CS 194-6
Digital Systems Project Laboratory

Lecture 2 – Graphics Processors

2008-9-15

John Lazzaro
(www.cs.berkeley.edu/~lazzaro)

TA: Greg Gibeling

www-inst.eecs.berkeley.edu/~cs194-6/
Today: Graphics Processors

- **Computer Graphics.** A brief introduction to “the pipeline”.

- **Stream Processing.** Casting the graphics pipeline into hardware.

- **Unified Pipelines.** GeForce 8800, from Nvidia, introduced in 2006.

- **Larrabee.** Intel multi-core graphics architecture, SIGGRAPH 2008.
Personal computer graphics architecture

Case Study:  Mac Mini (PowerPC edition)
Personal computer graphics architecture
Mac Mini G4: System block diagram

Processor bus. How the CPU talks to everything else.

Bus controller. Low-cost Mac Mini only has 1. Most PCs have two: fast North Bridge, slow South Bridge.

CPU: PowerPC G4 (Freescale)

DVI/VGA/composite/S-video output port
FireWire 400 port
Ethernet port 10/100 Mbps
Headphone/audio line-out jack
USB 2.0 port (480 Mbps)
Modem port

CS 194-6 L2: Graphics Processors
The bus controller talks to everything else

**AGP 4X bus.**
Graphics chip.

**ATA/100 bus.**
For hard disk, DVD/CD ROM.

**PCI, ATA, AGP devices can be bus master, for Direct Memory Access (DMA).**
Disk can write RAM directly.
Mac Mini: Graphics sub-system

**AGP 4X: Hi-Speed Graphics Bus**

**Dedicated Graphics RAM**

- PowerPC G4 microprocessor (L2 cache: 512K 1:1)
- 167 MHz MaxBus
- AGP 4X bus
- DDR SDRAM DIMM slot
- 167 MHz Memory bus
- Radeon 9200 graphics IC
- 32 MB DDR RAM
- FireWire PHY
- Ethernet PHY

**ATI Radeon 9200: Graphics Processing Unit (GPU).**

- DVI/VGA/composite/S-video output port
- FireWire 400 port
- Ethernet port 10/100 Mbps

**Average selling price (ASP) for GPUs:** $30
GPU cost a significant part of total “Bill of Materials”

Parts cost in volume: $274.69
Parts + manufacturing cost: $283.37
Source: iSuppli corporation
About 12 MB/frame (24-bit pixels)
24 frames/sec: 300 MB/second
Anatomy of a “dumb” graphics card ...

**AGP 4X:** 1.1 GB/s. Can handle 24 f/s (300 MB/s) for a 2560x1600 display.

**Problem:** CPU has to compute a new pixel every 10 ns. 10 clock cycles for a 1 GHz CPU clock.

**Double Buffering:**
- CPU writes “next frame” in one buffer.
- Control logic sends “this frame” out of other buffer to display.

---

**Diagram:**
- **12 MB Frame Buffer**
- **Control Logic**
- **DVI Formatter**
- **D/A**
Q. What kind of graphics are we accelerating?
A. In 2008, interactive entertainment (3-D games). In the 1990s, 2-D acceleration (fast windowing systems, games like Pac-Man).

Q. In a multi-core world, why should we use a special processor for graphics?
A. Programmers generally use a certain coding style for graphics. We can design a processor to fit the style.

Next: An intro to 3-D graphics.
The Triangle ...

Simplest closed shape that may be defined by straight edges.

With enough triangles, you can make anything.
A cube whose faces are made up of triangles. This is a 3-D model of a cube -- model includes faces we can’t see in this view.

A sphere whose faces are made up of triangles. With enough triangles, the curvature of the sphere can be made arbitrarily smooth.
A teapot (famous object in computer graphics history). A “wire-frame” of triangles can capture the 3-D shape of complex, man-made objects.
Triangle defined by 3 vertices

By transforming \((v' = f(v))\) all vertices in a 3-D object (like the teapot), you can move it in the 3-D world, change its size, rotate it, etc.

vertex \(v_0 = (x_0, y_0, z_0)\)

vertex \(v_1 = (x_1, y_1, z_1)\)

vertex \(v_2 = (x_2, y_2, z_2)\)

If a teapot has 10,000 triangles, need to transform 30,000 vertices to move it in a 3-D scene ... per frame!
Vertex can have color, lighting info ...

