University of Colorado
Boulder

Introduction to Virtual Reality

Linear Algebra for 3D Graphics

Professor **Dan Szafir**

Computer Science & ATLAS Institute
University of Colorado Boulder

Goal: specify geometry, positions, translations, rotations, etc. in 3D space

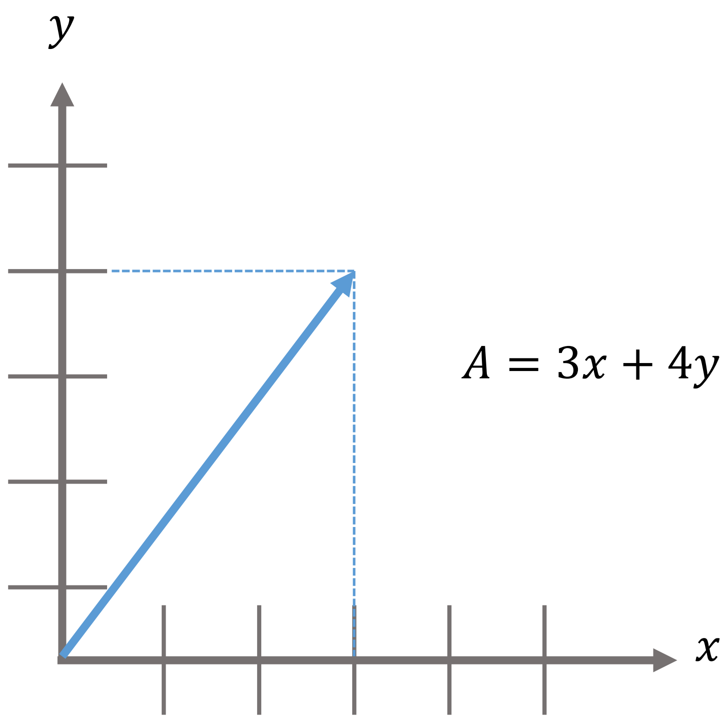
Terminology & Operations

Points and Vectors

Point: a position in space
Vector: an oriented line segment
Written as $\vec{a}$ or in bold (**a**)
Magnitude (norm): $||\vec{a}||$
Vectors have **length** and **direction**
- Absolute position not important

Stores offsets, displacements, locations
Often used to represent a position (but requires an origin)

Vectors and Coordinate Systems



$$A = \begin{pmatrix} x \\ y \end{pmatrix}$$

$$A^T = \begin{pmatrix} x & y \end{pmatrix}$$

$$\|A\| = \sqrt{x^2 + y^2}$$

Vector Operations

Vector addition

- Geometric perspective: parallelogram rule
- Cartesian perspective: add coordinate components

Vector scaling

Vector multiplication

- Dot (scalar) product
- Cross product

Coordinate Systems and Basis Vectors

Define a (2D) coordinate system with

- A reference point O (origin)
- 2 basis vectors $\vec{u}, \vec{v}$ (for now assume they are not parallel)

Translating between coordinate systems

- How to convert from $\binom{a}{b}$ in $O\ \vec{u}, \vec{v}$ to $\binom{a'}{b'}$ in $O'\ \overrightarrow{u'}, \overrightarrow{v'}$

$$\begin{pmatrix} a \\ b \end{pmatrix} = \vec{t} + M \begin{pmatrix} a' \\ b' \end{pmatrix}$$

Vectors and Matrices

Vector

Matrix

Index

$$\begin{bmatrix} 4 \\ 47 \\ 5 \\ 68 \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix} \begin{array}{l} \\ \leftarrow \text{second row} \\ \\ \\ \\ \end{array}$$

