Why GLSL?

(1) It doesn't matter
Hard part is model

Alternatives are very similar HLSL, Cg
└ NOT FOR OPENGL

(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
  ex: lighting
- Attribute - per-vertex information
  ex's normal, color, ...
- Varying - interpolated value
  (thing to pass from Vertex → Fragment)
  ex: colors, texcoords

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
gl_Normal
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 ⟶ Basic Abstractions
- Triangles
- Texture
- Local Lighting
  - + per-pixel
  - + multi-texture
  - + Stencil

① NOT FLAT
- Displacement Map
- Bump Map
  - Coordinate System Issues

② Shadow Map
- Hack Shadows
- Stencil Buffers

Shadow Maps

③ Light Maps / Gloss Maps