If vertices colors are different, this means that a smooth gradient of color washes across triangle.

vertex $v_0 = (r_0, g_0, b_0)$
vertex $v_1 = (r_1, g_1, b_1)$
vertex $v_2 = (r_2, g_2, b_2)$

More realistic graphics models include light sources in the scene. Per-vertex information can carry information about how light hits the vertex.
We see a 2-D window into the 3-D world

Let's follow one 3-D triangle.
From 3-d triangles to screen pixels

First, **project** each 3-D triangle that might “face” the “eye” onto the **image plane**.

Then, create “**pixel fragments**” on the **boundary** of the image plane triangle.

Then, create “**pixel fragments**” to **fill in** the triangle (rasterization).

**Why “pixel fragments”?** A screen pixel color might depend on many triangles (example: a glass teapot).
Process pixel fragment to “shade” it.

Algorithmic approach: Per-pixel computational model of metal and how light reflects off of it. Move teapot and what reflects off it changes.
Process each fragment to “shade” it.

Artistic approach: Artist paints surface of teapot in Photoshop. We “map” this “texture” onto each pixel fragment during shading.

Real-world texture maps: Bike decals
Applying texture maps: Quality matters

“Good” algorithm. B and C look blurry.

“Better” algorithm. B and C are detailed.
Putting it All Together ...

**Luxo, Jr:** Short movie made by Pixar, shown at SIGGRAPH in 1986.

First Academy Award given to a computer graphics movie.
Graphics Acceleration
The graphics pipeline in hardware (2004)

3-D vertex “stream” sent by CPU

Programmable CPU
"Vertex Shader"

Programming Language/API?
DirectX, OpenGL

Programmable CPU
"Pixel Shader"

DVI/VGA/composite/S-video output port
To display

Process each vertex

Create pixels fragments

Process pixel fragments

Output Merge

Algorithms are usually hardwired

PowerPC G4 microprocessor
(L2 cache: 512K 1:1)

167 MHz MaxBus
AGP 4X bus

To display

CS 194-6 L2: Graphics Processors
Vertex Shader: A “stream processor”

Input Registers (Read Only)

Output Registers (Write Only)

Shader CPU

Shader Program Memory

Constant Registers (Read Only)

Working Registers (Read/Write)

Vertex “stream” from CPU

From CPU: changes slowly (per frame, per object)

Only one vertex at a time placed in input registers.

Vertex “stream” ready for 3-D to 2-D conversion

Short (ex: 128 instr) straight-line code. Same code runs on every vertex.

Shader creates one vertex out for each vertex in.
Optimized instructions and data formats

128-bit registers, holding four 32-bit floats.

Typical use: (x,y,z,w) representation of a point in 3-D space.

From CPU

Input Registers

x y z w

Shader CPU

rsq dest src

dest.{x,y,z,w} = 1.0/sqrt(abs(src.w)). If src.w=0, dest ∞.

The 1/sqrt() function is often used in graphics.

Shader Program Memory

Typical instruction:

dest.{x,y,z,w} = 1.0/sqrt(abs(src.w)). If src.w=0, dest ∞.

To 3-D/2-D
Easy to parallelize: Vertices independent

From CPU

Caveat:
Care might be needed when merging streams.

Why?
3-D to 2-D may expect triangle vertices in order in the stream.

Shader CPUs easy to multithread.

Input Registers
x y z w

Shader CPU

Output Registers
x y z w

To 3-D/2-D
Pixel shader specializations ...

Texture maps (look-up tables) play a key role.

Pixel shader needs fast access to the map of Europe on teapot (via graphics card RAM).

“Pixel Shader” CPU →

Process each vertex

Create pixels fragments

Process pixel fragments

Output Merge

DVI/VGA/composite/S-video output port

PowerPC G4 microprocessor (L2 cache: 512K 1:1)

167 MHz MaxBus

AGP 4X bus
Pixel Shader: Stream processor + Memory

Pixel fragment stream from rasterizer

- Input Registers (Read Only)
  - Only one fragment at a time placed in input registers.
- Texture Registers
  - Indices into texture maps.
- Shader CPU
  - Engine does interpolation.
  - Shader creates one fragment out for each fragment in.
- Registers (Read/Write)
  - Register R0 is pixel fragment, ready for output merge
- Texture Engine
- Memory System
- Constant Registers (Read Only)
  - From CPU: changes slowly (per frame, per object)

Indices into texture maps.
Example Design: Nvidia GeForce 7900

278 Million Transistors, 650 MHz clock, 90 nm process

3-D to 2-D

Texture Cache

Vertex Shaders: 8

Pixel Shaders: 24

Output Merge Units

CS 194-6 L2: Graphics Processors
Basic idea: Replace specialized logic (vertex shader, pixel shader, hardwired algorithms) with many copies of one unified CPU design.

