Lecture 19:

Course Recap + Project Presentation Talk Tips

Visual Computing Systems

Stanford CS348K, Spring 2023

Today

- Small group discussions about projects
- A quick course recap
- Kayvon’s tips for giving clear project presentations

Course recap

A few course themes

- *Thinking like a systems architect*
  - What are inputs/outputs, constraints, and goals?
  - What are the "services" the system should perform (what is hard for a user to do in a world without the system)
  - Once you can establish answers to these questions, you can consider your solution options

- Knowledge of applications and systems is necessary to choose efficient solutions
  - Algorithms folks use their hammer to reduce cost of algorithms (smaller DNN models, new optimization hyperparameters, use different data structures, etc.)
  - HW designers use their hammer (custom accelerators, new interconnects) to execute a given workload faster
  - SW systems folks use their hammer (parallel/distributed computing, workload-specific scheduling, high level programming abstractions, etc.)

- The best solutions architects pick the right mixture of hammers for the job

A few course themes

- Knowledge of applications and systems is necessary to do meaningful evaluation
  - Is this the right workload to evaluate?
    - Pitfall: measuring speedup on a part of the workload that is not the most significant
    - Does dataset I'm using have the right data distribution?
  - Am I measuring cost in FLOPs, but increasing data movement?
  - Am I optimizing an algorithm that is not the right choice of algorithm for this problem?

Communicating like an architect
(Aka. project presentation tips)

Why get good at communication?

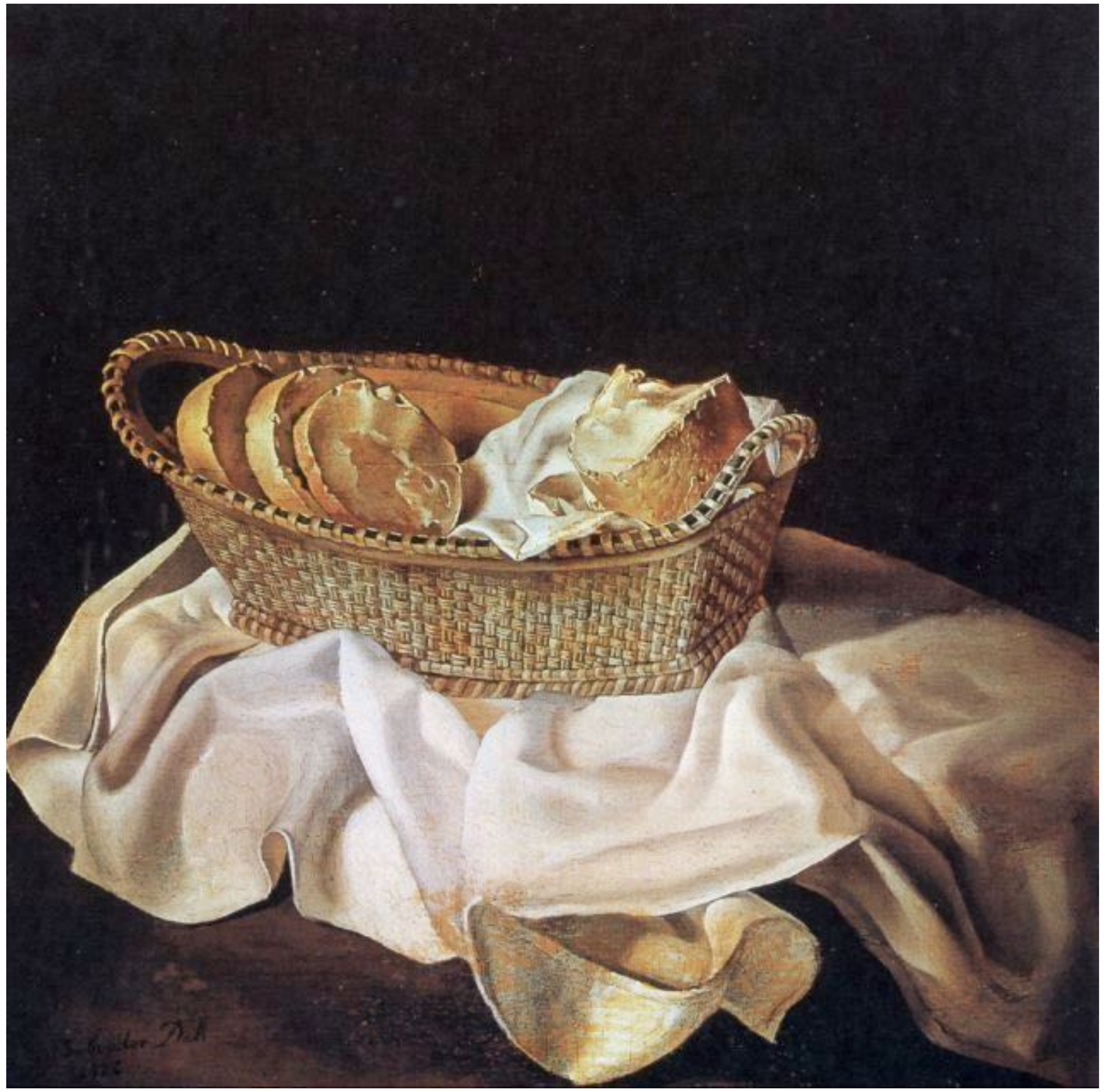
- Systems architects need to be some of the best communicators in an organization
- Must be able to:
  - Communicate with users to understand workloads and constraints
  - Communicate with the individual contributors/engineers to:
    - Understand emerging problems/constraints
    - Understand how to evolve/extending existing designs to enable new functionality
  - Must constantly be communicating to various parties why:
    - Their desired features are not being added
    - They must do the same work with 50% of the resources
  - Communicate to everyone a strategic vision for a system
  - Communicate to executives/management/funders how goals are being met.

My motivation

- I have found I give nearly the same feedback over and over to students making talks
  - It is not profound feedback, it is just application of a simple set of techniques and principles that are consistently useful when making talks

- I am hoping these slides serve as a useful checklist you can refer to vet your own project presentation talks before giving them next Tuesday
  - Don’t worry: I still make these mistakes all the time when creating first drafts of talks

Who painted this painting?



Salvador Dali (age 22)

My point: learn the basic principles before you consciously choose to break them

Put yourself in your audience's shoes

Tip 1

This is a major challenge for most technical speakers. (including professors)

(Tip: recite a sentence out loud to yourself. * Do you really expect someone who has not been working with you everyday on the project to understand what you just said?)

*** I'm not kidding. Say it out loud. I find hearing myself say something out loud makes it easier to parse it from an audience's perspective.**

Consider your audience

- **Everyone in the audience knows about course readings/topics**
  - Terminology/concepts we all know about need not defined (just say “remember we talked about X”)

- **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
  - What they can get their head around is inputs/outputs and goals/constraints

- **Everyone wants to know the “most interesting” thing that you found out or accomplished (your job is to define most interesting for them)**

Tip 2

A good principle for any talk (or paper): "Every sentence matters"

What are you trying to say?
What technical story are you trying to tell?
What is point you are trying to make?

You might be trying to define a hypothesis.
Or define goals
Or establish inputs/outputs
Or show data that suggests you were successful.

Is what you just said making that point? (If not, remove it)

If you can't justify how it will help the listener understand the point, take it out.
If it's not likely to be clear, take it out.

Pick a focus

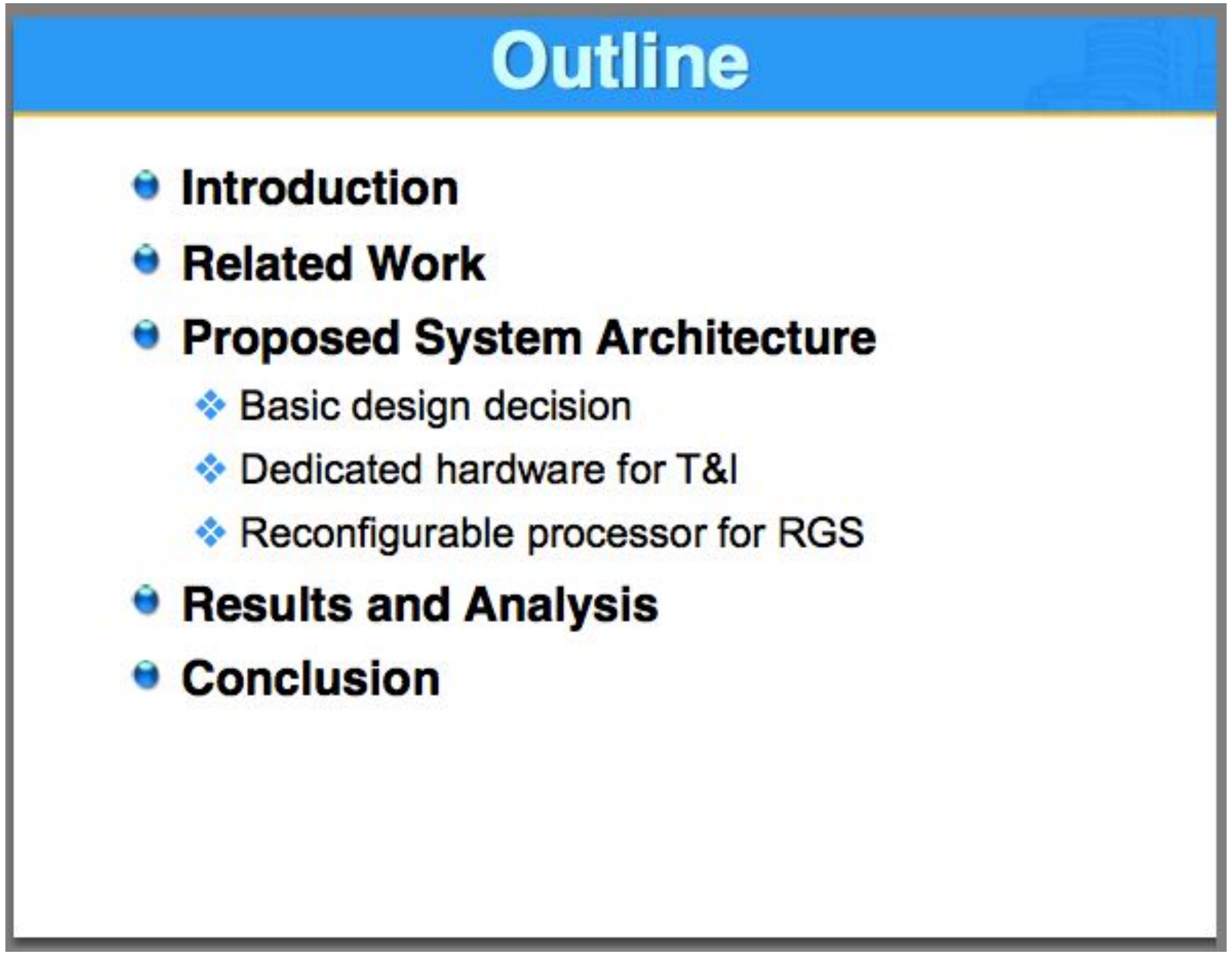
- In this class, different projects should stress different results
- Some projects may wish to show a flashy demo and describe how it works (proof by "it works")
- Other projects may wish to show a sequence of graphs (path of progressive optimization) and describe the optimization that took system from 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 not to explain what you did, but to explain what you think we should know.

And really the most important thing we want to know is (1) what was your goal? and (2) what's the evidence you have that you were successful?

Ignoring every sentence matters

Never ever, ever, ever do this!

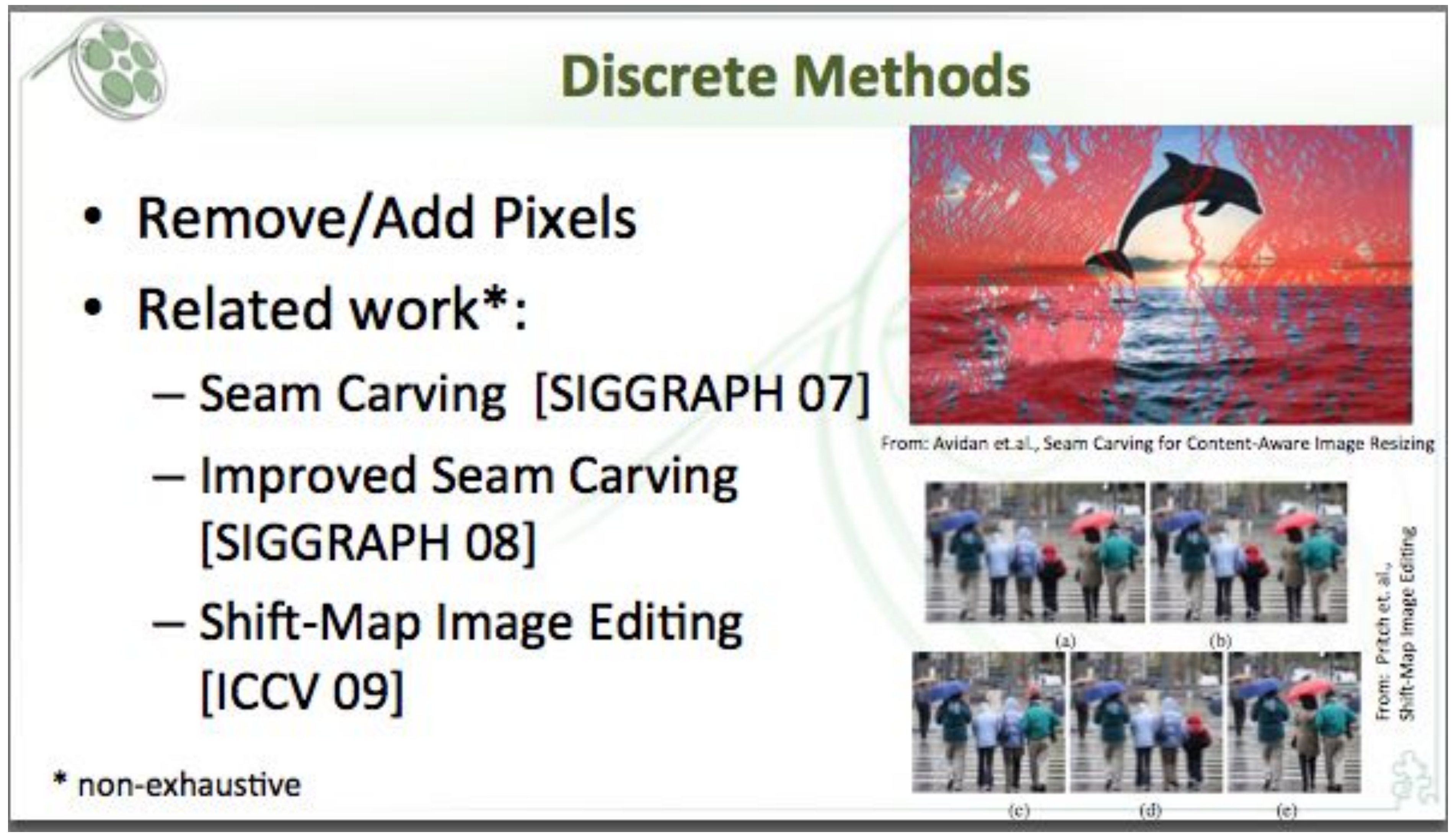


Outline

- Introduction
- Related Work
- Proposed System Architecture
  - Basic design decision
  - Dedicated hardware for T&I
  - Reconfigurable processor for RGS
- Results and Analysis
- Conclusion

Bad example 2

- Who is the audience for this? (how does this benefit them?)



