

MODELER HELP SESSION

Assigned: Friday, April 16th

Due: Wednesday, April 28th *at the stroke of midnight!*

TA: Jeff Booth

Help Session Overview



- Checking Out and Building Your Code
- Understanding The Modeler Application
- Constructing Your Model
- Warnings and Hints
- Examples

Checking Out Your Code

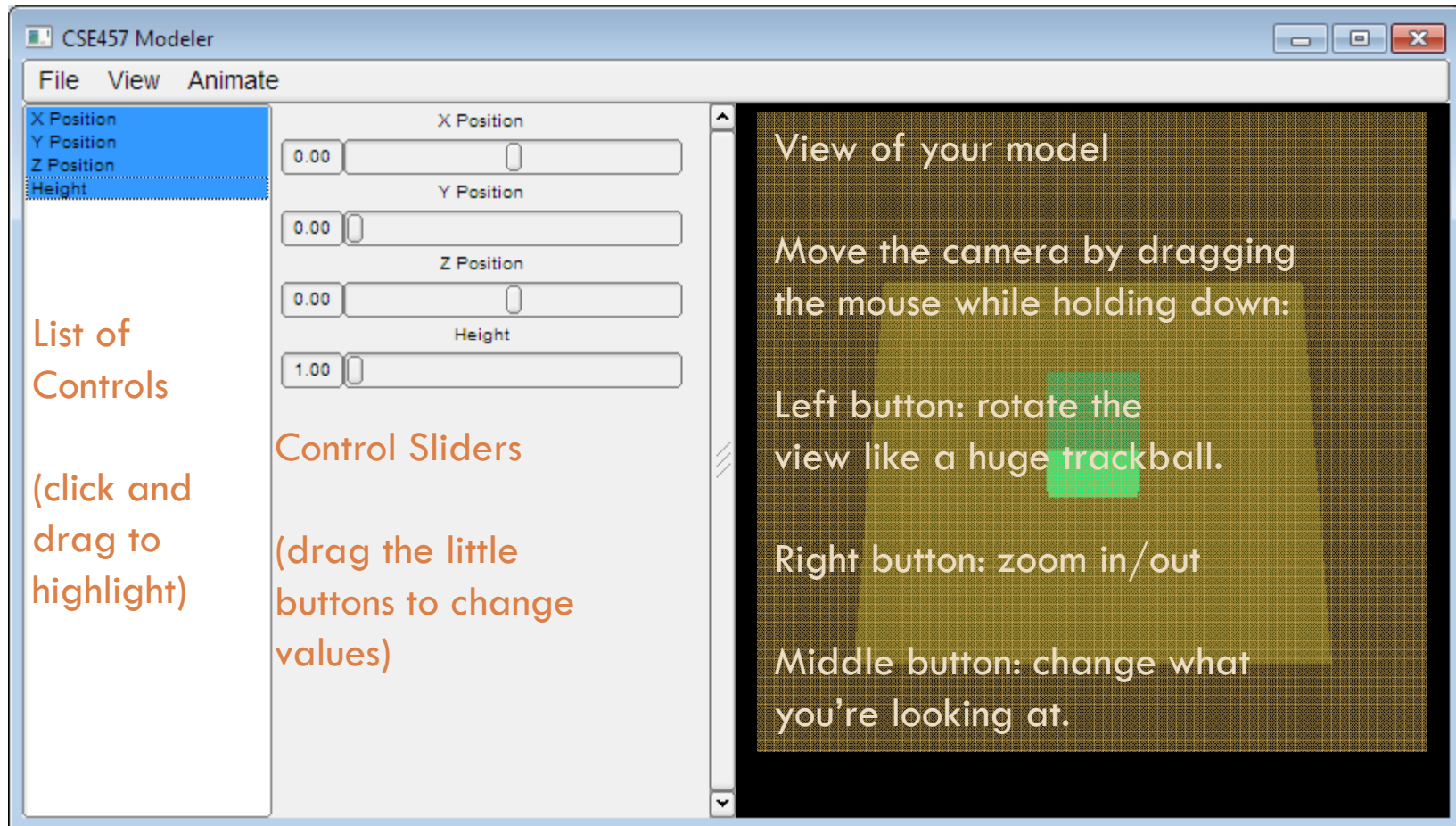
- Go to the Modeler course page for detailed check-out directions.
- Repository path:
 - `svn+ssh://Your CSE NetID@attu.cs.washington.edu/projects/instr/10sp/cse457/modeler/Your Group ID/source`
- Check out to
 - `C:\Users\Your CSE NetID\modeler`

