CS475/CS675
Computer Graphics

3D Transformations

3D Transformations

$$S(s_x, s_y, s_z) = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad S^{-1}(s_x, s_y, s_z) = S(\frac{1}{s_x}, \frac{1}{s_y}, \frac{1}{s_z})$$

Scaling

3D Transformations

$$T(l,m,n)=\begin{bmatrix}1 & 0 & 0 & l\\ 0 & 1 & 0 & m\\ 0 & 0 & 1 & n\\ 0 & 0 & 0 & 1\end{bmatrix} \qquad T^{-1}(l,m,n)=T(-l,-m,-n)$$

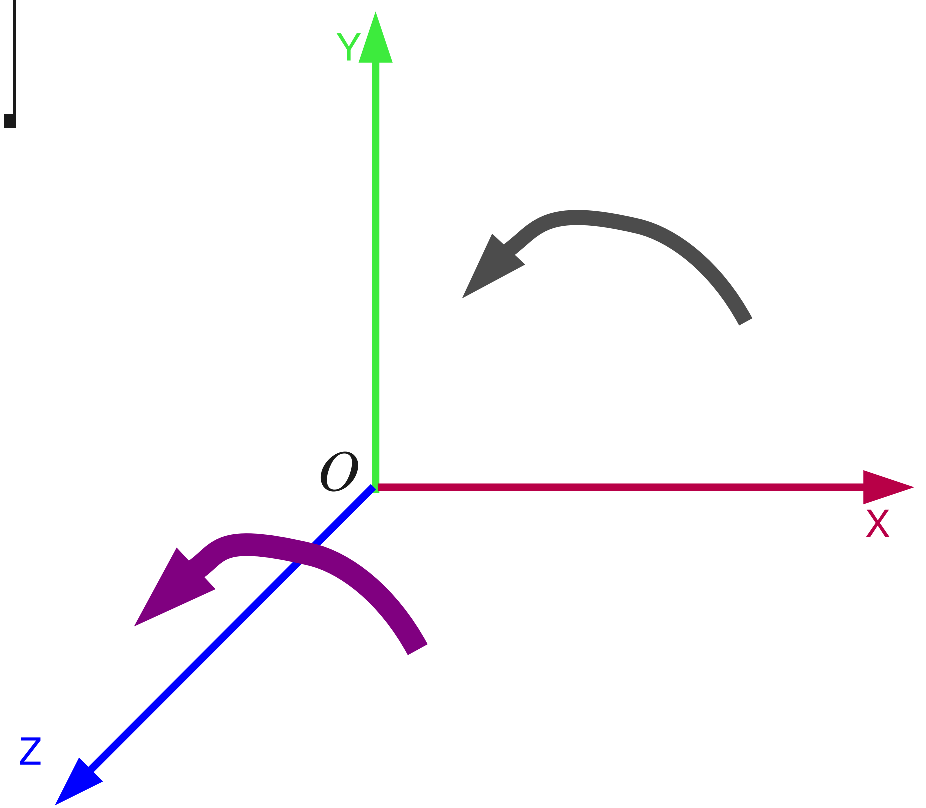
Translation

3D Transformations

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_z^{-1}(\theta) = R_z(-\theta) = R_z^T$$