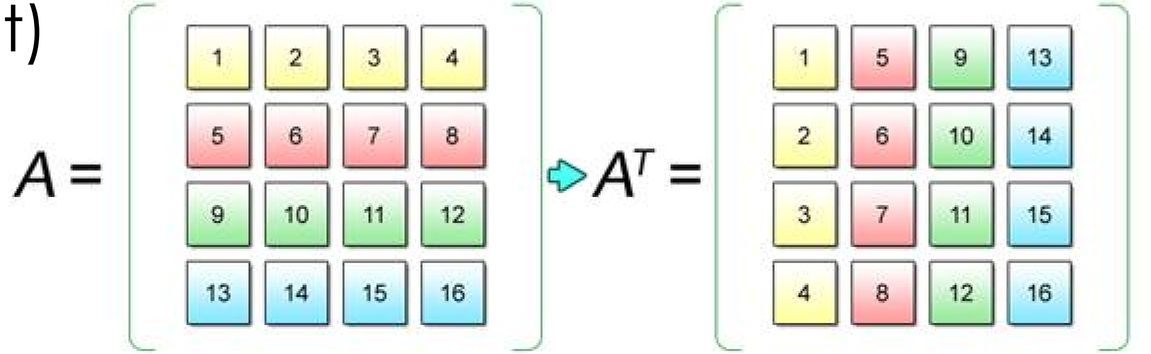
$$\uparrow$$

third column

Matrix Operations

Main ones we will use:

- **Transpose**
- Dot product
- Multiplication (matrix product)



Others:

- Addition
- Subtraction
- Etc.

Matrix Operations

Main ones we will use:

- Transpose
- **Dot product**
- Multiplication (matrix product)

$$\begin{bmatrix} a & b \end{bmatrix} \bullet \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax + by \end{bmatrix}$$

Others:

- Addition
- Subtraction
- Etc.

Matrix Operations

Main ones we will use:

- Transpose
- Dot product
- **Multiplication (matrix product)**

Others:

- Addition
- Subtraction
- Etc.

$$\begin{bmatrix} a & b \end{bmatrix} \bullet \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax + by \end{bmatrix}$$

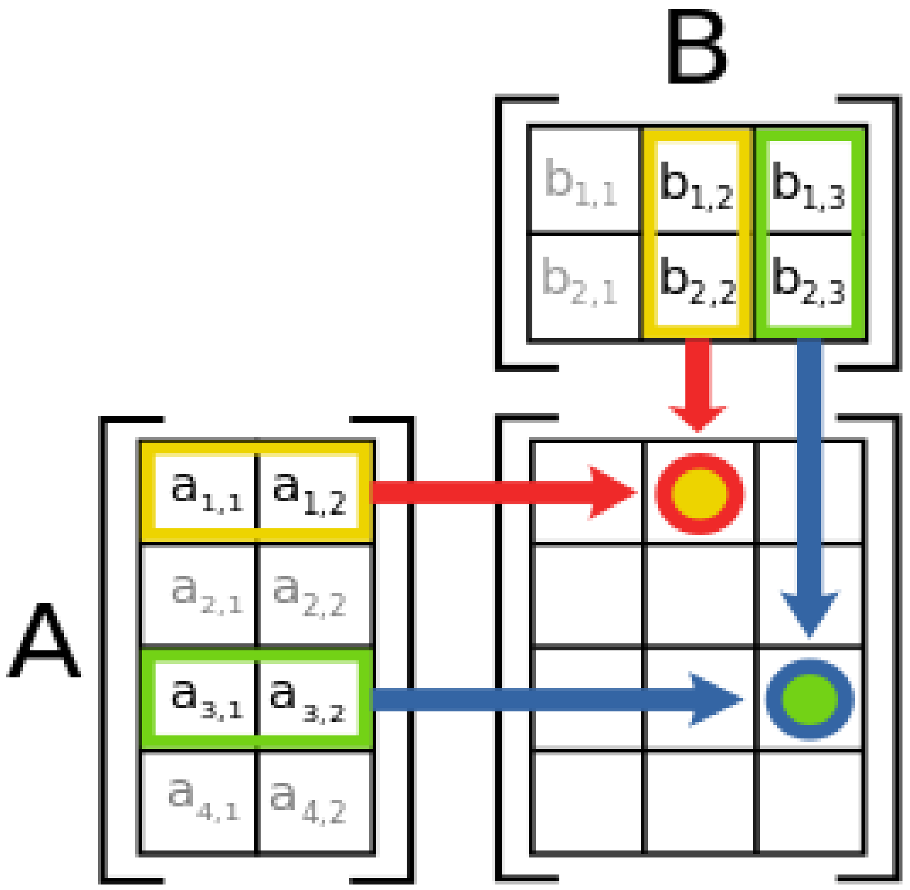
$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \bullet \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax + by \\ cx + dy \end{bmatrix}$$

