CMPS 327: Introduction to Video Game Design and Development

Dr. Arun K. Kulshreshth

Fall 2020

Lecture 8: Math Concepts II

*Adapted from Dunn and Parberry

Reminder

- Midterm Exam 1
  - Wednesday, September 23, 2020
  - ODS: Schedule your exam with ODS by September 18, 2020

About Today's Lecture

- Last Class:
  - Cartesian coordinate system
    - 2D cartesian space
    - 3D cartesian space
  - Vectors and operations on vectors

- Today
  - Multiple coordinate spaces
  - Matrix operations
  - Orientation representations

Why Multiple Coordinate Spaces?

- Some things become easier in the correct coordinate space.
- We can leave the details of transforming between coordinate spaces to the graphics hardware.
- Coordinate Systems in Games/Graphics
  - World space
  - Object space
  - Camera space

World Space

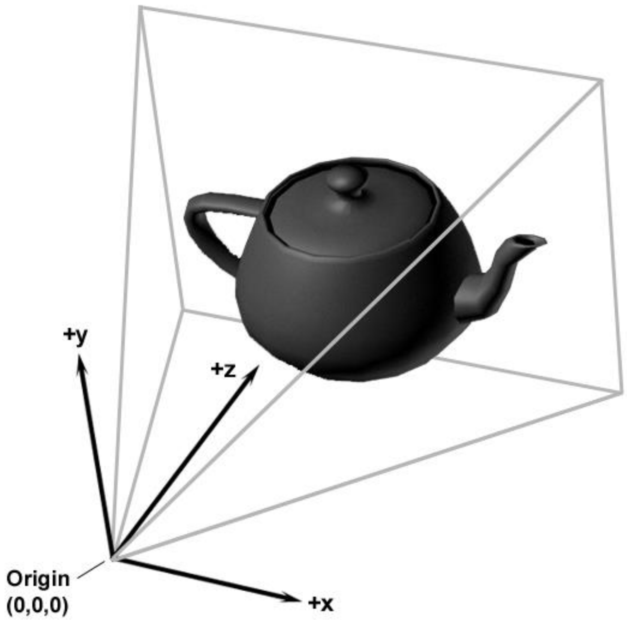
- World space is the global coordinate system.
- Use it to keep track of position and orientation of every object in the game.
- There is only one world space!

Object Space

Every object in the game has:

- Its own origin (where it is),
- Its own concept of “up” and “right” and “forwards”,
- That is, its own coordinate space.
- Use it to keep track of relative positions and orientation (eg. Collision detection, AI)

Camera Space



- Object space for the viewer
- represented by a camera
- used to project 3D space onto screen space

Matrix

Useful in transformations

Definitions

- Algebraic definition of a matrix: a table of scalars in square brackets.
- Matrix *dimension* is the width and height of the table, w x h.
- Typically we use dimensions 2 x 2 for 2D work, and 3 x 3 for 3D work.

Matrix Components

- Entries are numbered by row and column, eg. m_{ij} is the entry in row *i*, column *j*.

- Start numbering at 1, not 0.

$$\begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}$$

Square Matrices

- Same number as rows as columns.
- Entries m_{ii} are called the *diagonal* entries. The others are called *nondiagonal* entries

$$\begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}$$

Diagonal Matrices

A diagonal matrix is a square matrix whose nondiagonal elements are zero.

$$\begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -5 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}$$

The Identity Matrix

The identity matrix of dimension *n*, denoted $\mathbf{I}_n$, is the *n* x *n* matrix with 1s on the diagonal and 0s elsewhere.

$$\mathbf{I}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Vectors as Matrices

- A row vector is a 1 x *n* matrix.
- A column vector is an *n* x 1 matrix.
- They were pretty much interchangeable in the lecture on Vectors.
- They're not once you start treating them as matrices.

$$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \qquad \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix}$$

Transpose of a Matrix

- The transpose of an *r* x *c* matrix **M** is a *c* x *r* matrix called $\mathbf{M}^T$.
- Take every row and rewrite it as a column.
- Equivalently, flip about the diagonal

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}^T = \begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix}$$