Building in Visual Studio



- Go to your project folder
- Double-click the .vcproj file
- Configuration menu next to green arrow
 - ▣ Debug – lets you set breakpoints
 - ▣ Release – for turn-in
- Pick **Debug**, then click the green arrow next to it to build and run your project
- Let us know if it doesn't build!

One Window to Rule Them All



The screenshot shows the CSE457 Modeler application window. The title bar reads "CSE457 Modeler" and the menu bar includes "File", "View", and "Animate". On the left, a "List of Controls" panel contains a scrollable list with "X Position", "Y Position", "Z Position", and "Height" selected. To the right of this list are "Control Sliders" for each parameter, with "X Position", "Y Position", and "Z Position" set to 0.00 and "Height" set to 1.00. The main 3D view area on the right contains a yellow translucent box with the following text:

View of your model

Move the camera by dragging the mouse while holding down:

- Left button: rotate the view like a huge trackball.
- Right button: zoom in/out
- Middle button: change what you're looking at.

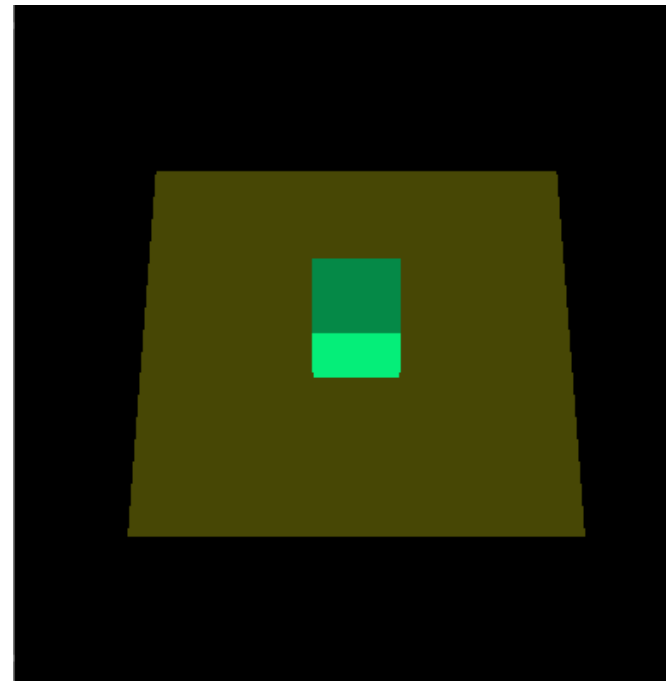
Modeler Code Overview



- `modelerapp.*` and `modelerui.*` handle user interface
- `modelerdraw.*` has functions for drawing primitive shapes and setting material attributes
- `camera.*` contains the Camera class

The Modeler View

- **Modelerview.h**
 - ▣ Base class for your model
 - ▣ Handles OpenGL drawing and mouse events
- **Modelerview.cpp**
 - ▣ Sets up lighting and a basic camera



What Should You Change?



