Joshua Kyle K. Entrata - 4CSC

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from mpl_toolkits.mplot3d.art3d import Poly3DCollection
```

1. Translate the prism in the last activity using the parameter [5, 7, 2]

In [ ]:

```
#DRAWING PRISM
# Parameters for the rectangular prism
length = 21  # Length of the prism
width = 16   # Width of the prism
height = 23  # Height of the prism

# Define the vertices of the rectangular prism
vertices = np.array([
    [0, 0, 0],  # Vertex 0
    [length, 0, 0],  # Vertex 1
    [length, width, 0],  # Vertex 2
    [0, width, 0],  # Vertex 3
    [0, 0, height],  # Vertex 4
    [length, 0, height],  # Vertex 5
    [length, width, height],  # Vertex 6
    [0, width, height],  # Vertex 7
])

# Define the 6 faces of the rectangular prism
faces = [
    [vertices[0], vertices[1], vertices[2], vertices[3]],  # Bottom face
    [vertices[4], vertices[5], vertices[6], vertices[7]],  # Top face
    [vertices[0], vertices[1], vertices[5], vertices[4]],  # Front face
    [vertices[1], vertices[2], vertices[6], vertices[5]],  # Right face
    [vertices[2], vertices[3], vertices[7], vertices[6]],  # Back face
    [vertices[3], vertices[0], vertices[4], vertices[7]],  # Left face
]
```

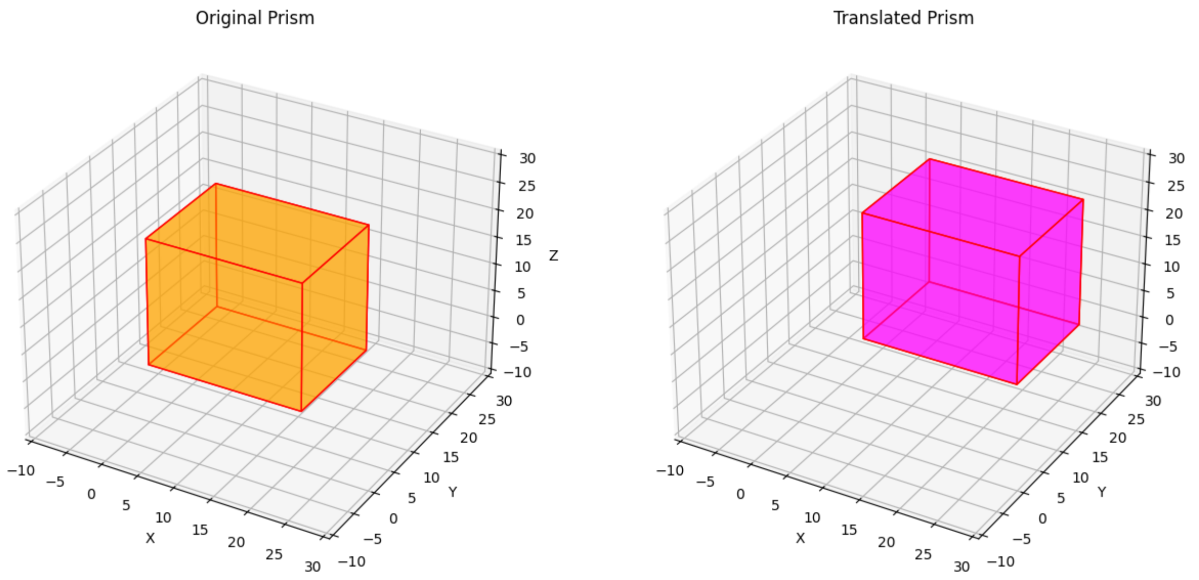
In [ ]:

```
translation = np.array([5, 7, 2])

vertices_translated = vertices + translation

faces_translated = [
    [vertices_translated[0], vertices_translated[1], vertices_translated[2], vertic
    [vertices_translated[4], vertices_translated[5], vertices_translated[6], vertic
    [vertices_translated[0], vertices_translated[1], vertices_translated[5], vertic
    [vertices_translated[1], vertices_translated[2], vertices_translated[6], vertic
    [vertices_translated[2], vertices_translated[3], vertices_translated[7], vertic
    [vertices_translated[3], vertices_translated[0], vertices_translated[4], vertic
]
```

In [3]:

