🚀 Visualizing 3D Vectors and Coordinate Systems: A Python Adventure! 🌍📊

Hello, network! 🌟 I'm excited to share a `Python program` I recently worked on that brings 3D vector operations and coordinate transformations to life!Whether you're exploring the intricacies of dot products, cross products, or visualizing transformations in Cylindrical and Spherical coordinate systems, this program has you covered.

🎛️ Features:

✅ Vector Operations:

Computes dot product and cross product of two 3D vectors. Displays results in Cartesian, Cylindrical, and Spherical formats.

✅ Coordinate Transformations:

Converts Cartesian vectors to Cylindrical coordinates: (r, θ, z). Converts Cartesian vectors to Spherical coordinates: (r, θ, φ).

✅ 3D Visualization:

Plots vectors in Cartesian, Cylindrical, and Spherical coordinate systems. Includes clearly labeled x, y, z axes for each visualization for better understanding.

🖼️ What You'll See:

A Cartesian plot with your vectors and their cross product. A Cylindrical plot showing the radial and azimuthal components. A Spherical plot with a sphere and your vector's orientation.

🌟 Why It Matters:

This project is a hands-on way to learn the math and geometry behind 3D vectors and coordinate systems. By combining programming with visualization, it turns abstract concepts into intuitive visuals that are both educational and fun.

Here's a quick look at what the code does:

Key Highlights:

- Computes dot & cross products
- Converts between Cartesian, Cylindrical, and Spherical coordinates

- Visualizes 3D vectors in multiple coordinate systems ## 🔧 Tools Used:
- Python 🐍
- NumPy for mathematical operations
- Matplotlib for 3D plotting This was a great exercise in blending math, programming, and visualization—perfect for students, engineers, and anyone curious about 3D geometry.

💬 What do you think? Have you worked on similar projects? Let me know in the comments! I'm always excited to learn new perspectives. 🌟

#Python #DataScience #3DVisualization #Mathematics #Programming

#Python #DataScience #3DVisualization #Mathematics

#VectorOperations #Programming #MachineLearning #AI

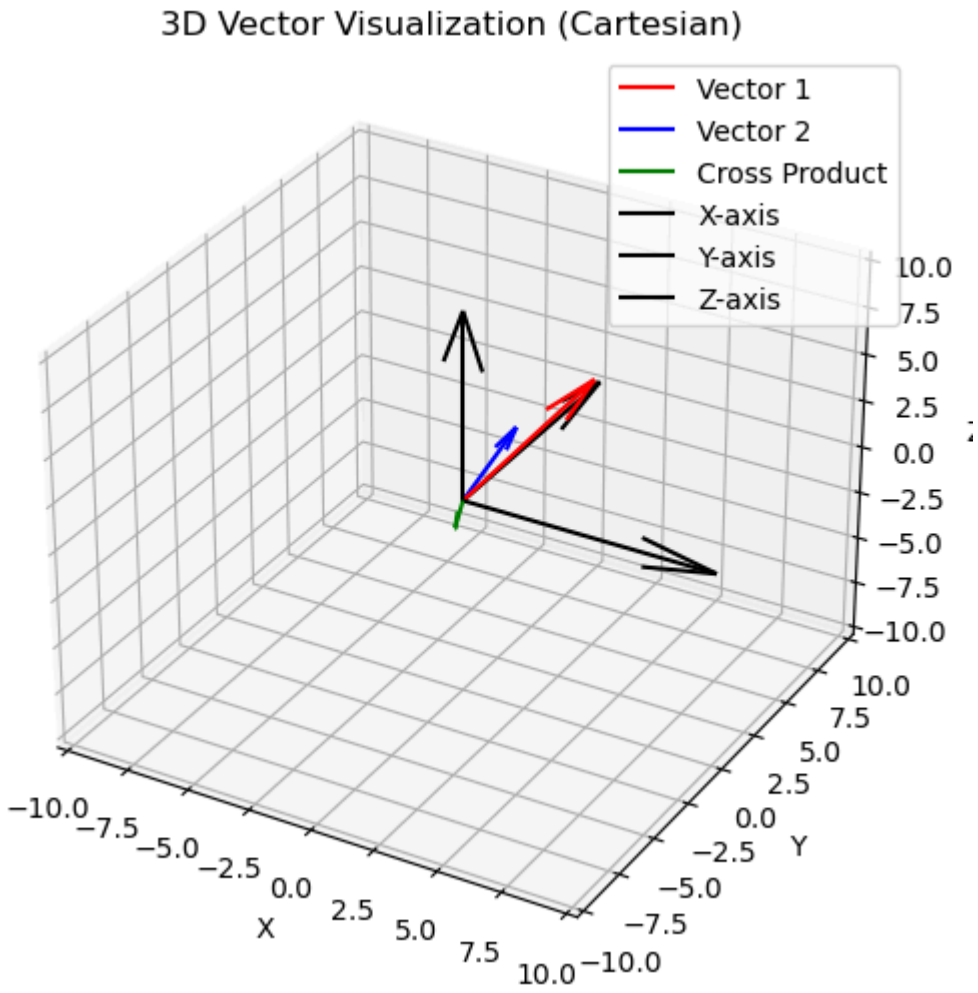
#TechInnovation #LearnToCode #SoftwareDevelopment