- Camera.cpp
 - ▣ Replace the call to `gluLookat()` with your own transformations.
- Sample.cpp
 - ▣ Construct your model by changing `draw()`, `main()`, and the enum at the top

Constructing Your Model

- Make all changes in `sample.cpp`
- `BoxModel::draw()` – build model here
- `main()` – Add slider controls
- `Enum statement` – Add control labels too
 - ▣ Each control label gets replaced with a number when the code is compiled
 - First label = 0, second label = 1, ...
 - ▣ Keep `NUMCONTROLS` at the end of the Enum list

OpenGL Is A State Machine



- `glEnable()/glDisable()` changes state
- Once you change something, it stays that way until you change it to something new
- OpenGL's state includes:
 - ▣ Current color
 - ▣ Transformation matrices
 - ▣ Drawing modes
 - ▣ Light sources

OpenGL's Transformation Matrix

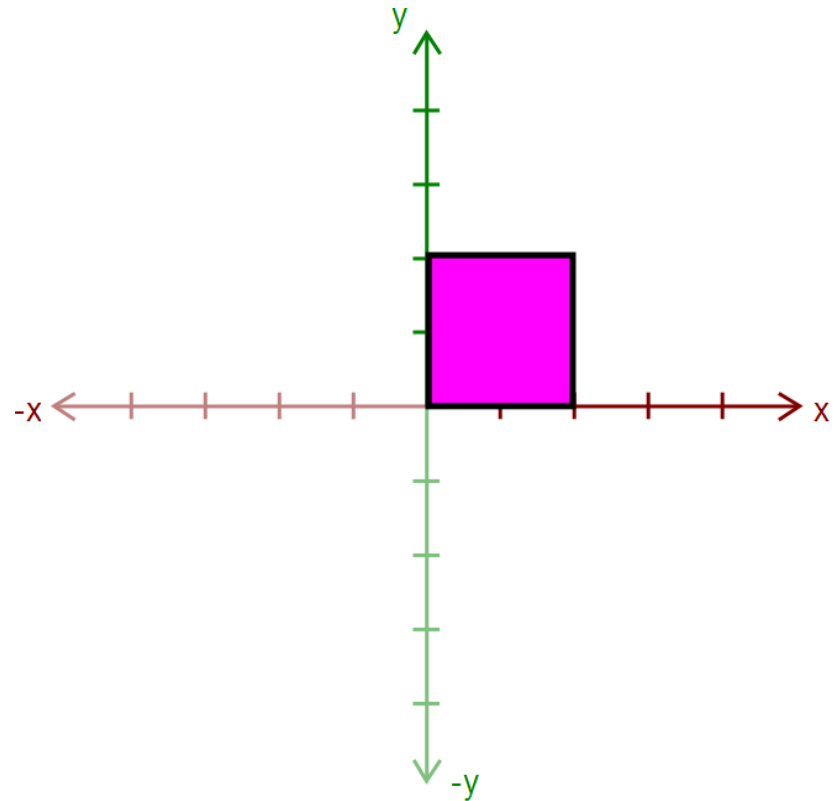
- Just two of them: **projection** and **modelview**. We'll modify **modelview**.
- Matrix applied to all vertices and normals
- These functions multiply transformations: **glRotated()**, **glTranslated()**, **glScaled()**
- Applies transformations in REVERSE order from the order in which they are called.
- Transformations are **cumulative**. Since they're all “squashed” into one matrix, you can't “undo” a transformation.

Transformations: Going “Back”

- How do we get back to an earlier transformation matrix?
- OpenGL maintains a **stack** of matrices.
- To store the current matrix, call **glPushMatrix()**.
- To restore the last matrix you stored, call **glPopMatrix()**.

