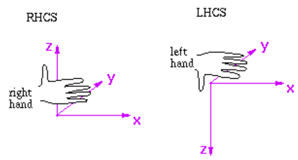
3D Coordinate Systems

- 3D computer graphics involves the additional dimension of depth, allowing more realistic representations of 3D objects in the real world
- There are two possible ways of “attaching” the Z-axis, which gives rise to a left-handed or a right-handed system



3D Transformation

- The translation, scaling and rotation transformations used for 2D can be extended to three dimensions
- In 3D, each transformation is represented by a 4x4 matrix
- Using homogeneous coordinates it is possible to represent each type of transformation in a matrix form and integrate transformations into one matrix
- To apply transformations, simply multiply matrices, also easier in hardware and software implementation
- Homogeneous coordinates can represent directions
- Homogeneous coordinates also allow for non-affine transformations, e.g., perspective projection

Homogeneous Coordinates

- In 2D, use three numbers to represent a point
- (x,y) = (wx,wy,w) for any constant w≠0
- To go backwards, divide by w, (x,y) becomes (x,y,1)
- Transformation can now be done with matrix multiplication

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{xx} & a_{xy} & b_x \\ a_{yx} & a_{yy} & b_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Basic 2D Transformations

- Translation: $\begin{bmatrix} 1 & 0 & b_x \\ 0 & 1 & b_y \\ 0 & 0 & 1 \end{bmatrix}$
- Scaling: $\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$
- Rotation: $\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Translation and Scalling Matrice

- The translation and scaling transformations may be represented in 3D as follows:

Translation matrix

$$\begin{pmatrix} 1 & 0 & 0 & tr_X \\ 0 & 1 & 0 & tr_Y \\ 0 & 0 & 1 & tr_Z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Scaling matrix

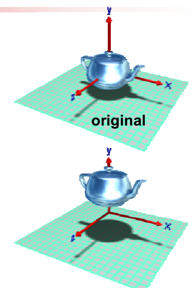
$$\begin{pmatrix} S_X & 0 & 0 & 0 \\ 0 & S_Y & 0 & 0 \\ 0 & 0 & S_Z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Translation

A translation vector $\mathbf{V}$ is defined by its components as $\mathbf{V} = a\mathbf{i} + b\mathbf{j} + c\mathbf{k}$
In homogeneous matrix form this is

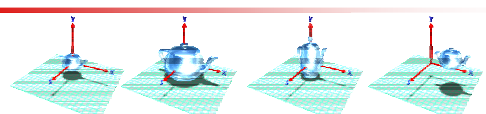
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

V=(a**i** + b**j** +c**k**) == (a, b, c)



translation along y, or V = (0, k, 0)

Scaling



Original | scale all axes | scale Y axis | offset from origin

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

3D Shearing

Shearing:

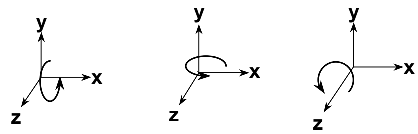
The change in each coordinate is a linear combination of all three

Transforms a cube into a general parallelepiped

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & a & b & 0 \\ c & 1 & d & 0 \\ e & f & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rotation

- In 2D, rotation is about a point
- In 3D, rotation is about a vector, which can be done through rotations about x, y or z axes
- Positive rotations are anti-clockwise, negative rotations are clockwise, when looking down a positive axis towards the origin



Major Axis Rotation Matrices

- about X axis

$$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- about Y axis

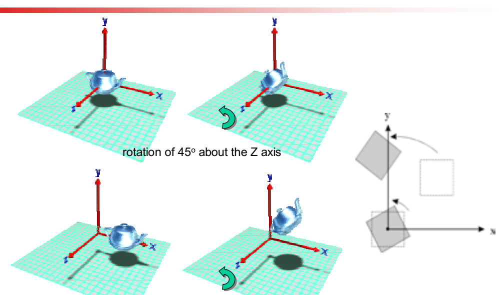
$$\mathbf{R}_y = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotations are orthogonal matrices, preserving distances and angles.

