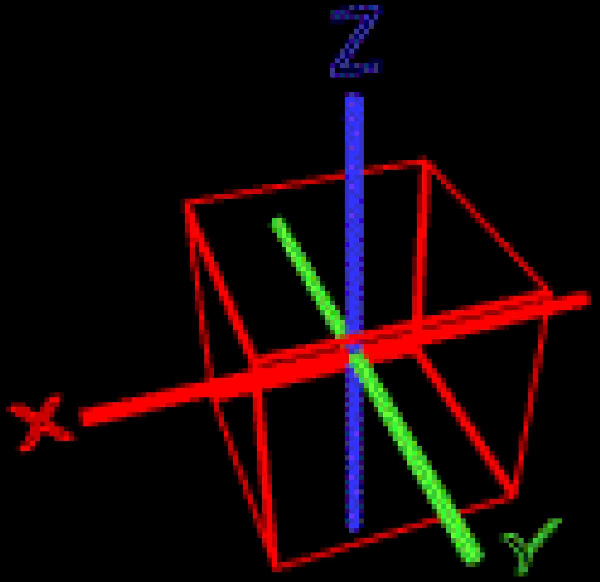
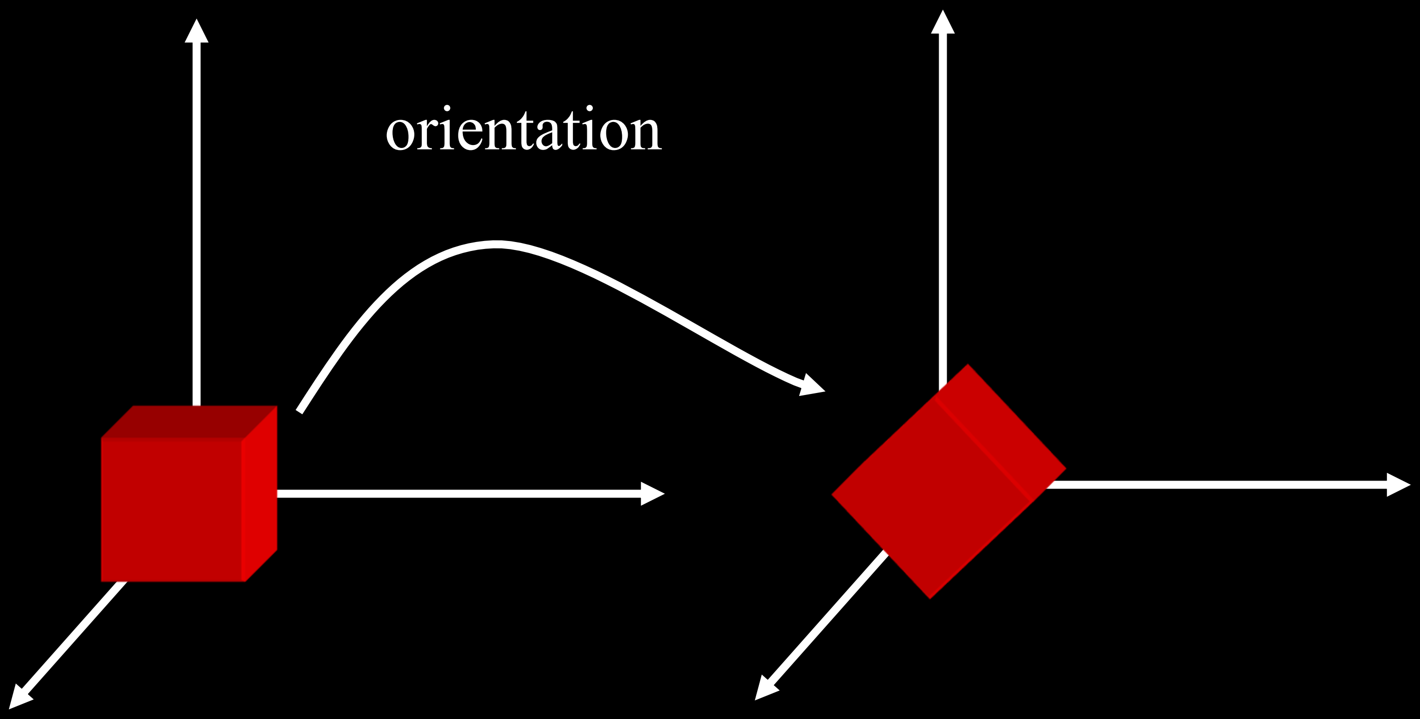
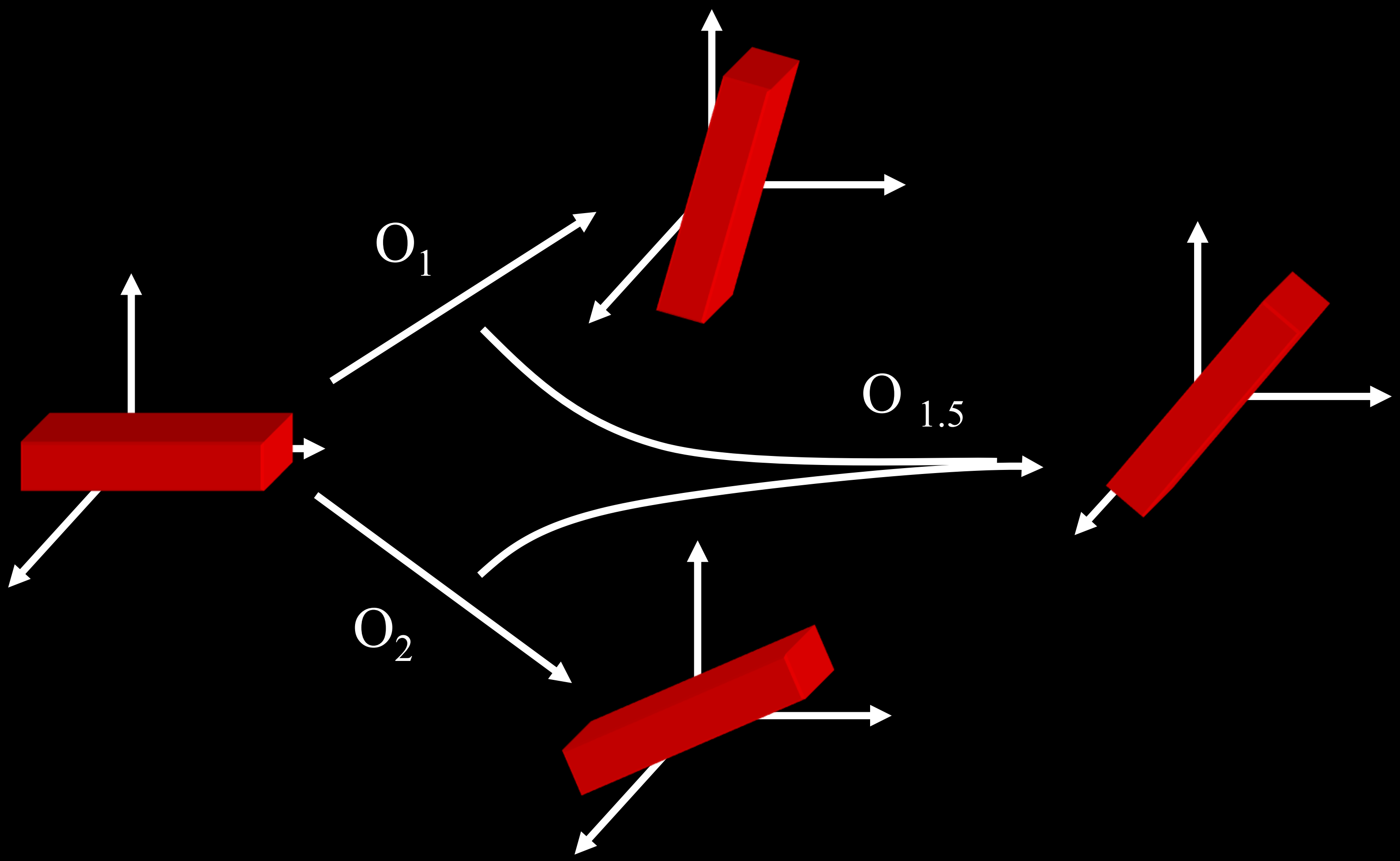
# Representing Rotations



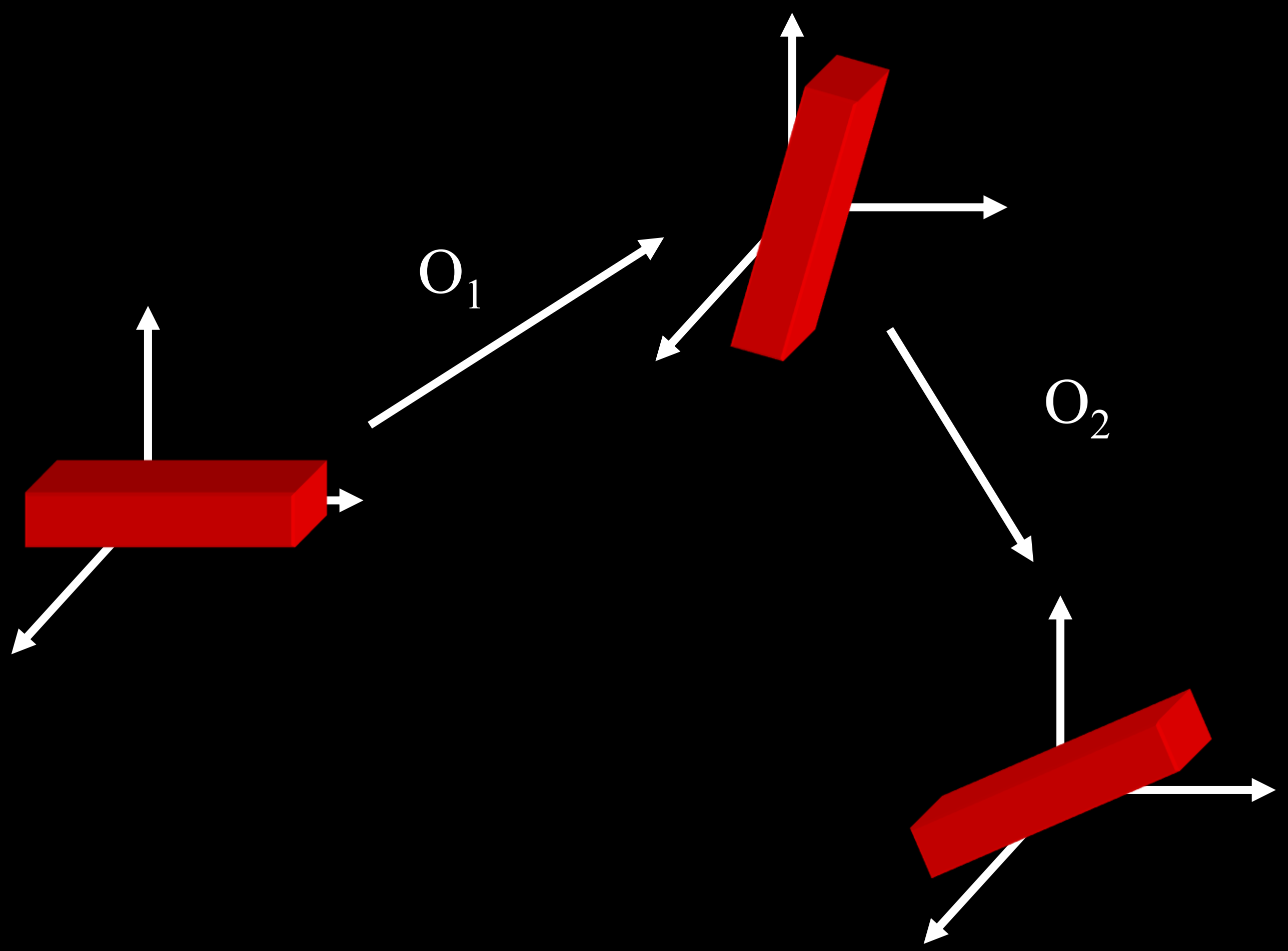
# Orientation Representation



# Interpolation



# Concatenation



# 3-D Transformations

- Translate, scale, or rotate a point P to P’
  - $P' = P + T$
  - $P' = SP$
  - $P' = RP$
- How to treat these transformations in a unified way?
  - $P' = MP$
- Representing P in the homogeneous coordinate
- M can be used for animation, viewing, or modeling