```
fig = plt.figure(figsize=(15, 12))
# Plot original prism
```

```python
ax1 = fig.add_subplot(121, projection='3d')
ax1.add_collection3d(Poly3DCollection(faces, facecolors='orange', linewidths=1, edg
ax1.set_title('Original Prism')
ax1.set_xlabel('X')
ax1.set_ylabel('Y')
ax1.set_zlabel('Z')
ax1.set_xlim([-10, 30])
ax1.set_ylim([-10, 30])
ax1.set_zlim([-10, 30])

# Plot translated prism
ax2 = fig.add_subplot(122, projection='3d')
ax2.add_collection3d(Poly3DCollection(faces_translated, facecolors='magenta', linew
ax2.set_title('Translated Prism')
ax2.set_xlabel('X')
ax2.set_ylabel('Y')
ax2.set_zlabel('Z')
ax2.set_xlim([-10, 30])
ax2.set_ylim([-10, 30])
ax2.set_zlim([-10, 30])

# Display the plot
plt.show()
```



Original Prism

Translated Prism

2. Scale the cube in the last activity using the parameter [0, 2, 1]

In [ ]:

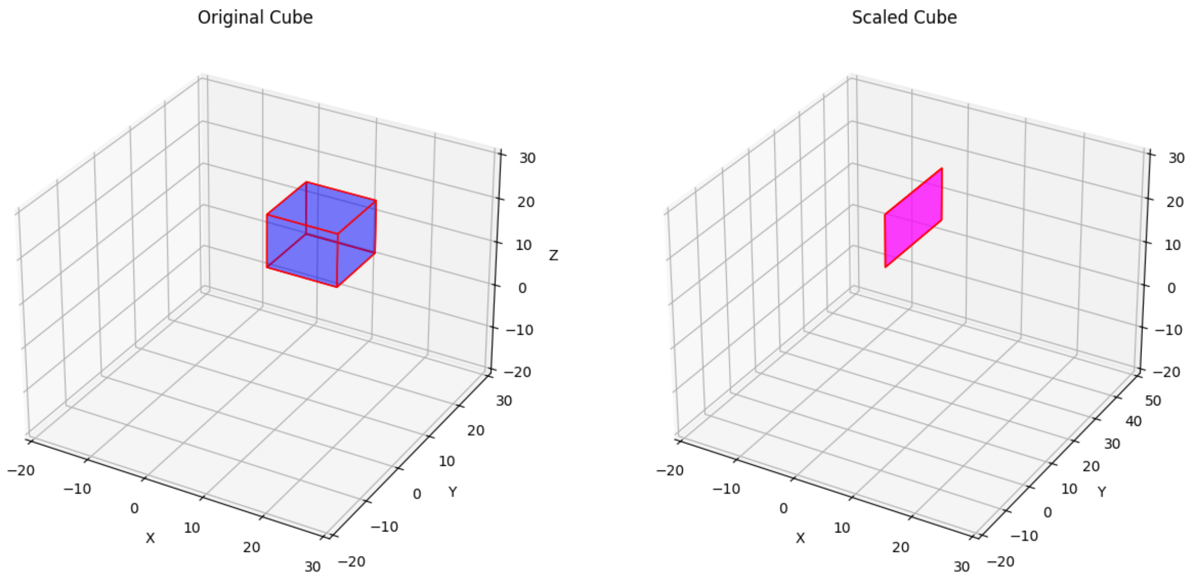
```python
#DRAWING CUBE
initial_coord = np.array([4, 8, 9])  # Corner (4, 8, 9)
edge_length = 12

vertices = np.array([
    initial_coord,  # Corner 1
    initial_coord + [edge_length, 0, 0],  # Corner 2
    initial_coord + [edge_length, edge_length, 0],  # Corner 3
    initial_coord + [0, edge_length, 0],  # Corner 4
    initial_coord + [0, 0, edge_length],  # Corner 5
    initial_coord + [edge_length, 0, edge_length],  # Corner 6
    initial_coord + [edge_length, edge_length, edge_length],  # Corner 7
    initial_coord + [0, edge_length, edge_length]  # Corner 8
])

