Animation

Orientation: Euler Angles

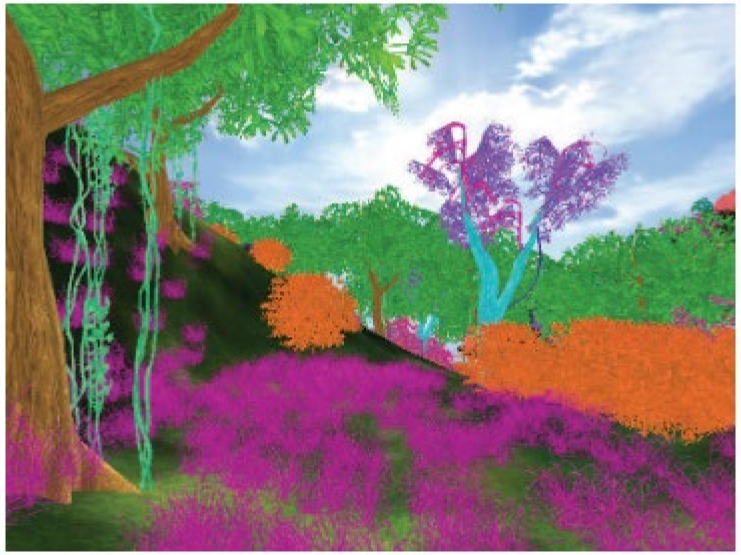
CS 415: Game Development

Professor Eric Shaffer

Instancing

The other approach to minimize application processing and API costs is to use some form of ***instancing.*** Most APIs support the idea of having an object and drawing it several times in a single call. This is typically done by specifying a base model and providing a separate data structure that holds information about each specific instance desired. Beyond position and ***orientation***, other attributes could be specified per instance, such as leaf colors or curvature due to the wind, or anything else that could be used by shader programs to affect the model.

Real-Time Rendering, Fourth Edition

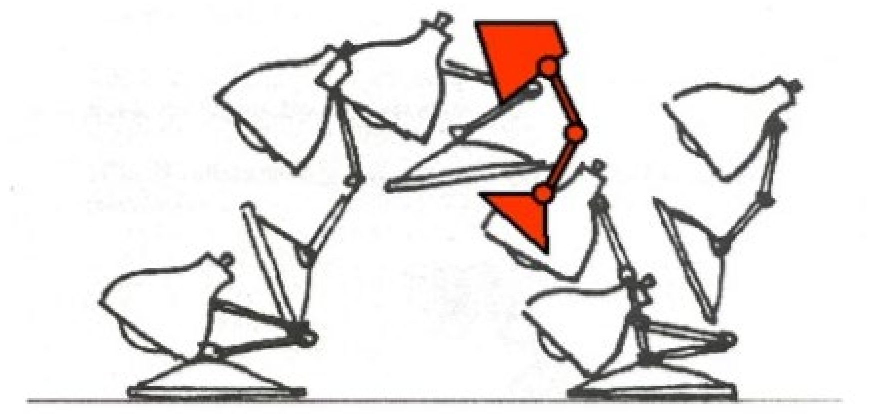


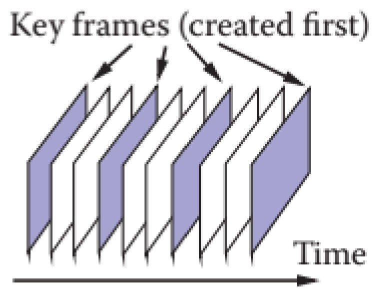
Vegetation instancing.

All objects the same color in the lower image are rendered in a single draw call.

You render the same number of triangles as non-instancing…
So why is instancing more performant than using more meshes?

Animation and Orientation





- Interpolation is used to generate in-between frames from keyframes
- At a minimum this means interpolating position and ***orientation*** of a model
- So…how can we represent orientation?

