Assignments

- project 1
  - out today, due 6pm Wed Feb 6
  - projects will go out before we’ve covered all the material
    - so you can think about it before diving in
  - build mouse out of cubes and 4x4 matrices
  - think cartoon, not beauty
  - template code gives you program shell, Makefile
  - written homework 1
    - out Monday, due 1pm sharp Wed Feb 6
    - theoretical side of material

Demo

- animal out of boxes and matrices

Real Mice

Think Cartoon

Armadillos!

Monkeys!

Monkeys!

Giraffes!

Giraffes!

Kangaroos!

Project 1 Advice

- do not model everything first and only then worry about animating
- interleave modelling, animation
  - for each body part: add it, then jumpcut animate, then smooth animate
  - discover if on wrong track sooner
  - dependencies: can’t get anim credit if no model
  - use body as scene graph root
  - check from all camera angles

- finish all required parts before going for extra credit
  - playing with lighting or viewing
  - ok to use glRotate, glTranslate, glScale
  - ok to use glutSolidCube, or build your own
  - where to put origin? your choice
    - center of object, range - .5 to +.5
    - corner of object, range 0 to 1

- visual debugging
  - color cube faces differently
  - colored lines sticking out of glutSolidCube faces
  - make your cubes wireframe to see inside
  - thinking about transformations
    - move physical objects around
    - play with demos
      - Brown scenegraph applets
  - smooth transition
    - change happens gradually over X frames
    - key click triggers animation
    - one way: redraw happens X times
    - linear interpolation:
      - each time, param += (new-old)/30
    - even better, but not required
  - where to put origin? your choice
  - center of object, range - .5 to +.5
  - corner of object, range 0 to 1

- project 1
  - out today, due 6pm Wed Feb 6
  - projects will go out before we’ve covered all the material
    - so you can think about it before diving in
  - build mouse out of cubes and 4x4 matrices
  - think cartoon, not beauty
  - template code gives you program shell, Makefile
  - written homework 1
    - out Monday, due 1pm sharp Wed Feb 6
    - theoretical side of material

Demo

- animal out of boxes and matrices

Real Mice

Think Cartoon

Armadillos!

Monkeys!

Monkeys!

Giraffes!

Giraffes!

Kangaroos!
Project 1 Advice
• transitions
  • safe to linearly interpolate parameters for glRotate/glTranslate/glScale
  • do not interpolate individual elements of 4x4 matrix!

Style
• you can lose up to 15% for poor style
  • most critical: reasonable structure
  • yes: parametrized functions
  • no: cut-and-paste with slight changes
  • reasonable names (variables, functions)
  • adequate commenting
  • rule of thumb: what if you had to fix a bug two years from now?
• global variables are indeed acceptable

Version Control
• bad idea: just keep changing same file
  • save off versions often
  • after got one thing to work, before you try starting something else
  • just before you do something drastic
• how?
  • not good: commenting out big blocks of code
  • a little better: save off file under new name
  • p1:painlessweak.spp, p1:fixedbug.spp
• much better: use version control software
  • strongly recommended

Graphical File Comparison
• installed on lab machines
  • xdiff4 (side by side comparison)
  • xwdiff (in-place, with crossouts)
• Windows: windiff
  • http://keithdevens.com/files/windiff
• Macs: FileMerge
  • in /Developer/Applications/Utilities

Readings for Jan 16-25
• FCG Chap 6 Transformation Matrices
  • except 6.1, 6.3.1
• FCG Sect 13.3 Scene Graphs
• RB Chap Viewing
  • Viewing and Modeling Transforms until Viewing Transformations
  • Examples of Composing Several Transformations through
    Building an Articulated Robot Arm
• RB Appendix Homogeneous Coordinates and Transformation
  • Matrices
  • until Perspective Projection
• RB Chap Display Lists

Review: Event-Driven Programming
• main loop not under your control
  • vs. procedural
• control flow through event callbacks
  • redraw the window now
  • key was pressed
  • mouse moved
• callback functions called from main loop when events occur
  • mouse/keyboard state setting vs. redrawing

Review: 2D Rotation
\[ (x', y') = \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \]

Review: Linear Transformations
• linear transformations are combinations of
  • shear
  • scale
  • rotate
  • reflect
• properties of linear transformations
  • satisfies T(ax+by) = a T(x) + b T(y)
  • origin maps to origin
  • lines map to lines
  • parallel lines remain parallel
  • ratios are preserved
  • closed under composition

3D Rotation About Z Axis
\[ \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \]

3D Rotation in X, Y
around x axis: \( \text{glRotatef}(\theta, 1, 0, 0); \)
around y axis: \( \text{glRotatef}(\theta, 0, 1, 0); \)

3D Scaling
\[ \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix} \]

3D Translation
\[ \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \]

3D Shear
\[ \begin{pmatrix} 1 & a & 0 \\ b & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \]
to avoid ambiguity, always say "shear along <axis> in direction of <axis>"
Composing Transformations

- scaling
  \[ \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \]
  so scales multiply

- rotation
  \[ \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \]
  so rotations add

Undoing Transformations: Inverses

- translation
  \[ T(x, y, z) = (x', y', z') \]
  \[ T^{-1}(x', y', z') = (x, y, z) \]

- rotation
  \[ R(\alpha, \beta, \gamma) \]
  \[ R^{-1}(\alpha, \beta, \gamma) \]
  \( R \) is orthogonal

- scale
  \[ S(a, b, c) \]
  \[ S^{-1}(a, b, c) \]
  \( S \) scales multiply

Summary: Transformations

- translate \((a, b, c)\)
- scale \((a, b, c)\)
- rotate \((\alpha, \beta, \gamma)\)
- rotate \((\alpha, \beta, \gamma)\)

Interpreting Transformations

- translate by \((-1, 0)\)
- moving object

Matrix Composition

- matrices are convenient, efficient way to represent series of transformations
  \[ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \]

Rotation About a Point: Moving Object

- rotate about \(p\) by \(\theta\)
- translate \(p\) to origin
- rotate about origin
- translate \(p\) back
Rotation: Changing Coordinate Systems

- Same example: rotation around arbitrary center
- Rotation around arbitrary center
  - Step 1: translate coordinate system to rotation center
  - Step 2: perform rotation
  - Step 3: back to original coordinate system

General Transform Composition

- Transformation of geometry into coordinate system where operation becomes simpler
  - Typically translate to origin
  - Perform operation
  - Transform geometry back to original coordinate system

Rotation About an Arbitrary Axis

- Axis defined by two points
- Translate point to the origin
- Rotate to align axis with z-axis (or x or y)
- Perform rotation
- Undo aligning rotations
- Undo translation

Arbitrary Rotation

- Arbitrary rotation: change of basis
  - Given two orthonormal coordinate systems XYZ and ABC
  - A's location in the XYZ coordinate system is \((a_x, a_y, a_z, 1)\), ...
  - Transformation from one to the other is matrix \(R\) whose columns are \(A, B, C\):
    \[
    R(X) =
    \begin{bmatrix}
    a_x & b_x & c_x & 0 \\
    a_y & b_y & c_y & 0 \\
    a_z & b_z & c_z & 0 \\
    0 & 0 & 0 & 1
    \end{bmatrix}
    \]