GPUs

CS 4620 Lecture 24

© 2015 Kavita Bala •

Announcements

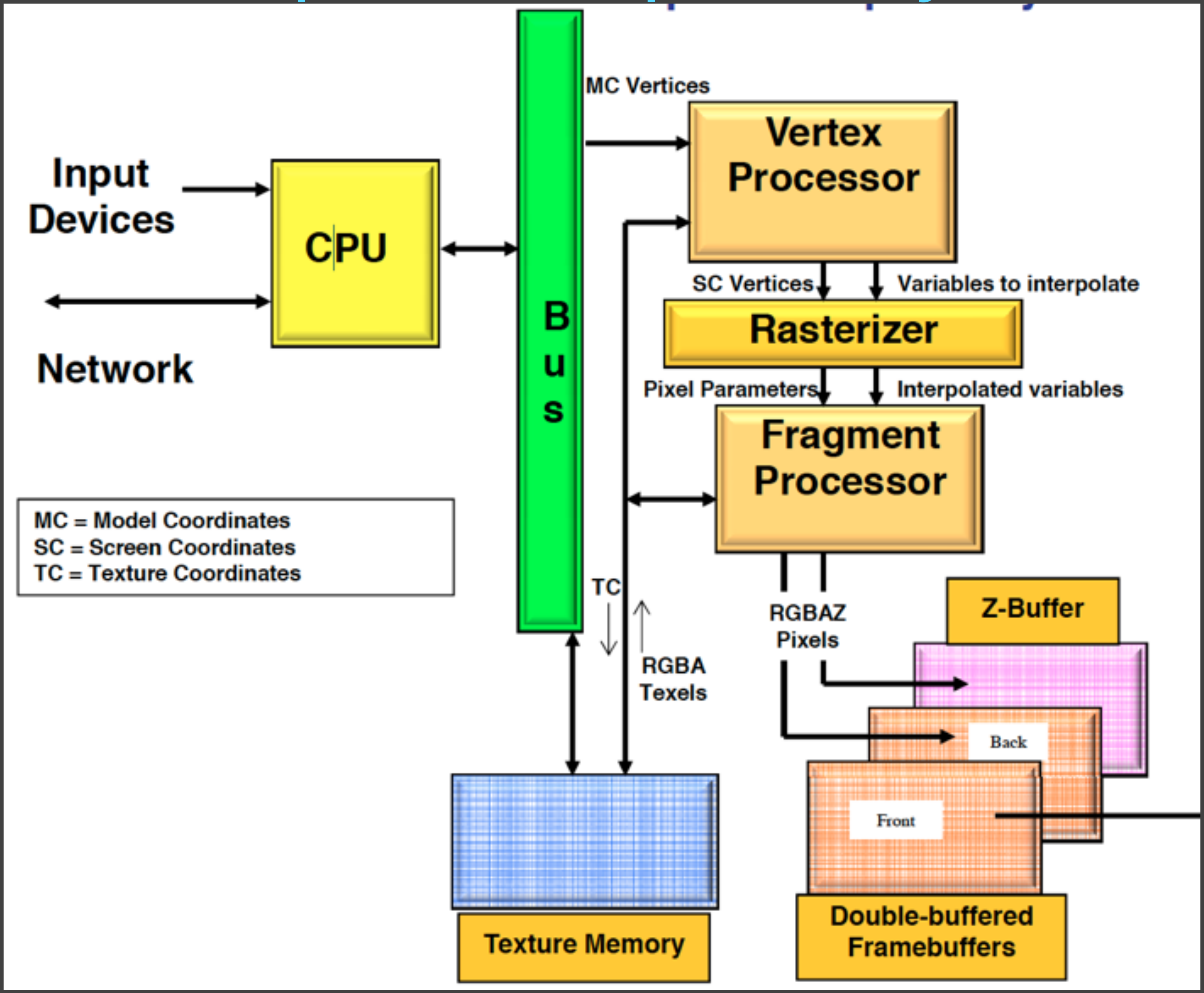
- Prelim will be in homework hand back room after class
  - Not before
- Solutions at end of class