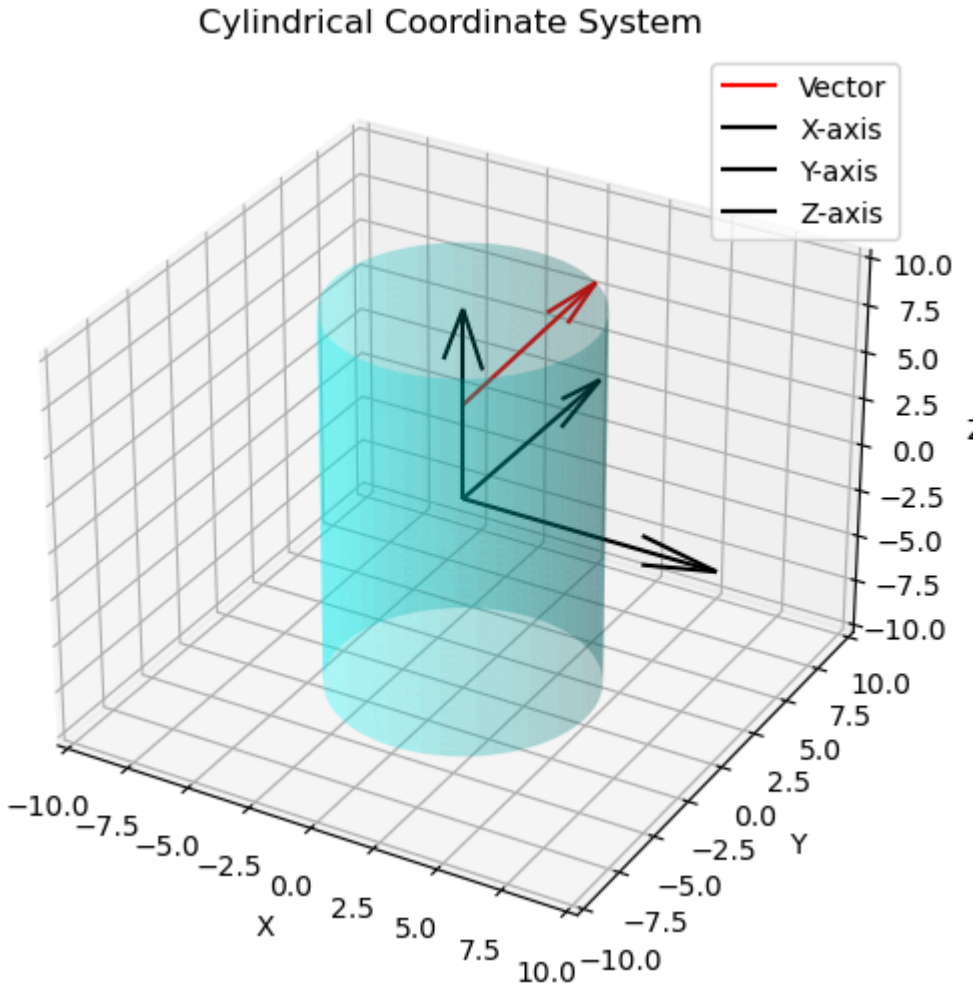
In [1]:
```python
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Function to get a 3D vector input from the user in the form (x, y, z)
def get_vector_input(name):
    while True:
        try:
            # Ask the user for the vector input
            vector_str = input(f"Enter the components of {name} vector in the form
            # Convert the input string to a tuple and then to a numpy array
            vector = tuple(map(float, vector_str.strip('()').split(',')))
            if len(vector) == 3:
                return np.array(vector)
            else:
                print("Error: Please enter exactly 3 components (x, y, z).")
        except ValueError:
            print("Error: Invalid input. Please enter numbers in the format (x, y,

# Function to calculate the dot product
def dot_product(v1, v2):
    return np.dot(v1, v2)

# Function to calculate the cross product
def cross_product(v1, v2):
    return np.cross(v1, v2)

# Convert Cartesian to Cylindrical coordinates (r, ϑ [degrees], z)
def cartesian_to_cylindrical(x, y, z):
    r = np.sqrt(x**2 + y**2)  # Radial distance
    theta = np.degrees(np.arctan2(y, x))  # Azimuthal angle in degrees
    return r, theta, z

# Convert Cartesian to Spherical coordinates (r, ϑ [degrees], φ [degrees])
def cartesian_to_spherical(x, y, z):
    r = np.sqrt(x**2 + y**2 + z**2)  # Radial distance
    theta = np.degrees(np.arccos(z / r)) if r != 0 else 0  # Polar angle in degrees
    phi = np.degrees(np.arctan2(y, x))  # Azimuthal angle in degrees
    return r, theta, phi

# Plot the vectors in 3D Cartesian coordinates
def plot_vectors(vector1, vector2, cross):
```