# Homogeneous Coordinate

- In graphics, we use homogeneous coordinate for transformation
- 4x4 matrix can represent translation, scaling, and rotation and other transformations

$$(\frac{x}{w}, \frac{y}{w}, \frac{z}{w}) = [x, y, z, w]$$

$$(x, y, z) = [x, y, z, 1]$$

- Typically, when transforming a point in 3D space, we set w = 1

# Translation

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

New point in 3D space

Point in 3D space

Transformation matrix

# Scaling

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

# Rotation

- X axis

$$R_x(\theta)\quad \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \boxed{\begin{matrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{matrix}} & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ \begin{matrix} 0 & 0 & 0 \end{matrix} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- Y axis

$$R_y(\theta)\quad \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- Z axis

$$R_z(\theta)\quad \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

# Compounding Transformations

- Transformations can be treated as a series of matrix multiplications

$$P' = M_1 M_2 M_3 \cdots M_n P \qquad P'^T = (M_1 M_2 M_3 \cdots M_n P)^T$$

$$M = M_1 M_2 M_3 \cdots M_n \qquad M^T = M_n^T M_{n-1}^T \cdots M_2^T M_1^T$$

$$P' = MP \qquad P'^T = P^T M^T$$

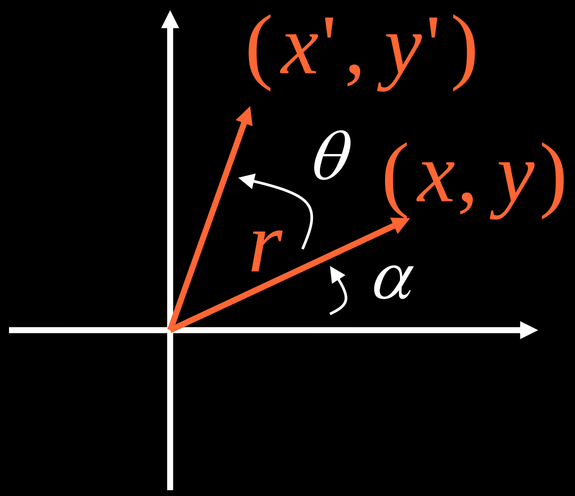
rotation, scaling

$$\begin{bmatrix} s_x \cos\theta & -\sin\theta & 0 & t_x \\ \sin\theta & s_y \cos\theta & 0 & t_y \\ 0 & 0 & s_z & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

translation

# Two Ways of Interpreting a Rotation Matrix

- Rotating a vector

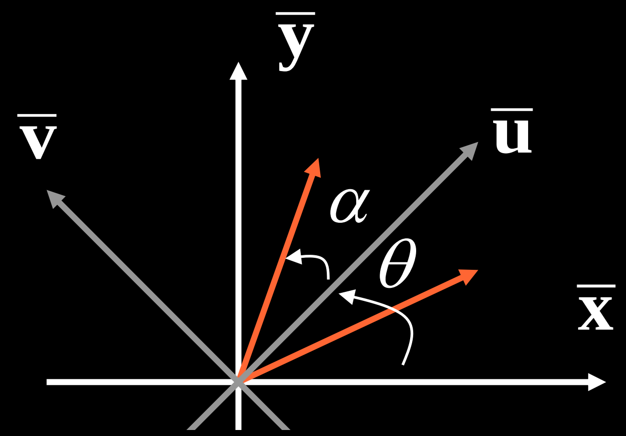



$$x' = r\cos(\theta+\alpha)$$
$$= r(\cos\theta\cos\alpha - \sin\theta\sin\alpha)$$
$$= (\cos\theta r\cos\alpha - \sin\theta r\sin\alpha)$$
$$= (\cos\theta x - \sin\theta y)$$
$$y' = r\sin(\theta+\alpha)$$
$$= (\sin\theta r\cos\alpha + \cos\theta r\sin\alpha)$$
$$= (\sin\theta x + \cos\theta y)$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

# Two Ways of Interpreting a Rotation Matrix

- Rotating a coordinate




$$\mathbf{V} = x^0\overline{\mathbf{x}} + y^0\overline{\mathbf{y}} = x^1\overline{\mathbf{u}} + y^1\overline{\mathbf{v}}$$

$$= x^1(\cos\theta\overline{\mathbf{x}} + \sin\theta\overline{\mathbf{y}}) + y^1(-\sin\theta\overline{\mathbf{x}} + \cos\theta\overline{\mathbf{y}})$$

$$\begin{bmatrix} x^0 \\ y^0 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x^1 \\ y^1 \end{bmatrix} = \begin{bmatrix} \overline{\mathbf{u}} & \overline{\mathbf{v}} \end{bmatrix} \begin{bmatrix} x^1 \\ y^1 \end{bmatrix}$$

Transformation that maps from coordinate 1 to coordinate 0

Axes of coordinate 1 represented in coordinate 0

# Rotation Matrix

- Rows/columns of matrix must be orthonormal
  - Unit length and orthogonal
- Numerical errors cause a nonorthonomral matrix when a series of rotations apply
- How to interpolate between matrices?
  - Interpolating the components of two matrices doesn’t maintain the orthonormality
  - The generated matrix is not a rotation matrix

# Interpolate Rotation Matrix?

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

$90^o$ z-axis $\qquad\qquad$ $-90^o$ z-axis

- The halfway matrix you get by linearly interpolating each entry is

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

- Not a rotation matrix any more!

# Representing 3D rotations

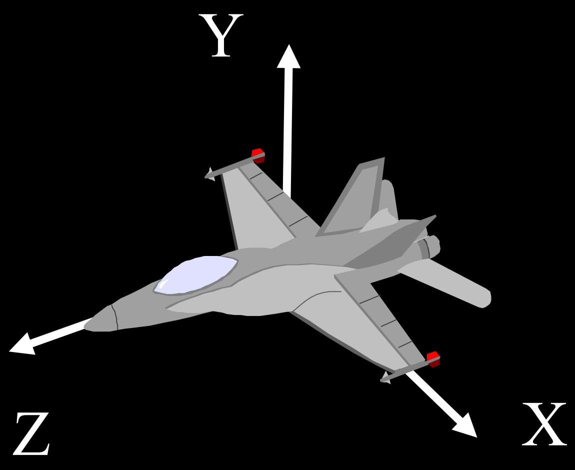
- Rotation Matrix
- Fixed Angle
- Euler Angle
- Axis angle
- Quaternion

# Fixed Angle Representation

- Ordered triple of rotations about global axes
- Any triple can be used that doesn't repeat an axis immediately, e.g., x-y-z is fine, so is x-y-x. But x-x-z is not.

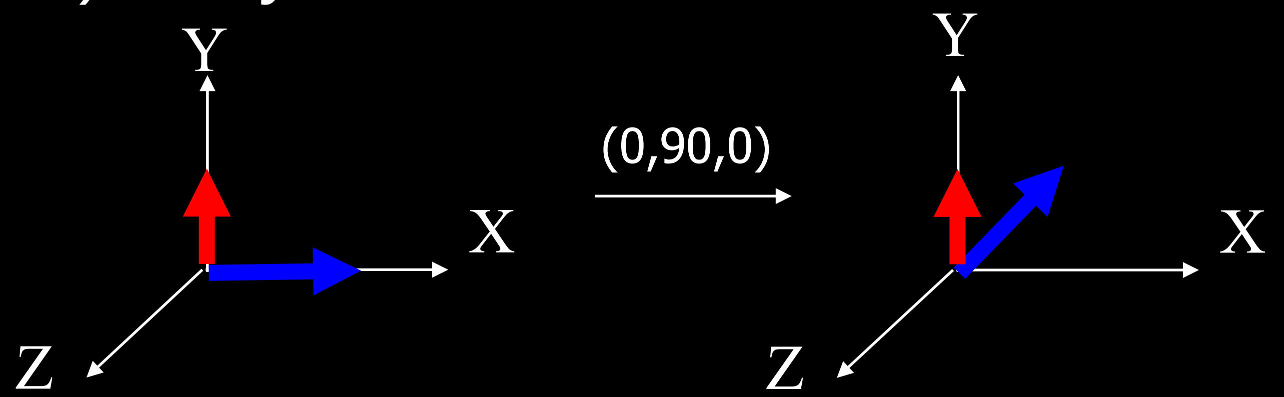
e.g., x-y-z order $(\theta_x, \theta_y, \theta_z)$

