Lecture 1:

Introduction

Modern Parallel Computing

John Owens

EEC 289Q, UC Davis, Winter 2018

Bio

- John Owens <jowens@ece>, professor, ECE
  - Kemper 3175
  - Office hours: Th 9:30–10:20a (Coffee House)
- Current research:
  - GPU computing
    - Fundamentals: algorithms & data structures
    - Applications
    - Multi-GPU computing
  - Programmability of GPUs
  - GPUs for data center applications
- Taught Udacity CS344 to 90k+ students in 2012
- I take the train.

Instructional staff

- Jason Mak, jwmak@ucdavis, TA
- Possibly some lectures from Kerry Seitz, kaseitz@ucdavis, co-instructor in 2016

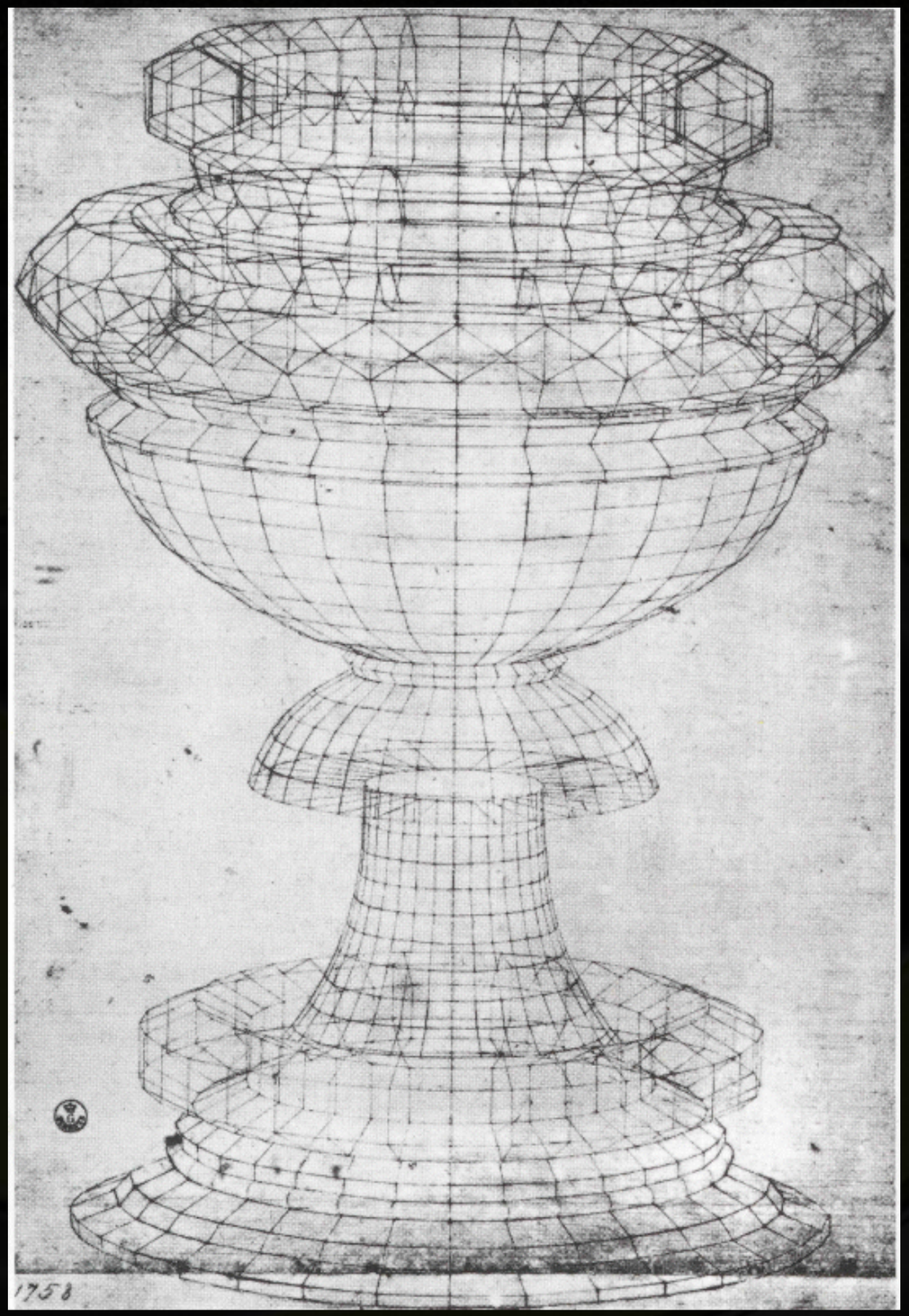
Second time we're teaching this

- Slides will be a little rough
- Assignments will also be rough
- Bear with us and give us good feedback!

Lecture 1: Introduction / Overview

- Intro & Motivation
- Class outline
- Administrivia

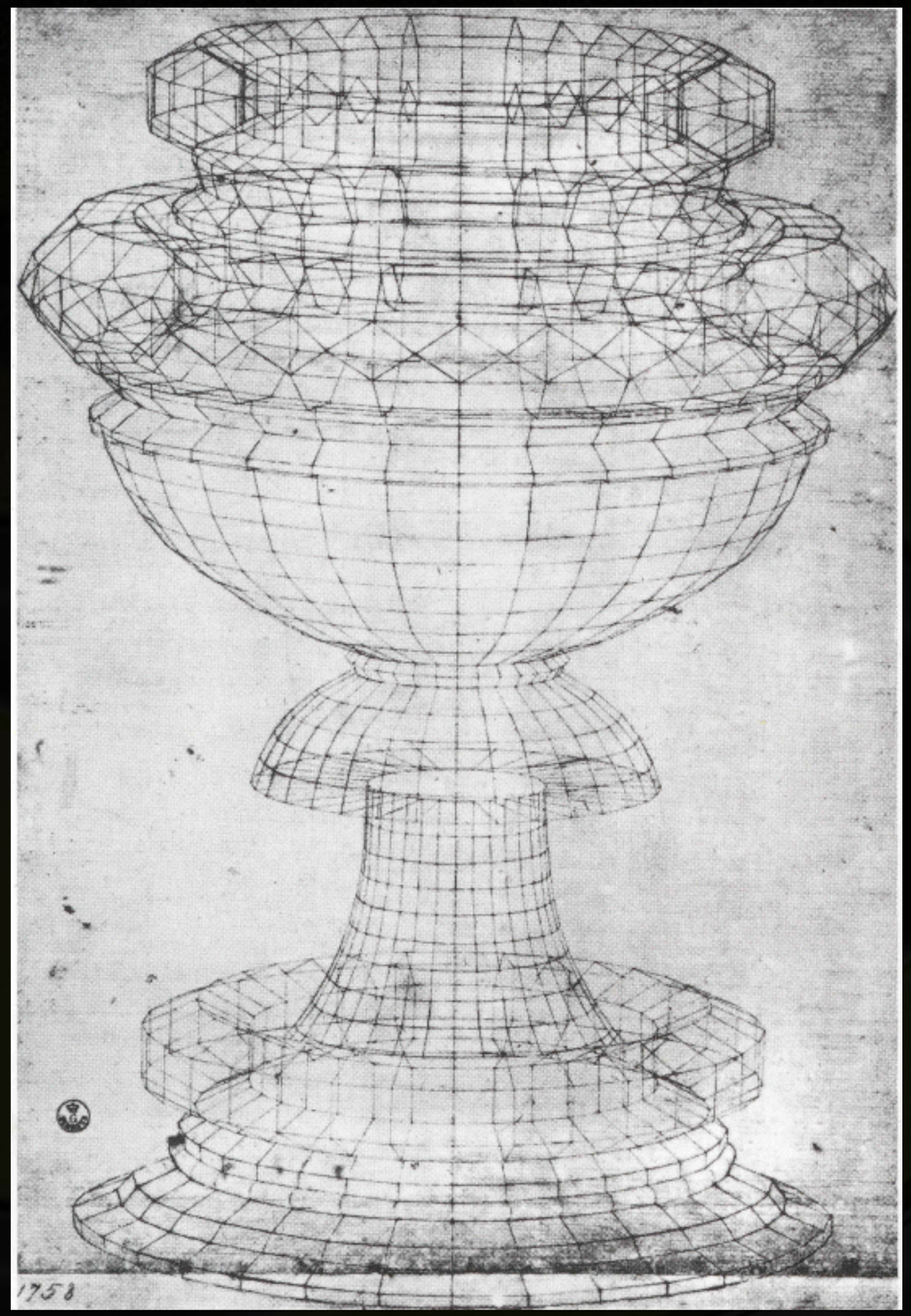
Early 3D Graphics

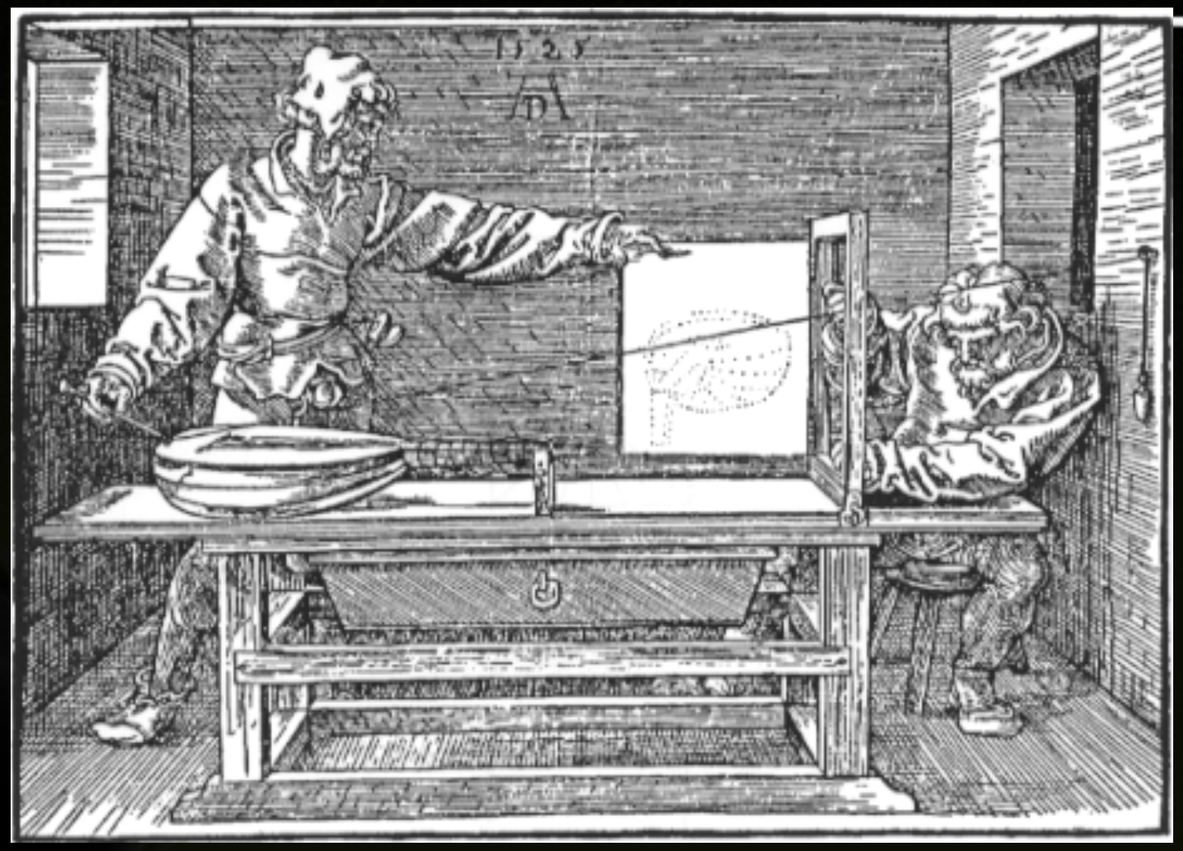


Thanks to Dave Luebke, NVIDIA, for these slides!

Perspective study of a chalice
Paolo Uccello, circa 1450

Early Graphics Hardware





Artist using a perspective machine
Albrecht Dürer, 1525

Perspective study of a chalice
Paolo Uccello, circa 1450

Early *Electronic* Graphics Hardware

SKETCHPAD: A Man-Machine Graphical Communication System
Ivan Sutherland, 1963

The Graphics Pipeline

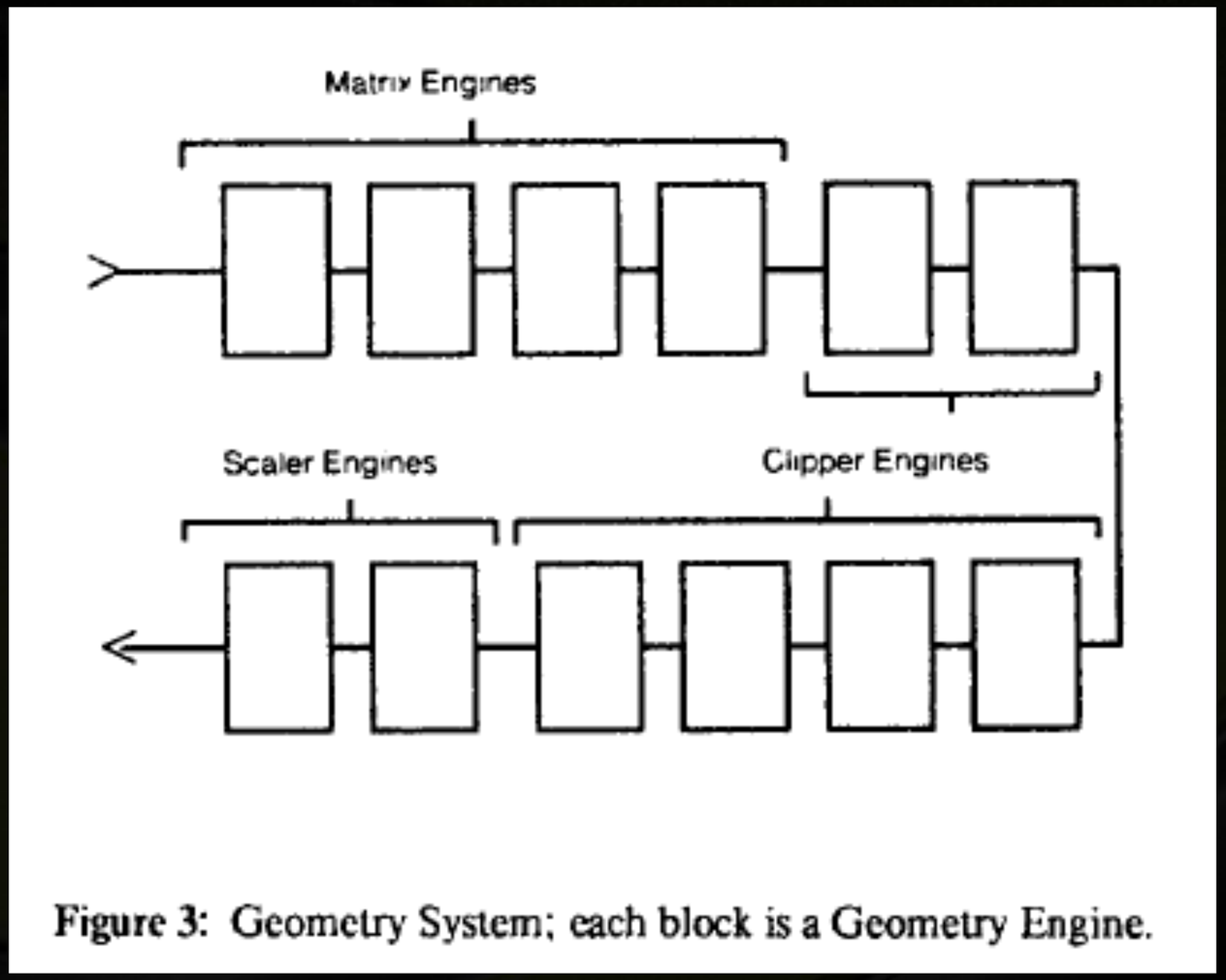


Figure 3: Geometry System; each block is a Geometry Engine.