Discrete Methods

- Remove/Add Pixels
- Related work*:
  - Seam Carving [SIGGRAPH 07]
  - Improved Seam Carving [SIGGRAPH 08]
  - Shift-Map Image Editing [ICCV 09]

* non-exhaustive

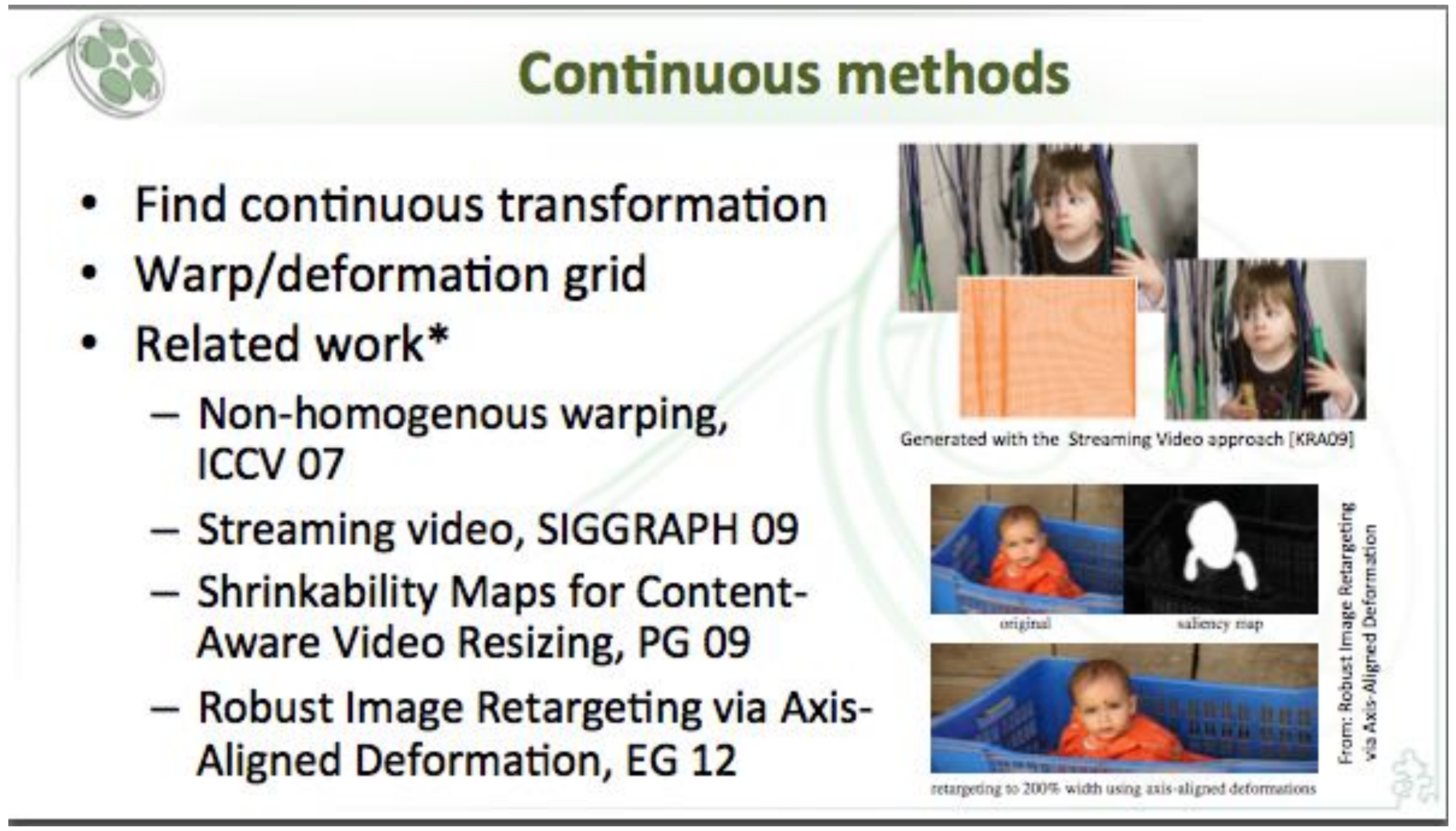
From: Avidan et.al., Seam Carving for Content-Aware Image Resizing

From: Pritch et al., Shift-Map Image Editing

- Experts?
  - They likely know these papers exist. These slides don't tell them what about these papers is most relevant to this talk
- Non-experts?
  - They won't learn the related work from these two slides

This type of related work section says little more than "others have worked in this area before".

- I suspect your audience assumes this is the case.
- Every sentence matters: if it doesn't provide value, take it out (or replace it with comments that do provide value)



Continuous methods

- Find continuous transformation
- Warp/deformation grid
- Related work*
  - Non-homogenous warping, ICCV 07
  - Streaming video, SIGGRAPH 09
  - Shrinkability Maps for Content-Aware Video Resizing, PG 09
  - Robust Image Retargeting via Axis-Aligned Deformation, EG 12

Generated with the Streaming Video approach [KRA09]

From: Robust Image Retargeting via Axis-Aligned Deformation

Tip 3

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 <u>do not</u> want to interpret any of your figures or graphs, they want to be directly told how to interpret them (e.g., what to look for in a graph).
  - They want to be told about your key assumptions

- **The audience <u>does</u> want to spend their energy thinking about:**
  - Potential problems with what you did (did you consider all edge cases? Is your evaluation methodology sound? Is this a good platform for this workload?)
  - Implications of your approach to other things
  - Connections to their own project or interests

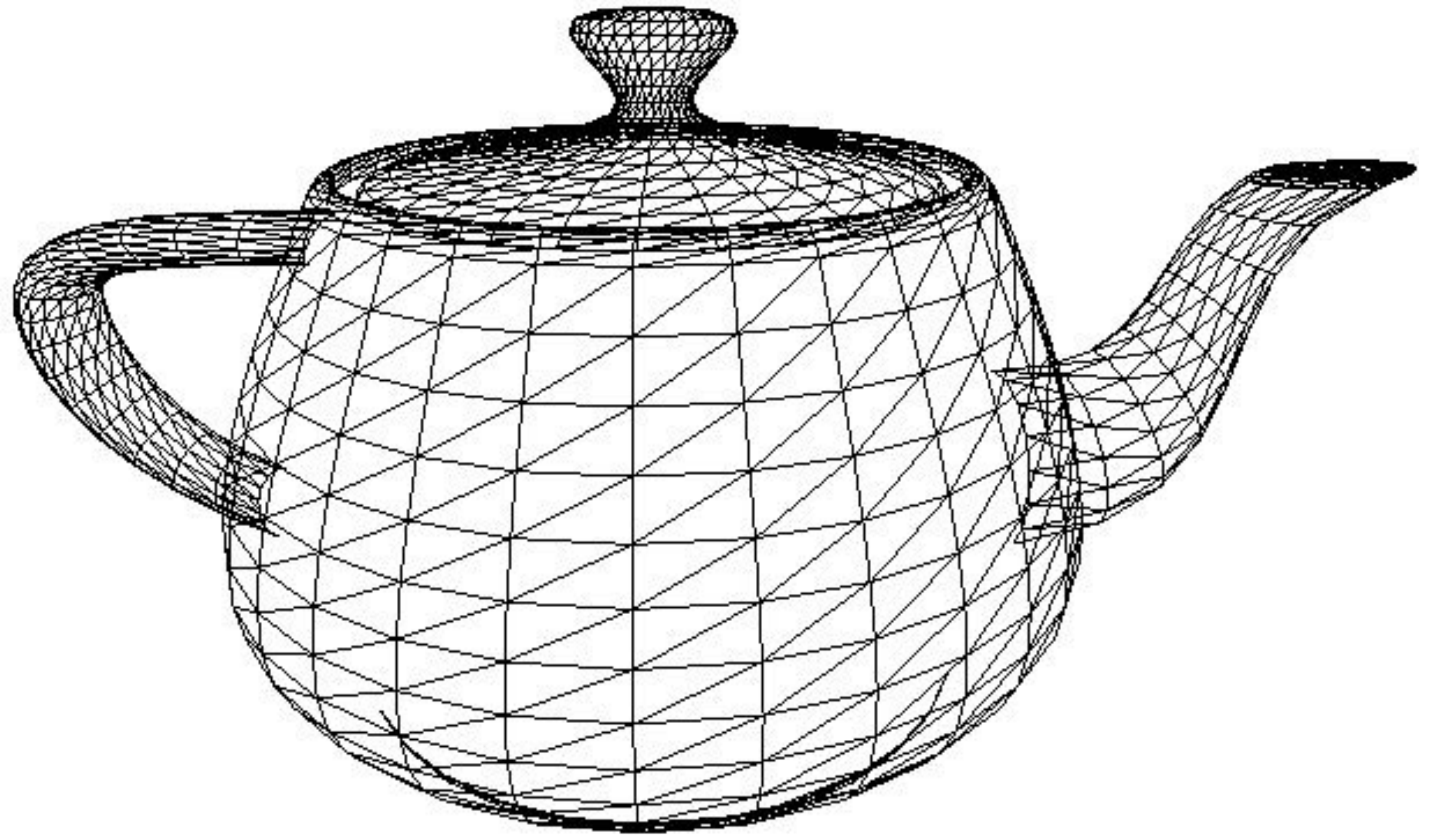
Tip 4

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

Basics of problem setup

- **What is the computation performed (or system built)?**
  - What are the inputs? What are 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 should we reward you for?)**
  - What turned out to be the hardest part of the problem?
  - Optimization projects: e.g., overcoming SIMD divergence, increasing arithmetic intensity
  - Applications projects: coming up with an algorithm to estimate depth, dealing with motion blur
  - Abstraction/API projects: deciding between stateless or stateful interfaces

Example: 3D rendering problem



$\rightarrow$

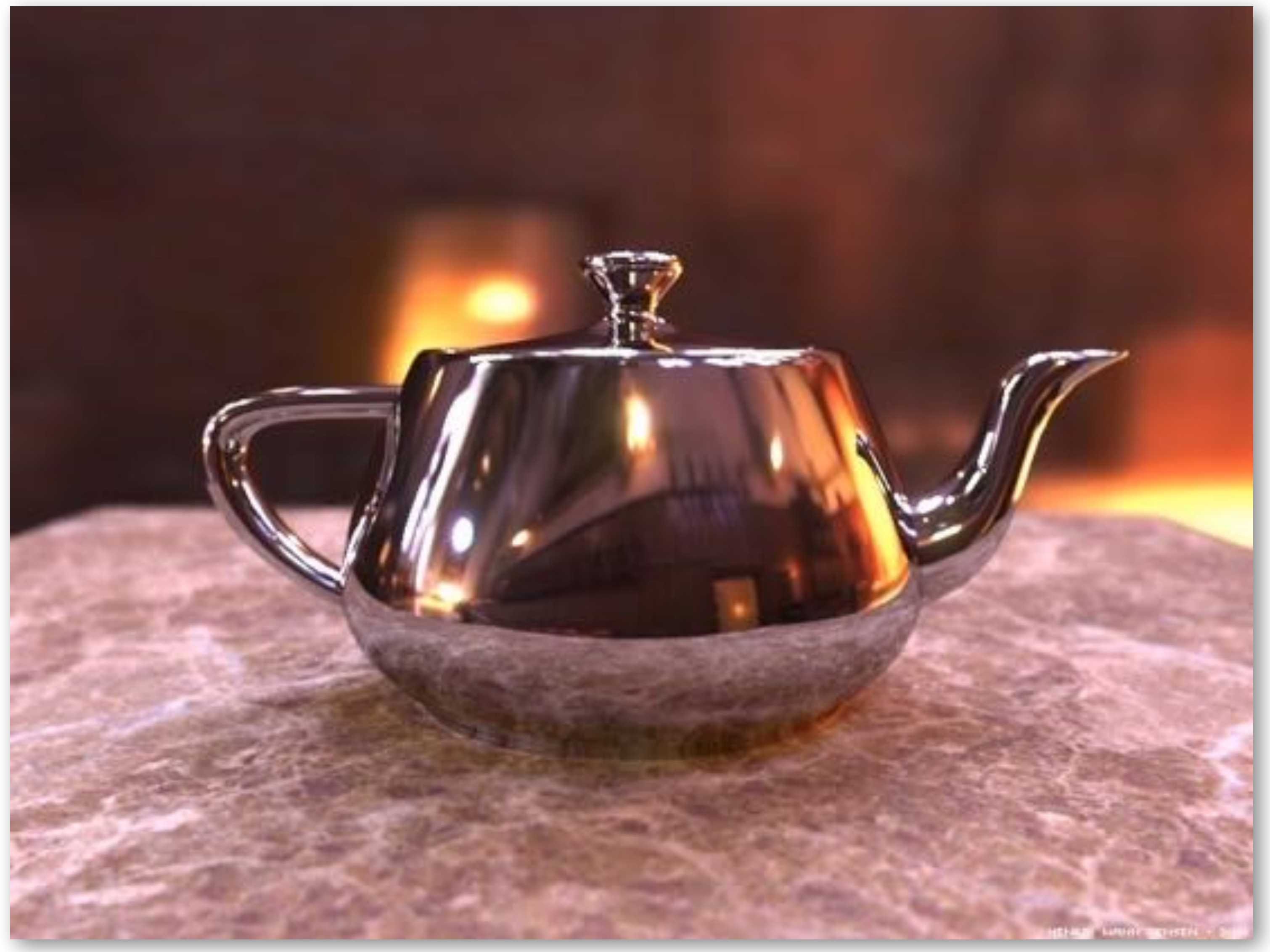


Image credit: Henrik Wann Jensen

Input: description of a scene:
3D surface geometry (e.g., triangle mesh)
surface materials, lights, camera, etc.

Output: image of the scene

Simple definition of rendering task: computing how each triangle in 3D mesh contributes to appearance of each pixel in the image?

Why is knowing the goals and constraints important?

Your contribution is typically a system or algorithm that meets the stated goals under the stated constraints.

Understanding whether a solution is “good” requires having this problem context.

Tip 5

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 <u>state</u> in the system?