$$P' = R_z(\theta_z)R_y(\theta_y)R_x(\theta_x)P$$

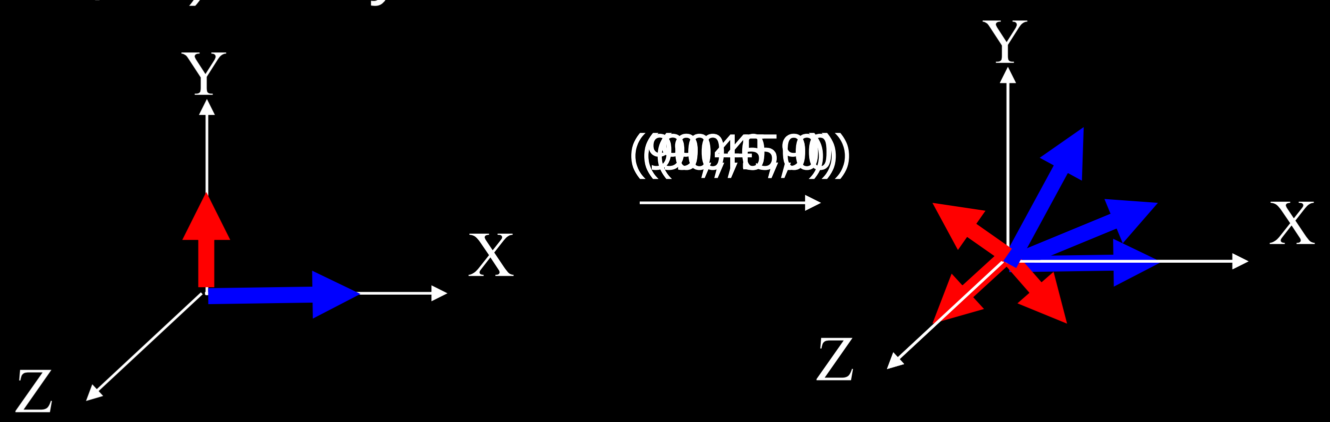


# Fixed Angle Representation

- (0,90,0) in x-y-z order

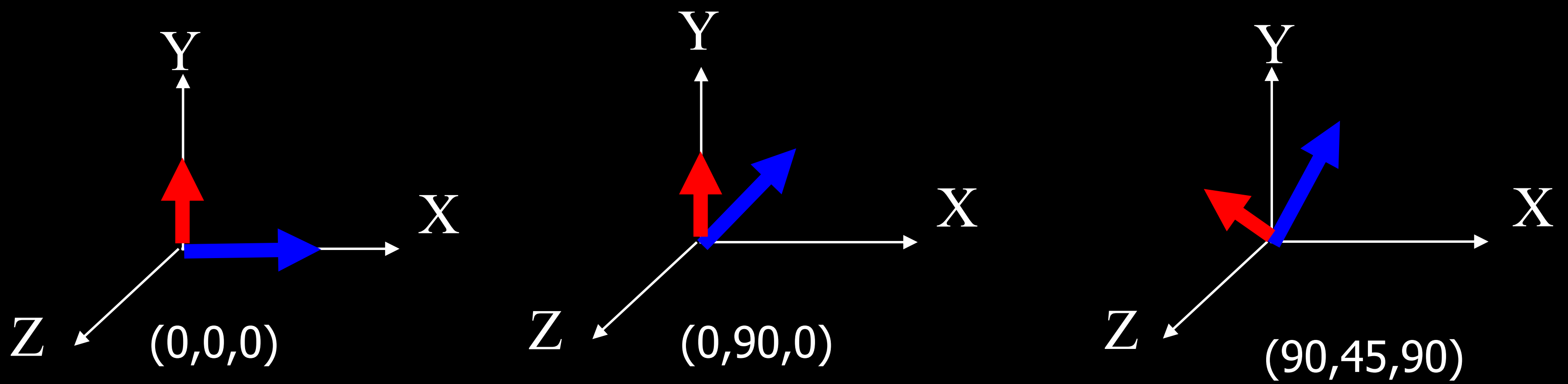



- (90,45,90) in x-y-z order



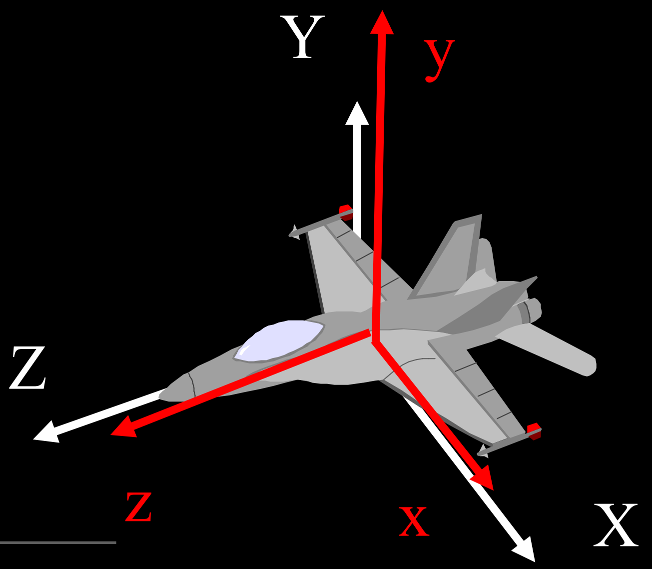
# Interpolation Problem in Fixed Angle

- The rotation from (0,90,0) to (90,45,90) is a 45-degree x-axis rotation
- Directly interpolating between (0,90,0) and (90,45,90) produces a halfway orientation (45, 67.5, 45)
- Desired halfway orientation is (90, 22.5, 90)



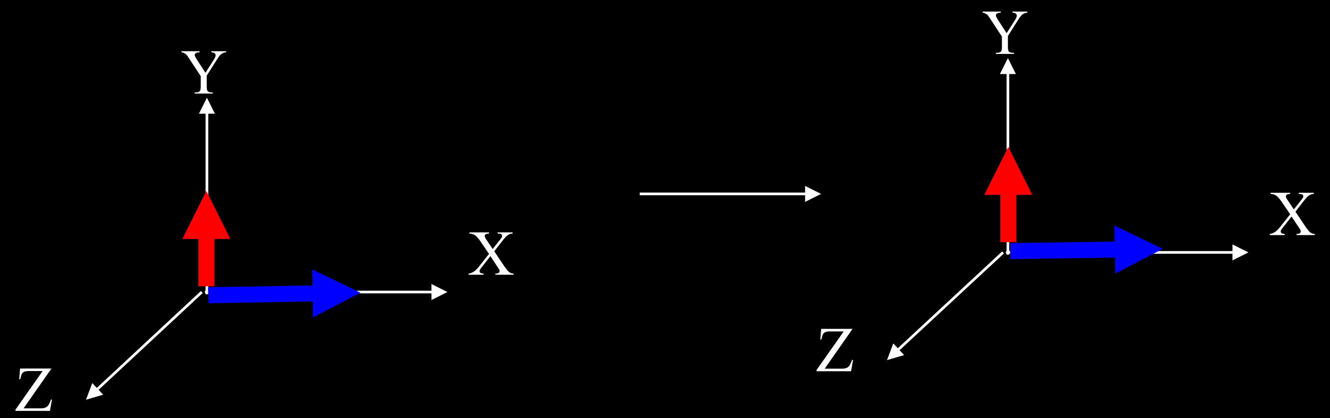
# Euler Angle

- Ordered triple of rotations about local axes
- As with fixed angles, any triple can be used that doesn't immediately repeat an axis, e.g., x-y-z, is fine, so is x-y-x. But x-x-z is not.
- Euler angle ordering is equivalent to reverse ordering in fixed angles

