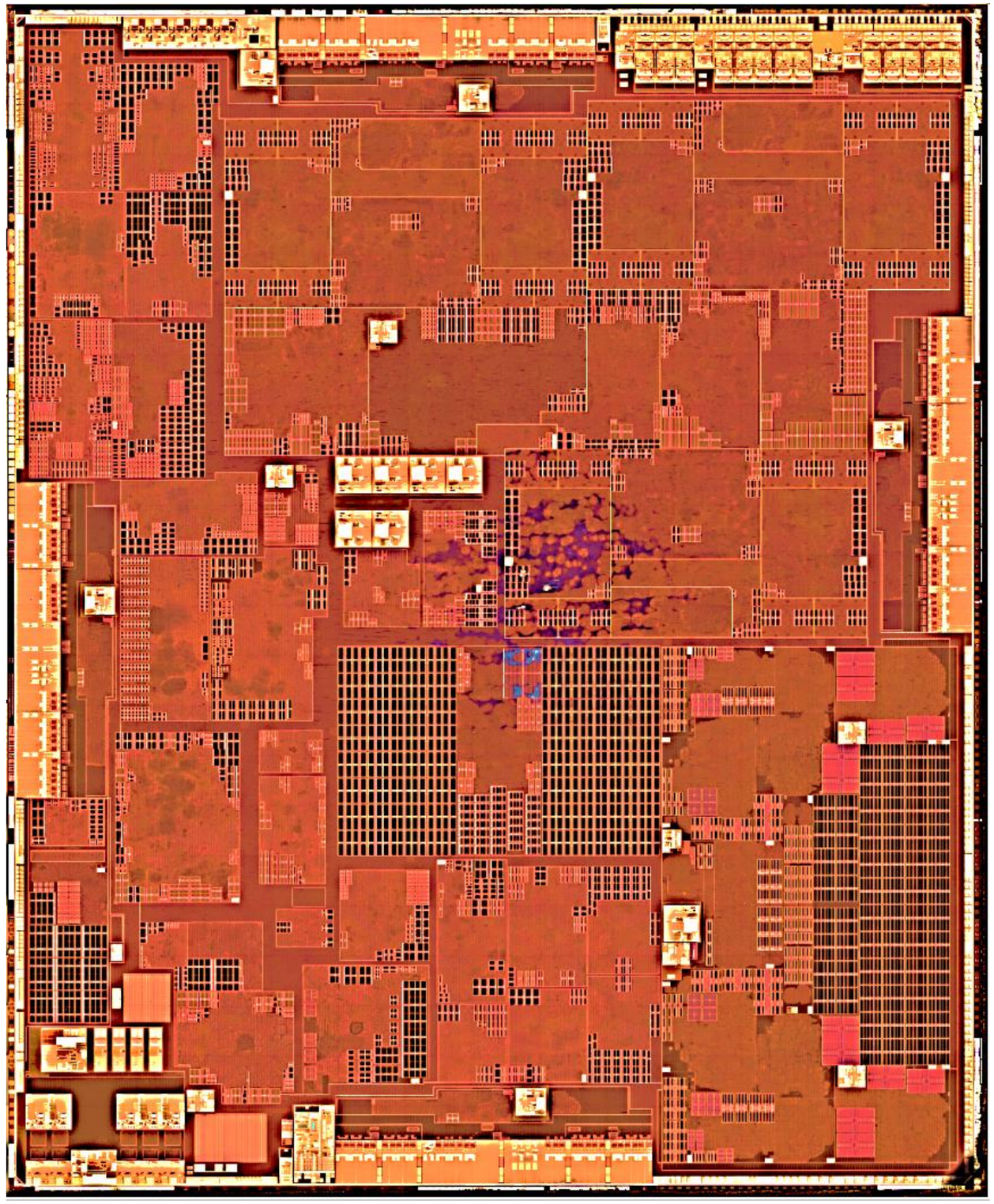
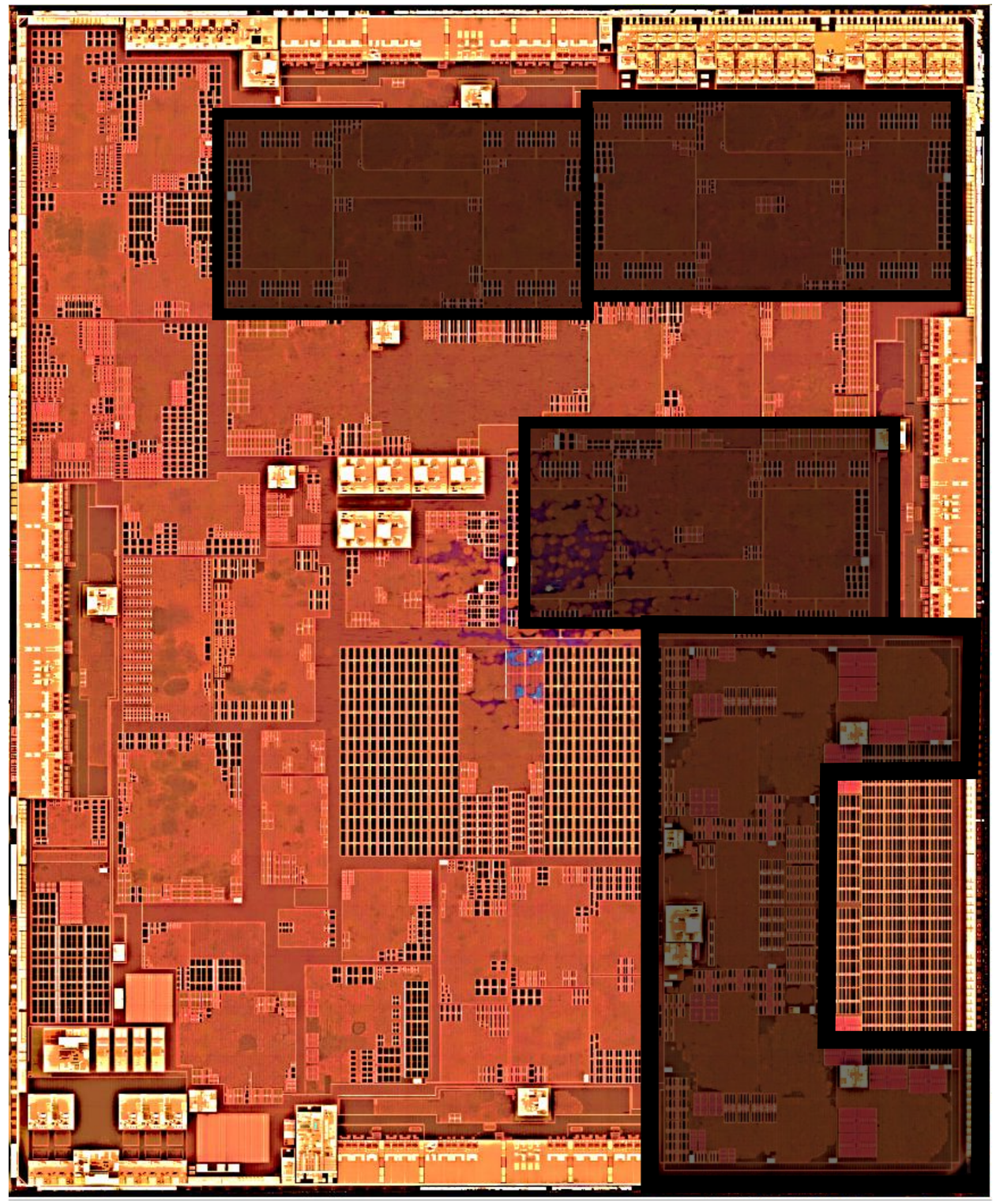
Static Stages for Heterogeneous Programming

