3D Transformations and Perspective

CS 4620 Lecture 12
Announcements

• Demos on Monday
  – If you can’t make it, send mail to cs4620-staff-l@cornell.edu
  – Post to piazza

• A3 out tonight
  – Written and code due on Thu before break
  – Grading will be after break
  – Start early
    • Hierarchies, transformations

• 4621 class today. PPA1 out tonight
General Rotation Matrices

- 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
Specifying Rotations

- Many ways to specify rotation
  - Indirectly through frame transformations
  - Directly through
    - Euler angles: 3 angles about 3 axes
    - (Axis, angle) rotation: based on Euler’s theorem
    - Quaternions
Houston. Tranquility is going to put the track modes in P00 now.

Columbia, Houston. Over.

Columbia. Go.

Columbia, Houston. We noticed you are maneuvering very close to gimbal lock. I suggest you move back away. Over.

Yes. I am going around it, doing this CMC AUTO maneuvers to the PAD values of roll 270, pitch 101, yaw 45.

Roger, Columbia.

How about sending me a fourth gimbal for Christmas.

Columbia, Houston. You were unreadable. Say again please.

Disregard.
Building general rotations

• 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 \( T = FR_x(\theta)F^{-1} \)
  – interpretation: move to \( x \) axis, rotate, move back
  – interpretation: rewrite \( u \)-axis rotation in new coordinates
  – (each is equally valid)

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

– (note above is linear transform; add affine coordinate)
Derivation of General Rotation Matrix

- Axis angle rotation
Axis-angle ONB

\[ \vec{x}_\parallel = (\vec{a} \cdot \vec{x}) \vec{a} \]

\[ \vec{x}_\perp = (\vec{x} - \vec{x}_\parallel) = (\vec{x} - (\vec{a} \cdot \vec{x}) \vec{a}) \]

\[ \vec{a} \times \vec{x}_\perp = \vec{a} \times (\vec{x} - \vec{x}_\parallel) = \vec{a} \times (\vec{x} - (\vec{a} \cdot \vec{x}) \vec{a}) = \vec{a} \times \vec{x} \]
Axis-angle rotation

\[ x_{\text{rotated}} = \vec{x}_\parallel + \vec{v} \]

\[ x_{\text{rotated}} = \alpha \vec{a} + \beta \vec{x}_\perp + \gamma \vec{a} \times \vec{x} \]

\[ \vec{v} = \cos \theta \vec{x}_\perp + \sin \theta \vec{a} \times \vec{x} \]

\[ x_{\text{rotated}} = \vec{x}_\parallel + \cos \theta \vec{x}_\perp + \sin \theta \vec{a} \times \vec{x} \]

\[ x_{\text{rotated}} = (\vec{a} \cdot \vec{x})\vec{a} + \cos \theta (x - (\vec{a} \cdot \vec{x})\vec{a}) + \sin \theta \vec{a} \times \vec{x} \]

\[ x_{\text{rotated}} = (\vec{a} \cdot \vec{x})(1 - \cos \theta)\vec{a} + \cos \theta \vec{x} + \sin \theta \vec{a} \times \vec{x} \]
Rotation Matrix for Axis-Angle

\[ x_{rotated} = (\vec{a} \cdot \vec{x})(1 - \cos \theta)\vec{a} + \cos \theta \vec{x} + \sin \theta \vec{a} \times \vec{x} \]

\[ x_{rotated} = (\text{Sym}(\vec{a})(1 - \cos \theta) + I \cos \theta + \text{Skew}(\vec{a}) \sin \theta )\vec{x} \]

\[
\text{Sym}(\vec{a}) = \begin{bmatrix}
a_x \\
a_y \\
a_z \\
0
\end{bmatrix}
\begin{bmatrix}
a_x & a_y & a_z & 0 \\
a_x & a_y & a_z & 0 \\
a_x & a_y & a_z & 0 \\
0 & 0 & 0 & 0
\end{bmatrix}
\]

\[
\text{Skew}(\vec{a}) = \begin{bmatrix}
0 & -a_z & a_y & 0 \\
a_z & 0 & -a_x & 0 \\
-a_y & a_x & 0 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix}
\]

\[ \text{Skew}(\vec{a})\vec{x} = \vec{a} \times \vec{x} \]
Transforming normal vectors

- Transforming surface normals
  - differences of points (and therefore tangents) transform OK
  - normals do not. Instead, use inverse transpose matrix

\[
\begin{align*}
\text{have: } \mathbf{t} \cdot \mathbf{n} &= \mathbf{t}^T \mathbf{n} = 0 \\
\text{want: } M\mathbf{t} \cdot X\mathbf{n} &= \mathbf{t}^T M^T X \mathbf{n} = 0 \\
\text{so set } X &= (M^T)^{-1} \\
\text{then: } M\mathbf{t} \cdot X\mathbf{n} &= \mathbf{t}^T M^T (M^T)^{-1} \mathbf{n} = \mathbf{t}^T \mathbf{n} = 0
\end{align*}
\]
Perspective
Parallel projection

- To render an image of a 3D scene, we *project* it onto a plane.
- Simplest kind of projection is *parallel projection*.
Classical projections—parallel

- Emphasis on cube-like objects
  - traditional in mechanical and architectural drawing