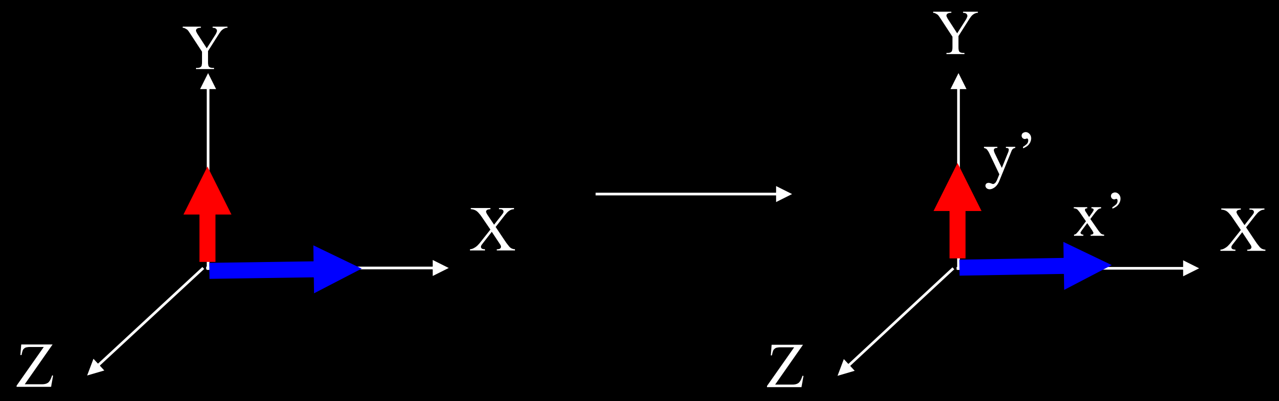


# Fixed Angle vs. Euler Angle

- Fixed angle: (90,45,90) in x-y-z order

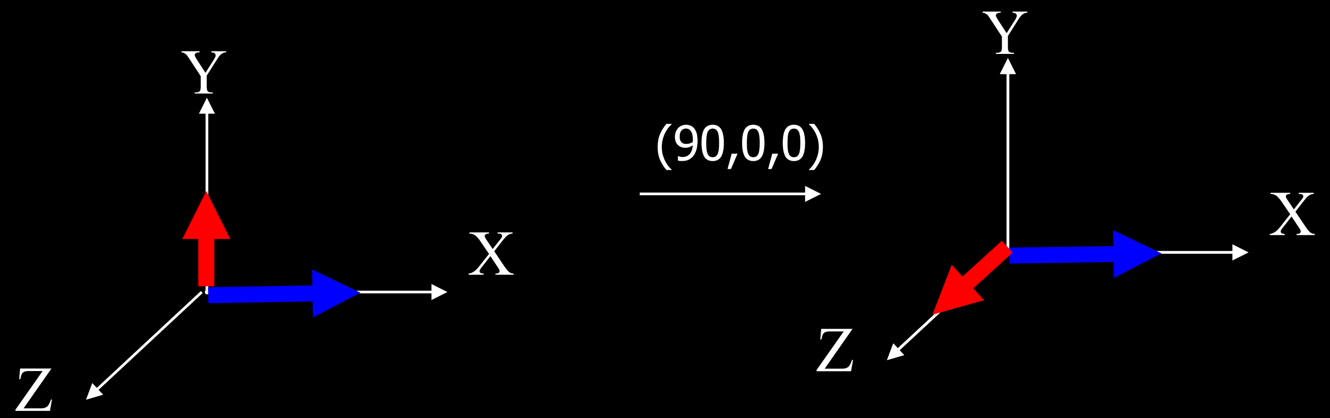



- Euler angle: (90,45,90) in z'-y'-x' order

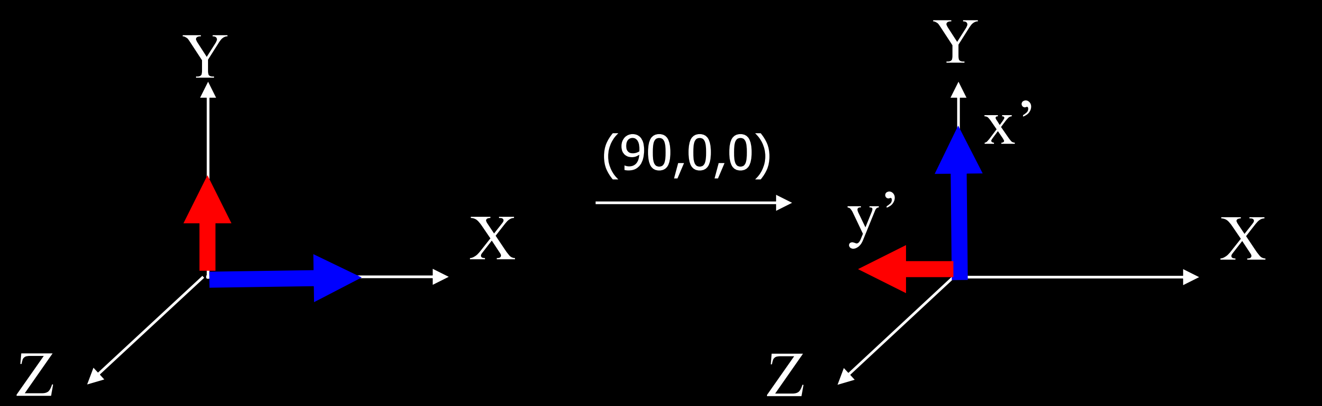


# Fixed Angle vs. Euler Angle

- Fixed angle: (90,45,90) in x-y-z order

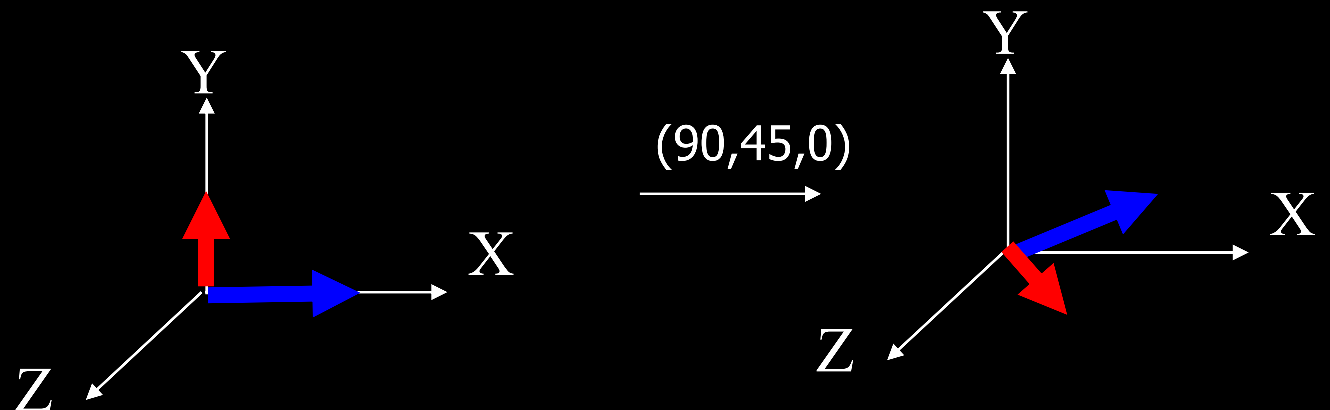



- Euler angle: (90,45,90) in z'-y'-x' order

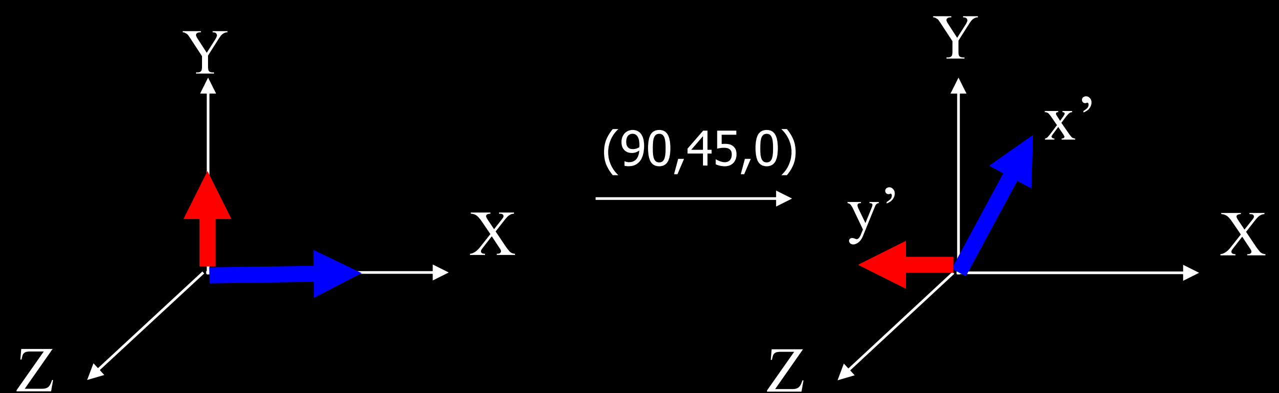


# Fixed Angle vs. Euler Angle

- Fixed angle: (90,45,90) in x-y-z order

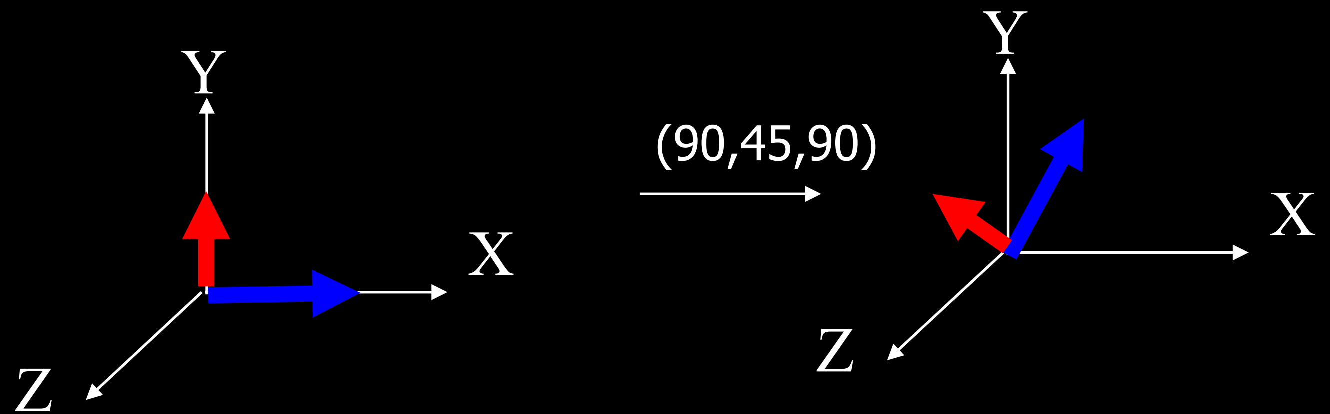



- Euler angle: (90,45,90) in z’-y’-x’ order

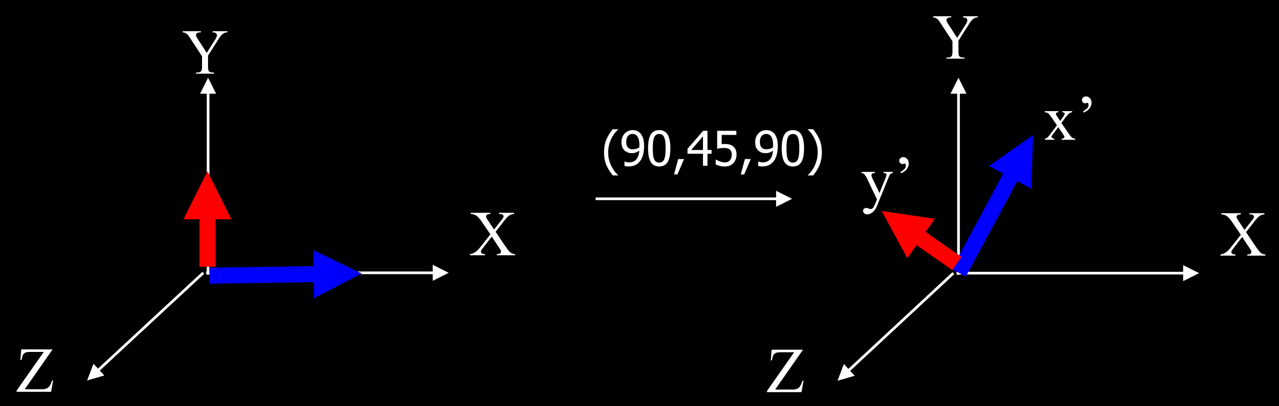


# Fixed Angle vs. Euler Angle

- Fixed angle: (90,45,90) in x-y-z order

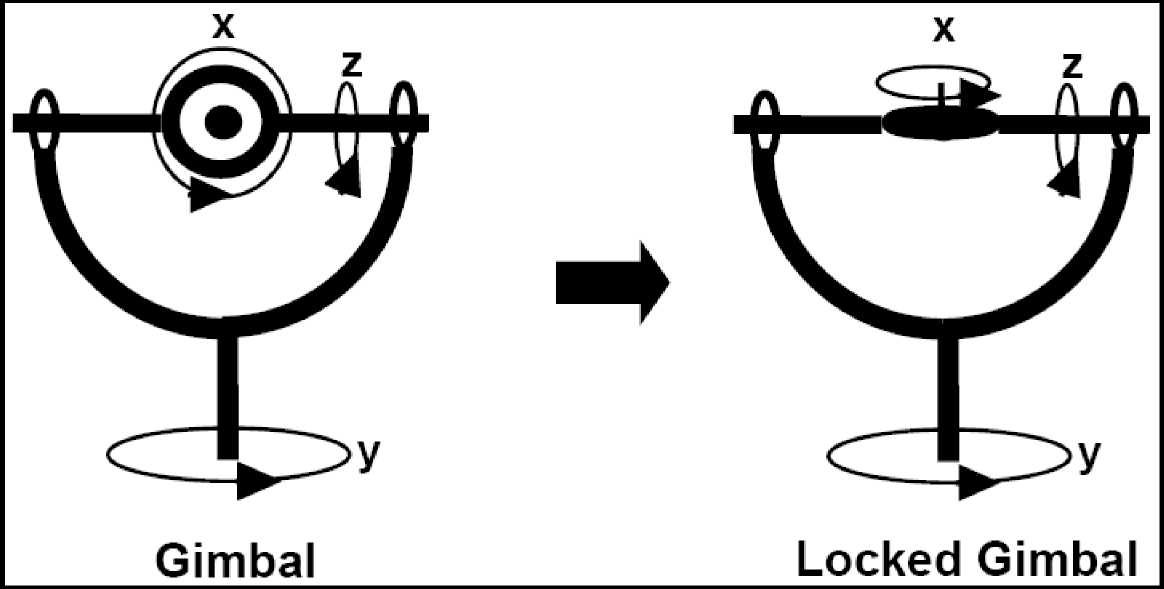



- Euler angle: (90,45,90) in z'-y'-x' order



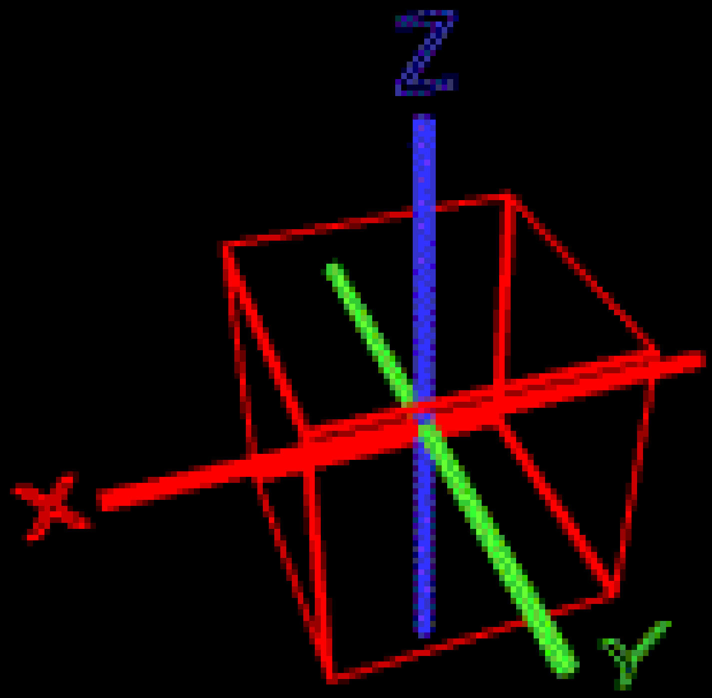
# Gimbal Lock

- A gimbal is a mechanical device allowing the rotation of an object in multiple dimensions
- Gimbal lock occurs when two of the rotation axes align



