12:06:17 From Jacob Eckroth(He/him/his) : for any of y’all that didn’t get a chance last class, we have a lil student discord for the class if ya wanna join it https://discord.gg/YxJU6w

You heard it here first, folks.

12:13:12 From Jacob Eckroth(He/him/his) : ooooh it’d be fun to do like solar system with showing trails behind planets

Solar systems are very fun! We will talk about them and the Final Project on Monday.

12:15:07 From Todankar, Diksha Pritam : any example of a motion blur so that I can see for myself?

It’s in all the Pixar movies, but there are especially many great motion-blur moments from *The Incredibles* since so much action happened so fast. Watch for it.

First-ever example of motion blur was this Pixar image:
You can also do it using OpenGL. I used it once in mechanical fourbar linkage animation:

![OpenGL Animation](https://www.youtube.com/watch?v=d7uwGIPQdWs)

This is great! I had not seen this before.

12:41:12 From Koning, Jonathan Scott: Why don't the objects change [stripe] color when using world coordinates?
12:41:48 From Haines, Grant A: I'm guessing that the color is being set per-object

That's correct – good answer. The color for each object was being passed in as a uniform variable right before that object was drawn.

12:42:54 From Koning, Jonathan Scott: What is more taxing to the GPU? Shaders or textures?

It depends how simple or complex the shaders are. It could go either way. Doing texturing from a shader is probably in a tie with doing it from stock OpenGL.

12:43:54 From Hoang: Are the uniform variables passed in?

Yes, they are made available to the shader program.

12:55:05 From Todankar, Diksha Pritam: we could use this to make slimy objects
12:55:35 From Jiroux, Kira Lynn: and transparent! make some goopy looking things

Yeah, like the ghosts in *GhostBusters*, especially Slimer. 😊

12:56:47 From Evan: And the shader calculations are done in the GPU right?

That's right. Your code gets compiled, linked, and downloaded into GPU memory. It gets executed by the GPU, taking advantage of all that parallelism.

13:30:55 From Luong, Kenny: [https://www.youtube.com/watch?v=lxUPo9tRao](https://www.youtube.com/watch?v=lxUPo9tRao)

Dancing cow. OMG.
I remember seeing this video at SIGGRAPH one year. That is so funny!

**From Lloyd, Doug:** How is that blending [the morphing cow] being achieved?

Most of it happens in the vertex shader:

```glsl
#version 330 compatibility

uniform float uBlend;
uniform bool uCube;

out vec4 vColor;
out float vLightIntensity;

const float SIDE = 1.;

void main( )
{
    vec4 vertex0 = gl_Vertex;
    vertex0.xyz *= 2./length(vertex0.xyz);
    if( uCube )
    {
        vertex0.xyz = clamp( vertex0.xyz, -SIDE, SIDE );
    }
    vec3 tnorm = normalize( vec3( gl_NormalMatrix * gl_Normal ) );
    vec3 LightPos = vec3( 5., 10., 10. );
    vec3 ECposition = vec3( gl_ModelViewMatrix * gl_Vertex );
    vLightIntensity = abs( dot( normalize(LightPos - ECposition), tnorm ) );

    vColor = gl_Color;
    gl_Position = gl_ModelViewProjectionMatrix * mix( gl_Vertex, vertex0, uBlend );
}
```

`mix( )` is a built-in GLSL function that blends 2 quantities like this:

\[
    \text{output} = (1.-\text{uBlend})\times\text{input0} + \text{uBlend}\times\text{input1};
\]

\[0. \leq \text{uBlend} \leq 1.\]

I got a question of the form “where can I find a list of the GLSL built-in functions”? After looking around some, I concluded that the best place is pages 9-12 of the OpenGL Reference Card: