CIS 441/541: Project #2B
Due May 17, 2021 (which means May 18, 2021 at 6am)
Worth 7% of your grade

Goal:
You will make a dog out of spheres and cylinders. This project will help you learn more about the ModelView matrix and also better understand how geometries are constructed.

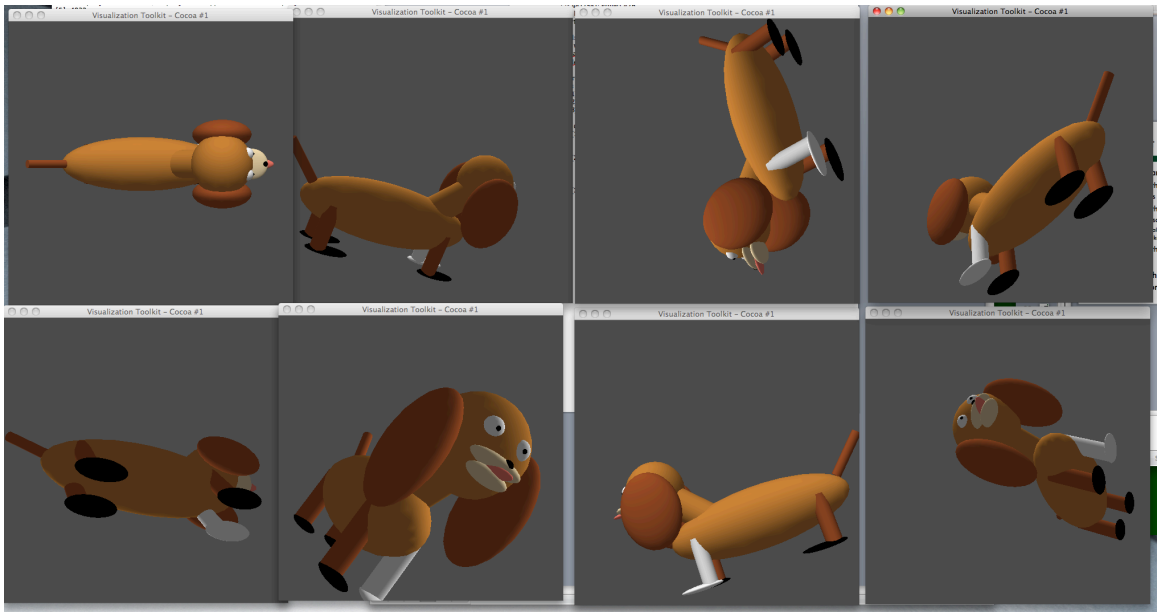
There is a new skeleton program, project2B.cxx.
This program contains 4 parts:

1) code to set up spheres and cylinders
2) a "RenderManager" module
3) main function
4) the function you modify

It is intended that you will only need to modify Part 4. That said, you will need functions in Part 2 and should review those functions. You are encouraged to look through the entire code base.

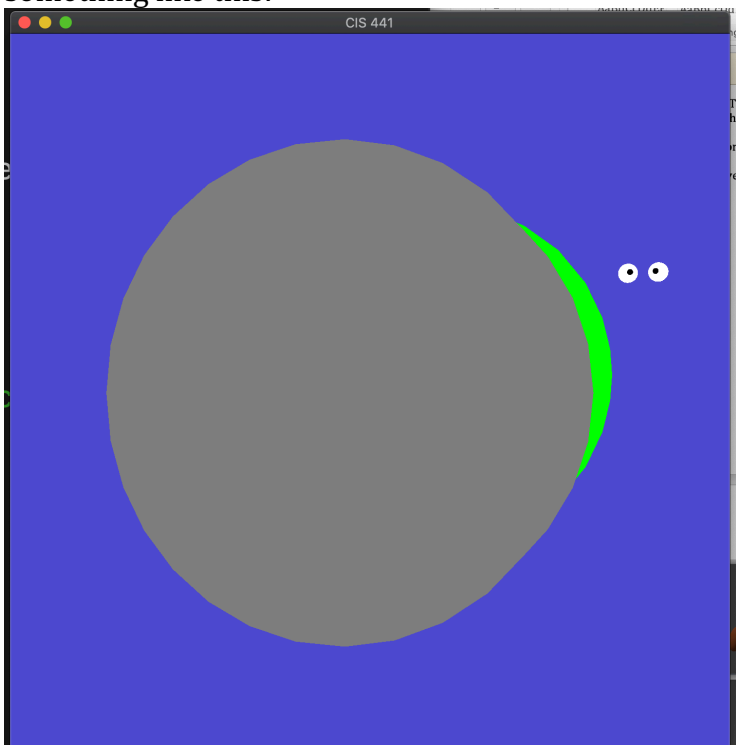
You do not need to produce exactly my dog. Your dog should:

(1) look more or less like a dog (i.e., as much as mine). The means no obvious problems with the geometry (example: the legs are super long and you left it because you didn't know how to fix it).
(2) Use the sphere and cylinder routines in project2B.cxx.
    a. If you want to use different geometries, let me know. The concern here would be if people bring in external geometries that simplify the problem too much.
(3) Have two elements that are not aligned with (1,0,0), (0,1,0), or (0,0,1). My dog has the tail and neck at an angle.
(4) Your dog should animate in some way. I expect most will have a leg move or a tail wag.

