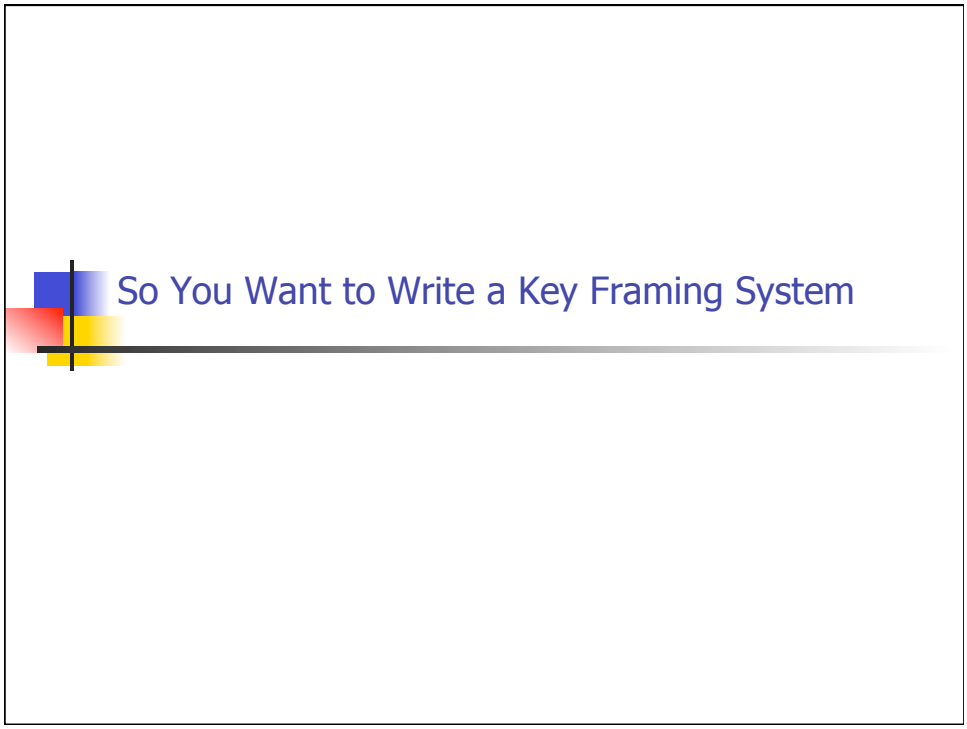
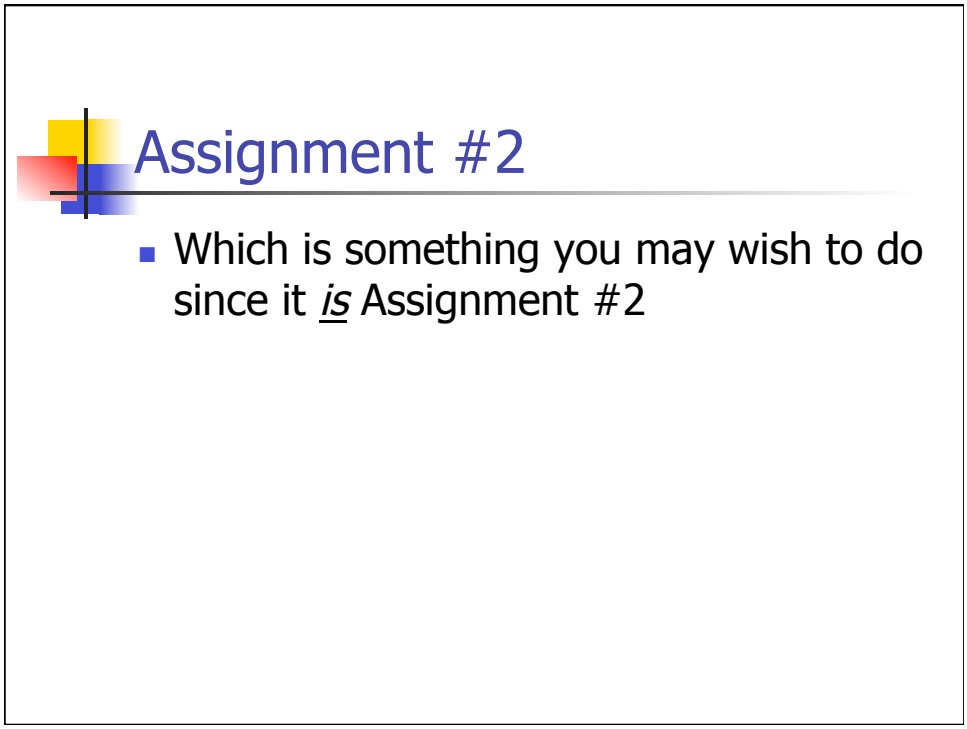

So You Want to Write a Key Framing System


Assignment #2
Which is something you may wish to do since it *is* Assignment #2

Assignment #2

- Write a simplified key framing system that will translate and rotate a single object based on a set of key frames.

Assignment #2

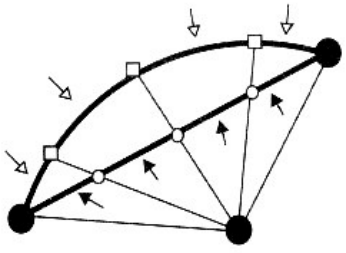
- Program may be:
  - Interactive – show the animation in window on the screen
  - Batch – create an app that will produce a set of input files for a renderer.
- In either case, program should be involved:
  - keyframe infile <outfile>
    - Outfile required only for batch applications.