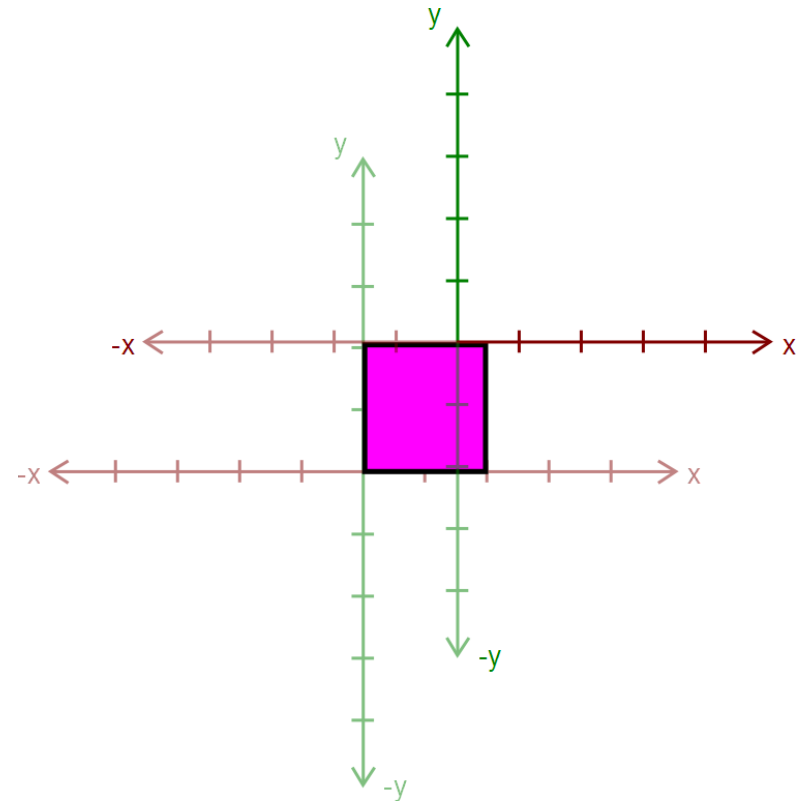
Hierarchical Modeling in OpenGL

- Draw the body
- Use `glPushMatrix()` to remember the body's "axes"



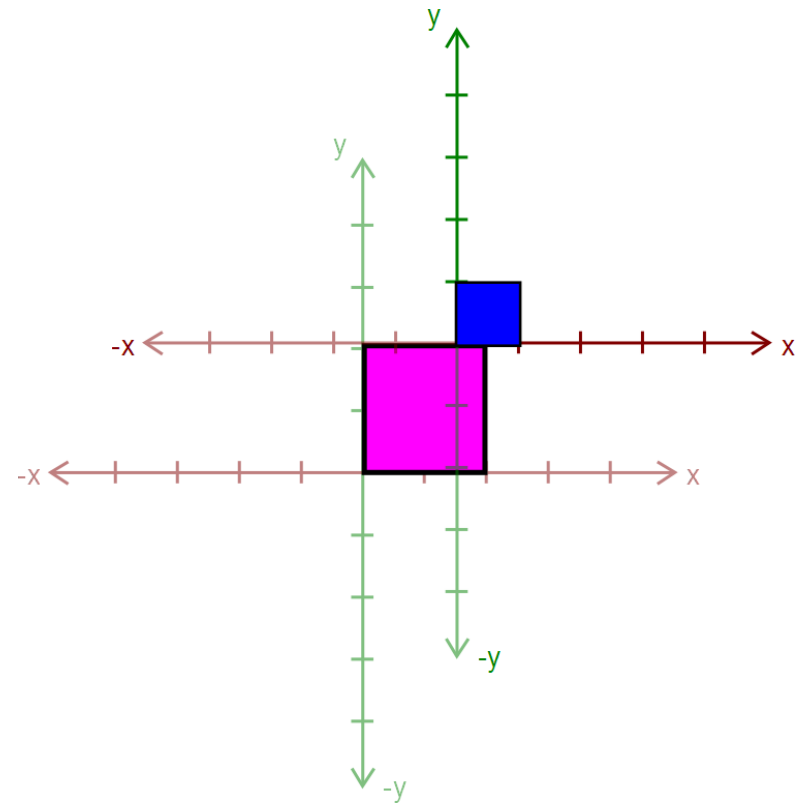
Hierarchical Modeling in OpenGL

- Apply a transform:
 - `glRotated()`
 - `glTranslated()`
 - `glScaled()`



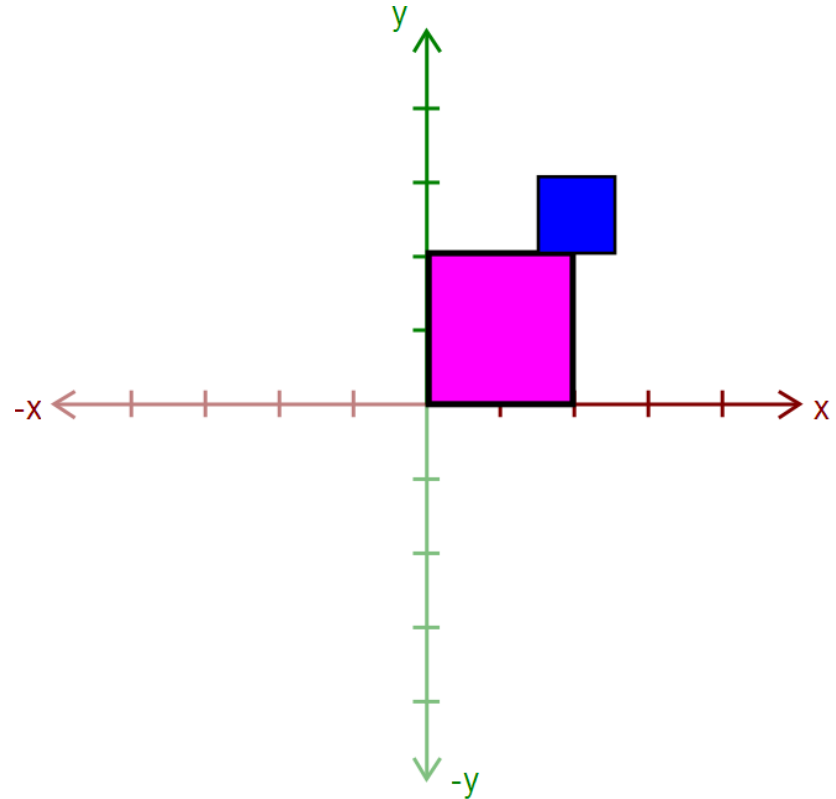
Hierarchical Modeling in OpenGL

- Draw an ear



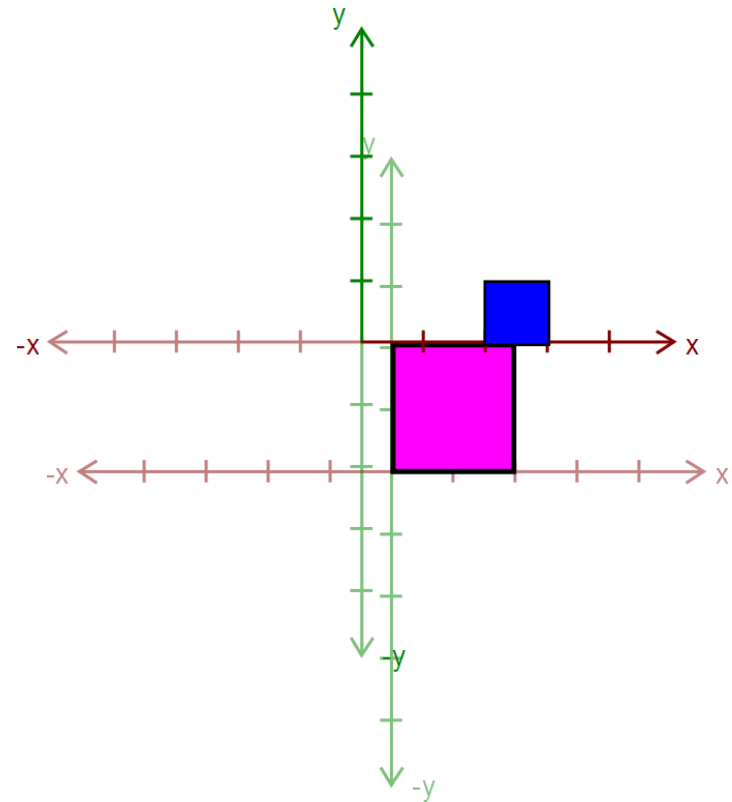
Hierarchical Modeling in OpenGL

- Call `glPopMatrix()` to return to the body's "axes"
- To draw the other ear, call `glPushMatrix()` again...