# Gimbal Lock

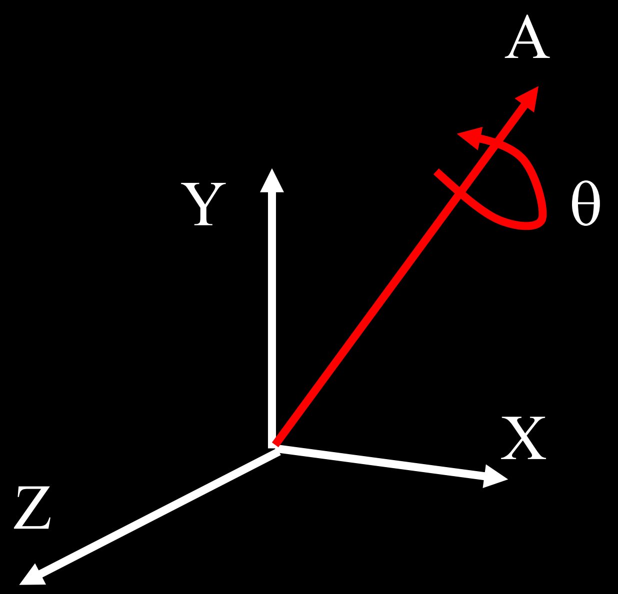
- Gimbal lock is a basic problem with 3D representations using fixed or Euler angles




http://www.anticz.com/eularqua.htm

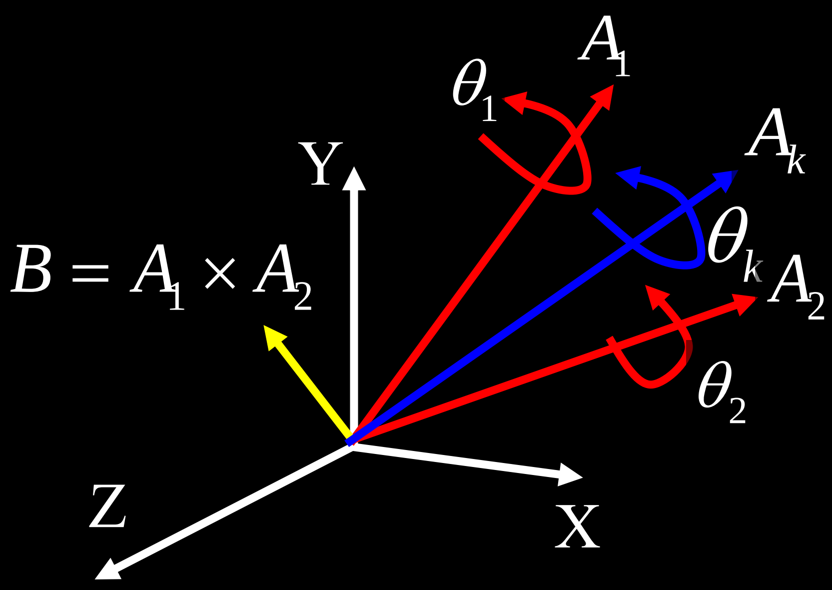
# Axis Angle Representation

- Euler's rotation theorem
  - Any 3-D rotation can be described by 4 parameters
- Rotate about $A$ by $\theta$ $(A_x, A_y, A_z, \theta)$



# Axis Angle Interpolation

- Interpolate axis and angle separately




$$B = A_1 \times A_2$$

$$\phi = \cos^{-1} \frac{A_1 \bullet A_2}{|A_1||A_2|}$$

$$A_k = R_B(k\phi)A_1$$

$$\theta_k = (1-k)\theta_1 + k\theta_2$$

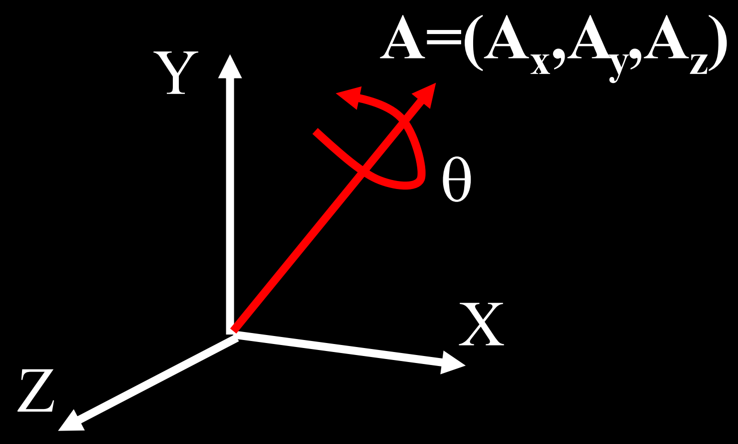
# Axis Angle vs. Quaternion

- Axis angle
  - Can interpolate the axis and angle separately
  - No gimbal lock
  - Cannot compose rotations efficiently
- Quaternion
  - Good interpolation
  - No gimbal lock
  - Can be composed

# Quaternion

- 4-tuple of real numbers
  - q=(s,x,y,z) or [s,v]
  - s is a scalar; v is a vector
- Same information as axis/angle but in a different form

$$q = \left[ \cos(\frac{\theta}{2}), \sin(\frac{\theta}{2}) \cdot (A_x, A_y, A_z) \right]$$



# Quaternion Math

- Addition

$$[s_1,\ v_1]+[s_2,\ v_2]=[s_1+s_2,\ v_1+v_2]$$

- Multiplication

$$[s_1,\ v_1]\cdot[s_2,\ v_2]=[s_1 s_2 - v_1\cdot v_2,\quad s_1 v_2 + s_2 v_1 + v_1\times v_2]$$

- Multiplication is associative but not commutative

$$q_1(q_2 q_3)=(q_1 q_2)q_3 \qquad\qquad q_1 q_2 \neq q_2 q_1$$

# Quaternion Math (cont.)

- A point in space is represented as [0, x, y, z]
- Multiplicative identity $$q \cdot [1, 0, 0, 0] = q$$
- Inverse $$q^{-1} = \frac{[s, \ -v]}{\|q\|^2}$$

$$\|q\| = \sqrt{s^2 + x^2 + y^2 + z^2}$$

$$qq^{-1} = [1, 0, 0, 0]$$


ILE5030 Computer Animation and Special Effects 22S

# Quaternion Rotation

- To rotate a vector *v* using quaternion
  - Represent the vector as [0, *v*]
  - Represent the rotation as a quaternion *q*

$$v' = Rot_q(v) = q \cdot v \cdot q^{-1}$$

- *q* and –*q* represent the same orientation

# Compose Rotations

$$
\begin{aligned}
Rot_q(Rot_p(v)) &= Rot_q(pvp^{-1}) \\
&= qpvp^{-1}q^{-1} \\
&= qpv(qp)^{-1} \\
&= Rot_{qp}(v)
\end{aligned}
$$

Prove by yourself that $p^{-1}q^{-1} = (qp)^{-1}$

# Summary of Rotation Representations

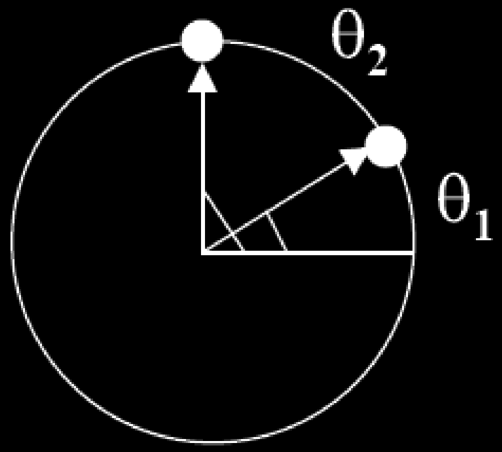
- Rotation Matrix
  - orthornormal columns/rows
  - bad for interpolation
- Fixed Angle
  - rotate about global axes
  - bad for interpolation, gimbal lock
- Euler Angle
  - rotate about local axes
  - same problem as fixed angle

# Summary of Rotation Representations

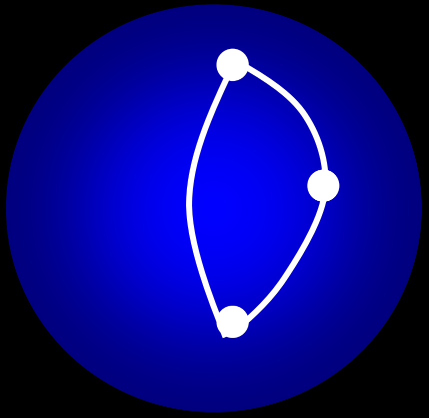
- Axis angle
  - rotate about $A$ by $\theta$, $(A_x, A_y, A_z, \theta)$
  - good interpolation, no gimbal lock
  - bad for compounding rotations
- Quaternion
  - similar to axis angle but in different form
  - $q=[s,v]$
  - good for compounding rotations

# Visualizing Rotations

- View rotations as points lying on an n-D sphere




1-angle rotation
unit circle in 2D space

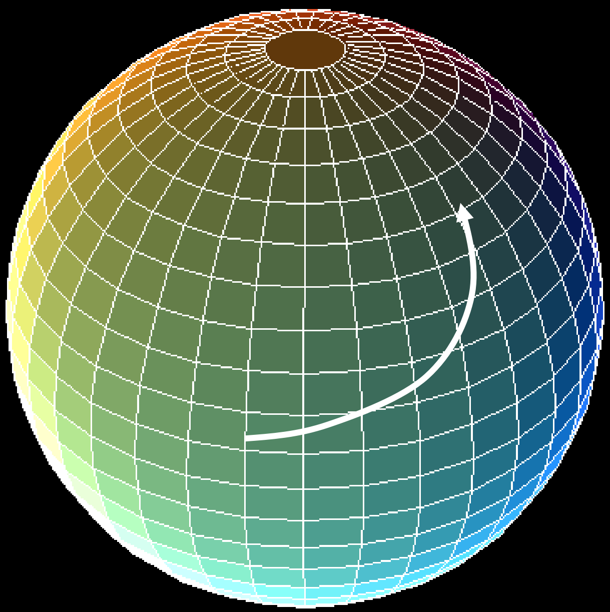


2-angle rotation
unit sphere in 3D space

- Interpolating rotation means moving on n-D sphere
- How about 3-angle rotation (quaternion)?