Adrian Sampson, Cornell
Kathryn S McKinley, Google
Todd Mytkowicz, Microsoft Research



Apple A9

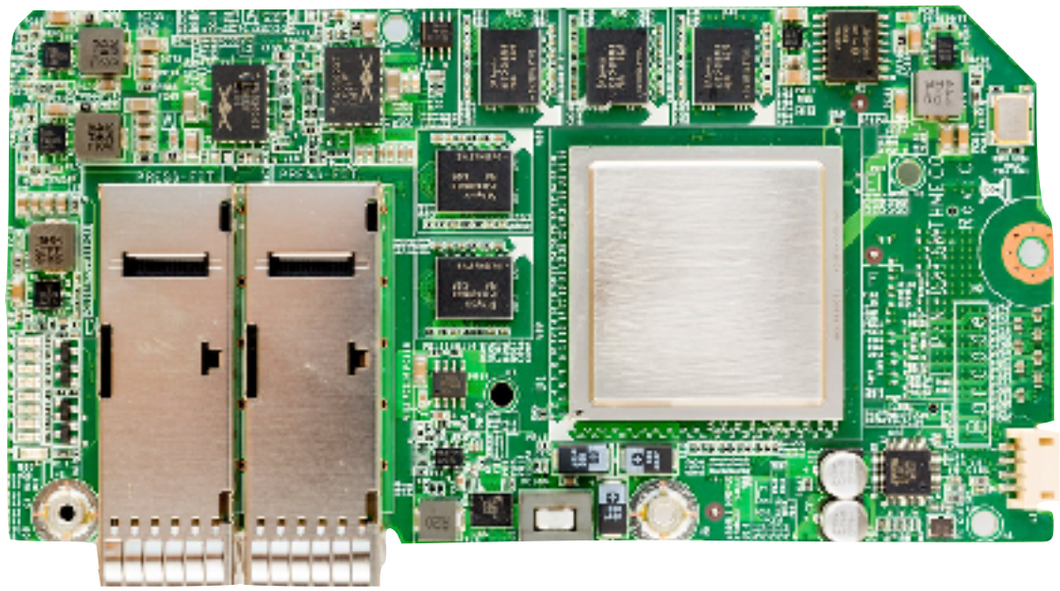
techinsights.com



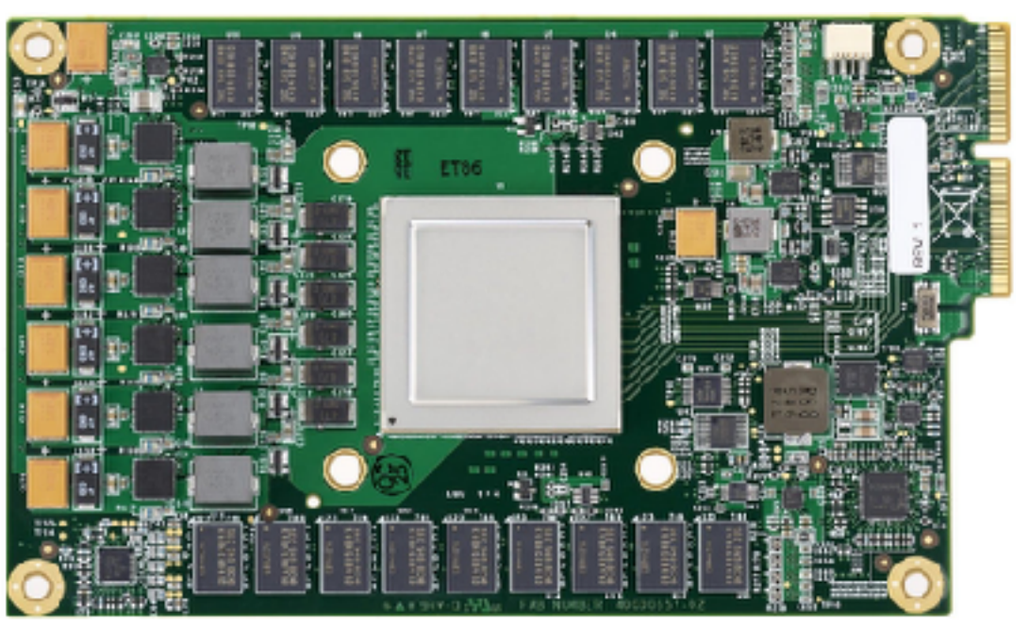
GPUs
DSP
ISP
audio codecs
video codecs
modems
— CPUs

