Mathematical Methods and Graphics

Lecture 3:

Transformations of 3D Worlds

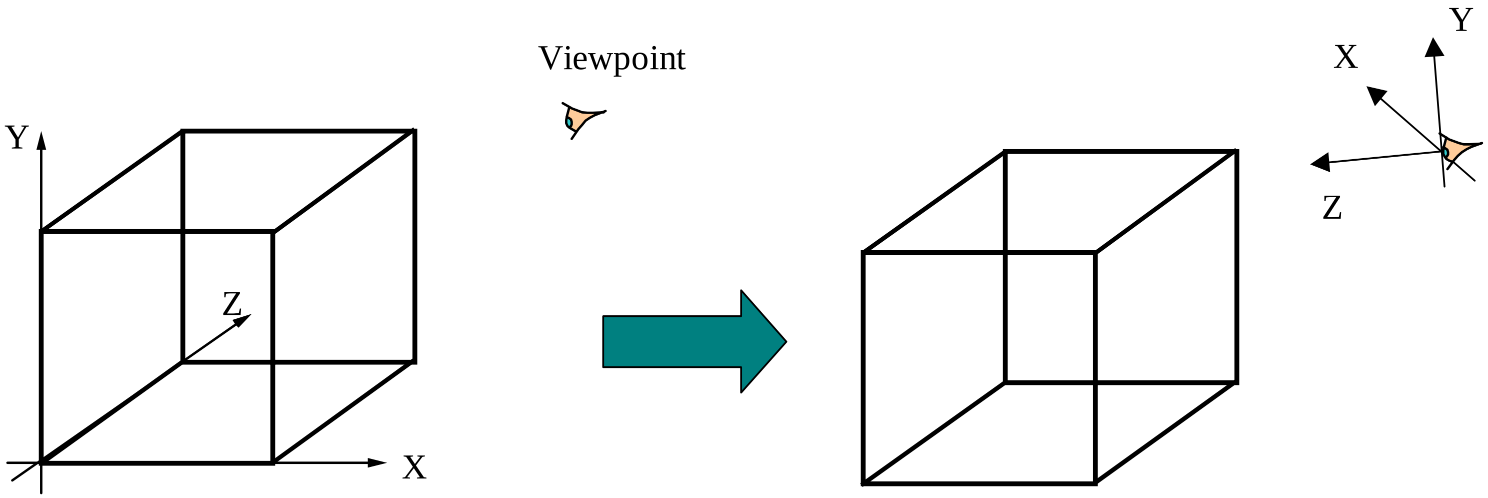
Lecture Overview

- Viewpoint Transformation
- Matrix Transformations: Translation and Scaling
- Homogeneous Coordinates
- Combined Transformations
- Rotation and their Signs
- Inverse Transformations
- Flying Sequences
- Projection by Matrix Multiplication
- Normalisation

The Need for Transformations

- Graphics scenes are defined in “world” coordinates
- We want to be able to look at a graphics scene from any angle
- To draw a graphics scene we need the viewpoint to be the origin and the z axis to be the direction of view
- Hence we need to be able to transform the coordinates of a graphics scene.

Transformation of viewpoint



Coordinate System for definition

Coordinate System for viewing

Matrix transformations of points

To transform points we use matrix multiplications.

For example to make an object at the origin twice as big we could use:

$$[x',y',z'] = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

yields

$$x' = 2x \qquad y' = 2y \qquad z' = 2z$$

Translation by Matrix multiplication

- Many of our transformations will require translation of the points.

- For example if we want to move all the points two units along the x axis we would require:

$$x' = x + 2$$
$$y' = y$$
$$z' = z$$

But how can we do this with a matrix?

Homogenous Coordinates

- The answer is to add a fourth dimension
- This representation is called *Homogeneous Coordinates*
- Using homogeneous coordinates, the translation (2,0,0) is represented as:

$$[x', y', z', 1] = [x, y, z, 1] \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \end{pmatrix}$$

General Homogenous Coordinates

In most cases the last ordinate will be 1, but in general we can consider it a scale factor.

