More on Transformations

COS 426, Spring 2020

Felix Heide

Princeton University

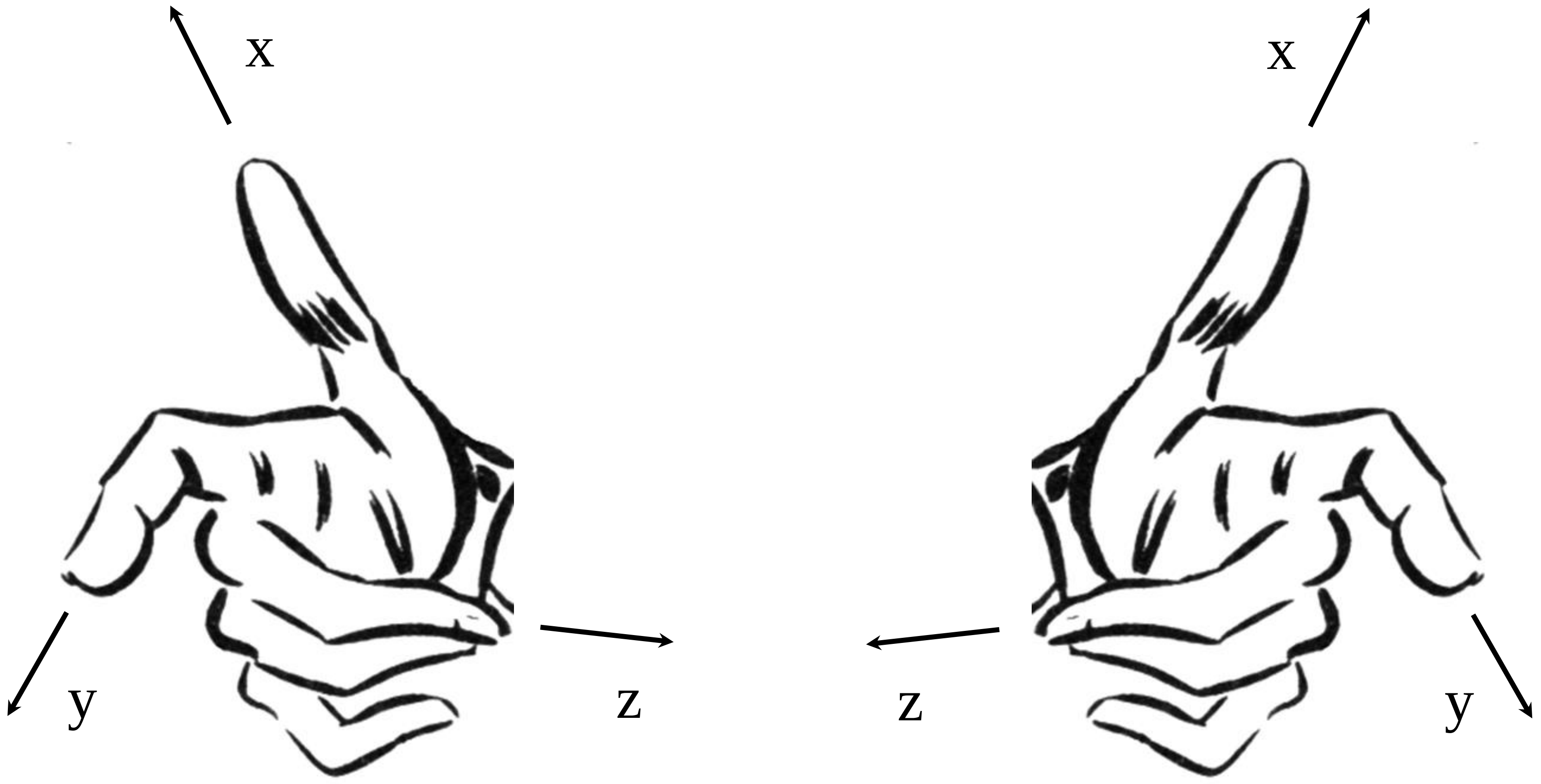
Agenda

Grab-bag of topics related to transformations:

- General rotations
  - Euler angles
  - Rodrigues's rotation formula
- Maintaining camera transformations
  - First-person
  - Trackball
- How to transform normals

