CMSC 635

Graphics Hardware

A Graphics Pipeline

- Vertex
- Triangle
- Fragment
Fragment vs. Pixel

- OpenGL terminology
- Pixel = on-screen RGBA+Z
- Fragment = proto-pixel
  - RGBA + Z + Texture Coordinates + ...
  - Multiple Fragments per Pixel
    - Depth Complexity
    - Supersamples

Computation & Bandwidth

Based on:
- 100 Mtri/sec
- 256 B vertex data
- 128 B interpolated
- 68 B fragment output
- 5x depth complexity
- 16 4-byte textures
- 223 ops/vert
- 1664 ops/frag
- No caching
- No compression

- 75 GB/s Vertex 67 GFLOPS
- 13 GB/s Triangle
- 335 GB/s Texture 1.1 TFLOPS
- 45 GB/s Fragment
Pipeline

- Transform
- Shade
- Clip
- Project
- Rasterize
- Interpolate
- Texture
- Z-buffer

Vertex

Triangle

Fragment

Data Parallel

Task

Distribute

Task

Task

Task

Merge
Graphics Data Organization

Object Order
- Vertex
- Triangle
- Fragment

Screen Order

Screen Subdivision
- Tiled
- Interleaved
- Scan-Line Interleaved
- Column Interleaved
Sort First

Objects

Distribute objects by screen tile

Vertex
Triangle
Fragment

Screen

Some pixels
Some objects

Sort Middle

Objects

Distribute objects or vertices

Vertex
Vertex
Vertex

Merge & Redistribute by screen location

Triangle
Triangle
Triangle
Triangle
Fragment
Fragment
Fragment
Fragment

Screen

Some objects
Some pixels
Some objects
Sort Last

Objects

Distribute by object

Vertex

Triangle

Fragment

Vertex

Triangle

Fragment

Z-merge

Screen

Full Screen

Some objects

GPU computation

CPU

Vertex

Geometry

Fragment

Texture / Buffer

Displayed

Pixels

- NVIDIA GeForce 7800
  - ~860M vertices/sec
  - ~172M triangles/sec
  - ~6.9G fragments/sec
  - ~10.3G texels/sec
  - ~165 GFLOPS
  - ~2.4x increase/year

- 3GHz Dual-core
  Pentium 4
  - ~24.6 GFLOPS
  - ~1.5x increase/year

[Luebke, GPGPU SIGGRAPH Course, 2005; Kilgard, Real-Time Shading SIGGRAPH course, 2006]
GPU graphics processing model

NVIDIA GeForce 6

[Kilgaraff and Fernando, GPU Gems 2]
Graphics Processing Unit

Vertex → Triangle → Pixel → Pipeline

Parallel

More Parallel

More Parallel

More Pipeline

NVIDIA GeForce 6

[Kilgaraff and Fernando, GPU Gems 2]
NVIDIA Fermi

Beyond3D NVIDIA Fermi GPU and Architecture Analysis, 2010

NVIDIA Fermi GPC

NVIDIA, NVIDIA’s Next Generation CUDA Compute Architecture: Fermi, 2009
__global__ void scan(float *g_odata, float *g_idata, int n) {
    extern __shared__ float temp[]; // allocated on invocation
    int thid = threadIdx.x; // unique thread ID
    int pout = 0, pin = 1; // ping-pong input & output

    // load input into shared memory
    temp[pout*n + thid] = (thid > 0) ? g_idata[thid-1] : 0;
    __syncthreads();

    for (int offset = 1; offset < n; offset *= 2) {
        pout = 1 - pout; pin = 1 - pout; // swap double buffer indices
        if (thid >= offset) temp[pout*n+thid] += temp[pin*n+thid - offset];
        else temp[pout*n+thid] = temp[pin*n+thid];
        __syncthreads();
    }
    g_odata[thid] = temp[pout*n+thid1]; // write output
}

[Harris, “Prefix Parallel Sum (Scan) with CUDA”, NVIDIA White Paper, April 2007]