Rotation about Z axis

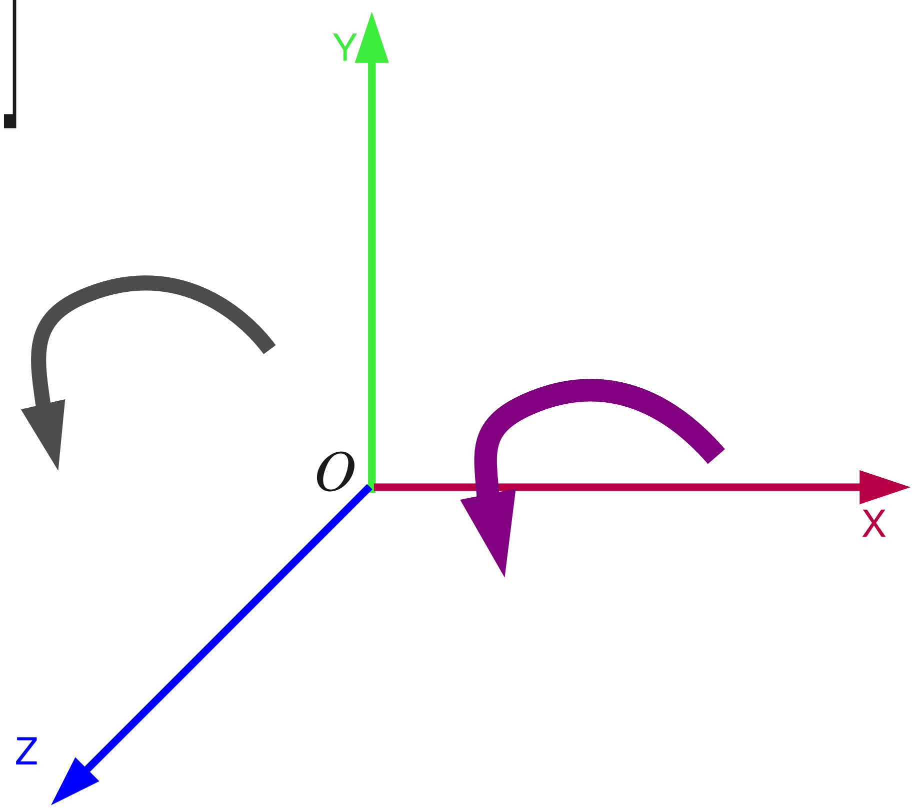


3D Transformations

$$R_x(\theta)=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \cos\theta & -\sin\theta & 0\\ 0 & \sin\theta & \cos\theta & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$$R_x^{-1}(\theta)=R_x(-\theta)=R_x^T$$

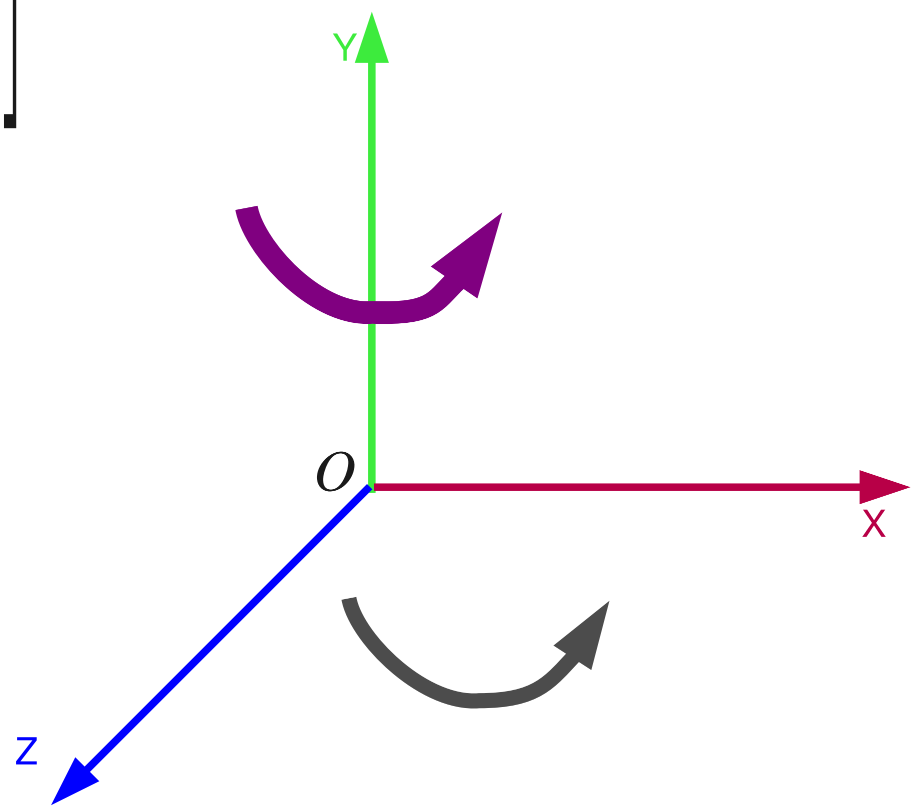
Rotation about X axis



3D Transformations

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_y^{-1}(\theta) = R_y(-\theta) = R_y^T$$



Rotation about Y axis

3D Transformations

In particular for Rotations

$$R^T_{axis}(\theta).R_{axis}(\theta)=R_{axis}(\theta).R^T_{axis}(\theta)=I$$

Rotations are orthogonal matrices.