Apple A9
techinsights.com

Datacenter Servers



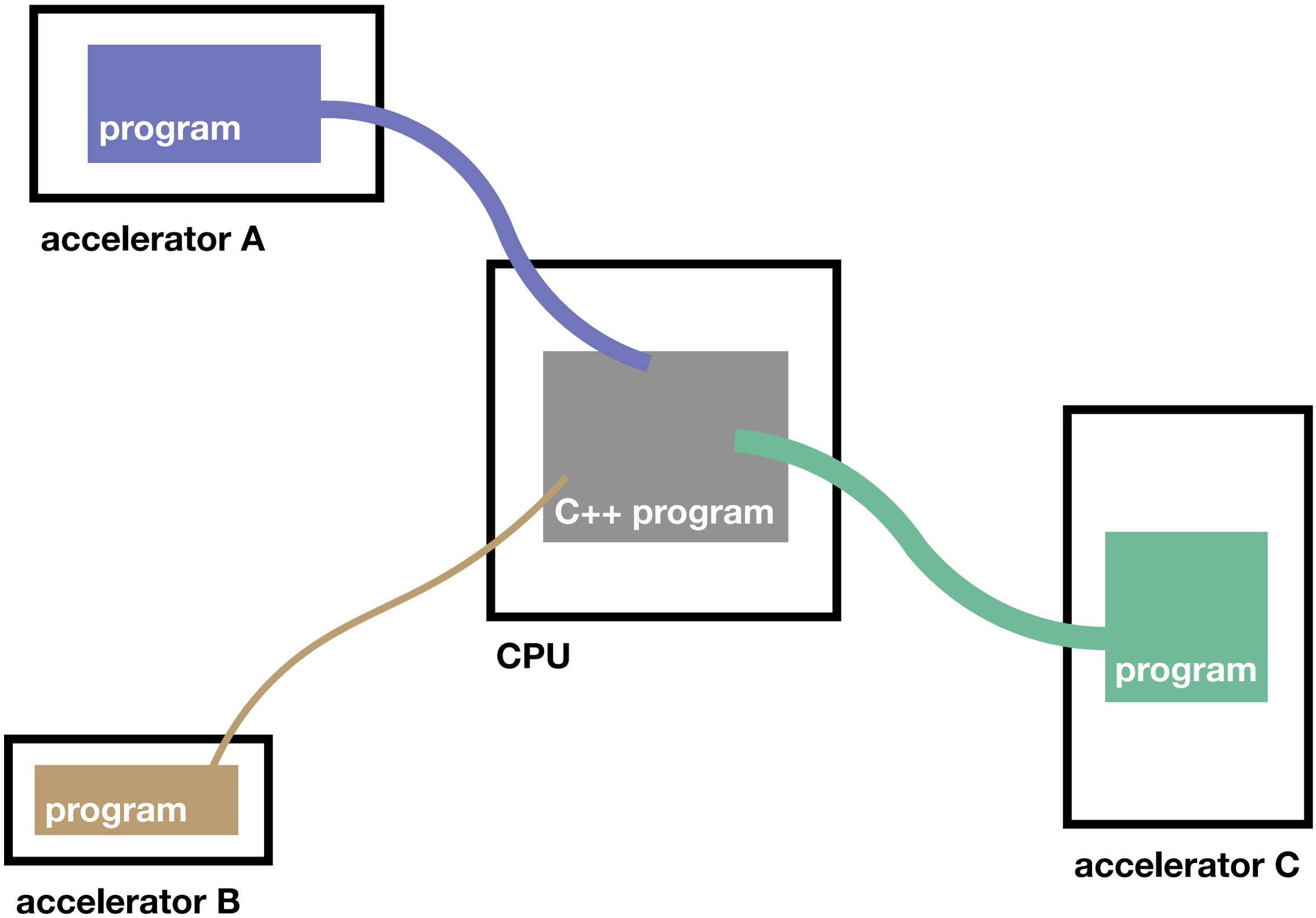
Microsoft Catapult

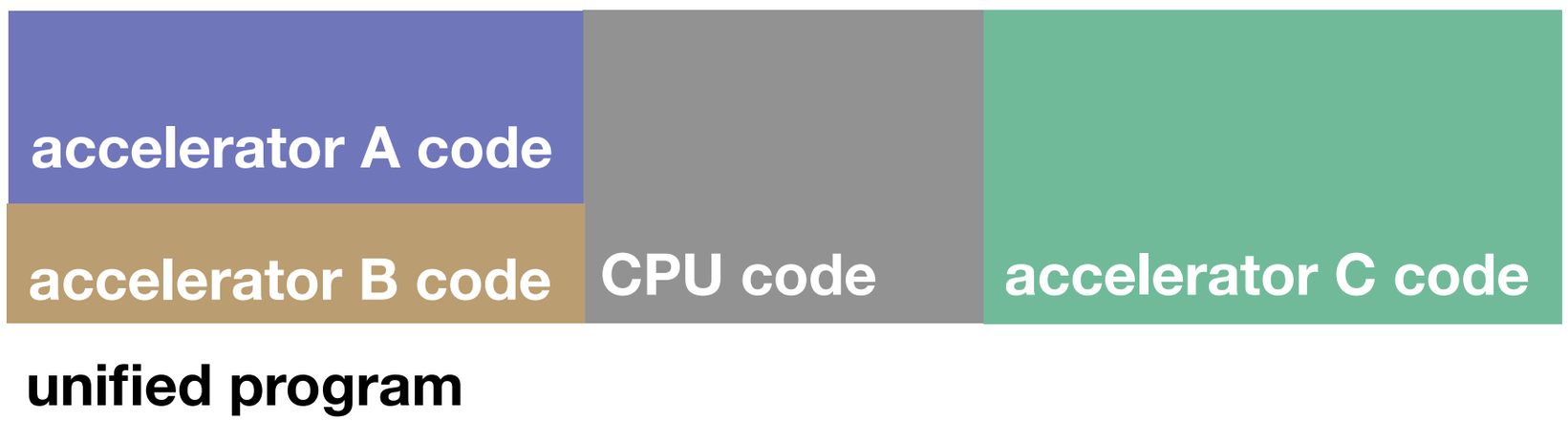


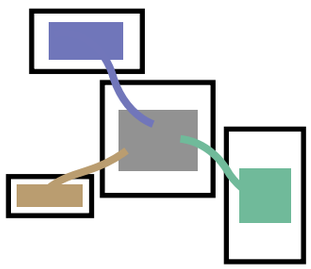
Google TPU

Mobile SoCs

GPUs
DSP
ISP
audio codecs
video codecs
modems
CPUs


program
accelerator A
C++ program
CPU
program
accelerator C
program
accelerator B

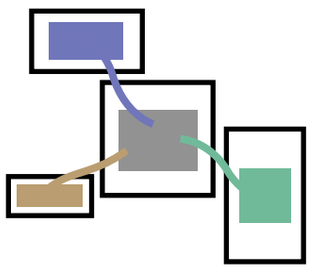

accelerator A code
accelerator B code
CPU code
accelerator C code
unified program



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.