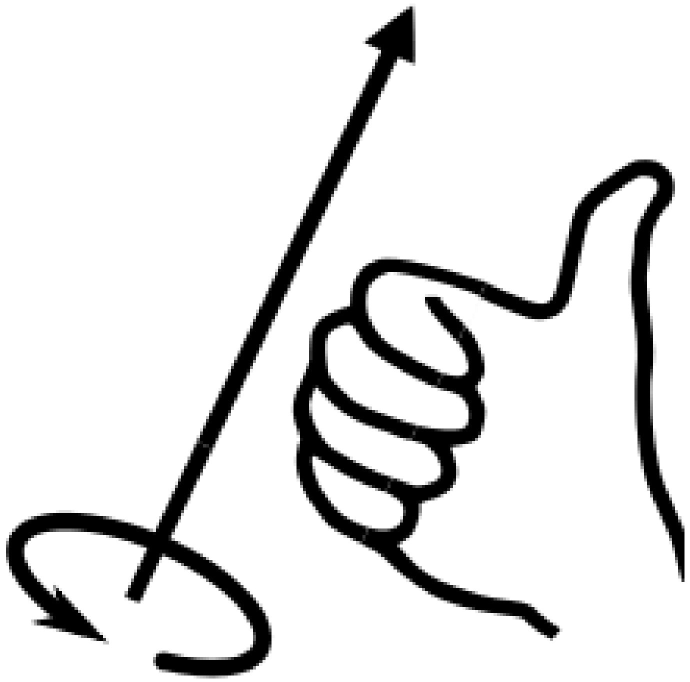
3D Coordinate Systems

- Right-handed vs. left-handed



3D Coordinate Systems

- Right-handed vs. left-handed
- Right-hand rule for rotations:
  positive rotation = counterclockwise
  rotation about axis

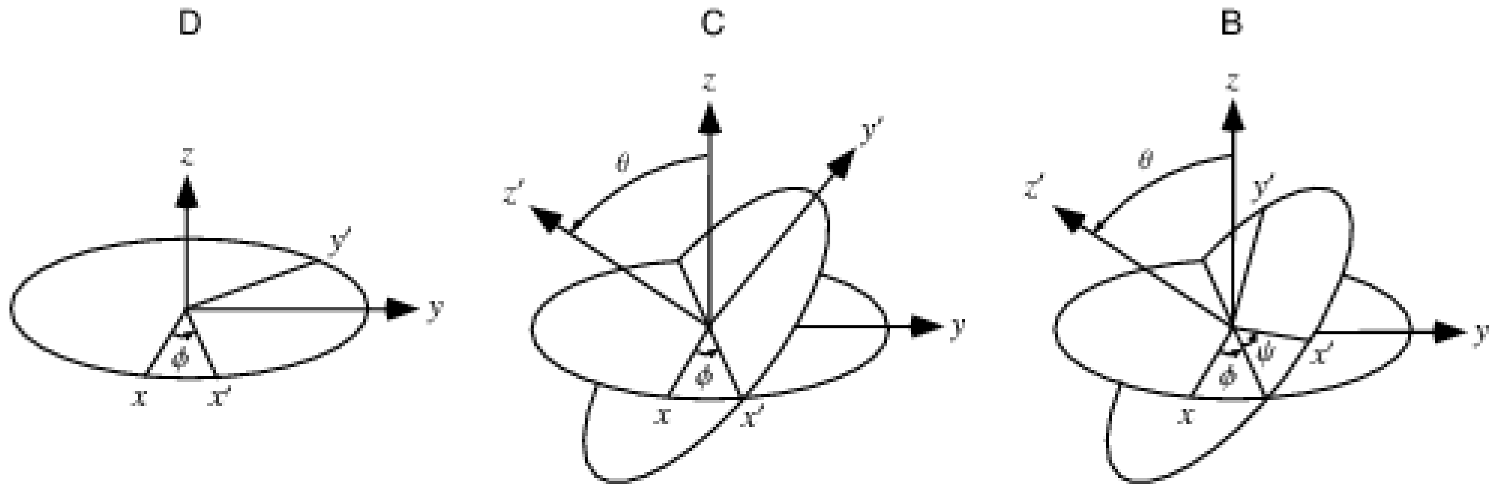


General Rotations

- Recall: set of rotations in 3-D is 3-dimensional
  - Rotation group SO(3)
  - Non-commutative
  - Corresponds to orthonormal 3×3 matrices with determinant = +1
- Need 3 parameters to represent a general rotation (Euler’s rotation theorem)