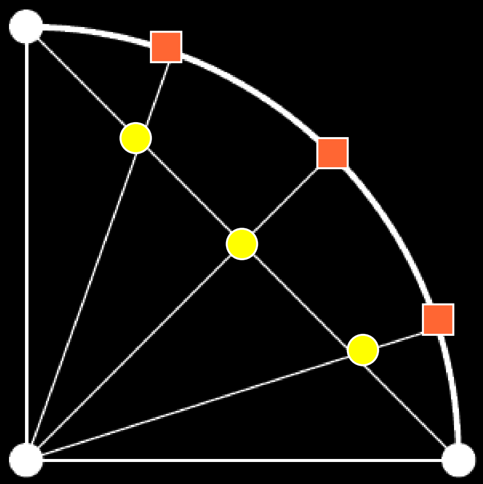
# Quaternion Interpolation

- A quaternion is a point on a 4D unit sphere
- Unit quaternion: $q=(s,x,y,z)$, $||q|| = 1$
- Interpolating rotations means moving on 4D sphere



# Linear Interpolation

- Linear interpolation generates unequal spacing of points after projecting to circle

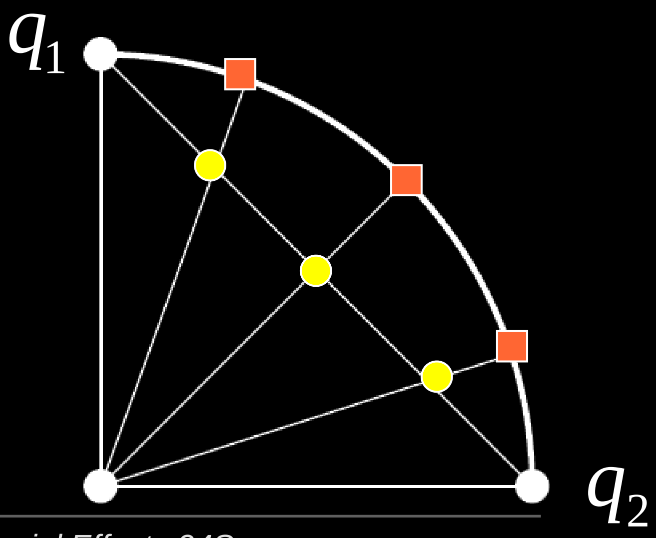


# Spherical Linear Interpolation (slerp)

- Want equal increment along arc connecting two quaternions on the spherical surface

$$slerp(q_1, q_2, u) = \frac{\sin(1-u)\Omega}{\sin\Omega} q_1 + \frac{\sin u\Omega}{\sin\Omega} q_2$$

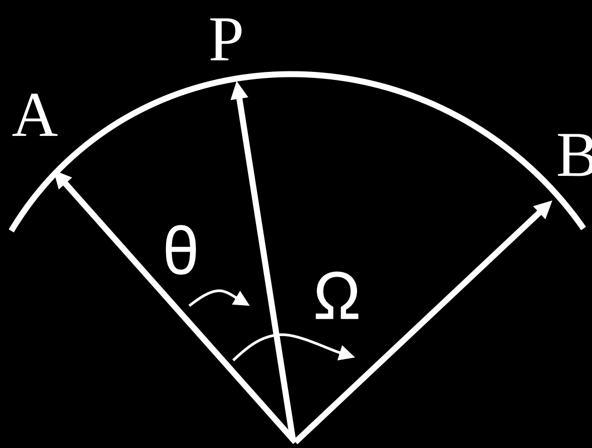
- Normalize to regain unit quaternion



# Proof of Slerp Equation

- It can be proved that

$$P = \frac{\sin(\Omega - \theta)}{\sin\Omega} A + \frac{\sin\theta}{\sin\Omega} B$$



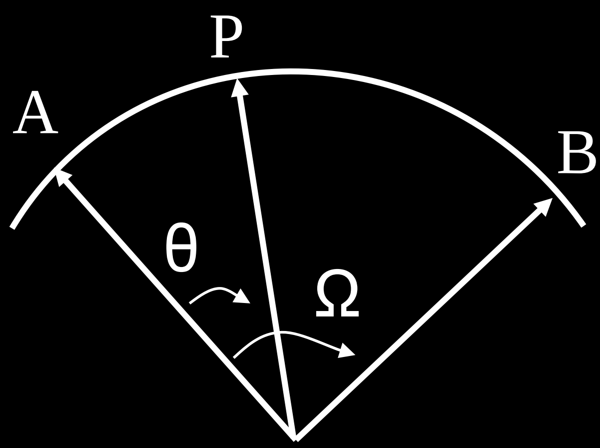

$$P = \alpha A + \beta B$$

$$\|P\| = 1$$

$$AB = \cos\Omega$$

$$AP = \cos\theta$$

# Proof of Slerp Equation (cont.)




$$P = \alpha A + \beta B$$

$$\|P\| = 1$$

$$A \cdot B = \cos\Omega$$

$$A \cdot P = \cos\theta$$

$$A(\alpha A + \beta B) = \alpha|A|^2 + \beta A \cdot B = \alpha|A|^2 + \beta\cos\Omega = \cos\theta$$

$$\alpha + \beta\cos\Omega = \cos\theta$$

$$|P|^2 = P \cdot P = \alpha^2|A|^2 + 2\alpha\beta A \cdot B + \beta^2|B|^2 = 1$$

$$\alpha^2 + 2\alpha\beta\cos\Omega + \beta^2 = 1$$

# Two equations for Two unknowns

$$\alpha + \beta \cos\Omega = \cos\theta$$

$$\longrightarrow \alpha^2 + 2\alpha\beta\cos\Omega + \beta^2\cos^2\Omega = \cos^2\theta$$

$$\alpha^2 + 2\alpha\beta\cos\Omega + \beta^2 = 1$$

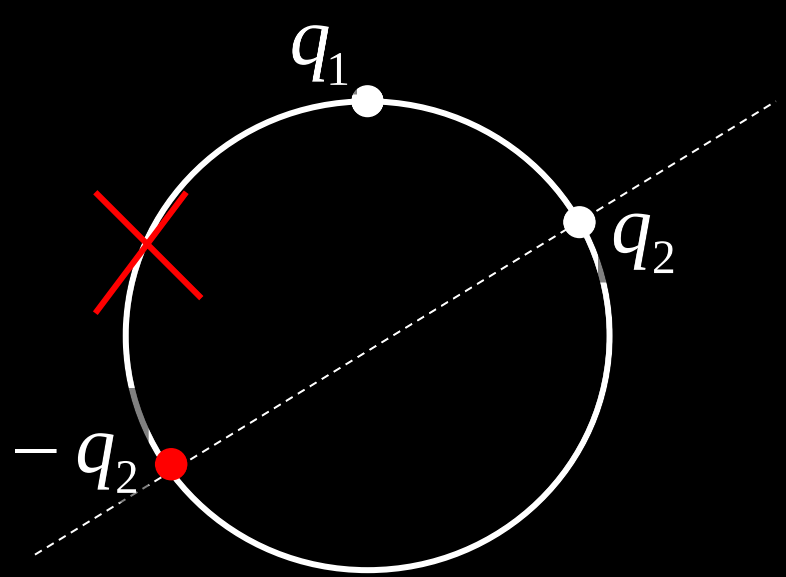
$$1 - \beta^2 + \beta^2\cos^2\Omega = \cos^2\theta$$

$$\beta^2\sin^2\Omega = \sin^2\theta$$

$$\alpha = \frac{\sin(\Omega - \theta)}{\sin\Omega} \longleftarrow \beta = \frac{\sin\theta}{\sin\Omega}$$

# Slerp: Pick Shortest Path

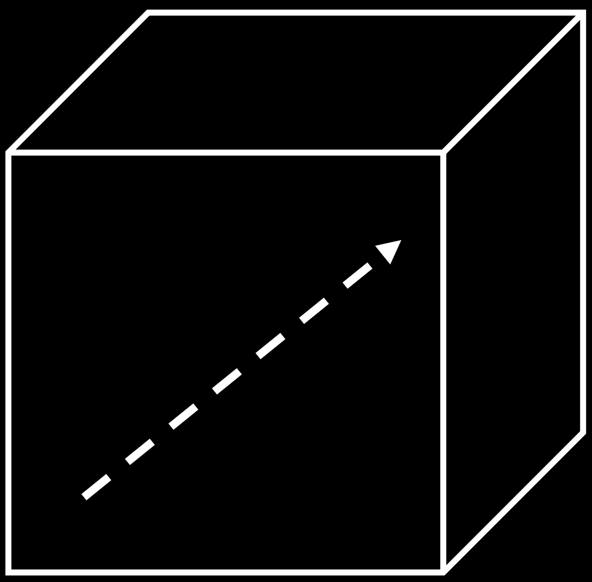
- Recall that $q$ and $-q$ represent the same rotation
- Slerp can go the LONG way!
- Have to go the short way $q_1 \cdot q_2 > 0$

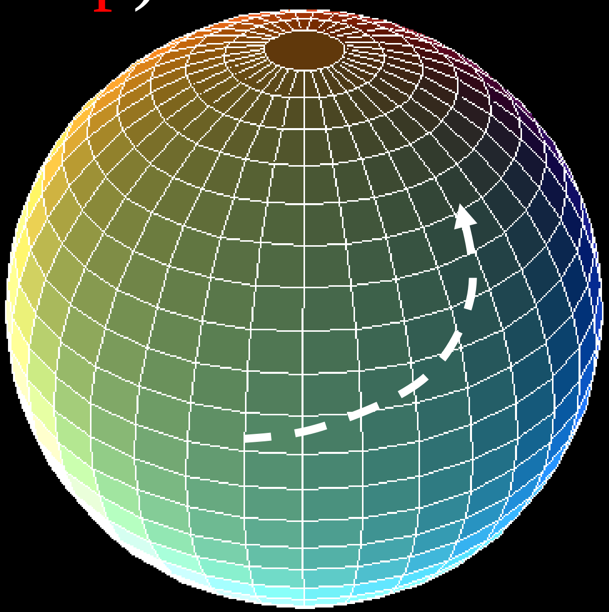


# Useful Analogies

Euclidean Space
Position
Linear interpolation

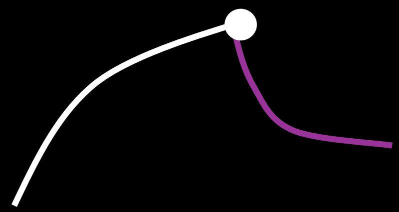
4D Spherical Space
Orientation
Spherical linear interpolation (slerp)





# What if there are multiple segments?

- As linear interpolation in Euclidean space, we can have first order discontinuity

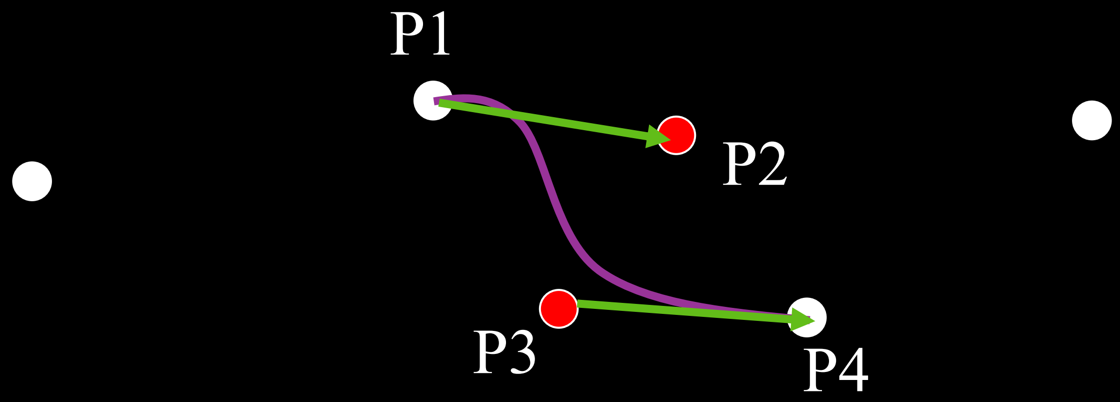


- Need a cubic curve interpolation to maintain first order continuity in Euclidean space
- Similarly, slerp can have 1st order discontinuity
- We also need a cubic curve interpolation in 4D spherical space for 1st order continuity

# Bezier Interpolation in Euclidean Space

$$p(u) = [u3\ u2\ u\ 1] \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 3 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} P1 \\ P2 \\ P3 \\ P4 \end{pmatrix}$$