Unified Architectures

Consequence: You no longer “see” the graphics pipeline when you look at the architecture block diagram.

Designed for: DirectX 10 (Microsoft Vista), and new non-graphics markets for GPUs.
DirectX 10 (Vista): Towards Shader Unity

Earlier APIs: Pixel and Vertex CPUs very different ...

<table>
<thead>
<tr>
<th>Feature</th>
<th>1.1</th>
<th>2.0</th>
<th>3.0</th>
<th>4.0</th>
<th>4.0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2001</td>
<td>2002</td>
<td>2004</td>
<td>2006</td>
<td></td>
</tr>
<tr>
<td>instruction slots</td>
<td>128</td>
<td>256</td>
<td>≥512</td>
<td>≥64K</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4+8²</td>
<td>32+64⁴</td>
<td>≥512</td>
<td></td>
<td></td>
</tr>
<tr>
<td>constant registers</td>
<td>≥96</td>
<td>≥256</td>
<td>≥256</td>
<td>16x4096</td>
<td></td>
</tr>
<tr>
<td>tmp registers</td>
<td>12</td>
<td>12</td>
<td>32</td>
<td>4096</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>12</td>
<td>32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>input registers</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>4+2²</td>
<td>8+2⁴</td>
<td>10</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>render targets</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>samplers</td>
<td>8</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>textures</td>
<td>8</td>
<td>16</td>
<td>4</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>16</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2D tex size</td>
<td></td>
<td>2Kx2K</td>
<td>8Kx8K</td>
<td></td>
<td></td>
</tr>
<tr>
<td>integer ops</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>load op</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>sample offsets</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>transcendental ops</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>derivative op</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>flow control</td>
<td></td>
<td>static</td>
<td>stat/dyn</td>
<td>dynamic</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Shader model feature comparison summary.
DirectX 10: New Pipeline Features ...

Geometry Shader: Lets a shader program create new triangles.

Also: Shader CPUs are more like RISC machines in many ways.

Stream Output: Lets vertex stream recirculate through shaders many times ...
(and also, back to CPU)
Why?  Particle systems ...
Why? Fractal images ...
NVidia 8800: Unified GPU, announced Fall 2006

Thread processor sets shader type of each CPU

128 Shader CPUs

1.35 GHz Shader CPU Clock, 575 MHz core clock
Graphics-centric functionality...

Texture engine and memory system

3-D to 2-D (vertex to pixel)

Pixel fragment output merge
Can be reconfigured with graphics logic hidden ... 


3 TeraFlops Peak Performance Ships with a C compiler.

1000s of active threads

Texture system set up to look like a conventional memory system (768MB GDDR3, 86 GB/s)
Chip Facts
90nm process
681M Transistors
80 die/wafer (pre-testing)

Design Facts
4 year design cycle
$400 Million design budget
600 person-years: 10 people at start, 300 at peak

A big die. Many chips will not work (low yield). Low profits.
GeForce 8800 GTX Card: $599 List Price

PCI-Express 16X Card - 2 Aux Power Plugs!

185 Watts Thermal Design Point (TDP) -- TDP is a “real-world” maximum power spec.
Some products are “loss-leaders”

Breakthrough product creates “free” publicity you can’t buy.

(1) Hope: when chip “shrinks” to 65nm fab process, die will be smaller, yields will improve, profits will rise.

(2) Simpler versions of the design will be made to create an entire product family, some very profitable. “We tape out a chip a month”, NVidia CEO quote.
And it happened! 2008 nVidia products