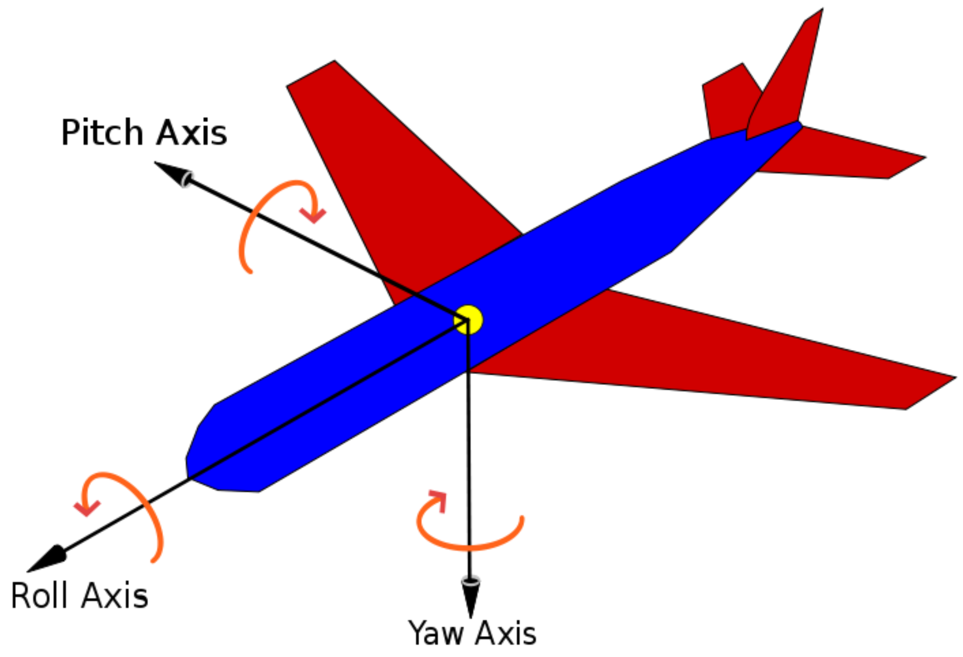
Euler Angles

- Specify rotation by giving angles of rotation about 3 coordinate axes
- 12 possible conventions for order of axes, but one standard is Z-X-Z



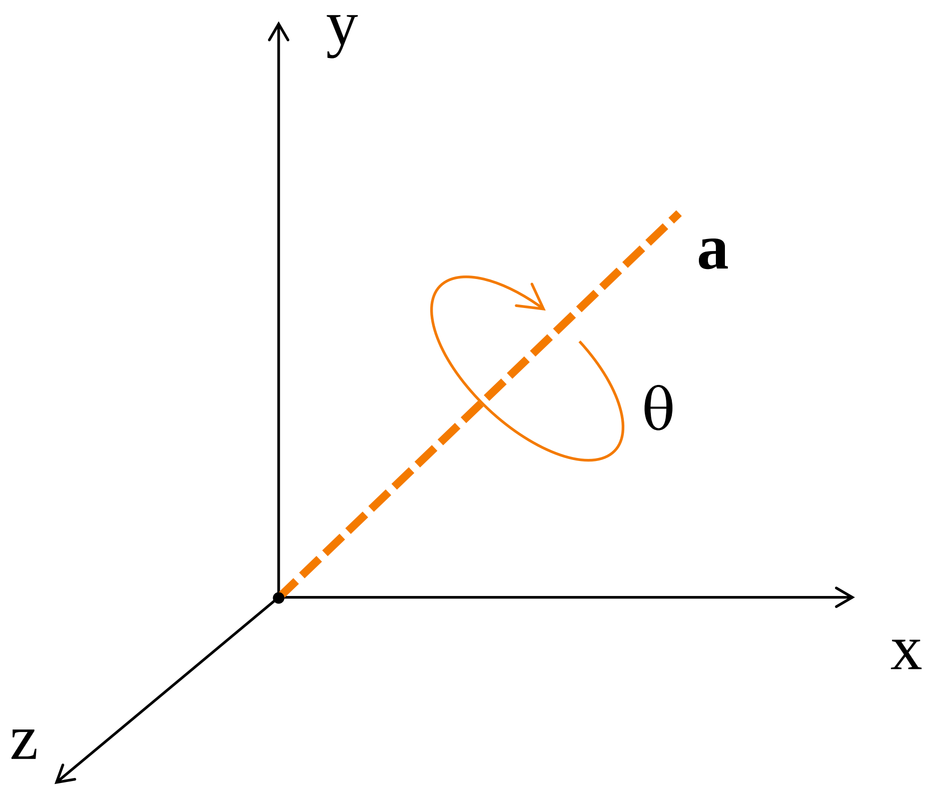
Euler Angles

- Another popular convention: X-Y-Z
- Can be interpreted as yaw, pitch, roll of airplane