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \bullet \begin{bmatrix} w & x \\ y & z \end{bmatrix} = \begin{bmatrix} aw + by & ax + bz \\ cw + dy & cx + dz \end{bmatrix}$$

Operations

Main ones we will use:

- Transpose
- Dot product
- **Multiplication (matrix product)**



Others:

- Addition
- Subtraction
- Etc.

Why is this important for us?

Must specify objects in 3 dimensions!

GPU designed for matrix operations

Matrices provide a simple way of understanding **transformations**

Transformations

Motivation

Many different coordinate systems
- World (scene), model, camera, etc.

Must transform between coordinate systems to ensure everything appears as intended
- Objects are at the correct location in the world
- Can view objects from different angles
- May want to move objects around (animation)
- May want to scale objects (perspective)

General Idea

Object in model coordinates
Transform into world coordinates
Represent points (vertices) in an object as vectors
Multiply by matrices to achieve desired results

Examples:
http://jsbin.com/satunaromo/edit?html,js,output
http://jsbin.com/zanurugena/edit?html,js,output
http://jsbin.com/wovupusife/edit?js,output

See the JavaScript Matrix Transformation Examples on Moodle

Outline

2D transformations: rotation, scale, shear

Composing transforms

3D rotations

Homogeneous Coordinates

General transformation Matrix

Other options for 3D rotations

- Euler angles
- Axis-angle
- Quaternions

Scale

$$Scale(s_x, s_y) = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix}$$

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \bullet \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax + by \\ cx + dy \end{bmatrix}$$

Example: 2D reflection about y-axis:

$$Scale(-1, 1) = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$$

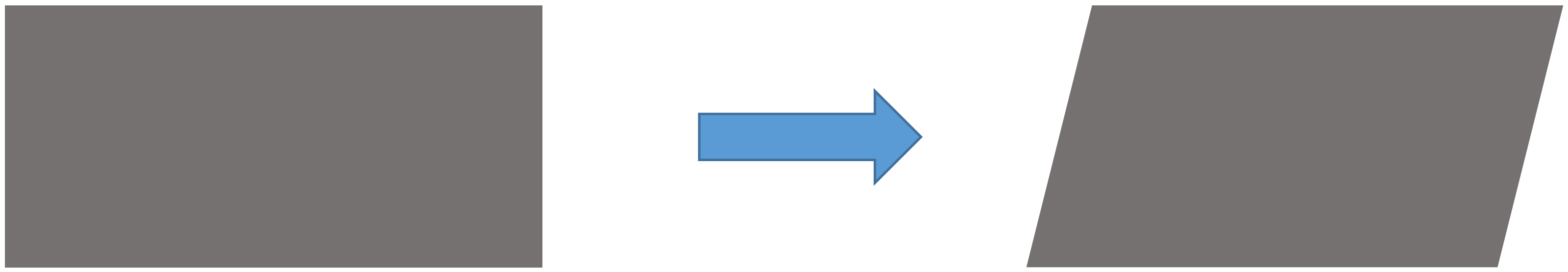
What would this be in 3 dimensions?

Shear

$$Shear = \begin{pmatrix} 1 & a \\ 0 & 1 \end{pmatrix}$$

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \bullet \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax + by \\ cx + dy \end{bmatrix}$$

What if you wanted to shear vertically?