- about Z axis

$$\mathbf{R}_z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation

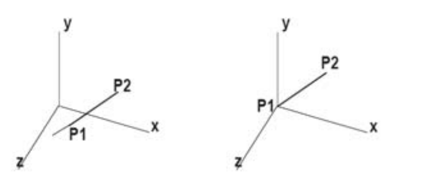


rotation of 45° about the Z axis

offset from origin translation then rotate about Z axis

Rotation Axis

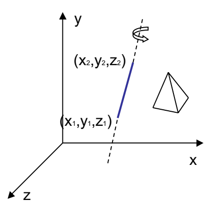
- In general rotation vector does not pass through origin



Translate P1 to origin to force the axis to pass through the origin

Rotation about an Arbitrary Axis

- Rotation about an Arbitrary Axis



Basic Idea

T
1. Translate (x_1, y_1, z_1) to the origin

R
2. Rotate (x'_2, y'_2, z'_2) on to the z axis
3. Rotate the object around the z-axis
4. Rotate the axis to the original orientation

R^{-1}

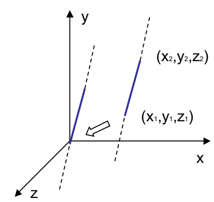
5. Translate the rotation axis to the original position

T^{-1}

$$[T_{R_{arb}}]=[T_{T_R}]^{-1}[T_{R_x(\alpha)}]^{-1}[T_{R_y(\phi)}]^{-1}[T_{R_z(\theta)}][T_{R_y(\phi)}][T_{R_x(\alpha)}][T_{T_R}]$$

Rotation about an Arbitrary Axis

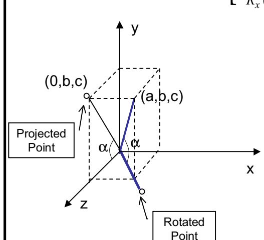
- Step 1. Translation



$$T_{T_R}=\begin{bmatrix}1 & 0 & 0 & -x_1\\ 0 & 1 & 0 & -y_1\\ 0 & 0 & 1 & -z_1\\ 0 & 0 & 0 & 1\end{bmatrix}$$

Rotation about an Arbitrary Axis

- Step 2. Establish $\left[T_{R_x(\alpha)}\right]$



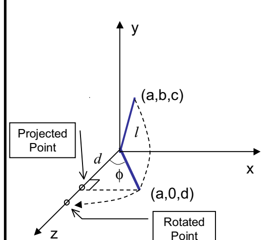
$$\sin\alpha=\frac{b}{\sqrt{b^2+c^2}}=\frac{b}{d}$$

$$\cos\alpha=\frac{c}{\sqrt{b^2+c^2}}=\frac{c}{d}$$

$$\left[T_{R_x(\alpha)}\right]=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \cos\alpha & -\sin\alpha & 0\\ 0 & \sin\alpha & \cos\alpha & 0\\ 0 & 0 & 0 & 1\end{bmatrix}=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & c/d & -b/d & 0\\ 0 & b/d & c/d & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

Rotation about an Arbitrary Axis

- Step 3. Rotate about *y* axis by ϕ



$$\sin\phi=\frac{a}{l},\qquad \cos\phi=\frac{d}{l}$$

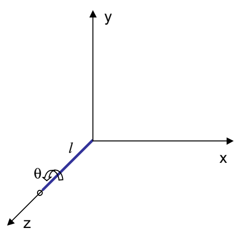
$$l^2=a^2+b^2+c^2=a^2+d^2$$

$$d=\sqrt{b^2+c^2}$$

$$\left[T_{R_y(\phi)}\right]=\begin{bmatrix}\cos\phi & 0 & -\sin\phi & 0\\ 0 & 1 & 0 & 0\\ \sin\phi & 0 & \cos\phi & 0\\ 0 & 0 & 0 & 1\end{bmatrix}=\begin{bmatrix}d/l & 0 & -a/l & 0\\ 0 & 1 & 0 & 0\\ a/l & 0 & d/l & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

Rotation about an Arbitrary Axis

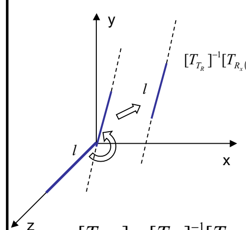
- Step 4. Rotate about *z* axis by the desired angle θ



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

Rotation about an Arbitrary Axis

- Step 5. Apply the reverse transformation to place the axis back in its initial position

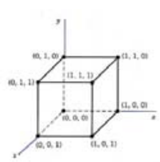


$$[T_{T_R}]^{-1}[T_{R_x(\alpha)}]^{-1}[T_{R_y(\phi)}]^{-1}=\begin{bmatrix}1 & 0 & 0 & x_1\\ 0 & 1 & 0 & y_1\\ 0 & 0 & 1 & z_1\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \cos\alpha & \sin\alpha & 0\\ 0 & -\sin\alpha & \cos\alpha & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\cos\phi & 0 & \sin\phi & 0\\ 0 & 1 & 0 & 0\\ -\sin\phi & 0 & \cos\phi & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$$[T_{R_{arb}}]=[T_{T_R}]^{-1}[T_{R_x(\alpha)}]^{-1}[T_{R_y(\phi)}]^{-1}[T_{R_z(\theta)}][T_{R_y(\phi)}][T_{R_x(\alpha)}][T_{T_R}]$$

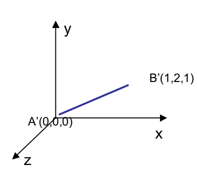
Rotation about an Arbitrary Axis

Find the new coordinates of a unit cube 90º-rotated about an axis defined by its endpoints A(2,1,0) and B(3,3,1).