Rodrigues's Formula

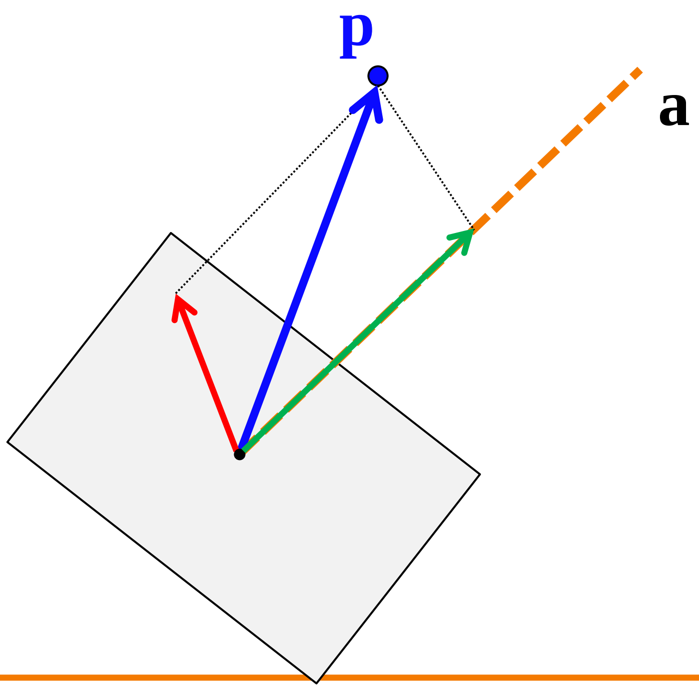
- Even more useful: rotate by an arbitrary angle (1 number) about an arbitrary axis (3 numbers, but only 2 degrees of freedom since unit-length)



Rodrigues's Formula

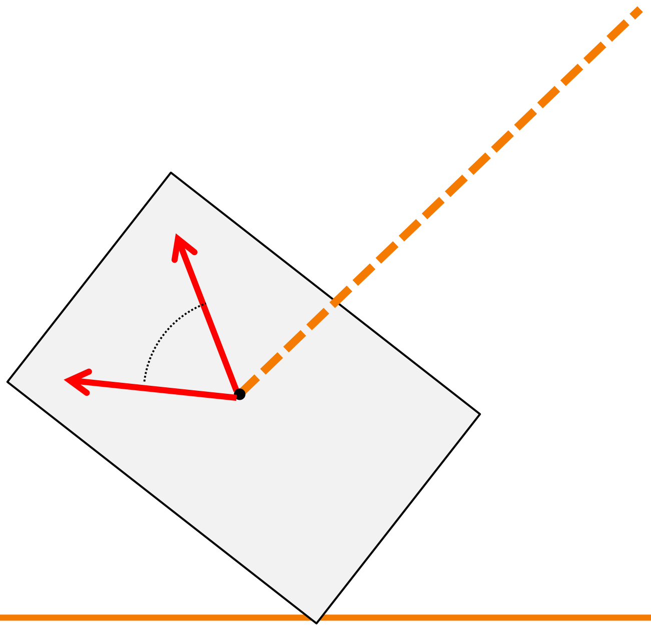
- An arbitrary point **p** may be decomposed into its components along and perpendicular to **a**

$$\mathbf{p} = \mathbf{a}\,(\mathbf{p}\cdot\mathbf{a}) + [\mathbf{p} - \mathbf{a}\,(\mathbf{p}\cdot\mathbf{a})]$$



Rodrigues's Formula

- Rotating component along **a** leaves it unchanged
- Rotating component perpendicular to **a** (call it $\mathbf{p}_\perp$) moves it to $\mathbf{p}_\perp \cos\theta + (\mathbf{a} \times \mathbf{p}_\perp) \sin\theta$



Rodrigues' Formula

- Putting it all together:

Why?