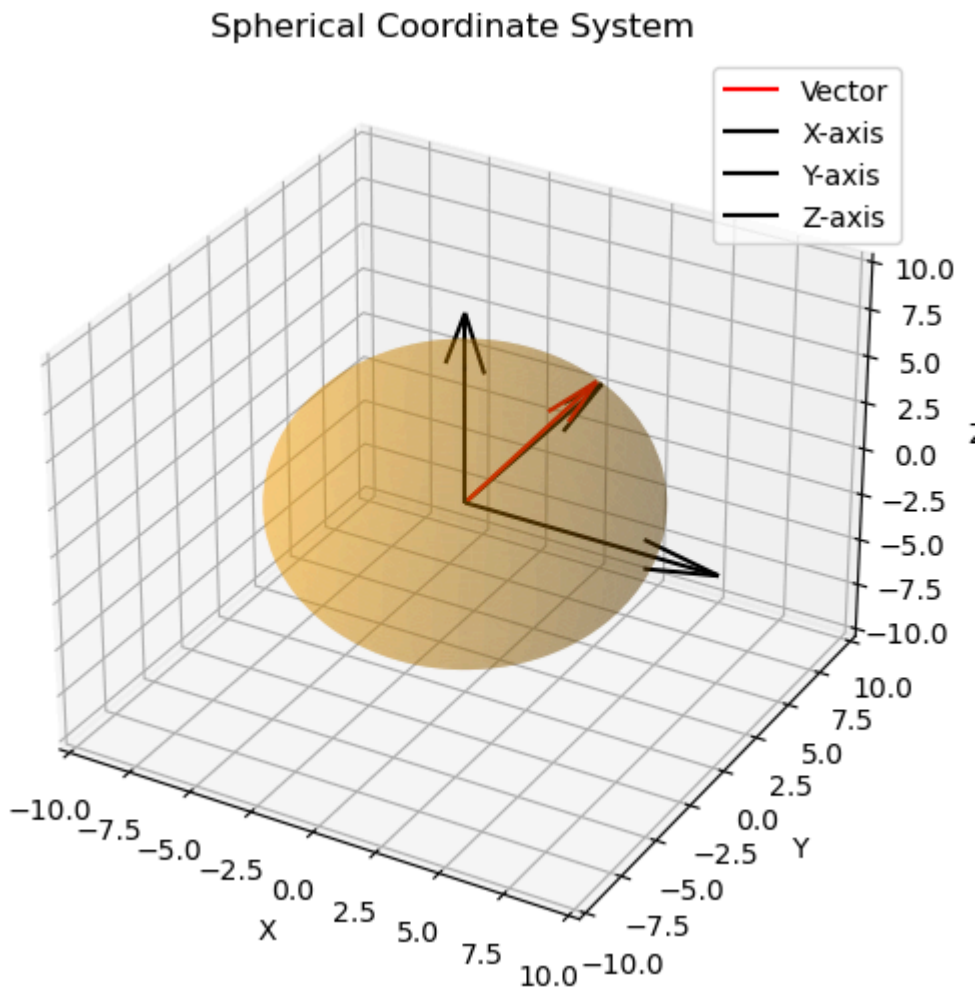
```
    fig = plt.figure(figsize=(8, 6))
    ax = fig.add_subplot(111, projection='3d')

    # Plot vector 1 and vector 2
    ax.quiver(0, 0, 0, vector1[0], vector1[1], vector1[2], color='r', label='Vector
    ax.quiver(0, 0, 0, vector2[0], vector2[1], vector2[2], color='b', label='Vector
    ax.quiver(0, 0, 0, cross[0], cross[1], cross[2], color='g', label='Cross Produc

    # Draw x, y, z axes
    ax.quiver(0, 0, 0, 10, 0, 0, color='k', label='X-axis')
    ax.quiver(0, 0, 0, 0, 10, 0, color='k', label='Y-axis')
    ax.quiver(0, 0, 0, 0, 0, 10, color='k', label='Z-axis')

    ax.set_xlabel('X')
    ax.set_ylabel('Y')
    ax.set_zlabel('Z')
    ax.set_xlim([-10, 10])
    ax.set_ylim([-10, 10])
    ax.set_zlim([-10, 10])
    ax.legend()
    plt.title("3D Vector Visualization (Cartesian)")
    plt.show()

# Plot the cylindrical coordinates
def plot_cylindrical(r, theta, z):
    fig = plt.figure(figsize=(8, 6))
    ax = fig.add_subplot(111, projection='3d')

    # Create the cylindrical surface (a simple 3D circle for visualization)
    theta_vals = np.linspace(0, 2 * np.pi, 100)
    z_vals = np.linspace(-10, 10, 100)
    theta_grid, z_grid = np.meshgrid(theta_vals, z_vals)
    x_cyl = r * np.cos(theta_grid)
    y_cyl = r * np.sin(theta_grid)

    ax.plot_surface(x_cyl, y_cyl, z_grid, alpha=0.3, color='cyan')

    # Plot the vector in cylindrical coordinates
    ax.quiver(0, 0, z, r * np.cos(np.radians(theta)), r * np.sin(np.radians(theta))

    # Draw x, y, z axes
    ax.quiver(0, 0, 0, 10, 0, 0, color='k', label='X-axis')
    ax.quiver(0, 0, 0, 0, 10, 0, color='k', label='Y-axis')
    ax.quiver(0, 0, 0, 0, 0, 10, color='k', label='Z-axis')

    ax.set_xlabel('X')
    ax.set_ylabel('Y')