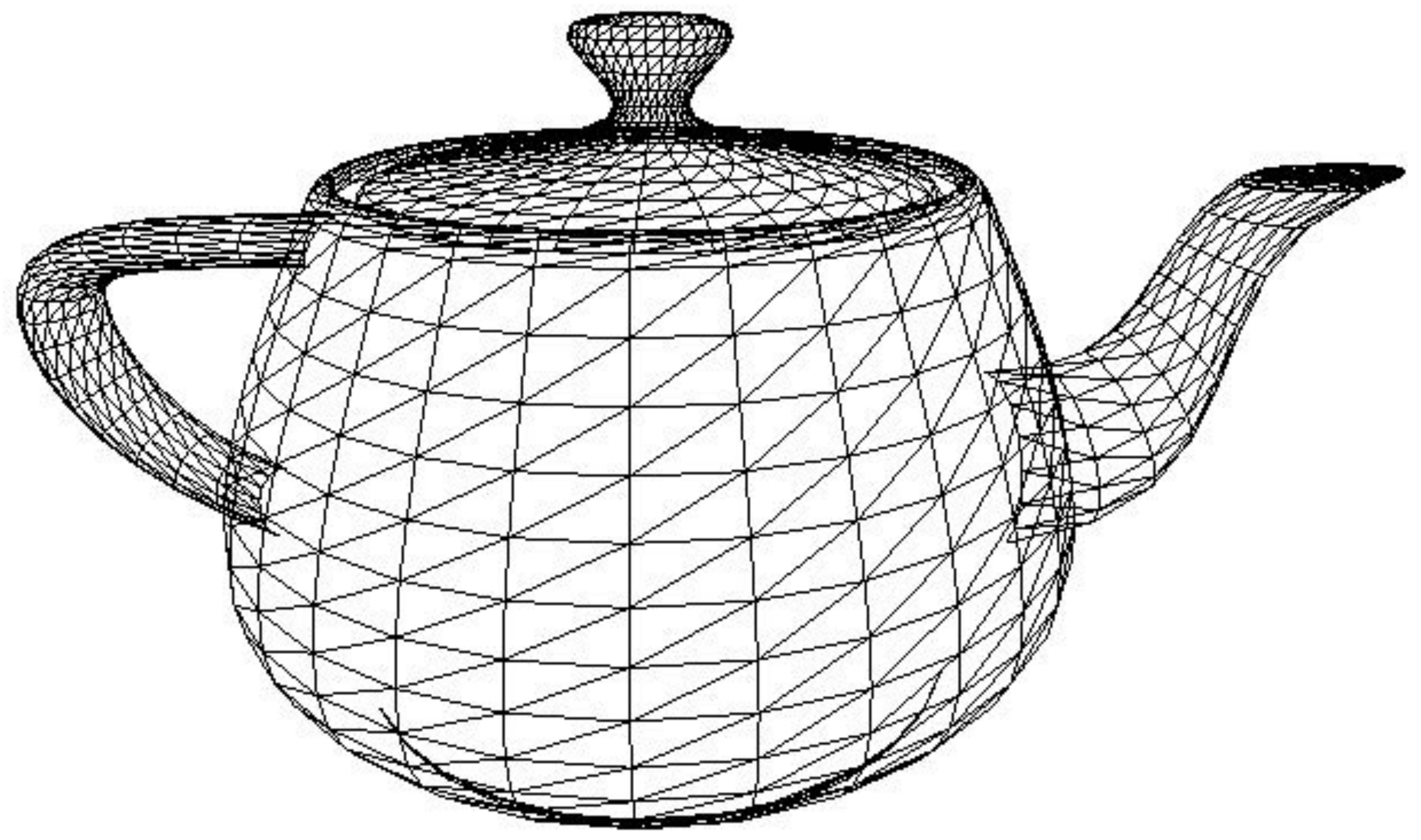
- **Then describe the verbs**
  - Operations that can be performed on the state (update particle positions, compute gradient of pixels, traverse graph, etc.)
  - Operations produce, consume, or transform entities

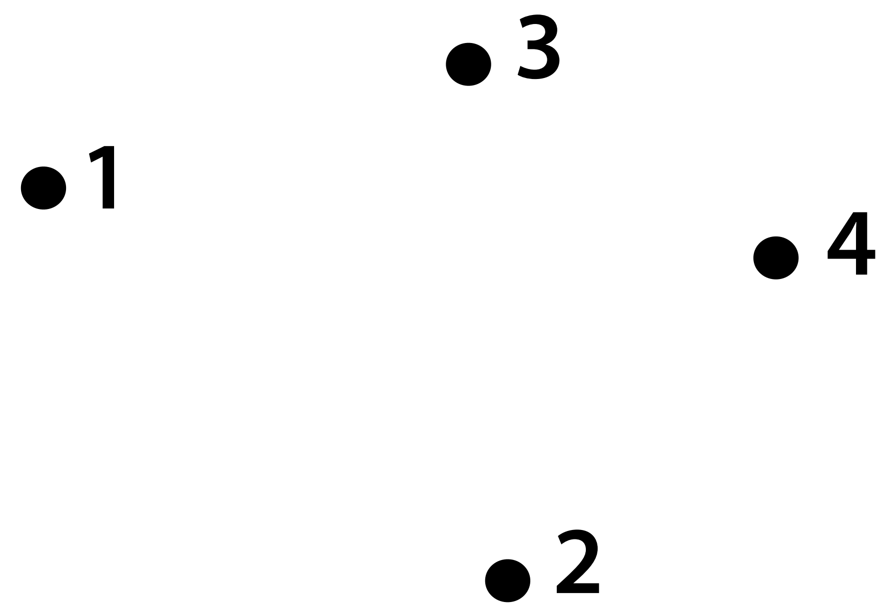
Tip: how to explain "a system"

- Step 1: describe the <u>things</u> (key entities) that are manipulated
  - The nouns

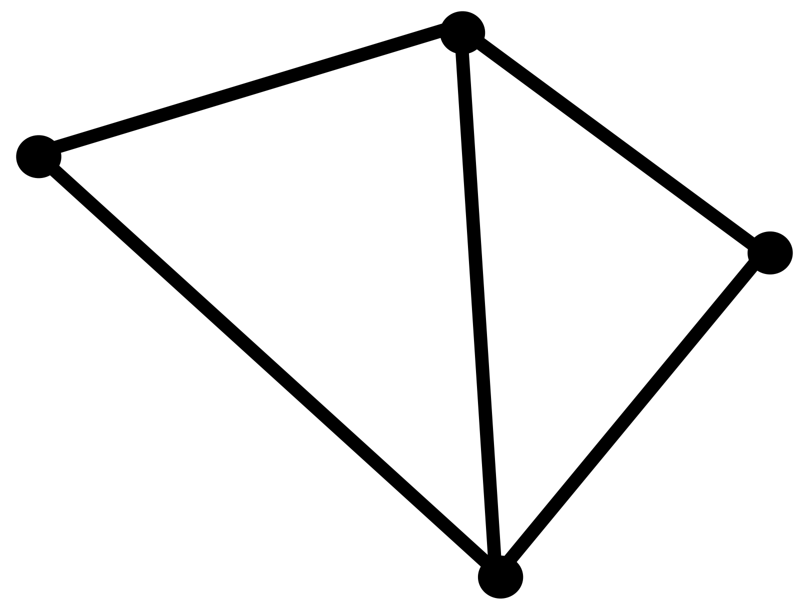
Example: real-time graphics primitives (entities)

Represent surface as a 3D triangle mesh



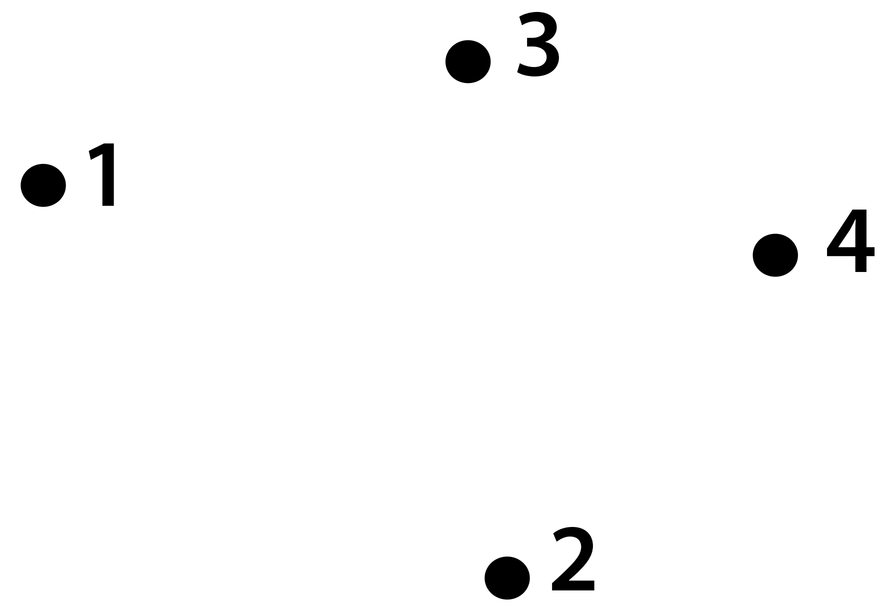


Vertices
(points in space)

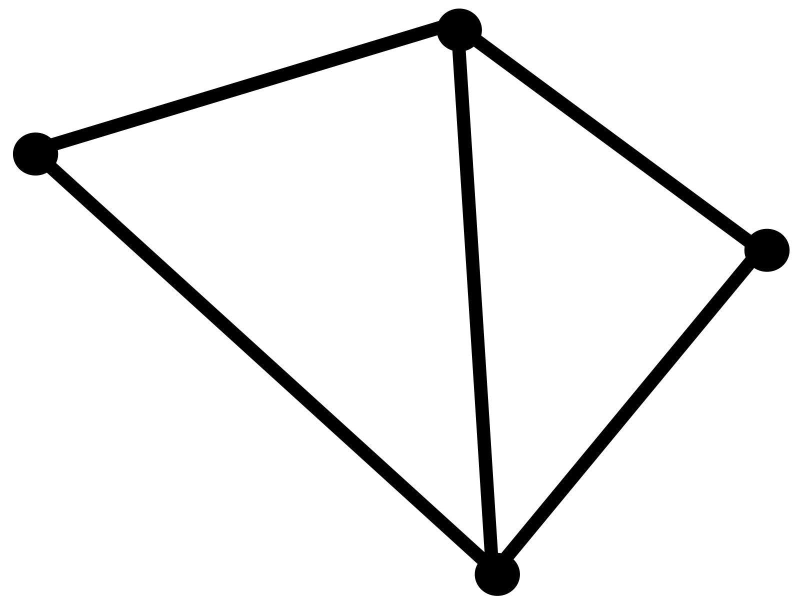


Primitives
(e.g., triangles, points, lines)

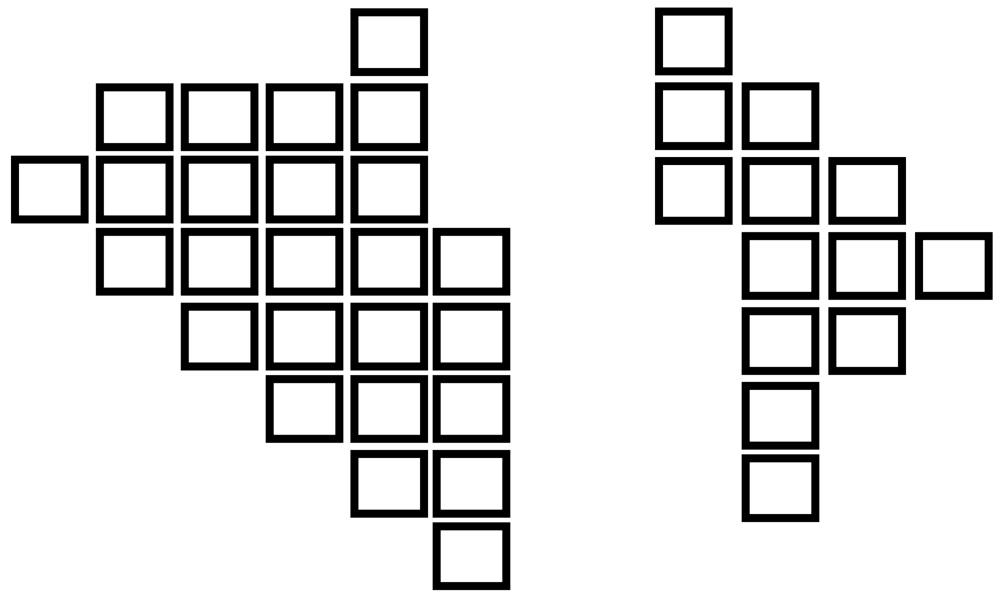
Real-time graphics primitives (entities)



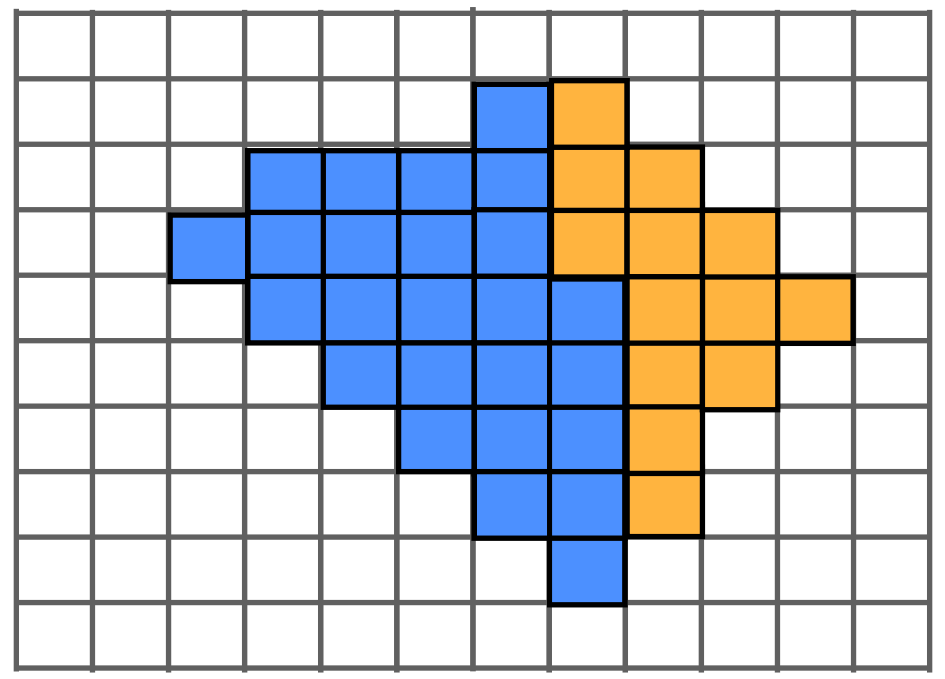
Vertices
(points in space)



Primitives
(e.g., triangles, points, lines)



Fragments



Pixels (in an image)

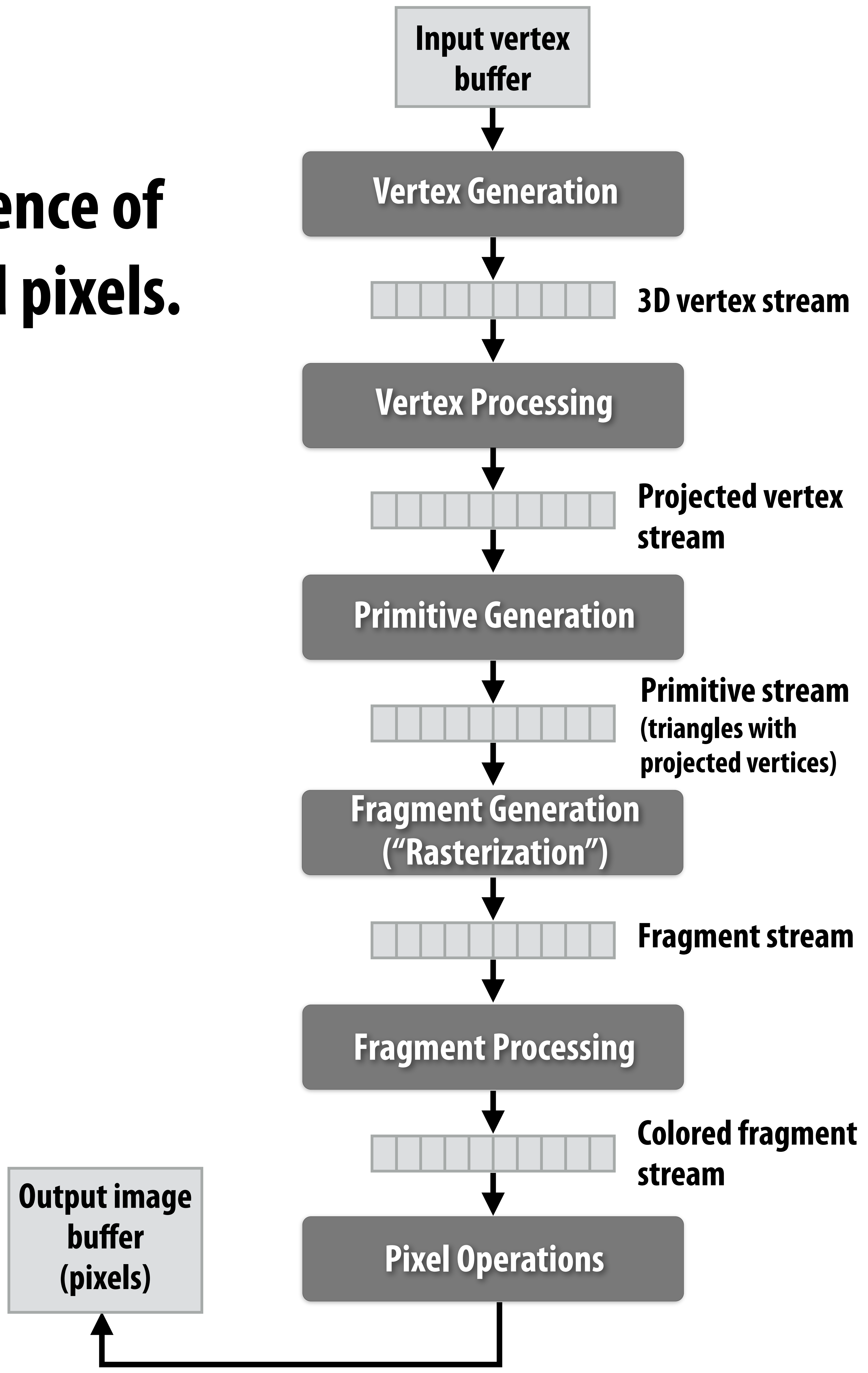
How to explain "a system"

- Step 1: describe the <u>things</u> (key entities) that are manipulated
  - The nouns

- Step 2: describe the operations the system performs on these entities
  - The verbs

Real-time graphics pipeline

- Abstracts process of rendering a picture as a sequence of operations on vertices, primitives, fragments, and pixels.