© 2015 Kavita Bala

State of the art in GPUs

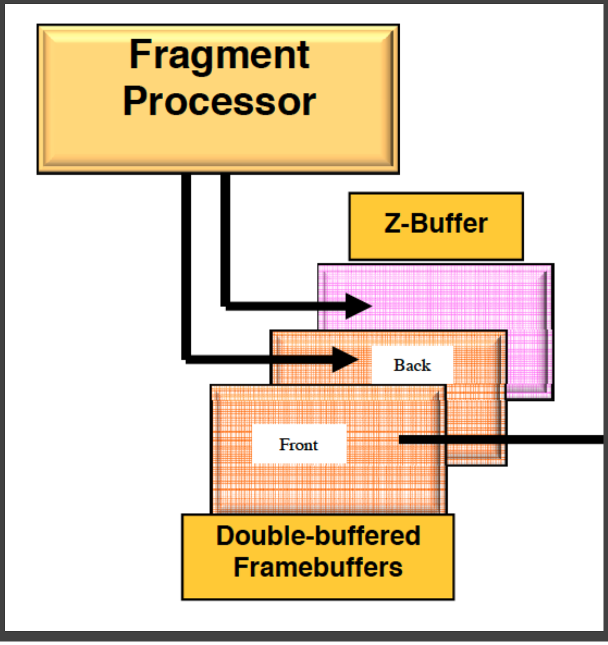
Unreal Engine 4 2015 Features Trailer

Computer Graphics System



The Framebuffer

- RGB
  - floats for HDR and compute
- Alpha
  - transparency
- Z-buffer
  - hidden surface removal
- Double buffering
  - avoid tearing



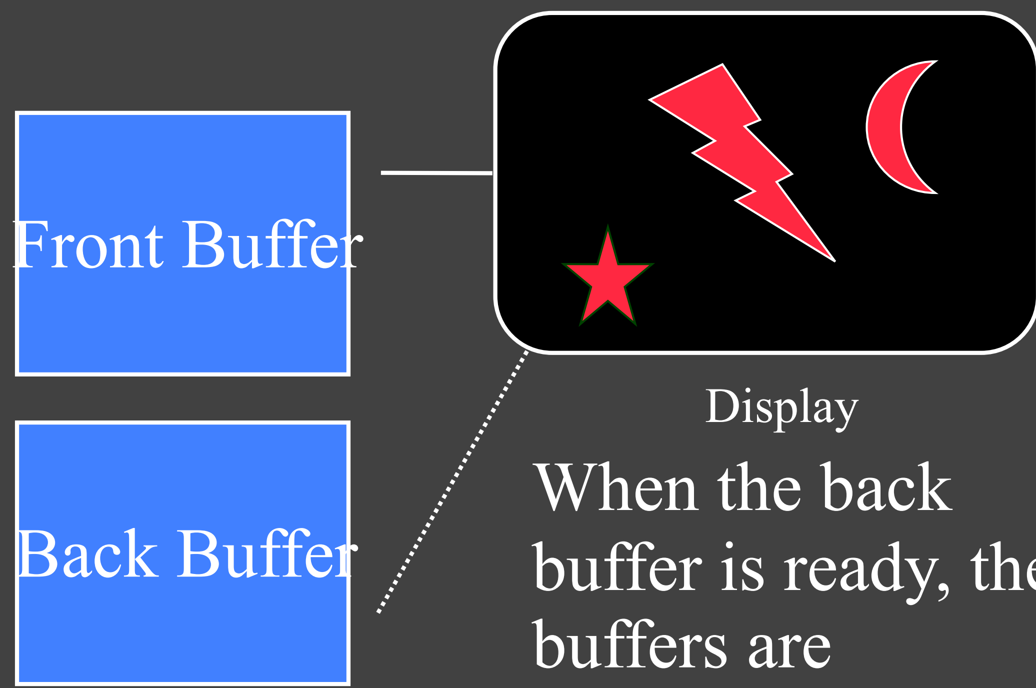
© Kavita Bala, Computer Science, Cornell University

Double buffering

- The monitor displays one image at a time
- Tearing/popping
- Use two buffers: one front and one back

As the front buffer is displayed...

the back buffer is where graphics data is sent to be rendered



When the back buffer is ready, the buffers are swapped

© Kavita Bala, Computer Science, Cornell University

Buffers, buffers, buffers!!!