    ax.set_zlabel('Z')
    ax.set_xlim([-10, 10])
    ax.set_ylim([-10, 10])
    ax.set_zlim([-10, 10])
    ax.legend()
    plt.title("Cylindrical Coordinate System")
    plt.show()

# Plot the spherical coordinates
def plot_spherical(r, theta, phi):
    fig = plt.figure(figsize=(8, 6))
    ax = fig.add_subplot(111, projection='3d')

    # Create a sphere for visualization
    u = np.linspace(0, 2 * np.pi, 100)
    v = np.linspace(0, np.pi, 100)
    x_sph = r * np.outer(np.cos(u), np.sin(v))
```

```
    y_sph = r * np.outer(np.sin(u), np.sin(v))
    z_sph = r * np.outer(np.ones(np.size(u)), np.cos(v))

    ax.plot_surface(x_sph, y_sph, z_sph, alpha=0.3, color='orange')

    # Plot the vector in spherical coordinates
    ax.quiver(0, 0, 0, r * np.sin(np.radians(theta)) * np.cos(np.radians(phi)),
              r * np.sin(np.radians(theta)) * np.sin(np.radians(phi)),
              r * np.cos(np.radians(theta)), color='r', label='Vector')

    # Draw x, y, z axes
    ax.quiver(0, 0, 0, 10, 0, 0, color='k', label='X-axis')
    ax.quiver(0, 0, 0, 0, 10, 0, color='k', label='Y-axis')
    ax.quiver(0, 0, 0, 0, 0, 10, color='k', label='Z-axis')

    ax.set_xlabel('X')
    ax.set_ylabel('Y')
    ax.set_zlabel('Z')
    ax.set_xlim([-10, 10])
    ax.set_ylim([-10, 10])
    ax.set_zlim([-10, 10])
    ax.legend()
    plt.title("Spherical Coordinate System")
    plt.show()

# Main program
if __name__ == "__main__":
    # Get vectors from the user
    vector1 = get_vector_input("first")
    vector2 = get_vector_input("second")

    # Calculate dot and cross products
    dot = dot_product(vector1, vector2)
    cross = cross_product(vector1, vector2)

    # Transform the vectors to cylindrical and spherical coordinates