Orientation

- A model has an initial geometry (from the artist)
- Changing that orientation requires rotation
- How about just using a single rotation matrix ?

Consider a 2D example

- Starting rotation is 90 degree clockwise
- Ending rotation is 90 degree counter-clockwise
- Doing element-by-element linear interpolation to get intermediate rotation

$$\frac{1}{2}\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} + \frac{1}{2}\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

- Does this make sense? Should we have expected something different?

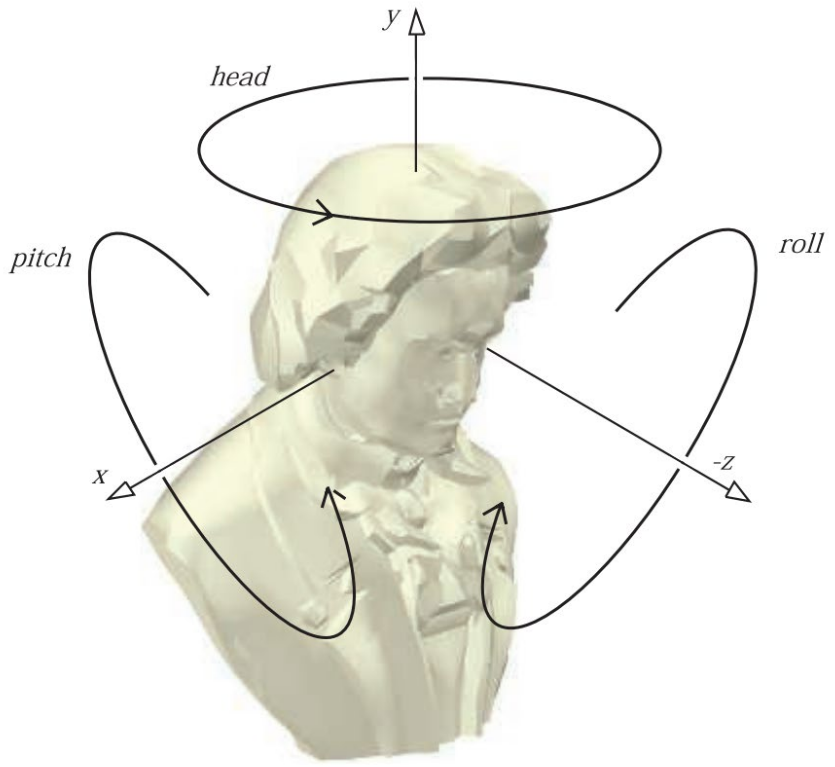
Orientation: Euler Angles

We can record an orientation as

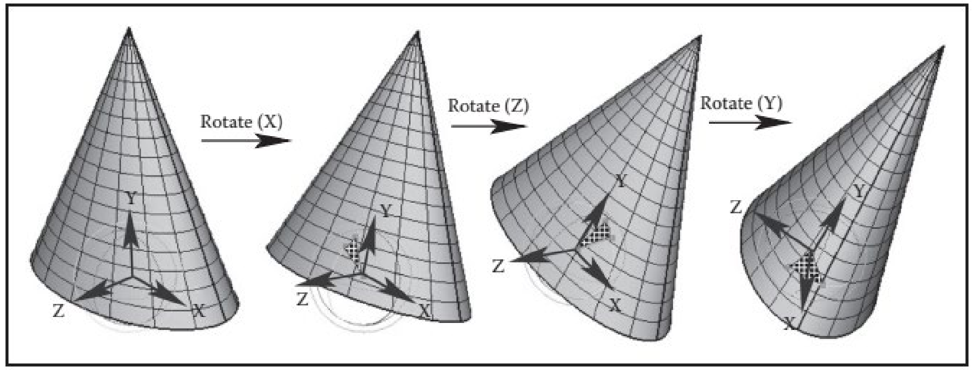
- Three rotations
- Each rotation is around a principal axis
- Example: $\mathbf{E}(h,p,r) = \mathbf{R}_z(r)\mathbf{R}_x(p)\mathbf{R}_y(h)$
- This not only option
  - 24 different possible orders one could use
- Also, people use different terminology
  - e.g. “yaw” instead of “head”
  - Also which axis corresponds to a name is not standardized