$$det(R_{axis}(\theta))=1$$

3D Transformations

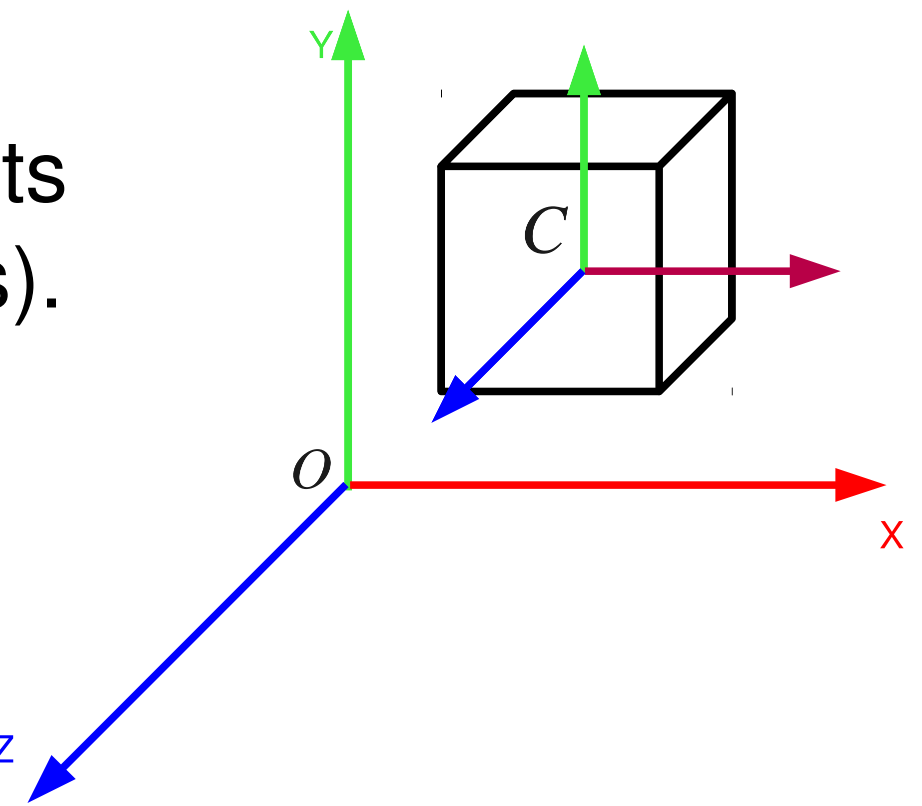
Shear

$$Sh = \begin{bmatrix} 1 & d & g & 0 \\ b & 1 & h & 0 \\ c & f & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3D Transformations

Rotating a cube about its center (about the z axis).