Transpose of a Vector

If **v** is a row vector, $\mathbf{v}^T$ is a column vector and vice-versa

$$\begin{bmatrix} x & y & z \end{bmatrix}^T = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad \begin{bmatrix} x \\ y \\ z \end{bmatrix}^T = \begin{bmatrix} x & y & z \end{bmatrix}$$

Multiplying By a Scalar

- Can multiply a matrix by a scalar.
- Result is a matrix of the same dimension.
- To multiply a matrix by a scalar, multiply each component by the scalar.

$$k\mathbf{M} = k\begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \\ m_{41} & m_{42} & m_{43} \end{bmatrix} = \begin{bmatrix} km_{11} & km_{12} & km_{13} \\ km_{21} & km_{22} & km_{23} \\ km_{31} & km_{32} & km_{33} \\ km_{41} & km_{42} & km_{43} \end{bmatrix}$$

Matrix Multiplication

Multiplying an *r* x *n* matrix **A** by an *n* x *c* matrix **B** gives an *r* x *c* result **AB**.

$$\underset{\substack{r \times n \\ 4 \times 2}}{\overset{\mathbf{A}}{\begin{bmatrix} ? & ? \\ ? & ? \\ ? & ? \\ ? & ? \end{bmatrix}}} \quad \underset{\substack{n \times c \\ 2 \times 5}}{\overset{\mathbf{B}}{\begin{bmatrix} ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? \end{bmatrix}}} = \underset{\substack{r \times c \\ 4 \times 5}}{\overset{\mathbf{AB}}{\begin{bmatrix} ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? \end{bmatrix}}}$$

Multiplication: Result

- Multiply an r x n matrix **A** by an n x c matrix **B** to give an r x c result **C** = **AB**.
- Then **C** = $[c_{ij}]$, where c_{ij} is the dot product of the ith row of **A** with the jth column of **B**.
- That is:

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} .$$

Example

$$\begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} & c_{15} \\ c_{21} & c_{22} & c_{23} & c_{24} & c_{25} \\ c_{31} & c_{32} & c_{33} & c_{34} & c_{35} \\ c_{41} & c_{42} & c_{43} & c_{44} & c_{45} \end{bmatrix} =$$

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \\ a_{41} & a_{42} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} & b_{15} \\ b_{21} & b_{22} & b_{23} & b_{24} & b_{25} \end{bmatrix}$$

$$c_{24} = a_{21}b_{14} + a_{22}b_{24}$$

2 x 2 Case

$$
\begin{aligned}
\mathbf{AB} &= \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \\
&= \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{bmatrix}
\end{aligned}
$$

3 x 3 Case

$$\mathbf{AB} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}$$
$$= \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} & a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} & a_{11}b_{13} + a_{12}b_{23} + a_{13}b_{33} \\ a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} & a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} & a_{21}b_{13} + a_{22}b_{23} + a_{23}b_{33} \\ a_{31}b_{11} + a_{32}b_{21} + a_{33}b_{31} & a_{31}b_{12} + a_{32}b_{22} + a_{33}b_{32} & a_{31}b_{13} + a_{32}b_{23} + a_{33}b_{33} \end{bmatrix}$$

Row Vector Times Matrix Multiplication

Can multiply a row vector times a matrix

$$\begin{bmatrix} x & y & z \end{bmatrix} \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} =$$

$$\begin{bmatrix} xm_{11} + ym_{21} + zm_{31} & xm_{12} + ym_{22} + zm_{32} & xm_{13} + ym_{23} + zm_{33} \end{bmatrix}$$

