CS4620/5620: Lecture 10

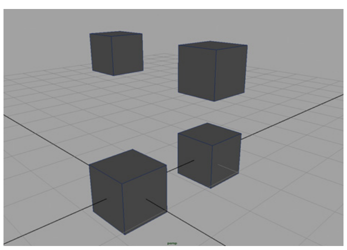
3D Transforms

Announcements

- 4621/5621 today in Olin 255 at 3:35
  - Go to TA office hours if you miss it

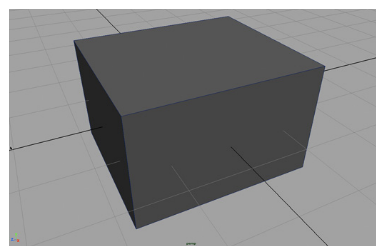
Translation

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



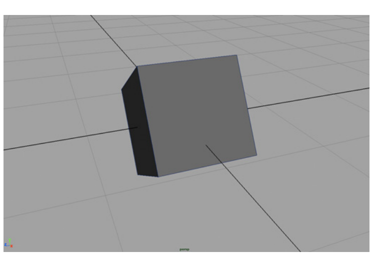
Scaling

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



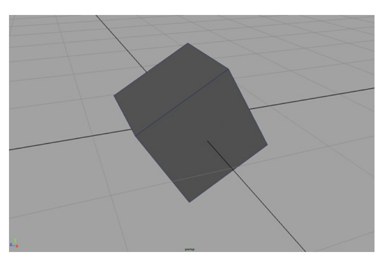
Rotation about *z* axis

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



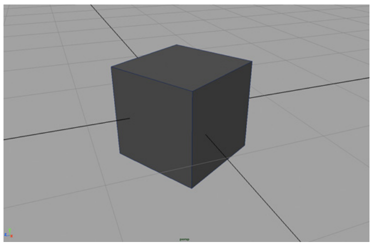
Rotation about *x* axis

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



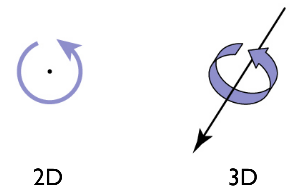
Rotation about y axis

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



General rotations

- A rotation in 2D is around a point
- A rotation in 3D is around an axis
  - so 3D rotation is w.r.t a line, not just a point
  - there are many more 3D rotations than 2D
    - a 3D space around a given point, not just 1D

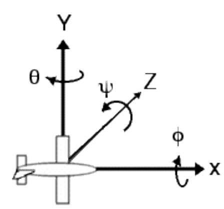


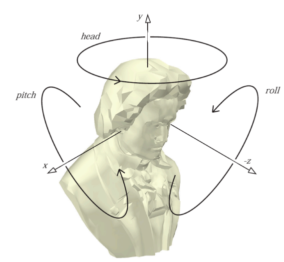
Specifying rotations

- In 3D, specifying a rotation is more complex
  - basic rotation about origin: unit vector (axis) and angle
    - convention: positive rotation is CCW when vector is pointing at you
- Many ways to specify rotation
  - Euler angles: 3 angles about 3 axes
  - (Axis, angle) rotation
  - Quaternions

3D rotations

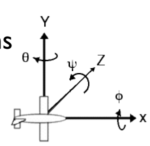
- An object can be oriented arbitrarily
- Euler angles: stack up three coord axis rotations
  - ZYX case: Rz(thetaz)*Ry(thetay)*Rx(thetax)
  - heading, attitude, bank (NASA standard airplane coordinates)
  - pitch, head, roll

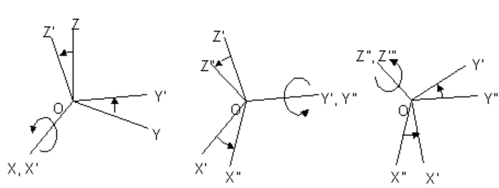




3D rotations

- NASA standard
- Euler angles: stack up three coord axis rotations
  - ZYX case: Rz(thetaz)*Ry(thetay)*Rx(thetax)