Thus:

[x, y, z, s] is equivalent to [x/s, y/s, z/s]

Homogenous Cartesian

Translation by vector **T**

$$[\, x, y, z, 1] \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ tx & ty & tz & 1 \end{pmatrix} = [x+tx, y+ty, z+tz, 1\,]$$

Scaling by scaling vector **S**

$$[x, y, z, 1] \begin{pmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = [sx*x, sy*y, sz*z, 1]$$

Combining transformations

- Suppose we want to make an object at the origin twice as big and then move it to a point [5, 5, 20].

- The transformation is a scaling followed by a translation:

$$[x',y',z',1] = [x, y, z, 1] \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 5 & 5 & 20 & 1 \end{pmatrix}$$

Combined transformations

Multiply out the transformation matrices first, then transform the points

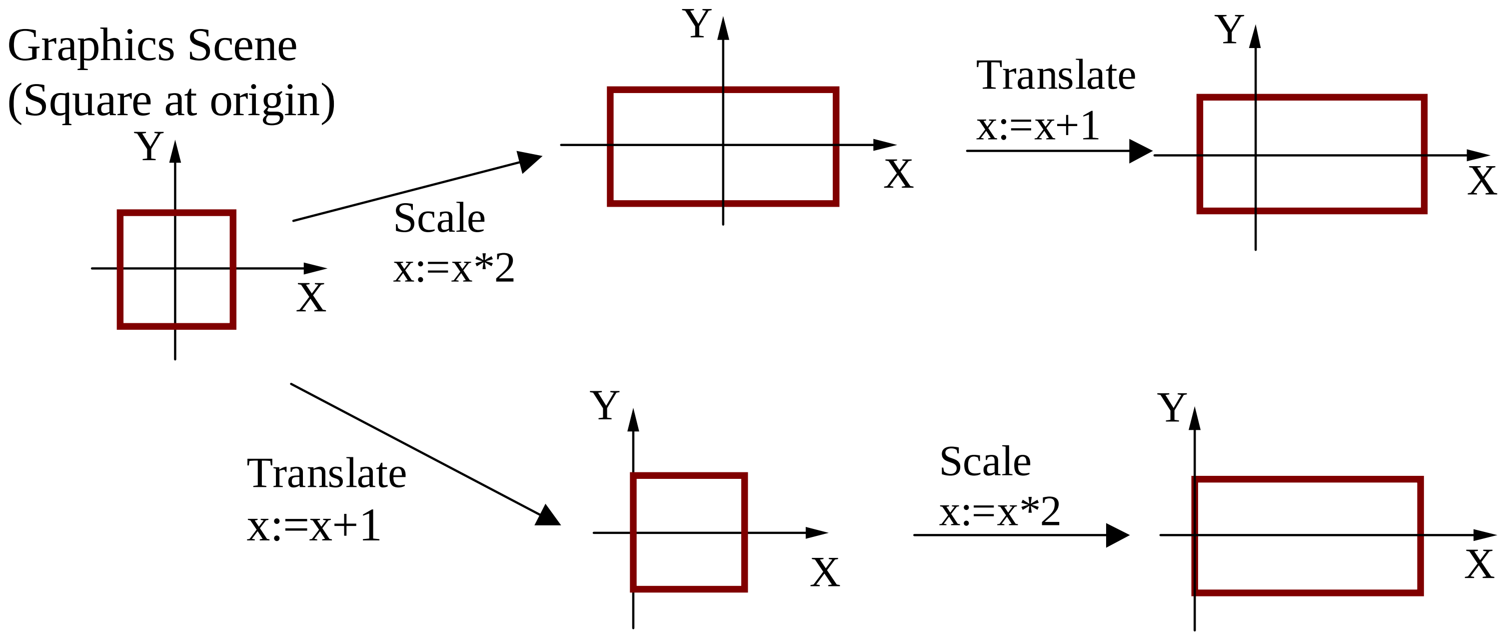
$$[x',y',z',1] = [x, y, z, 1] \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 5 & 5 & 20 & 1 \end{pmatrix}$$