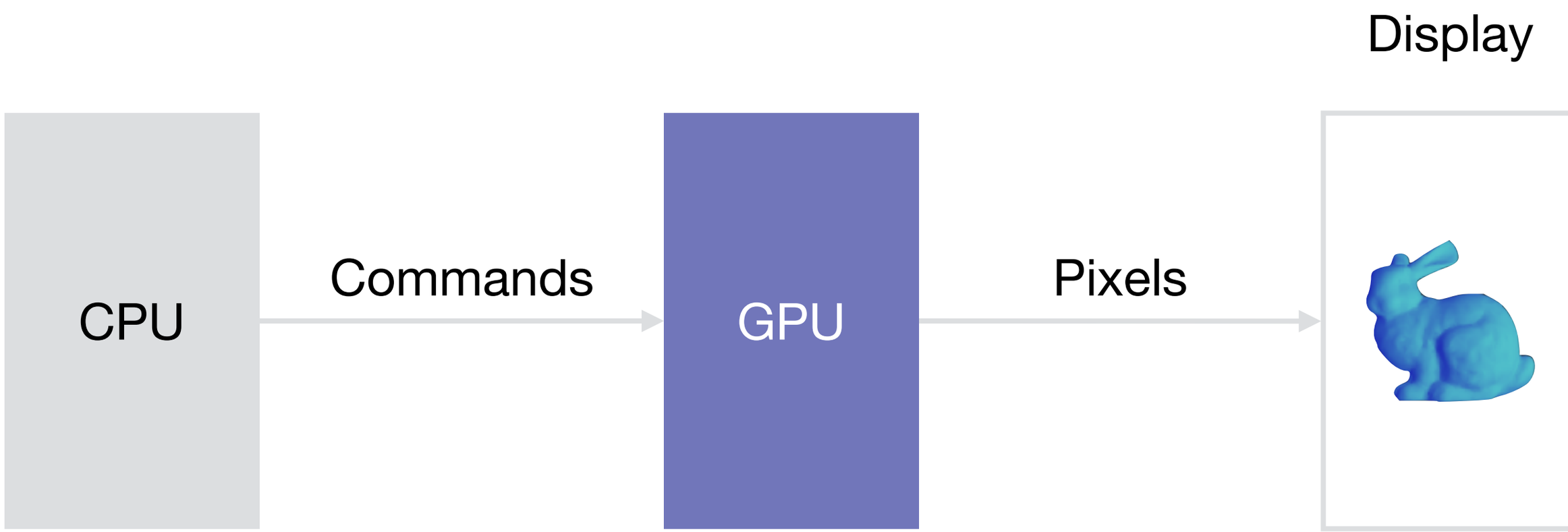


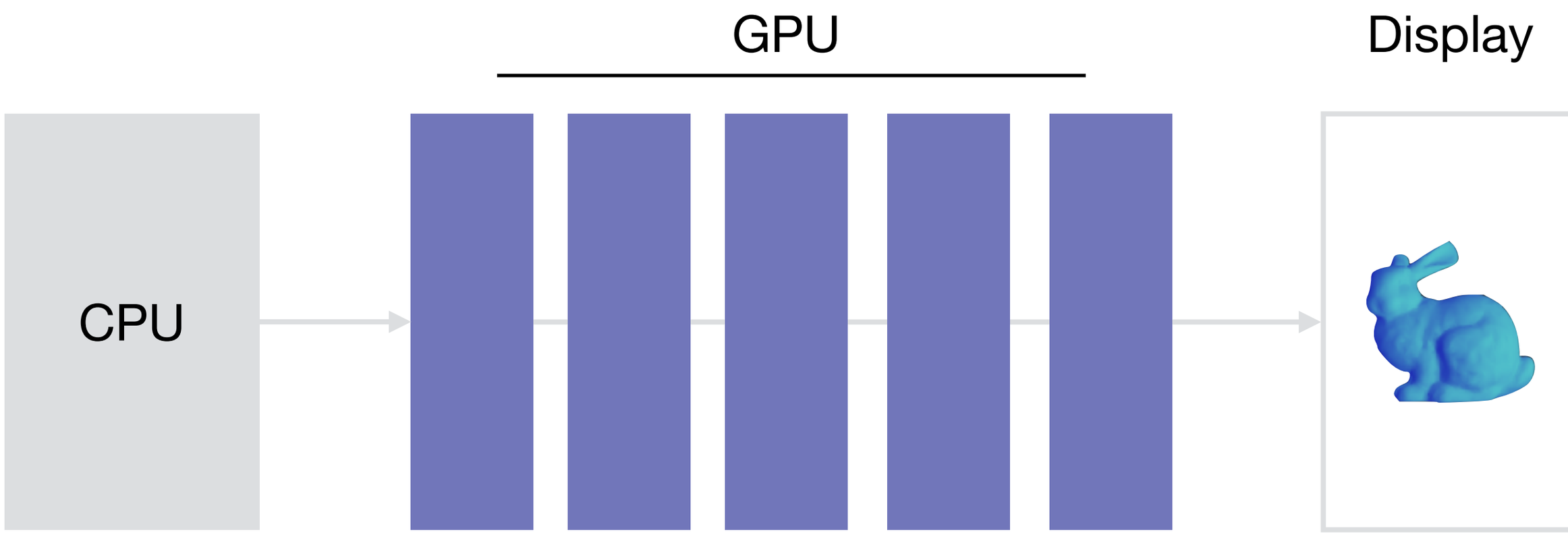
Heterogeneous programming languages need support for **placement** and **specialization**.

