**Explicación línea por línea**

1. import numpy as np
   * Importa la librería NumPy y la asigna al alias np.
2. import matplotlib.pyplot as plt
   * Importa la librería Matplotlib y la asigna al alias plt.
3. from mpl\_toolkits.mplot3d import Axes3D
   * Importa Axes3D para habilitar gráficos en 3D en Matplotlib.
4. from matplotlib.animation import FuncAnimation # Para la animación
   * Importa FuncAnimation para crear animaciones en Matplotlib.
5. def traslacion\_3d(punto, tx, ty, tz):
   * Define la función traslacion\_3d que traslada un punto en 3D.
6. """
   * Inicio del comentario de documentación (docstring).
7. Aplica una transformación de traslación 3D a un punto.
   * Explicación breve de la función.
8. Args:
   * Inicia la descripción de los argumentos.
9. punto: un Array NumPy de 3 elementos que presenta el punto (x, y, z).
   * Explica el parámetro punto.
10. tx: La cantidad de traslación en el eje X,
    * Explica el parámetro tx.
11. ty: La cantidad de traslación en el eje Y,
    * Explica el parámetro ty.
12. tz: La cantidad de traslación en el eje Z
    * Explica el parámetro tz.
13. Returns:
    * Indica lo que devuelve la función.
14. Un array NumPy de 3 elementos que representa el punto trasladado.
    * Explica el valor de retorno.
15. """
    * Fin del comentario de documentación.
16. matriz\_traslacion = np.array([[1, 0, 0, tx], [0, 1, 0, ty], [0, 0, 1, tz], [0, 0, 0, 1]])
    * Crea la matriz de traslación en 3D.
17. punto\_homogeneo = np.append(punto, 1)
    * Convierte el punto a coordenadas homogéneas.
18. punto\_trasladado\_homogeneo = np.dot(matriz\_traslacion, punto\_homogeneo)
    * Aplica la transformación de traslación.
19. punto\_trasladado = punto\_trasladado\_homogeneo[:3]
    * Extrae las coordenadas del punto trasladado.
20. return punto\_trasladado
    * Devuelve el punto trasladado.
21. # Ejemplo de uso
    * Comentario indicando que se ejecutará un ejemplo.
22. punto\_original = np.array([1, 2, 3])
    * Define un punto en el espacio 3D.
23. tx = 5
    * Define la traslación en X.
24. ty = -2
    * Define la traslación en Y.
25. tz = 1
    * Define la traslación en Z.
26. punto\_trasladado = traslacion\_3d(punto\_original, tx, ty, tz)
    * Aplica la traslación al punto original.
27. print("Punto original:", punto\_original)
    * Imprime el punto original.
28. print("Punto trasladado:", punto\_trasladado)
    * Imprime el punto trasladado.
29. fig = plt.figure()
    * Crea una figura en Matplotlib.
30. ax = fig.add\_subplot(111, projection="3d")
    * Agrega un subplot 3D a la figura.
31. ax.scatter(punto\_original[0], punto\_original[1], punto\_original[2], c="r", marker="o", label="Punto Original")
    * Dibuja el punto original en rojo.
32. ax.scatter(punto\_trasladado[0], punto\_trasladado[1], punto\_trasladado[2], c="b", marker="^", label="Punto Trasladado")
    * Dibuja el punto trasladado en azul.
33. ax.set\_xlabel("Eje X")
    * Etiqueta el eje X.
34. ax.set\_ylabel("Eje Y")
    * Etiqueta el eje Y.
35. ax.set\_zlabel("Eje Z")
    * Etiqueta el eje Z.
36. ax.plot([punto\_original[0], punto\_trasladado[0]], [punto\_original[1], punto\_trasladado[1]], [punto\_original[2], punto\_trasladado[2]], "g--")
    * Dibuja una línea entre los puntos.
37. plt.legend()
    * Muestra la leyenda.
38. plt.title("Traslación 3D")
    * Coloca un título.
39. def update(angle):
    * Define la función para animar la vista.
40. ax.view\_init(elev=30, azim=angle)
    * Cambia la vista de la gráfica.
41. return (ax,)
    * Retorna el eje modificado.
42. ani = FuncAnimation(fig, update, frames=np.arange(0, 360, 2), interval=50)
    * Crea una animación de rotación.
43. plt.show()
    * Muestra la figura y la animación.