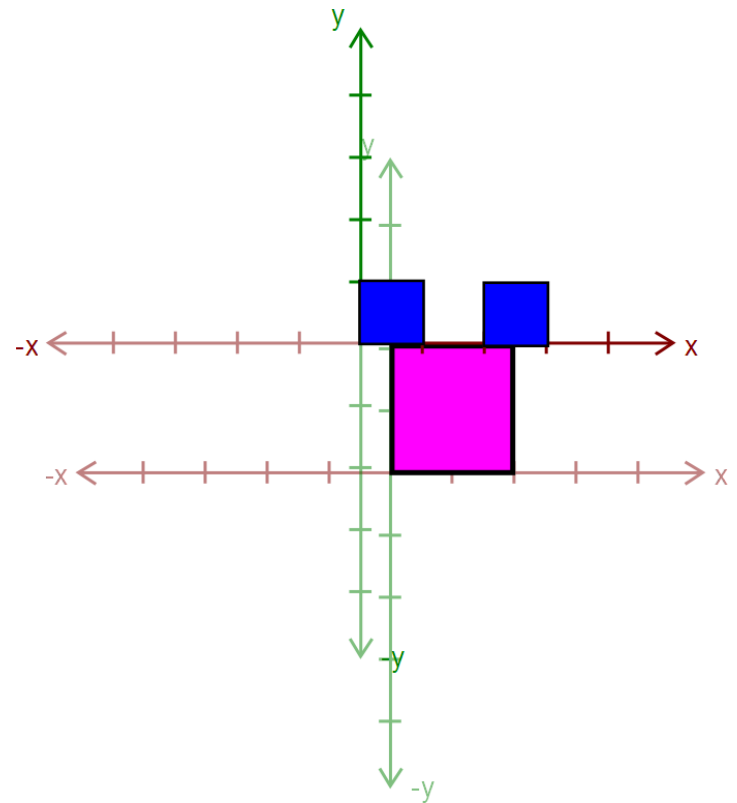
Hierarchical Modeling in OpenGL

- Apply another transform...