“Most manufacturing processes, including 3D printing, consider the z-direction to be up in world space; aviation and sea vehicles consider −z to be up. Architecture and GIS normally use z-up, as a building plan or map is two-dimensional, x and y. Media-related modeling systems often consider the y-direction as up in world coordinates, matching how we always describe a camera’s screen up direction in computer graphics.”

Real-Time Rendering, Fourth Edition



Euler Angles: The Good Part



- Three Euler angles can be used to specify arbitrary object orientation
  - i.e. any orientation we want can be achieved with these 3 rotations

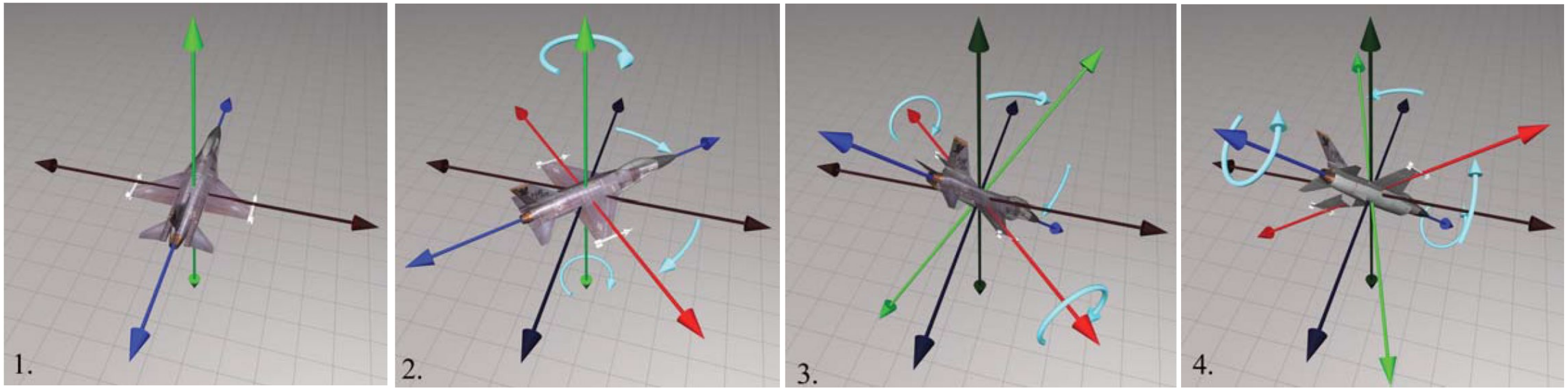
- Euler angles are used in a lot of applications…they are believed to be intuitive

- They are compact…requiring only 3 numbers

- Easily converted into a single rotation matrix

Euler Angles: The Problematic Parts

Are they really intuitive? Hmmm….



“Three Euler angles can be used to specify arbitrary object orientation through a sequence of three rotations around coordinate axes embedded into the object (axis Y always points to the tip of the cone). Note that each rotation is given in a new coordinate system.”