2D Rotations

$$Rotate(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \bullet \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax + by \\ cx + dy \end{bmatrix}$$

Translations

Example: move x by +5, leave y and z unchanged

Need matrix:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \bullet \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax + by \\ cx + dy \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = M \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x + 5 \\ y \\ z \end{bmatrix}$$

Translations

Example: move x by +5, leave y and z unchanged

Need matrix:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \bullet \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax + by \\ cx + dy \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = M \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x + 5 \\ y \\ z \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{pmatrix} 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} x + 5 \\ y \\ z \\ w \end{bmatrix}$$

Homogenous Coordinates

$$\begin{array}{rcll}
\text{Homogeneous} & & \text{Cartesian} & \\
(1,2,3) & \Rightarrow & \left(\frac{1}{3},\frac{2}{3}\right) & \\
(2,4,6) & \Rightarrow & \left(\frac{2}{6},\frac{4}{6}\right) & = \left(\frac{1}{3},\frac{2}{3}\right) \\
(4,8,12) & \Rightarrow & \left(\frac{4}{12},\frac{8}{12}\right) & = \left(\frac{1}{3},\frac{2}{3}\right) \\
\vdots & & \vdots & \\
(1a,2a,3a) & \Rightarrow & \left(\frac{1a}{3a},\frac{2a}{3a}\right) & = \left(\frac{1}{3},\frac{2}{3}\right)
\end{array}$$

$$(x, y, w) \quad \leftrightarrow \quad (\frac{x}{w}, \frac{y}{w})$$

Homogeneous Cartesian

Homogeneous Transformation Matrix

Homogeneous: combine rotation and translation into a single matrix!

For 2D: Always 3 x 3!

For 3D: Always 4 x 4!

This does the displacement

This does the rotation

$$= \begin{bmatrix} n_x & s_x & a_x & d_x \\ n_y & s_y & a_y & d_y \\ n_z & s_z & a_z & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Example 1: Just Translation

Amount to
translate by

$$\begin{pmatrix} 1 & 0 & 0 & Tx \\ 0 & 1 & 0 & Ty \\ 0 & 0 & 1 & Tz \\ 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} = \begin{pmatrix} (1*x)+(Tx*w) \\ (1*y)+(Ty*w) \\ (1*z)+(Tz*w) \\ w \end{pmatrix}$$

Coordinate we
want to translate
(e.g., a vertex)

Position of new
coordinate

Example 1: Just Translation

Amount to translate by

$$\begin{bmatrix} 1 & 0 & 0 & 10 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 10 \\ 10 \\ 10 \\ 1 \end{bmatrix} = \begin{bmatrix} 1*10+0*10+0*10+10*1 \\ 0*10+1*10+0*10+0*1 \\ 0*10+0*10+1*10+0*1 \\ 0*10+0*10+0*10+1*1 \end{bmatrix} = \begin{bmatrix} 10+0+0+10 \\ 0+10+0+0 \\ 0+0+10+0 \\ 0+0+0+1 \end{bmatrix} = \begin{bmatrix} 20 \\ 10 \\ 10 \\ 1 \end{bmatrix}$$

Coordinate we want to translate (e.g., a vertex)

Position of new coordinate

Example 2: Scale

$$\begin{pmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} x * sx \\ y * sy \\ z * sz \\ w \end{bmatrix}$$

Example 2: Scale

$$\begin{bmatrix} .8 & 0 & 0 & 0 \\ 0 & .5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 150 \\ 150 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 120 + 0 + 0 + 0 \\ 0 + 75 + 0 + 0 \\ 0 + 0 + 1 + 0 \\ 0 + 0 + 0 + 1 \end{bmatrix} = \begin{bmatrix} 120 \\ 75 \\ 1 \\ 1 \end{bmatrix}$$

Example 3: Rotation

Rotation is more tricky!

What axis do you want to rotate around?