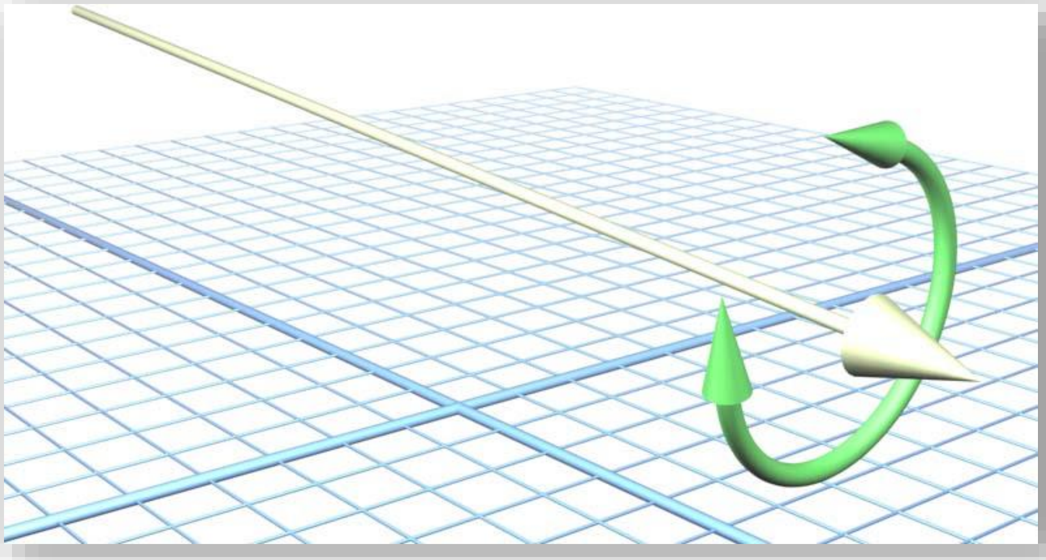
Matrix Times Column Vector Multiplication

Can multiply a matrix times a column vector.

$$\begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} xm_{11} + ym_{12} + zm_{13} \\ xm_{21} + ym_{22} + zm_{23} \\ xm_{31} + ym_{32} + zm_{33} \end{bmatrix}$$

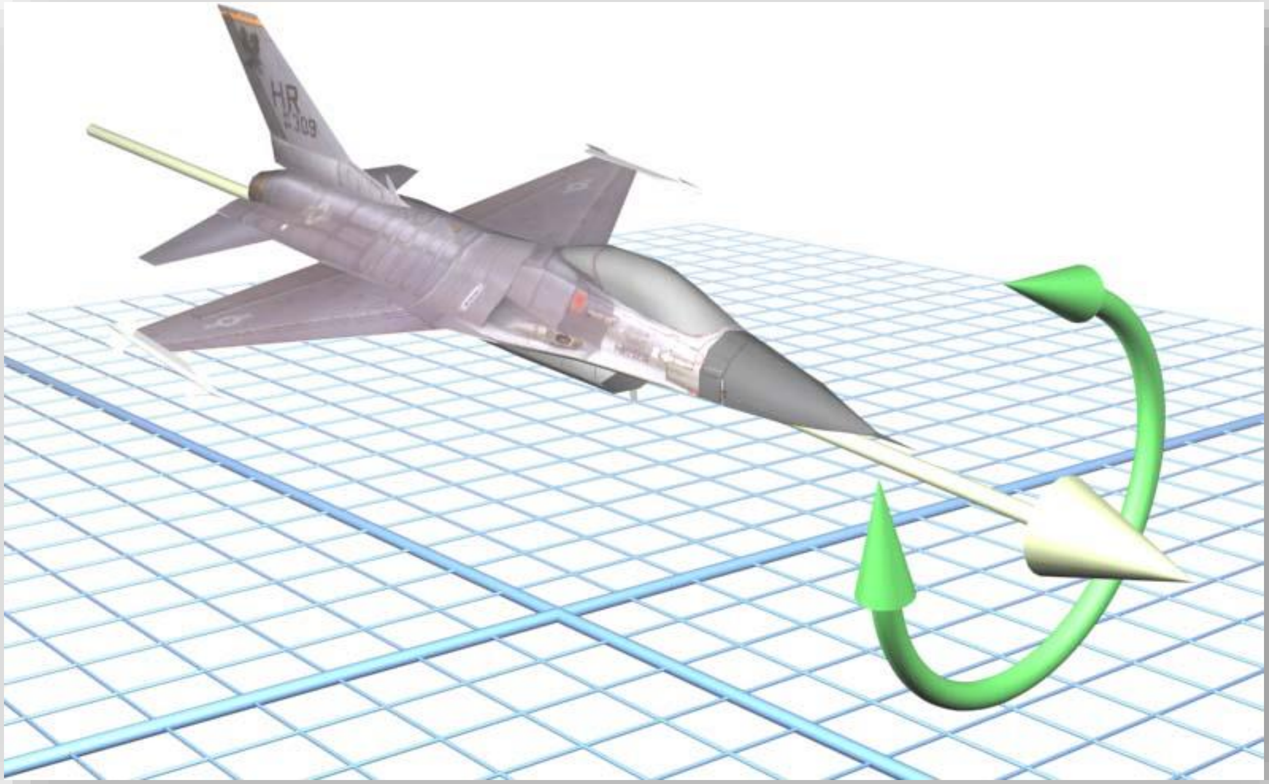
Orientation

- What is orientation?
- More than direction.
- A vector specifies direction, but it can also be twisted.