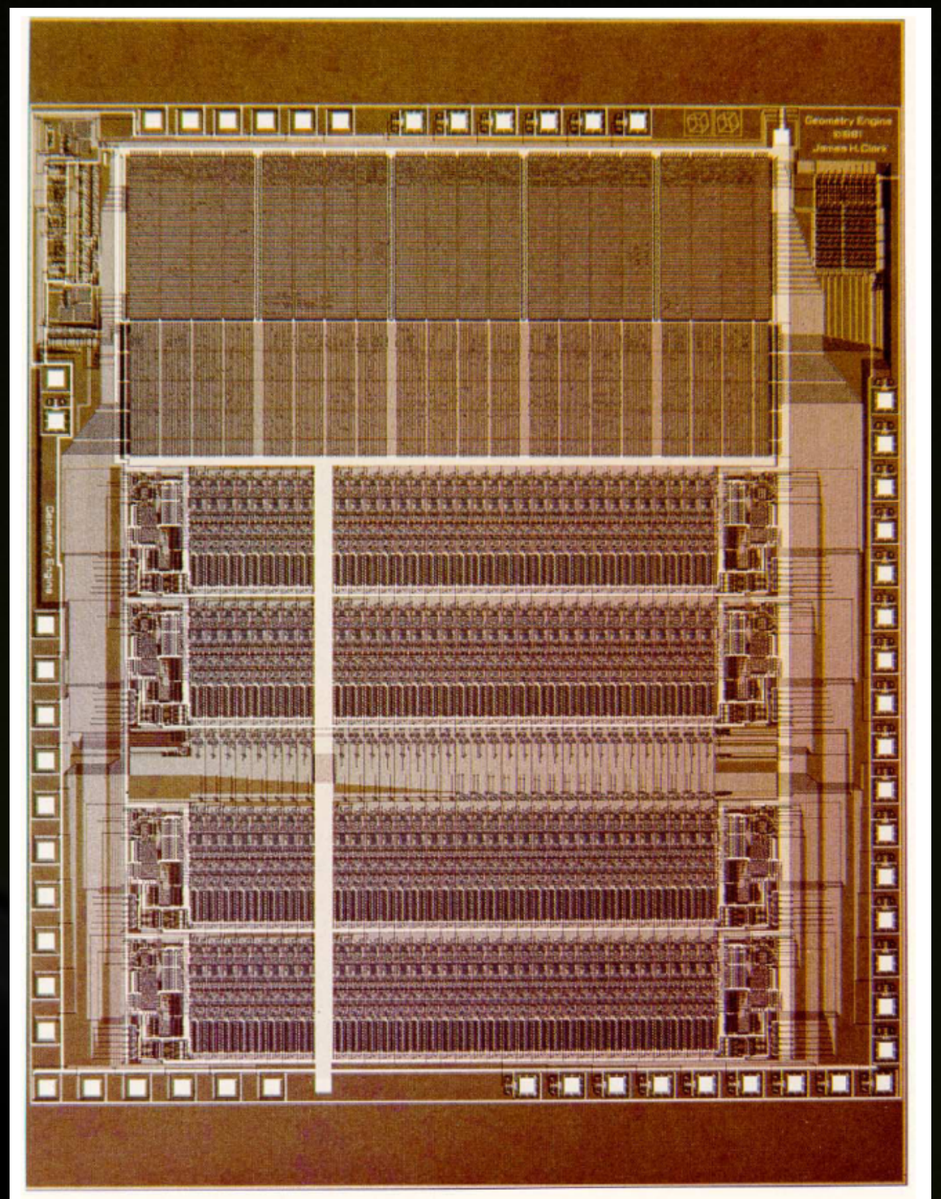
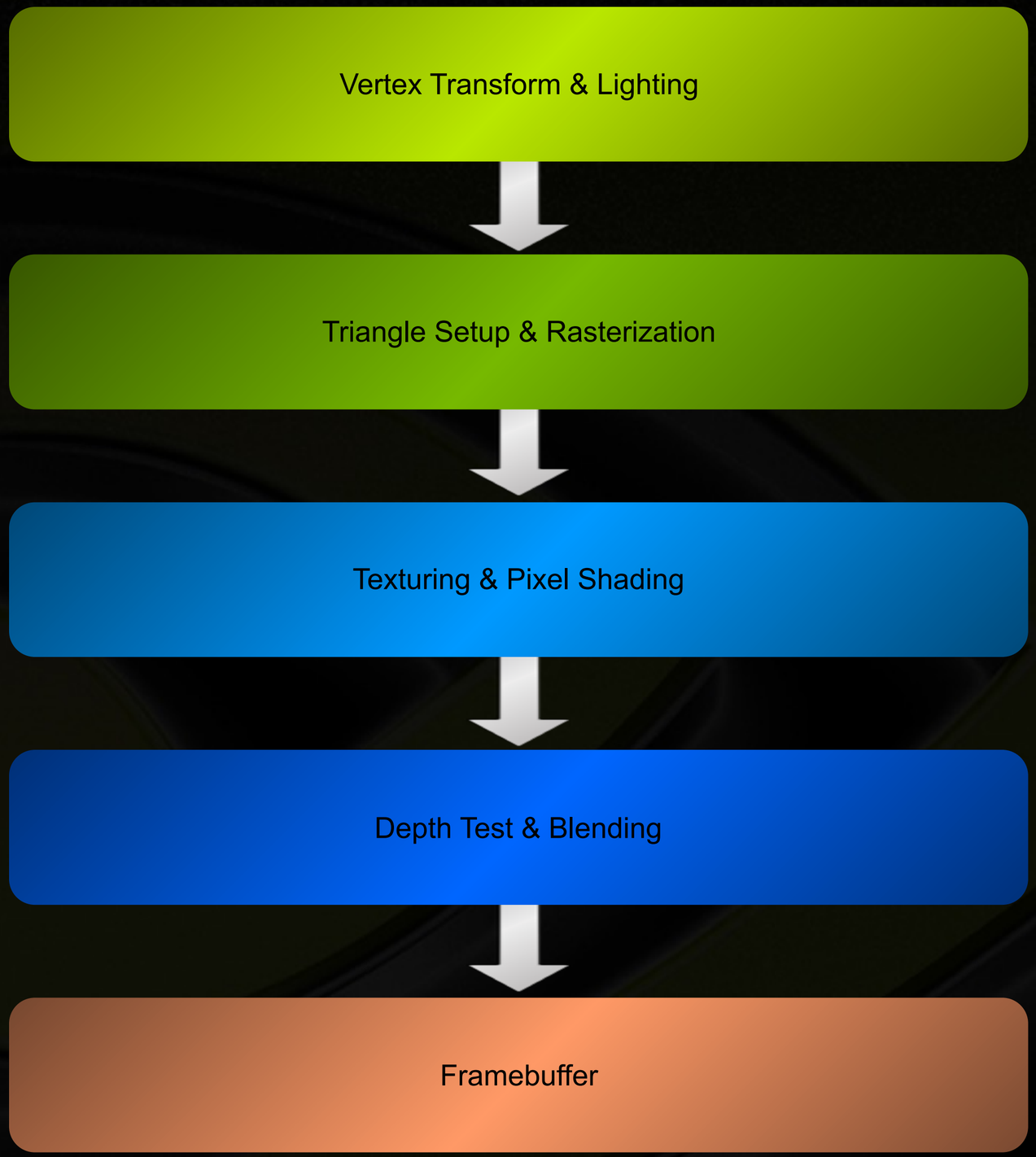


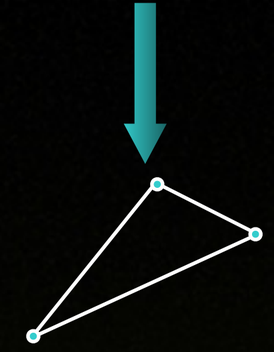
Figure 2: Photograph of the Geometry Engine.