Tip 6

Surprises* are almost always bad:

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

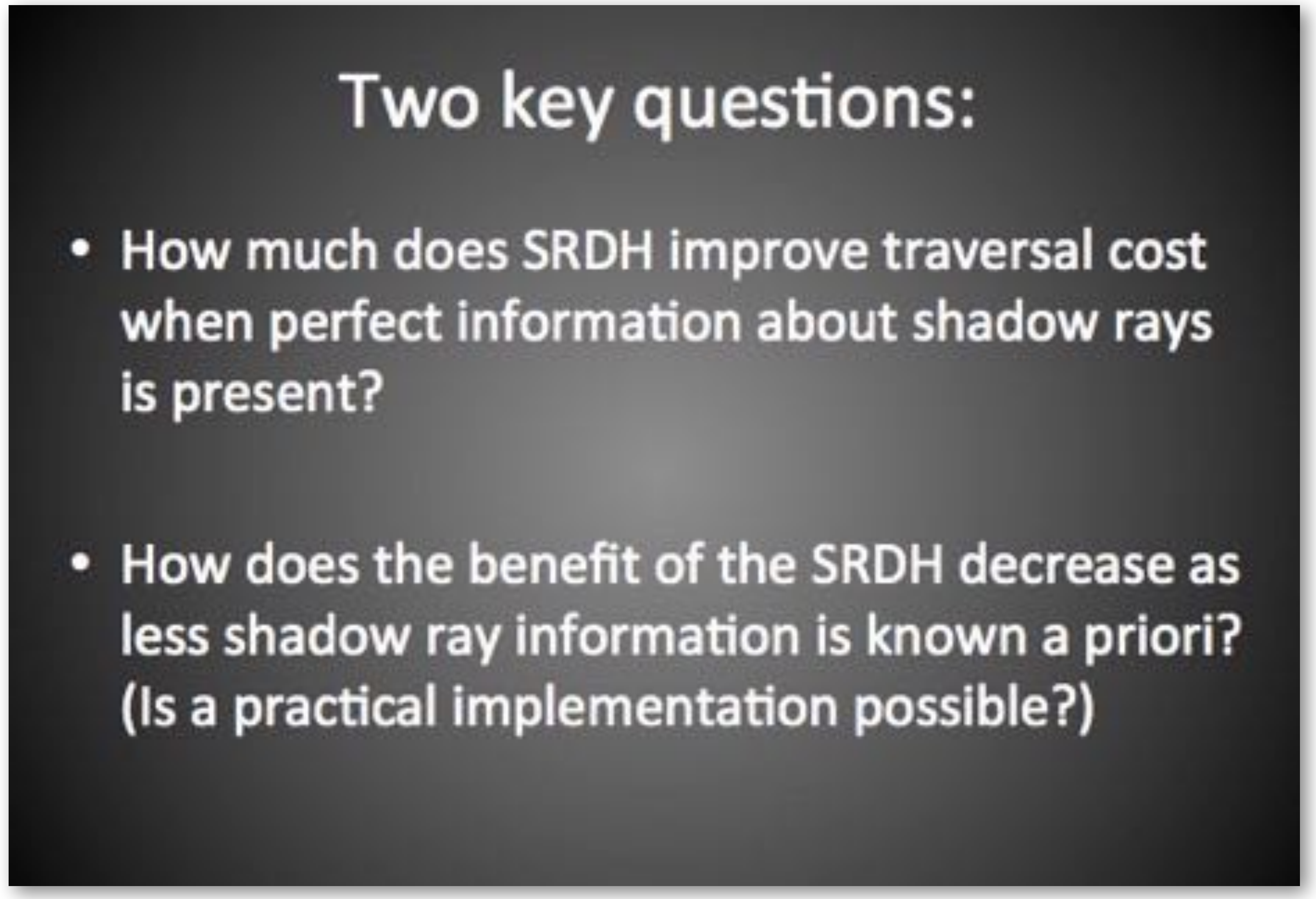
*** I am referring to surprises in talk narrative and/or exposition. A surprising result is great.**

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

- **Why 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 framework?")

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


Two key questions:

- How much does SRDH improve traversal cost when perfect information about shadow rays is present?
- How does the benefit of the SRDH decrease as less shadow ray information is known a priori? (Is a practical implementation possible?)

Big surprises in a narrative are a bad sign

- Ideally, you want the audience to always be able to <u>anticipate</u>* what you are about to say
  - This means: your story is so clear it's obvious!
  - It also means the talk is really easy to present without notes or text on slides (it just flows)

- If you are practicing your talk, and you keep forgetting what's coming on the next slide (that is, you can't anticipate it)...
  - This means: you probably need to restructure your talk because a clear narrative is not there.
  - It's not even obvious to you! Ouch!

* Credit to Abhinav Gupta for suggesting the term anticipation, and for the example on this slide

Tip 7

Show, don’t tell

It’s much easier to communicate with figures/images than text

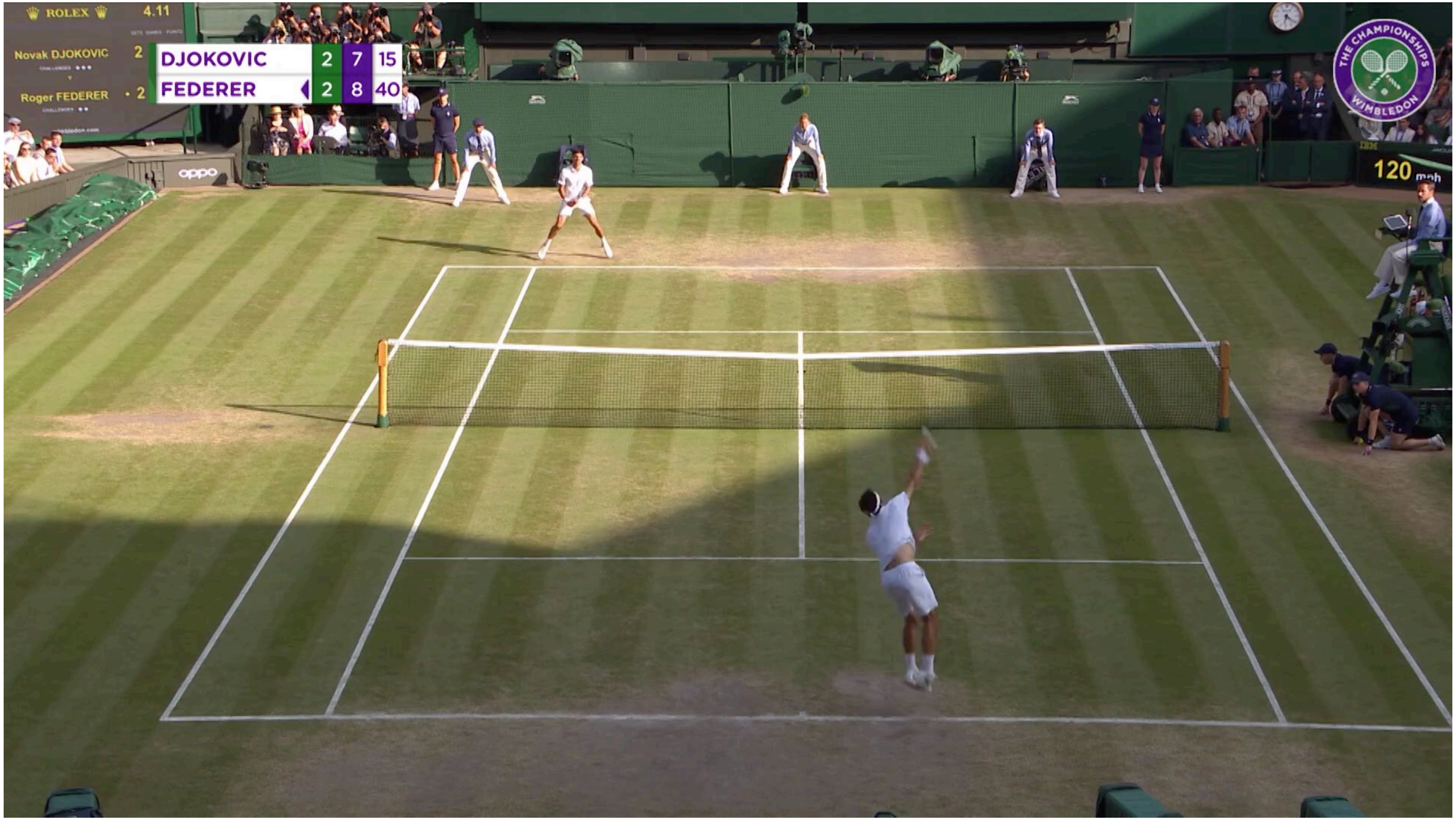
(And it saves the speaker a lot of work explaining… you can just describe the picture)

Example:

- In a recent project, we asked the question… given enough video of tennis matches of a professional athlete, could we come up with an algorithm for turning all this input video into a controllable video game character?

Compare the description above to the following sequence…

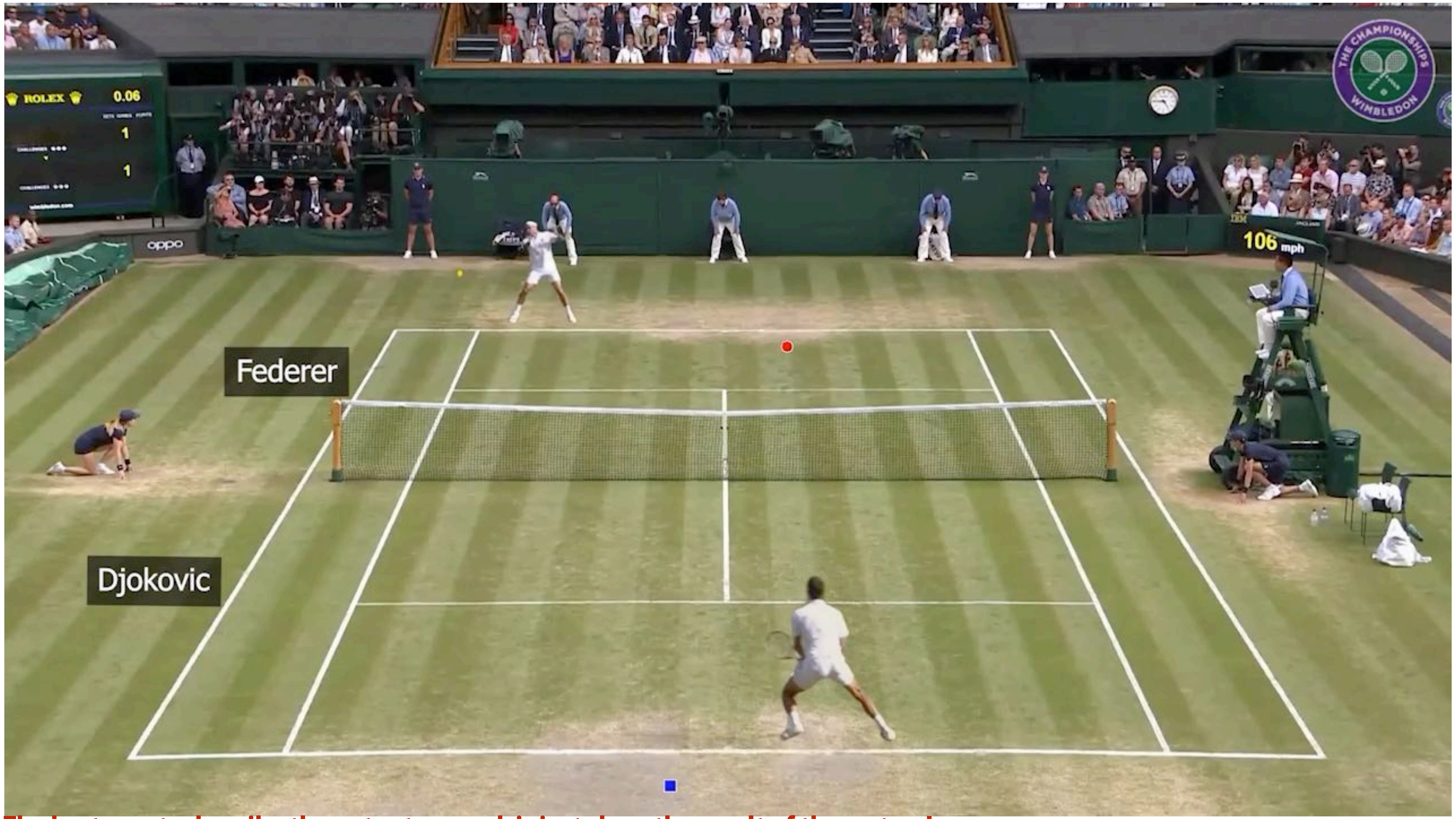
Here's an example of that source video



The best way to describe the input data is just show it! ("This is what the input looks like!")

And there's a lot of it out there!