- Fundamentals of Computer Graphics

Plus, different triples can be used to generate the same orientation

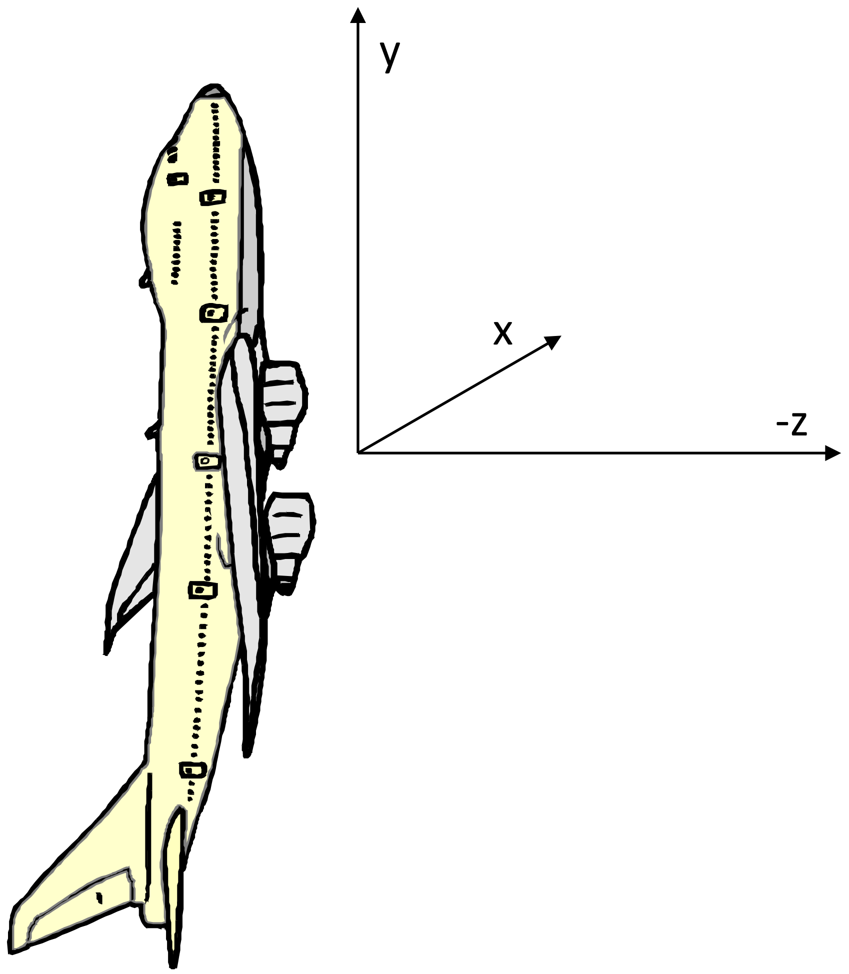
Gimbal Lock

- Airplane orientation
  - Rz(roll) Rx(pitch) Ry(yaw)
- Two axes have collapsed onto each other
- Think about this from a user-interface perspective

Imagine you have 3 dials…one for each angle

What action caused the orientation you see?
What happens when z dial is moved now?

What problem could this cause for someone playing a game with this interface?