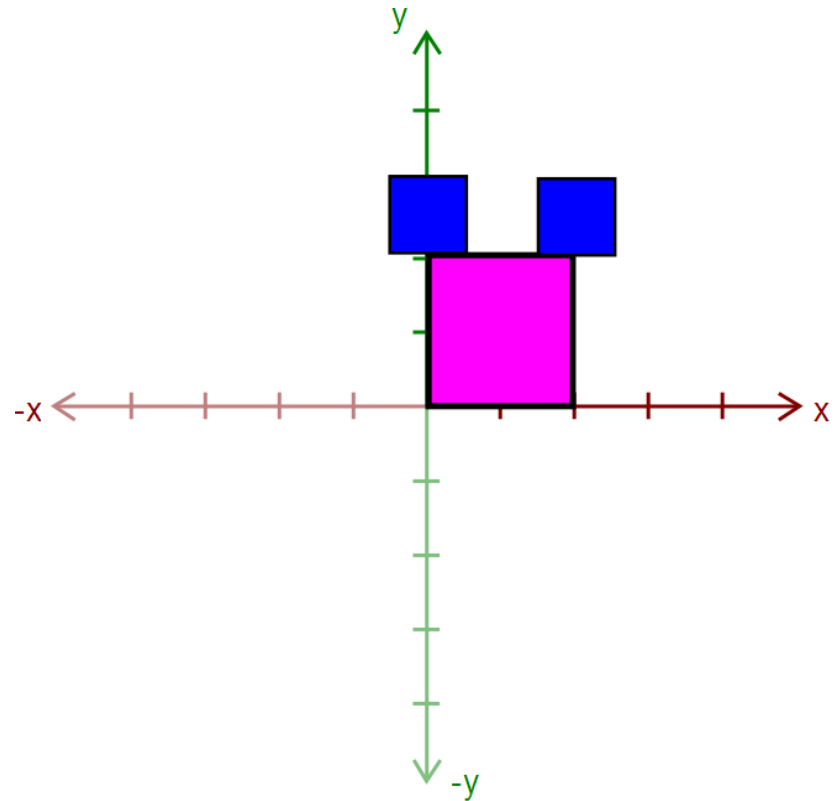
Hierarchical Modeling in OpenGL

- Draw the other ear



Hierarchical Modeling in OpenGL

- Then, call `glPopMatrix()` to return to the body's "axes"
 - Technically, you don't need to if that second ear is the last thing you draw.
 - But what if you wanted to add something else?



Warnings and Hints

- Make sure there's a `glPushMatrix()` for every `glPopMatrix()`!
 - ▣ You can divide your `draw()` function into a series of nested methods, each with a push at the beginning and a pop at the end.
- See lecture slides on April 14th for `gluLookAt()`
 - ▣ Make sure you *understand* how works
 - ▣ Lots of “magic code” on the Internet
 - ▣ You might be asked about it during grading

Animation Slider



- Needs to control **multiple aspects** of your model.
 - Example: Rotate multiple joints at once
- Don't get too complicated!
 - Wait for Animator in four weeks!

Interesting Extra Credit Options

- Texture Mapping
 - ▣ Look in the OpenGL Programming Guide for details
 - ▣ Use the load function in imageio.cpp to load a JPEG or PNG to use as a texture map
 - ▣ **WARNING:** There is a bug in imageio.cpp; patch will be released soon...
- Shaders
 - ▣ More complex lighting effects
 - ▣ Use the OpenGL Extension Wrangler library
- Cool lighting / camera effects
- Smooth curves and swept surfaces

Helpful Files

- `vec.h` contains useful Vector classes
- `mat.h` contains useful Matrix classes
- `modelerdraw.*` contains methods for drawing primitives:
 - Sphere
 - Box
 - Cylinder
 - Triangle
- We recommend not changing these files unless you know what you're doing

What SHOULDN'T You Change?



- Unless you're doing extra credit, don't change:
 - `modelerapp.*`
 - `modelerui.*`
 - `modelerdraw.*`
 - `modelerview.*`
- If you change `modelerapp.*` or `modelerdraw.*`, you might not be able to use your model in the Animator project.

Preparing Your Work Environment



- Make sure that your repository works by:
 - ▣ Checking it out
 - ▣ Building it
 - ▣ Tweaking something
 - ▣ Committing
- Do this on each work environment you plan to use, even if you aren't going to start work yet:
 - ▣ Lab machines
 - ▣ Your home computer
 - ▣ The sooner we know of a problem, the sooner we can fix it.

Avoiding SVN Conflicts

- In general, **never** put anything besides source code into source control:
 - Debug and Release folders
 - Modeler.suo
 - Modeler.ncb
 - *.user files
- DO put **source files** (*.cpp, *.h, *.vcproj, image files, etc.) in the repository
 - Make sure you both **add** AND **commit** the files.
 - TortoiseSVN: when you commit, make sure all the files you added have a checkmark.

Quick Summary

Things To Do

- Replace the `gllookat()` function in `camera.cpp`
- Create a model (like `sample.cpp`) with at least
 - ▣ 4 hierarchical levels
 - ▣ 10 primitive shapes
- Animation Slider
- An Additional Bell

Warnings

- Don't modify:
 - ▣ `modelerapp.*`
 - ▣ `modelerui.*`
 - ▣ `modelerdraw.*`
 - ▣ `modelerview.*`
 - ▣ `vec.h`
 - ▣ `mat.h`
- Make sure you can check out, commit, and build!

Before You Leave



- Try adjusting the sample model
 - ▣ Let us know if you have problems
- **COMMIT BEFORE LOGOFF!**
 - ▣ Your files in C:\User\... will **go away** when you log out, due to Deep Freeze!