X-axis

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Y-axis

$$\begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

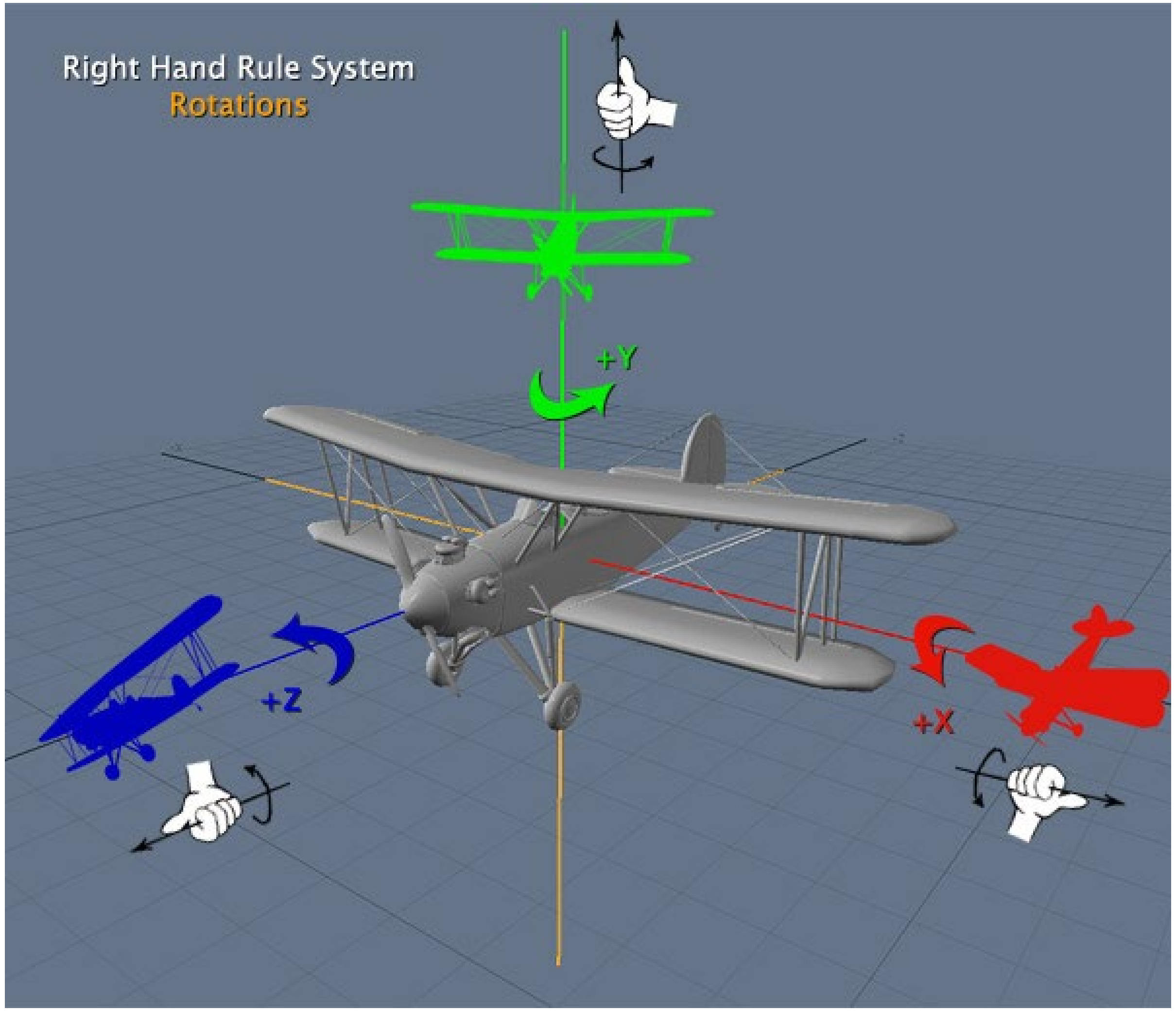
Z-axis

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Which way is positive?

Normally:

- Use the right-hand rule



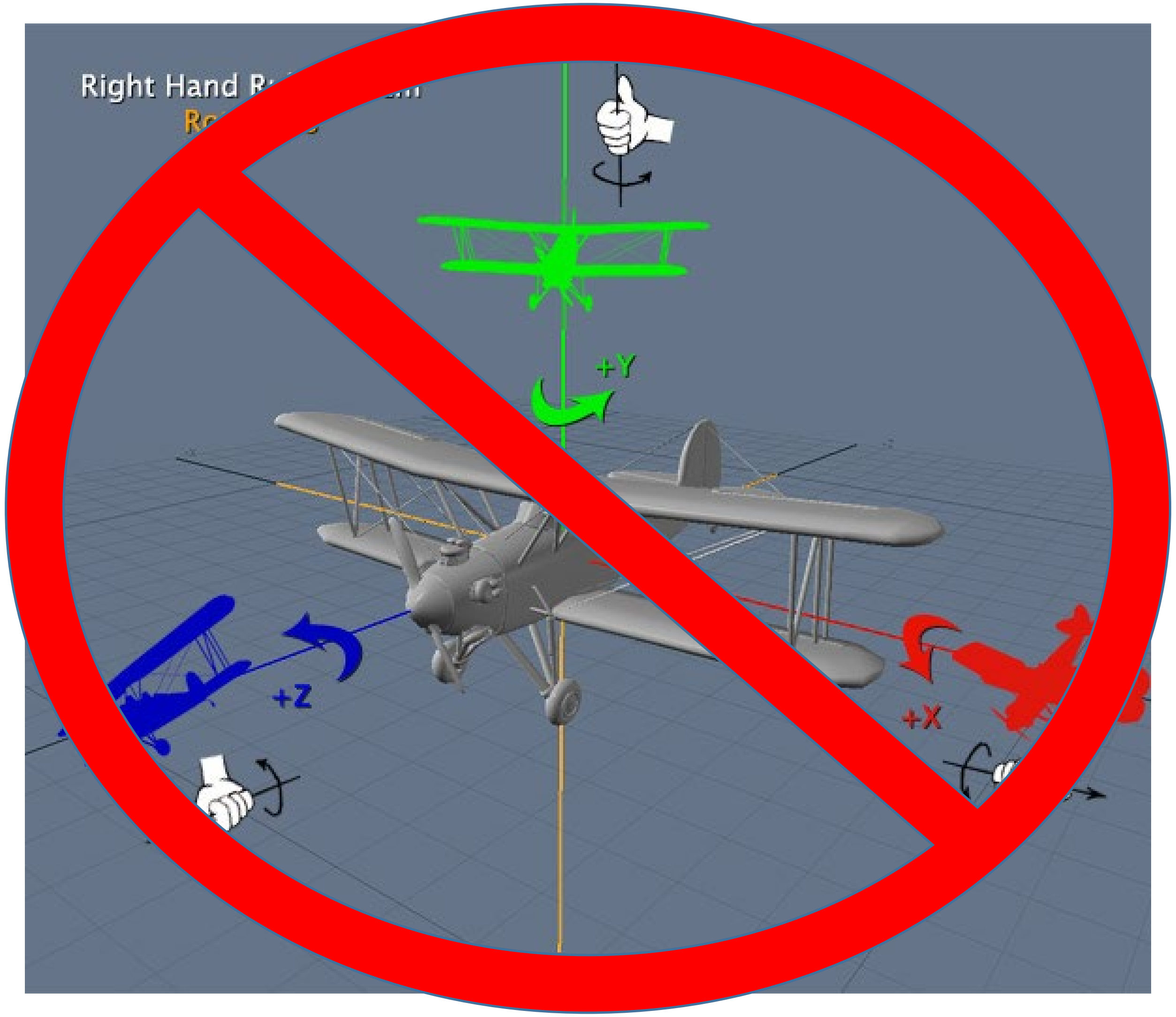
Which way is positive?

Normally:

Use the right-hand rule

Unity:

Left-hand rule



Cheat Sheet

X-Rotation in 3D

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi & 0 \\ 0 & \sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Z-Rotation in 3D

$$\begin{bmatrix} \cos\phi & -\sin\phi & 0 & 0 \\ \sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Scale in 3D

$$\begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(4x4)*(4x1) = (4x1)

Y-Rotation in 3D

$$\begin{bmatrix} \cos\phi & 0 & \sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Translation in 3D

$$\begin{bmatrix} 1 & 0 & 0 & Tx \\ 0 & 1 & 0 & Ty \\ 0 & 0 & 1 & Tz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matrix Multiplication

$$\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \\ q \end{bmatrix}$$

Combining Transformations

What if we want to rotate and translate?

Simply multiply the matrices

ORDER MATTERS!

Generally the order you want is:

1. Scale
2. Rotate
3. Translate

University of Colorado
Boulder

THANKS!

Professor **Dan Szafir**

Computer Science & ATLAS Institute
University of Colorado Boulder