Lecture 27:

Course Wrap Up & Project Presentation Tips

Parallel Computer Architecture and Programming
CMU 15-418/15-618, Spring 2014

Today

- Exam 2 discussion

- Parallelism competition hints/guidelines

- Wrap up: a few final comments about class topics

Announcements

- Please fill out course evaluations

- Parallelism competition
  - Friday May 9th, 8:30-11:30am
  - Porter Hall 100
  - Project reports due at 11:59pm that evening, no late days

Exam 2 discussion (no slides)

Presentation format

- Each group has six minutes to talk
  - Plus 1-2 minutes for questions from judges
- Presentation format is up to you: slides, live demo, etc.
  - Start with name(s) of students, title of project on a title slide
  - Both students in a 2-person group should speak
- Present off your own laptop, or make arrangements with staff

Project presentation tips

Benefit TO YOU of a good (clear) talk

- Non-linear increase in <u>impact</u> of work
  - Others are more likely to remember and build upon your work
  - Others are more likely to come up to you after the talk

- Clarity is highly prized: the audience remembers you
  - "Hey man, that was a great talk… are you looking for a job anytime soon?"

Your #1 priority should be to be clear, rather than be comprehensive

(your project writeup is the place for completeness)

Everything you say should be understandable by someone in this class. If you don't think the audience will understand, leave it out (or change). (spend the time saying something we will understand)

This will be much harder than it seems.

Here are some tips to help.

1.

Achieving clarity tip 1:

Put yourself in your audience's shoes

Consider your audience

- Everyone in the audience knows about parallel programming
  - CS terminology/concepts need not defined

- Most of the audience knows little-to-nothing about the specific application domain or problem you are trying to solve
  - Application-specific terminology should be defined or avoided

- The judges are trying to figure out the "most interesting" thing that you found out or did (your job is to define most interesting for them)

2.

Pick a focus.

Figure out what you want to say. Then say it.

(and nothing more: “every sentence matters”)

Pick a focus

- In this class, different projects should stress different results
- Some projects may wish to show a flashy demo and describe how it works (proof by "it works")
- Other projects may wish to show a sequence of graphs (path of progressive optimization) and describe the optimization that took system from performance A to B to C
- Other projects may wish to clearly contrast parallel CPU vs. parallel GPU performance for a workload

Your job is <u>not</u> to explain what you did, but to explain what you think we should know

3.

Achieving clarity tip 2:

The audience prefers not to think (much)

The audience has a finite supply of mental effort

- **The audience does not want to burn mental effort about things <u>you know</u> and can just tell them.**
  - **They want to be led by hand through the major steps of your story**
  - **They do not want to interpret any of your figures or graphs, they want to be told how to interpret them (what to look for).**

- **The audience <u>does</u> want to spend their energy thinking about:**
  - **Potential problems with what you did (did you consider all edge cases?)**
  - **Implications of your approach**
  - **Connections to their work**

4.

Set up the problem.
Establish inputs, outputs, and constraints
(goals and assumptions)

Establish goals and assumptions early

- Given these inputs, we wish to generate these outputs
- We are working under the following constraints
  - Example: the outputs should have these properties
  - Example: the algorithm…
    - Should be real-time
    - Must interoperate with this existing system

Basic setup

- What is the computation performed (or system built)?
  - What are the inputs, and the outputs?

- Why does this problem stand to benefit from optimization?
  - "Real-time performance could be achieved"
  - "Researchers could run many more trials, changing how science is done"
  - "It is 90% of the execution time in this particular system"

- Why is it hard? (What made your project interesting?)
  - What turned out to be the hardest part of the problem?
  - This may involve describing a few key characteristics of the workload (e.g., overcoming divergence, increasing arithmetic intensity)

5.

Surprises* are almost always bad:

Say <u>where</u> you are going and why you must go there before you say <u>what</u> you did.

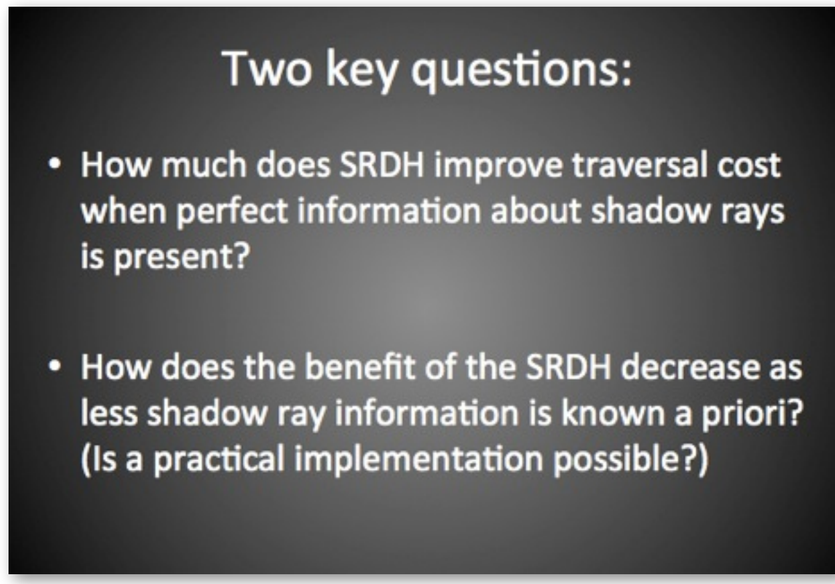
* I am referring to surprises in talk narrative and/or exposition. A surprising result is great.

Give the <u>why</u> before the <u>what</u>

- **<u>Why</u> provides the listener context for…**
  - Compartmentalizing: assessing how hard they should pay attention (is this a critical idea, or just an implementation detail?). Especially useful if they are getting lost.
  - Understanding how parts of the talk relate ("Why is the speaker now introducing a new optimization approach?")

- **In the algorithm description:**
  - "We need to first establish some terminology"
  - "Even given X, the problem we still haven't solved is…"
  - "Now that we have defined a cost metric we need a method to minimize it…"

- **In the results:**
  - Speaker: "Key questions to ask about our approach are…"
  - Listener: "Thanks! I agree, those are good questions. Let's see what the results say!"



6.

How to describe a system

How to describe a system

- Start with the nouns (the key boxes in a diagram)
  - Major components (processors, memories, interconnects, etc.)
  - Major entities (particles, neighbor lists, pixels, pixel tiles, features, etc.)
  - What is state in the system?

- Then describe the verbs
  - Operations that can be performed on the state (update particle positions, compute gradient of pixels, traverse graph, etc.)
  - Operations produce, consume, or transform entities

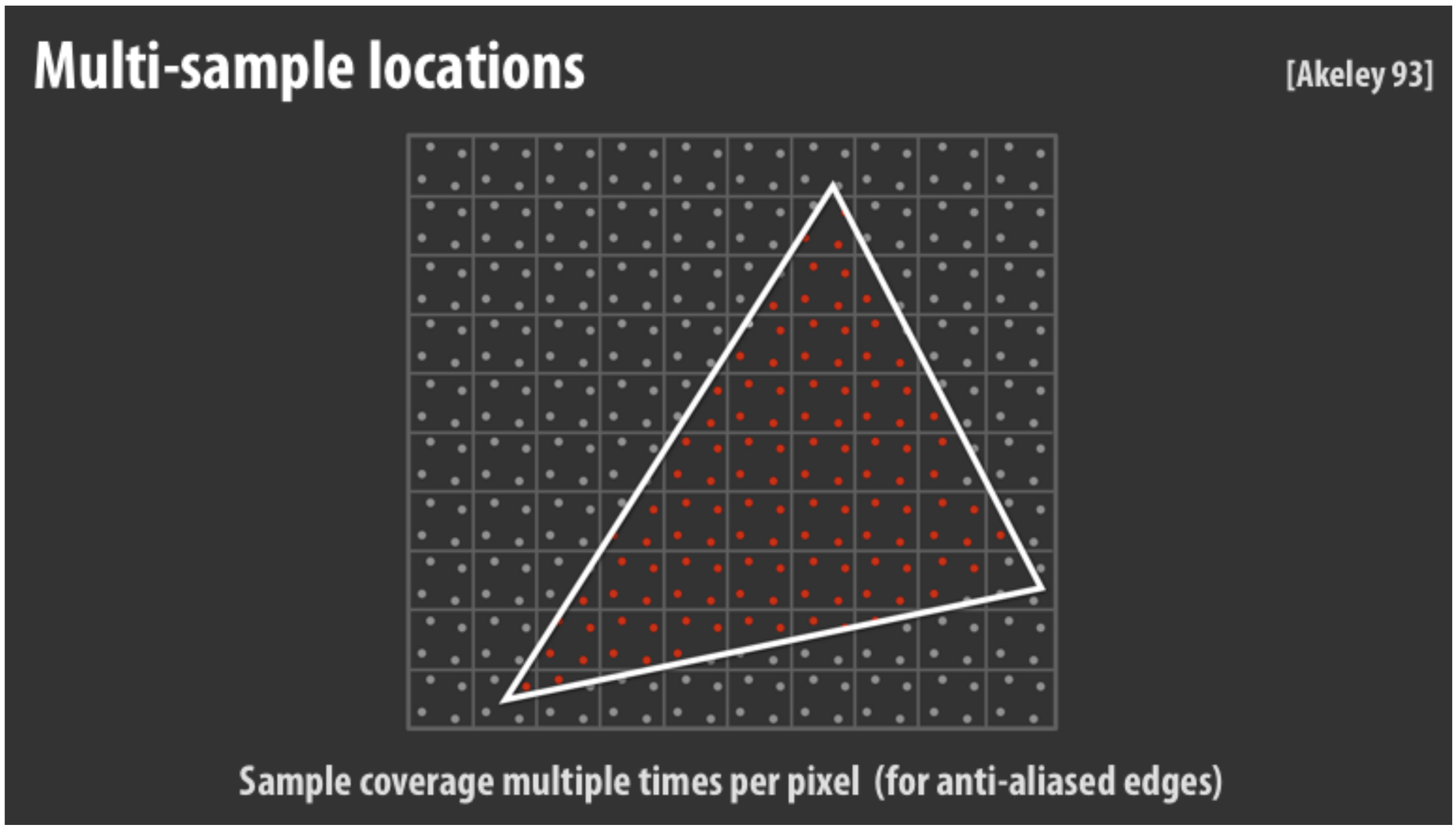
7.

Always, always, always explain any figure or graph

(remember, the audience does not want to think)

Explain every figure

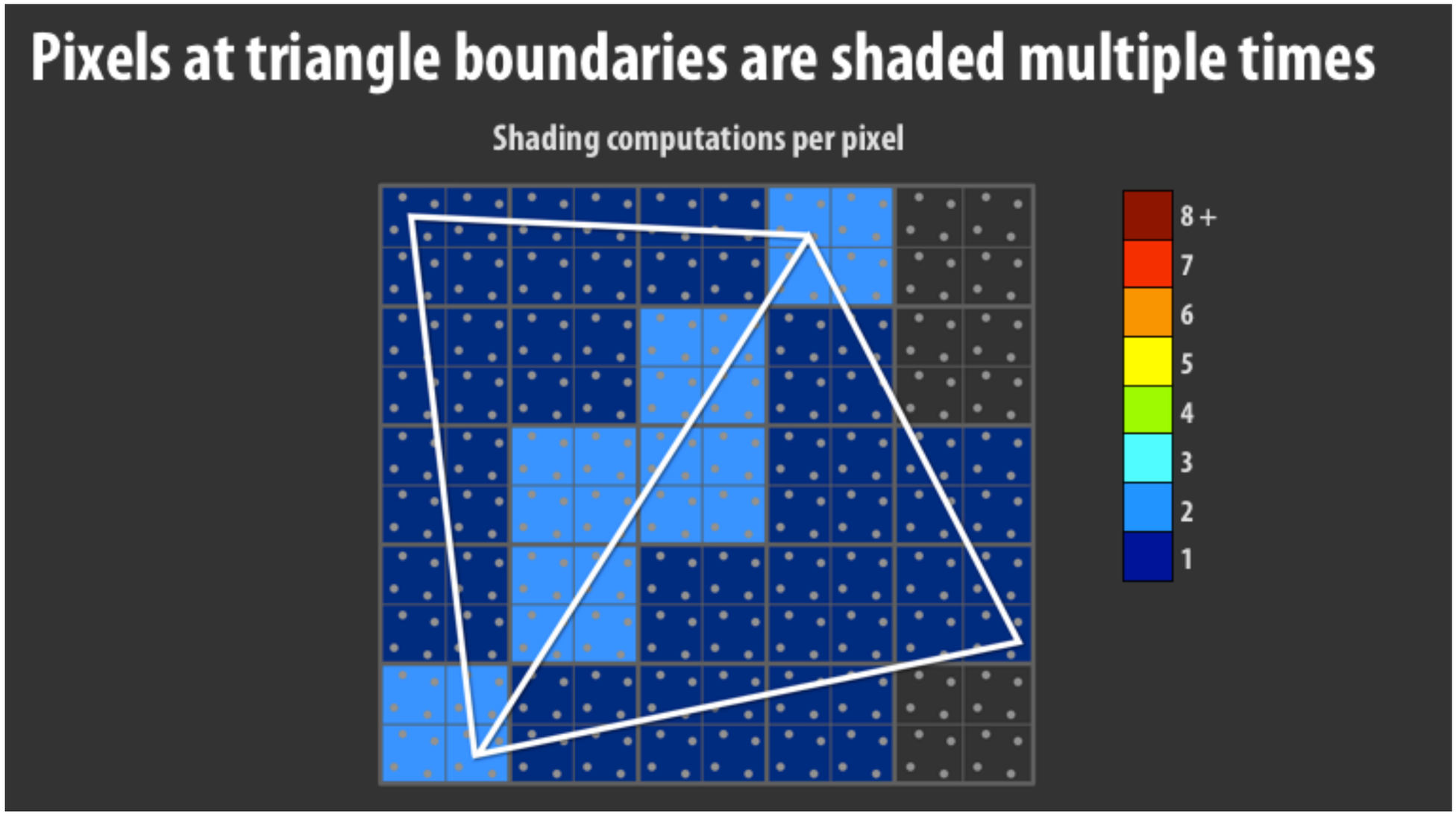
- Explain every visual element used in the figure (don't make the audience decode a figure)
- Refer to highlight colors explicitly (explain why the visual element is highlighted)



Example voice over: "Here I'm showing you a pixel grid, a projected triangle, and the location of four sample points at each pixel. Sample points falling within the triangle are colored red.

Explain every figure

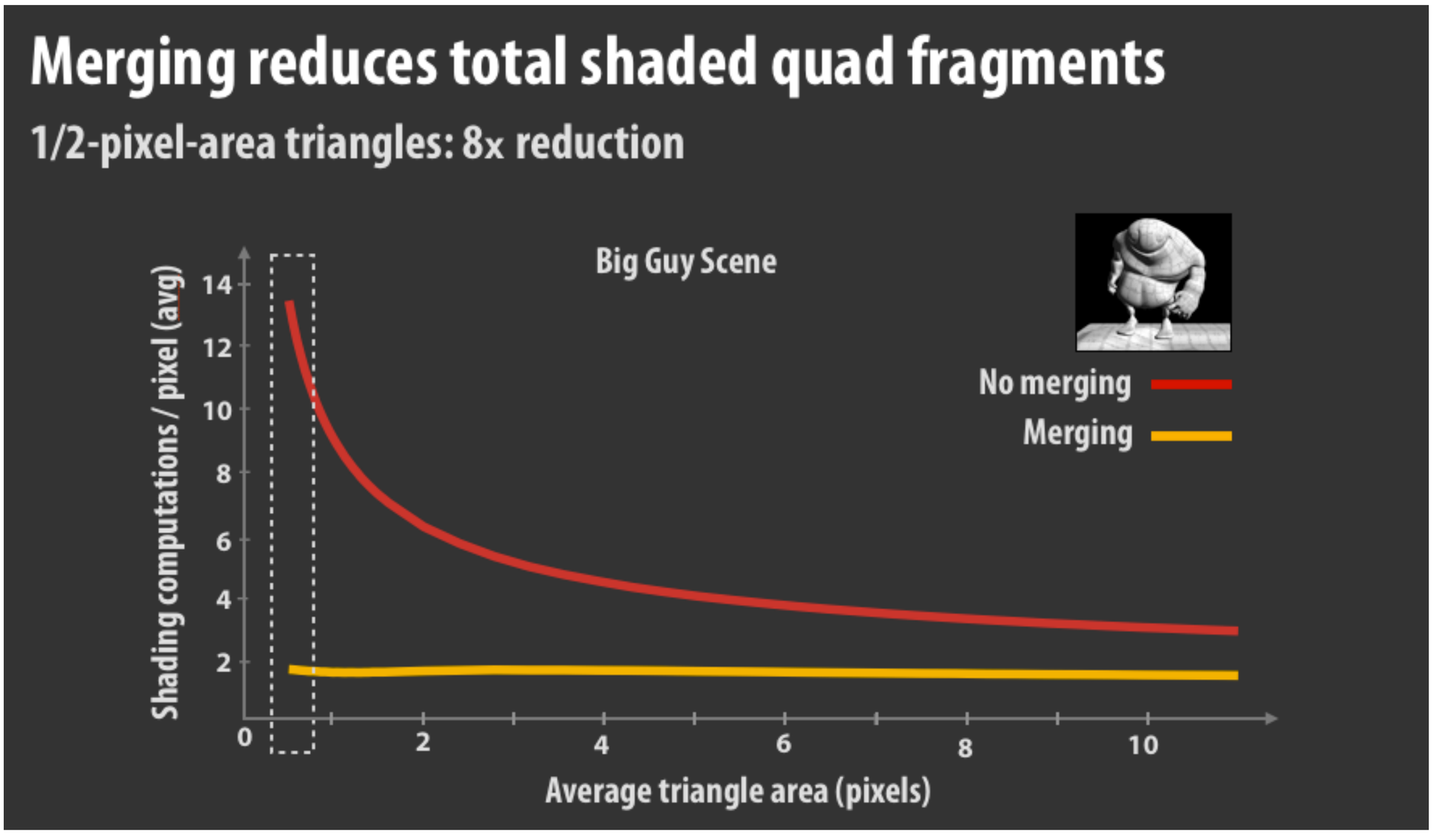
- Lead the listener through the key points of the figure
- Useful phrase: "As you can see…"
  - It's like verbal eye contact. It keeps the listener engaged and makes the listener happy… "Oh yeah, I can see that! I'm following this talk!"



Example voice over: "Now I'm showing you two adjacent triangles, and I'm coloring pixels according to the number of shading computations that occur at each pixel as a result of rendering these two triangles. As you can see in the light blue region, pixels near the boundary of the two triangles get shaded twice.

Explain every results graph

- May start with a general intro of what the graph will address.
- Then describe the axes (your axes better have labels!)
- Then describe the <u>one point</u> that you wish to make with this results slide (more on this later!)



Example voice over: "Our first questions were about performance: how much did merging reduce the number of the shaded quad-fragments? And we found out that the answer is a lot. This figure plots the number of shading computations per pixel when rendering different tessellations of the big guy scene. X-axis gives triangle size. If you look at the left side of the graph, which corresponds to a high-resolution micropolygon mesh, <u>you can see</u> that merging, shown by yellow line, shades over eight times less than the convention pipeline.

8.

In the results section:

One point per slide!

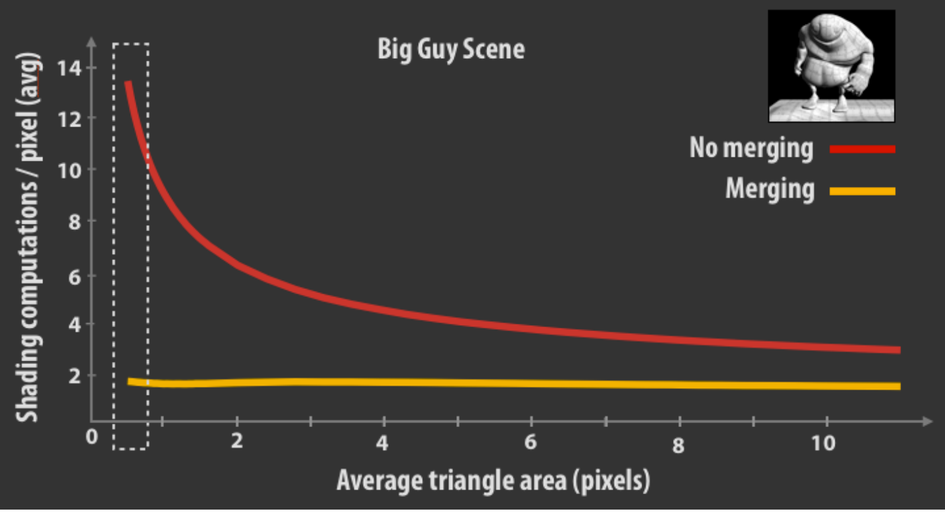
One point per slide!

One point per slide!

(and the point is the title of the slide!!!)

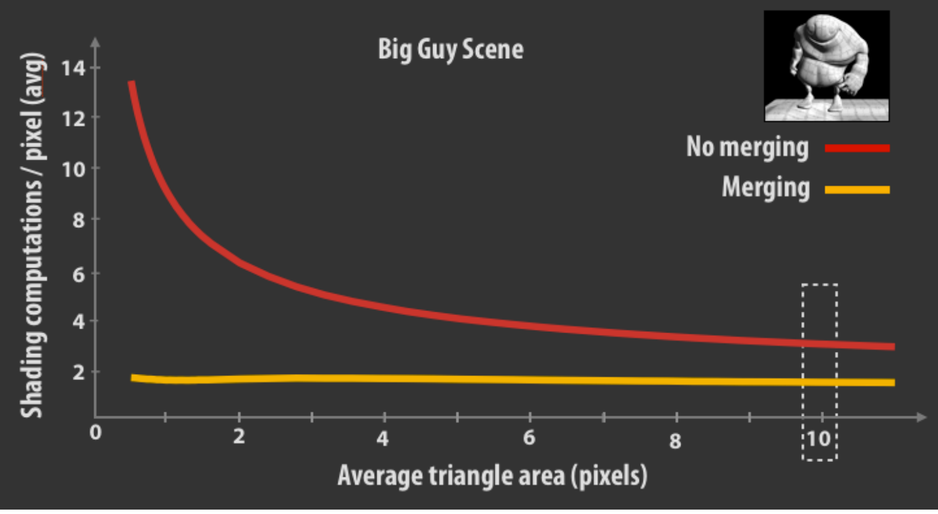
Merging reduces total shaded quad fragments

1/2-pixel-area triangles: 8x reduction

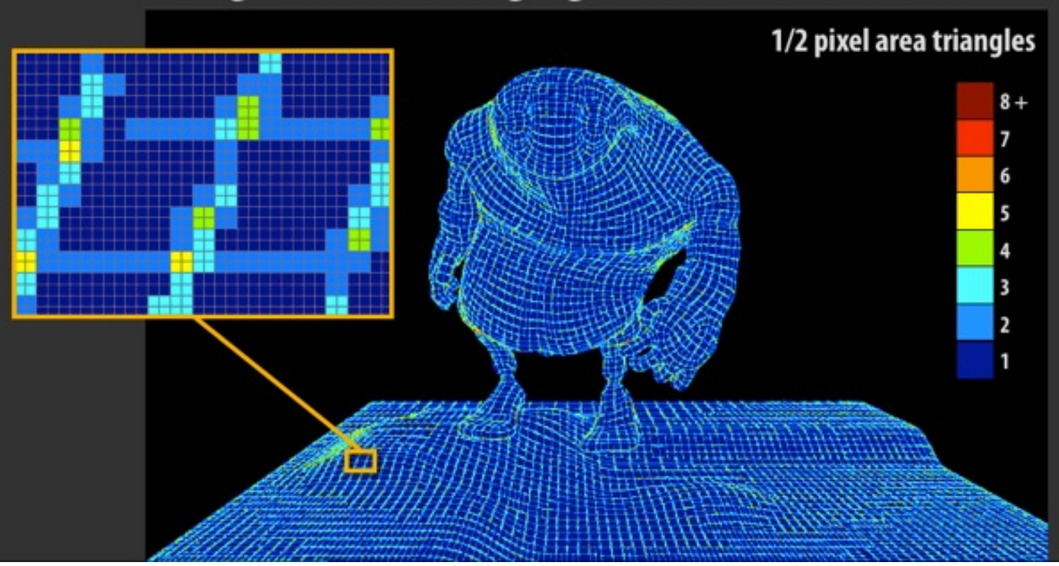


Merging reduces total shaded quad fragments

Ten-pixel-area triangles: 2x reduction

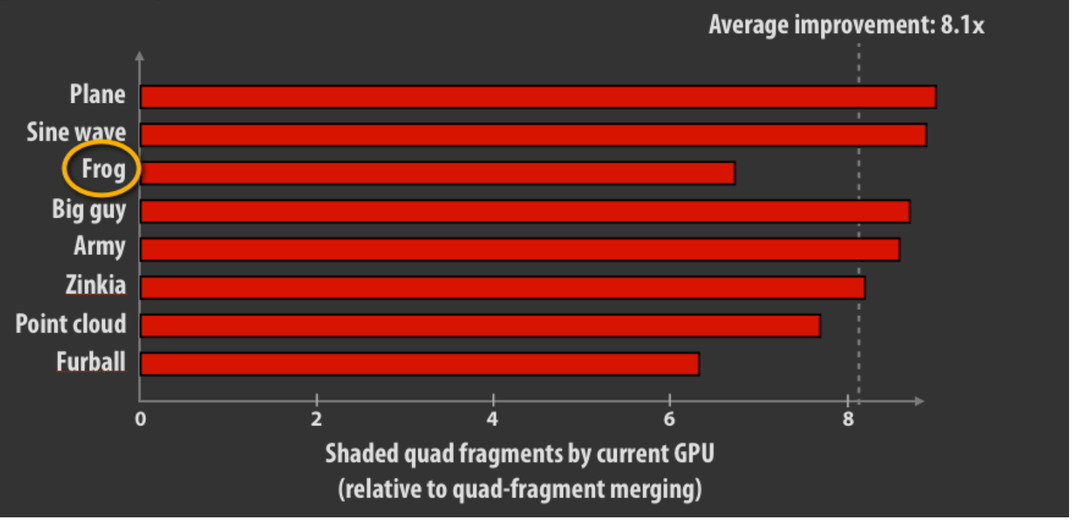


Extra shading occurs at merging window boundaries

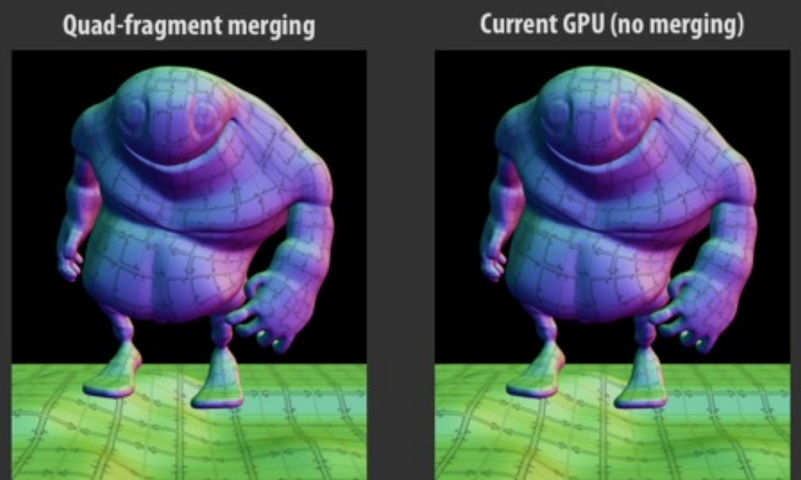


For micropolygons: factor of eight across scenes

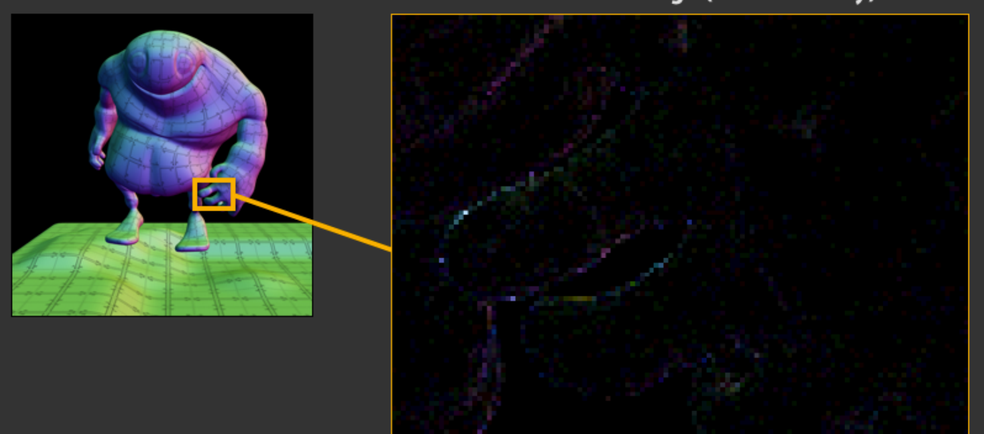
1/2 pixel area triangles

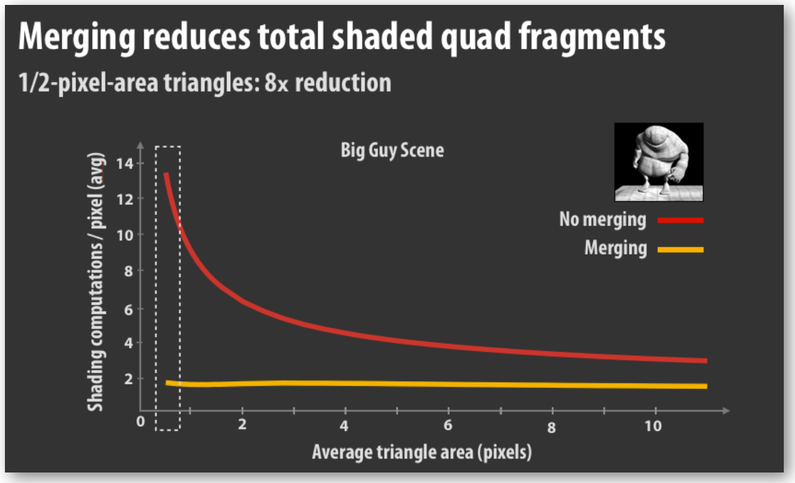


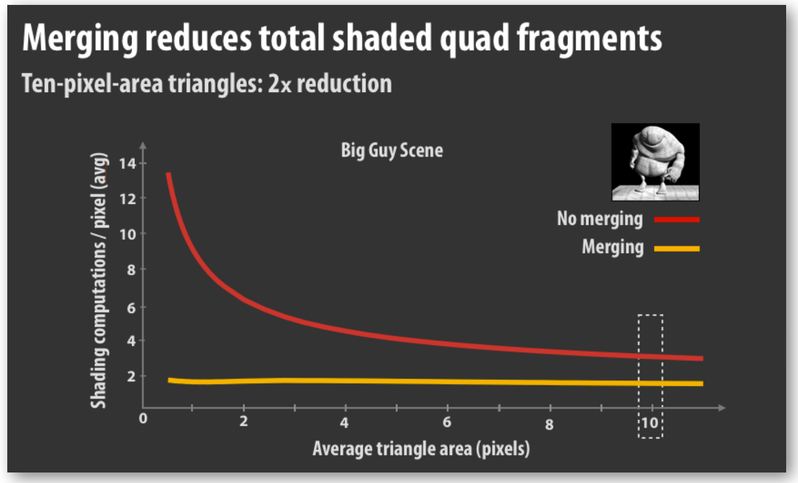
Nearly identical visual quality

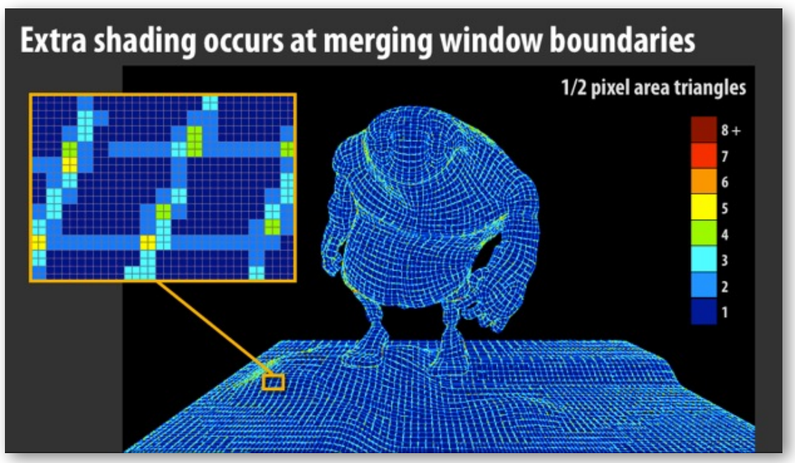


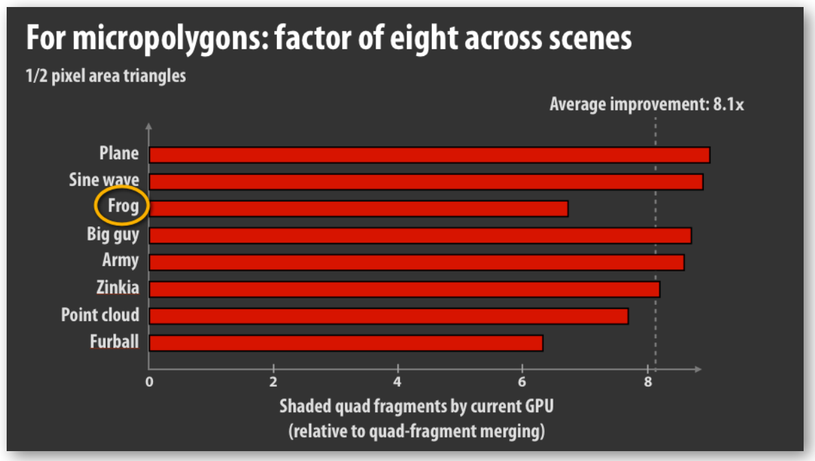
Differences exist near silhouettes

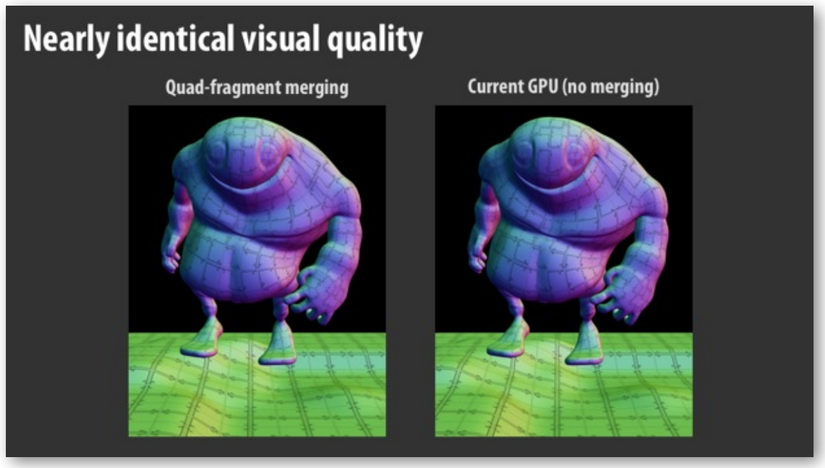


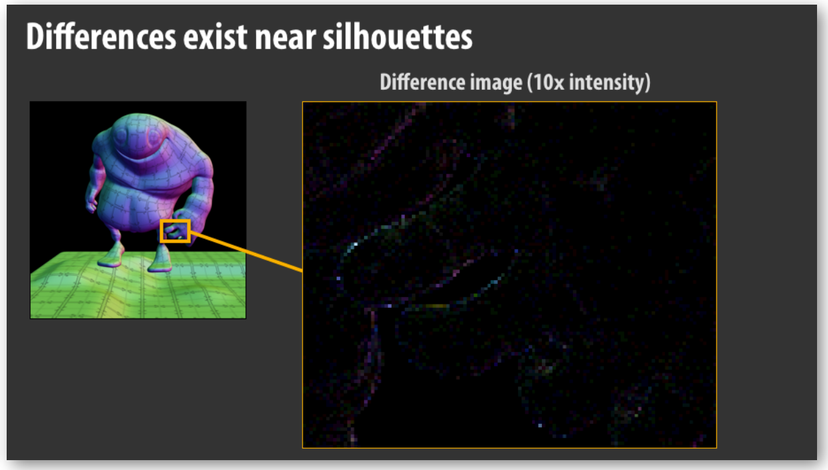










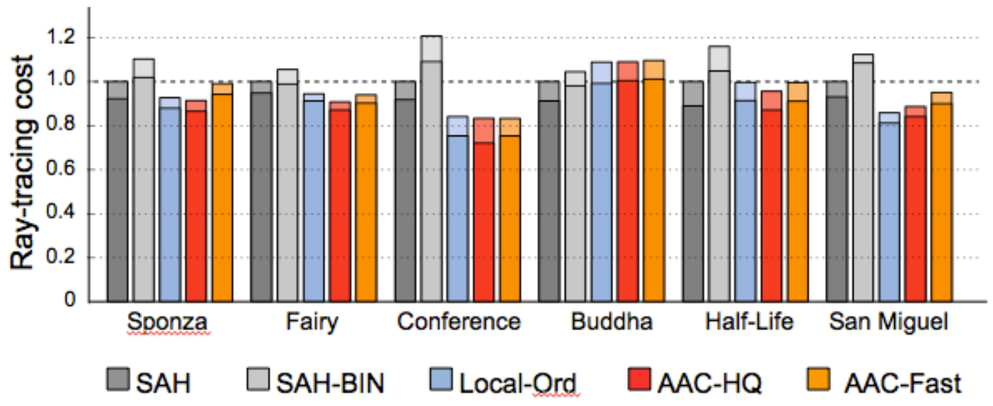


- **Place the point of the slide in the title:**
  - **Provide context for interpreting the graph (“Let me see if I can verify the point in the graph to check my understanding”)**
  - **Another example of the “audience prefers not to think” principle**

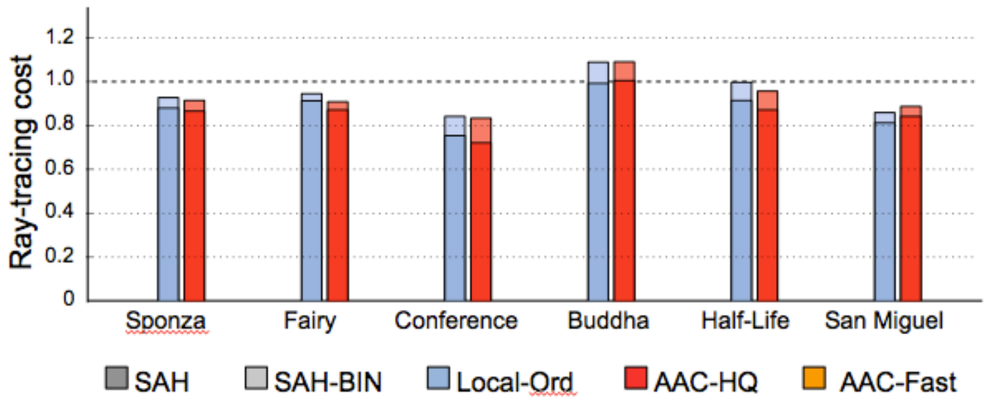
One point per results slide: a second example

TREE COST COMPARISON

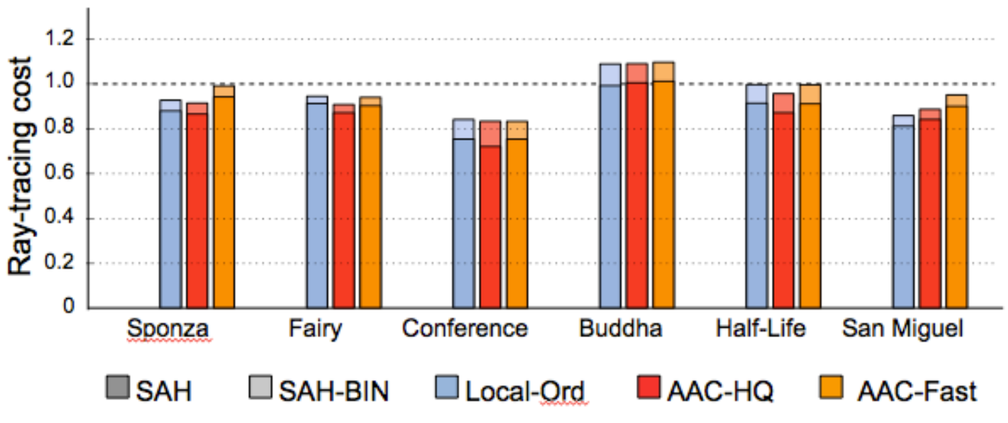
- Cost = number of traversal steps + intersection tests during ray tracing.



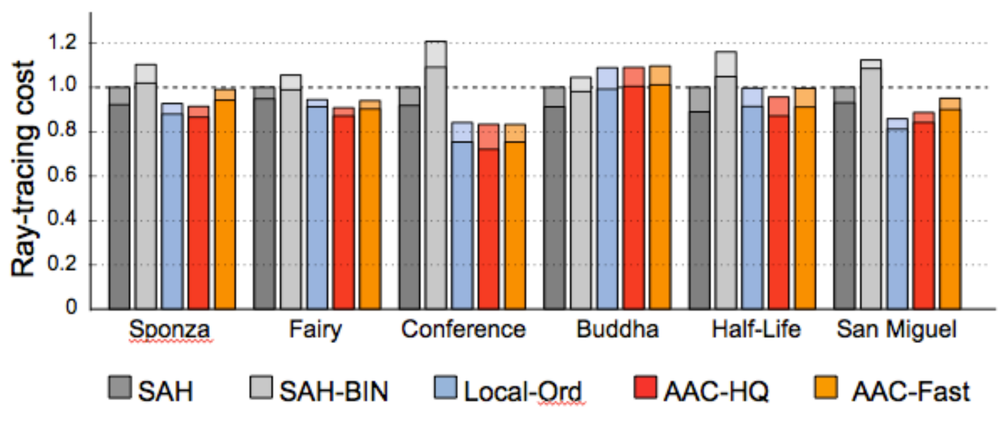
AAC-HQ produces BVHs that have similar cost as those produced by true agglomerative clustering builds.



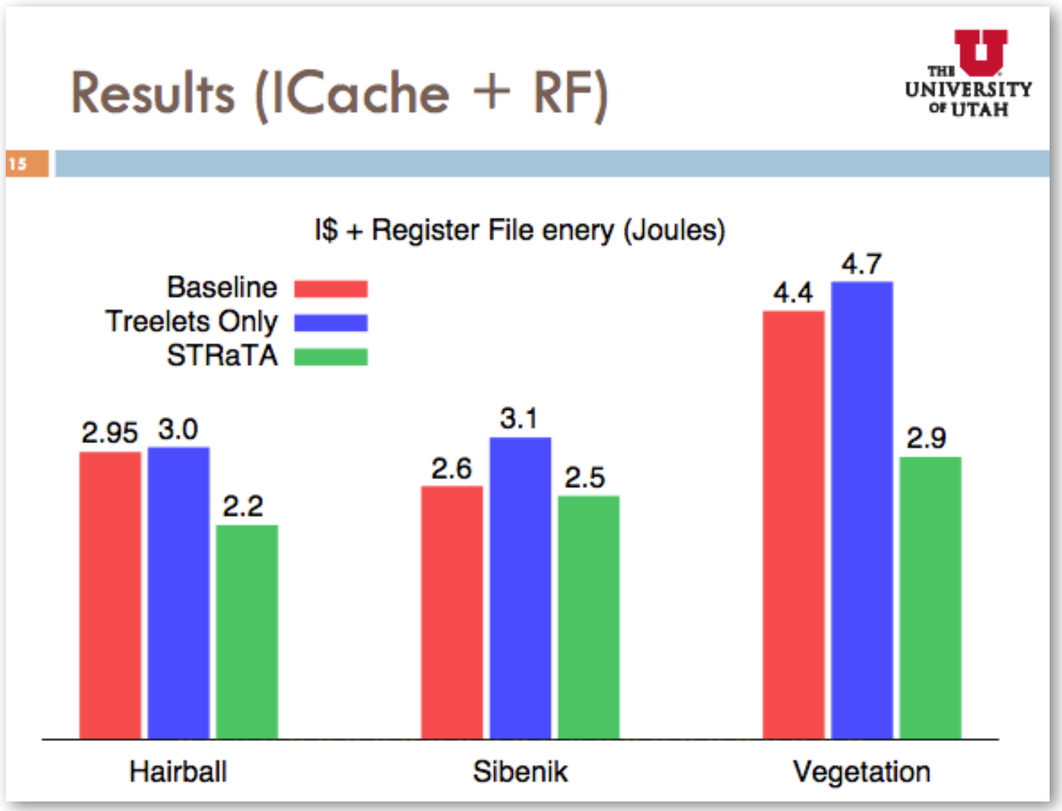
AAC-Fast produces BVHs with equal or lower cost than the **full sweep build** in all cases except Buddha.



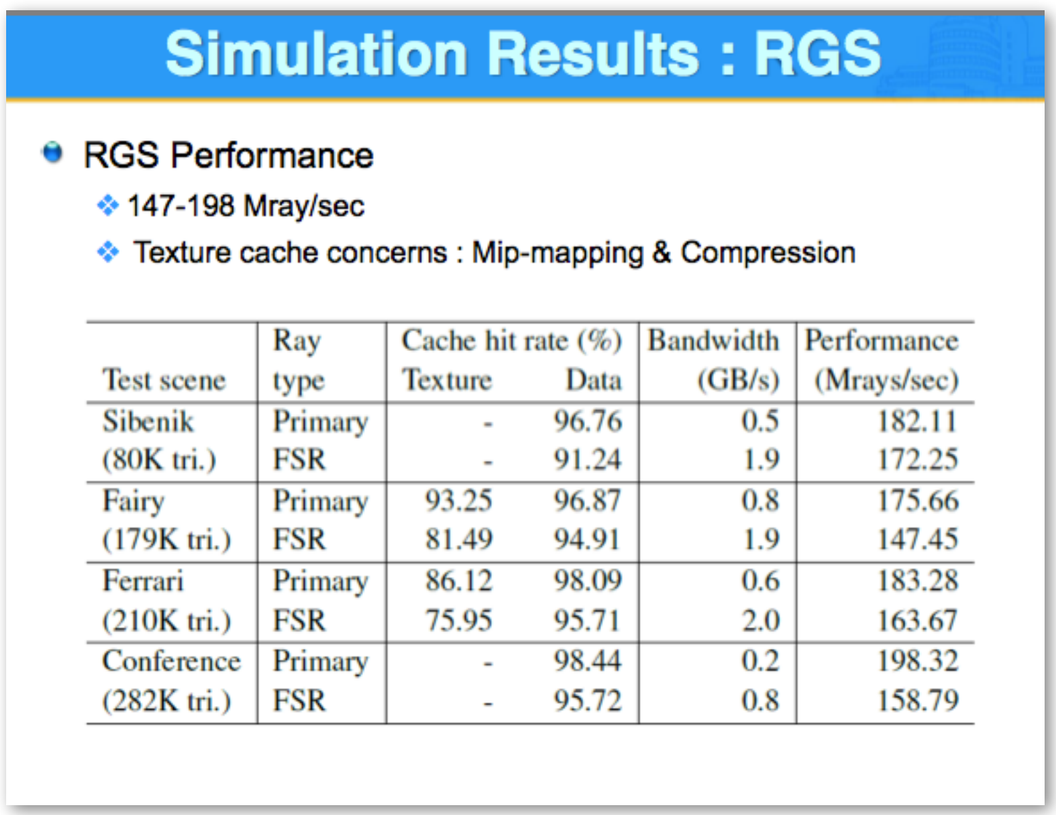
AAC-Fast produces BVHs with equal or lower cost than the **full sweep build** in all cases except Buddha.



Bad examples of results slides



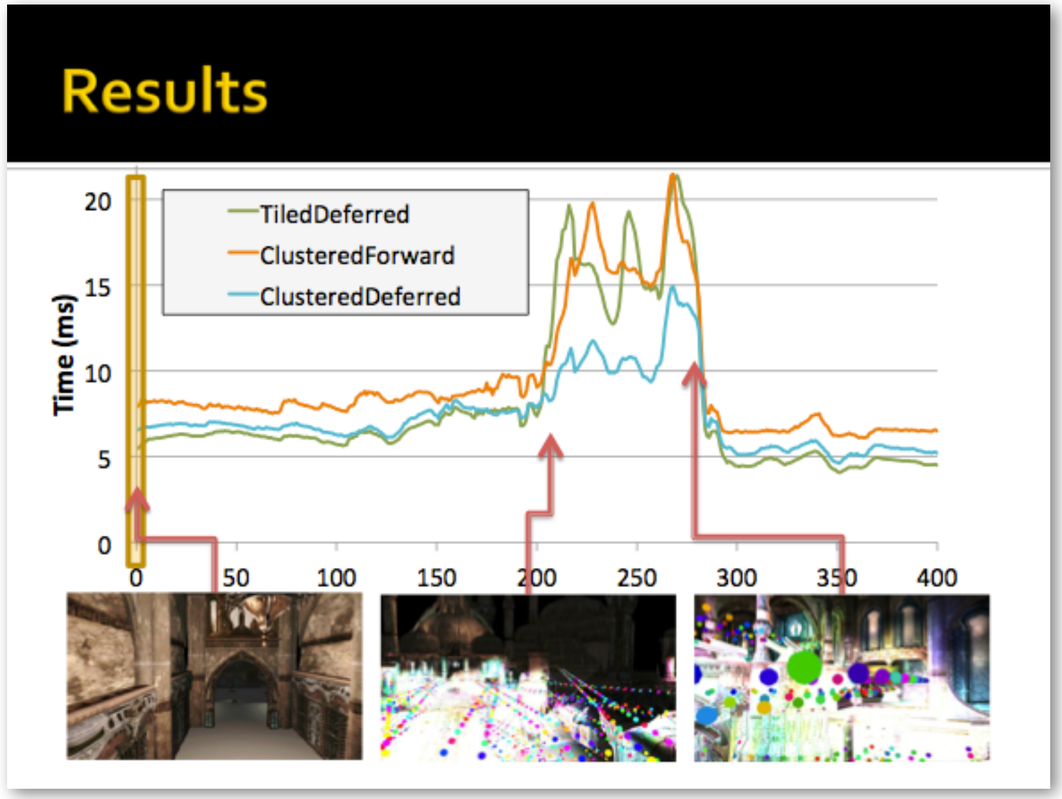
- Notice how you (as an audience member) are working to interpret the trends in these graphs
  - You are asking: what do these results say?
- You just want to be told what to look for



Simulation Results : RGS

- RGS Performance
  - 147-198 Mray/sec
  - Texture cache concerns : Mip-mapping & Compression

Test scene	Ray type	Cache hit rate (%) Texture	Data	Bandwidth (GB/s)	Performance (Mrays/sec)
Sibenik (80K tri.)	Primary	-	96.76	0.5	182.11
	FSR	-	91.24	1.9	172.25
Fairy (179K tri.)	Primary	93.25	96.87	0.8	175.66
	FSR	81.49	94.91	1.9	147.45
Ferrari (210K tri.)	Primary	86.12	98.09	0.6	183.28
	FSR	75.95	95.71	2.0	163.67
Conference (282K tri.)	Primary	-	98.44	0.2	198.32
	FSR	-	95.72	0.8	158.79



9.

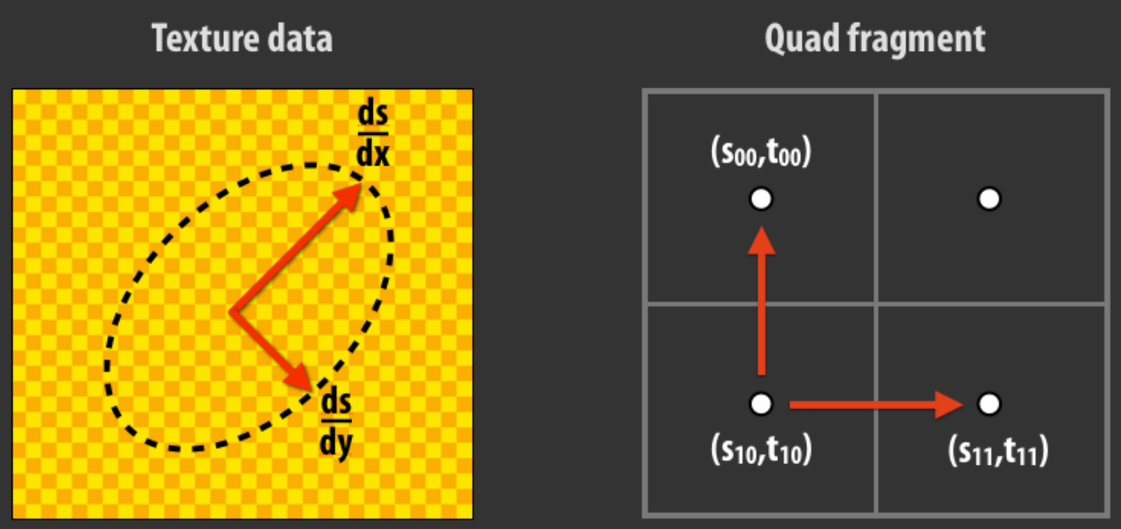
Titles matter.

If you read the titles of your talk all the way through, it should be a great summary of the talk.

(basically, this is “one-point-per-slide” for the rest of the talk)

Examples of good slide titles

GPUs shade quad fragments (2x2 pixel blocks)



use differences between neighboring texture coordinates to estimate derivatives

Greedy SRDH build optimizes over partitions and traversal policies

SAH:

```
forall (partitions in set-of-partitions)
    …evaluate SAH and pick min…
```

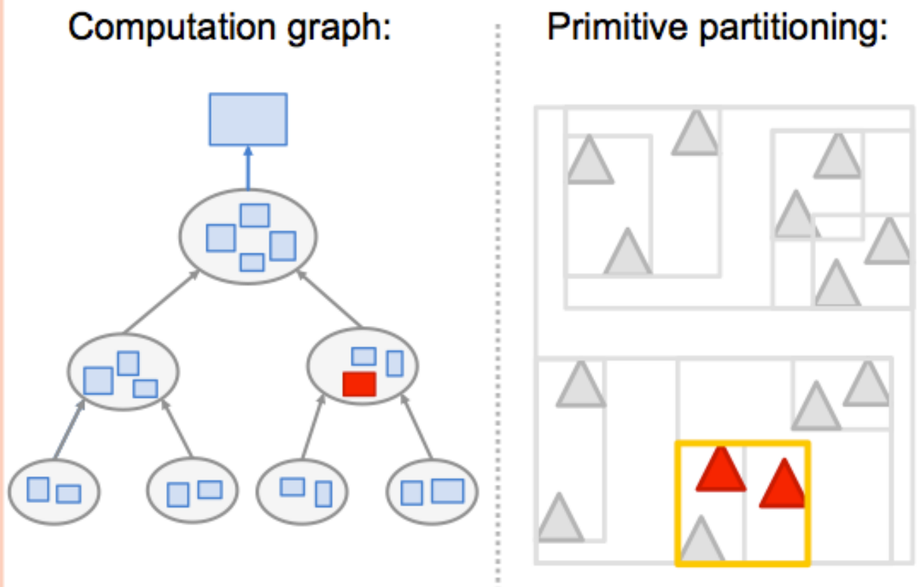
SRDH:

```
forall (partitions in set-of-partitions)
    forall (traversalKernels in set-of-kernels)
        …evaluate SRDH and pick min…
```

$$SRDH(R,L,\kappa,r)=(1-\kappa(r)H(L,r))|R|+(1-\kappa(r)H(R,r))|L|$$

51

AAC IS AN APPROXIMATION TO THE TRUE AGGLOMERATIVE CLUSTERING SOLUTION.



The reason for meaningful slide titles is convenience and clarity for the audience

"Why is the speaker telling me this again?"

(Why before what.)

Tip 10: provide evidence that your code is fast
(or your system is efficient… in other words, that you did a good job)

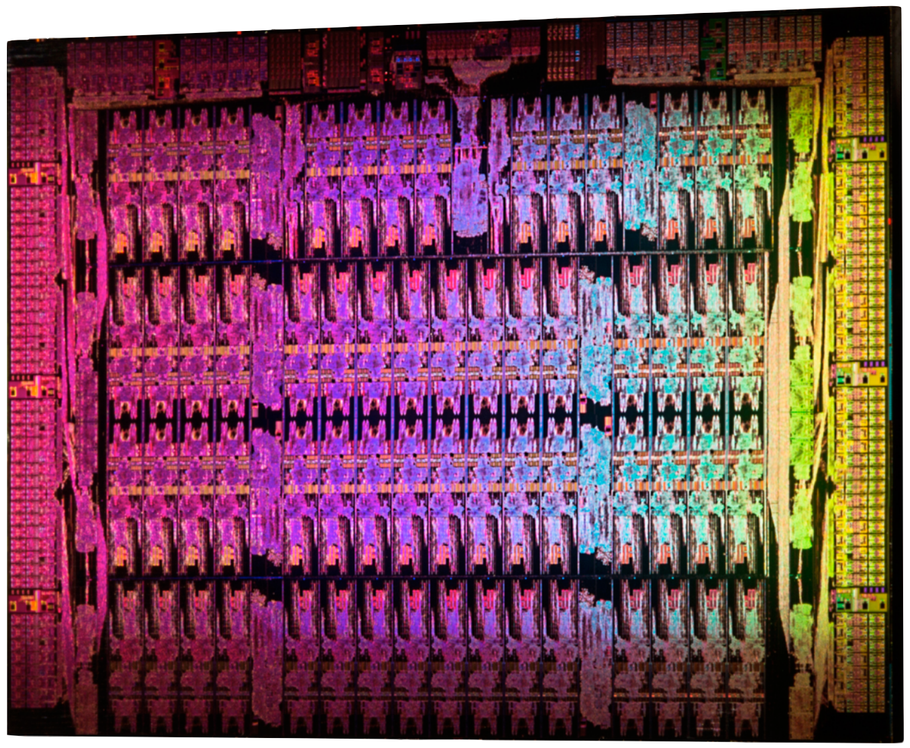
- Compare against published results
  - "Our code is 10% faster than this publication"
- Determine a fraction of peak
  - "We achieve 80% of peak performance on this machine"
- Be truthful about comparisons between a CUDA implementation utilizing an entire GPU and single threaded, non-SIMD C program on a CPU (I'd rather not hear the conclusion: "GPU is 100x faster than the CPU".)

Tip 11: practice the presentation

- Given the time constraints, you'll need to be smooth to say everything you want to say
- To be smooth you'll have to practice
- I hope you rehearse your demo or presentation a several times the night before (in front of a friend or two that's not in 418)
  - It's only a 6 minute talk. Many practice runs are possible in a small amount of time

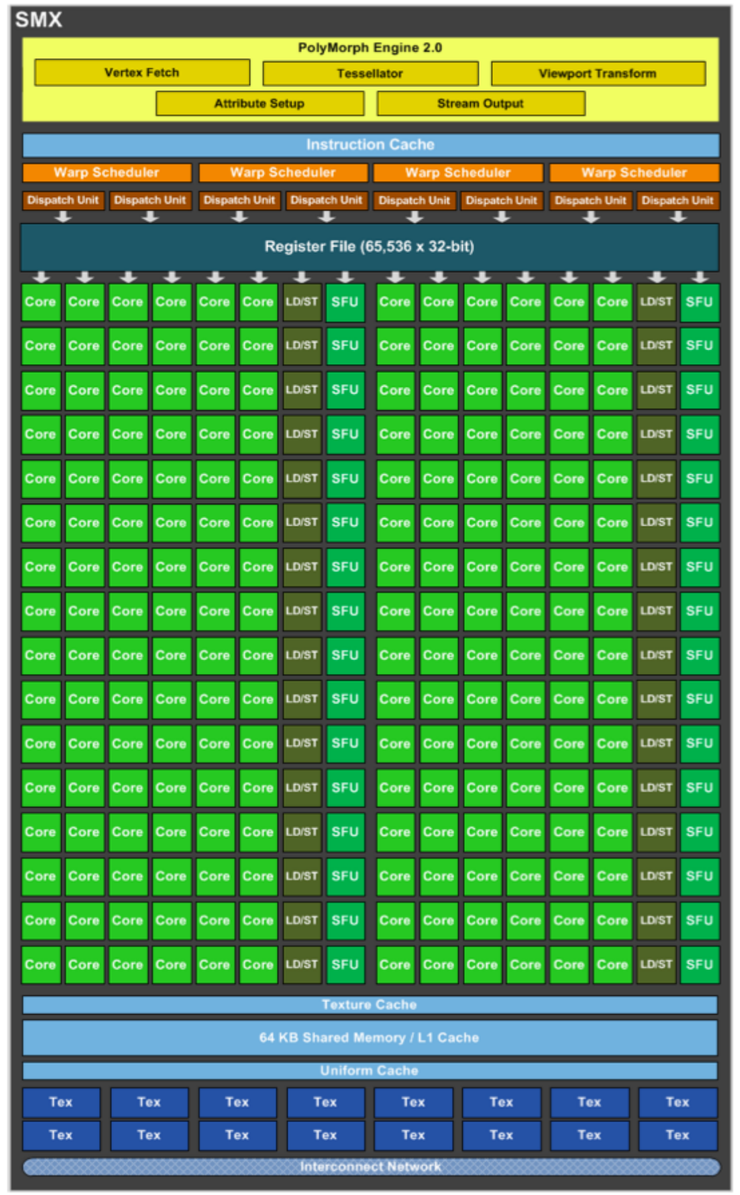
Course wrap up

For the foreseeable future, the primary way to obtain higher performance computing hardware is through increased parallelism and hardware specialization.



Intel Xeon Phi, 62 cores, 16-wide SIMD

Heterogeneous SoC



NVIDIA Kepler Core (SMX)
32 wide SIMD, up to 2048 CUDA/core threads

Today's software is surprisingly inefficient compared to the capability of modern machines

A lot of performance is currently left on the table (increasingly so as machines get more complex, and parallel processing capability grows)

Extracting this performance stands to provide a notable impact on many compute-intensive fields (or enable new applications of computing!)

Given current software programming systems and tools, understanding how a parallel machine works is important to achieving high performance.

A major challenge going forward is making it simpler for programmers to extract performance on these complex machines.

Efficiency matters more, not less!

Key issues

Identifying parallelism

(or conversely, identifying dependencies)

Scheduling parallelism

1. Achieving good workload balance

2. Overcoming communication constraints:
Bandwidth limits, dealing with latency, synchronization
Exploiting locality in data or execution = managing state!

We addressed these issues at many scales and in many contexts

Single chip, multi-core CPU
Multi-core GPU
CPU+GPU connected via bus
Large scale, multi-node supercomputer
Collection of machines in cloud

Other classes I teach

- **15-869 Visual Computing Systems (Fall 2014)**
  - Parallel systems for "visual computing" applications (graphics, vision, computational photography).
  - Hoping for a mix of students
    - 15-418 gives you systems background
    - At least one of 15-462, 15-463, 15-385 gives you some applications background

Example topics: OpenGL pipeline implementation, camera imaging pipeline, systems for searching through big visual data

Beyond assignments and exams

- Come talk to me (or other professors) next fall about participating in research!
- Consider a senior thesis!
- Pitch a seed idea to Project Olympus
- Get involved with organizations like Hackathon or ScottyLabs

Kayvon's Final Claim

It is far more difficult (and creative) to deliver on a "once in a few years" final project in 15-418/618 than it is to take an extra class.

Ditto for pursuits like senior theses, independent study, starting a company...

The world rewards initiative.

The world rewards risk takers.

(Force yourself to get comfortable with risk taking and the possibility of occasional failure while at CMU.)

Many people are really smart.

Many people can work really hard.

Far fewer people develop the confidence and creativity to lead.

Thanks for being a great class!

Good luck on projects. Expectations are high.

See you a week from Friday!