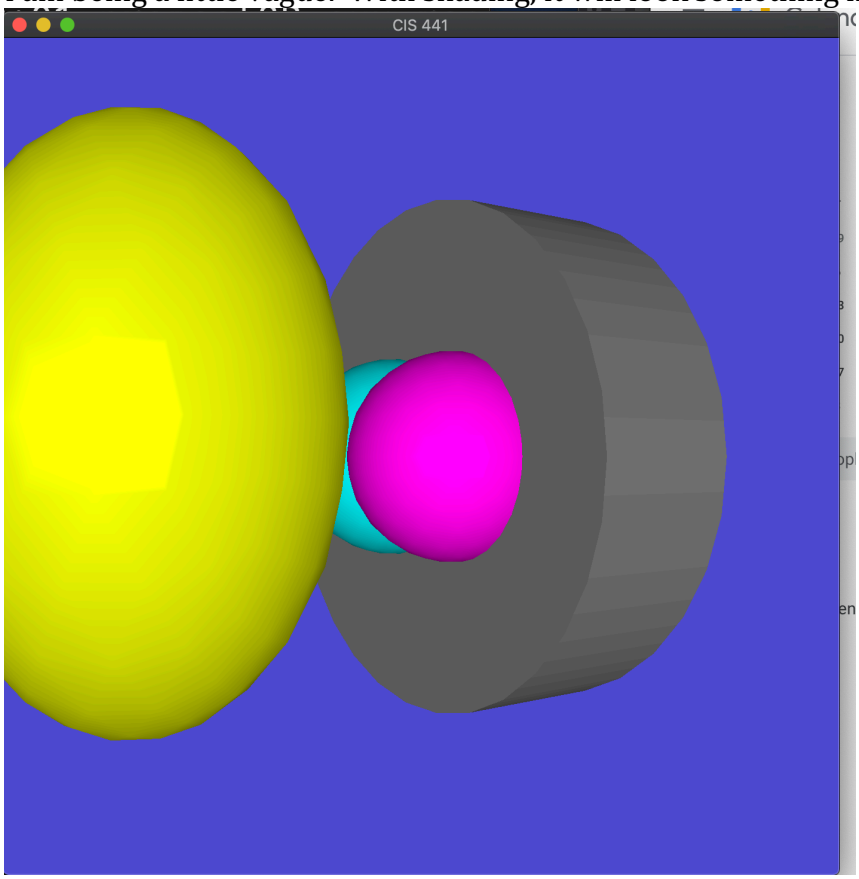


Steps: you should do these in order and not proceed to the next step until the current step is completed.

1) Compile the starter code and run it. It makes a few circles, a fat cylinder, and two eyeballs. Ultimately, you will want to discard the circles and cylinder. You are welcome to use the eyeballs I wrote for you. It should look something like this:



2) Add Phong shading to the shader code. This should be mostly a copy-and-paste from 2A. That said, you will need to do a little digging in the project2B.cxx starter code to see what goes where. I believe the experience of digging through the starter code and extending is invaluable, which is why I am being a little vague. With shading, it will look something like this:



3) Figure out what colors you want to use. This will require some googling to see what R/G/B values correspond to the colors you want to use. Also, note that these colors are often listed as 0-255, and you will have to normalize them to 0-1. Also: no Dalmatians / black+white only / crazy alien dogs that happen to be pure red / green / blue. The colors should be plausible.
4) Make your dog. This will involve modifying the method "SetUpDog." The view is currently circling around the XZ-plane. Therefore the dog's vertical orientation is along the Y-axis. Saying it another, way the dog's feet may be at Y=-2 and its head may be at Y=2. Of course, there is a lot of flexibility -- the feet may not all be on the "ground," etc. See tips below for more info.
5) Make some part of your dog animate. This will involve using the "counter" variable in the main loop. You can see I use counter to animate colors. You should not animate colors – you need to animate some part of the dog to move around (tail wag, leg move, etc).,

Grading rubrics:

- Most projects will receive 6/7. The dog I posted above would receive 6/7.

- Some folks will build beautiful dogs and receive 7/7, and some will receive scores between 6 and 7. Being explicit, fancy/involved models are needed to get the extra point.
- If you don't pick good colors (meaning black & white dog), then 2 points off. The colors in my example are not that exciting, but they are fine and would not receive a deduction.
- If you have no animation, then 2 points off.
- If you have obvious problems with the geometry, then 2 points off. (Examples: the legs are super long and you left it because you didn't know how to fix it. Or your neck doesn't connect to the torso.)

What to turn in?: Just a single file, which is your project2B.cxx code.

Tips on making the dog:

I think the best approach is a mashup of some planning, and some trial and error. The dog will probably live in the box X=-3 -> X=3, Y=-3 -> Y=3, Z=-3 -> Z =3. Make decisions about where the torso, head, and legs will be. Add the components one at a time. Compile often and run often.

You have two geometric types to play with – sphere and cylinder. With scaling, you can make either of these be pretty useful.

I provide three functions for transforming: TranslateMatrix, RotateMatrix, and ScaleMatrix. They are useful.

My example code, SetUpHead and SetUpEyeball, has a pattern that sets up a frame of reference and does operations relative to the frame. This is a powerful paradigm and I encourage you to emulate it.