The Geometry Engine: A VLSI Geometry System for Graphics
Jim Clark, 1982

The Graphics Pipeline



The Graphics Pipeline



Vertex Transform & Lighting

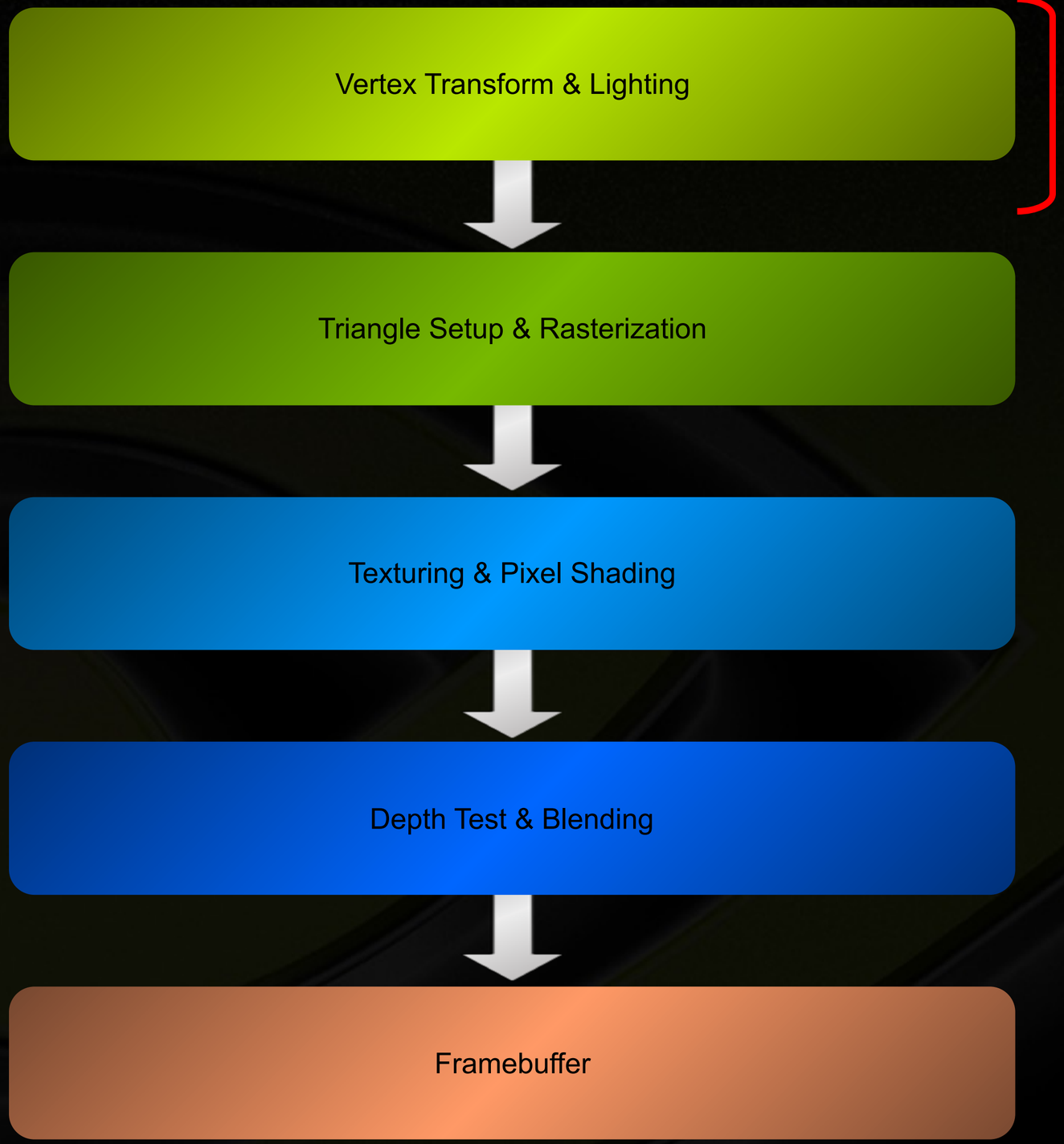
Triangle Setup & Rasterization

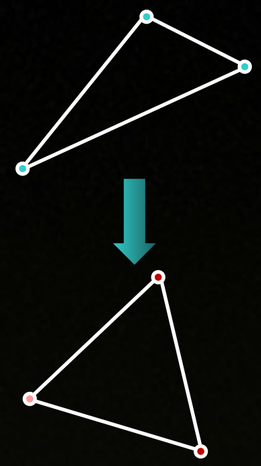
Texturing & Pixel Shading

Depth Test & Blending

Framebuffer

The Graphics Pipeline





The Graphics Pipeline

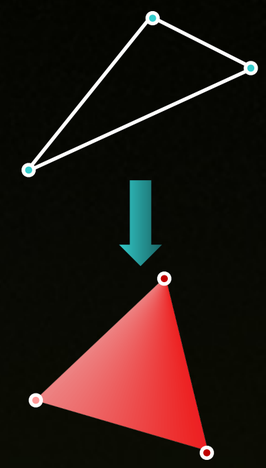
Vertex Transform & Lighting

Triangle Setup & Rasterization

Texturing & Pixel Shading

Depth Test & Blending

Framebuffer



The Graphics Pipeline

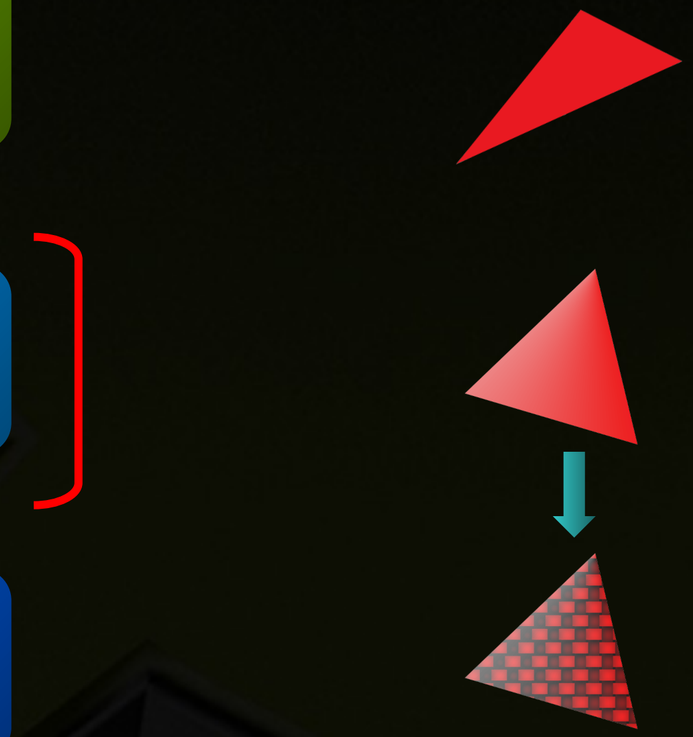
Vertex Transform & Lighting

Triangle Setup & Rasterization

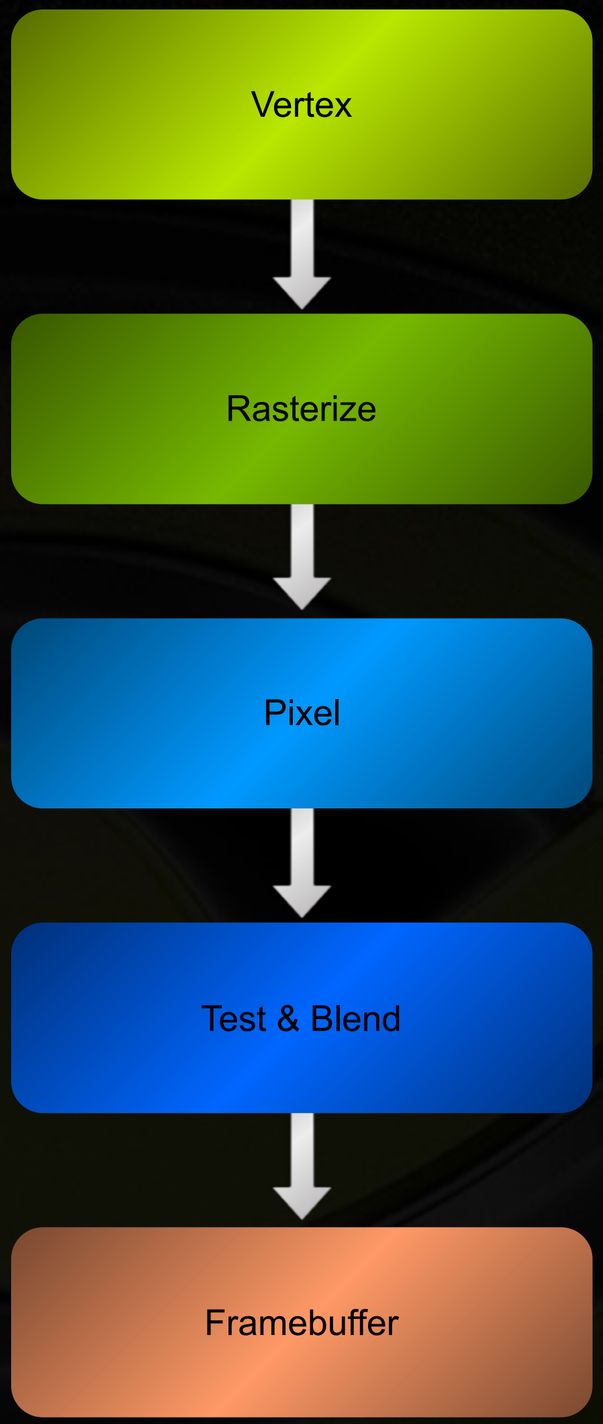
Texturing & Pixel Shading

Depth Test & Blending

Framebuffer

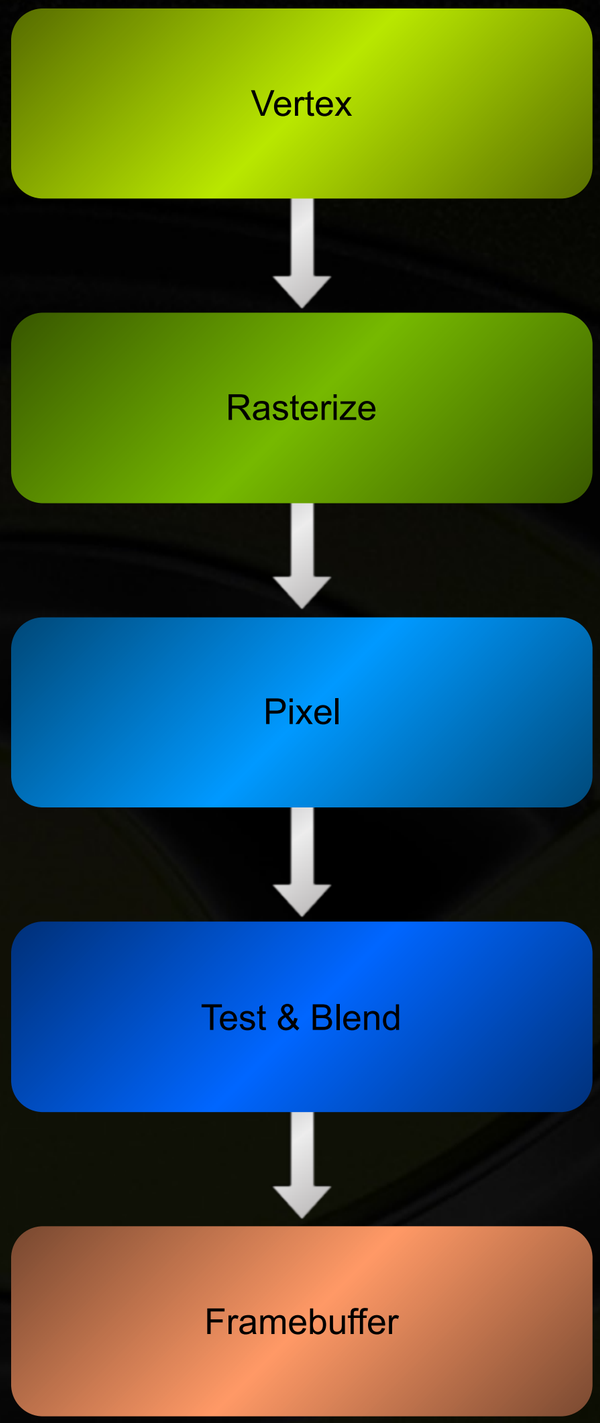


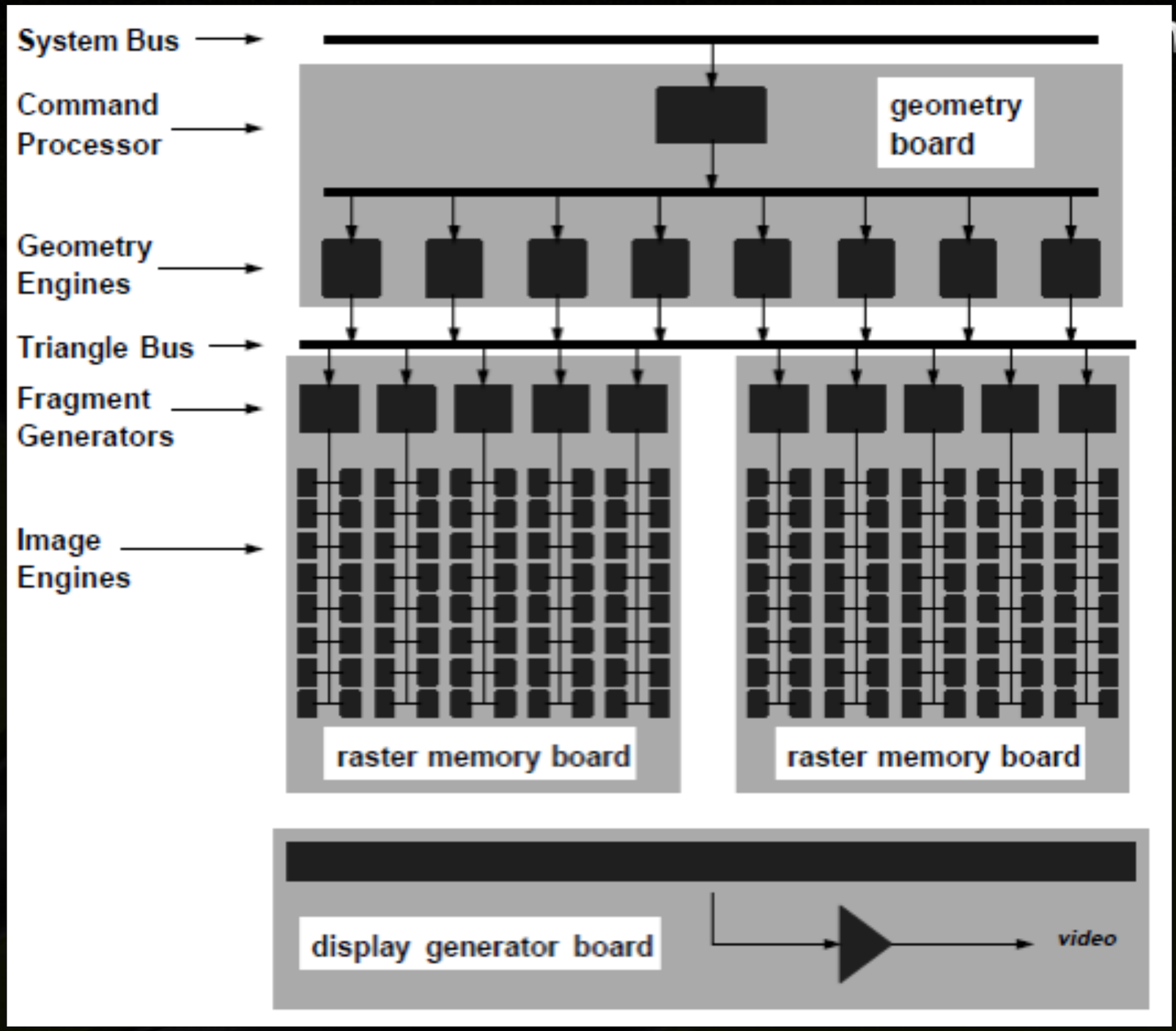
The Graphics Pipeline



- Key abstraction of real-time graphics
- Hardware used to look like this
- One chip/board per stage
- Fixed data flow through pipeline

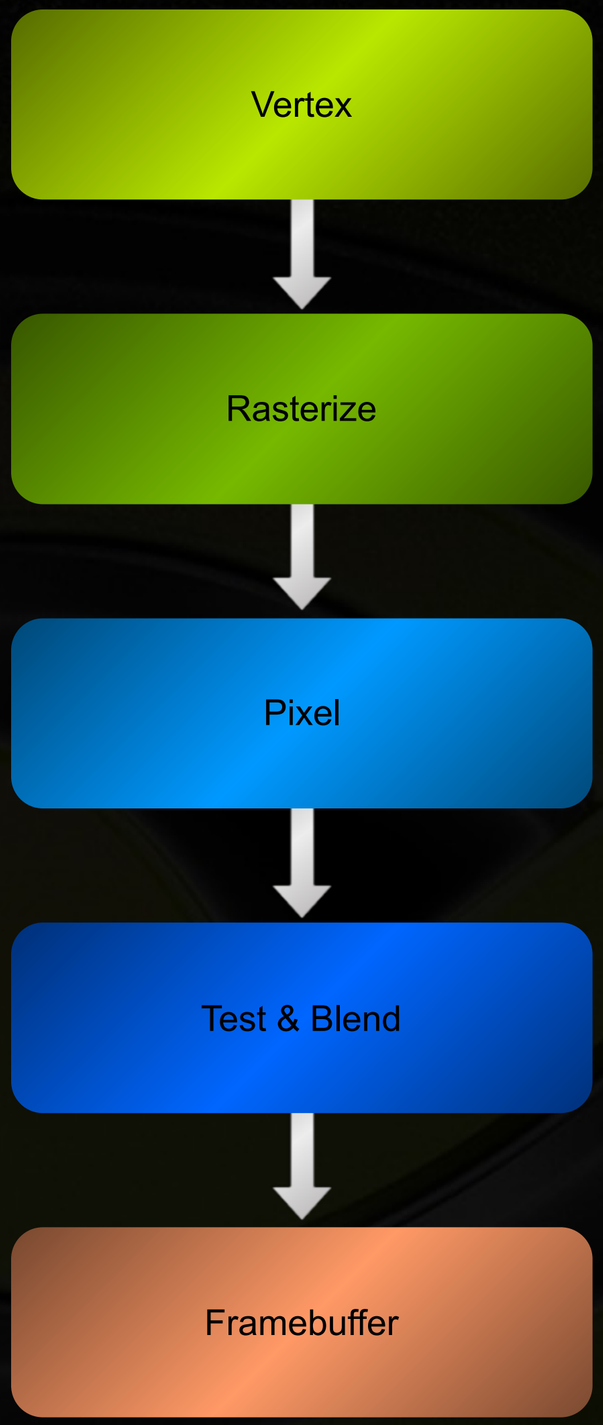
SGI RealityEngine (1993)