This is Important Because

Twisting an object changes its orientation.



Angular Displacement

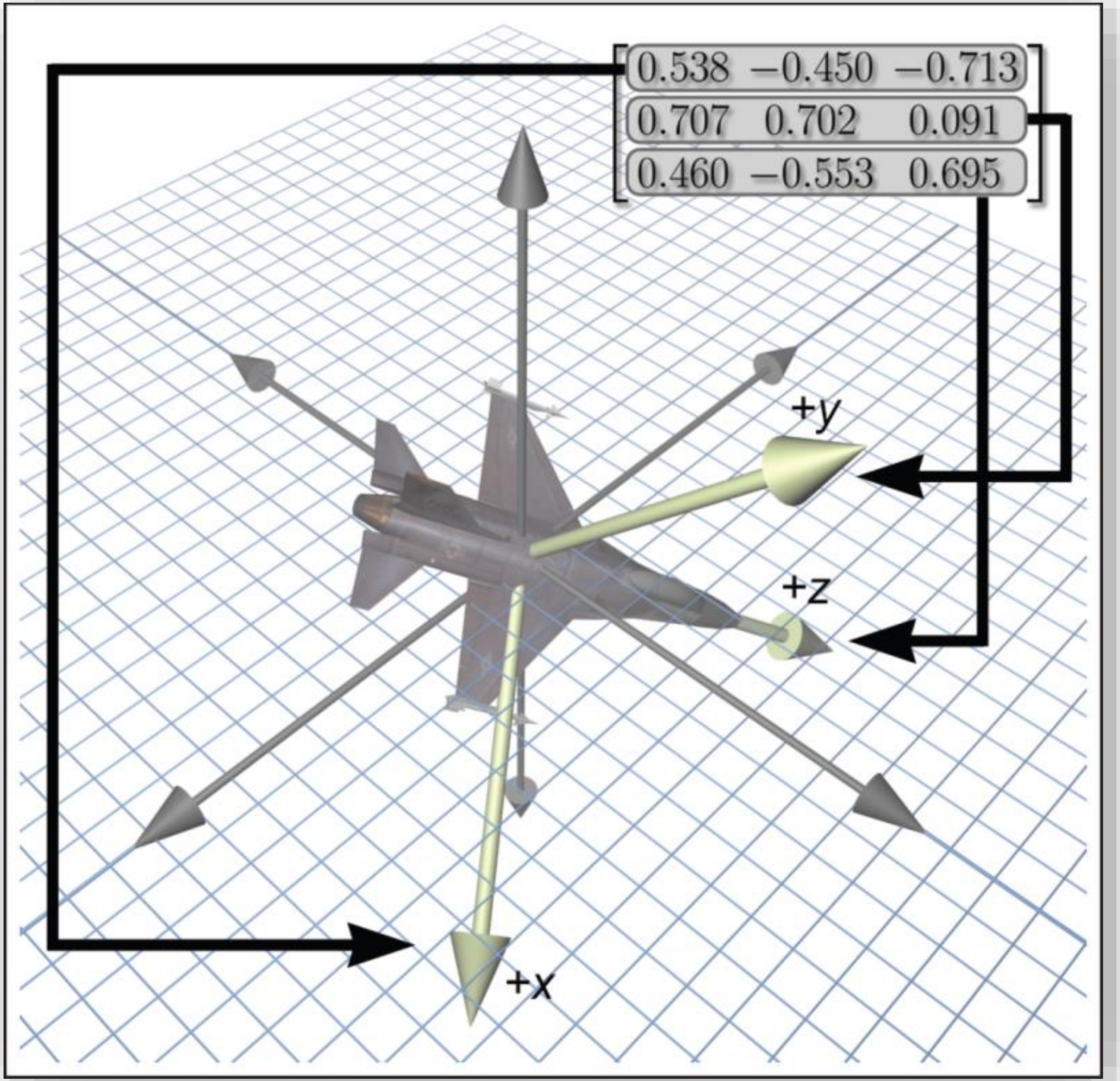
- Orientation can’t be given in absolute terms.
- Just as a position is a translation from some known point, an orientation is a rotation from some known reference orientation (often called the *identity* or *home* orientation).
- The amount of rotation is known as an *angular displacement*.