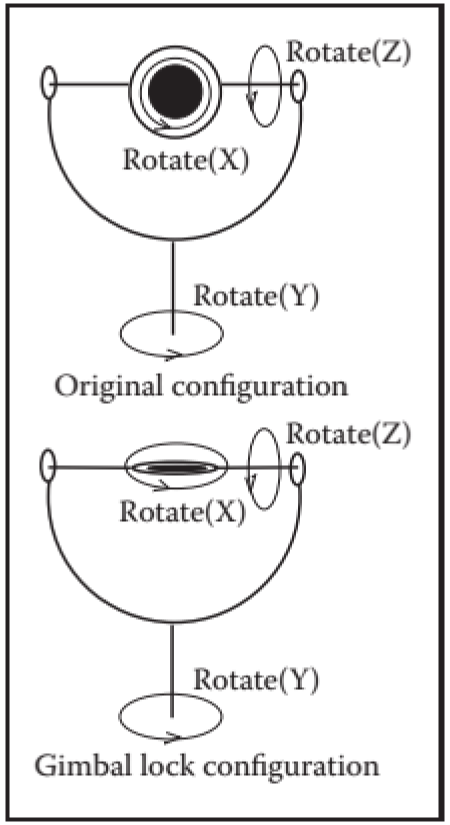
ILLINOIS

Gimbal Lock

When the middle rotation is a 90 or -90 rotation, we generate gimbal lock

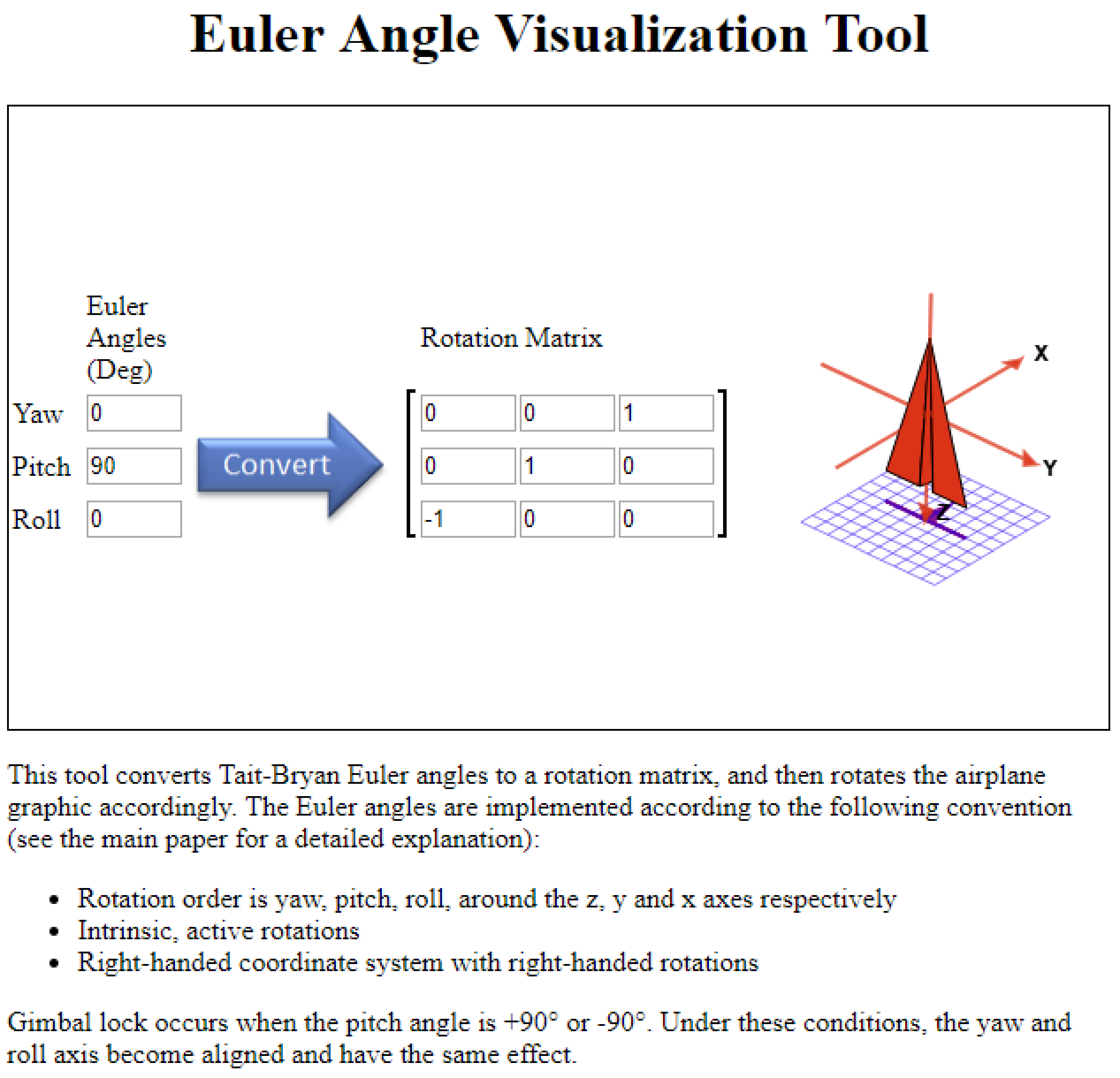
In this example, gimbal lock occurs when a 90 degree turn around axis Z is made