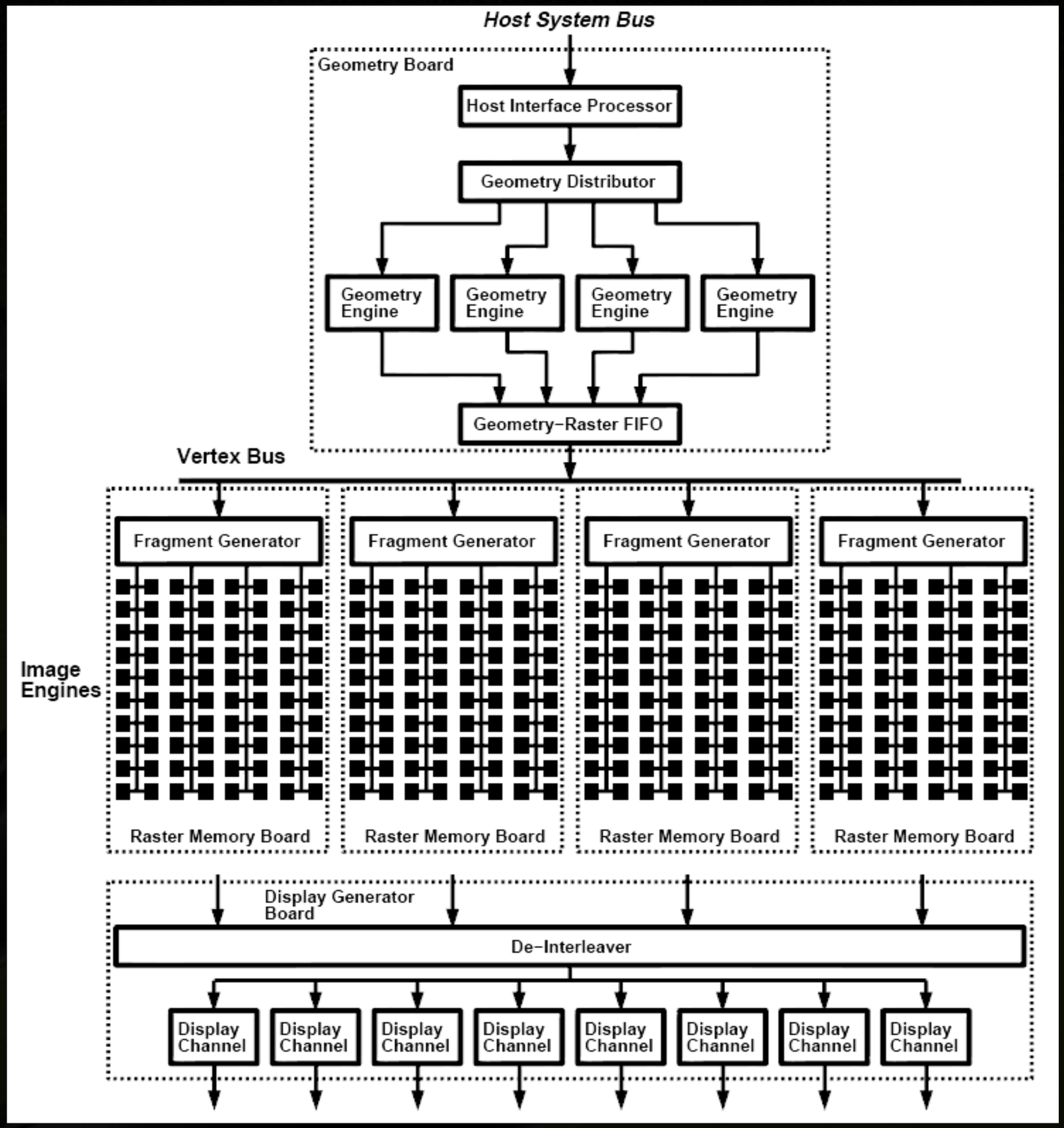




RealityEngine Graphics
Kurt Akeley , SIGGRAPH 93

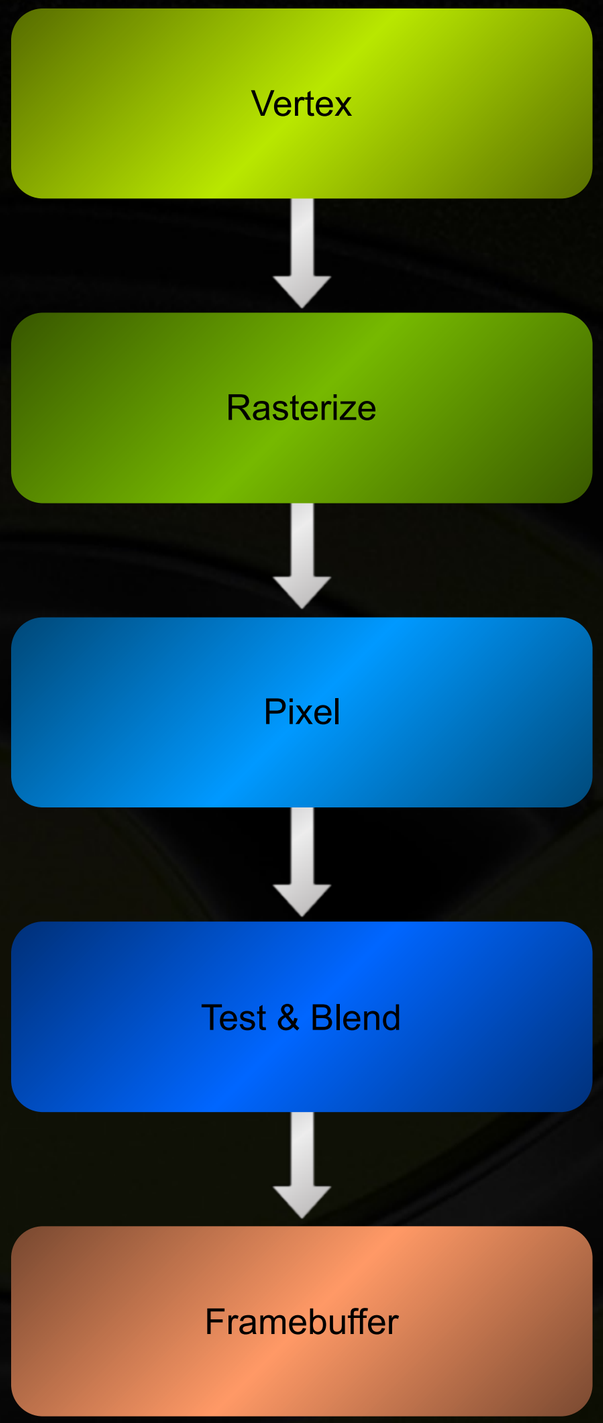
SGI InfiniteReality (1997)





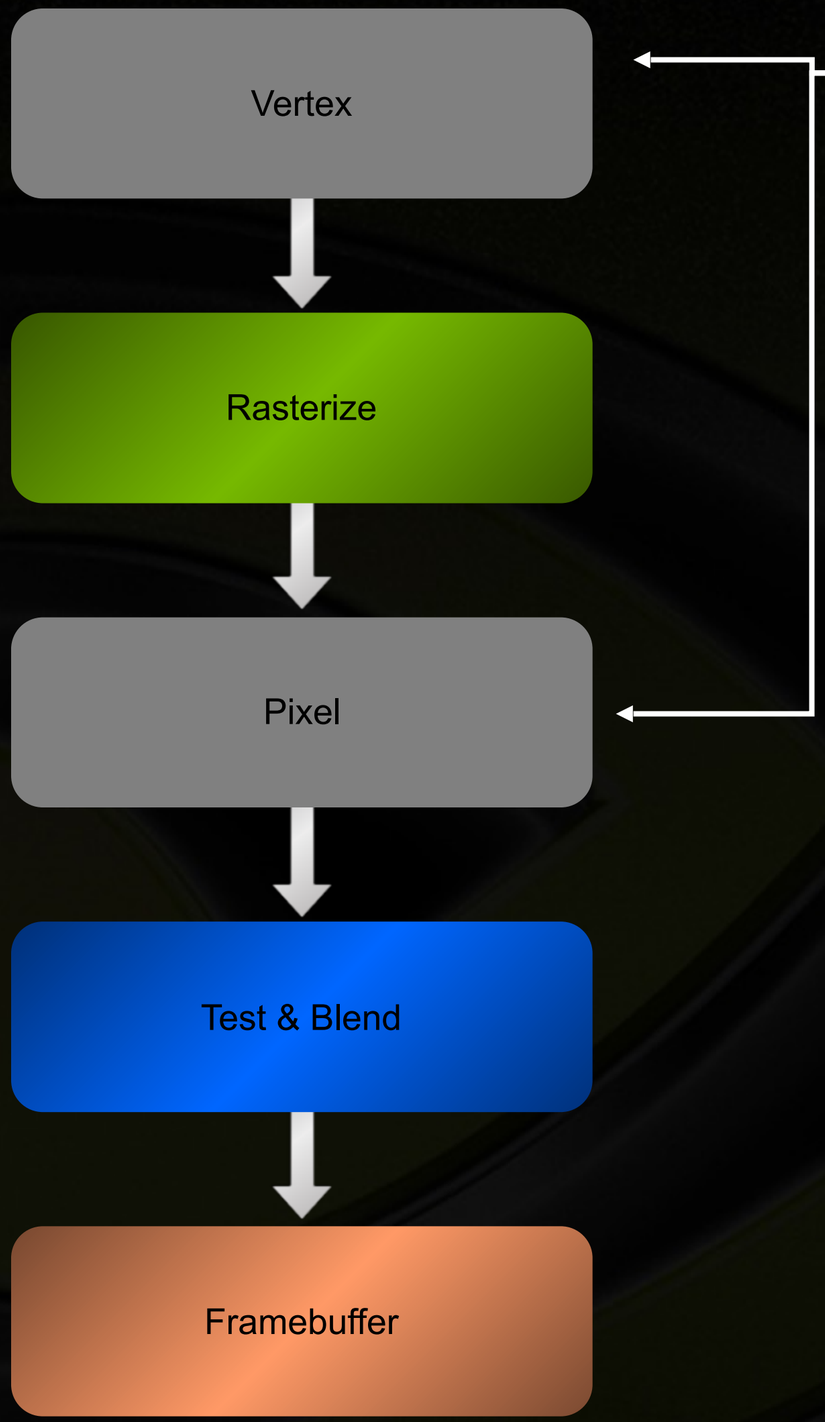
InfiniteReality: A real-time graphics system
Montrym et al., SIGGRAPH 97

The Graphics Pipeline



- Remains a useful abstraction
- Hardware used to look like this

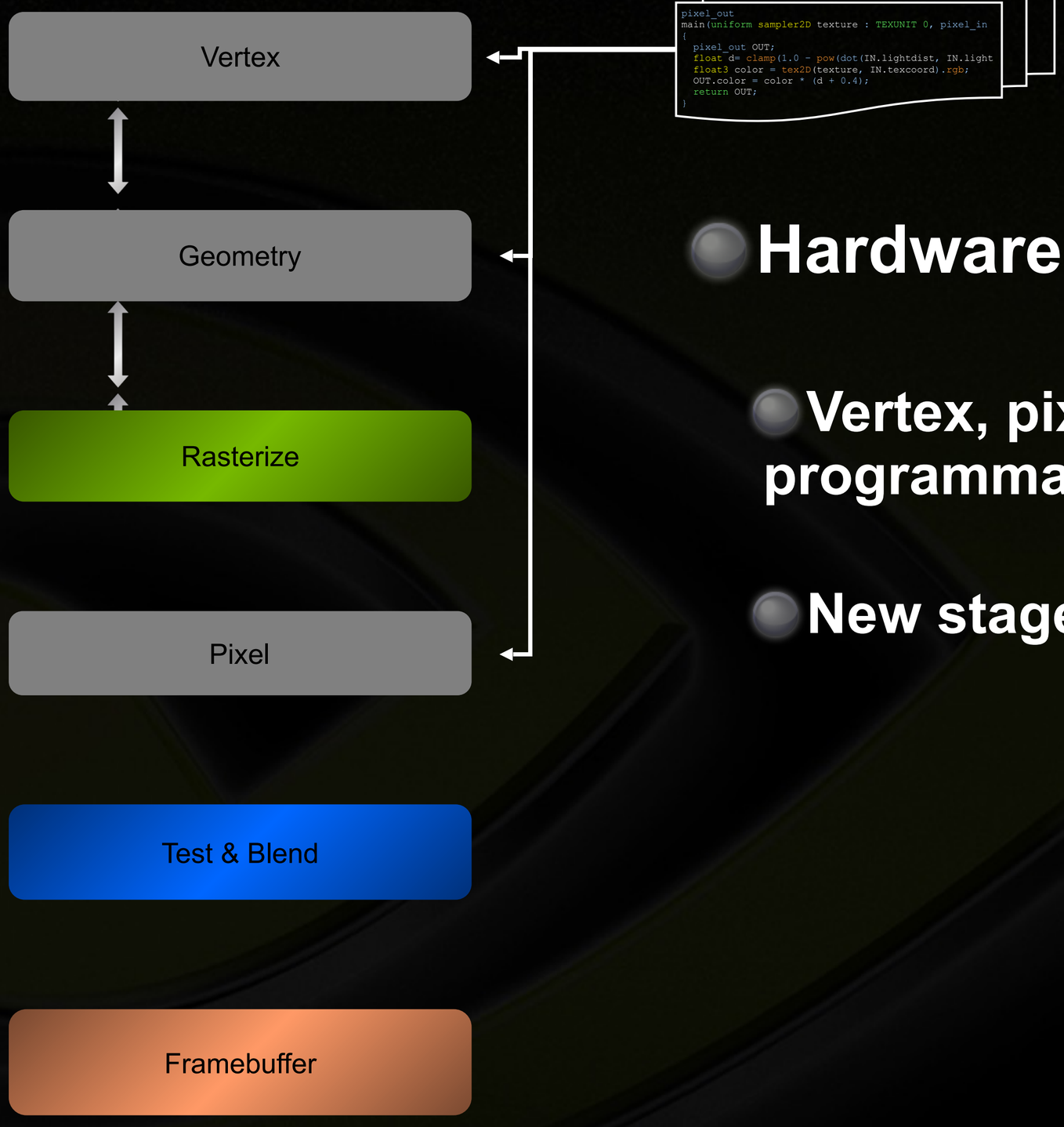
The Graphics Pipeline




```
pixel_out
main(uniform sampler2D texture : TEXUNIT 0, pixel_in
{
  pixel_out OUT;
  float d= clamp(1.0 - pow(dot(IN.lightdist, IN.light
  float3 color = tex2D(texture, IN.texcoord).rgb;
  OUT.color = color * (d + 0.4);
  return OUT;
}
```

