto
    entre la matrix de rotaci n y el vector de posici n del
    punto

print(P0)
print(P0_rotado_x)
print(P0_rotado_y)
print(P0_rotado_z)

dibujar_punto(P0_rotado_z, label = 'Rotacion ZYX', x_lim=[-1,
    1], y_lim=[-1, 1])
```

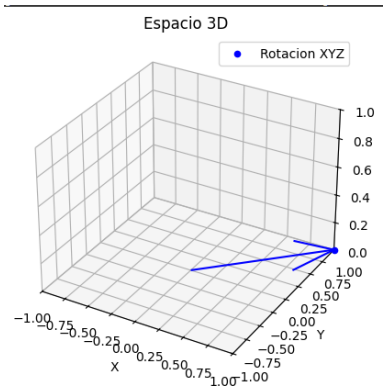

Figure 1: **Rotación XYZ**

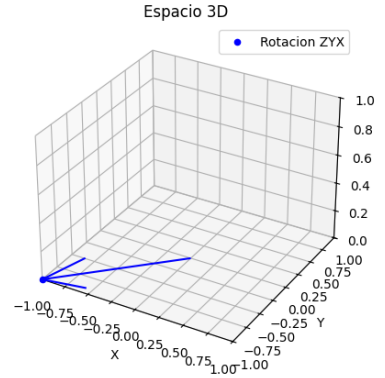

Figure 2: **Rotación ZYX, Se evidencia que el orden de las operaciones son muy importantes para la rotación de un objeto**

Listing 4: Funcion para relizar traslaciones

```
def trasladar_punto(punto, traslacion):
    return punto + np.array(traslacion)

traslacion = [1, -2, 0.8]
P_trasladado = trasladar_punto(P0, traslacion)
print("Punto original",P0)
dibujar_punto(P0, "Punto", x_lim=[-1, 2], y_lim=[-2, 1.5])
print("Punto trasladado:", P_trasladado)
dibujar_punto(P_trasladado, "Punto trasladado", x_lim=[-1, 2],
     y_lim=[-2, 1.5])
```

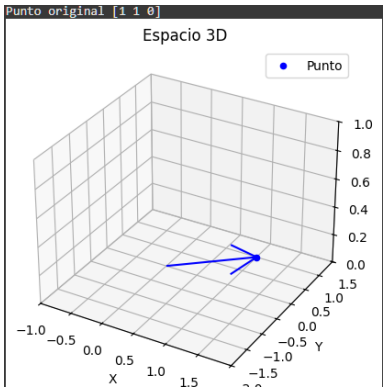

Figure 3: **Ubicacion del Punto original**

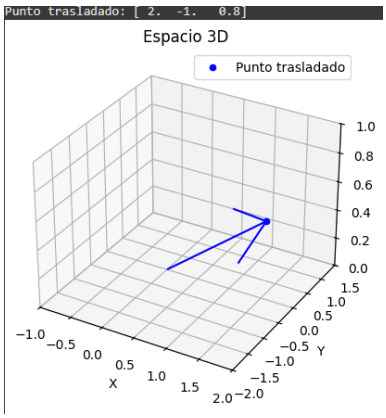

Figure 4: **Traslación del punto según datos ingresados**

4 Conclusiones

La actividad permitió comprender la importancia de las transformaciones geométricas, como la rotación y la traslación, en la simulación y modelado de movimientos en espacios 2D. Utilizando Python. Un hallazgo clave fue que el orden de las transformaciones afecta el resultado final.este proceso fue una experiencia útil para reforzar conceptos de programación orientada a objetos, uso de operaciones vectoriales y representaciones gráficas en Python.