# Define the 6 faces of the cube using vertex indices
faces = [
    [vertices[0], vertices[1], vertices[2], vertices[3]],  # Bottom
    [vertices[4], vertices[5], vertices[6], vertices[7]],  # Top
    [vertices[0], vertices[1], vertices[5], vertices[4]],  # Front
    [vertices[2], vertices[3], vertices[7], vertices[6]],  # Back
    [vertices[1], vertices[2], vertices[6], vertices[5]],  # Right
    [vertices[4], vertices[7], vertices[3], vertices[0]]   # Left
]
```

In [ ]:

```python
scaling_factors = np.array([0, 2, 1])

# Apply the scaling
vertices_scaled = vertices * scaling_factors

faces_scaled = [
    [vertices_scaled[0], vertices_scaled[1], vertices_scaled[2], vertices_scaled[3]
    [vertices_scaled[4], vertices_scaled[5], vertices_scaled[6], vertices_scaled[7]
    [vertices_scaled[0], vertices_scaled[1], vertices_scaled[5], vertices_scaled[4]
    [vertices_scaled[2], vertices_scaled[3], vertices_scaled[7], vertices_scaled[6]
    [vertices_scaled[1], vertices_scaled[2], vertices_scaled[6], vertices_scaled[5]
    [vertices_scaled[4], vertices_scaled[7], vertices_scaled[3], vertices_scaled[0]
]
```

In [5]:

```python
fig = plt.figure(figsize=(15, 12))
# Plot original cube
ax1 = fig.add_subplot(121, projection='3d')
ax1.add_collection3d(Poly3DCollection(faces, facecolors='blue', linewidths=1, edgec
ax1.set_title('Original Cube')
ax1.set_xlabel('X')
ax1.set_ylabel('Y')
ax1.set_zlabel('Z')
ax1.set_xlim([-20, 30])
ax1.set_ylim([-20, 30])
ax1.set_zlim([-20, 30])
# Plot a scaled cube
ax2 = fig.add_subplot(122, projection='3d')
ax2.add_collection3d(Poly3DCollection(faces_scaled, facecolors='magenta', linewidth
ax2.set_title('Scaled Cube')
ax2.set_xlabel('X')
ax2.set_ylabel('Y')
ax2.set_zlabel('Z')
ax2.set_xlim([-20, 30])
ax2.set_ylim([-20, 50])
ax2.set_zlim([-20, 30])

# Display the plot
plt.show()
```



3. Rotate the pyramid in the last activity using 120 degrees about x, y and z axis.

In [6]:

```
#DRAWING PYRAMID
# Parameters for the pyramid
length = 21  # Length of the rectangular base
width = 16   # Width of the rectangular base
height = 23  # Height of the pyramid

# Define the vertices of the pyramid
vertices = np.array([
    [-length / 2, -width / 2, 0],  # Base corner 1
    [ length / 2, -width / 2, 0],  # Base corner 2
    [ length / 2,  width / 2, 0],  # Base corner 3
    [-length / 2,  width / 2, 0],  # Base corner 4
    [ 0, 0, height]                # Apex
])