Transformations are not commutative

The order in which transformations are applied matters:

In general

$T * S$ is not the same as $S * T$


Graphics Scene
(Square at origin)
Y
X
Scale
x:=x*2
Y
X
Translate
x:=x+1
Y
X
Translate
x:=x+1
Y
X
Scale
x:=x*2
Y
X

Rotation

To define a rotation we need an axis.

The simplest rotations are about the Cartesian axes

e.g.

Rx - Rotate about the X axis

Ry - Rotate about the Y axis

Rz - Rotate about the Z axis

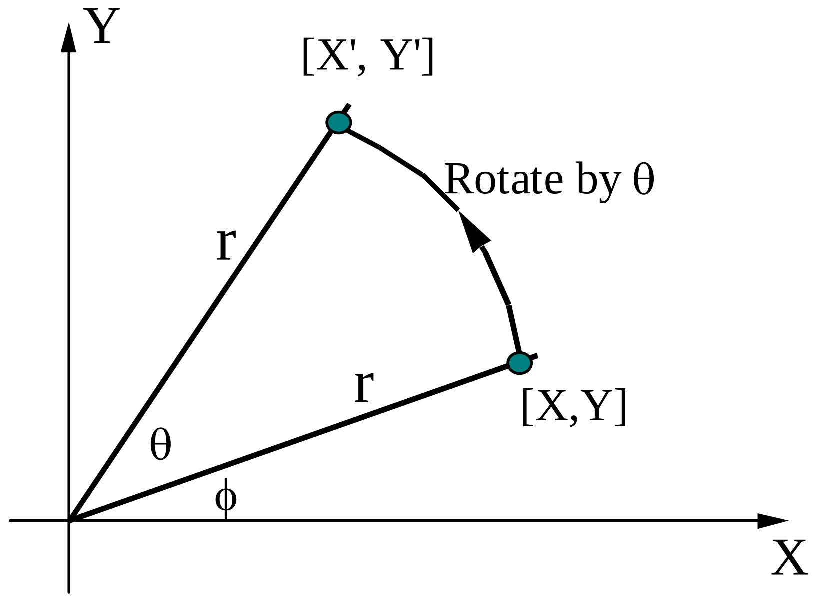
Rotation Matrices

$$\boldsymbol{R}x = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) & 0 \\ 0 & -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\boldsymbol{R}y = \begin{pmatrix} \cos(\theta) & 0 & -\sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\boldsymbol{R}z = \begin{pmatrix} \cos(\theta) & \sin(\theta) & 0 & 0 \\ -\sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Deriving Rz



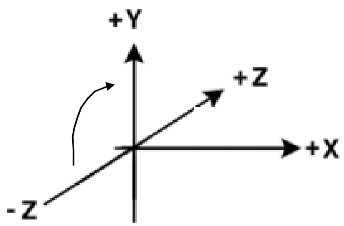
$$[X,Y] = [r\,\mathrm{Cos}\phi,\ r\,\mathrm{Sin}\phi]$$

$$\begin{aligned}[X',Y'] &= [\ r\,\mathrm{Cos}(\theta+\phi)\ ,\ r\,\mathrm{Sin}(\theta+\phi)\ ] \\ &= [\ r\,\mathrm{Cos}\phi\,\mathrm{Cos}\theta - r\mathrm{Sin}\phi\,\mathrm{Sin}\theta,\ \ r\mathrm{Sin}\phi\mathrm{Cos}\theta +\ r\mathrm{Cos}\phi\mathrm{Sin}\theta\ ] \\ &= [\ X\,\mathrm{Cos}\theta - Y\,\mathrm{Sin}\theta,\ Y\mathrm{Cos}\theta + X\mathrm{Sin}\theta] \\ &= [\ X\ \ Y\ ] \begin{pmatrix} \mathrm{Cos}\theta & \mathrm{Sin}\theta \\ -\mathrm{Sin}\theta & \mathrm{Cos}\theta \end{pmatrix}\end{aligned}$$