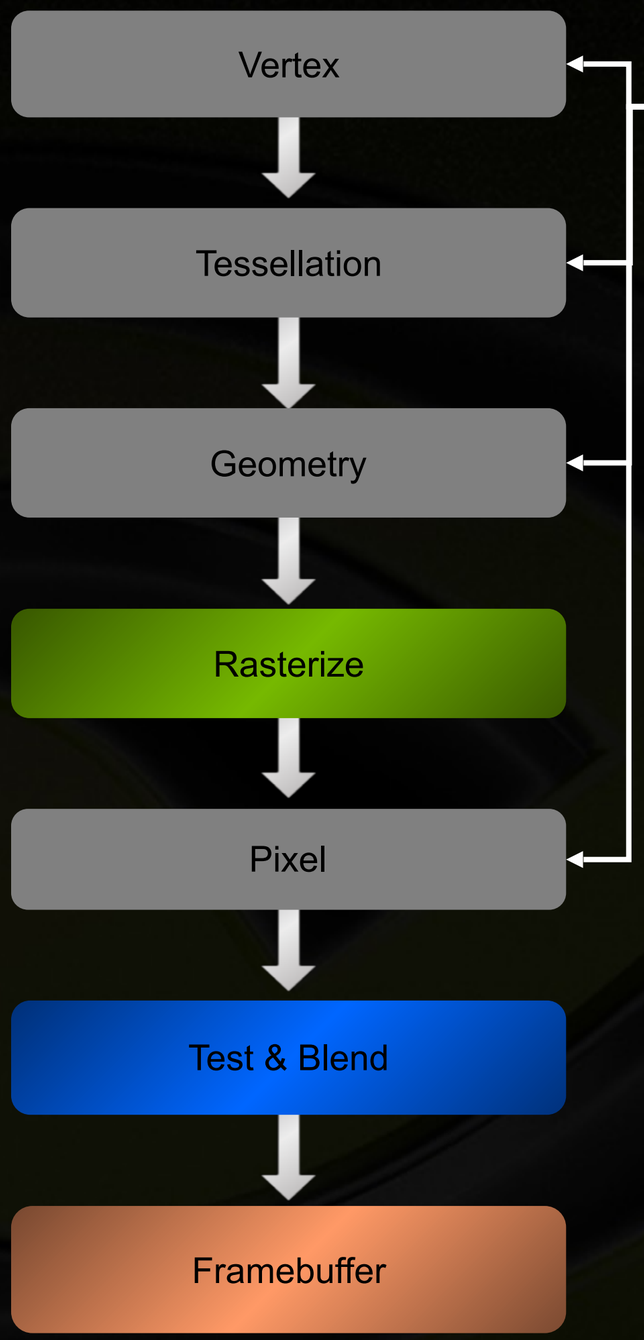
- Hardware used to look like this:
- Vertex, pixel processing became programmable

The Graphics Pipeline



- Hardware used to look like this:
  - Vertex, pixel processing became programmable
  - New stages added

The Graphics Pipeline

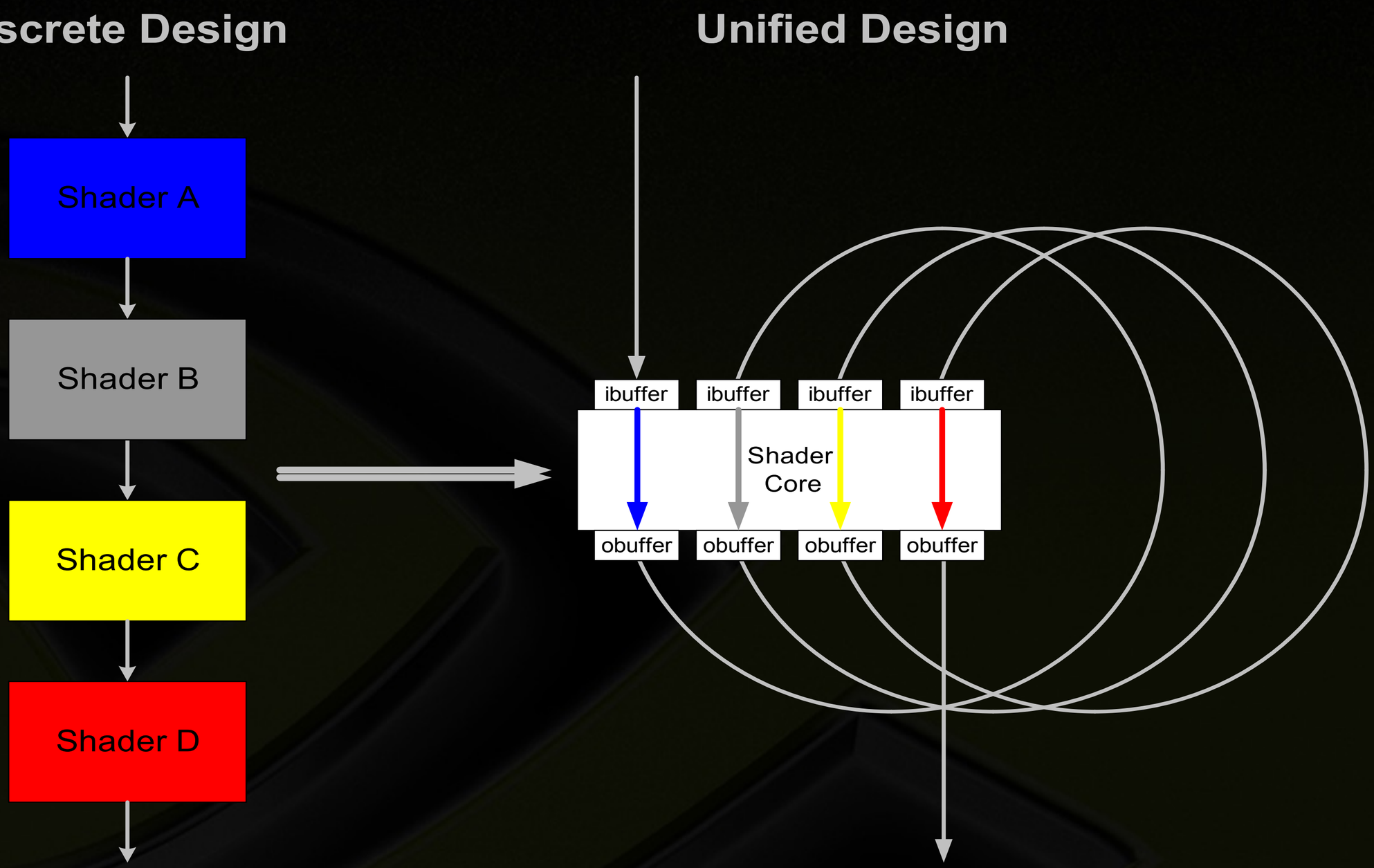



```
pixel_out
main(uniform sampler2D texture : TEXUNIT 0, pixel_in
{
  pixel_out OUT;
  float d= clamp(1.0 - pow(dot(IN.lightdist, IN.light
  float3 color = tex2D(texture, IN.texcoord).rgb;
  OUT.color = color * (d + 0.4);
  return OUT;
}
```

- **Hardware used to look like this**
- **Vertex, pixel processing became programmable**
- **New stages added**

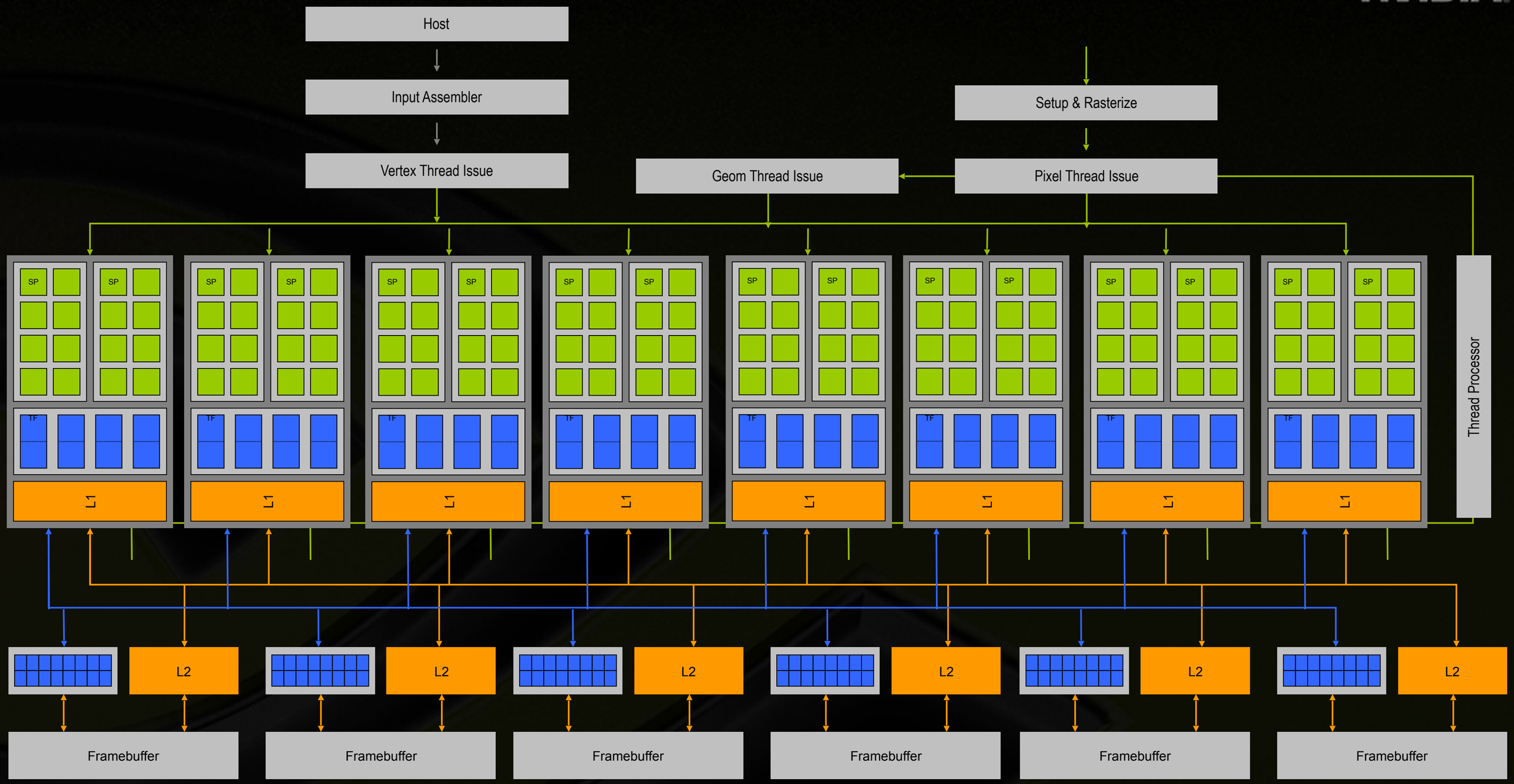
GPU architecture increasingly centers around shader execution

Modern GPUs: Unified Design



Vertex shaders, pixel shaders, etc. become *threads* running different programs on a flexible core