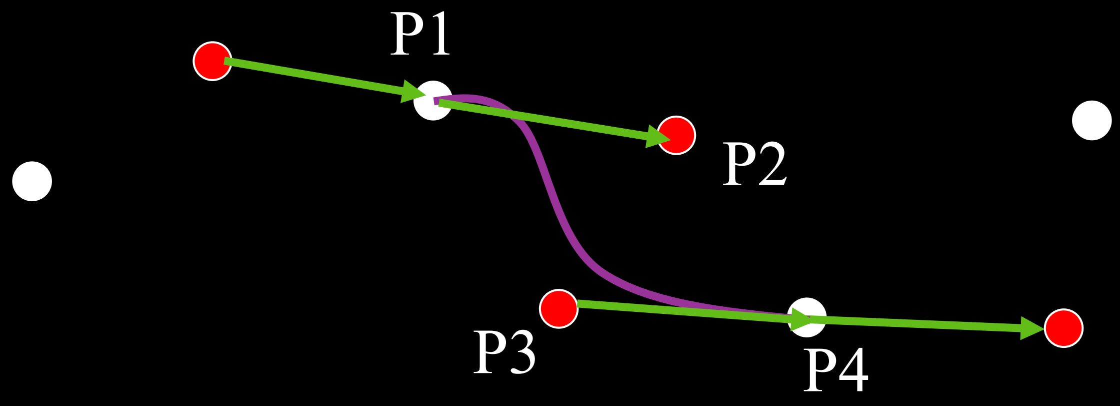
$$p'(0) = 3(p2-p1),\ p'(1)=3(p4-p3)$$



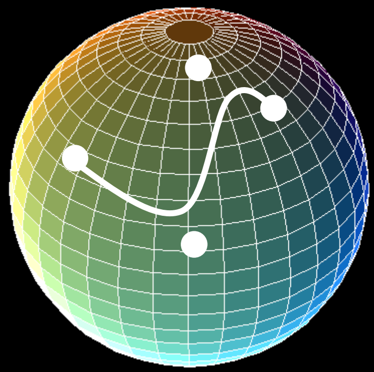
# Bezier Interpolation in Euclidean Space

Colinearity of the control points at either side of an endpoint guarantees the 1st order continuity

p'(0) = 3(p2-p1), P'(1)=3(p4-p3)



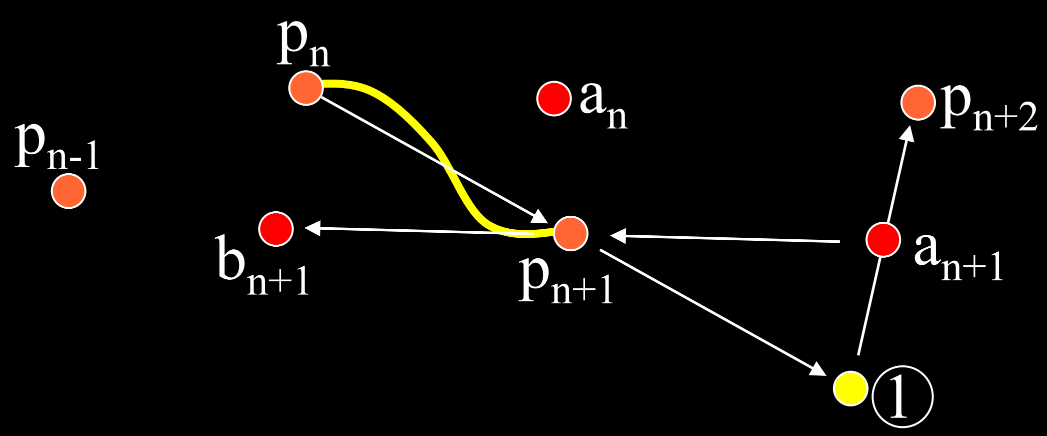
# Bezier Interpolation of Quaternions



- Bezier interpolation on 4D sphere?
  - How are control points generated?
  - How are cubic splines defined?
- Control points are automatically generated as it is not intuitive to manually adjust them on a 4D sphere
- Construct Bezier curves by iteratively linear interpolation → applying slerp
- Let's first see how to do the above two procedures in Euclidean space

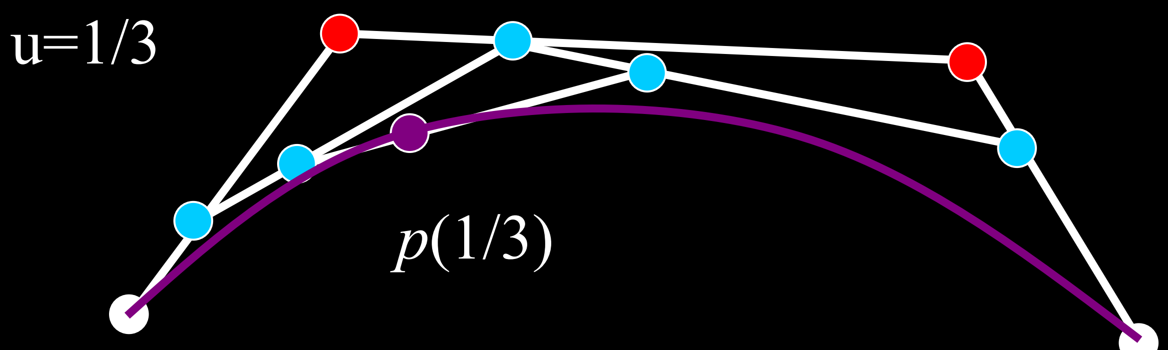
# Generating Collinear Control Points

$$p(u) = [u^3\ u^2\ u\ 1] \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 3 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{bmatrix} p_n \\ a_n \\ b_{n+1} \\ p_{n+1} \end{bmatrix}$$

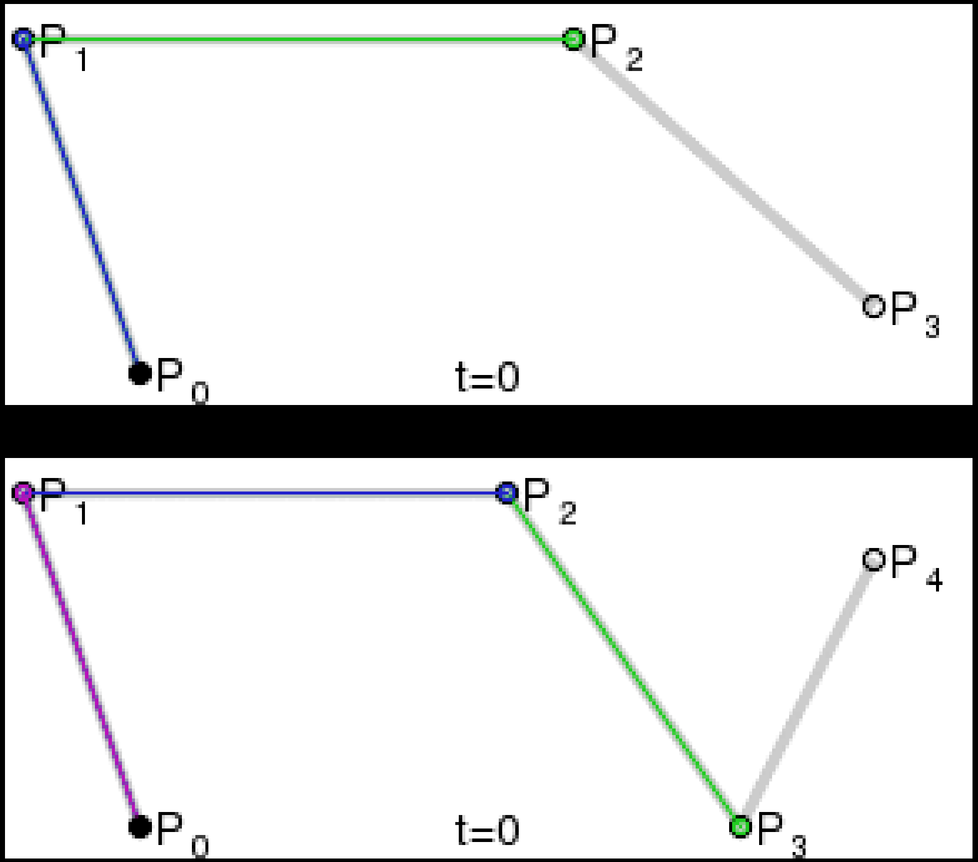


# De Casteljau Construction of Bézier Curve

- Constructing Bezier curve by multiple linear interpolation



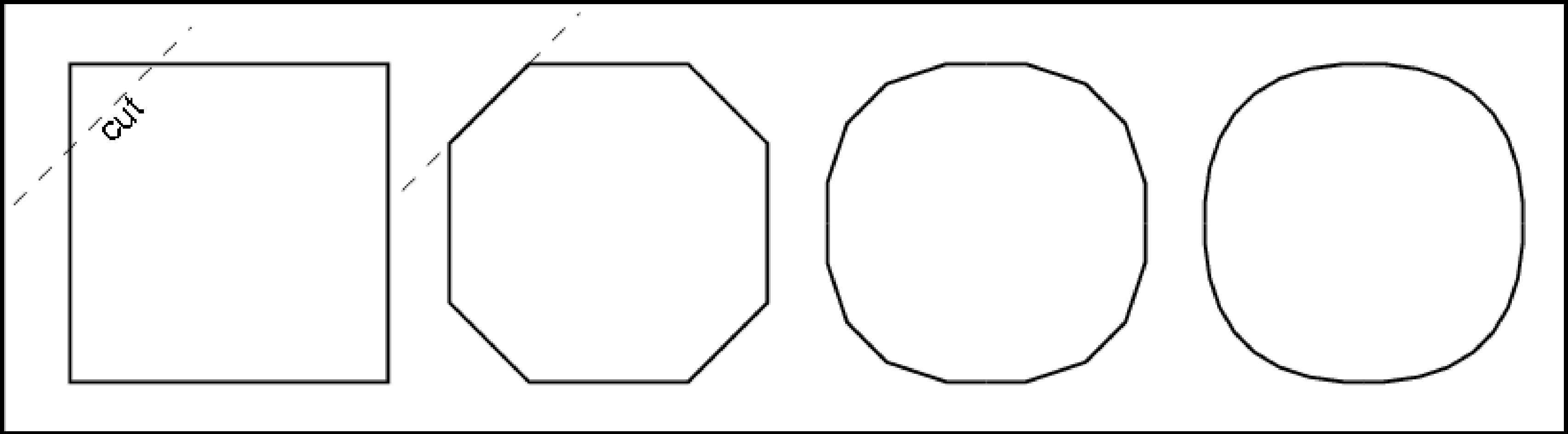
# De Casteljau Construction of Bézier Curve