$$\mathbf{Rp} = \mathbf{a}\,(\mathbf{p} \cdot \mathbf{a}) + \mathbf{p}_\perp \cos\theta + (\mathbf{a} \times \mathbf{p}_\perp) \sin\theta$$

$$= \mathbf{a}\mathbf{a}^\mathsf{T}\mathbf{p} + (\mathbf{p} - \mathbf{a}\mathbf{a}^\mathsf{T}\mathbf{p}) \cos\theta + (\mathbf{a} \times \mathbf{p}) \sin\theta$$

- So: $\mathbf{R} = \mathbf{a}\mathbf{a}^\mathsf{T} + (\mathbf{I} - \mathbf{a}\mathbf{a}^\mathsf{T}) \cos\theta + [\mathbf{a}]_\times \sin\theta$

where $[\mathbf{a}]_\times$ is the "cross product matrix"

$$[\mathbf{a}]_\times = \begin{pmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{pmatrix}$$

Olinde Rodrigues, "Des lois géometriques qui regissent les déplacements d' un systéme solide dans l' espace, et de la variation des coordonnées provenant de ces déplacement considérées indépendant des causes qui peuvent les produire", *J. Math. Pures Appl.* **5** (1840), 380–440.

Rotating One Direction into Another

- Given two directions $\mathbf{d}_1$, $\mathbf{d}_2$ (unit length), how to find transformation that rotates $\mathbf{d}_1$ into $\mathbf{d}_2$?
    - There are many such rotations!
    - Choose rotation with minimum angle
- Axis = $\mathbf{d}_1 \times \mathbf{d}_2$
- Angle = $\mathrm{acos}(\mathbf{d}_1 \cdot \mathbf{d}_2)$

Agenda

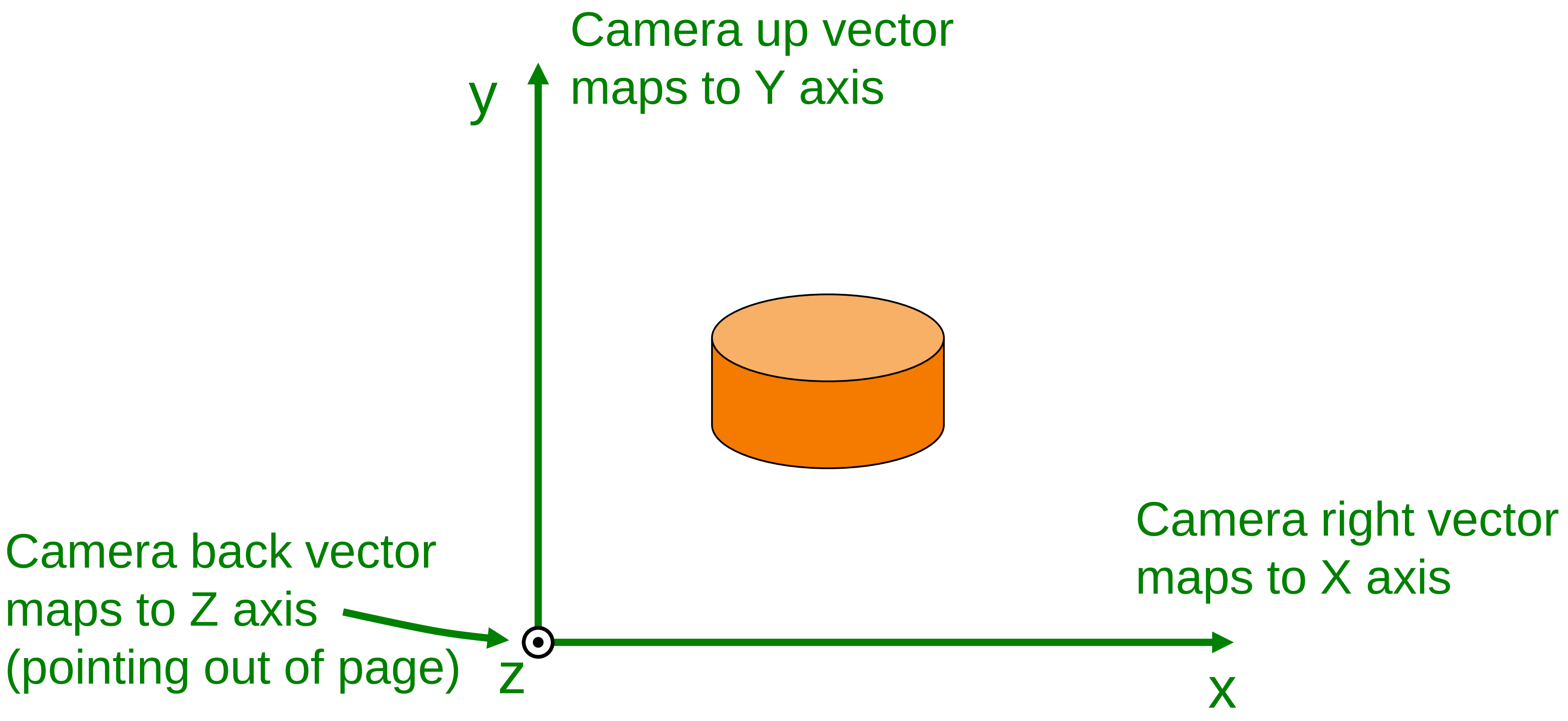
Grab-bag of topics related to transformations:

- General rotations
  - Euler angles
  - Rodrigues's rotation formula
- Maintaining camera transformations
  - First-person
  - Trackball
- How to transform normals

Camera Coordinates

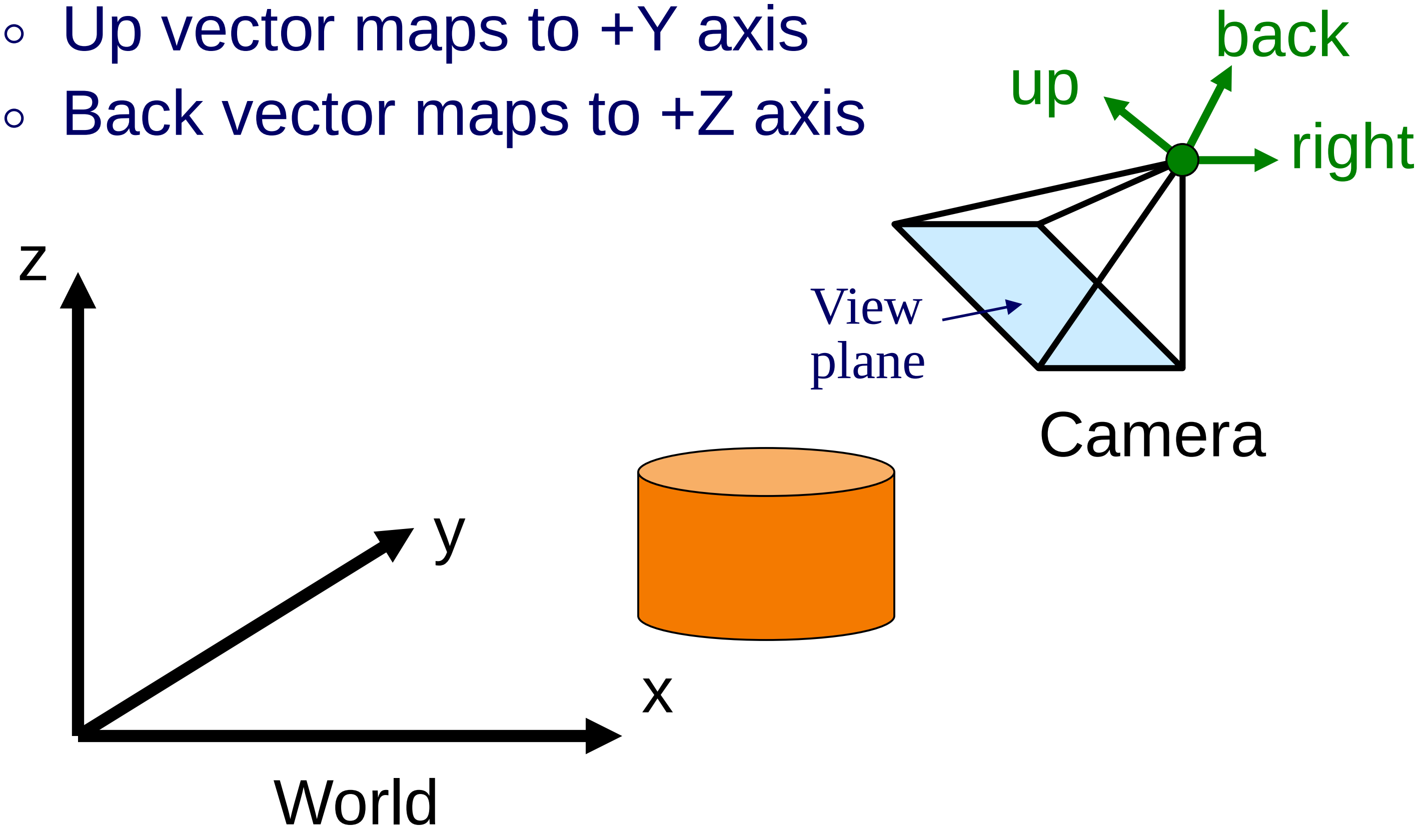
Canonical camera coordinate system

- Convention is right-handed (looking down –z axis)
- Convenient for projection, clipping, etc.