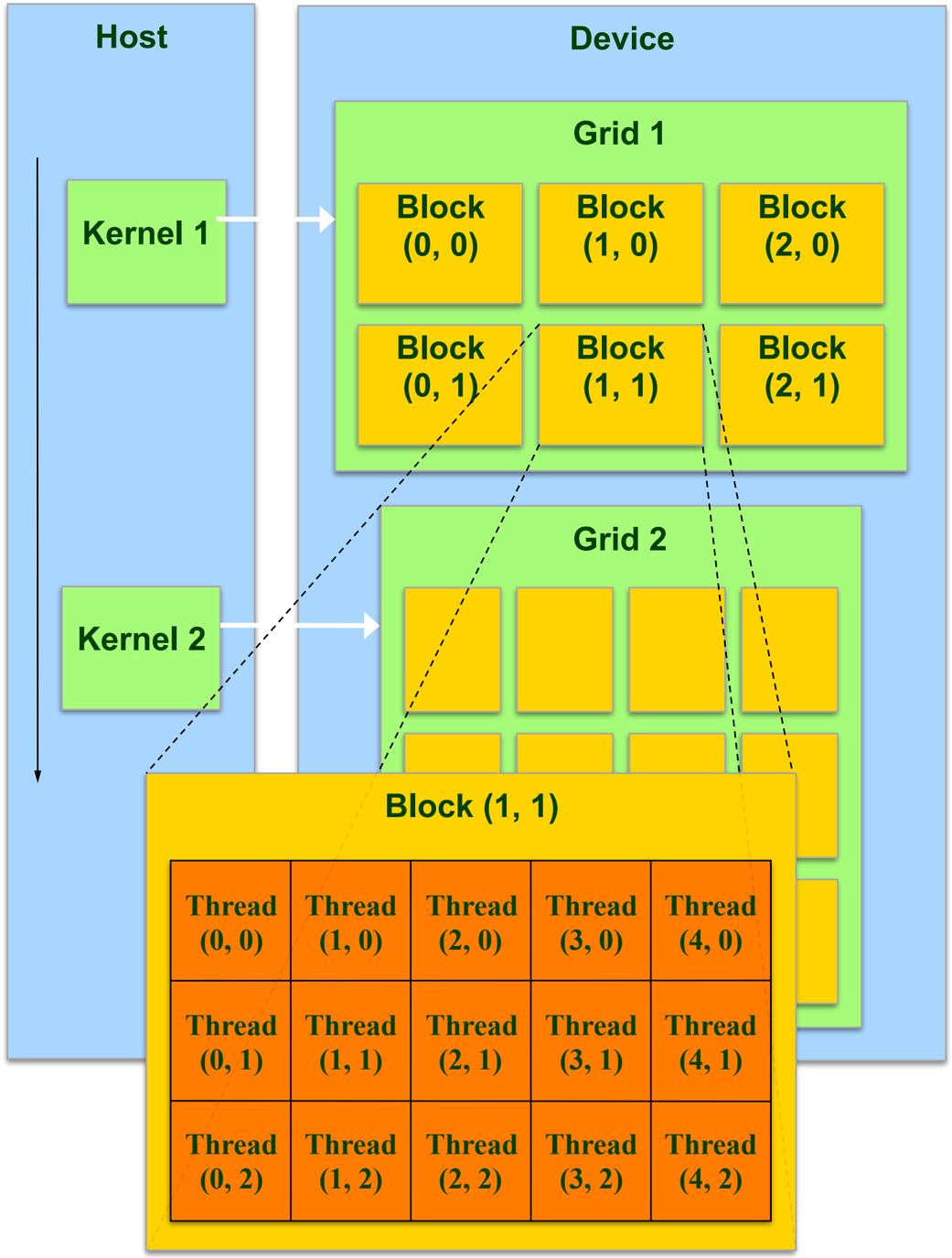
GeForce 8: 2008 GPU Architecture



More Modern GPU Architecture: GT200

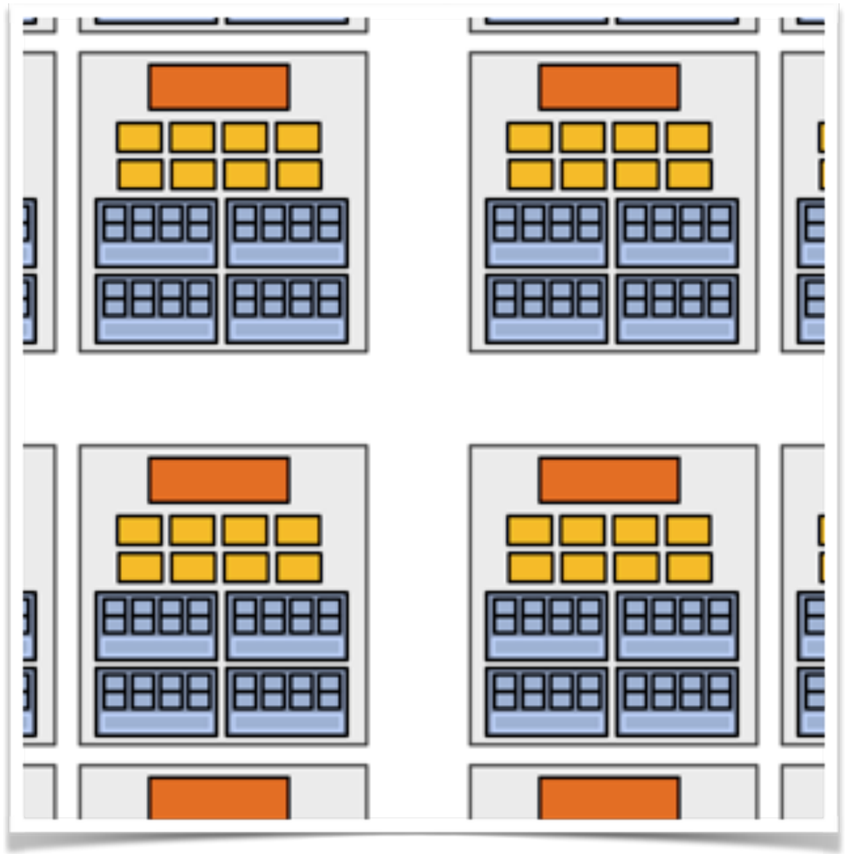
Programming Model Overview

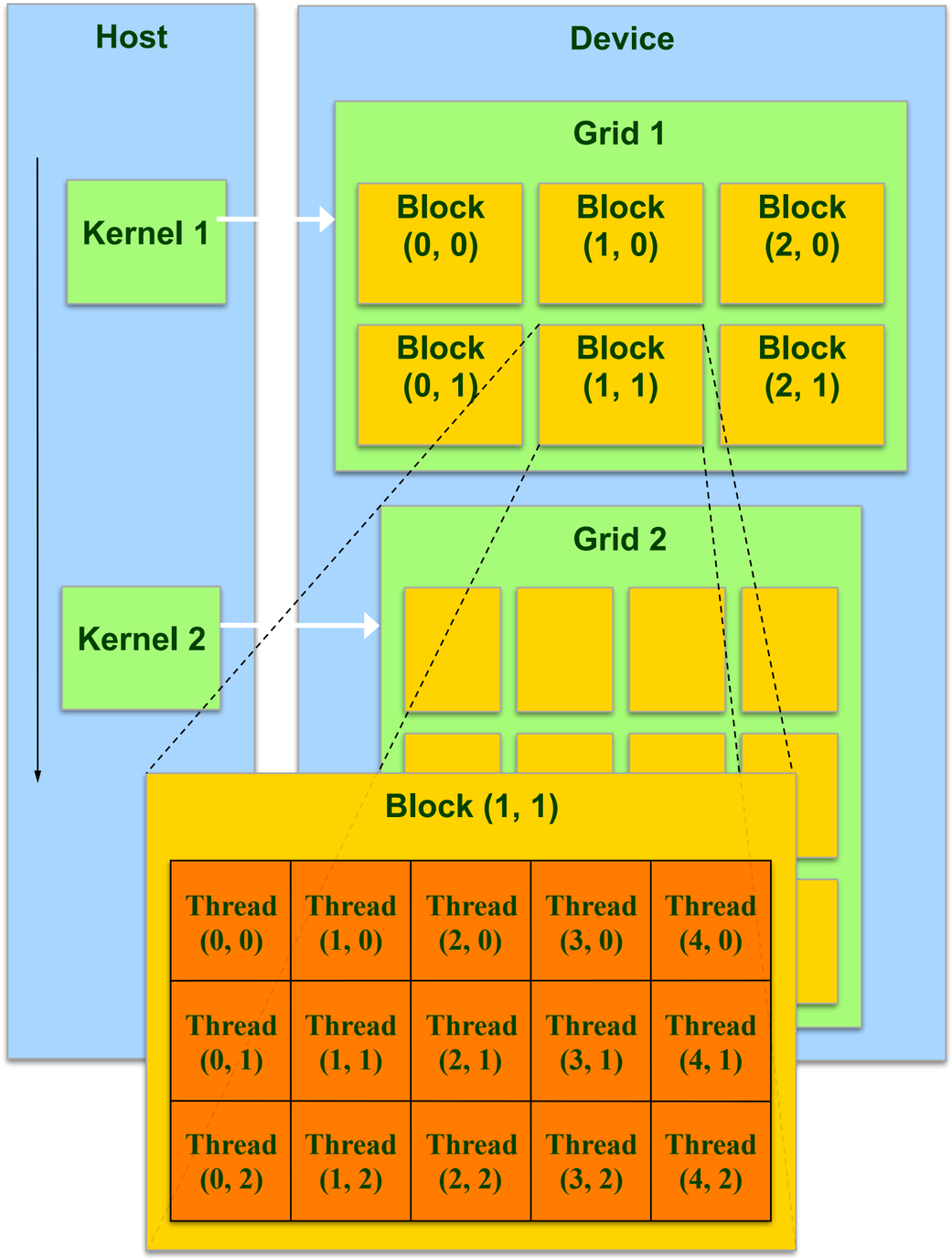
- A kernel is executed as a grid of thread blocks
- The programmer specifies the dimensions of the grid and thread block
- A thread block is a batch of threads that can cooperate with each other (shared memory, synchronization)
- Two threads from two different blocks cannot cooperate
  - Blocks are independent



What The Hardware Does

- Grids run on the entire machine
- Blocks are mapped to cores
- Threads are mapped to scalar processors

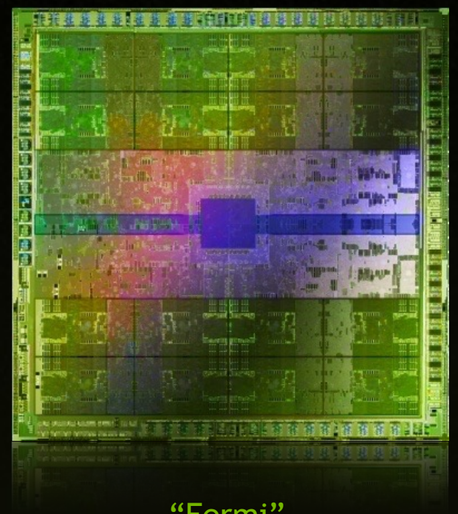




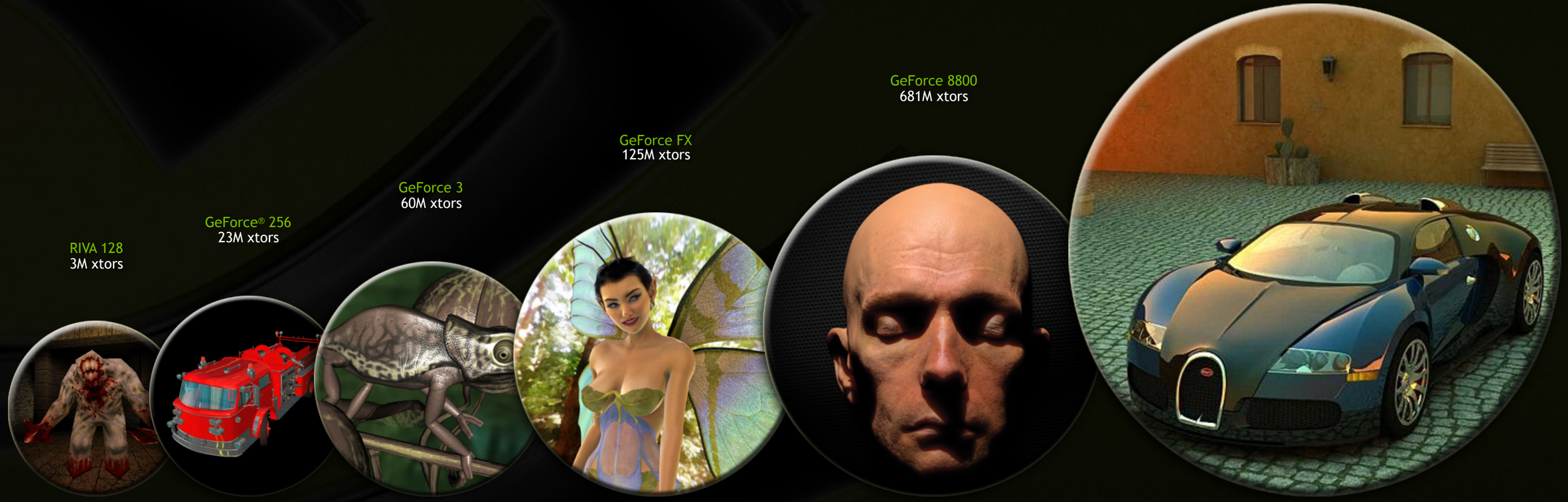
GPUs Today

Lessons from Graphics Pipeline

- **Throughput** is paramount
- Create, run, & retire **lots of threads** very rapidly
- Use **multithreading** to hide latency



"Fermi"
3B xtors



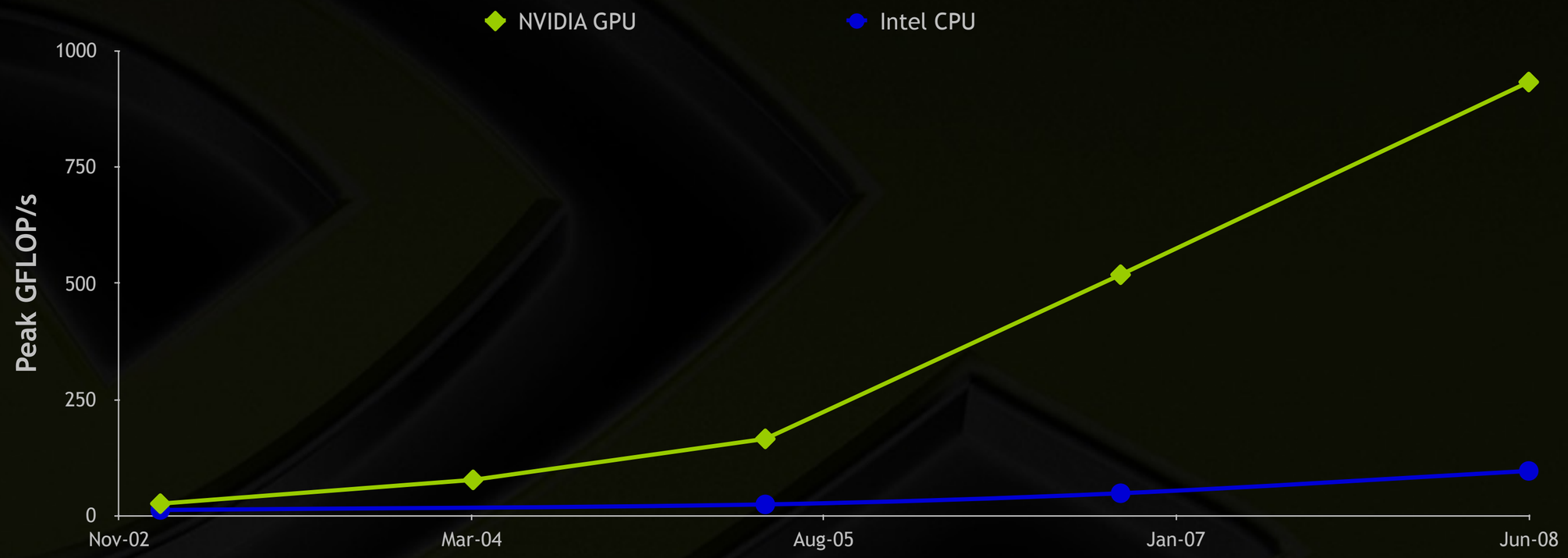
Wheel of Reincarnation

- T. H. Myer and I. E. Sutherland, "On the Design of Display Processors", Comm. ACM, vol. 11, no. 6, June 1968
- Important function moved to peripheral processor
- Peripheral processor gains more capability
- Supporting two general processors is stupid
- Functionality folded back into main processor

Why GPU Computing?

Fact:
nobody cares about theoretical peak

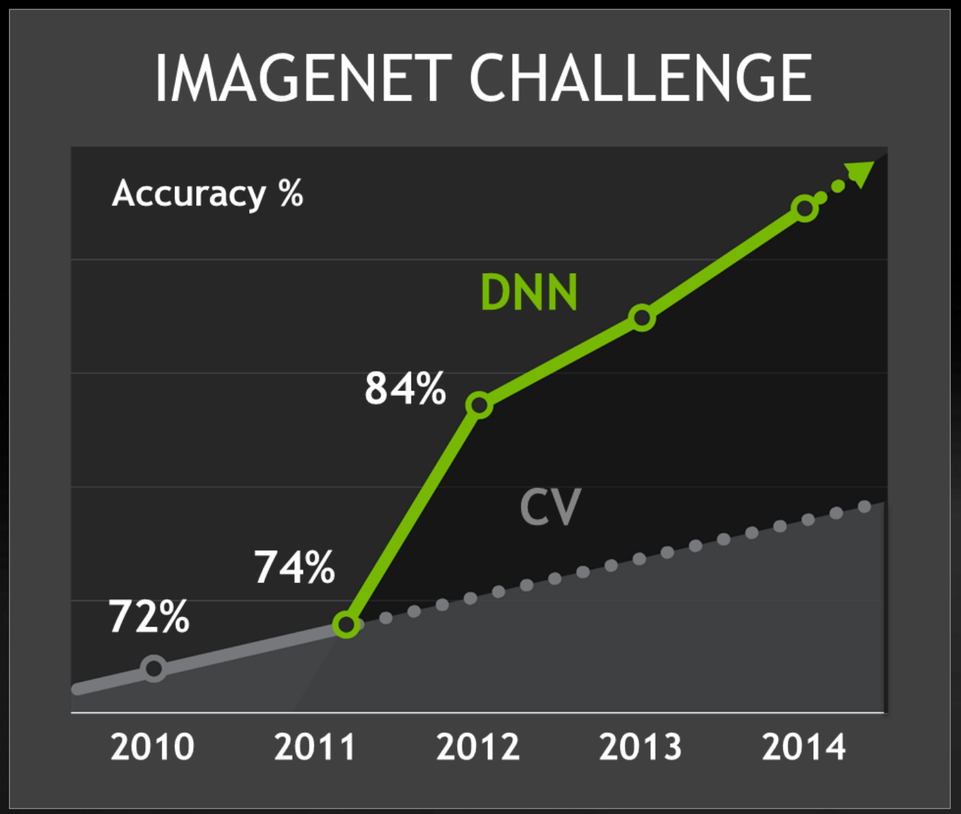
Challenge:
harness GPU power for real application performance



GPU Computing 3.0: *an emerging ecosystem*

- Hardware & product lines
- Algorithmic sophistication
- Cross-platform standards
- Education & research
- Consumer applications
- High-level languages

GPU begins to "vanish" behind codes, platforms, apps



NVIDIA GPUs have been broadly adopted in deep learning, a branch of artificial intelligence.