And here's an example of controllable output



The best way to describe the output we seek is just show the result of the system!
("We click to specify a target ball location, and the player hits the incoming ball back to the red dot")

Another example:

The problem (lighting differences)

After the fix

Another example: we recently created a renderer that achieved high frame rates by rendering many views of the scene at the same time

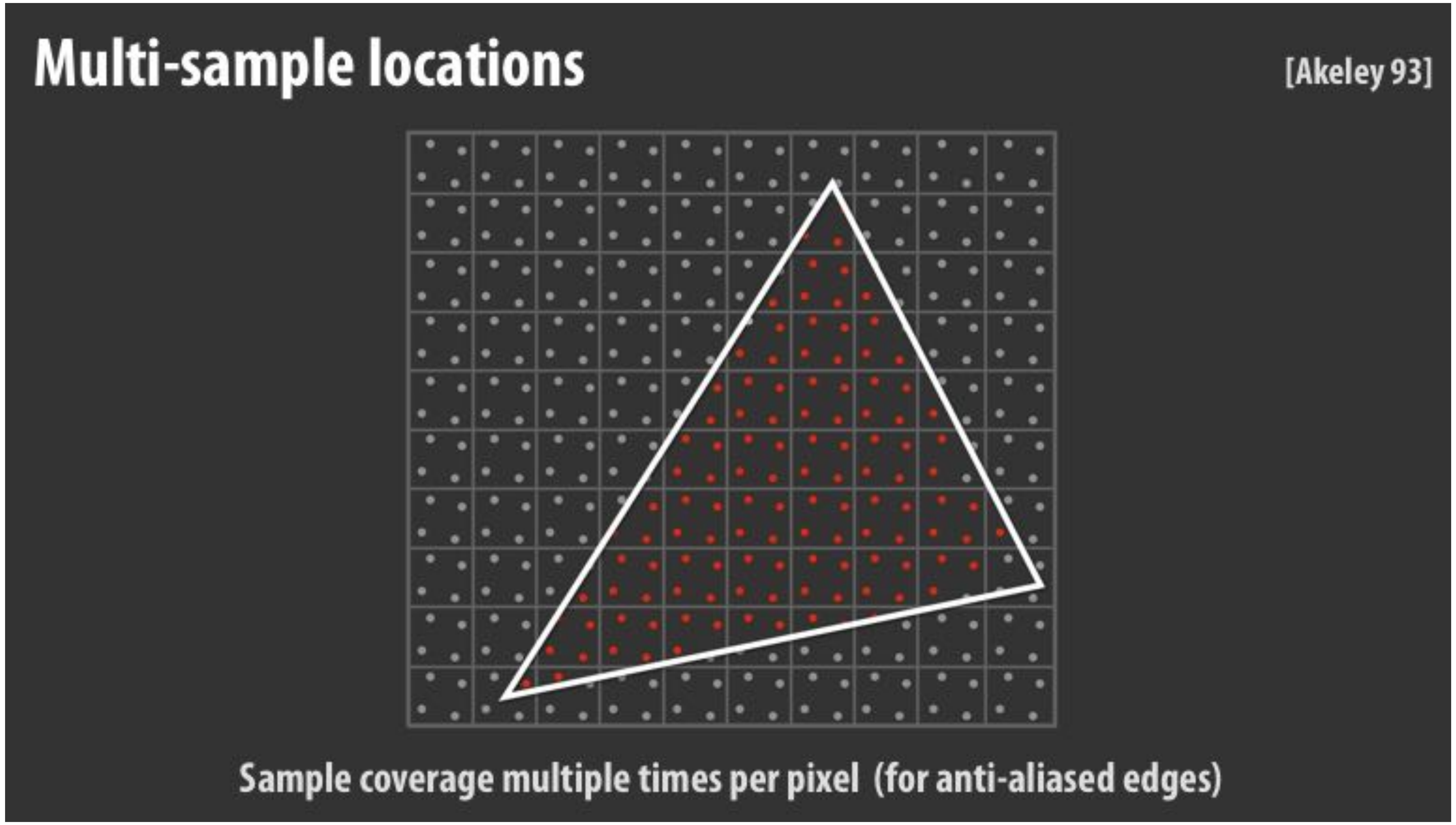
Tip 8

Always, always, always explain any figure or graph

(remember, the audience does not want to think about things you can tell them)

Explain every figure

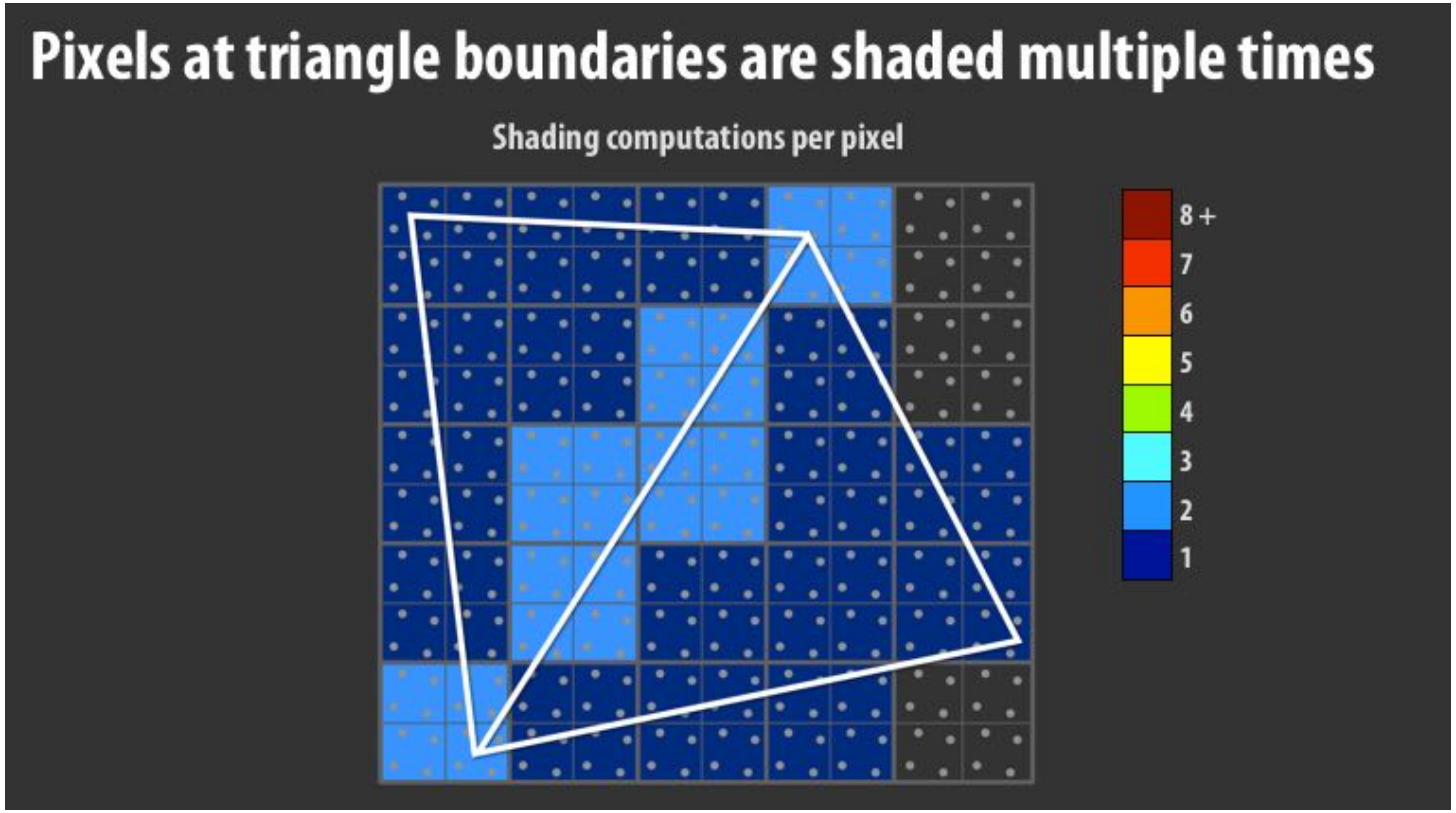
- Explain every visual element in the figure (never make the audience decode a figure)
- Refer to highlight colors explicitly (explain why the visual element is highlighted)


Multi-sample locations
[Akeley 93]
Sample coverage multiple times per pixel (for anti-aliased edges)

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 am following this talk!"

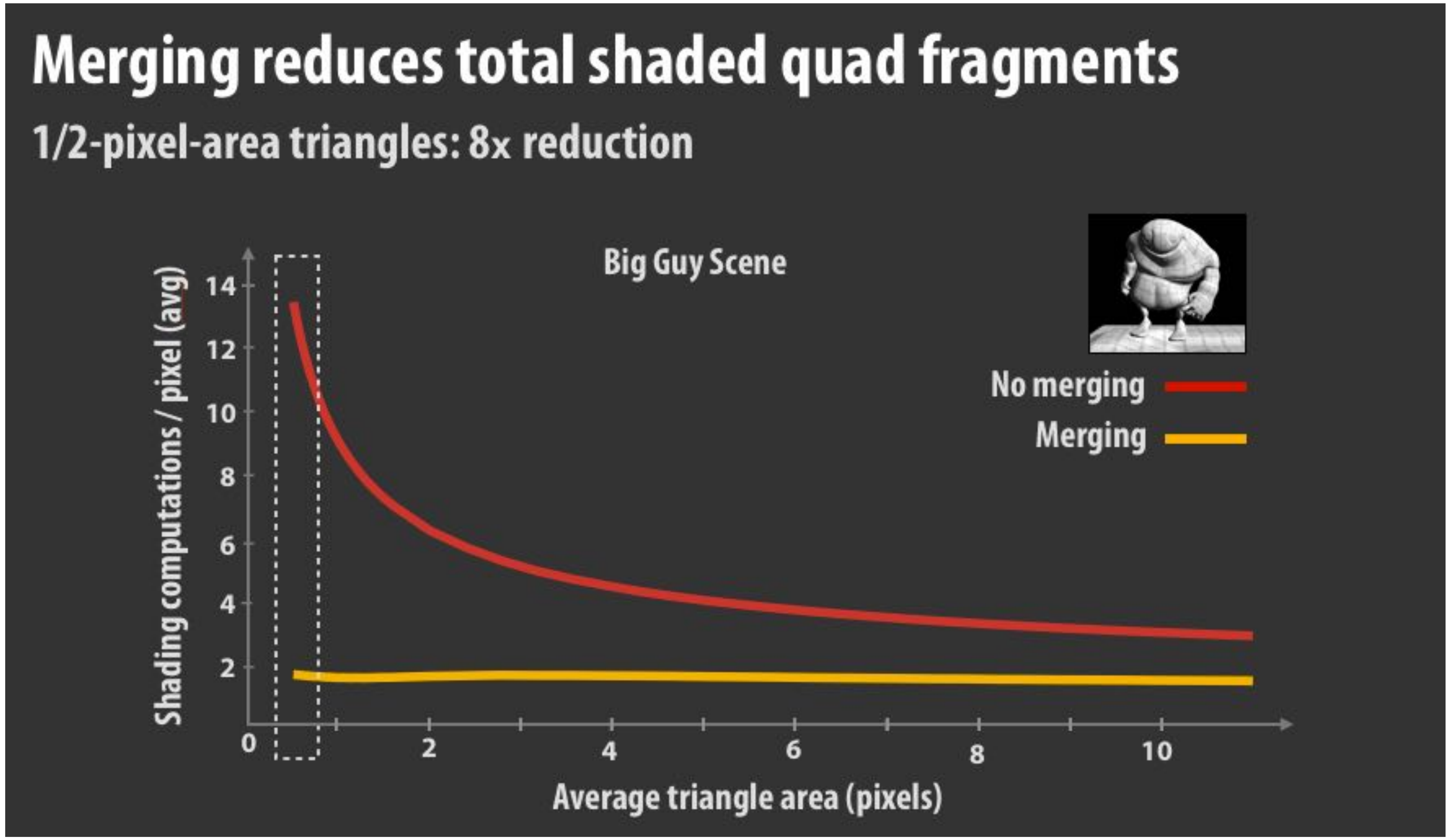


Pixels at triangle boundaries are shaded multiple times
Shading computations per pixel
8+
7
6
5
4
3
2
1

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. <u>As you can see</u> from the light blue region, pixels near the boundary of the two triangles get shaded twice."

Explain every results graph

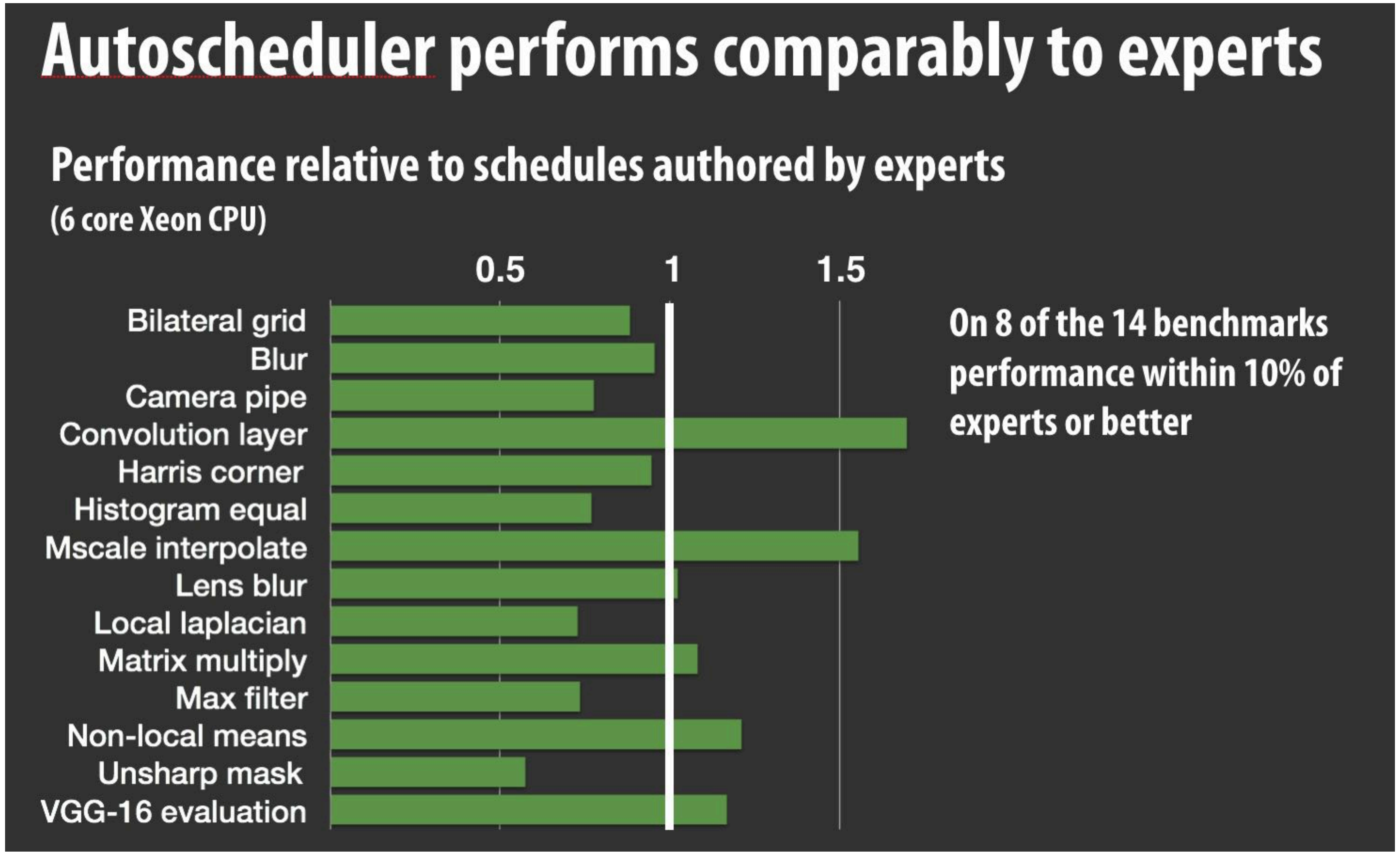
- May start with a general intro of what the graph will address (so audience "anticipates" the result)
- Then describe the axes (and your axes better have labels!)
- Then describe the <u>one point</u> that you wish to make with this results slide (more on this later!)



Example voice over: "Our first questions were about performance: how much did the algorithm reduce the number of the shading computations? 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.

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 one point that you wish to make with this results slide (more on this later!)



Example voice over: "Our first question was about performance: how fast is the auto scheduler compared to experts? And we found out that it's quite good. This figure plots the performance of the autoscheduler compared to that of expert code. So expert code is 1. Faster code is to the right. As you can see, the auto scheduler is within 10% of the performance of the experts in many cases, and always within a factor of 2.