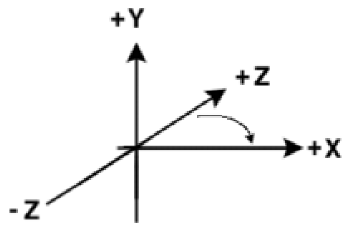
Signs of Rotations

Our coordinate system is left-handed: the positive x, y and z axes point right, up and forward, respectively.

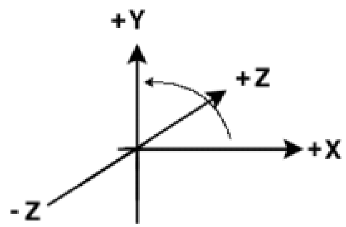
Positive rotation is clockwise about the axis of rotation (seen from the positive side of the axis!)



Positive X rotation



Positive Y rotation



Positive Z rotation

Inverting a translation

Since we know what transformation matrices do, we can write down their inversions directly

For example:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ tx & ty & tz & 1 \end{pmatrix} \text{ has inversion } \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -tx & -ty & -tz & 1 \end{pmatrix}$$

Inverting scaling

$$\begin{pmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \text{ has inversion } \begin{pmatrix} 1/sx & 0 & 0 & 0 \\ 0 & 1/sy & 0 & 0 \\ 0 & 0 & 1/sz & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Inverting Rotation

Inverting a rotation by an angle θ is equivalent to rotating through an angle of $-\theta$:

$$\mathrm{Cos}(-\theta) = \mathrm{Cos}(\theta)$$

and

$$\mathrm{Sin}(-\theta) = -\mathrm{Sin}(\theta)$$

Inverting Rz

$$\begin{pmatrix} \cos(\theta) & \sin(\theta) & 0 & 0 \\ -\sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \text{ has inversion } \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Problem

A graphics scene is to be transformed by:

1. Moving every point by 2 units in the positive z direction
2. Rotating the scene through 90º about the z-axis

   (the rotation direction is clockwise when looking from the positive side of the z axis).

(a) What is the transformation matrix?

(b) What is its inverse?

Solution (a) - Transformation Matrix

The transformation is made up of a translation followed by a rotation, so:

$$\mathbf{P'} = \mathbf{P} \cdot \boldsymbol{T} \cdot \boldsymbol{R}$$

$$[x',y',z',1] = [x, y, z, 1] \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 2 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

So the transformation matrix is:

$$[x',y',z',1] = [x, y, z, 1] \begin{pmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 2 & 1 \end{pmatrix}$$

Solution (b) - Inverse transformation

The transformation is made up of a translation followed by a rotation, so:

$$P' = P \cdot R^{-1} \cdot T^{-1}$$

$$[x',y',z',1] = [x, y, z, 1] \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -2 & 1 \end{pmatrix}$$

