Lecture 21: Course Review

Kayvon Fatahalian
CMU 15-869: Graphics and Imaging Architectures (Fall 2011)
Agenda

- Tips on final project presentations
- Course review
Final project presentations

- **Friday December 16th (yes, it’s Black Friday)**
  - GHC 4102 (not this room)
  - 5:30 - 8:30PM

- **We have 11 projects**

- **10 minute presentation (+3 minutes of questions)**
  - Quality of presentation will factor into final project grade
  - For team projects, each team member should talk 1/2 the time

- **Final written report:**
  - Sunday December 18th, 11:59PM
Presentation Tips
Tip #1: it’s not about you

- The audience doesn’t care about everything you did

- They only care what you found out, that they ought to know

- A talk is a service (a responsibility)
  - Ask yourself: What can I say about my work in the allotted time that is the most interesting for my audience?
  - Think about the man/woman-hours wasted by a bad talk
Tip #2: know your audience

- What should be reviewed as background?

- Consider your project:
  - What should the rest of the class know based on the lectures?
  - What does your project dig into that you don’t expect everyone to know?
Tip #3: state the problem clearly

- What is the problem you are trying to solve?

- E.g.,
  - This is all about minimizing latency
  - This is problem of reducing bandwidth
  - I am relaxing assumptions that are hurting performance
  - I am creating instrumentation to understand a certain aspect of a workload
  - There are two solutions with different strengths/weaknesses, I want a solution that provides the best of both worlds
Tip #4: text is a crutch **

- Common error: add text to slide since it’s a point you want to say (don’t want to forget it)
  - This is what speakers notes are for

- Slides should primarily be figures

- Slides augment what you say
  - They are not a text version of what you are saying
  - You want people to be listening to you, not reading ahead in your slides

- I remove text as I edit my slides as I prep for a talk

** This is a do as I say, not as I do slide
(a good example of visual slides is shown in class)
Tip #5: explain every figure or graph

1. Overview
   - This figures shows the effect of rasterizing two triangles

2. Part-by-part explanation:
   - Pixels are the boxes, they are colored according to the number of fragments generated ...
   - The sample points are given by the dots

3. Point: as you can see pixel ...
Tip #5: explain every figure or graph

1. In this graph, the X-axis is ______.
2. The Y-axis is ______.
3. If you look at the left side ...
4. So the trend that you see means ...

Common error: only explaining the result
Tip #6: prioritize clarity over coverage!

- Aim to have your entire talk understood

- As a result, every talk can only really have a few points

- If you think the audience won’t get it, or you’ll have to rush through it, then take it out
  - That’s what your final writeup is for!
Tip #7: transition sentences

- Good voice over when transitioning between slides can really make a talk flow

- I use speaker’s notes to remind myself of good transition sentences

- Slide N has a note for what to say as I am transitioning to slide N+1
  - e.g., “and if you make assumption X, what you get is ...”
Tip #8: practice!

- Rehearsing your presentation will pay off!
  - Important for determining how you stand on time (10 minutes is short!)
  - In general, your real talk will be a little faster (nerves make you speed up)
  - These are short talks, so they are easy to practice

- I often do a final practice 1-2 hours before the presentation
  - To get in rhythm, like an athlete’s pre-game warm up
  - I already know the talk well at this point
Tip #9: three aspects of describing a system

1. What are the components or entities (nouns)?
   - Major components (processors, memories, interconnects, pipeline stages)
   - Major entities (e.g., vertices, triangles, pixels, shots, frames)

2. What is the state associated with the nouns?

3. What are the operations that can be performed?
   - State manipulation operations
   - Operations that create or consume entities
Tip #10: do your analysis

Many of your projects have an analysis component (the most important component of certain projects)

1. Consider the low/high watermarks (best/worst case)
   - What if a particular component of an algorithm was infinitely fast?
   - If your algorithm was perfect, what is the best it could achieve?

2. Consider all the possible “attacks”:
   - If I ask you a question about a graph can you explain it?
Course Review
The graphics pipeline

- **Vertices**
  - Vertex Generation
  - Vertex Processing

- **Primitives**
  - Primitive Generation
  - Primitive Processing

- **Fragments**
  - Rasterization (Fragment Generation)
  - Fragment Processing

- **Pixels**
  - Frame-Buffer Ops

**Memory**

- Uniform data
- Texture buffers

**Frame Buffer**
Homogeneous collection of throughput-optimized programmable processing cores
Augmented by fixed-function logic
Throughput processing
Summary: three key ideas for high-throughput execution

1. Use many “slimmed down cores,” run them in parallel

2. Pack cores full of ALUs (by sharing instruction stream overhead across groups of fragments)
   - Option 1: Explicit SIMD vector instructions
   - Option 2: Implicit sharing managed by hardware

3. Avoid latency stalls by interleaving execution of many groups of fragments
   - When one group stalls, work on another group
GPU processing core

NVIDIA GeForce GTX 480 “core”

- The core contains 32 functional units (2 sets of 16 share instruction stream)
- Two groups of 32 fragments (“warps”) are selected every other clock (decode, fetch, and execute two instruction streams in parallel)
- Up to 48 groups are interleaved (switch to new group on stall)

Source: Fermi Compute Architecture Whitepaper
CUDA Programming Guide 3.1, Appendix G
Thought experiment