Tip 9

In the results section:

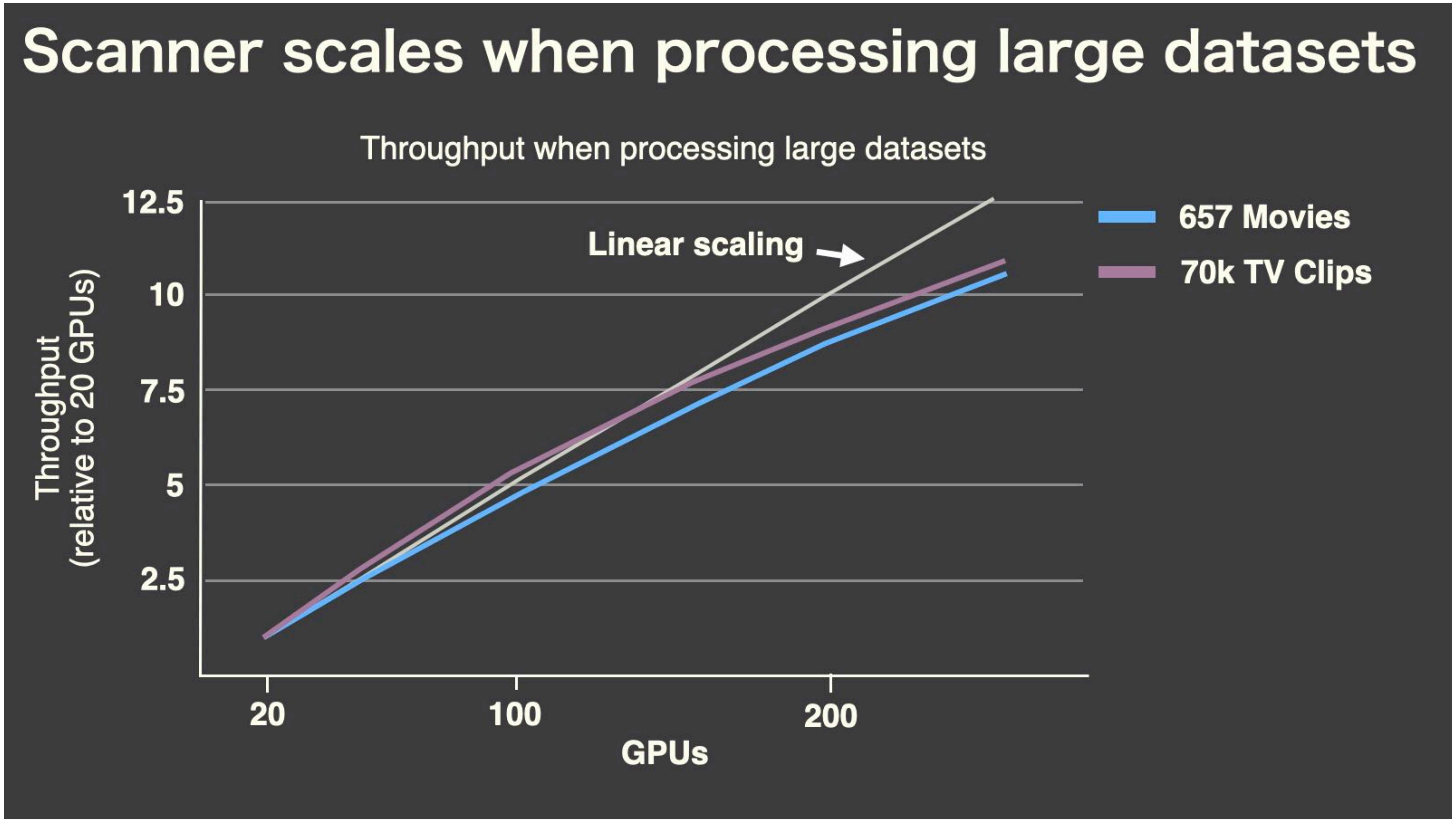
One point per slide!

One point per slide!

One point per slide!

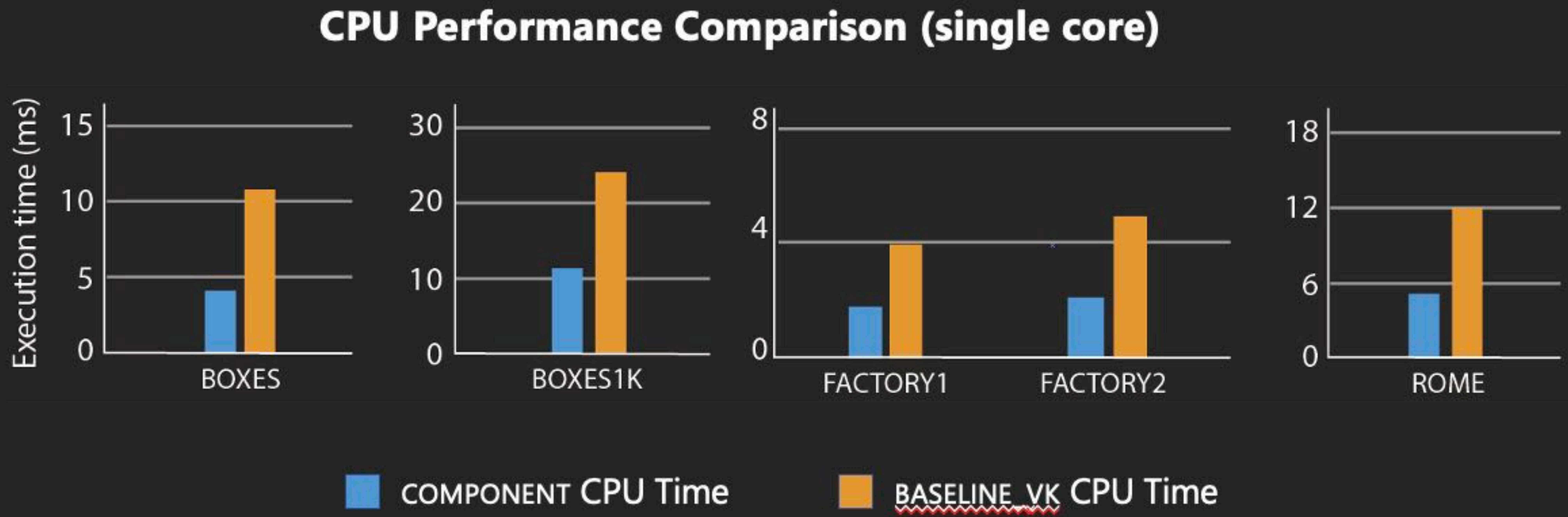
(and the point is the title of the slide!!!)

- **Make the point of the graph the slide's title:**
  - **It provides audience context for interpreting the graph ("Let me see if I can verify that point in the graph to check my understanding")**
  - **Another example of the "audience prefers not to think" principle**



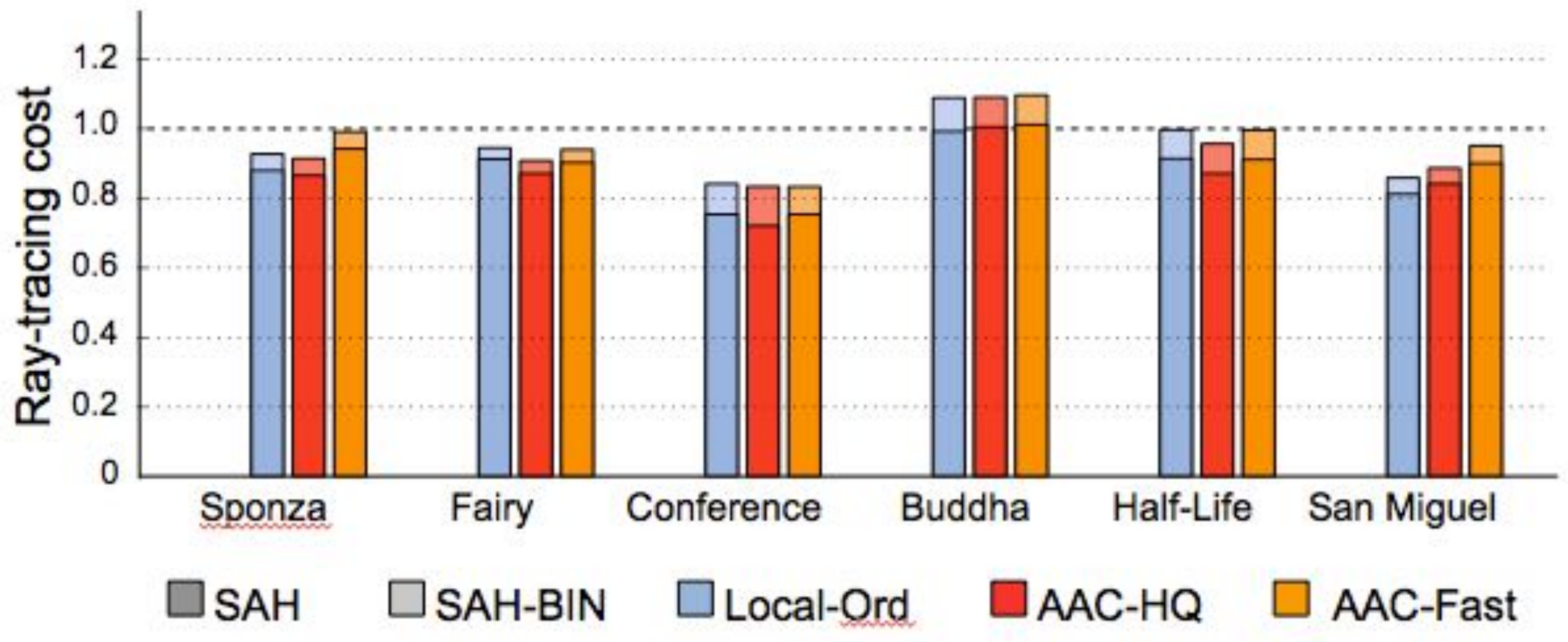
Another example:

The COMPONENTS renderer uses 2x less CPU time than BASELINE_VK

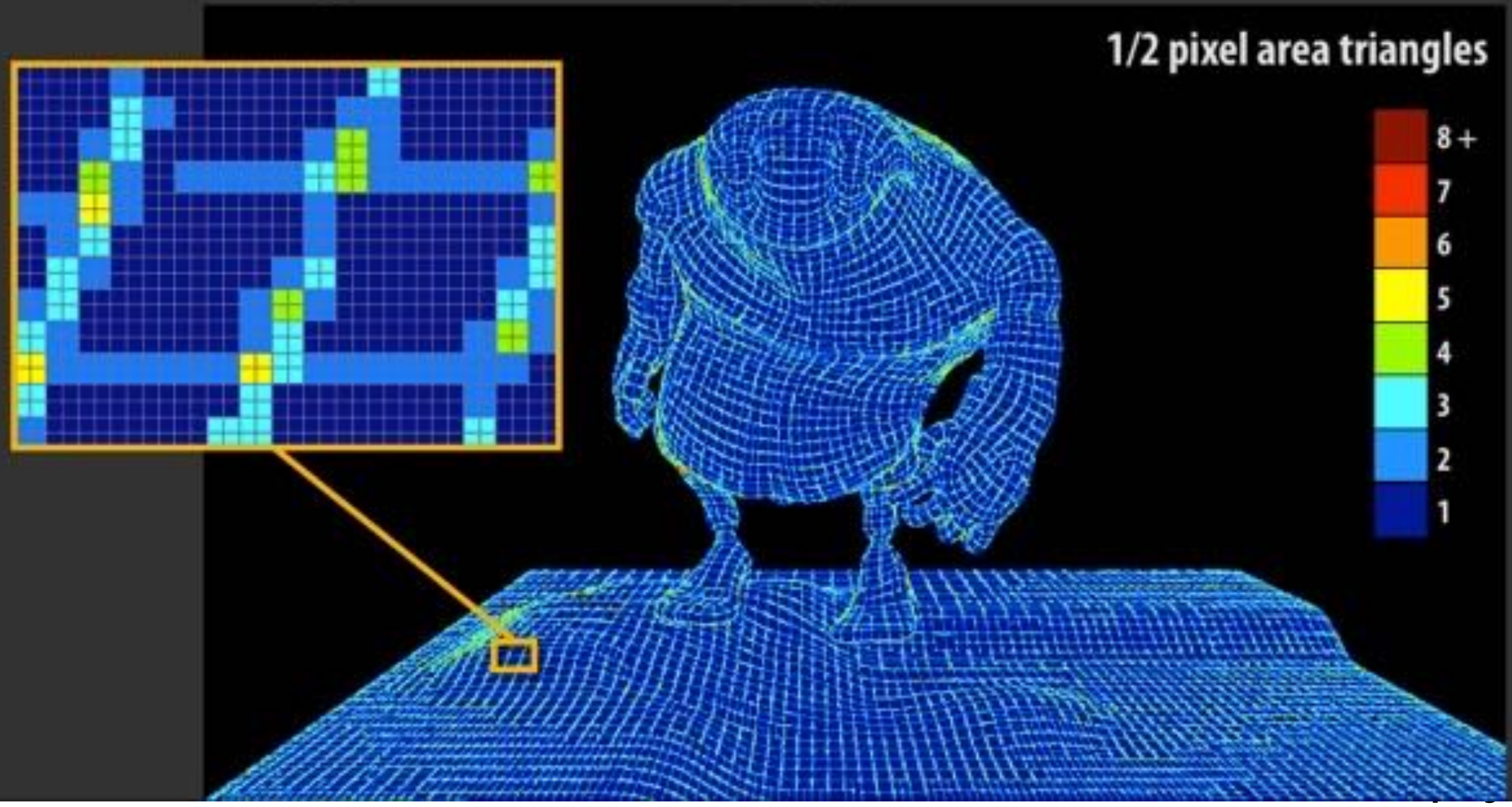


More examples

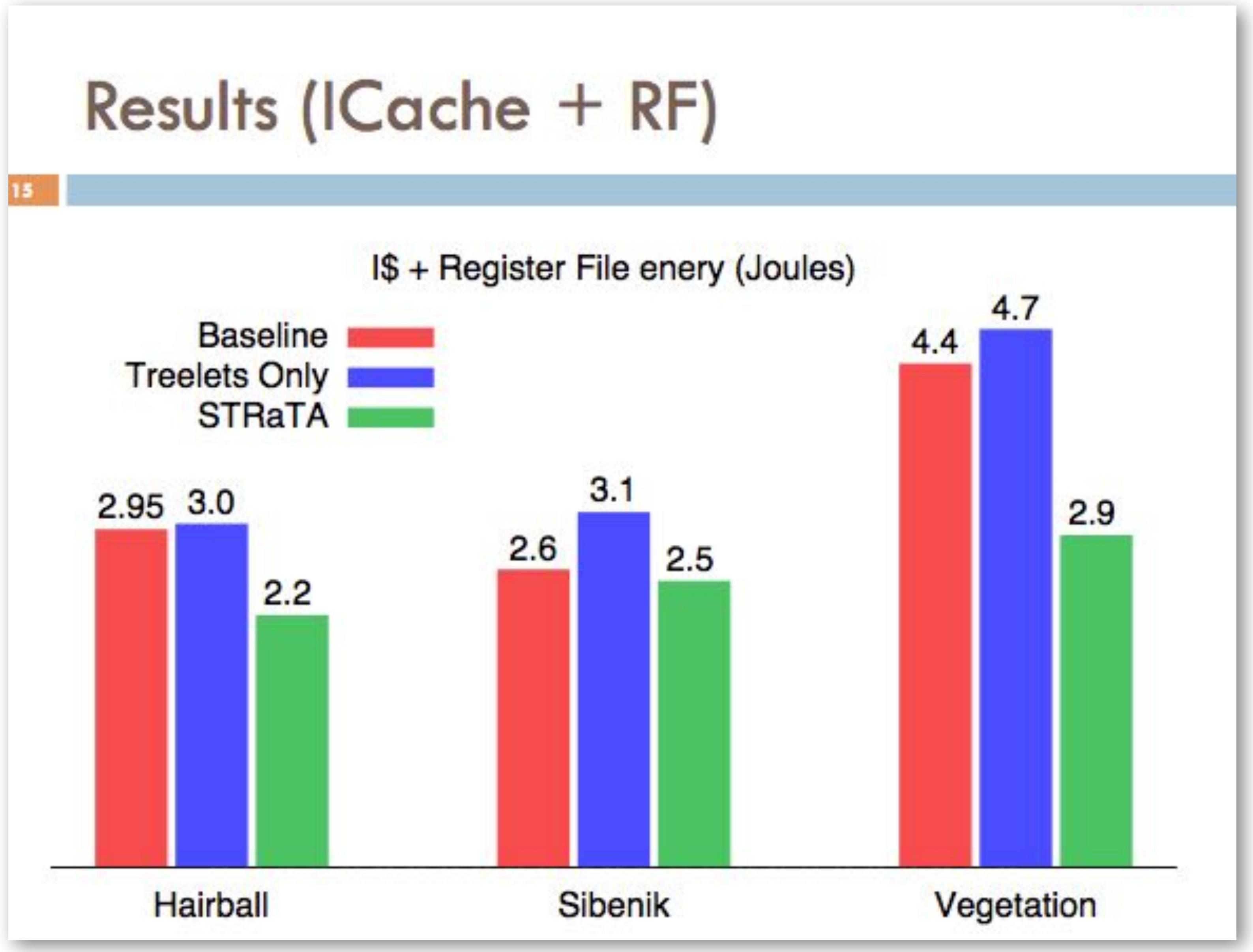
AAC-Fast produces BVHs with equal or lower cost than the **full sweep build** in all cases except Buddha.

