Modeler Help Session
CSE 457, Autumn 2008

Modeler Due: Tuesday, November 3rd
Modeler Artifact Due: Friday, October 30th

Project TA: Travis Veralrud

Help Session Overview

- The Modeler Application
- Modeler Code Overview
- Constructing Your Model
- Hierarchical Modeling in OpenGL
- Warnings & Hints
- Example Models
- Summary

The Modeler Application

- Two main windows:
  - Control window with sliders
  - Model view window

- To navigate in the model view window:
  - **Left-click**: Rotate the camera
  - **Middle-click**: Move the point that the camera is looking at (Translation/Dolly)
  - **Right-click**: Zoom in/out

Modeler Code Overview

- modelerapp.* and modelerui.* handles the user interface
- modelerapp.h defines the ModelerApplication class which handles the setting of control values
- modelerdraw.* supports the drawing of primitive shapes and material attributes

Modeler Code Overview

- modelerview.h
  - defines the ModelerView object
  - base class of your model – your model will be a subclass of ModelerView
  - handles OpenGL drawing and event handling
- modelerview.cpp
  - provides some base functionality such as setting up lighting and handling camera controls

Modeler Code Overview

- DO NOT TOUCH THE FOLLOWING FILES:
  - modelerapp.*
  - modelerui.*
  - modelerdraw.*
  - modelerview.*
- For the animator project, you will be re-using your model source file and plugging it into a different application. If you change modelerapp.* or modelerdraw.*, your model may not work with the animator project!

Modeler Code Overview

What DO you get to change?

- Camera.*
  - Controls camera functions
  - Look in camera.cpp to implement your own version of gluLookAt()
- Sample.cpp
  - Example BoxModel - you will replace this with your own model file
  - To start: copy sample code and then modify in order to include the methods you need.
  - Eventually remove sample.cpp file and replace with <YourModel>.cpp

Modeler Code Overview

Some helpful files (that you should also not change)

- Modelerui.fl is a data file that controls the FLTK user interface
- Vec.h & Mat.h contains useful vector/matrix operations

Constructing Your Model

- Make all changes in Sample.cpp
  - Draw() function is where you will build your model
  - Main() function initializes the controls
    - Add slider controls
  - Enum statement at the top of the file defines the names and number of controls
    - Add controls both to Enum and main
    - Remember to keep NUMCONTROLS variable at the end of the Enum list

Hierarchical Modeling in OpenGL

- OpenGL is a state machine
  - glEnable()/glDisable() changes the state
  - Once you change something, it will stay that way until you change it to something new!
  - This includes: current color, transformation details, drawing modes, information about the lights, etc.
- OpenGL maintains a transformation matrix that is applied to everything that is drawn
  - In other words: transformations are cumulative
  - Perform transformations glRotated(), glTranslated(), glScaled() relative to the previous drawn object

Hierarchical Modeling in OpenGL

How do we get back to an earlier transformation matrix?

- glPushMatrix() & glPopMatrix()
  - Keeps track of the state of your model in a stack
  - If you want to make changes and then undo the transformation matrix, glPushMatrix() will keep track of where everything was
  - When popped off the stack, will return to those values

Warnings & Hints

- Keep track of your pushes() and pops() – having unmatched pushes and pops can cause a lot of grief!
  - It can help to divide the draw routine into a series of nested methods, each with their own push and pop.
- Implementing gluLookAt(): Look in your slides and in the OpenGL Blue Book, but make sure you understand how it works!
- Implementing the animation sequence: have a slider control multiple aspects of the model

Warnings & Hints

Worthwhile bells & whistles that will help you out down the road:

- Cool lighting/camera effects
- Smooth curve functionality/swept surfaces

Warnings & Hints

Texturing mapping FAQ:

- Look in the OpenGL Programming Guide to see how to set up texture mapping
- Use the load function in imageio.cpp to load a JPEG or PNG to use as a texture map

Example Models

Looking for inspiration?

- Example models can be found on previous quarters' websites
- A short list of sample executables available at: http://www.cs.washington.edu/education/courses/cse457/Cu
- A quarter of very impressive models: http://www.cs.washington.edu/education/courses/cse457/02
- More links on the project page

Avoiding SVN conflicts

- In general, never put automatically generated binaries into source control
  - modeler.suo, modeler.ncb, Debug*, Release*
  - Avoid *.user files too
- These binaries will cause a conflict at practically every commit when both people are working on the project
  - http://svnbook.red-bean.com/
  - TortiseSVN: http://tortoisesvn.tigris.org/

Save your files!

- **DO** put source files (*.cpp, *.h, *.sln, *.vcproj), image files, etc. in the repository
  - If you create any new files remember to both add AND commit the files to avoid loss
- Work from the ThawSpace drive (typically “Z:”) on the lab computers

Test the Source Control Early

- The only way we can fix problems is if we know about them
- So, verify that your repository works by checking it out, building it, tweak something, and commit
  - If something fails, please let us know so we can fix it

Summary

Things To Do:	Bad Things Will Happen if you modify:
▪ Replace the glulookat() function in camera.cpp ▪ Create a model (like sample.cpp) with at least 4 hierarchical levels and 10 primitive shapes ▪ Animation Slider ▪ An Additional Bell	▪ modelerapp.* ▪ modelerui.* ▪ modelerdraw.* ▪ modelerview.* ▪ vec.h ▪ mat.h