Viewing Transformation

- Mapping from world to camera coordinates
  - Eye position maps to origin
  - Right vector maps to +X axis
  - Up vector maps to +Y axis
  - Back vector maps to +Z axis



Finding the viewing transformation

- We have the camera (in world coordinates)
- We want T taking objects from world to camera

$$p^C = T\ p^W$$

- Trick: find T^{-1} taking objects in camera to world

$$p^W = T^{-1} p^C$$

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

Finding the Viewing Transformation

- Trick: map from camera coordinates to world
  - Origin maps to eye position
  - Z axis maps to Back vector
  - Y axis maps to Up vector
  - X axis maps to Right vector

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} R_x & U_x & B_x & E_x \\ R_y & U_y & B_y & E_y \\ R_z & U_z & B_z & E_z \\ R_w & U_w & B_w & E_w \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

- This matrix is T^{-1} so we invert it to get T … easy!

Maintaining Viewing Transformation

For first-person camera control, need 2 operations:

- Turn: rotate(θ, 0,1,0) in local coordinates
- Advance: translate(0, 0, –v*Δt) in local coordinates

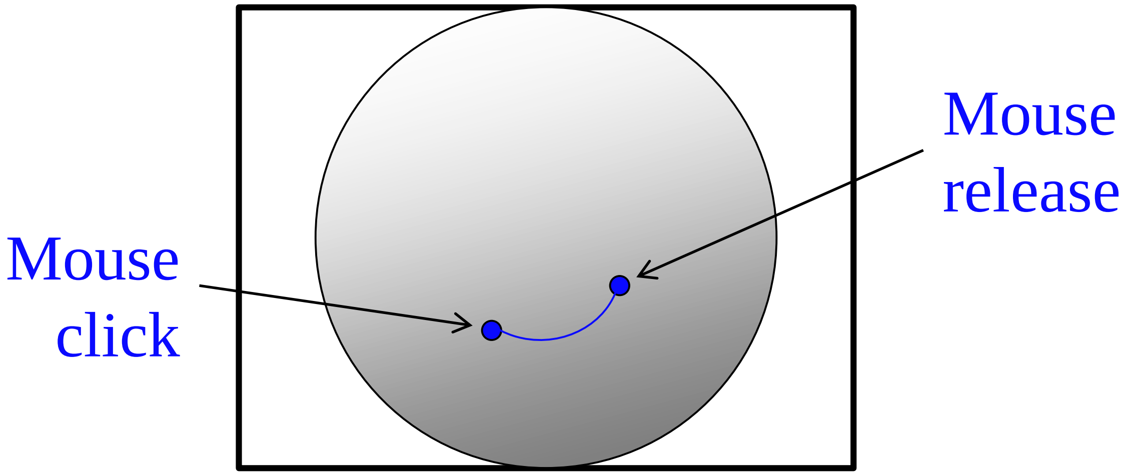
- Key: transformations act on local, not global coords
- To accomplish: right-multiply by translation, rotation

$$\mathbf{M}_{new} \leftarrow \mathbf{M}_{old}\mathbf{T}_{-v*\Delta t,z}\mathbf{R}_{\theta,y}$$