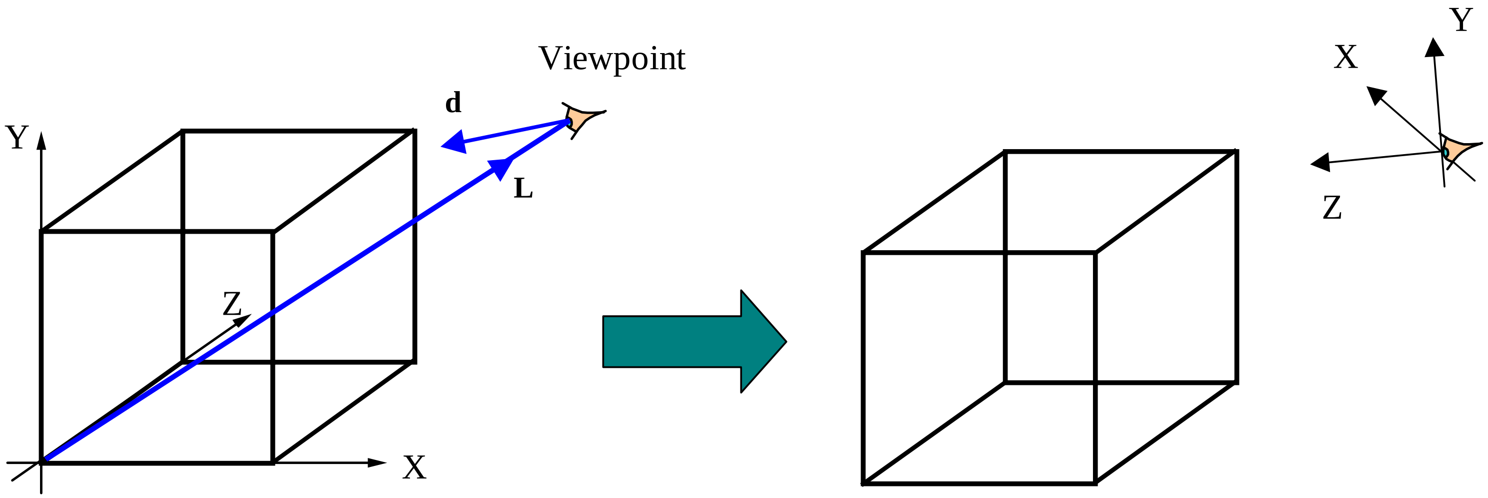
So the inverse transformation matrix is:

$$[x',y',z',1] = [x, y, z, 1]\begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -2 & 1 \end{pmatrix}$$

Flying Sequences

- We now return to the question of transforming the origin of a graphics scene.
- This would be used in generating animated flying sequences where the viewpoint moves round the scene.
- Let the required viewpoint be **L** = [Lx,Ly,Lz] and the required view direction be **d** = [dx,dy,dz]. Let |**d**| = 1

Transformation of viewpoint



Coordinate System for definition

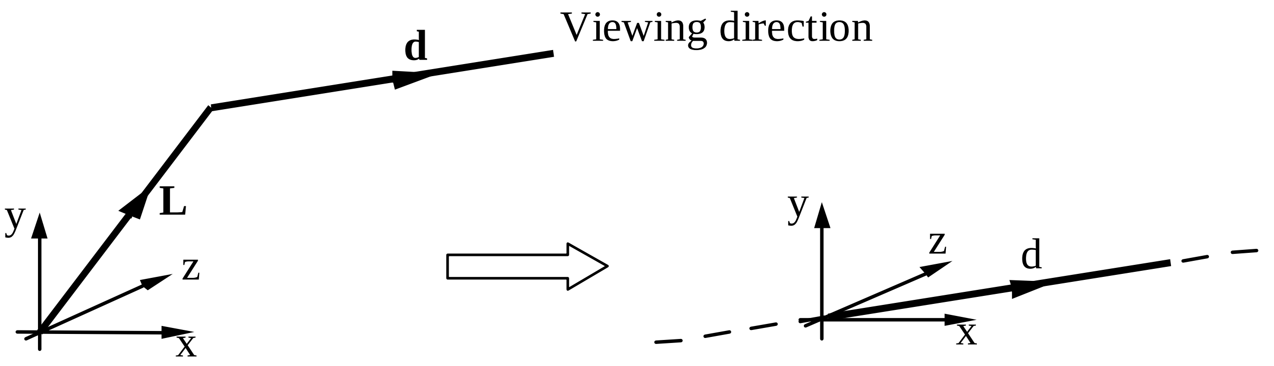
Coordinate System for viewing

Flying Sequences

The required transformation is in three parts:

1. Translation of the Origin
2. Rotate about Y
3. Rotate about X

Translation of the Origin



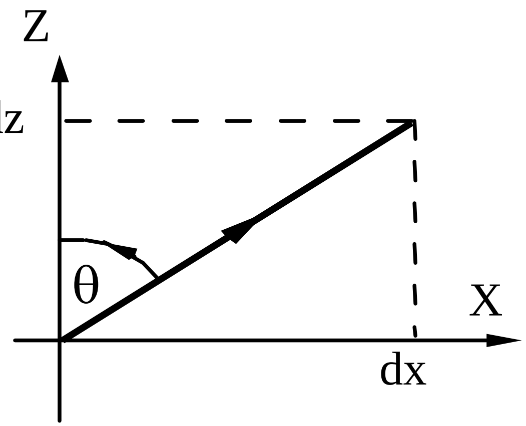
Step 1: Move origin to the required viewpoint

$$\boldsymbol{A} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -L_x & -L_y & -L_z & 1 \end{pmatrix}$$

Surgical Robotics & Imaging

Imperial College London

Rotate about Y until dx = 0

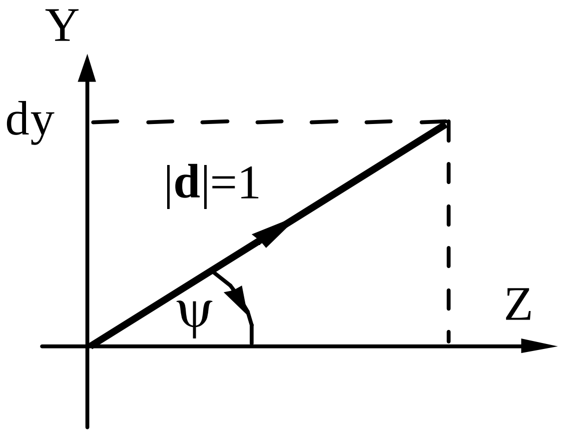


Step 2: Rotate about Y

Cos θ= dz/√(dx*dx + dz*dz)
Sin θ = dx/√(dx*dx + dz*dz)

$$\boldsymbol{B} = \begin{pmatrix} d_z/v & 0 & d_x/v & 0 \\ 0 & 1 & 0 & 0 \\ -d_x/v & 0 & d_z/v & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotate about X until dy = 0



Step 3: Rotate about X

Cos ψ = √(dx*dx+dz*dz)/|d|
Sin ψ = dy/|**d**| = dy

$$\mathbf{C} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & v & d_y & 0 \\ 0 & -d_y & v & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Combining the matrices

The matrix that transforms the origin is:

$$\boldsymbol{T} = \boldsymbol{A} * \boldsymbol{B} * \boldsymbol{C}$$

for every point in the graphics scene we calculate:

$$\mathbf{P'} = \mathbf{P} * \boldsymbol{T}$$

Projection by Matrix multiply

Usually projection and drawing of a scene comes after transformation.

It is therefore convenient to **combine the projection with the other parts** of the transformation

Orthographic Projection Matrix

For orthographic projections we simply drop the z coordinate

$$M_o = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Perspective Projection Matrix

$$[x,y,z,1] \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1/f \\ 0 & 0 & 0 & 0 \end{pmatrix} = [x,y,z,z/f]$$

f: normal distance from the centre of the world to the plane of projection

Normalisation

- Homogenous coordinates need to be normalised, so we need to divide by the last ordinate as a final step:

[x,y,z,z/f] is normalised to [x*f/z, y*f/z, f, 1]

as required by perspective projection

Projection matrices are singular

- Notice that projection matrices are singular (they cannot be inverted)
- This is because a projection cannot be inverted, ie projection is non-reversible
- Given a single 2D image, we cannot in general reconstruct the 3D original.