Both X and Y rotations are now performed around the same axis



Experiment

http://danceswithcode.net/engineeringnotes/rotations_in_3d/demo3D/rotations_in_3d_tool.html



Euler Angle Visualization Tool

Euler Angles (Deg)

Yaw	0
Pitch	90
Roll	0

Rotation Matrix

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix}$$

This tool converts Tait-Bryan Euler angles to a rotation matrix, and then rotates the airplane graphic accordingly. The Euler angles are implemented according to the following convention (see the main paper for a detailed explanation):

- Rotation order is yaw, pitch, roll, around the z, y and x axes respectively
- Intrinsic, active rotations
- Right-handed coordinate system with right-handed rotations

Gimbal lock occurs when the pitch angle is +90° or -90°. Under these conditions, the yaw and roll axis become aligned and have the same effect.

Is Gimbal Lock that Important?

"In the video game industry there has been some back-and-forth battles about whether this problem is crucial. In an FPS game, the avatar is usually not allowed to pitch his head all the way to $\pm\pi/2$, thereby avoiding this problem. In VR, it happens all the time that a user could pitch her head straight up or down. The kinematic singularity often causes the viewpoint to spin uncontrollably…"

-- *Virtual Reality* by Lavalle Section 3.3

ILLINOIS

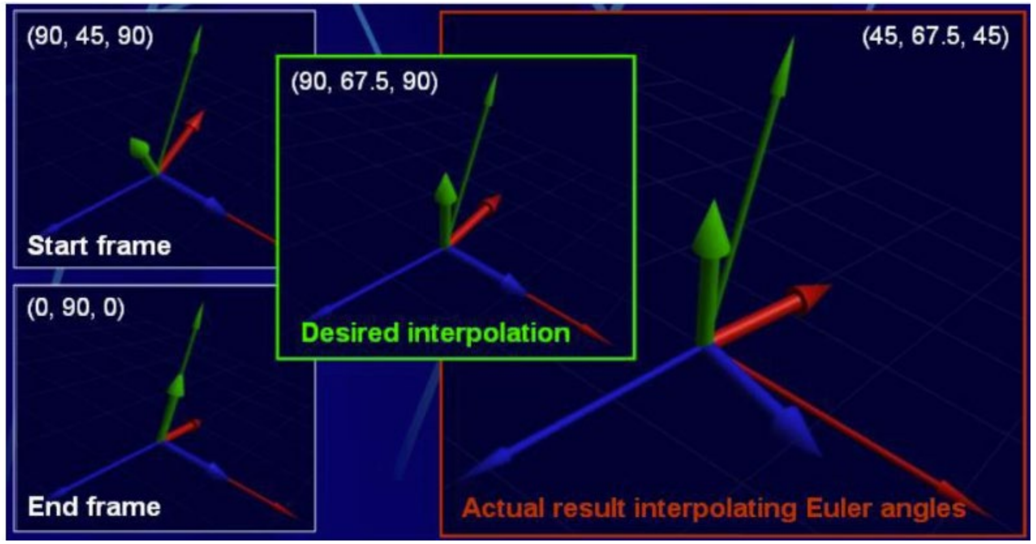
Euler Angles: Interpolation

Could linearly interpolate between each angle in a start pose and end pose to generate intermediate poses

This doesn’t work well

It treats rotations as if they were independent of each other and completely ignores the effect they have on each other

Can result in jerky, unnatural movement


(90, 45, 90)
Start frame
(0, 90, 0)
End frame
(90, 67.5, 90)
Desired interpolation
(45, 67.5, 45)
Actual result interpolating Euler angles