    cylindrical1 = cartesian_to_cylindrical(*vector1)
    cylindrical2 = cartesian_to_cylindrical(*vector2)
    spherical1 = cartesian_to_spherical(*vector1)
    spherical2 = cartesian_to_spherical(*vector2)

    # Display results in a concise manner
    print("\nResults:")
    print(f"First Vector (Cylindrical): r={cylindrical1[0]:.2f}, θ={cylindrical1[1]
    print(f"Second Vector (Cylindrical): r={cylindrical2[0]:.2f}, θ={cylindrical2[1
    print(f"First Vector (Spherical): r={spherical1[0]:.2f}, θ={spherical1[1]:.2f}°
    print(f"Second Vector (Spherical): r={spherical2[0]:.2f}, θ={spherical2[1]:.2f}
    print(f"Dot Product: {dot:.2f}")
    print(f"Cross Product: {cross}")

    # Plot the results
    plot_vectors(vector1, vector2, cross)
    plot_cylindrical(*cylindrical1)
    plot_spherical(*spherical1)
```

```
Results:
First Vector (Cylindrical): r=5.00, θ=53.13°, z=5.00
Second Vector (Cylindrical): r=2.24, θ=63.43°, z=3.00
First Vector (Spherical): r=7.07, θ=45.00°, ϕ=53.13°
Second Vector (Spherical): r=3.74, θ=36.70°, ϕ=63.43°
Dot Product: 26.00
Cross Product: [ 2. -4.  2.]
```


3D Vector Visualization (Cartesian)
Vector 1
Vector 2
Cross Product
X-axis
Y-axis
Z-axis
X
Y


Cylindrical Coordinate System
Vector
X-axis
Y-axis
Z-axis
X
Y



This is `Gaafer Mohsen Gouda`

I'm defending my position as the best upcoming Astroinformatics Specialist In shaa Allah & This is My LinkedIn profile Just a Tool to Flourish

https://www.linkedin.com/in/gaafer-gouda-nasa

I'm participating as a Trainee at Digital Egypt Pioneers Initiative - DEPI , **Under Supervision of** Ministry of Communications and Information Technology (MCIT), Egypt **AI** & **Data Science** IBM. **I'm asking Allah always for more guidance** & **prosperity**

【Amidst Seniors,Ever present】