!<[ ]>

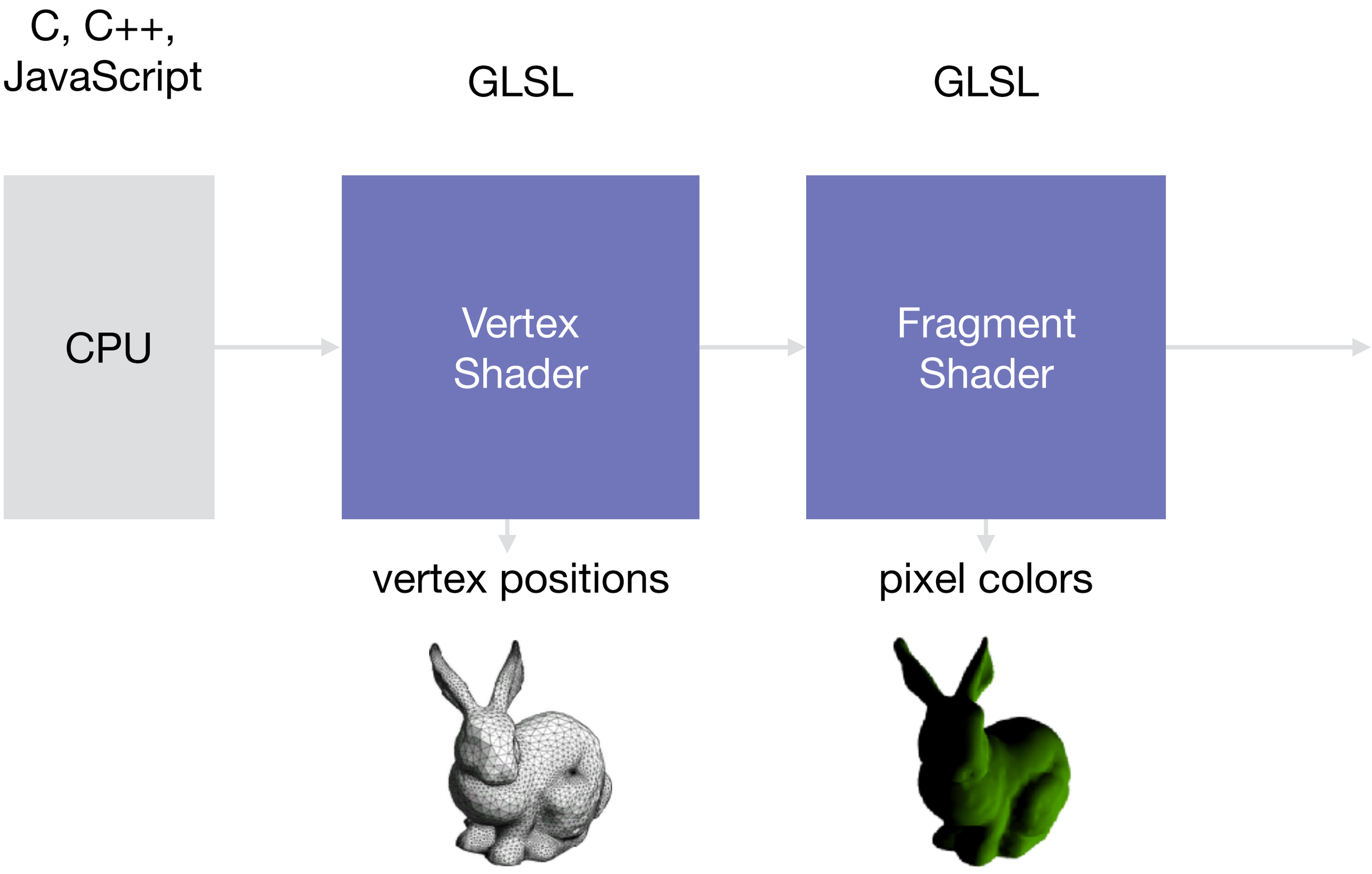
With extensions, **multi-stage programming** can support both concepts.

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


Display
CPU
Commands
GPU
Pixels



Rendering Pipeline
programmable & fixed-function stages


C, C++,
JavaScript
GLSL
GLSL
CPU
Vertex
Shader
Fragment
Shader
vertex positions
pixel colors

Vertex Shader

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

Fragment Shader

```
in vec4 fragPos;
void main() {
  gl_FragColor =
    abs(fragPos);
}
```

CPU “Host Code”

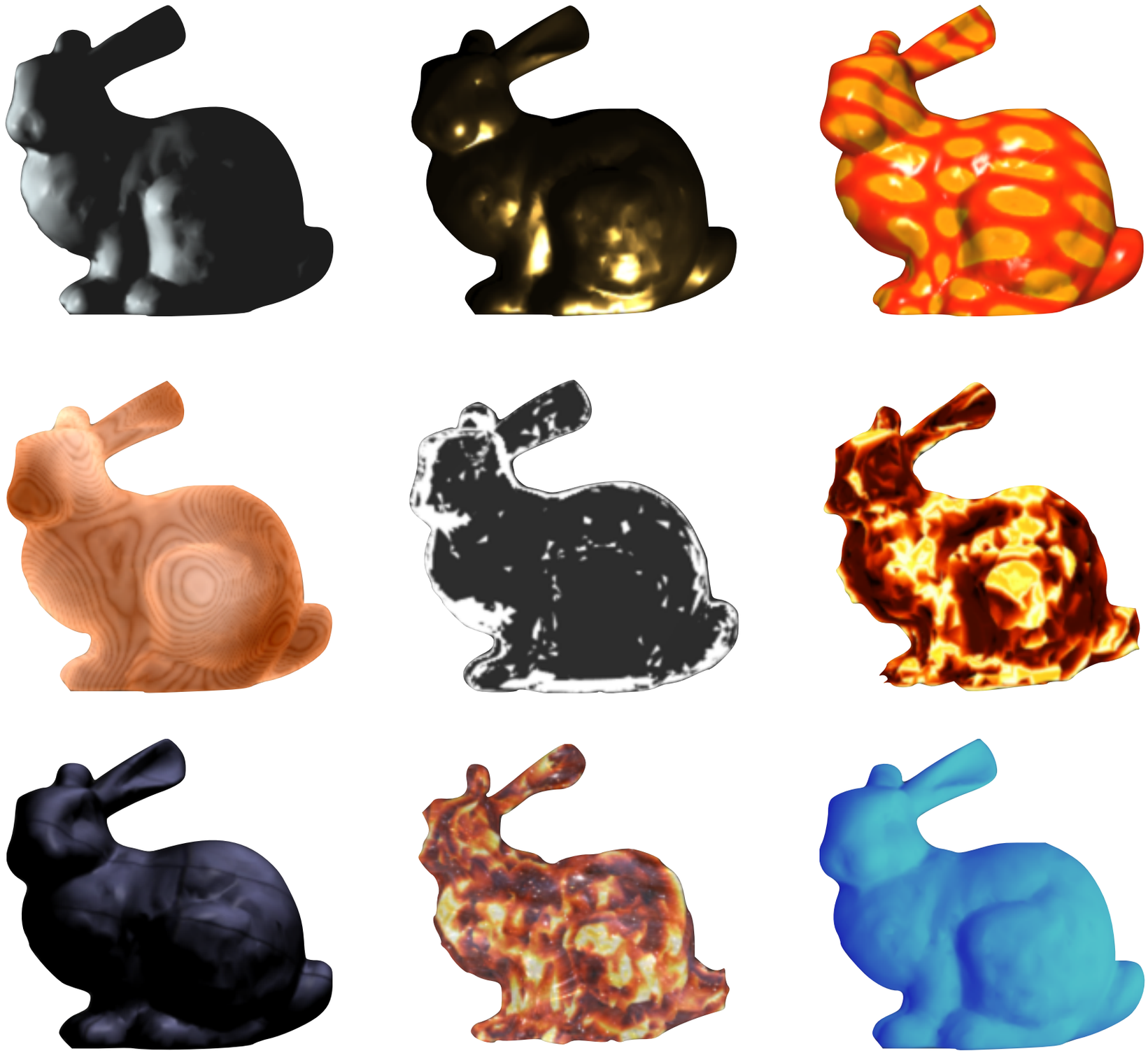
setup

```
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");
```

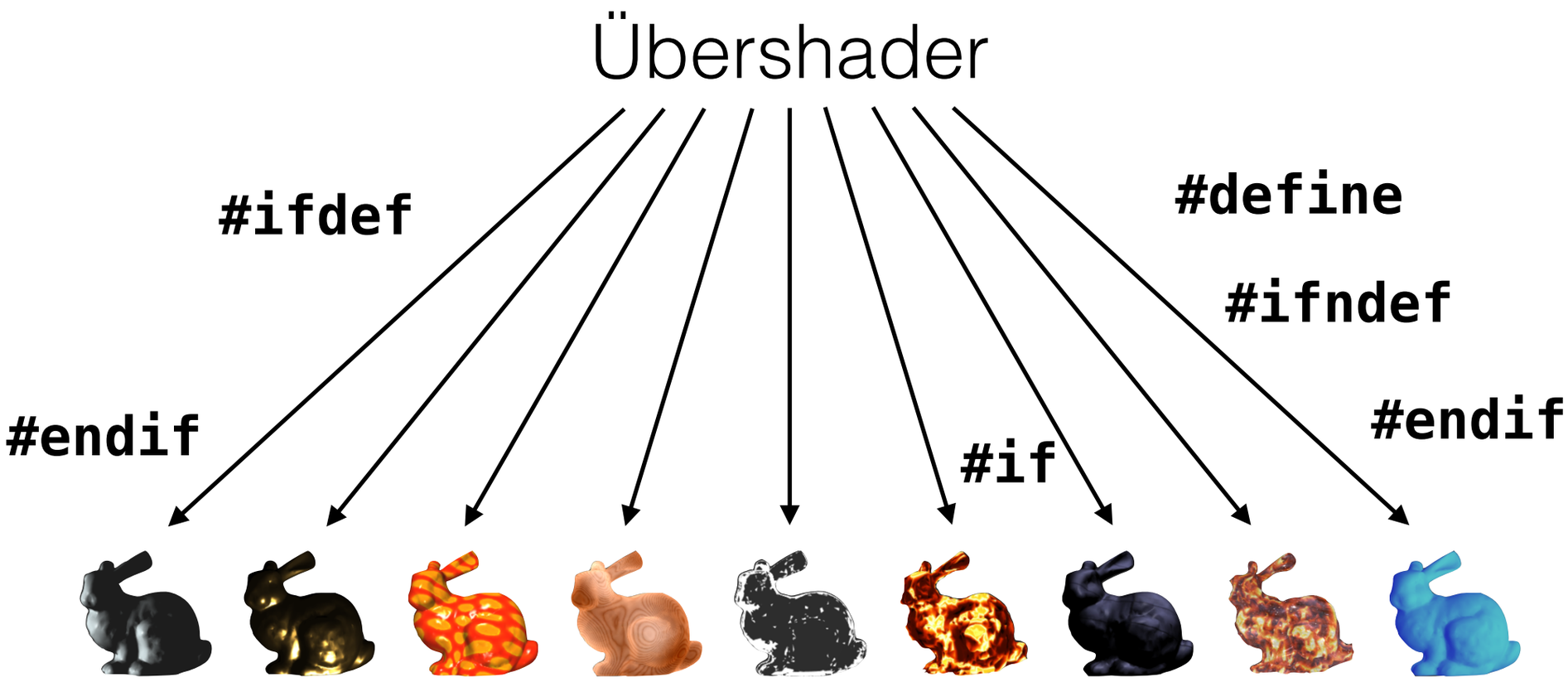
"dits"

render a frame

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



GPU shader specialization

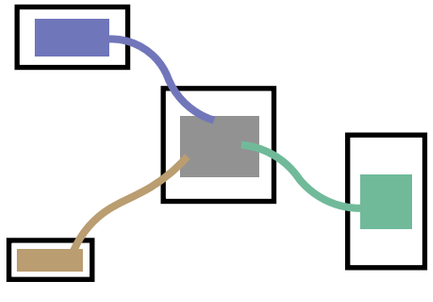


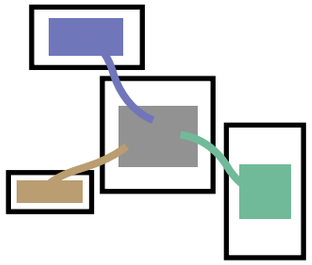
Heterogeneous programming today

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

```
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
```