<table>
<thead>
<tr>
<th></th>
<th>GTX 280</th>
<th>GTX 260</th>
<th>9800 GX2</th>
<th>9800 GTX+</th>
<th>9800 GTX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stream Processors</td>
<td>240</td>
<td>192</td>
<td>256</td>
<td>128</td>
<td>128</td>
</tr>
<tr>
<td>Texture Address / Filtering</td>
<td>80 / 80</td>
<td>64 / 64</td>
<td>128 / 128</td>
<td>64 / 64</td>
<td>64 / 64</td>
</tr>
<tr>
<td>ROPs</td>
<td>32</td>
<td>28</td>
<td>32</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>Core Clock</td>
<td>602MHz</td>
<td>576MHz</td>
<td>600MHz</td>
<td>738MHz</td>
<td>675MHz</td>
</tr>
<tr>
<td>Shader Clock</td>
<td>1296MHz</td>
<td>1242MHz</td>
<td>1500MHz</td>
<td>1836MHz</td>
<td>1690MHz</td>
</tr>
<tr>
<td>Memory Clock</td>
<td>1107MHz</td>
<td>999MHz</td>
<td>1000MHz</td>
<td>1100MHz</td>
<td>1100MHz</td>
</tr>
<tr>
<td>Memory Bus Width</td>
<td>512-bit</td>
<td>448-bit</td>
<td>256-bit x 2</td>
<td>256-bit</td>
<td>256-bit</td>
</tr>
<tr>
<td>Frame Buffer</td>
<td>1GB</td>
<td>896MB</td>
<td>1GB</td>
<td>512MB</td>
<td>512MB</td>
</tr>
<tr>
<td>Transistor Count</td>
<td>1.4B</td>
<td>1.4B</td>
<td>1.5B</td>
<td>754M</td>
<td>754M</td>
</tr>
<tr>
<td>Manufacturing Process</td>
<td>TSMC 65nm</td>
<td>TSMC 65nm</td>
<td>TSMC 65nm</td>
<td>TSMC 55nm</td>
<td>TSMC 65nm</td>
</tr>
<tr>
<td>Price Point</td>
<td>$650</td>
<td>$400</td>
<td>$500</td>
<td>$229</td>
<td>$199</td>
</tr>
</tbody>
</table>

**GTX 280**

Price similar to 8800, stream CPU count > 2X ...

**9800 GTX**

Specs similar to 8800, card sells for $199.
Face was “scanned” to create a vertex model. 8800 GTX was used to do skin, eye, lips and hair rendering.
History and Graphics Processors

Create standard model from common practice: Wire-frame geometry, triangle rasterization, pixel shading.

Put model in hardware: Block diagram of chip matches computer graphics math.

Evolve to be programmable: At some point, it becomes hard to see the math in the block diagram.

“Wheel of reincarnation” -- Hardwired graphics hardware evolves to look like general-purpose CPU. EECS visitor Ivan Sutherland co-wrote a paper on this topic in 1968!
Intel @ SIGGRAPH 2008: Larrabee

Larrabee: A Many-Core x86 Architecture for Visual Computing

Larry Seiler¹, Doug Carman¹, Eric Sprangle¹, Tom Forsyth¹, Michael Atrash²,
Pradeep Dubey¹, Stephen Junkins¹, Adam Lake¹, Jeremy Sugarman³,
Robert Cavin¹, Roger Espasa¹, Ed Grochowski¹, Toni Juan¹, and Pat Hanrahan³

A GPU that looks a lot like a many-core CPU. However, cores are specialized for graphics ...
x86-ISA Scalar Unit based on the 1992 original Pentium.

In-order static pipeline, dual issue.

16-lane Vector Unit takes 2/3 of CPU core chip area.

Handles the graphics “heavy lifting”.

Larrabee CPU core microarchitecture
Vector Unit specializations ...

Load/store instruction variants may specify type conversion.

This supports a toll-free way to store low resolution data in packed form, while letting the CPU compute on the data in a high-resolution format.
GPU project ideas ...

Idea #1: Implement part of an nVidia or AMD architecture on the Virtex-5.

Idea #2: Translate shader programs into hardwired logic on the Virtex-5.

Idea #3: Special-purpose GPU for particle systems, fractal synthesis, etc.

(1) The LX 110T has 64 DSP ALUs -- enough to get significant parallelism on a graphics task.

(2) Don’t need to do entire algorithm on the Virtex-5 -- precompute parts in advance (“cooking show” solution).
Recall: “dumb” graphics card...

**AGP 4X: 1.1 GB/s. Can handle 24 f/s (300 MB/s) for a 2560x1600 display.**

**Double Buffering:** CPU writes “next frame” in one buffer.

Control logic sends “this frame” out of other buffer to display.

**12 MB Frame Buffer**

**Control Logic**

**DVI Formatter**

**D/A**

---

**PowerPC G4 microprocessor** (L2 cache: 512K 1:1)

167 MHz MaxBus

AGP 4X bus

---

DVI/VGA/composite/S-video output port
On ML 505: The CH7301C DVI chip

CH7301C DVI Transmitter Device

CH7301C

Clock Driver

DVI PLL

DVI Encode

DVI Serialize

DVI Driver

Sync Decode
Color space conversion

Three 10-bit DAC's

Serial Port Control

Codec, Clock, and Data Latch, Demux

H.V,DE Latch

VREF

XCLK, XCLK*

D[11:0]

H.V,DE

VREF

HPDET

GPIO[1:0]

AS

SPC

SPD

RESET*

H_SYNC

V_SYNC

CS 194-6 L2: Graphics Processors

UC Regents Fall 2008 © UCB
Next Monday’s Lecture:

Single-cycle CPU design