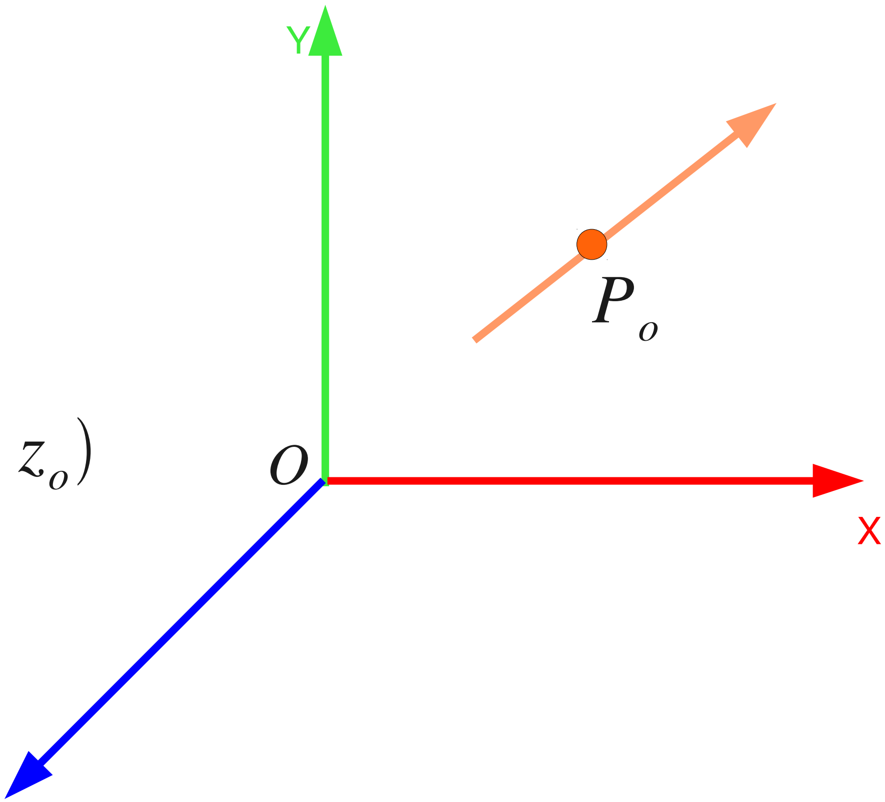
$$P' = T(C).R_z(\theta).T(-C).P$$



3D Transformations

Rotating about
an arbitrary axis.

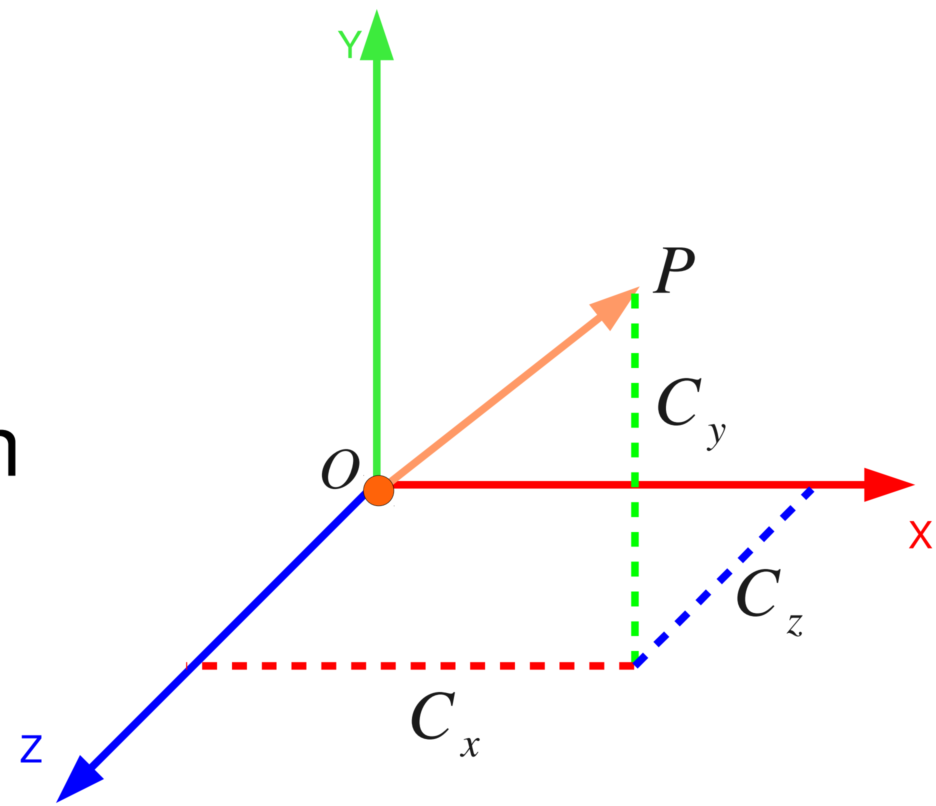
Passing through $P_o(x_o, y_o, z_o)$
and
with direction cosines as
C_x, C_y, C_z
by an angle δ



3D Transformations

Rotating about
an arbitrary axis.

Translate P_o to the origin
using $T(-x_o, -y_o, -z_o)$



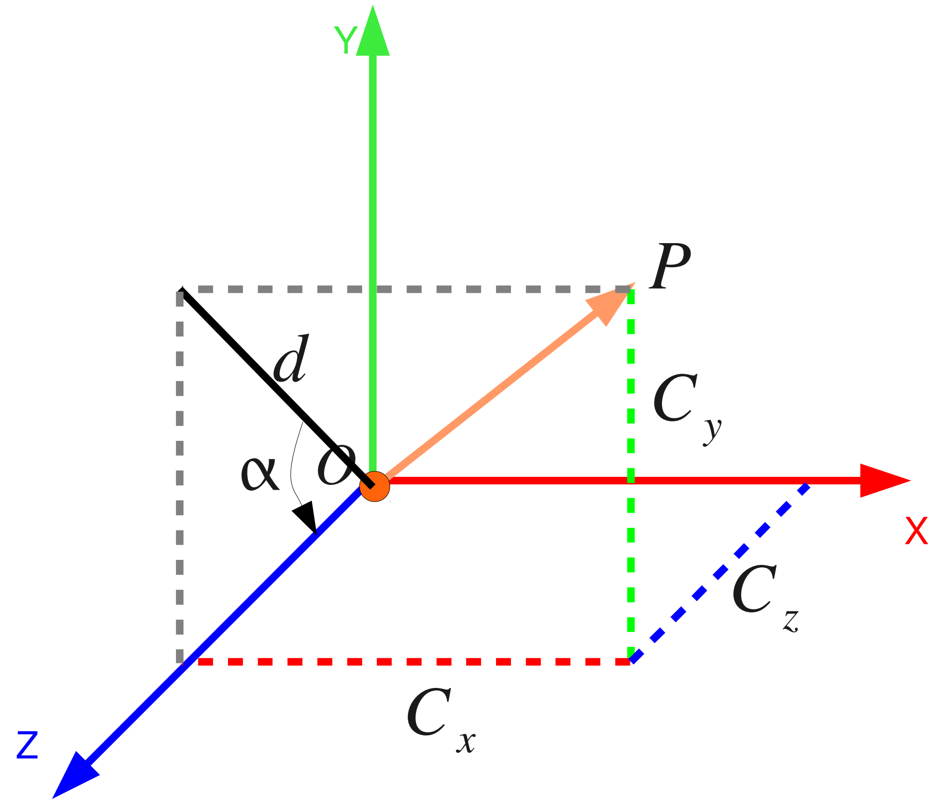
Consider the unit vector
in the direction C_x, C_y, C_z

3D Transformations

Rotating about
an arbitrary axis.

Align the vector $\vec{OP}$
to the z axis

Rotate $\vec{OP}$ such that
it lies on the XZ plane
i.e., rotate about the X axis by α

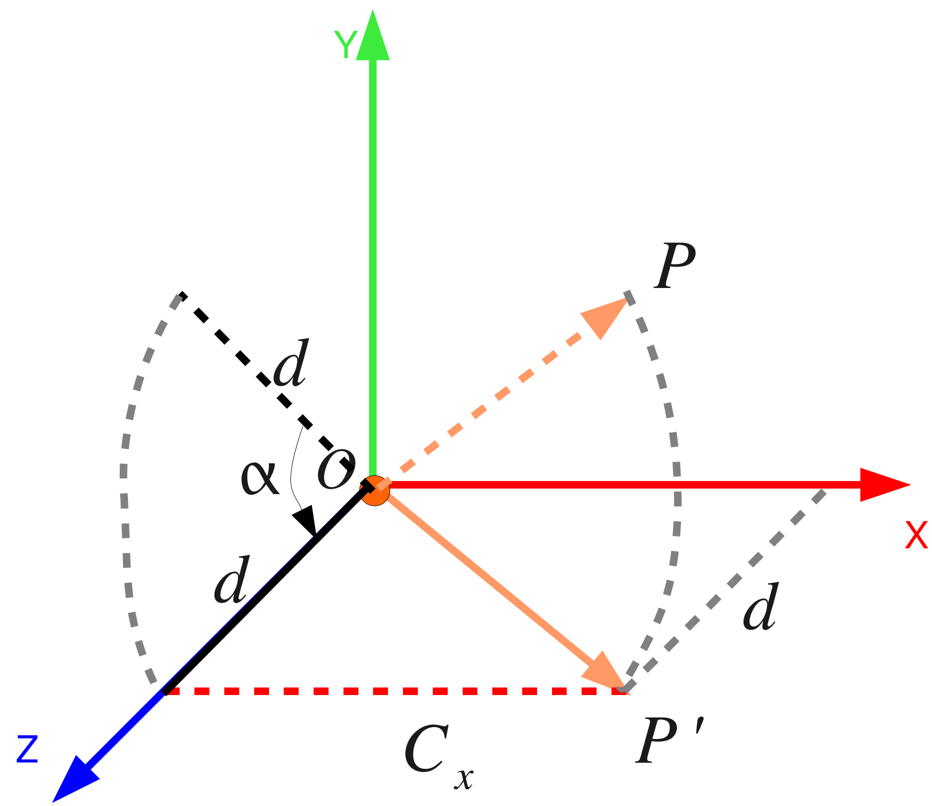


$$d = \sqrt{C_y^2 + C_z^2} \qquad \cos\alpha = \frac{C_z}{d} \qquad \sin\alpha = \frac{C_y}{d} \qquad R_x(\alpha)$$

3D Transformations

Rotating about
an arbitrary axis.