Task: element-wise multiplication of two vectors A and B

1. Load input $A[i]$
2. Load input $B[i]$
3. Load input $C[i]$
4. Compute $A[i] \times B[i] + C[i]$
5. Store result into $D[i]$

Four memory operations (16 bytes) for every MUL-ADD
Radeon HD 5870 can do 1600 MUL-ADDs per clock
Need \(~20\ TB/sec\) of bandwidth to keep functional units busy

Less than 1\% efficiency... but 6x faster than CPU!
Alternative Rendering Algorithms
Deferred shading pipeline

Fragment shader outputs surface properties (e.g., position, normal, material diffuse color, specular color)

Traditional pipeline does not output RGB image. Output is a 2D buffer representing information about the surface geometry visible at each pixel (a.k.a. “g-buffer”)

After all geometry has been rendered, shader is executed for each sample in the G-buffer, yielding RGB values

( shading is deferred until all geometry processing -- including all occlusion computations -- is complete)
$\text{G-buffer} = \text{geometry buffer}$

Image Credit: J. Klint, “Deferred Rendering in Leadworks Engine”

Kayvon Fatahalian, Graphics and Imaging Architectures (CMU 15-869, Fall 2011)
Motivation: why deferred shading?

- Shade only surface fragments that are visible
- Forward rendering is inefficient when shading small triangles (quad-fragment granularity)
- Increasing complexity of lighting computations
  - Growing interest in scaling scenes to hundreds of light source
Blue = active ray after node box test

r6 does not pass node F box test due to closest-so-far check
Packet tracing best practices

- Use large packets for higher levels of BVH  
  - Ray coherence always high at the top of the tree  

- Switch to single ray (intra-ray SIMD) when packet utilization drops below threshold  
  - For wide SIMD machine, a single branching-factor 4 BVH works well for both packet and single ray traversal  

- Can use packet reordering to postpone time of switch  
  - Reordering allows packets to provide benefit deeper into tree  

[Wald et al. 2007]
[Benthin et al. 2011]
[Boulos et al. 2008]
Image Processing Pipeline
Image processing pipeline

- The signal a camera captures is very different than the image that is ultimately produced for the user.

- Understanding of human perception is fundamental to many operations/optimizations in the image processing pipeline.
# Simplified image processing pipeline

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>Sample Pixel Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct for sensor bias (using measurements of optically black pixels)</td>
<td>▪ Correct pixel defects</td>
<td>12-bits per pixel</td>
</tr>
<tr>
<td></td>
<td>▪ Vignetting compensation</td>
<td>1 intensity per pixel</td>
</tr>
<tr>
<td></td>
<td>▪ Dark frame subtract (optional)</td>
<td>Pixel values linear in energy</td>
</tr>
<tr>
<td></td>
<td>▪ White balance</td>
<td></td>
</tr>
<tr>
<td></td>
<td>▪ Demosaic</td>
<td></td>
</tr>
<tr>
<td></td>
<td>▪ Denoise / sharpen, etc.</td>
<td>3x12-bits per pixel</td>
</tr>
<tr>
<td></td>
<td>▪ Color Space Conversion</td>
<td>RGB intensity per pixel</td>
</tr>
<tr>
<td></td>
<td>▪ Gamma Correction</td>
<td>Pixel values linear in energy</td>
</tr>
<tr>
<td></td>
<td>▪ Color Space Conversion (Y’CbCr)</td>
<td>3x8-bits per pixel</td>
</tr>
<tr>
<td></td>
<td>▪ 4:4:4 to 4:2:2 chroma subsampling</td>
<td>(until 4:2:2 subsampling)</td>
</tr>
<tr>
<td></td>
<td>▪ JPEG compress</td>
<td>Pixel values perceptually linear</td>
</tr>
</tbody>
</table>
Light field inside a camera

- Lens aperture: (U,V)
- Scene focal plane
- Sensor plane: (X,Y)
- Pixel P1
- Pixel P2

Ray space plot

Kayvon Fatahalian, Graphics and Imaging Architectures (CMU 15-869, Fall 2011)
New types of cameras
Light field camera: each sensor pixel records a beam of light
Infrared image of Kinect illuminant output

Credit: www.futurepicture.org
Infrared image of Kinect illuminant output

Credit: www.futurepicture.org

Kayvon Fatahalian, Graphics and Imaging Architectures (CMU 15-869, Fall 2011)
Structured light depth camera

One light source emitting known beam, one camera
If the scene is at reference plane, image recorded by camera is known

Single spot illuminant is inefficient!
(must to “scan” scene with spot to get depth)

\[ z = \frac{bf}{d} + z_{\text{ref}} \]
Mobile system on a chip

Texas Instruments OMAP 5 (2012)

Think of a modern mobile system-on-chip as a Swiss Army Knife of computing. Software (programmer? compiler? runtime?) picks the right tool(s) for the job. Heterogeneity is very likely the future at many scales of computing!
Class themes

- Visual computing applications (graphics, image/video processing, vision) are driving the design of many computing architectures.

- Big difference between FAST and EFFICIENT
  - Graphics systems are very efficient, they have to be
  - Highly optimized algorithms and heterogeneous HW implementations

- Good system design: hardware implementation, algorithms, and abstractions all designed with each other in mind

- Go understand your workloads!
  - Where is the parallelism, communication, locality
Thank you!