Let’s Fix OpenGL
Adrian Sampson, Cornell
CPU

Rendering Pipeline
programmable & fixed-function stages

Display
C, C++, JavaScript

CPU → GLSL (Vertex Shader) → GLSL (Fragment Shader)

- Vertex positions
- Pixel colors
Vertex Shader

```cpp
in vec4 position;
in float dist;
out vec4 fragPos;
void main() {
    fragPos = position;
    gl_Position = position + dist;
}
```

Fragment Shader

```cpp
in vec4 fragPos;
void main() {
    gl_FragColor = abs(fragPos);
}
```
```c
static const char *vertex_shader =
    "in vec4 position; ...";
static const char *fragment_shader =
    "in vec4 fragPos; ...";

GLuint vshader = glCreateShader(GL_VERTEX_SHADER);
// ... more boilerplate ...
glLinkProgram(program);

GLuint loc_dist =
    glGetUniformLocation(program, "dist");

glUseProgram(program);
glUniform1f(loc_dist, 4.0);
// ... assign other "in" parameters ...
glDrawArrays(...);
```
Shader languages are subsets of supersets of C.
 Shader languages are subsets of supersets of C.

```
struct T { ... }
```
declares a typed named:

A. `struct T`

B. `T`  

Like C++
#1 Shader languages are subsets of supersets of C.

Declarations inside conditions

```c
if (bool b = ...) { ... }
```

are allowed.

true

false Like C
ARM Mali GPU on Android. Correct output.

“It looks like there was indeed an obscure register allocation bug in the driver...” —ARM

from “Bugs can be beautiful,” by Alastair Donaldson. https://medium.com/@afd_iocl/65b93c5c58f9

#1 Shader languages are subsets of supersets of C.

⇒ Apply work on language extensibility.
Loose CPU-to-GPU and stage-to-stage coupling.

```c
GLuint loc_dist = glGetUniformLocation(program, "dist");

glUniform1f(loc_dist, 4.0);

in float dist;
```

CPU setup

CPU render

vertex shader
#2 Loose CPU-to-GPU and stage-to-stage coupling.

```cpp
GLuint loc_dist = glGetUniformLocation(program, "dist");

float dist;
vec4 fragPos;

fragPos = position;
```

```cpp
in float dist;
out vec4 fragPos;

in vec4 fragPos;
```

CPU setup

CPU render

vertex shader

fragment shader
#2 Loose CPU-to-GPU and stage-to-stage coupling.
⇒ Cross-language static analysis for the short term.
⇒ Single-source CPU+GPU languages for the long term.
#3 Massive metaprogramming without hygiene.
#3 Massive metaprogramming without hygiene.

Übershader
#3 Massive metaprogramming without hygiene.

```c
in bool param;
if (param) {
    ...
}
```

VS.

```c
#ifdef _PARAM
    ...
#endif

glShaderSource("#define _PARAM", "...");

#define _PARAM

#endif

#define _PARAM

#endif

glUniform1b(param_loc, true);
```
#3 Massive metaprogramming without hygiene.

⇒ Scale up safe metaprogramming tools to generate thousands of shader variants.
#4 Missing general theory for execution rates.

- **CPU:**
  ```
  float* pos = malloc(...);
  ```

- **Indexing**
  ```
  fragPos = position;
  ```

- **Interpolation**
  ```
  gl_FragColor = abs(fragPos);
  ```
#4 Missing general theory for execution rates.
#4 Missing general theory for execution rates.

Define a core $\lambda_{\text{GPU}}$ to describe programming with rates and a translation from OpenGL.
#5 Latent types for linear algebra.

- bunny space
- slide space
- teapot space
Latent types for linear algebra.

```cpp
in mat4 bunny_model;
in vec4 position;

vec4 position_slide = bunny_model * position;
```
Latent types for linear algebra.

```
in mat4 bunny_model;
in vec4 position;
in vec4 normal;

vec4 position_slide = bunny_model * position;
vec4 normal_slide = bunny_model * normal;

normal_slide - position_slide

× normal - position_slide
```
#5 Latent types for linear algebra.

⇒ Use a type system to track the space for each vector.
⇒ Synthesize matrix–vector multiplications to declaratively obtain a vector in the right space.
#6 Visual correctness.
#6 Visual correctness.

Add aspirational Phong test and its intrinsics

@sampsyo committed on Oct 31, 2015

10 months!

Fix wrong variable reference in phong shader

@sampsyo committed on Aug 4, 2016

Showing 1 changed file with 1 addition and 1 deletion.

```glsl
12 12    fragment glsl<
13 13       # Convert to world space.
14 14       var position_world = vec3(model * vec4(pos, 1.0));
15 15       var normal_world = normalize(vec3(model * vec4(pos, 0.0)));
15 15       var view_dir_world = normalize(camera_pos - position_world);
```
#6 Visual correctness.

⇒ Apply work on live coding to shorten the debug cycle.
⇒ Apply crowdsourcing to evaluate visual quality.
Application

“Engine”

new programming models?

Vulkan

GPU
For everyone saying "Vulkan!", the conclusion is that there is an opportunity for an API between Vulkan and the game engines. I agree.
Special thanks to:
Kathryn McKinley
Todd Mytkowicz
Yong He
Kayvon Fatahalian
Tim Foley
Pat Hanrahan