Align the vector $\vec{OP}$
to the z axis



Rotate $\vec{OP}$ such that
it lies on the XZ plane
i.e., rotate about the X axis by α

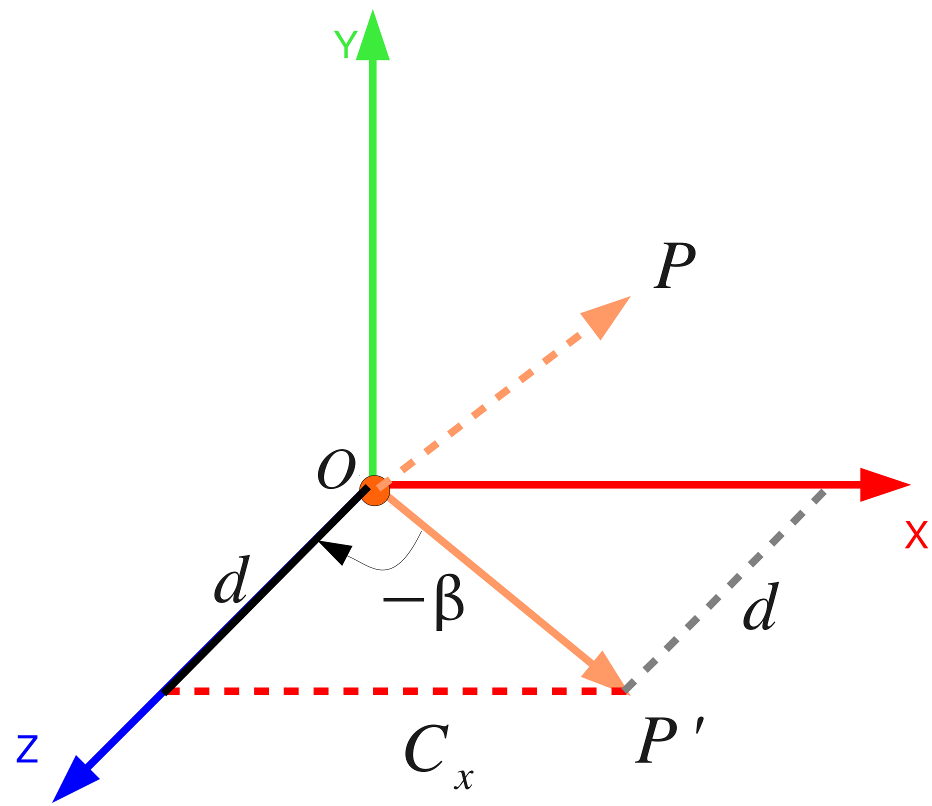
$$d = \sqrt{C_y^2 + C_z^2} \qquad \cos\alpha = \frac{C_z}{d} \qquad \sin\alpha = \frac{C_y}{d} \qquad R_x(\alpha)$$

3D Transformations

Rotating about
an arbitrary axis.