Maintaining Viewing Transformation

Object manipulation: “trackball” or “arcball” interface

- Map mouse positions to surface of a sphere



- Compute rotation axis, angle
- Apply rotation to global coords: left-multiply

$$\mathbf{M}_{new} \leftarrow \mathbf{R}_{\theta,a}\,\mathbf{M}_{old}$$

Agenda

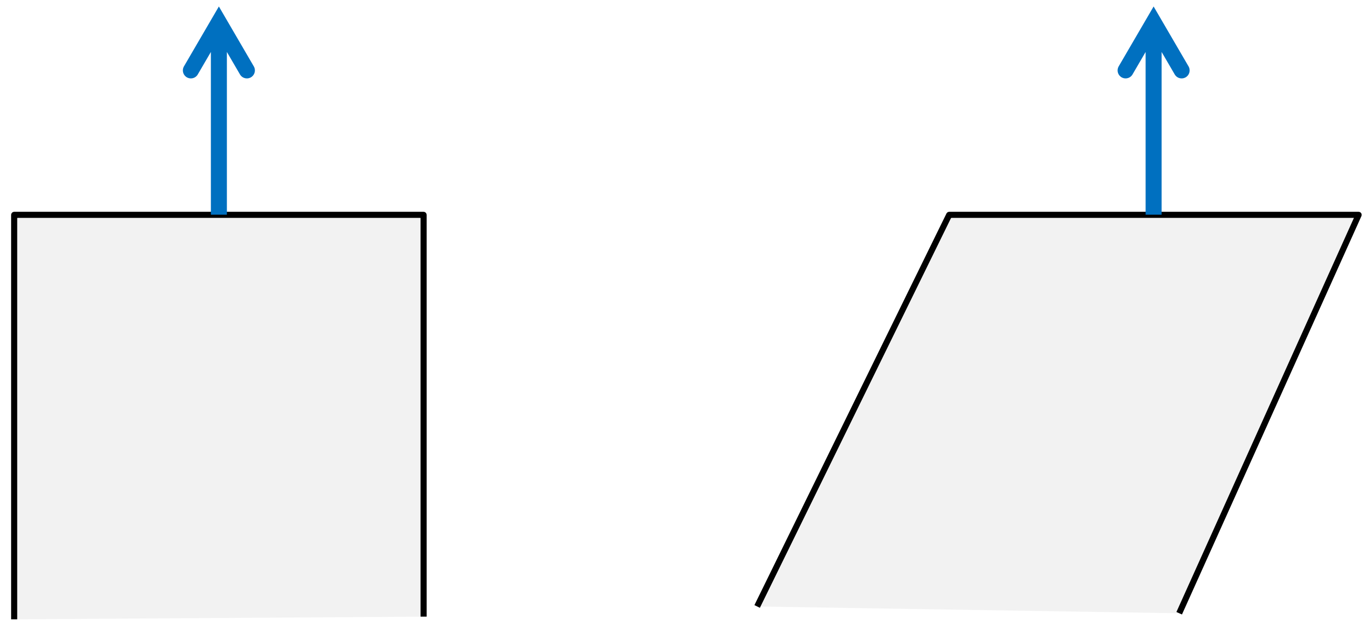
Grab-bag of topics related to transformations:

- General rotations
  - Euler angles
  - Rodrigues’s rotation formula
- Maintaining camera transformations
  - First-person
  - Trackball
- How to transform normals

Transforming Normals

Normals do not transform the same way as points!

- Not affected by translation
- Not affected by shear perpendicular to the normal



Transforming Normals

- Key insight: normal remains perpendicular to surface tangent

- Let **t** be a tangent vector and **n** be the normal

$$\mathbf{t} \cdot \mathbf{n} = 0 \qquad \text{or} \qquad \mathbf{t}^T\mathbf{n} = 0$$

- If matrix **M** represents an affine transformation, it transforms **t** as

$$\mathbf{t} \rightarrow \mathbf{M}_L\mathbf{t}$$

where $\mathbf{M}_L$ is the linear part (upper-left 3×3) of **M**

Transforming Normals

- So, after transformation, want

$$(\mathbf{M_L t})^T \mathbf{n}_{\text{transformed}} = 0$$

- But we know that

$$\mathbf{t}^T \mathbf{n} = 0$$

$$\mathbf{t}^T \mathbf{M_L}^T (\mathbf{M_L}^T)^{-1} \mathbf{n} = 0$$

$$(\mathbf{M_L t})^T (\mathbf{M_L}^T)^{-1} \mathbf{n} = 0$$

- So,

$$\mathbf{n}_{\text{transformed}} = (\mathbf{M_L}^T)^{-1} \mathbf{n}$$

Transforming Normals

- Conclusion: normals transformed by *inverse transpose* of *linear part* of transformation

- Note that for rotations, inverse = transpose, so inverse transpose = identity
  - normals just rotated

COS 426 Midterm exam

- This Thursday, March 12
- Regular time/place: 3:00-4:20, Friend 101
- Covers everything through week 5:
  color, image processing, shape representations, transformations (but not today's lecture)
  - Also responsible for material in required parts of first two programming assignments
- Closed book, no electronics,
  one page (double sided) of notes / formulas