Static Stages for Heterogeneous Programming

Adrian Sampson, Cornell
Kathryn S McKinley, Google
Todd Mytkowicz, Microsoft Research
Apple iPhone 6s

Smartphone

Application Processors

– Sneak Peak, as promised!

APL1022 TSMC 16 nm FinFET

APL0898 Samsung 14 nm FinFET

NOTE: False color and image sharpening has been applied to the photos for the purposes of this article. High resolution images in Chipworks reports are not retouched.

Apple A9

techinsights.com
Apple A9

techinsights.com
Datacenter Servers

Microsoft Catapult

Mobile SoCs

GPUs
DSP
ISP
audio codecs
video codecs
modems
CPUs

Google TPU
unified program
Heterogeneous programming languages need support for **placement** and **specialization**.

With extensions, **multi-stage programming** can support both concepts.

Current APIs for **real-time graphics** are especially unsafe, verbose, and brittle. We can help.
Heterogeneous programming languages need support for placement and specialization.

With extensions, multi-stage programming can support both concepts.

Current APIs for real-time graphics are especially unsafe, verbose, and brittle. We can help.
CPU

Rendering Pipeline
programmable & fixed-function stages

GPU

Display
C, C++, JavaScript

CPU

Vertex Shader

vertex positions

GLSL

Fragment Shader

pixel colors

GLSL
Vertex Shader

```glsl
in vec4 position;
in float dist;
out vec4 fragPos;

void main() {
    fragPos = position;
    gl_Position = position + dist;
}
```

Fragment Shader

```glsl
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 program = compileAndLink(vertex_shader,
                                 fragment_shader);

// ... more boilerplate ...

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

// "dits"

CPU "Host Code"

// setup

// render a frame

glUseProgram(program);
glUniform1f(loc_dist, 4.0);
// ... assign other "in" parameters ...
glDrawArrays(...);
```
GPU shader specialization

Übershader

#define

#if

#ifndef

#endif

#endif

#ifdef

#endif
Heterogeneous programming today

Separate programs in separate languages

Stringly typed communication

Unscalable, unsafe specialization
Heterogeneous programming languages need support for **placement** and **specialization**.

With extensions, **multi-stage programming** can support both concepts.

Current APIs for **real-time graphics** are especially unsafe, verbose, and brittle. We can help.
Classic multi-stage programming: types for metaprogramming

```javascript
function pow(x, n) {
    if (n == 1) {
        return x;
    } else {
        return x * pow(x, n - 1);
    }
}
```

`pow(2, 3) → 8`

`genpow("2", 3) → "2 * 2 * 2"`

`eval(genpow("2", 3)) → 8`
function genpow(x, n) {
    if (n == 1) {
        return x;
    } else {
        return x * pow(x, n - 1);
    }
}

genpow("2", 3) → "2 * 2 * 2"
Classic multi-stage programming: types for metaprogramming

```javascript
function genpow(x, n) {
    if (n == 1) {
        return x;
    } else {
        return x + " * " + pow(x, n - 1);
    }
}

genpow("2", 3) → "2 * 2 * 2"
```
Specializing on a compile-time parameter

```
gl_FragColor = if matte diffuse (diffuse + ...)
```

render-time parameter

```
gl_FragColor = [ if matte <diffuse> <diffuse + ...> ]
```

condition on the GPU

host-side parameter

condition on the host
Performance impact of specialization in BraidGL

frame latency (ms)

- original
- GPU if
- specialized
- per-vertex
Performance impact of specialization in BraidGL
Heterogeneous programming languages need support for **placement** and **specialization**.

With extensions, **multi-stage programming** can support both concepts.

Current APIs for **real-time graphics** are especially unsafe, verbose, and brittle. We can help.