Planar Geometric Projections

Parallel

- Orthographic
- Oblique

Perspective

- One-point
- Two-point
- Three-point

Multiview Orthographic

Axonometric

[after Carlbom & Paciorek 78]
Orthographic

Figure 2-1. Multiview orthographic projection: plan, elevations, and section of a building.
Orthographic

- projection plane parallel to a coordinate plane
- projection direction perpendicular to projection plane
Off-axis parallel

**axonometric**: projection plane perpendicular to projection direction but not parallel to coordinate planes

**oblique**: projection plane parallel to a coordinate plane but not perpendicular to projection direction
“Orthographic” projection

• In graphics usually we lump axonometric with orthographic
  – projection plane perpendicular to projection direction
  – image height determines size of objects in image
View volume: orthographic
Oblique projection

- View direction no longer coincides with projection plane normal (one more parameter)
  - objects at different distances still same size
  - objects are shifted in the image depending on their depth
Specifying views in a ray tracer

```xml
<camera type="ParallelCamera">
  <viewPoint>2.0 4.0 7.0</viewPoint>
  <viewDir>-2.0 -4.0 -7.0</viewDir>
  <viewUp>0.0 1.0 0.0</viewUp>
  <viewWidth>8.0</viewWidth>
  <viewHeight>4.5</viewHeight>
</camera>

<camera type="ParallelCamera">
  <viewPoint>2.0 4.0 7.0</viewPoint>
  <viewDir>-2.0 -4.0 -7.0</viewDir>
  <projNormal>0.0 0.0 1.0</projNormal>
  <viewUp>0.0 1.0 0.0</viewUp>
  <viewWidth>8.0</viewWidth>
  <viewHeight>4.5</viewHeight>
</camera>
```
History of projection

- Ancient times: Greeks wrote about laws of perspective
- Renaissance: perspective is adopted by artists

Duccio c. 1308
History of projection

• Later Renaissance: perspective formalized precisely

*da Vinci c. 1498*
Plane projection in drawing

The concept of the picture plane may be better understood by looking through a window or other transparent plane from a fixed viewpoint. Your lines of sight, the multitude of straight lines leading from your eye to the subject, will all intersect this plane. Therefore, if you were to reach out with a grease pencil and draw the image of the subject on this plane you would be “tracing out” the infinite number of points of intersection of sight rays and plane. The result would be that you would have “transferred” a real three-dimensional object to a two-dimensional plane.
Plane projection in photography

- This is another model for what we are doing
  - applies more directly in realistic rendering
Classical projections—perspective

- Emphasis on cube-like objects
  - traditional in mechanical and architectural drawing
Perspective projection (normal)

- Perspective is projection by lines through a point;
- “normal” = plane perpendicular to view direction
  - magnification determined by:
    • image height
    • object depth
    • image plane distance
  - f.o.v. $\alpha = 2 \text{ atan}(h/(2d))$
  - $y' = \frac{d y}{z}$
- “normal” case corresponds to common types of cameras
View volume: perspective
Shifted perspective projection

- Perspective but with projection plane not perpendicular to view direction
  - additional parameter: projection plane normal
  - exactly equivalent to cropping out an off-center rectangle from a larger “normal” perspective
  - corresponds to view camera in photography
Generating eye rays—perspective

- Use window analogy directly
- Ray origin (constant): viewpoint
- Ray direction (varying): toward pixel position on viewing window
Generating eye rays—perspective

• Positioning the view rectangle
  – establish three vectors to be camera basis: \( \mathbf{u}, \mathbf{v}, \mathbf{w} \)
  – view rectangle is parallel to \( \mathbf{u} - \mathbf{v} \) plane, at \( \mathbf{w} = -d \), specified by \( l, r, t, b \)

• Generating rays
  – for \( (u, v) \) in \([l, r] \times [b, t]\)
  – ray.origin = \( e \)
  – ray.direction = \(-d \mathbf{w} + u \mathbf{u} + v \mathbf{v}\)
Oblique perspective views

• Positioning the view rectangle
  – establish three vectors to be 
    camera basis: \( u, v, w \)
  – view rectangle is the same, but shifted so that the center is in the direction \( d \) from \( e \)

• Generating rays
  – for \((u, v)\) in \([l, r] \times [b, t]\)
  – \(\text{ray.origin} = e\)
  – \(\text{ray.direction} = d \cdot d + u \cdot u + v \cdot v\)
Field of view (or f.o.v.)

- The angle between the rays corresponding to opposite edges of a perspective image
  - simpler to compute for “normal” perspective
  - have to decide to measure vert., horiz., or diag.

- In cameras, determined by focal length
  - confusing because of many image sizes
  - for 35mm format (36mm by 24mm image)
    - 18mm = 67° v.f.o.v. — super-wide angle
    - 28mm = 46° v.f.o.v. — wide angle
    - 50mm = 27° v.f.o.v. — “normal”
    - 100mm = 14° v.f.o.v. — narrow angle (“telephoto”)
Field of view

• Determines “strength” of perspective effects

close viewpoint
wide angle
prominent foreshortening

far viewpoint
narrow angle
little foreshortening
Choice of field of view

- In photography, wide angle lenses are specialty tools
  - “hard to work with”
  - easy to create weird-looking perspective effects
- In graphics, you can type in whatever f.o.v. you want
  - and people often type in big numbers!
Perspective distortions

- Lengths, length ratios
Pipeline of transformations

- Standard sequence of transforms