# Define the faces of the pyramid (4 triangles + 1 square base)
faces = [
    [vertices[0], vertices[1], vertices[4]],  # Front face
    [vertices[1], vertices[2], vertices[4]],  # Right face
    [vertices[2], vertices[3], vertices[4]],  # Back face
    [vertices[3], vertices[0], vertices[4]],  # Left face
    [vertices[0], vertices[1], vertices[2], vertices[3]]  # Base face
]
```

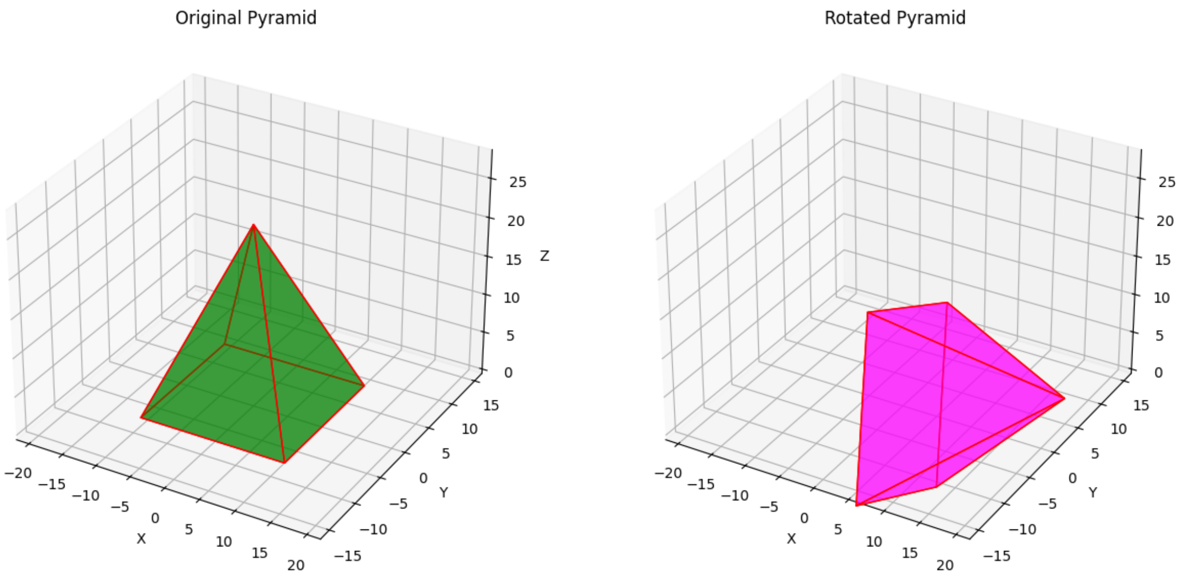
In [7]:

```
angle = 120 # Angle in degrees
theta = np.radians(angle)  # Convert to radians



rotation_matix_x = np.array([
    [1, 0, 0],
    [0, np.cos(theta), -np.sin(theta)],
    [0, np.sin(theta), np.cos(theta)]
])
```

```
rotation_matix_y = np.array([
    [np.cos(theta), 0, np.sin(theta)],
    [0, 1, 0],
    [-np.sin(theta), 0, np.cos(theta)]
])

rotation_matix_z = np.array([
    [np.cos(theta), -np.sin(theta), 0],
    [np.sin(theta), np.cos(theta), 0],
    [0, 0, 1]
])

# Apply the rotation to all vertices
rotated_vertices = np.dot(vertices, rotation_matix_x.T)
rotated_vertices = np.dot(rotated_vertices, rotation_matix_y.T)
rotated_vertices = np.dot(rotated_vertices, rotation_matix_z.T)

# Define the faces of the pyramid (4 triangles + 1 square base)
faces_rotated = [
    [rotated_vertices[0], rotated_vertices[1], rotated_vertices[4]],  # Front face
    [rotated_vertices[1], rotated_vertices[2], rotated_vertices[4]],  # Right face
    [rotated_vertices[2], rotated_vertices[3], rotated_vertices[4]],  # Back face
    [rotated_vertices[3], rotated_vertices[0], rotated_vertices[4]],  # Left face
    [rotated_vertices[0], rotated_vertices[1], rotated_vertices[2], rotated_vertice
]
```

In [8]:

```
fig = plt.figure(figsize=(15, 12))
# Plot original pyramid
ax1 = fig.add_subplot(121, projection='3d')
ax1.add_collection3d(Poly3DCollection(faces, facecolors='green', linewidths=1, edge
ax1.set_title('Original Pyramid')
ax1.set_xlabel('X')
ax1.set_ylabel('Y')
ax1.set_zlabel('Z')
ax1.set_xlim([-length, length])
ax1.set_ylim([-width, width])
ax1.set_zlim([0, height + 5])
# Plot rotated pyramid
ax2 = fig.add_subplot(122, projection='3d')
ax2.add_collection3d(Poly3DCollection(faces_rotated, facecolors='magenta', linewidt
ax2.set_title('Rotated Pyramid')
ax2.set_xlabel('X')
ax2.set_ylabel('Y')
ax2.set_zlabel('Z')
ax2.set_xlim([-length, length])
ax2.set_ylim([-width, width])
ax2.set_zlim([0, height + 5])

# Display the plot
plt.show()
```



Original Pyramid

Rotated Pyramid