Classic multi-stage programming: types for metaprogramming

expression (string)

number

```
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

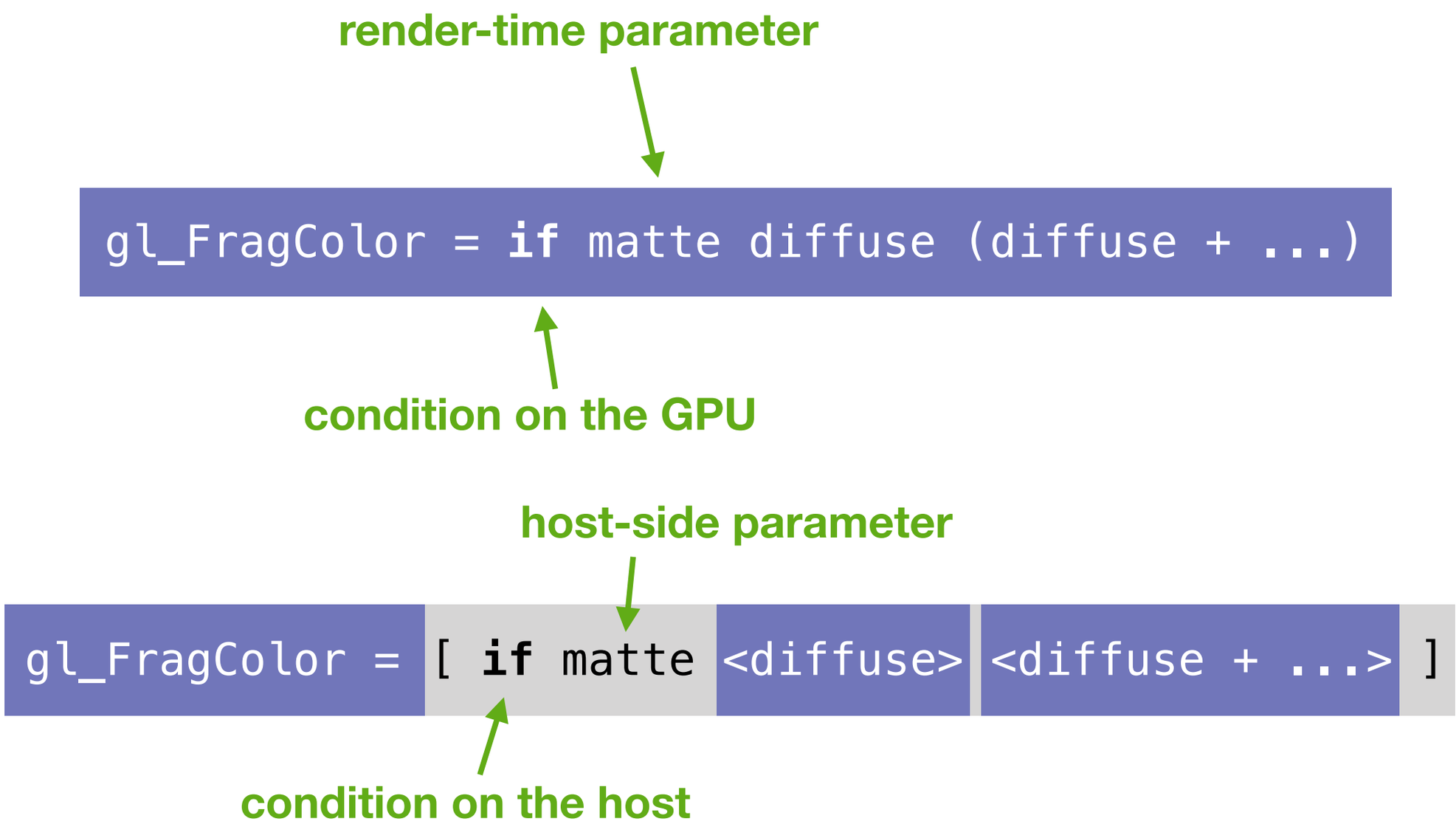
expression (string)

number

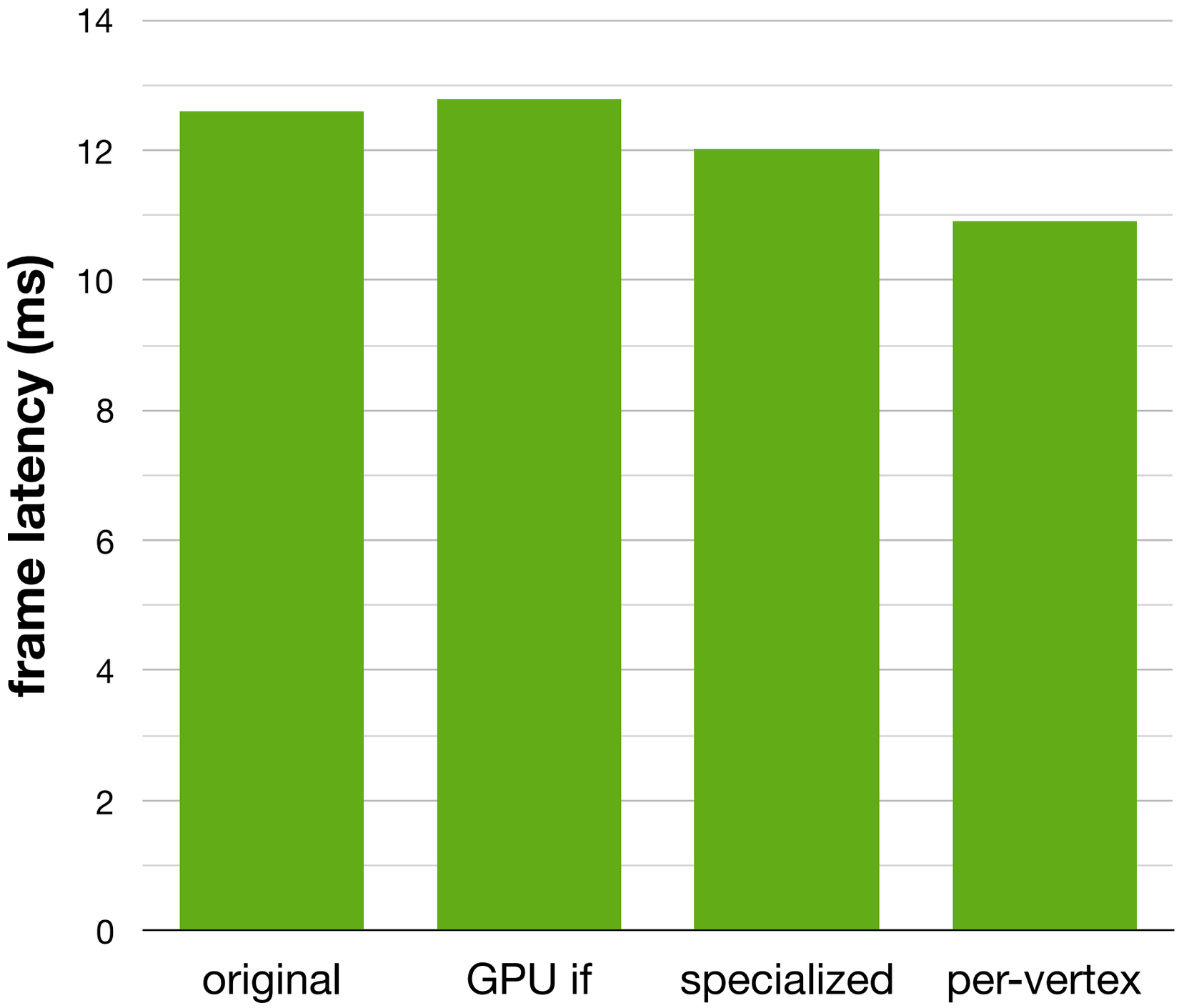
```
function genpow(x, n) {
  if (n == 1) {
    return x;
  } else {
    return x + " * " + pow(x, n - 1);
  }
}
```

genpow("2", 3) ⟶ "2 * 2 * 2"

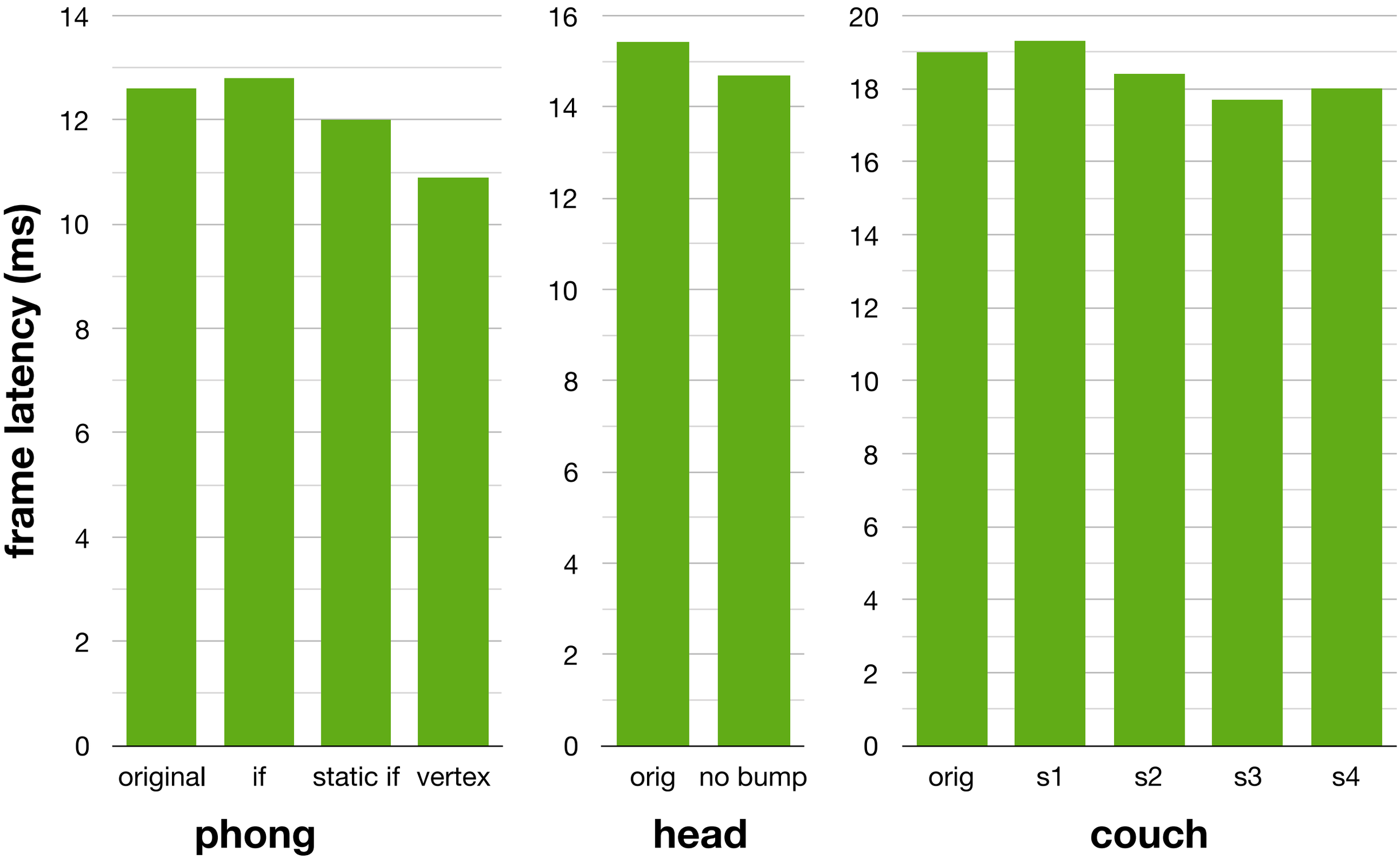
Specializing on a compile-time parameter

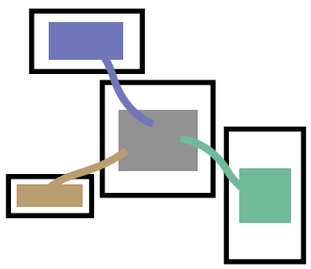


Performance impact of specialization in BraidGL



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.

braidgl.com