How to Represent Orientation

1. Matrices
2. Euler angles
3. Quaternions

Matrix Form

- List the relative orientation of two coordinate spaces by listing the transformation matrix that takes one space to another.
- For example: from object space to world space.
- Transform back by using the inverse matrix.



0.538 −0.450 −0.713
0.707 0.702 0.091
0.460 −0.553 0.695
+y
+z
+x

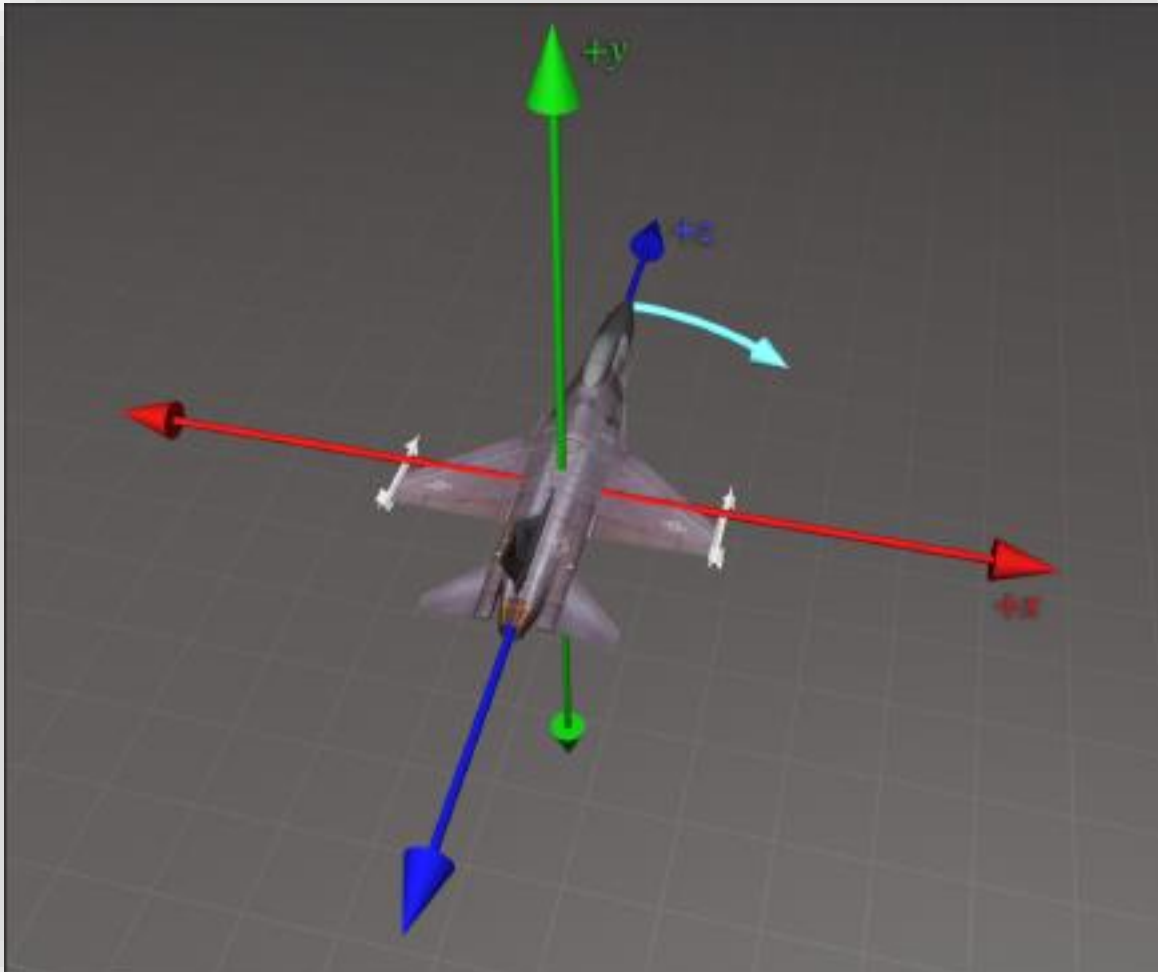
Euler Angles

- Euler angles are another common method of representing orientation.