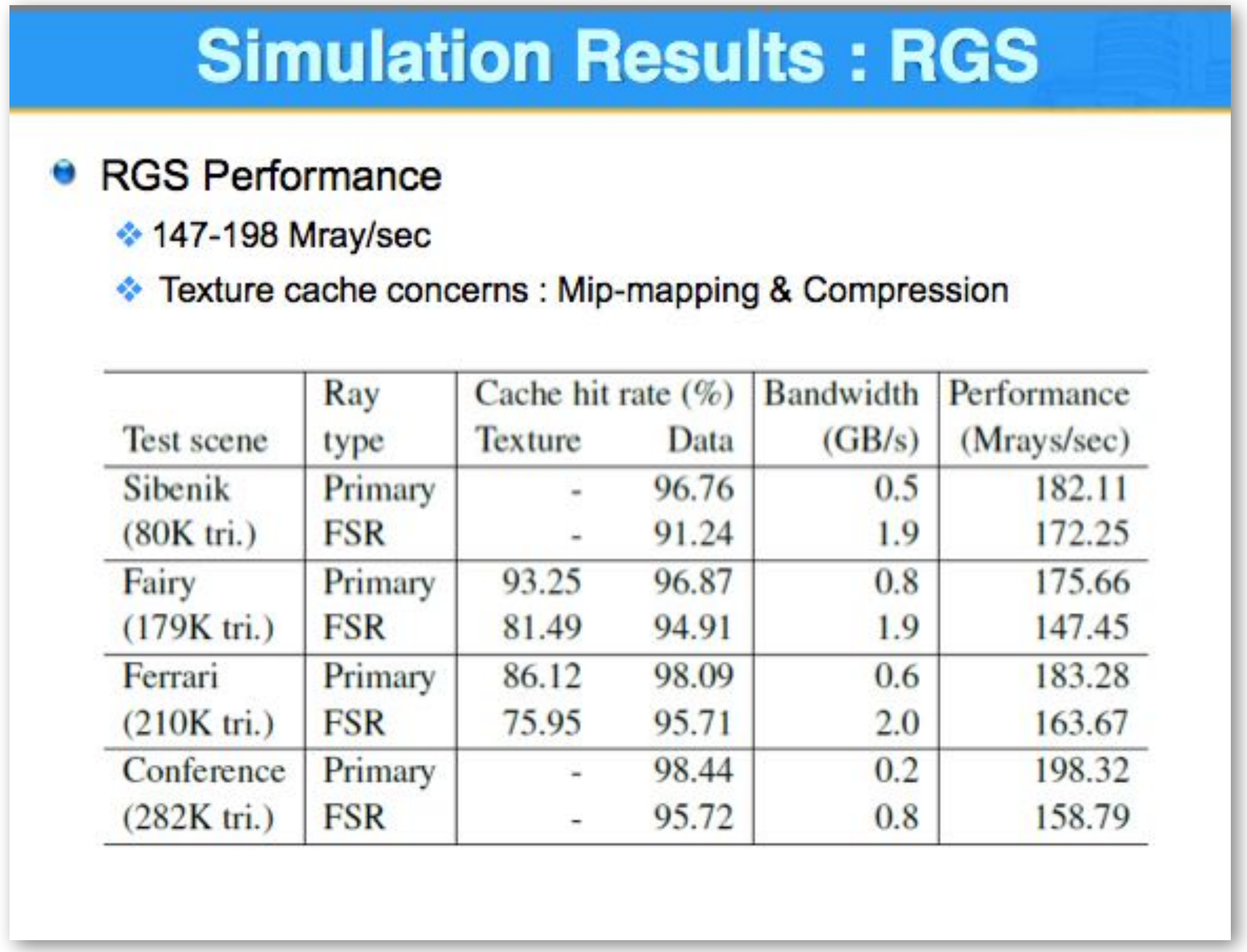


Extra shading occurs at merging window boundaries



Bad examples of results slides





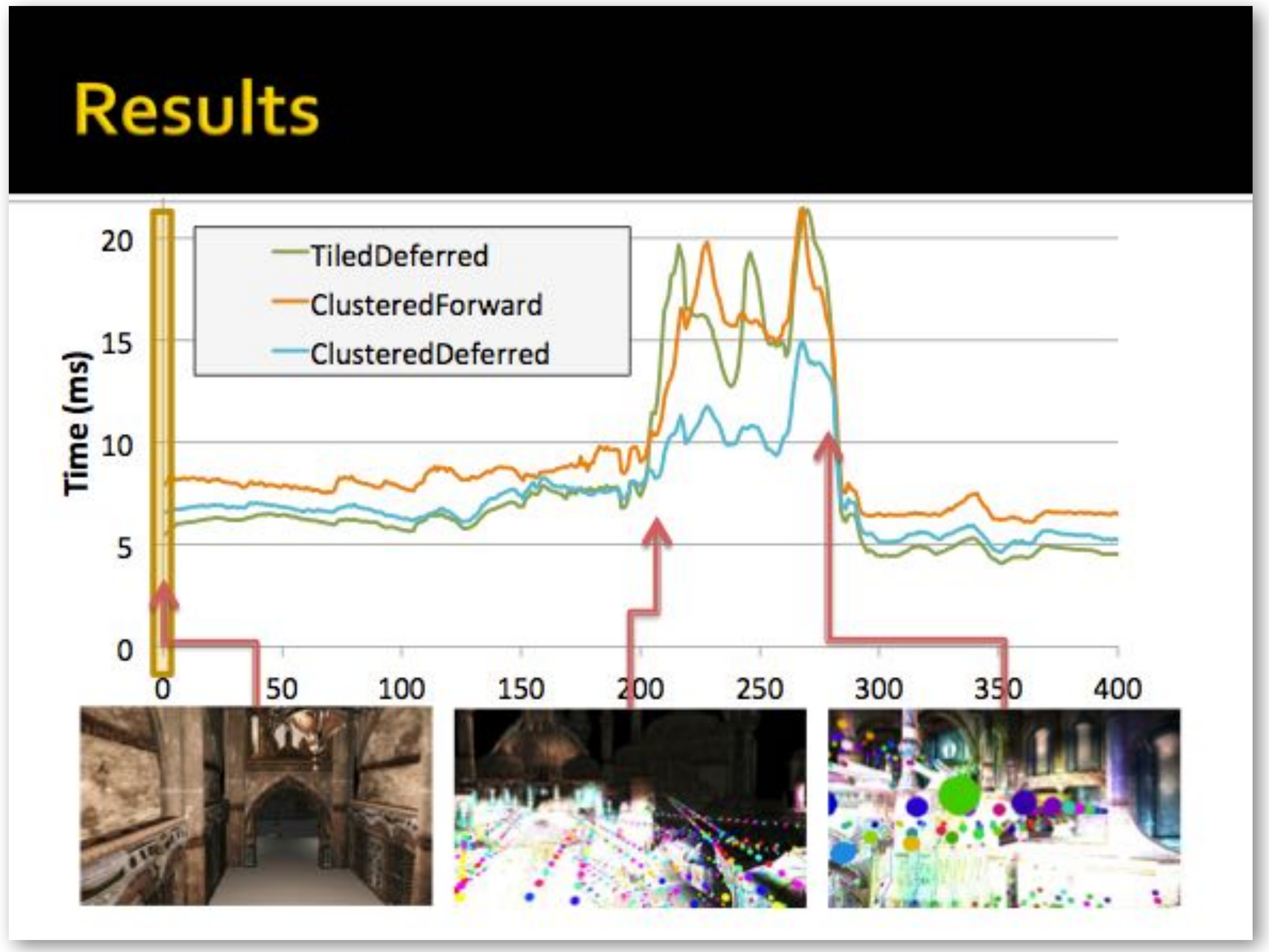
Simulation Results : RGS

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

Test scene	Ray type	Cache hit rate (%)		Bandwidth	Performance
		Texture	Data	(GB/s)	(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

- Notice how you (as an audience member) are working hard to interpret the trends in these graphs
  - You are asking: what do these results say?
  - What am I supposed to be concluding?

- The audience just wants to be told what to look for!
  - They are reading the graphs to verify the main point, not determine the main point.



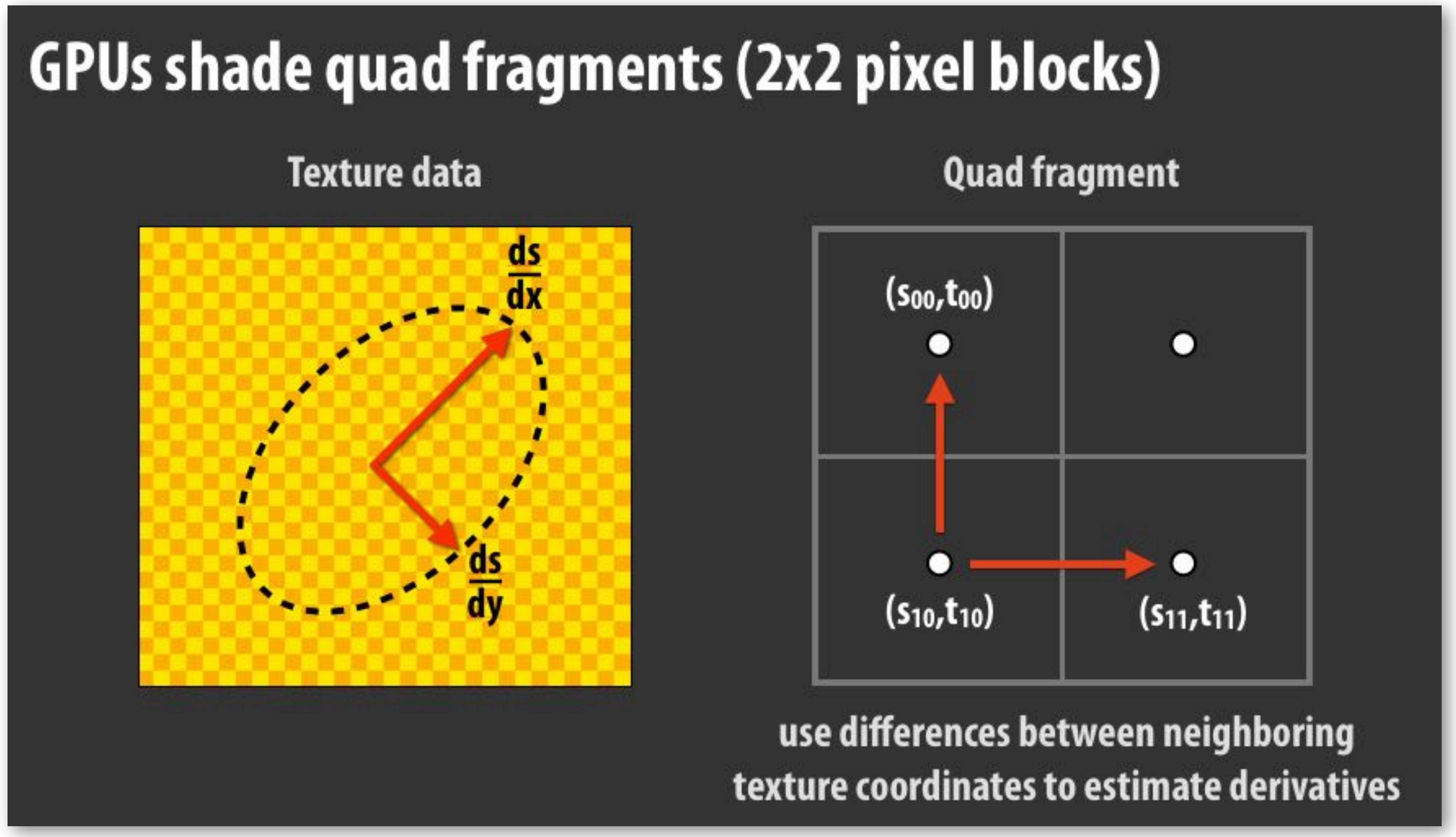
Tip 10

Titles matter

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

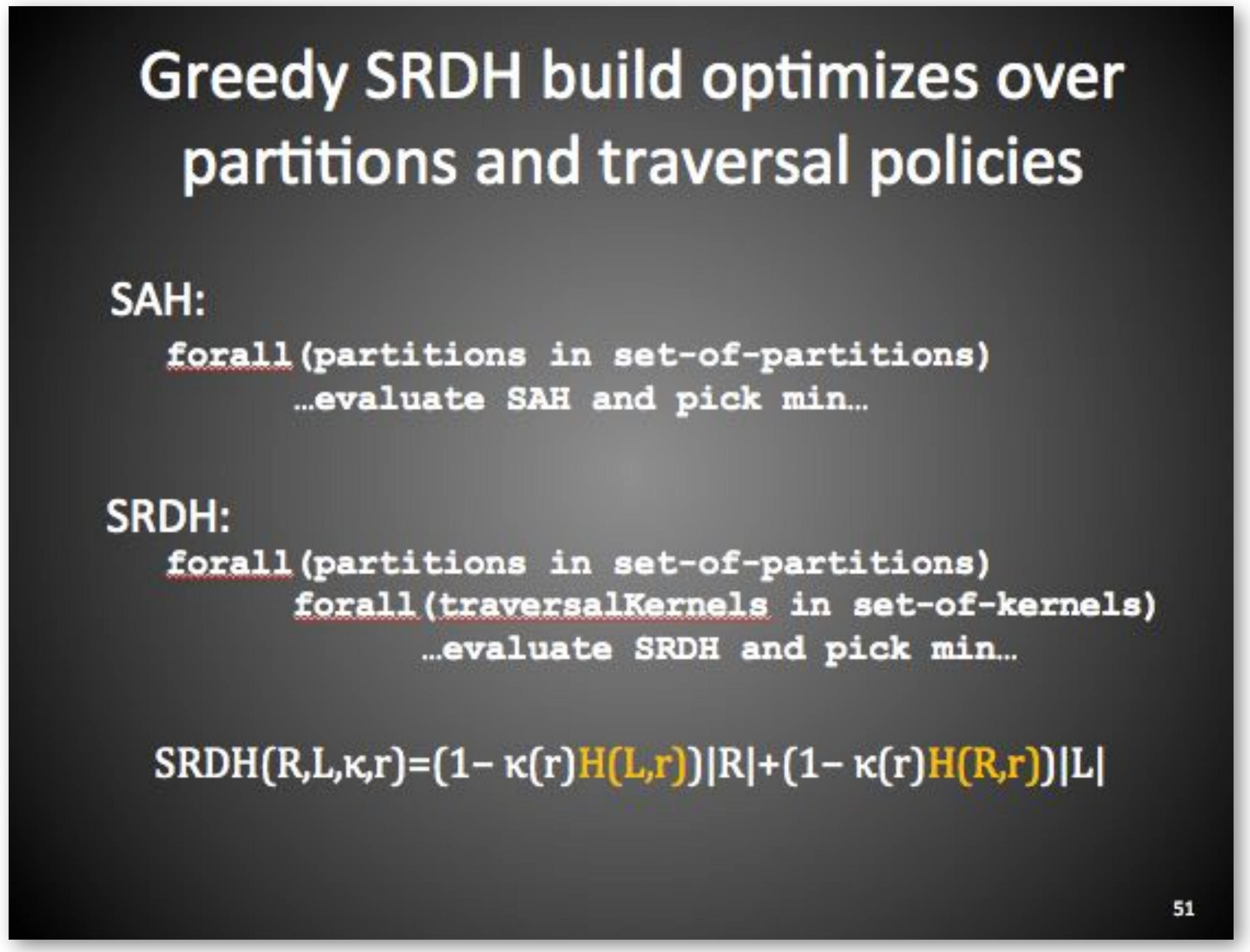
(basically, this is “one-point-per-slide” for the whole talk)