Deep learning has been ignited by the convergence of three trends: the flood of data brought by web services companies, recent algorithm breakthroughs, and the ability to compute massive amounts of data with GPUs.

Today, machines are being trained to recognize images, text and speech. But this is just the tip of the iceberg. The world's largest and most innovative companies are deploying deep learning across a variety of applications.

In 2012, GPUs enabled a breakthrough in the ImageNet Challenge, the World Cup of deep learning and computer vision. GPUs have recently enabled machines to outperform humans at this task.

http://www.slideshare.net/NVIDIA/gtc-2015-highlights

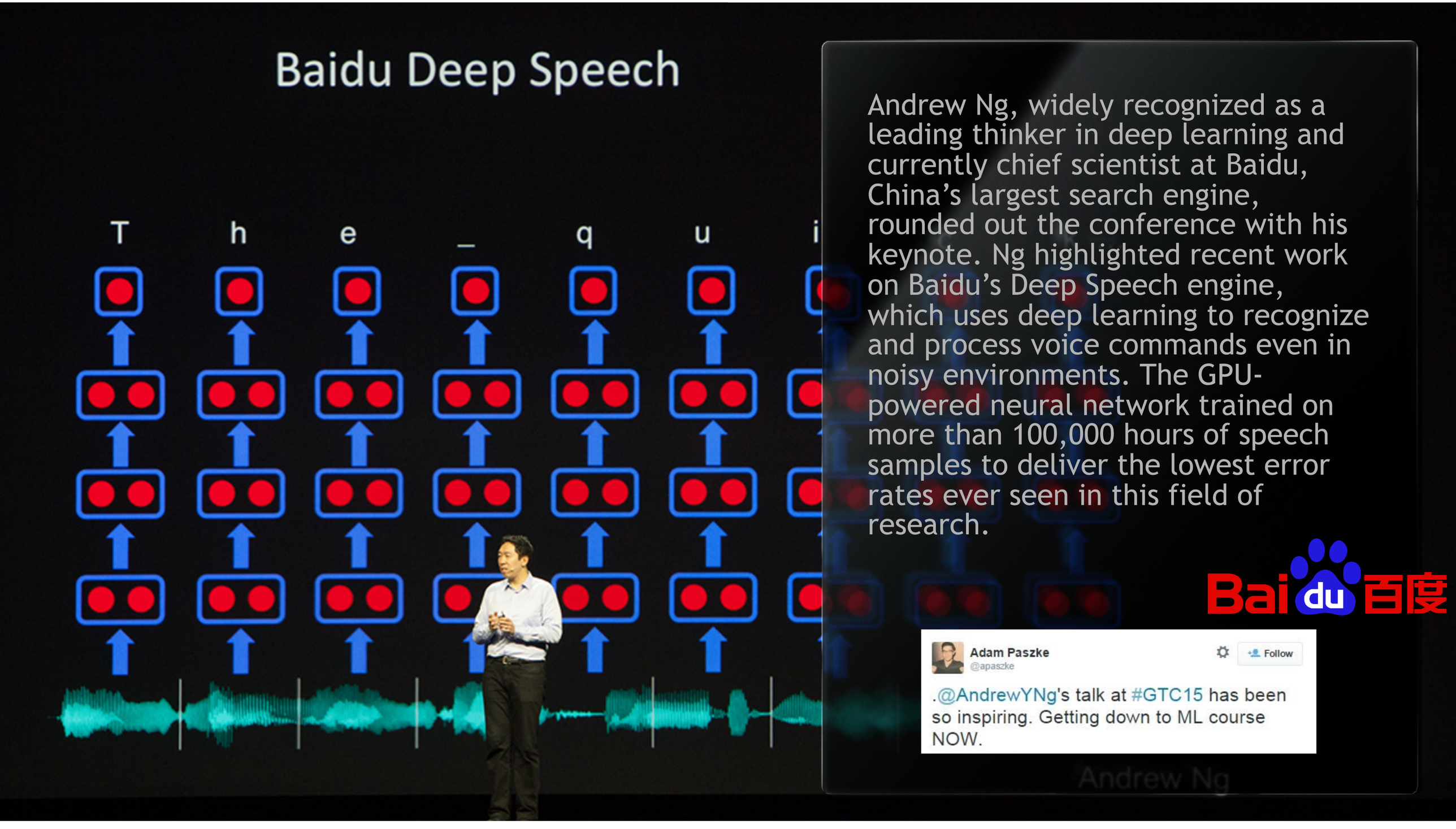
NVIDIA Corporation (NVDA) 9 Jan 2018

The theme of deep learning carried through our guest keynotes. Jeff Dean, senior fellow at Google, described how the company is using GPU-powered deep neural networks to bring greater levels of intelligence to image, text, and speech recognition. He also highlighted work done by the recently acquired Deep Mind. Using Atari video games, the researchers trained a network to not just classify, but take actions in an environment. Ultimately, the network beat a series of games and the work earned the cover of *Nature* magazine.

"We love GPU cards. We just use a lot of them."

— Jeff Dean, Google

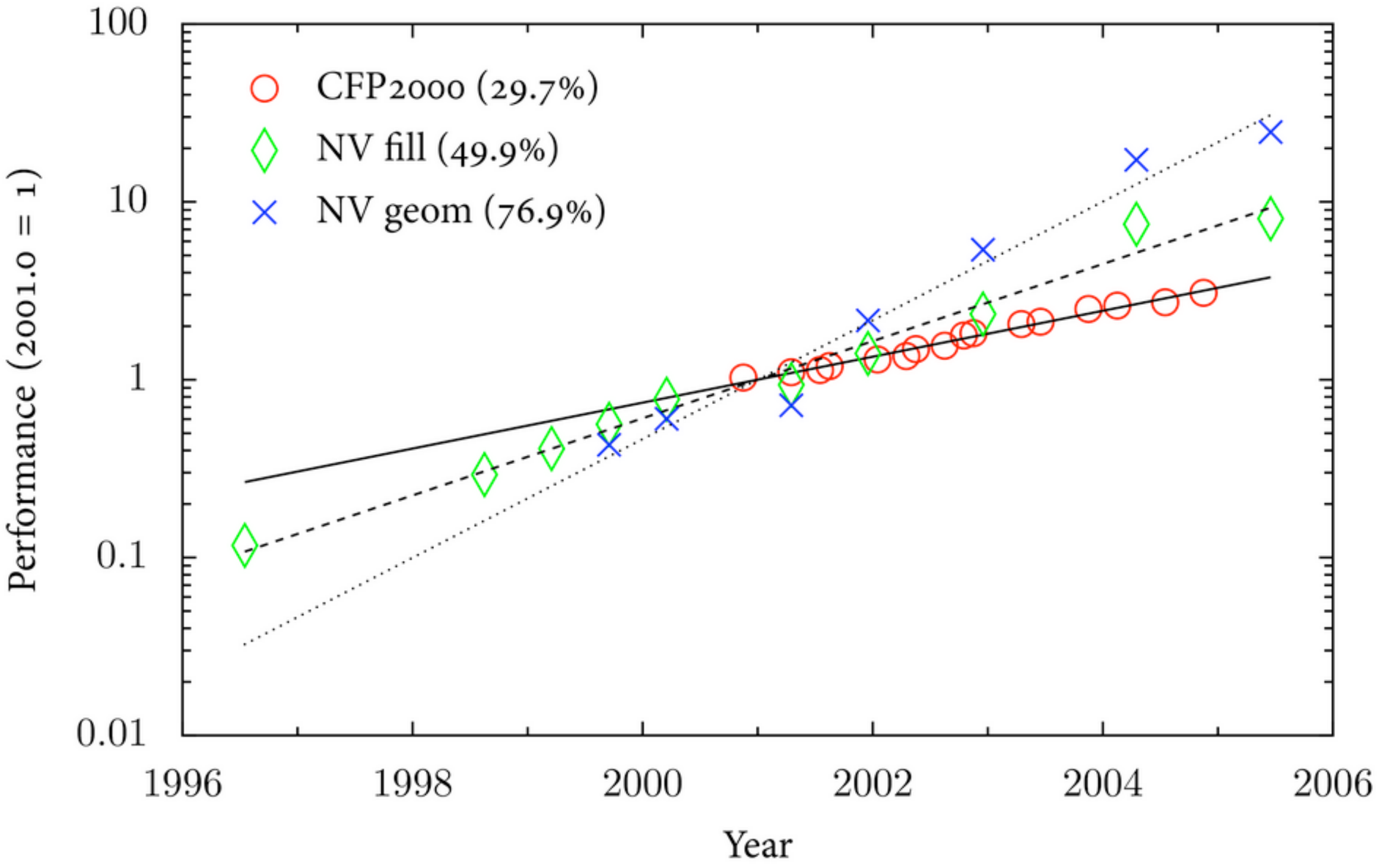
http://www.slideshare.net/NVIDIA/gtc-2015-highlights



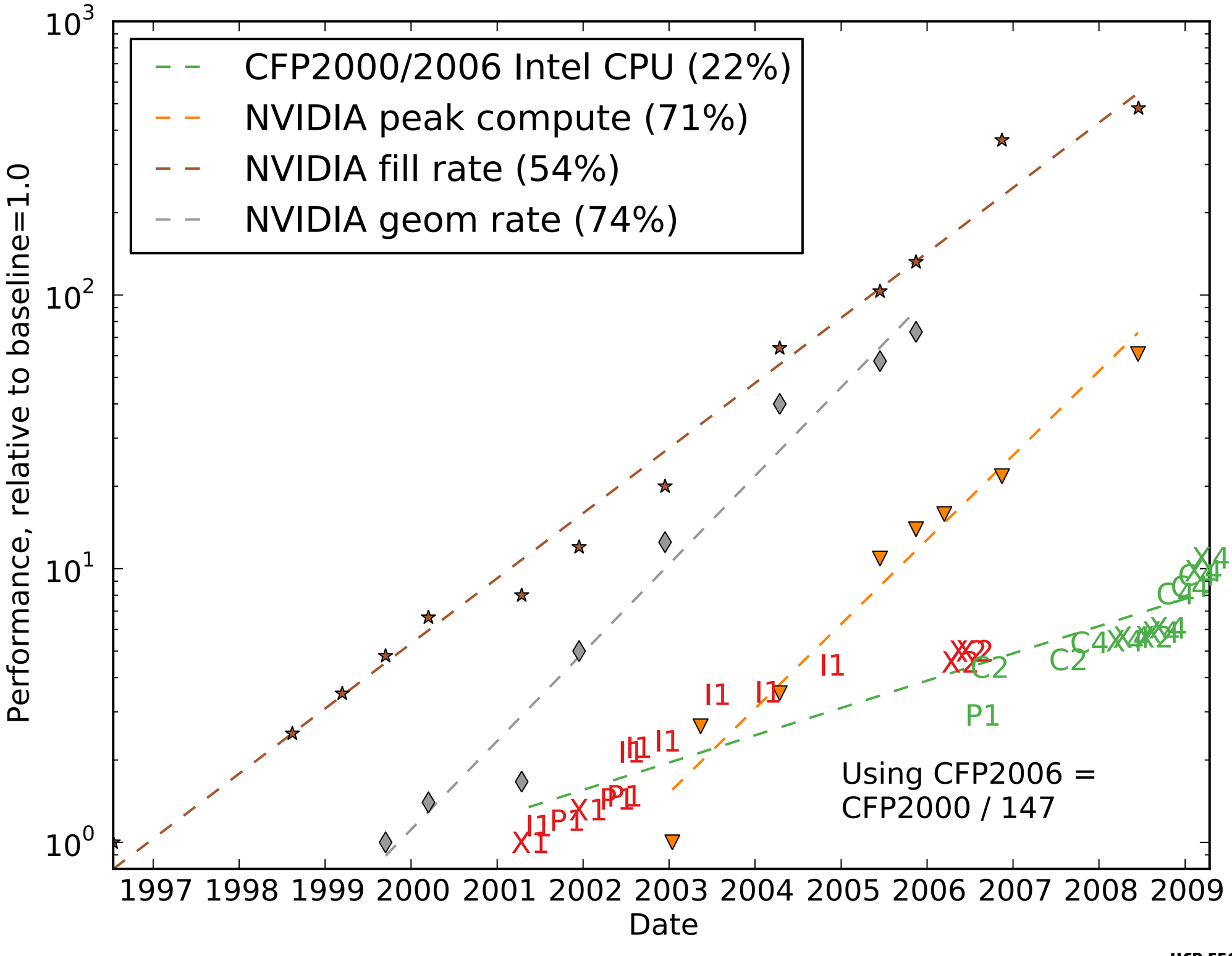
http://www.slideshare.net/NVIDIA/gtc-2015-highlights