- Euler is pronounced “oiler,” not “yoolur.”
- They are named for the famous mathematician who developed them, Leonhard Euler (1707 – 1783).

Euler Angles

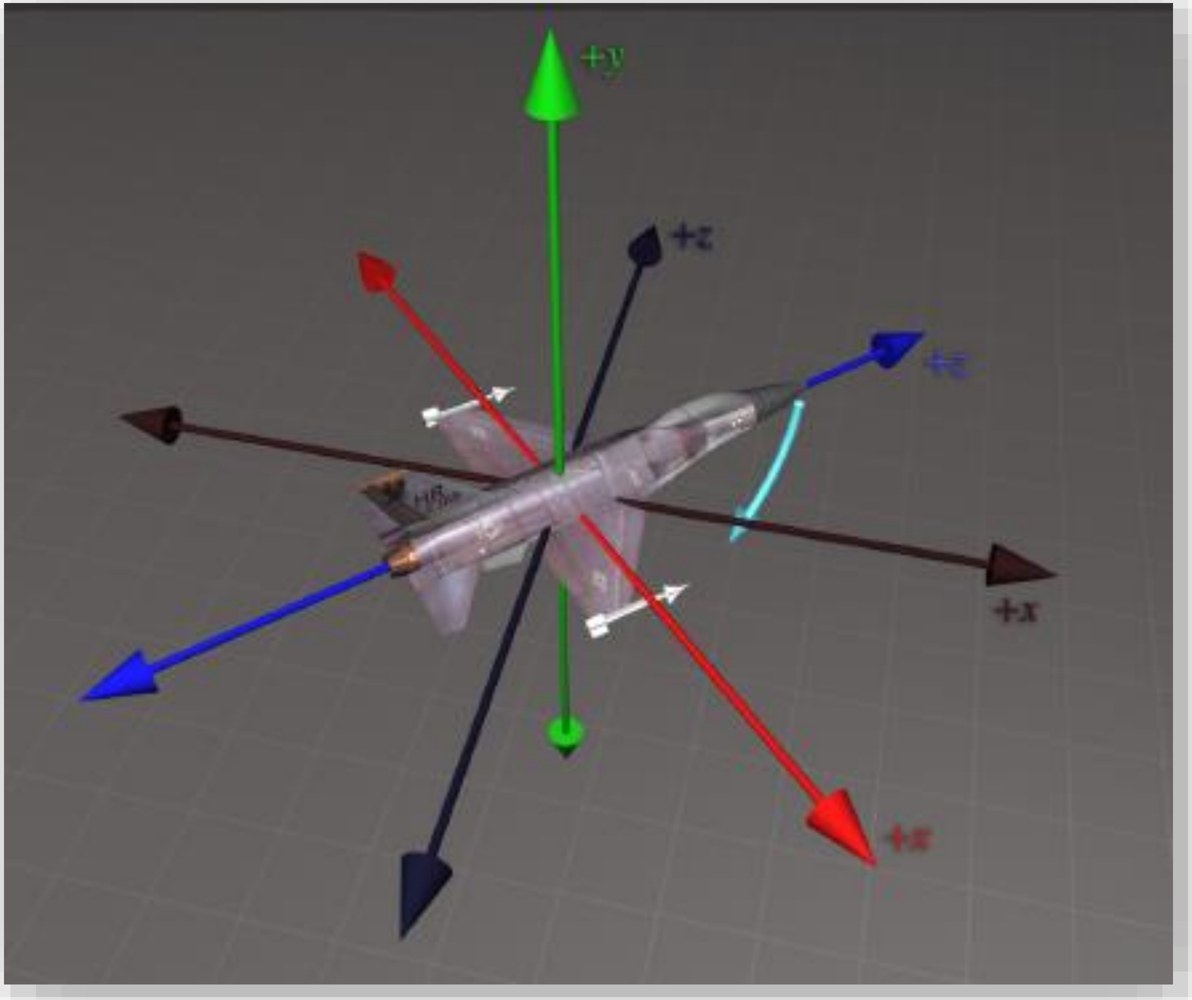
- Specify orientation as a series of 3 angular displacements from upright space to object space.
- Which axes? Which order?
- Need a convention.
- Heading-pitch-bank
  - Heading: rotation about *y* axis (aka "yaw")
  - Pitch: rotation about *x* axis
  - Bank: rotation about *z* axis (aka "roll")

Heading



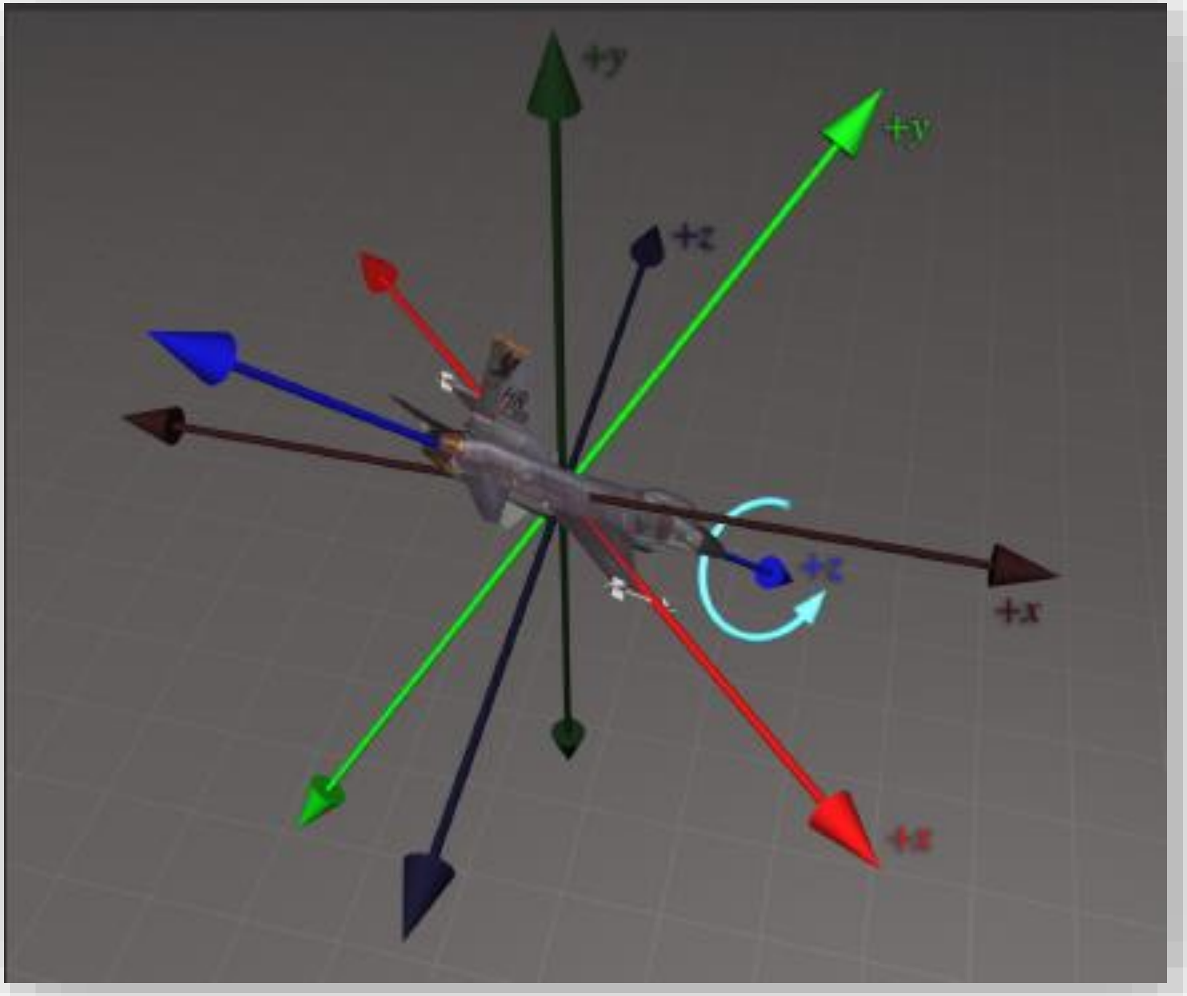
Rotation about y-axis

Pitch



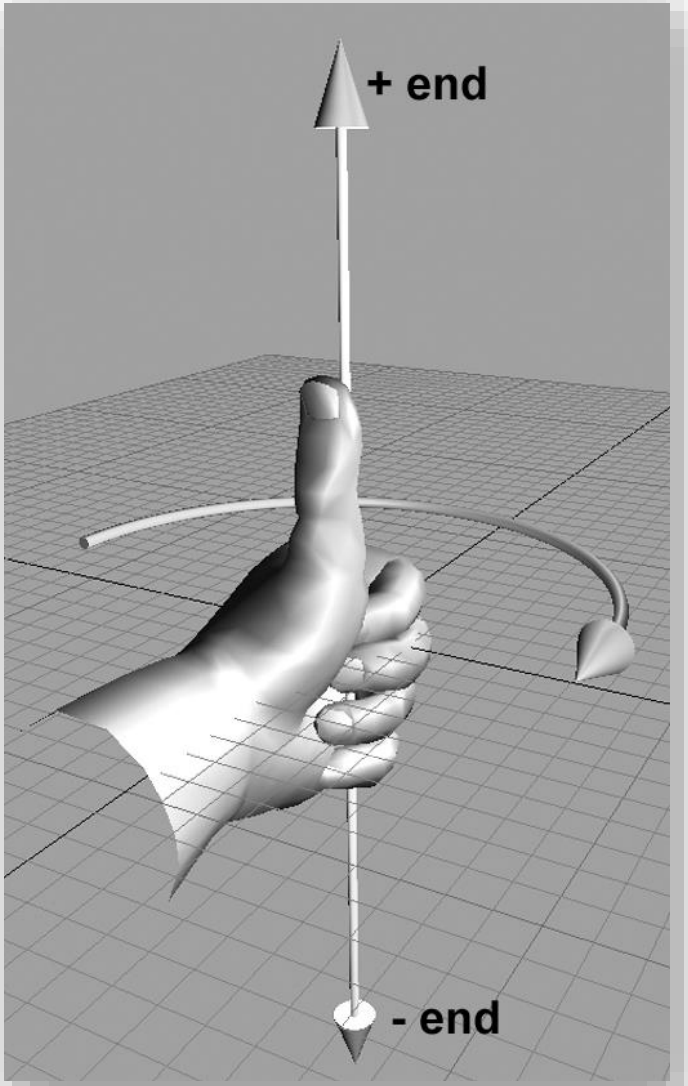
Rotation about x-axis

Bank



Rotation about z-axis

The Sign Matters



- Use the hand rule again.
- Thumb points along positive axis of rotation.
- Fingers curl in direction of positive rotation.

The Order Matters

- Heading is first: it is relative to the upright frame of reference – that is, vertical.
- Pitch is next because it is relative to the horizon. But the *x*-axis may have been moved by the heading change. (Object *x* is no longer the same as upright *x*.)
- Bank is last. The *z*-axis may have been moved by the heading and pitch change. (Object *z* is no longer the same as upright *z*.)

Advantages of Euler Angles

- Easy for humans to use.
- Really the only option if you want to enter an orientation by hand.

- Minimal space – 3 numbers per orientation.
  - Bottom line: if you need to store a lot of 3D rotational data in as little memory as possible, as is very common when handling animation data, Euler angles are the best choices.

Advantages of Euler Angles

- Another reason to choose Euler angles when you need to save space is that the numbers you are storing are more easily compressed.
  - Each number is <= 360

- Every set of 3 numbers makes sense – unlike matrices and quaternions.

Summary

- Multiple coordinate spaces
  - World space
  - Object space
  - Camera space
- Matrix and operations
- Orientation Representations