Why GLSL?

1. It doesn't matter
   Had part is model
   Alternatives are very similar, not HLSL, Cg

2. Built-in to OpenGL
   Documented
   Standard
   Available
   No extra software

3. Compiler built into Driver
   No extra tools
   No runtime issues (did compiler know about card)
   No separate compilation phase

But: Run-time compiler errors
     Compatibility between Drivers

4. Language designed for the task
   data types, vectors, matrices correspond to OpenGL
   manipulations (swizzles, operators, functions)
Types of "variables" ← things passed to/from programs

Uniform - stays constant over triangle
can't change in begin/end

Attribute - per-vertex information

Varying - interpolated value
(thing to pass from Vertex → Fragment)

Special Variables (built ins)

`gl_Position ← output of Vertex Program`
`gl_ModelViewProjectionMatrix ← gl_Vertex`

Which one are set up automatically?

have access to all OpenGL state (as attributes/uniforms)
unclear what is made varying without explicit

`gl_Vertex → Vertex Properties`
`gl_Normal ← Vertex Properties`
`gl_Color ← Vertex Properties`

How do Normals transform?

\((M^{-1})^T\) (adjoint)

*note what happens if rotation

Vertex Outputs

`gl_Position`
`gl_TexCoord`
`gl_FrontColor ← notice!`
Accessing Texture Maps (beware - might not work on old machines)
Via a "Sampler"
- set up a texture unit in C++
- tell program about it
- special data type "Sampler"
  uniform sampler2D tex;
  (see the C++ syntax in the book)
- texture functions look up in samplers
  texture2D (tex, vec2)

Dependent texture read
one lookup depends on another (indirection)
  texture2D (tex2, texture2D (tex1, vec))
Hack Shading

1. Not Flat
   - Displacement Map
   - Bump Map
   - Coordinate System Issues

2. Shadow Map
   - Hack Shadows
   - Stencil Buffers

3. Light Maps / Glass Maps