Examples of descriptive 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:

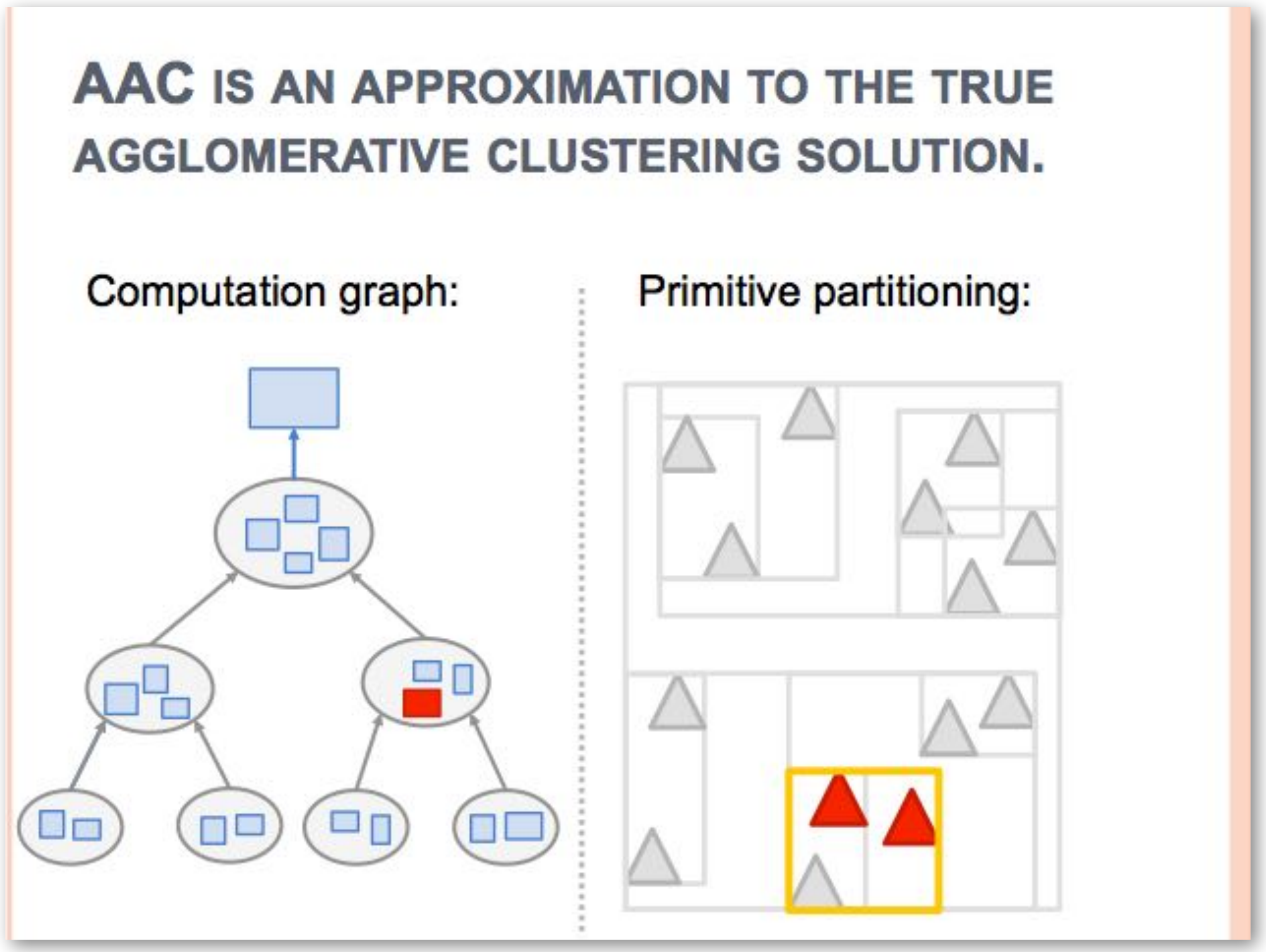
```
forall (partitions in set-of-partitions)
    …evaluate SAH and pick min…
```

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.

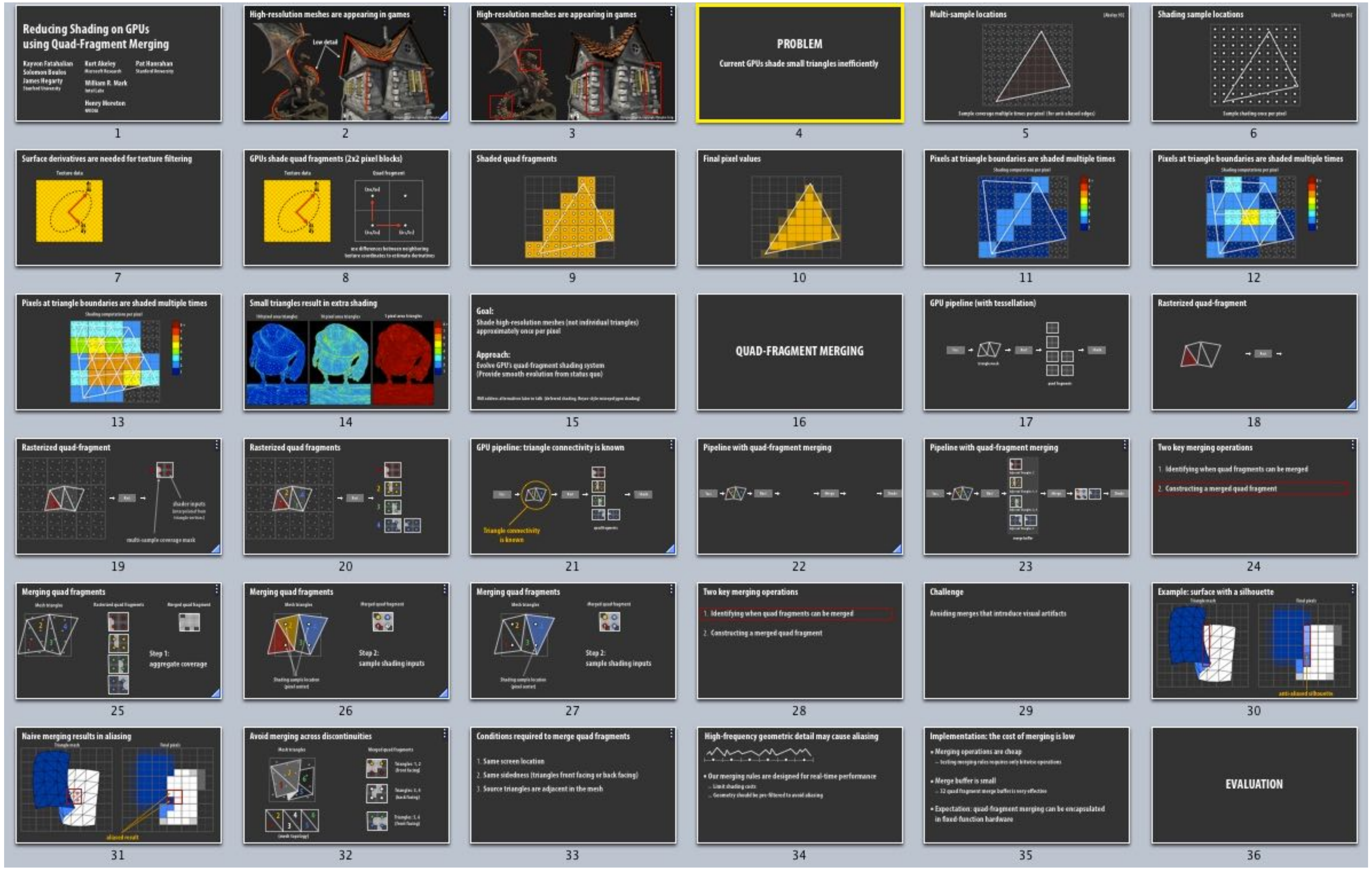
The reason for meaningful slide titles is convenience and clarity for the audience

“Why is the speaker telling me this again?”

(Recall “why before what”)

Read your slide titles in thumbnail view

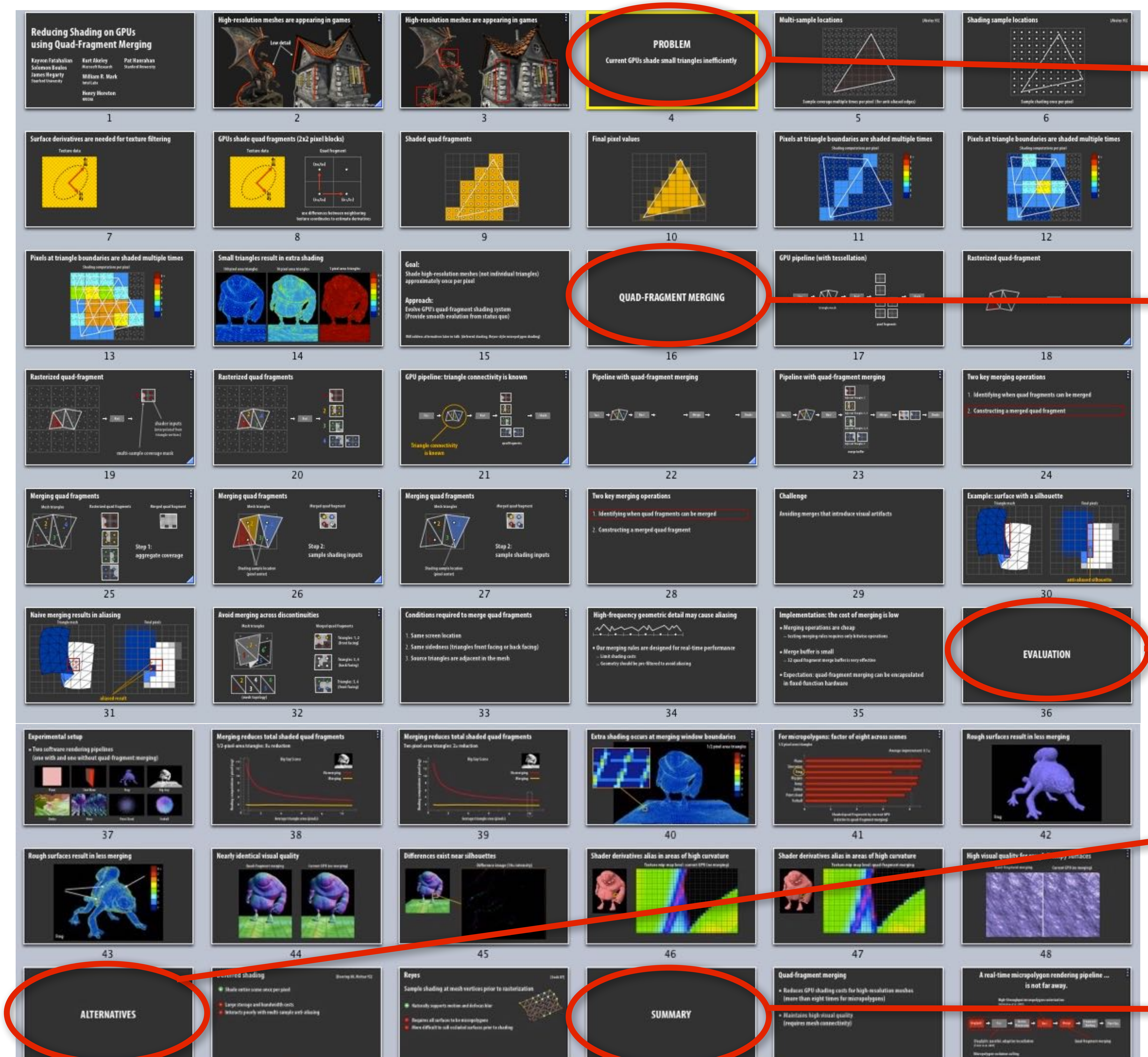
Do they make all the points of the story you are trying to tell?



1. Reducing Shading on GPUs using Quad-Fragment Merging
2. High-resolution meshes are appearing in games
3. High-resolution meshes are appearing in games
4. PROBLEM — Current GPUs shade small triangles inefficiently
5. Multi-sample locations
6. Shading sample locations
7. Surface derivatives are needed for texture filtering
8. GPUs shade quad fragments (2x2 pixel blocks)
9. Shaded quad fragments
10. Final pixel values
11. Pixels at triangle boundaries are shaded multiple times
12. Pixels at triangle boundaries are shaded multiple times
13. Pixels at triangle boundaries are shaded multiple times
14. Small triangles result in extra shading
15. Goal: Shade high-resolution meshes (not individual triangles) approximately once per pixel. Approach: Evolve GPU's quad-fragment shading system (Provide smooth evolution from status quo)
16. QUAD-FRAGMENT MERGING
17. GPU pipeline (with tessellation)
18. Rasterized quad-fragment
19. Rasterized quad-fragment
20. Rasterized quad fragments
21. GPU pipeline: triangle connectivity is known
22. Pipeline with quad-fragment merging
23. Pipeline with quad-fragment merging
24. Two key merging operations — 1. Identifying when quad fragments can be merged 2. Constructing a merged quad fragment
25. Merging quad fragments
26. Merging quad fragments
27. Merging quad fragments
28. Two key merging operations — 1. Identifying when quad fragments can be merged 2. Constructing a merged quad fragment
29. Challenge — Avoiding merges that introduce visual artifacts
30. Example: surface with a silhouette
31. Naive merging results in aliasing
32. Avoid merging across discontinuities
33. Conditions required to merge quad fragments — 1. Same screen location 2. Same sidedness (triangles front facing or back facing) 3. Source triangles are adjacent in the mesh
34. High-frequency geometric detail may cause aliasing — Our merging rules are designed for real-time performance
35. Implementation: the cost of merging is low — Merging operations are cheap; Merge buffer is small; Expectation: quad-fragment merging can be encapsulated in fixed-function hardware
36. EVALUATION

Tip 11

Organize your talk with section slides


I'm about to frame the problem in my terms

Ok, let's talk about our solution

Let's take a look at how well it works

Of course there are some potential alternatives that we didn't go with

Let me wrap up

Stage your talk with section slides

- **Useful for your audience**
  - It provides guidance for <u>what</u> you hope to achieve next (no surprises!)
  - Compartmentalization: it's absolutely clear where the shifts are
  - If a listener got lost, it's a good place for them to re-engage
  - It's a place for the audience to take a breath
  - It gives the talk a more colloquial tone

- **Useful for you**
  - It's a chance for you to pause and take a breath
  - It's a great breakdown of the talk for practicing subsections

Tip 12

Practice the presentation

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
- Rehearse your presentation several times the night before (in front of a partner or friend)
    - It's only a short presentation, so a couple of practice runs are possible in a small amount of time

Thanks for being a great class.

Good luck working on your projects.

Email us for 1-on-1s if you need help.

See you in a week!

"Students working hard on final projects. Synthwave. Low lighting."