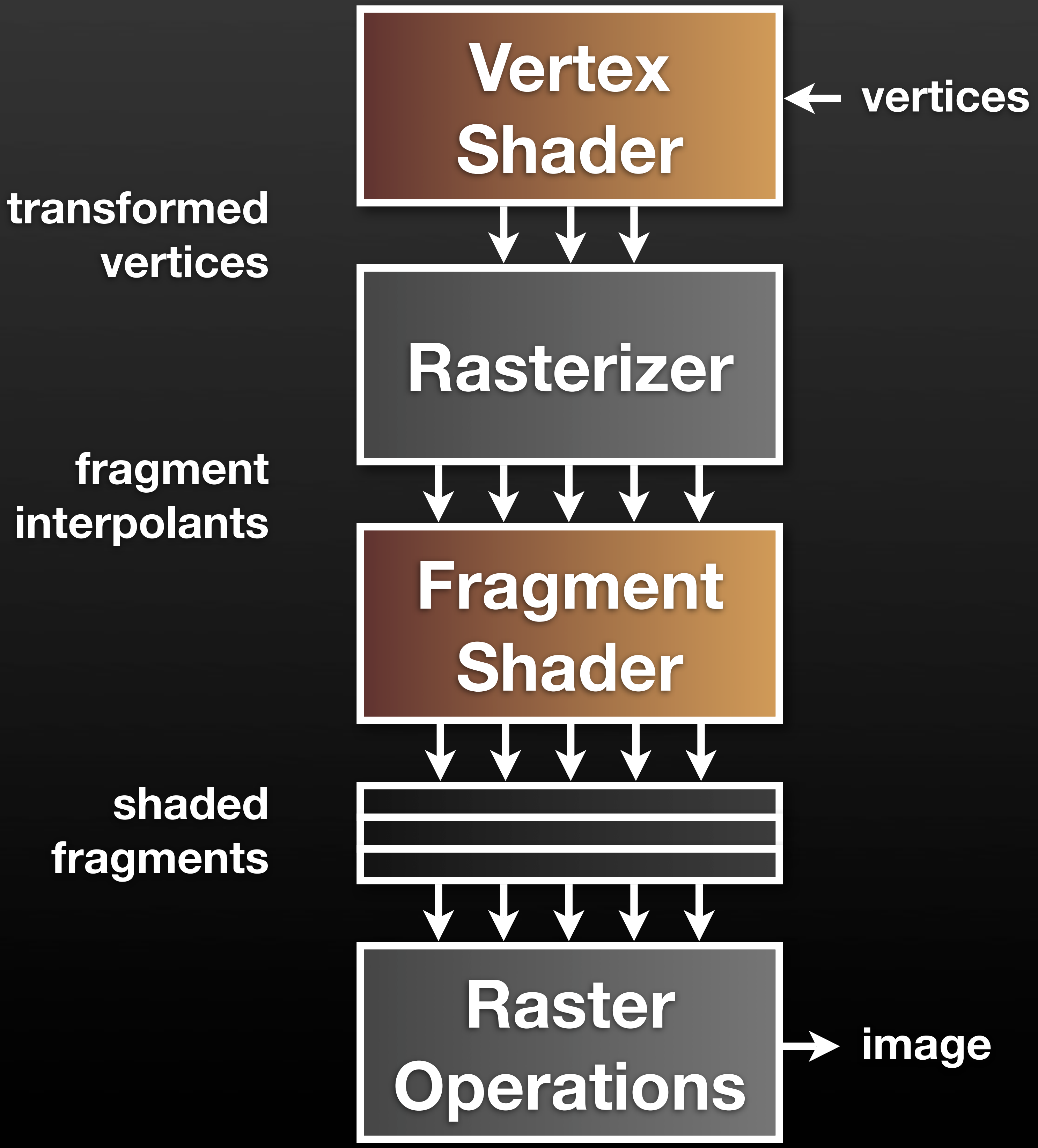
The Graphics Pipeline on a Heterogeneous Multicore

Jiawen Chen & Jonathan Ragan-Kelley

The Graphics Pipeline



vertices

Vertex Shader

transformed vertices

Rasterizer

fragment interpolants

Fragment Shader

shaded fragments

Raster Operations

image

Motivation: Convergence

GPUs becoming general-purpose

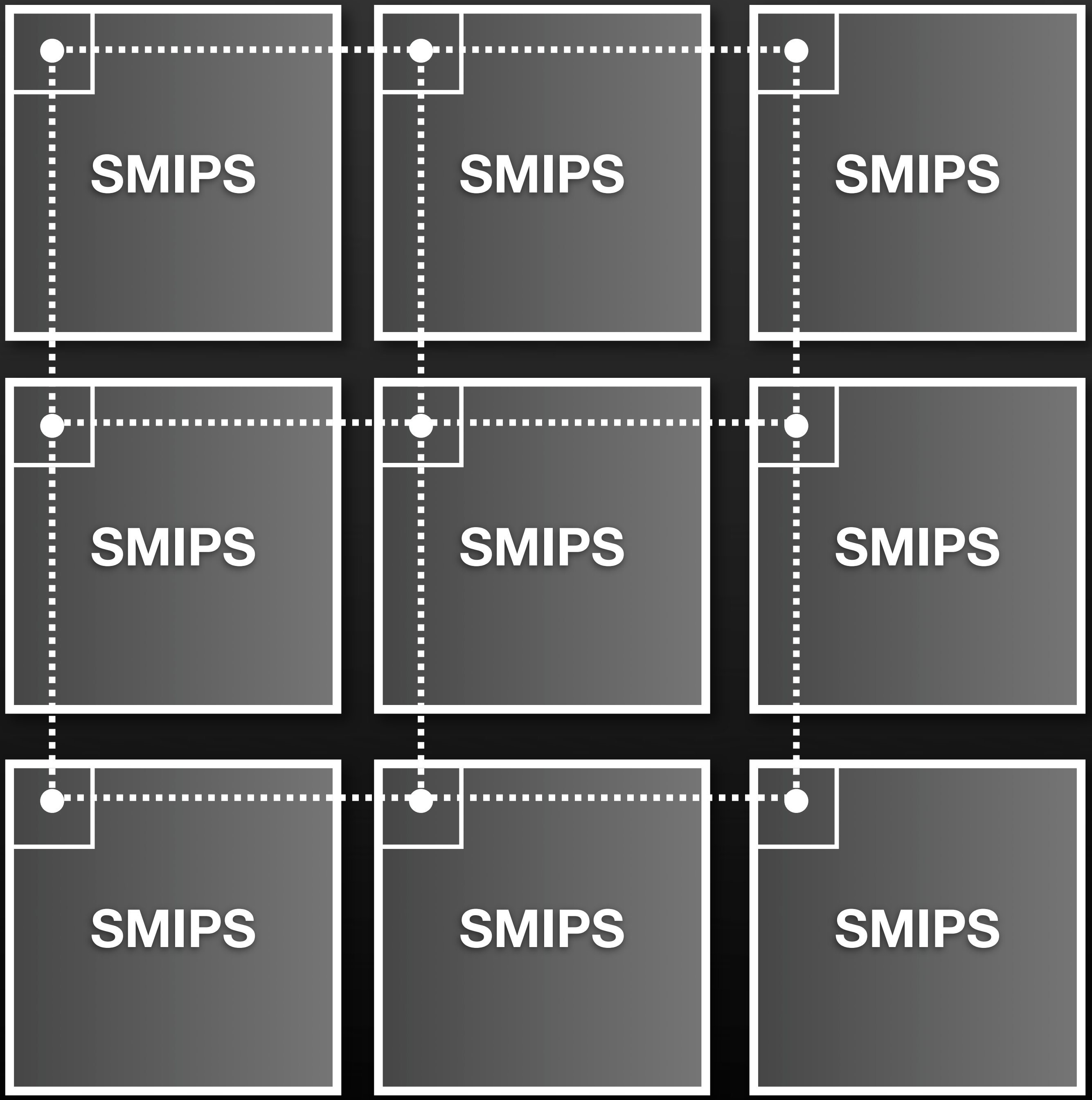
CPUs becoming parallel

Efficiency, economic advantages to convergence

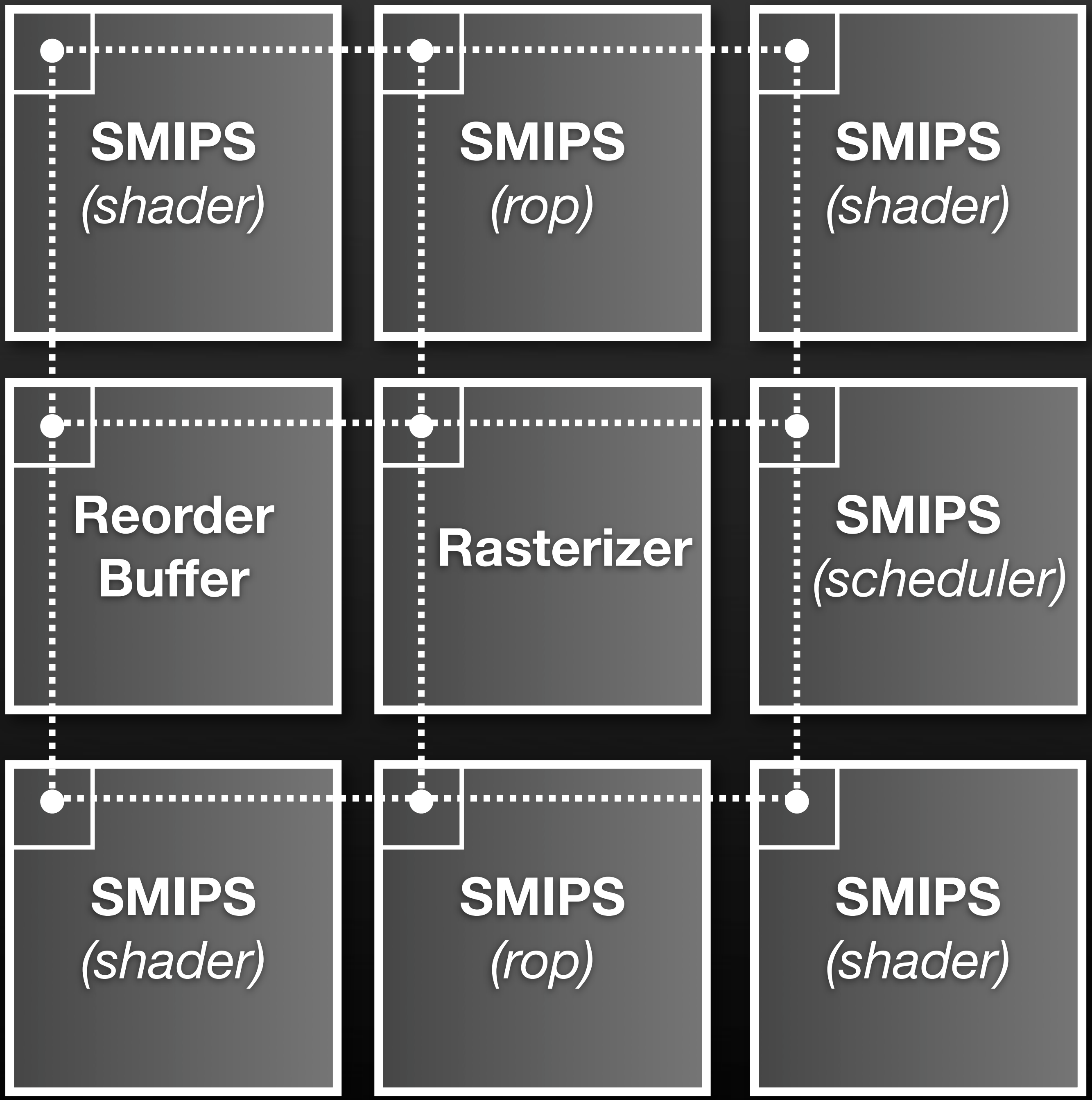
Motivation: Performance

Key parts of the graphics pipeline still benefit dramatically from fixed-function logic

Manycore



Heterogeneous Manycore



SMIPS *(shader)*

SMIPS *(rop)*

SMIPS *(shader)*

Reorder Buffer

Rasterizer

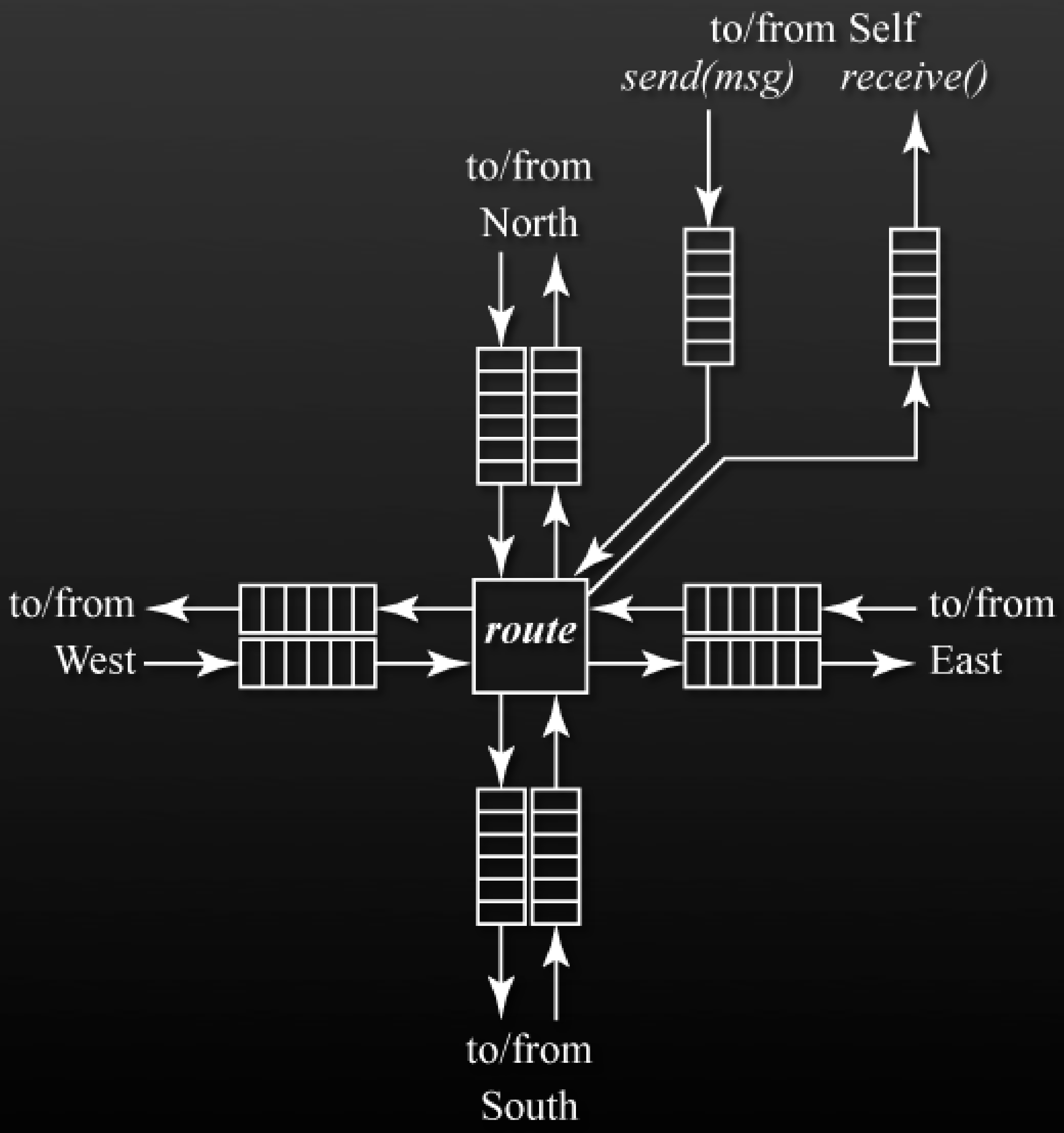
SMIPS *(scheduler)*

SMIPS *(shader)*

SMIPS *(rop)*

SMIPS *(shader)*

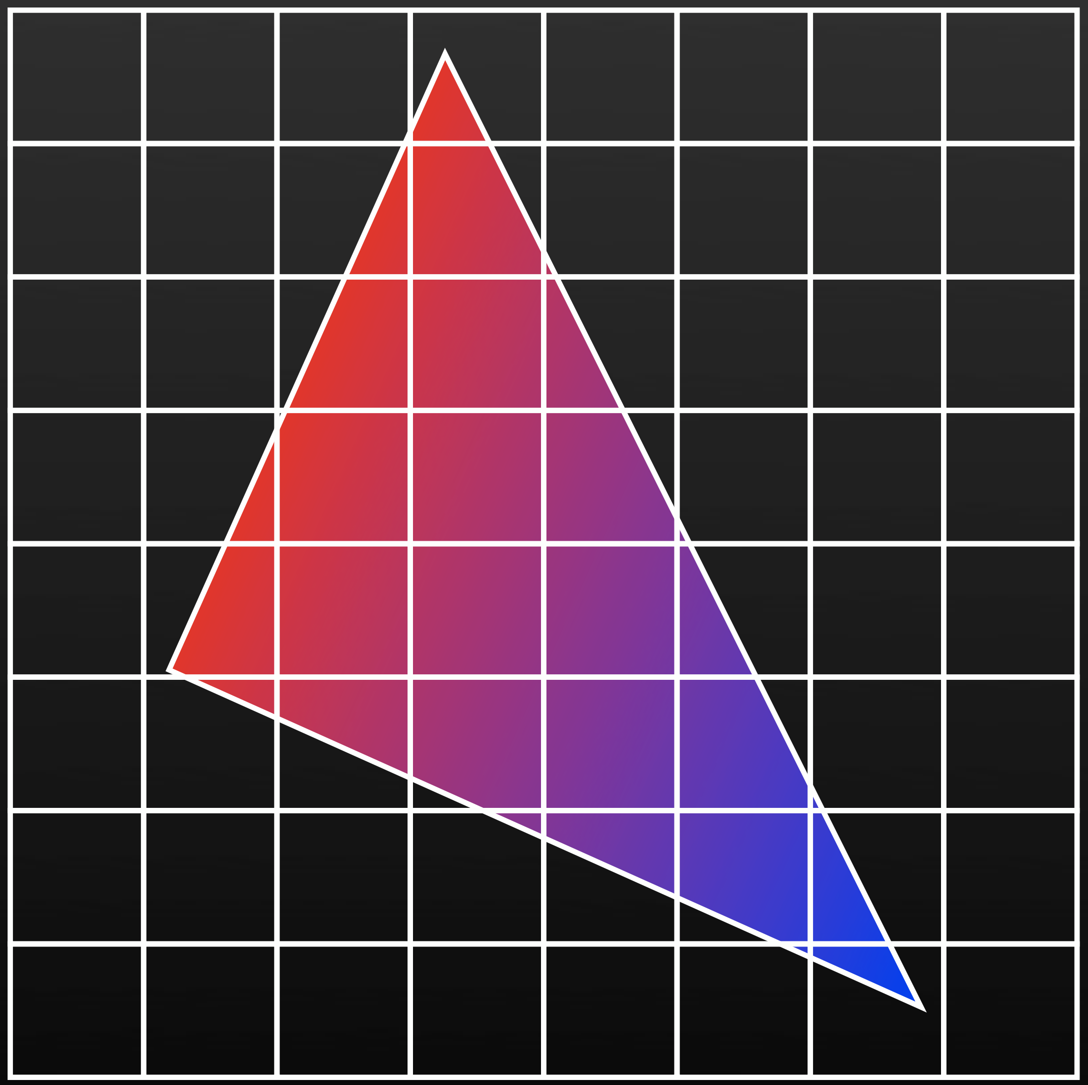
Network Router



SMIPS Network Interface

- Hijack coprocessor interface
  - MFC2/MTC2 => read/write from network
  - COP2 register reads/writes a word from the network packet
  - COP2 $r0 commits the packet
- Blocking I/O from Bluespec scheduling
  - (non-blocking I/O possible with status query)
- Completely programmable *in C*

Hardware Rasterizer



Hardware Rasterizer

- Pipelined
  - Assembly
  - Setup
  - Interpolate
- Division
  - 1 per-pixel
- Tuned fixed-point

Shaders

- User-programmable pipeline stages
- Vertex shader: transforms geometry
- Fragment shader: computes pixel color
- *Implemented in C, through toolchain*

Raster Operations

- Z-buffer: visibility
- Color buffer
- *Implemented in C*

- Hardware memory extension to scan out final image
  - Ridiculously inefficient in software

SMIPS C Support Libraries

- High-level network interface
- Fixed-point vector math, divide, sqrt
- Mini-*stdio*
  - Critical for debugging
- Flush framebuffer
  - Flush data cache

Flushing the Pipeline

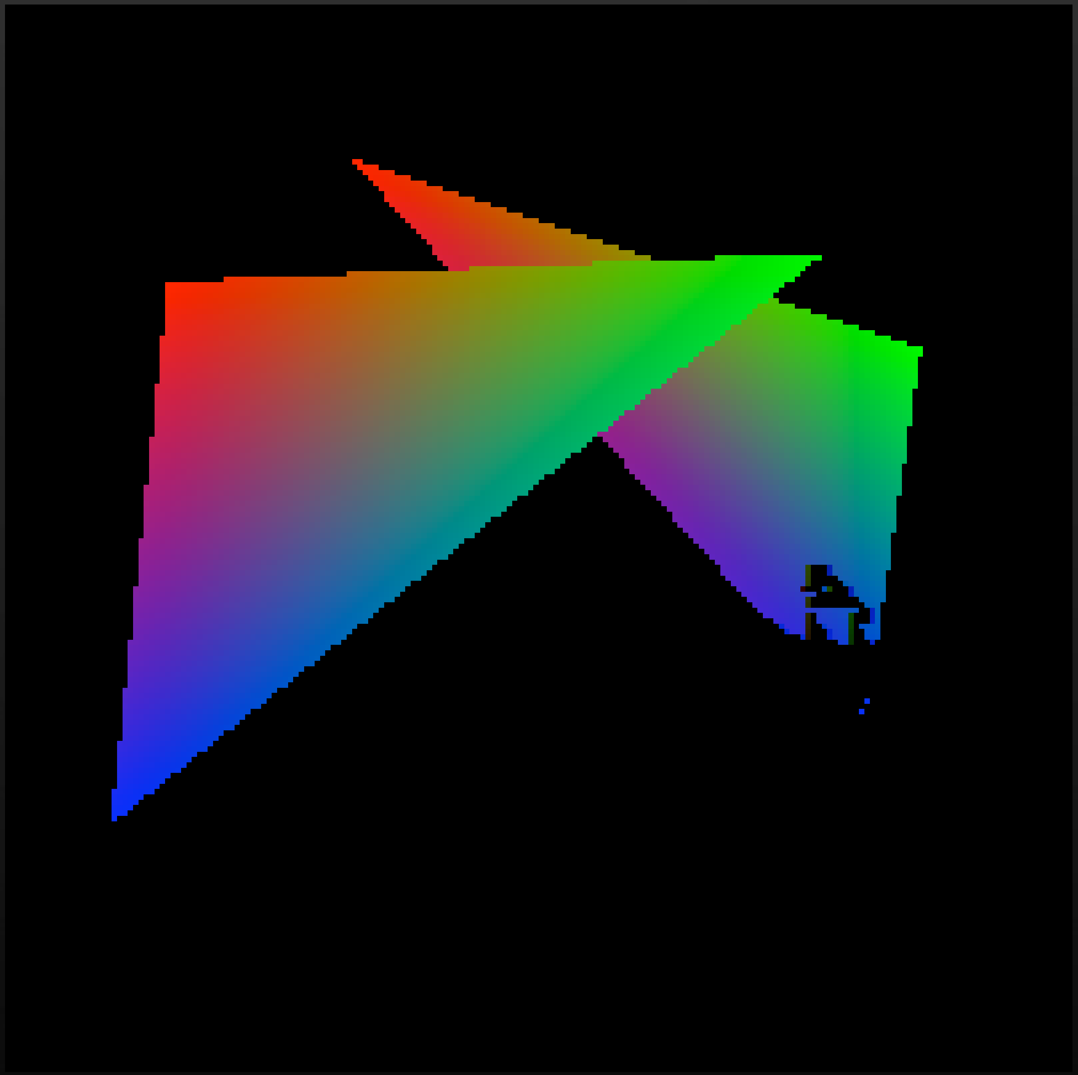
- Hard!
  - But without it, broken/partial images
- Encode end-of-stream token in data stream
- Carry token through every pipeline stage in-order
  - Between network tiles
  - Within tile pipeline stages
- Flush SMIPS ROP data cache

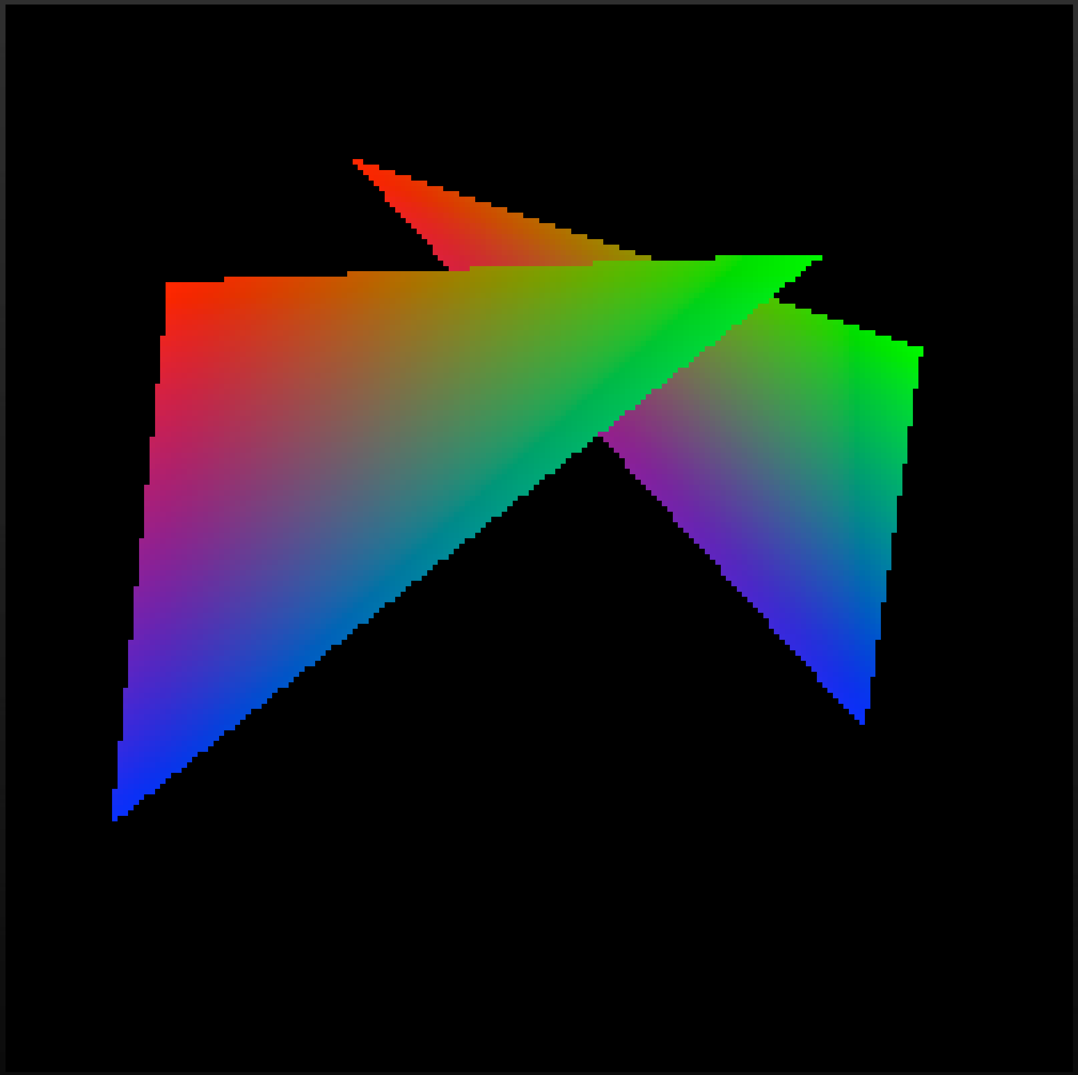
Toolchain

- makeFBO
- modelViewer
- relocateSP
- memory scanout
- txt2ppm

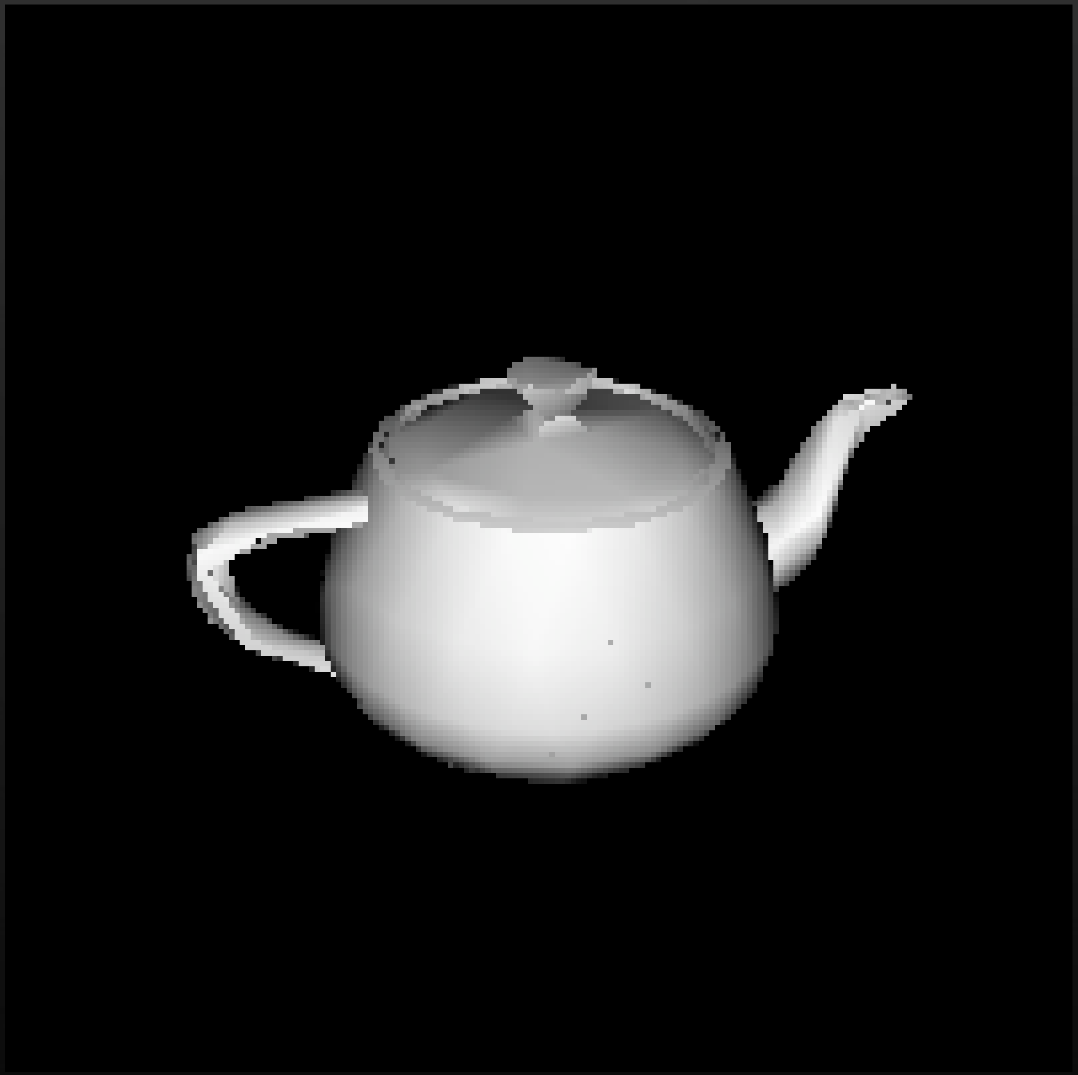
Demo

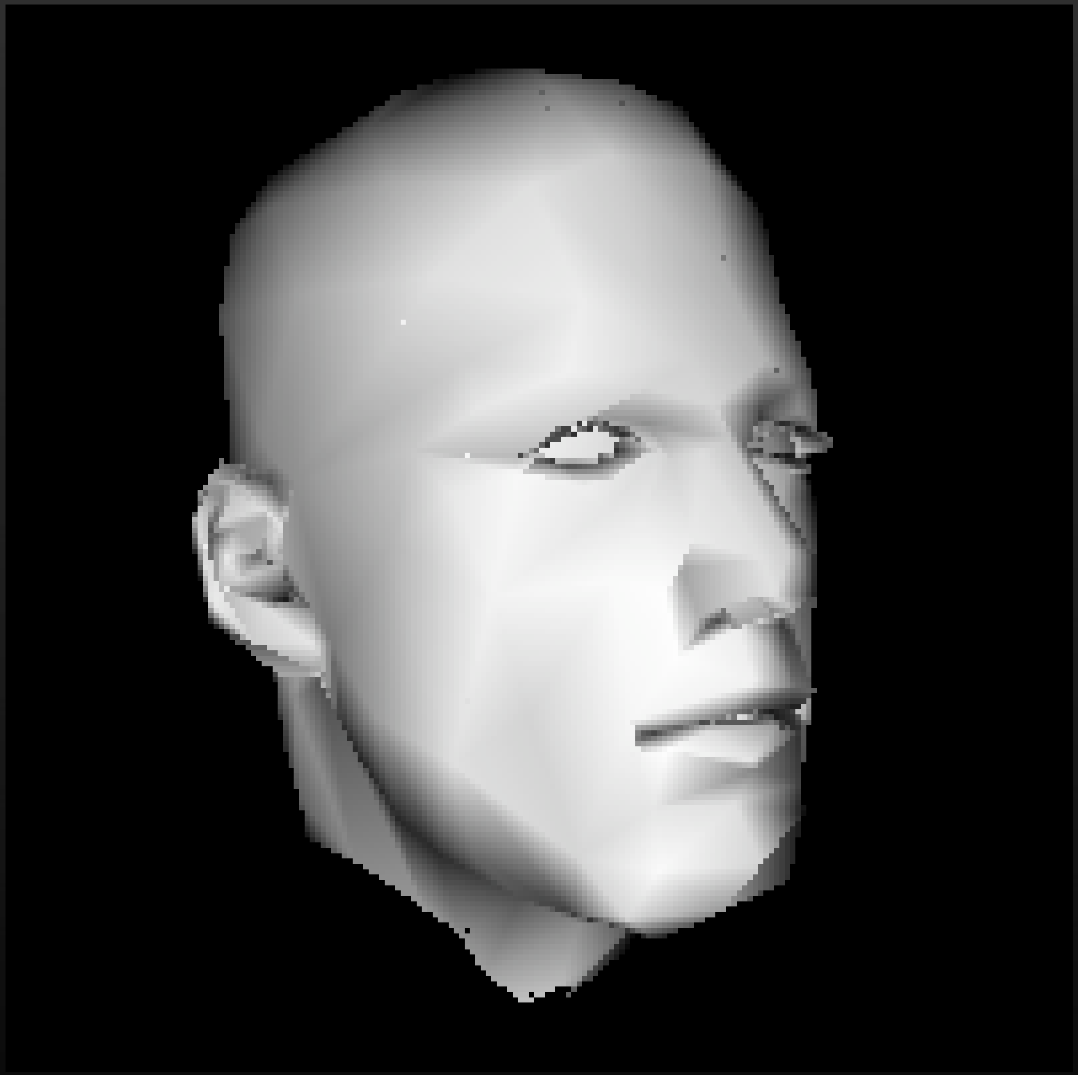
Preliminary Results





Preliminary Results





Performance

Teapot

- Configuration
  - 633 triangles
  - Vertex lighting
  - 200x200 resolution
- Performance
  - 328,927,040 cycles
  - VS never blocked
  - Rast blocked: 10,878,532 c.
  - FS blocked: 20,767,325 c.
  - ROP blocked: 23,541,712 c.
  - Simulation time: 52 mins on Core 2 Duo E6600

Head

- Configuration
  - 1,214 triangles
  - Vertex lighting
  - 200x200 resolution
- Performance
  - 633,569,28 cycles
  - VS never blocked
  - Rast blocked: 15,967,788 c.
  - FS blocked: 30,752,199 c.
  - ROP blocked: 38,441,460 c.
  - Simulation time: 120 mins on Core 2 Duo E6600

Performance Lessons

- SMIPS is *slow*
  - Narrow data-path
    - graphics workloads need wide data
    - *many* cycles to fill/read network packets
  - DIV, RSQRT *really slow* in software
- Compilation, simulation is slow for parallel hardware