Amit Divekar | Practical 9

3D Transformations

In [1]:
```
from sympy import Point3D, Matrix, cos, sin, pi
```

Q1. Using Python and SymPy, perform a translation of a 3D point P(1,2,3) by the vector (3,−1,2)

In [2]:
```
from sympy import Point3D
P = Point3D(1, 2, 3)
P_t = P.translate(3, -1, 2)
print("Original Point:", P)
print("Translated Point:", P_t)
```

```
Original Point: Point3D(1, 2, 3)
Translated Point: Point3D(4, 1, 5)
```

Q2. Apply scaling transformation on a 3D point using a transformation matrix.

In [3]:
```
from sympy import Matrix
P = Matrix([2, 1, 3])
S = Matrix([[2,0,0], [0,3,0], [0,0,1]])
P_s = S * P
print("Original Point:", P)
print("Scaled Point:", P_s)
```

```
Original Point: Matrix([[2], [1], [3]])
Scaled Point: Matrix([[4], [3], [3]])
```

Q3. Rotate a 3D point P(2,1,3) about the Z-axis by 90 degrees.

In [4]:
```
from sympy import Matrix, cos, sin, pi
P = Matrix([2, 1, 3])
Rz = Matrix([[cos(pi/2), -sin(pi/2), 0],[sin(pi/2), cos(pi/2), 0], [0, 0, 1]
P_r = Rz * P
print("Original Point:", P)
print("Rotated Point:", P_r)
```

```
Original Point: Matrix([[2], [1], [3]])
Rotated Point: Matrix([[-1], [2], [3]])
```

Q4. Find the reflection of a 3D point about the XY-plane using a transformation matrix.

In [5]:
```
from sympy import Matrix
P = Matrix([3, -2, 4])
R = Matrix([[1,0,0], [0,1,0],[0,0,-1]])
P_ref = R * P
print("Original Point:", P)
print("Reflected Point:", P_ref)
```

```
Original Point: Matrix([[3], [-2], [4]])
Reflected Point: Matrix([[3], [-2], [-4]])
```

Q5. Apply shearing transformation to a 3D point along the X-axis.

In [6]:
```
from sympy import Matrix
P = Matrix([1, 2, 3])
Sh = Matrix([[1,1,0], [0,1,0], [0,0,1]])
P_sh = Sh * P
print("Original Point:", P)
print("Sheared Point:", P_sh)
```

```
Original Point: Matrix([[1], [2], [3]])
Sheared Point: Matrix([[3], [2], [3]])
```

Q6. Apply rotation about Z-axis followed by translation on a 3D point.

In [7]:
```
from sympy import Matrix, cos, sin, pi
P = Matrix([1, 2, 1])
R = Matrix([[cos(pi/2), -sin(pi/2), 0],[sin(pi/2), cos(pi/2), 0],[0, 0, 1]])
P_r = R * P
P_rt = P_r + Matrix([2, 3, 1])
print("Original Point:", P)
print("Rotated + Translated Point:", P_rt)
```

```
Original Point: Matrix([[1], [2], [1]])
Rotated + Translated Point: Matrix([[0], [4], [2]])
```

Q7. Perform scaling transformation on a 3D line segment defined by two points.

In [8]:
```
from sympy import Point3D, Matrix
A = Matrix([1, 1, 1])
B = Matrix([3, 2, 1])
S = Matrix([[2,0,0], [0,2,0],[0,0,2]])
A_s = S * A
B_s = S * B
```

```
print("Original Line Points:", A, B)
print("Scaled Line Points:", A_s, B_s)
```

```
Original Line Points: Matrix([[1], [1], [1]]) Matrix([[3], [2], [1]])
Scaled Line Points: Matrix([[2], [2], [2]]) Matrix([[6], [4], [2]])
```