Rotation about an Arbitrary Axis

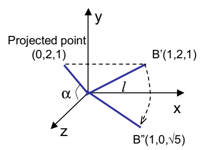
- Step1. Translate point A (2,1,0) to the origin



$$[T_{T_R}]=\begin{bmatrix}1 & 0 & 0 & -2\\ 0 & 1 & 0 & -1\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

Rotation about an Arbitrary Axis

- Step 2. Rotate axis *A'B'* about the x axis by and angle α, until it lies on the *xz* plane.



$$\sin\alpha = \frac{2}{\sqrt{2^2+1^2}} = \frac{2}{\sqrt{5}} = \frac{2\sqrt{5}}{5}$$

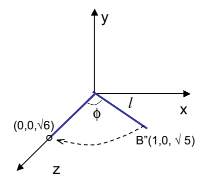
$$\cos\alpha = \frac{1}{\sqrt{5}} = \frac{\sqrt{5}}{5}$$

$$l = \sqrt{1^2+2^2+1^2} = \sqrt{6}$$

$$\left[T_{R_x(\alpha)}\right] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{\sqrt{5}}{5} & -\frac{2\sqrt{5}}{5} & 0 \\ 0 & \frac{2\sqrt{5}}{5} & \frac{\sqrt{5}}{5} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation about an Arbitrary Axis

- Step 3. Rotate axis *A'B''* about the y axis by and angle ϕ, until it coincides with the z axis.



$$\sin\phi = \frac{1}{\sqrt{6}} = \frac{\sqrt{6}}{6}$$

$$\cos\phi = \frac{\sqrt{5}}{\sqrt{6}} = \frac{\sqrt{30}}{6}$$

$$\left[T_{R_y(\phi)}\right] = \begin{bmatrix} \frac{\sqrt{30}}{6} & 0 & -\frac{\sqrt{6}}{6} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{\sqrt{6}}{6} & 0 & \frac{\sqrt{30}}{6} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation about an Arbitrary Axis

- Step 4. Rotate the cube 90˚ about the z axis

$$\left[T_{R_z(90°)}\right] = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Finally, the concatenated rotation matrix about the arbitrary axis AB becomes,

$$[T_{R_{arb}}] = [T_{T_R}]^{-1}[T_{R_x(\alpha)}]^{-1}[T_{R_y(\phi)}]^{-1}[T_{R_z(90^\circ)}][T_{R_y(\phi)}][T_{R_x(\alpha)}][T_{T_R}]$$

Rotation about an Arbitrary Axis

$$[T_{R_{arb}}] = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{\sqrt{5}}{5} & \frac{2\sqrt{5}}{5} & 0 \\ 0 & -\frac{2\sqrt{5}}{5} & \frac{\sqrt{5}}{5} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{\sqrt{30}}{6} & 0 & \frac{\sqrt{6}}{6} & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{\sqrt{6}}{6} & 0 & \frac{\sqrt{30}}{6} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \frac{\sqrt{30}}{6} & 0 & -\frac{\sqrt{6}}{6} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{\sqrt{6}}{6} & 0 & \frac{\sqrt{30}}{6} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{\sqrt{5}}{5} & -\frac{2\sqrt{5}}{5} & 0 \\ 0 & \frac{2\sqrt{5}}{5} & \frac{\sqrt{5}}{5} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -2 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.166 & -0.075 & 0.983 & 1.742 \\ 0.742 & 0.667 & 0.075 & -1.151 \\ -0.650 & 0.741 & 0.167 & 0.560 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation about an Arbitrary Axis

- Multiplying $[T_R]_{AB}$ by the point matrix of the original cube

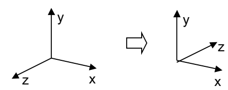
$$[P^*] = [T_{R_{arb}}] \cdot [P]$$

$$[P^*] = \begin{bmatrix} 0.166 & -0.075 & 0.983 & 1.742 \\ 0.742 & 0.667 & 0.075 & -1.151 \\ -0.650 & 0.741 & 0.167 & 0.560 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2.650 & 1.667 & 1.834 & 2.816 & 2.725 & 1.742 & 1.909 & 2.891 \\ -0.558 & -0.484 & 0.258 & 0.184 & -1.225 & -1.151 & -0.409 & -0.483 \\ 1.467 & 1.301 & 0.650 & 0.817 & 0.726 & 0.560 & -0.091 & 0.076 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Rotation about an Arbitrary Axis