Why Graphics HW? CPU vs. GPU

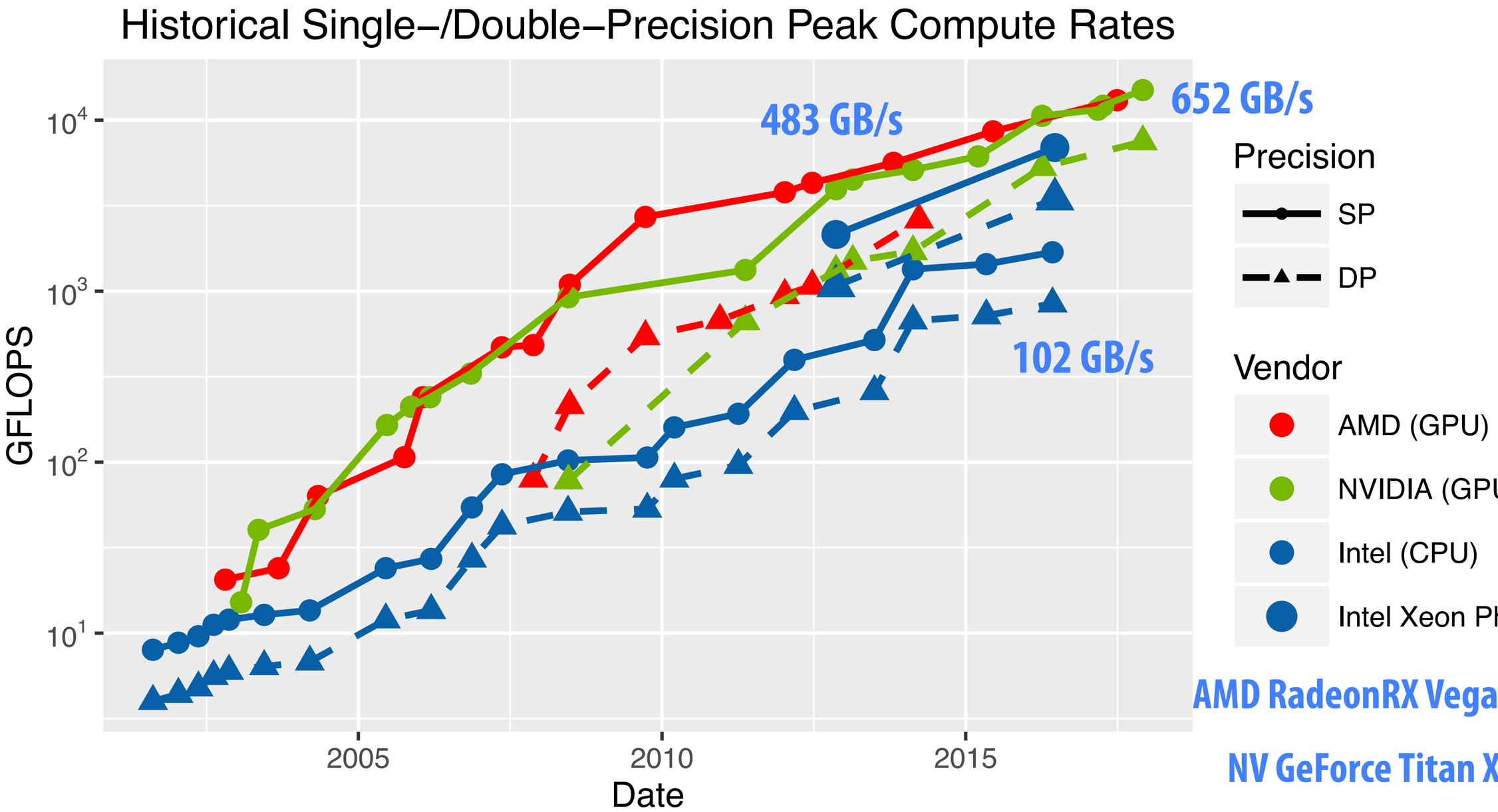
- Historically superior performance increases to CPU
- Also adding features!



Why Graphics HW? Compute



Recent GPU Performance Trends



AMD RadeonRX Vega64: $820

NV GeForce Titan X: $999

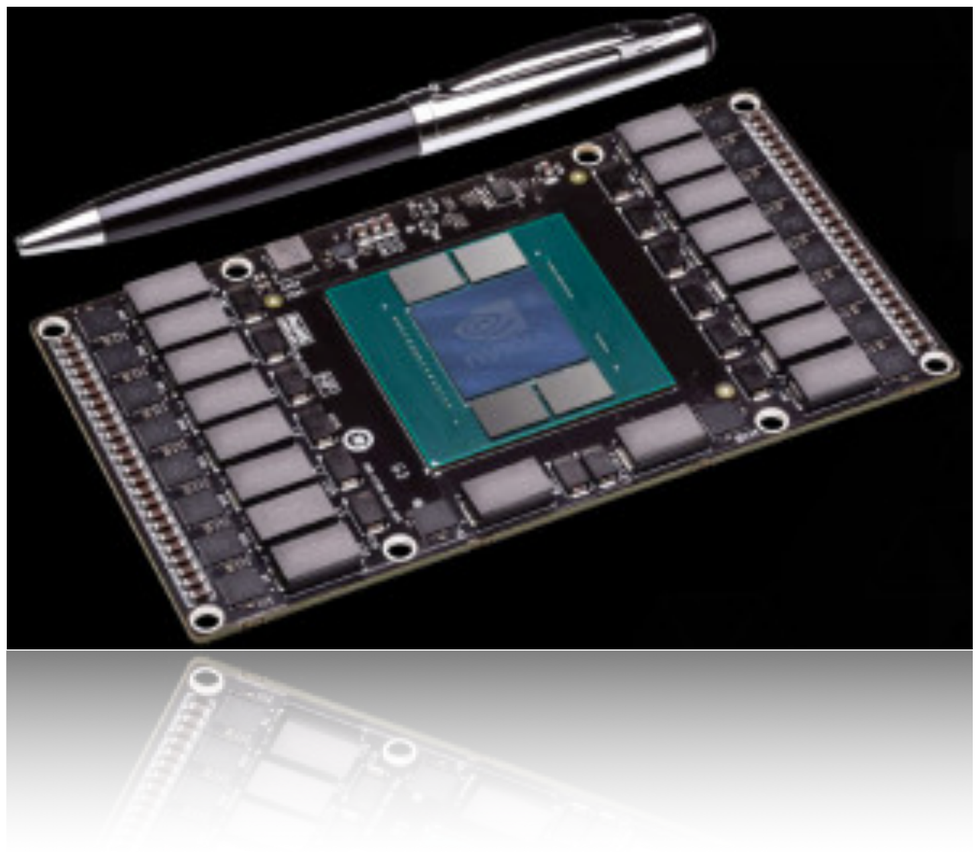
NV Tesla P100: $9750

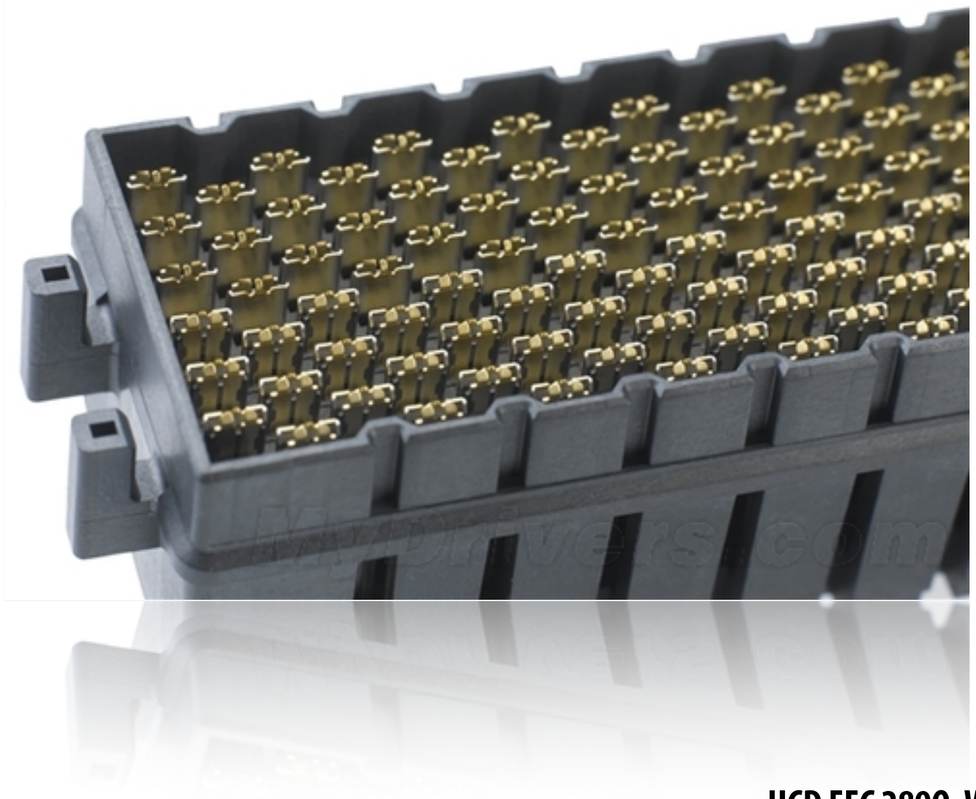
Xeon E7-8890v4: $6650

Xeon Phi 7290: $3094

NVIDIA “Pascal” (2016)

- Stacked memory (4x)
- NVLink high-speed CPU-GPU connection (“5–11x”)
- IBM Power will be first platform supporting NVLink





My goals for you

- Understand the GPU programming model and the hardware, how they work together, and why
- Understand the technology trends behind why parallel computing is important
- Understand how to program GPUs, not just use them
- Understand how to write efficient GPU programs
- Understand parallel computing patterns and how to approach them in a parallel way

Lecture 1: Introduction / Overview

- Intro & Motivation
- Class outline
- Administrivia

Class schedule

- 3 lectures on technology trends, core GPU concepts, CUDA programming model + GPU hardware, the CUDA programming language
- 2 lectures on algorithms & data structures
- 1 lecture on optimizations
- 3 lectures on computational patterns (grids, matrices, graphs, trees, FFT, n-body)
  - This will include the guts of deep learning
- 2 lectures on load balancing, task parallelism, multi-GPU, higher-level programmability

Guest speakers

- Steven Dalton, NVIDIA Research (Thursday!)
  - Title: The road to software engineering 2.0
  - “Loosely I will talk about the way we write optimized code and how we plan to integrate higher level systems and deep learning to write code in the future. Very high level but I will outline the intersection with several ongoing projects at Nvidia.”
- Stephen Jones, NVIDIA
- Emad Barsoum, Microsoft
- Todd Mostak, MapD

Last day of class is Project Presentations

Lecture 1: Introduction / Overview

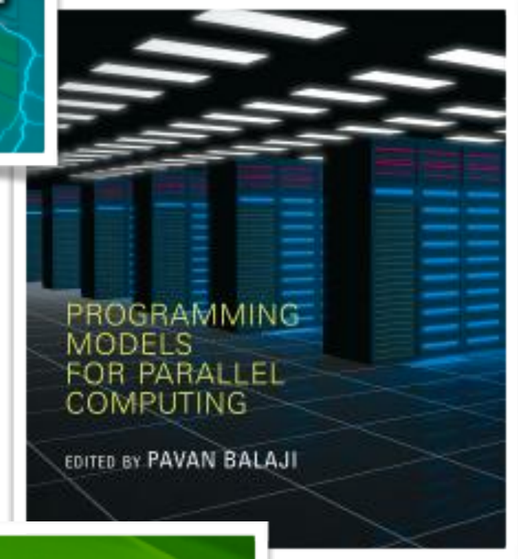
- Intro & Motivation
- Class outline
- Administrivia

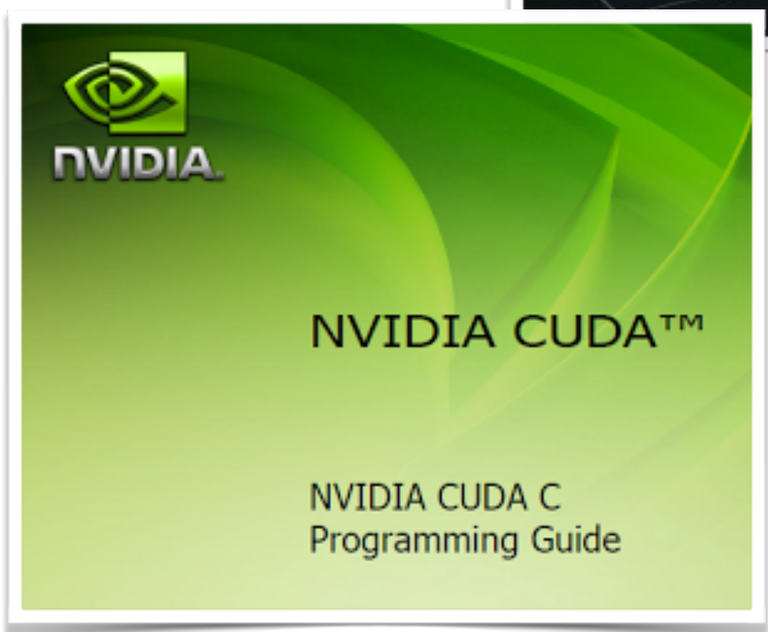
Background required

- Computer architecture highly desirable (at the level of EEC 170 or ECS 154B)
  - "Register", "cache", "pipeline", etc.
- Reasonable familiarity with the C/C++ programming language
- Computer graphics, algorithms, parallel programming experience are relevant
- 4 units

Textbooks (none required)

- Recommended/most relevant book: David B. Kirk and Wen-mei W. Hwu, Programming Massively Parallel Processors: A Hands-on Approach, 3rd Edition, Morgan Kaufmann, Dec. 2016
- Pavan Balaji, Programming Models for Parallel Computing, The MIT Press, 2015.
- CUDA C Programming Guide (on web, free)





No exams, no notes, communication

- Please come to class
- Please listen
- Please participate

- I'll post slides online (Canvas)

- All communication: through Slack
  - https://join.slack.com/t/eec289q-w18/signup (sign up with @ucdavis.edu address)

Assignments

- 3 2-week assignments, designed to give you experience writing and optimizing CUDA programs
  - Released on Thursdays, due in 2 weeks
  - First one released on Thursday
  - 15% of grade each

Assignments

- Project. Groups of 1–2. You choose topic.
  - Complete, limited scope projects better than broad, incomplete projects
  - Must focus on implementing something on the GPU (not just using GPU libraries, e.g., you can't do "I'm training a DNN for this interesting ML task")
  - Be bold, but make sure your contribution is clear
  - Talk to me about it

  - Requires preproposal (20 Feb), proposal (27 Feb), revision if necessary (6 Mar), report (15 Mar), presentation (15 Mar)
  - 45% of grade

Computing resources

- "Snake" workstations in 2107 Kemper Hall all have CUDA installed with NVIDIA GPUs. Generic login is to "snake.ece.ucdavis.edu". You should do your development on these machines. You can use your own machine/GPU, but we're only going to support the class machines with respect to tool issues, and we will use CUDA 9 features
- We also have one big GPU that we will use for performance testing. You have access to this. mc.ece.ucdavis.edu has an NVIDIA Tesla K40 GPU and CUDA.
- We have 2 Jetson TK1s from NVIDIA for projects (to borrow)

Guidelines

- Papers must be:
- Well written and formatted
- Properly referenced
- Intellectually honest

- Prime grading standard: intellectual merit

- OK to fail, so long as you demonstrate something interesting.

Expectations

- Participate in class!
- Ask questions!
- Offer answers to my questions!
- Explore interesting topics!
- Challenge our guest speakers!
- Act honorably and honestly!
- NOT a lecture class. NOT a passive experience. ACTIVE learning.
- Most common problem: Not getting started
- Ask for help if you need it!