Week #1 Live Lecture Chat Window
October 4, 2023

13:58:06 Is there a glVertex that takes a glm vector or an array?

Yes, it is glVertex3fv(xyzarray); The ‘v’ stands for “vector”. A lot of OpenGL functions work this way, e.g., glColor3fv(rgbarray);

14:03:41 Anyone know where the information on how to turn in projects is? I can’t find it and I remember seeing/hearing something about a video being required?

Each project handout has that information.

14:04:55 What should the video contain?

Look at the points rubric at the end of each project handout and show us those things in your project so we can give you those points. Use your video to help us accurately grade your project!

14:05:06 Is it the code itself we explain how it works or?

Don’t walk us through the code in your video. If we want to see your code, we will look at the .cpp file you also uploaded.

14:05:22 What if we have multiple cpp/ header files. Just upload all of them?

Just upload your main .cpp program (usually “sample.cpp”). If I want to see more than that, I’ll just ask you.

14:05:50 So then it’s basically a short video just showing it being run?

Exactly.

14:06:59 Is there a [Live Lecture] recording for last week?

Yes. All the LL recordings end up in a table on the Class Resources Page.

14:07:29 I actually need some help on how to add a shape to a specific face of an existing object

[We discussed this during OHs which, unfortunately, was not recorded. But, basically re-use some of the x-y-z vertex coordinates that you specified when drawing an existing face and add on from there.]

14:20:40 What are quizzes about and is getting started included?

Let’s face it – these quizzes are a brazen attempt to get you to review the material in order to move it from short-term to long-term memory. Thus, *any* previous material is eligible to be on the quizzes. But usually, the emphasis is on what we covered that week or the week before. Remember that the quizzes are open notes. So, know the general concepts and what topic details can be found in what notesets.

14:21:04 Are there coding questions [on the quizzes]?

The quizzes (except for Quiz #0) are all multiple choice, so you won’t have to code anything. I *might* give you a line of code and ask what is wrong with it. Or something like that.

15:14:38 Sorry it's a bit off the color topic, but I'm having a little trouble figuring out the practical difference between a thick line and a cylinder/tube. A line is 2d, but a cylinder would be 3d. But... they kinda look the same to me. Is one better/worse to use, like for project 1?

In Project #1, there probably is no practical difference. In later projects, however, cylinders are nice because they have surface normal vectors that can be used for lighting and have texture coordinates that can be used for texturing.

15:15:44 I have a tangentially related question, how can I tell my normals are right with flat shading?

Create a scene with a light source (e.g., Project #3) and see if the surfaces respond correctly.

15:19:04 Does shading/lighting often mess up your color scheme?

I wouldn’t call it “mess up”. I would say that lighting “modulates” your colors. But we want it to. It is that variation in light-version-of-the-color to dark-version-of-the-color that gives us the better sense of 3D-ness.

15:23:56 Is Quiz #1 going to be on everything in week 1 notes?

It is only 10 questions so it can’t be on *everything*. It could ask things from both Week #0 and Week #1. But, again, remember it is open notes, so you don’t need to memorize absolutely everything. Understand it well enough to look it up if you have to.

15:28:32 How much of a deal is slight imprecision in applications like .obj files? If I want two triangles to share an edge, should the two shared points be exactly the same, or is approximately the same good enough?

Typically, you want a vertex on one face to have exactly the same x-y-z numbers as the corresponding vertex on the next face over. A little off is OK, but you don’t want to see any cracks in your model if you zoom in. OBJ files typically are OK about this because they list all the vertex coordinates first and then let the faces specify integer indices telling what vertex number to use. This makes it much more reliable for two adjacent faces to use the same vertex coordinates where they touch.