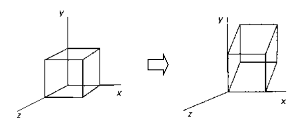
- Reflection Relative to the xy Plane



$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \boxed{-1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- Z-axis Shear

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & \boxed{a} & 0 \\ 0 & 1 & \boxed{b} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



Q1 - Translate by <1, 1, 1>

- A translation by an offset (tx, ty, tz) is achieved using the following matrix:

$$M_T(t_x, t_y, t_z) = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- So to translate by a vector (1, 1, 1), the matrix is simply:

$$M_T(1,1,1) = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Q2- Rotate by 45 degrees about x axis

- So to rotate by 45 degrees about the x-axis, we use the following matrix:

$$\mathbf{R}_x(45) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos 45 & -\sin 45 & 0 \\ 0 & \sin 45 & \cos 45 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Q3 - Rotate by 45 about axis <1, 1, 1>

- So a rotation by 45 degrees about <1, 1, 1> can be achieved by a few succesive rotations about the major axes. Which can be represented as a single composite transformation

$$n_x = 1,\quad n_y = 1,\quad n_z = 1 \quad \textbf{SO} \quad \begin{cases} d = \sqrt{n_x^{\,2} + n_z^{\,2}} = \sqrt{2} = 1.414 \\ \beta = \tan^{-1}\dfrac{n_y}{d} = \tan^{-1}\dfrac{1}{\sqrt{2}} = 35.264 \\ \alpha = \tan^{-1}\dfrac{n_x}{n_z} = \tan^{-1}\dfrac{1}{1} = 45 \end{cases}$$

Q3 - Arbitrary Axis Rotation

- The composite transformation can then be obtained as follows:

$$M_R(1,1,1) = R_y^{-1}(\alpha) \bullet R_x^{-1}(\beta) \bullet R_z(\theta) \bullet R_x(\beta) \bullet R_y(\alpha)$$

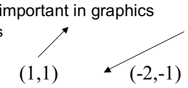
$$= \begin{bmatrix} \cos(-45) & 0 & \sin(-45) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(-45) & 0 & \cos(-45) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(-35.2) & -\sin(-35.2) & 0 \\ 0 & \sin(-35.2) & \cos(-35.2) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(45) & -\sin(45) & 0 & 0 \\ \sin(45) & \cos(45) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(35.2) & -\sin(35.2) & 0 \\ 0 & \sin(35.2) & \cos(35.2) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(45) & 0 & \sin(45) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(45) & 0 & \cos(45) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad ??$$

x
y
z

Directions vs. Points

- We have looked at transforming points
- Directions are also important in graphics
  - Viewing directions
  - Normal vectors
  - Ray directions



- Directions are represented by vectors, like points, and can be transformed, but not like points
- Say we define a direction as the difference of two points: d=a–b. This represents the direction of the line between two points
- Now we translate the points by the same amount: a'=a+t, b'=b+t
- Have we transformed d?

Homogeneous Directions

- Translation does not affect directions!
- Homogeneous coordinates give us a clear way of handling this, e.g., direction (x,y) becomes homogeneous direction (x,y,0), and remains the same after translation:

$$\begin{bmatrix} 1 & 0 & b_x \\ 0 & 1 & b_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 0 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 0 \end{bmatrix}$$

- (x, y, 0) is a vector, (x,y,1) is a point.
- The same applies to rotation and scaling, e.g., scaling changes the length of vector, but not direction
- Normal vectors are slightly different though (can't always use the matrix for points to transform the normal vector)

Alternative Rotations

- Specify the rotation axis and the angle (OpenGL method)
- Euler angles: Specify how much to rotate about X, then how much about Y, then how much about
- These are hard to think about, and hard to compose
- Quaternions
  - 4-vector related to axis and angle, unit magnitude, e.g., rotation about axis (nx,ny,nz) by angle θ:

$$\left(n_x\cos(\theta/2), n_y\cos(\theta/2), n_z\cos(\theta/2), \sin(\theta/2)\right)$$

  - Only normalized quaternions represent rotations, but you can normalize them just like vectors, so it isn't a problem
  - But we don't want to learn all the maths about quaternions in this module, because we have to learn how to create a basic application before trying to make rotation faster

OpenGL Transformations

- OpenGL internally stores two matrices that control viewing of the scene
  - The GL_MODELVIEW matrix for modelling and world to view transformations
  - The GL_PROJECTION matrix captures the view to canonical conversion
  - Mapping from canonical view volume into window space is through a glViewport function call
- Matrix calls, such as glRotate, glTranslate, glScale right multiply the transformation matrix M with the current matrix C (e.g., identity matrix initially), resulting in CM - the last one is the first applied