from www.wikipedia.org

# Geometric Intuition for Bézier Curves

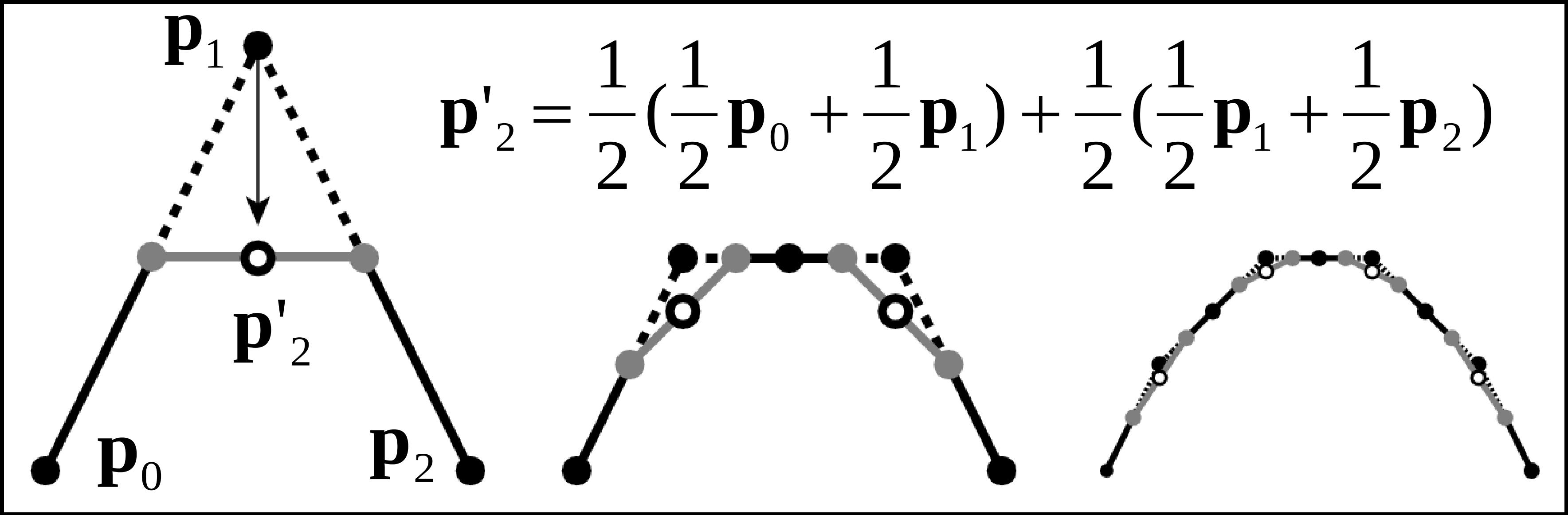
- By repeatedly cutting the corners off a polygon, we approach a smooth curve




- This is called subdivision!

# Subdivision Scheme

- Defines a curve by breaking a simpler curve into smaller pieces
- The limit curve (obtained by subdividing infinitely many times) will be a smooth curve



# Subdivision Rule

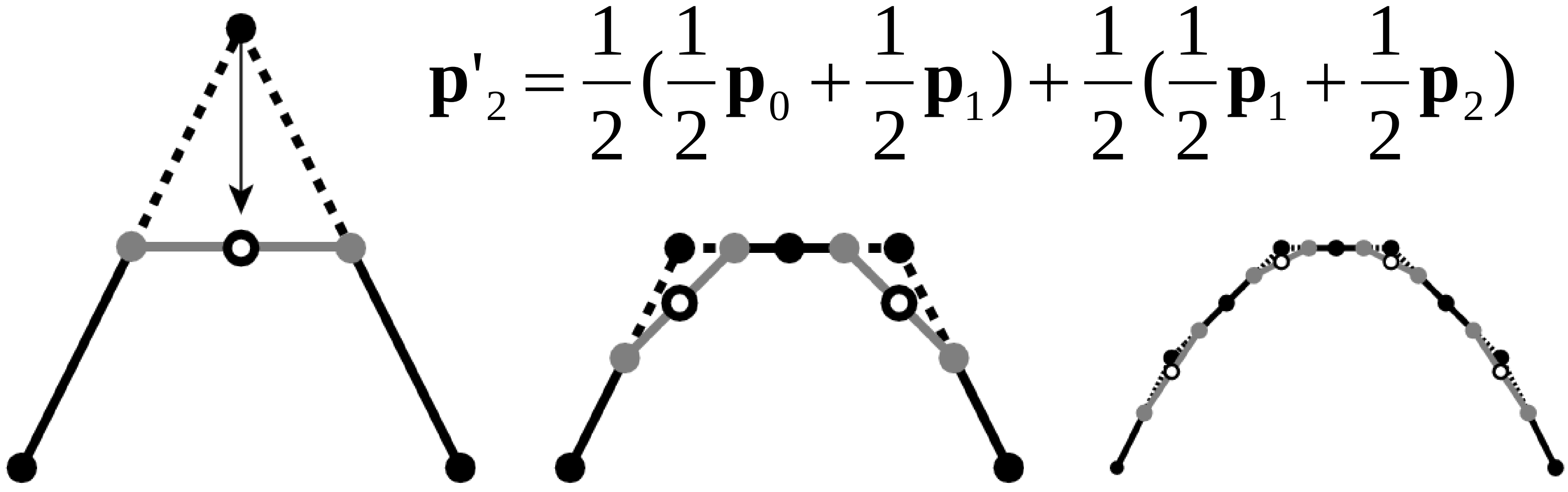
- The above rule can be generalized

$$\mathbf{p}(u) = (1-u)((1-u)\mathbf{p}_0 + u\mathbf{p}_1) + u((1-u)\mathbf{p}_1 + u\mathbf{p}_2)$$

- Regrouping terms gives the quadratic Bézier

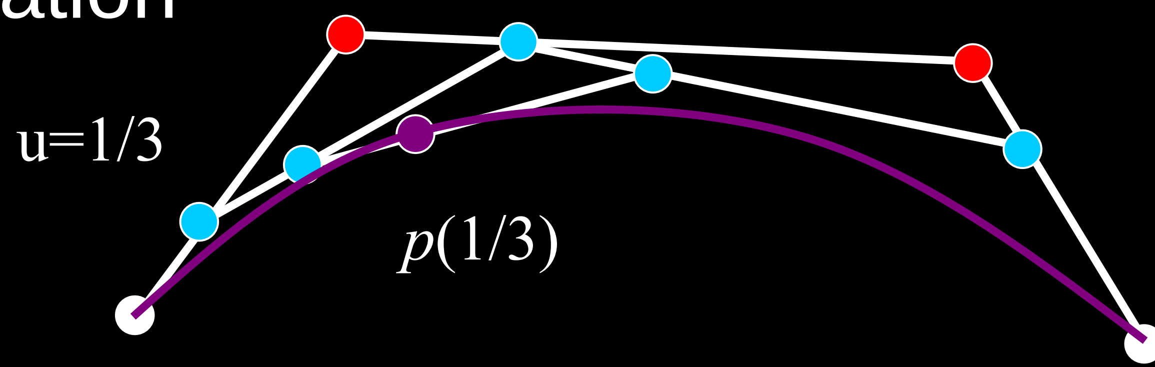
$$\mathbf{B}_2(u) = (1-u)^2\mathbf{p}_0 + 2u(1-u)\mathbf{p}_1 + u^2\mathbf{p}_2$$

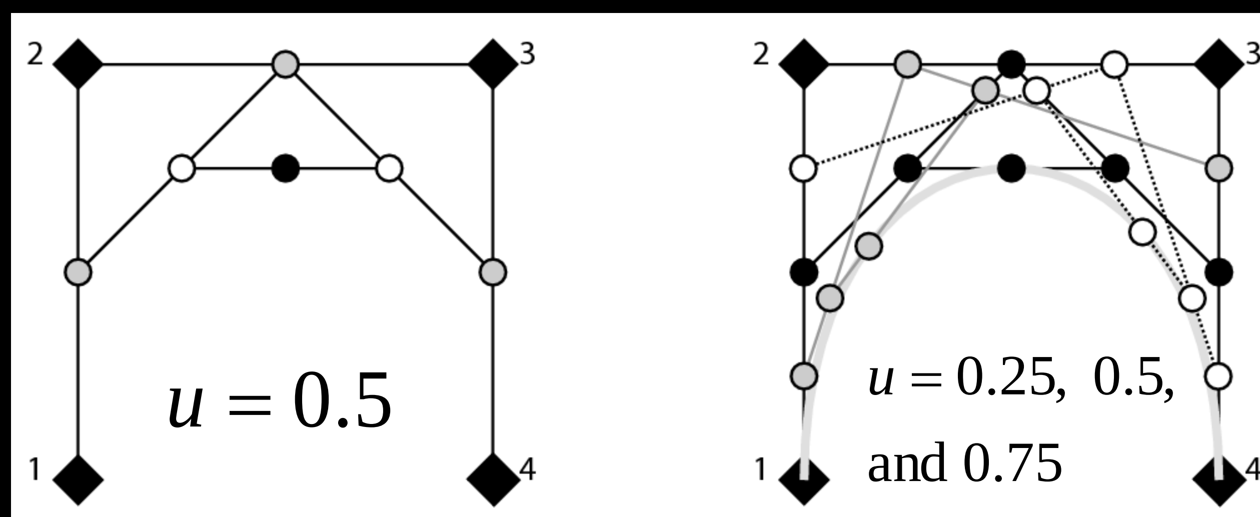
$$\mathbf{p}'_2 = \frac{1}{2}(\frac{1}{2}\mathbf{p}_0 + \frac{1}{2}\mathbf{p}_1) + \frac{1}{2}(\frac{1}{2}\mathbf{p}_1 + \frac{1}{2}\mathbf{p}_2)$$



# De Casteljau Algorithm

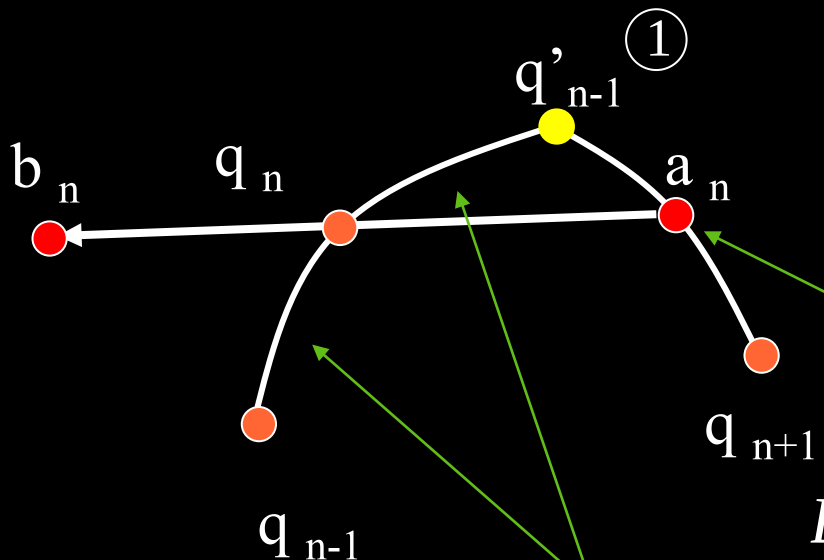
- Constructs Bezier curve using a sequence of linear interpolation



# Bezier Interpolation of Quaternions

- Automatically generating interior (spherical) control point




Bisect the span

$$Bisect(q'_{n-1}, q_{n+1}) = \frac{q'_{n-1} + q_{n+1}}{\left\| q'_{n-1} + q_{n+1} \right\|}$$

Double the arc

$$q'_{n-1} = double(q_{n-1}, q_n) = 2(q_{n-1} \cdot q_n) q_n - q_{n-1}$$

# De Casteljau Construction on 4D Sphere

$$p_1 = slerp(q_n, a_n, \tfrac{1}{3})$$
$$p_2 = slerp(a_n, b_{n+1}, \tfrac{1}{3})$$
$$p_3 = slerp(b_{n+1}, q_{n+1}, \tfrac{1}{3})$$
$$p_{12} = slerp(p_1, p_2, \tfrac{1}{3})$$
$$p_{23} = slerp(p_2, p_3, \tfrac{1}{3})$$
$$p = slerp(p_{12}, p_{23}, \tfrac{1}{3})$$