

## Live Lecture Chat Window

January 10, 2024

**12:23:10 Does the number of times you draw the shader per second = frames per second? Or is the fps sometimes faster?**

It's not so much drawing the shader as drawing the scene. It's really "scenes drawn/second".

**12:33:16 Why is the Light provided as a negative value to the reflect function in line 50?**

The convention in the lighting diagram is vector going from point to light. The convention in the GLSL reflect function is input vector comes into the point, not away from it.

**12:41:21 Just curious is the [Keytime] interpolation linear or smooth?**

Smooth. It uses a cubic curve to maintain first derivative continuity throughout the animation.

**12:49:05 I'm confused on the difference between putting this into the GLman GUI vs just doing run build etc.**

Glman basically has written the C-code for you – you just supply the shaders. In "run build", it's your C code.

**12:50:11 So for the GLman approach we are only modifying frag and vert?**

Yes, and creating the .glib file to describe the overall scene.

**12:50:23 So in this class we are more concerned with the shaders and not the C code?**

Yes. I am trying to keep you from diluting your shader expertise by worrying about too many other things.

**12:52:59 So, for the assignment do we need to download the glm folder and work on the shaders from there?**

You want to download one of the ShadersSample\* folders and work from there. You really don't need GLM in this class.

**13:08:25 Are we okay to use our ellipses equation from the Shaders assignment in 450?**

Yes, but because our P1 is to do a whole grid of ellipses, I don't think it will help you as much as you are hoping.

**13:14:07 Is glman mandatory?**

Definitely not. Do the projects whatever way serves you best.

**13:14:48 What requirements are there for citing the provided skeleton code?**

Any code I've given you (notesets, project handouts, etc.) can be used without citing it. Also you don't need to turn in any of those auxiliary functions. Just use them.

**13:15:06 So in our demo video, we're only adjusting the sliders that we're being graded on? uAd, uBd, uTol, etc**

That's correct. You are free to embellish all you want though.

**13:15:36 What do I need to download if I want to use the glman approach?**

Download glman.exe from the Downloadable Executables part of our Class Resources Page, along with at least the first 2 of the .dll files listed there.

**13:15:55 Assuming that if you use glman, you're turning in the glib, vert, and frag, and not any C++ because there isn't any? (and vice versa not a glib if you use C++)**

That's correct.

**13:22:35 This is more of a general question, currently our shader is code that conditionally changes the color of fragments based on their location. What else can they do? Are we gonna use them to create shadows later?**

We will talk about shadows, corrosion effects, image processing, non-linear transformations, morphing, bump-mapping, cube-mapping, physics effects, lens effects, wave motion, and so on. We spend about a week figuring out what it takes to do shader programming, followed by 9 weeks of tricks. Your friends will think you're a wizard. 😊

**13:24:25 Does the glib file contain information on the "origin" of the scene where all the vector values used are based on? I'm curious on what all the positions are based on, where is [0,0,0] on the scene and does everything have the same reference point?**

The origin of the universe is (0,0,0). The center-of-the-screen origin is wherever the eye is looking.

**13:26:56 Can shaders only manipulate opengl stuff or can they directly modify texture files as well?**

Shaders can read textures in and use them as image textures, data tables, etc. They can also render a scene into a texture which then can be read and used by another shader.

**13:27:50 Linear algebra is very cool.**

I think I will have that embroidered on a pillow...

**13:28:31 Is it a good way to think of shaders as a computer generated textures?**

Close. But I think the CG world would call them procedurally-generated scene displays.

**13:36:38 For Linux pals struggling with -lglew: install the `glew` package and try -lGLEW**

Good observation – thanks for sharing.

**13:47:50 The discord for this class is: <https://discord.com/invite/aW7PWdUt>**

Thanks for letting everyone know! I don't go there myself because I want you all to have a place to communicate safe from faculty oversight, so I had forgotten what the link is.

**13:53:40 Once I implement the ellipses with uAd and uBd will that negate the error? [The error about not able to find variables uAd and uBd]**

Yes. The vendors' shader compilers are *really* good. They have to be – they want your shaders to run as fast as possible. So, if a variable doesn't eventually contribute to coloring a pixel, the compiler will do you a favor by deleting that variable and any code used to create it and any code that uses it. But you put that variable in your .glib file to make its slider. So glman reads the .glib file and thinks it needs to connect a slider to a variable, but it can't find that variable in your symbol table. That's what the message means. It's really just a warning about something that might be wrong, but probably isn't.