Specifying rotations: Euler rotations

- Euler angles

$$R(\theta_x, \theta_y, \theta_z) = R_z(\theta_z) R_y(\theta_y) R_x(\theta_x)$$

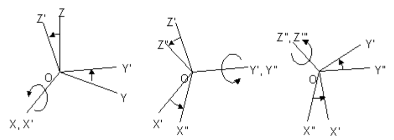
$$R(\theta_x, \theta_y, \theta_z) = \begin{bmatrix} c_y c_z & s_x s_y c_z - c_x s_z & c_x s_y s_z - s_x c_z & 0 \\ c_y s_z & s_x s_y s_z + c_x c_z & c_x s_y s_z - s_x c_z & 0 \\ -s_y & s_x c_y & c_x c_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$c_i = \cos(\theta_i)$$
$$s_i = \sin(\theta_i)$$

Euler angles

- Gimbal lock removes one degree of freedom

$$R(\theta_x,\theta_y,\theta_z)=\begin{bmatrix}0 & \sin(\theta_x-\theta_z) & \cos(\theta_x-\theta_z) & 0\\ 0 & \cos(\theta_x-\theta_z) & \sin(\theta_x-\theta_z) & 0\\ -1 & 0 & 0 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$



Coming up with the matrix

- Showed matrices for coordinate axis rotations
  - but what if we want rotation about some random axis?
- Compute by composing elementary transforms
  - transform rotation axis to align with *x* axis
  - apply rotation
  - inverse transform back into position
- Just as in 2D this can be interpreted as a similarity transform

Properties of Rotation Matrices

- Columns of R are mutually orthonormal: $RR^T=R^TR=I$
- Right-handed coordinate systems: $\det(R)=1$

$$\begin{vmatrix}a & b & c\\ d & e & f\\ g & h & k\end{vmatrix}=a\begin{vmatrix}e & f\\ h & k\end{vmatrix}-b\begin{vmatrix}d & f\\ g & k\end{vmatrix}+c\begin{vmatrix}d & e\\ g & h\end{vmatrix}$$

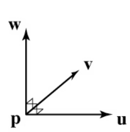
Building general rotations

- Using elementary transforms you need three
  - translate axis to pass through origin
  - rotate about *y* to get into *x-y* plane
  - rotate about *z* to align with *x* axis
- Alternative: construct frame and change coordinates
  - choose *p*, *u*, *v*, *w* to be orthonormal frame with *p* and *u* matching the rotation axis
  - apply similarity transform $T = F\, R_x(\theta)\, F^{-1}$

Orthonormal frames in 3D

- Useful tools for constructing transformations
- Recall rigid motions
  - affine transforms with pure rotation and translation
  - columns (and rows) form right-handed ONB
    - that is, an **o**rtho**n**ormal **b**asis

$$F=\begin{bmatrix}u_x & v_x & w_x\\ u_y & v_y & w_y\\ u_z & v_z & w_z\end{bmatrix}$$



Building 3D frames

- Given a vector **a** and a secondary vector **b**
  - The **u** axis should be parallel to **a**; the **u**–**v** plane should contain **b**
    - **u** = **u** / ||**u**||
    - **w** = **u** x **b**; **w** = **w** / ||w||
    - **v** = **w** x **u**
- Given just a vector **a**
  - The **u** axis should be parallel to **a**; don't care about orientation about that axis
    - Same process but choose arbitrary **b** first
    - Good choice is not near **a**: e.g. set smallest entry to 1

Building general rotations

- Alternative: construct frame and change coordinates
  - choose *p*, *u*, *v*, *w* to be orthonormal frame with *p* and *u* matching the rotation axis
  - apply similarity transform $T = F\, R_x(\theta)\, F^{-1}$
  - interpretation: move to *x* axis, rotate, move back
  - interpretation: rewrite *u*-axis rotation in new coordinates
  - (each is equally valid)

$$\begin{bmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) \\ 0 & sin(\theta) & cos(\theta) \end{bmatrix} \begin{bmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \\ w_x & w_y & w_z \end{bmatrix}$$