Assignment #2

- Infile – Describes keyframes.
  - Very simple formatted text file
  - 1 line per keyframe definition
    - t, x, y, z, xa, ya, za, θ
    - Where
      - t is the time (sec) at which the object should be in the given position/orientation
      - (x,y,z) is the position
      - (xa, yz, za, θ) is the orientation in axis/angle format

Interpolating Quaternions

- Normalized Quaternions are defined on the unit 4D sphere
- Linear interpolation generates unequal spacing when projected onto circle



Interpolating Quaternions

- Spherical Linear Interpolation

$$\text{slerp}\,(q_1, q_{2,} u) = q_1 \frac{\sin((1-u)\theta)}{\sin\theta} + q_2 \frac{\sin(u\theta)}{\sin\theta}$$

  - where

$$\cos\theta = q_1 q_2 = w_1 w_2 + (v_1 \bullet v_2)$$

Interpolating Quaternions

- Note:
  - Spherical Linear Interpolation has the same problem as linear interpolation (no tangential continuity)
  - For smooth interpolation, Shomake recommends:
    - Bezier curves using DeCasteljau construction
    - Use slerp in constructing these point

Assignment #2

- Interpolation rules:
  - Translation may be interpolated using:
    - Basic linear interpolation
  - Rotation may be interpolated using:
    - Spherical linear interpolation of quaternions
      - Will need to convert axis/angle -> quaternions
  - Make camera track object
    - Place camera so object will not move off screen

Assignment #2

- Can assume that u varies linearly with t.
  - I.e. No slow-in/fast-out

Quaternions

- Quaternions essentially encodes the info of an axis/angle rotation

$R_{\alpha, (x,y,z)} = [\ \cos(\alpha/2), \sin(\alpha/2) \bullet (x, y, z)\ ]$

Remember to normalize

Interpolating Quaternions

- Spherical Linear Interpolation

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

  - where

$$\cos\theta = q_1 q_2 = w_1 w_2 + (v_1 \bullet v_2)$$

Remember to normalize

Interpolating Quaternions

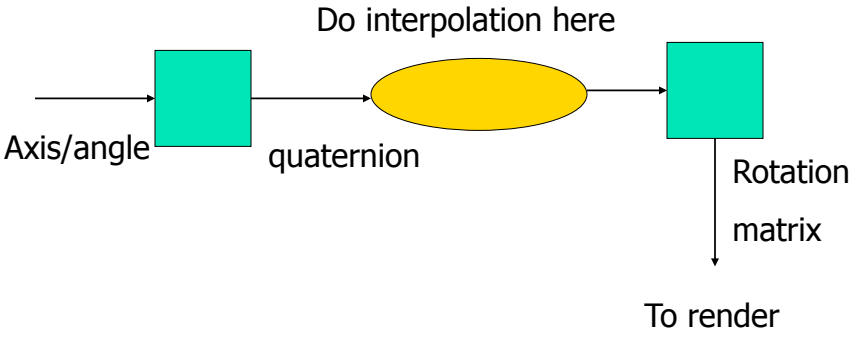
- Spherical Linear Interpolation (Slerp)
  - slerp (q_1,q_2, 1/2) can be efficiently computed (within a scale factor) as $q_1 + q_2$
  - Scale factor okay since you'll need to normalize anyway.

Quaternions

- Conversions: Quaternion -> Rotation Matrix
  - q = [w, x, y, z] , q normalized

$$\begin{bmatrix} 1-2y^2-2z^2 & 2xy-2wz & 2xz+2wy \\ 2xy+2wz & 1-2x^2-2z^2 & 2yz-2wx \\ 2xz-2wy & 2yz+2wx & 1-2x^2-2y^2 \end{bmatrix}$$

Rotation representation



Note on real time animation

- t in file represents actual time NOT number of display loops generated.
- For real time applications
  - Simulate a constant frame rate
  - Calculate time past since last "update" to determine t.

Questions?

Assignments

- Grading
  - Each assignment is worth 20 points:
    - 5 points – for something that compiles
    - 15 points – for something that runs incorrectly
    - 20 points – for something that runs correctly
    - Bonus points for extras…

Extras

- Explore some of the advanced options described in class. For 5 points:
  - Interpolate translation using Catmull-Rom curve
  - Allow for user defined t -> u mapping. (Slow in/fast out)
- For 10 points
  - Interpolate rotation using Bezier curve using De Casteljau construction
- Note
  - Organize your code well as you will be using these routines (at least translational interpolation) in future assignments!

Due dates

- Due
  - Monday, January 12th
- Submission
  - posted on mycourses (dropbox keyframe)
  - Please include documentation on
    - how to run your app
    - How to build your app
      - Makefile
      - Visual Studio (.dws, and .dsp files)
      - Mac (Xcode files)
    - Renderer used if batch
    - Platform (sun/Windows/Mac)

Questions?

- Next time:
  - Dynamics and Physics Simulation
  - Questions?
  - HAVE A GOOD BREAK!