Align the vector $\vec{OP}$
to the z axis

Rotate $\vec{OP}$ around
Y axis by $-\beta$

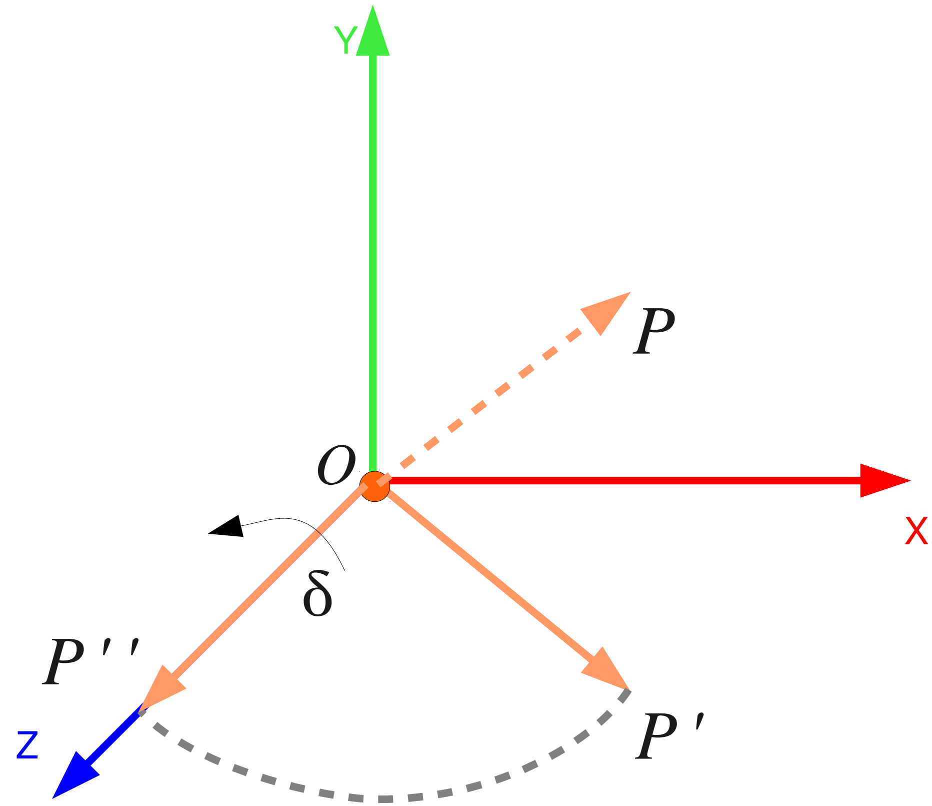


$$d = \sqrt{C_y^2 + C_z^2} \qquad \cos(-\beta) = d \qquad \sin(-\beta) = C_x \qquad R_y(-\beta)$$

3D Transformations

Rotating about
an arbitrary axis.

Now rotate about the
to the z axis by δ



$R_z(delta)$

3D Transformations

Rotating about
an arbitrary axis.

Now do the inverse transforms in reverse order to get the composite transformation matrix as:

$$M = T(P_o) . R_x(-\alpha) . R_y(\beta) . R_z(\delta) . R_y(-\beta) . R_x(\alpha) . T(-P_o)$$

3D Transformations

General 3D transformation:

$$T = \left[\begin{array}{ccc|c} a & d & g & l \\ b & e & h & m \\ c & f & i & n \\ \hline p & q & r & s \end{array}\right]$$

Transformations in OpenGL

- Matrices in GL are always 4x4 matrices and are interpreted in column major order.
- Matrices are pre-multipled to the vertices.
- To transform an object we form the corresponding matrix and multiply it to all the vertices of the object.
- Multiplication with a vertex has to be done explicitly in the shaders.

Transformations in OpenGL

```
in vec4 vPosition;
in vec4 vColor;
out vec4 color;
uniform vec3 theta;

void main()
{
    // Compute the sines and cosines of theta for
    // each of the three axes in one computation.
    vec3 angles = radians( theta );
    vec3 c = cos( angles );
    vec3 s = sin( angles );
```

Transformations in OpenGL

```
// Remember: these matrices are column-major

mat4 rx = mat4( 1.0,  0.0,  0.0, 0.0,
                0.0,  c.x,  s.x, 0.0,
                0.0, -s.x,  c.x, 0.0,
                0.0,  0.0,  0.0, 1.0 );

mat4 ry = mat4( c.y, 0.0, -s.y, 0.0,
                0.0, 1.0,  0.0, 0.0,
                s.y, 0.0,  c.y, 0.0,
                0.0, 0.0,  0.0, 1.0 );

mat4 rz = mat4( c.z, -s.z, 0.0, 0.0,
                 s.z,  c.z, 0.0, 0.0,
                 0.0,  0.0, 1.0, 0.0,
                 0.0,  0.0, 0.0, 1.0 );

    color = vColor;
    gl_Position = rz * ry * rx * vPosition;
}
```

Transformations in OpenGL

- Link the shader variable to application code

```
GLint theta = glGetUniformLocation( program, "theta" );
```

- Update the uniform variable

```
glUniform3fv( theta, 1, glm::value_ptr(vTheta) );
```

- Where vTheta is of the type glm::vec3