A-buffer - Carpenter, 1984
G-buffer - Saito & Takahashi, 1991
M-buffer - Schneider & Rossignac, 1995
P-buffer - Yuan & Sun, 1997
T-buffer - Hsiung, Thibadeau & Wu, 1990
W-buffer - 3dfx, 1996?
Z-buffer - Catmull, 1973 (?)
ZZ-buffer - Salesin & Stolfi, 1989

Accumulation Buffer - Haeberli & Akeley, 1990
Area Sampling Buffer - Sung, 1992
Back Buffer - Baum, Cohen, Wallace & Greenberg, 1986
Close Objects Buffer - Telea & van Overveld, 1997
Color Buffer
Compositing Buffer - Lau & Wiseman, 1994
Cross Scan Buffer - Tanaka & Takahashi, 1994
Delta Z Buffer - Yamamoto, 1991
Depth Buffer - 1984
Depth-Interval Buffer - Rossignac & Wu, 1989
Double Buffer - 1993

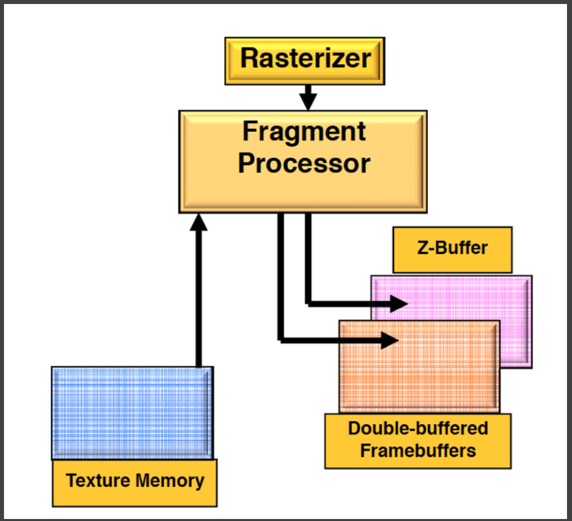
Escape Buffer - Hepting & Hart, 1995
Frame Buffer - Kajiya, Sutherland & Cheadle, 1975
Hierarchical Z-Buffer - Greene, 1993
Item Buffer - Weghorst, Hooper & Greenberg, 1984
Light Buffer - Haines & Greenberg, 1986
Mesh Buffer - Deering, 1995
Normal Buffer - Curington, 1985
Picture Buffer - Ollis & Borgwardt, 1988
Pixel Buffer - Peachey, 1987
Ray Distribution Buffer - Shinya, 1994
Ray-Z-Buffer - Lamparter, Muller & Winckler, 1990
Refreshing Buffer - Basil, 1977
Sample Buffer - Ke & Change, 1993
Shadow Buffer - GIMP, 1999
Sheet Buffer - Mueller & Crawfis, 1998
Stencil Buffer - 1992
Super Buffer - Gharachorloo & Pottle, 1985
Super-Plane Buffer - Zhou & Peng, 1992
Triple Buffer
Video Buffer - Scherson & Punte, 1987
Volume Buffer - Sramek & Kaufman, 1999

Source: Eric Haines

© Kavita Bala, Computer Science, Cornell University

The Fragment Processor

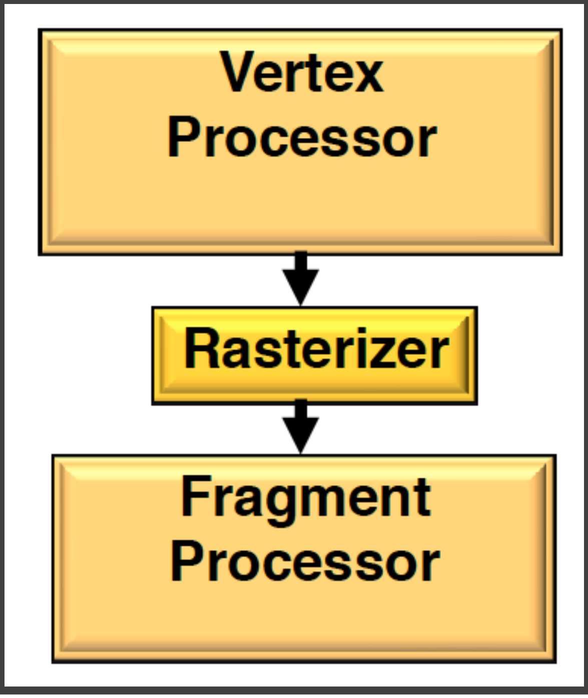
- Fragment
  - Pixel to be
- Produce RGBA output
- Shader
  - Color computation
  - Texturing
  - Per-pixel lighting
  - Fog
  - Blending
  - Discarding fragments



© Kavita Bala, Computer Science, Cornell University

The Rasterizer

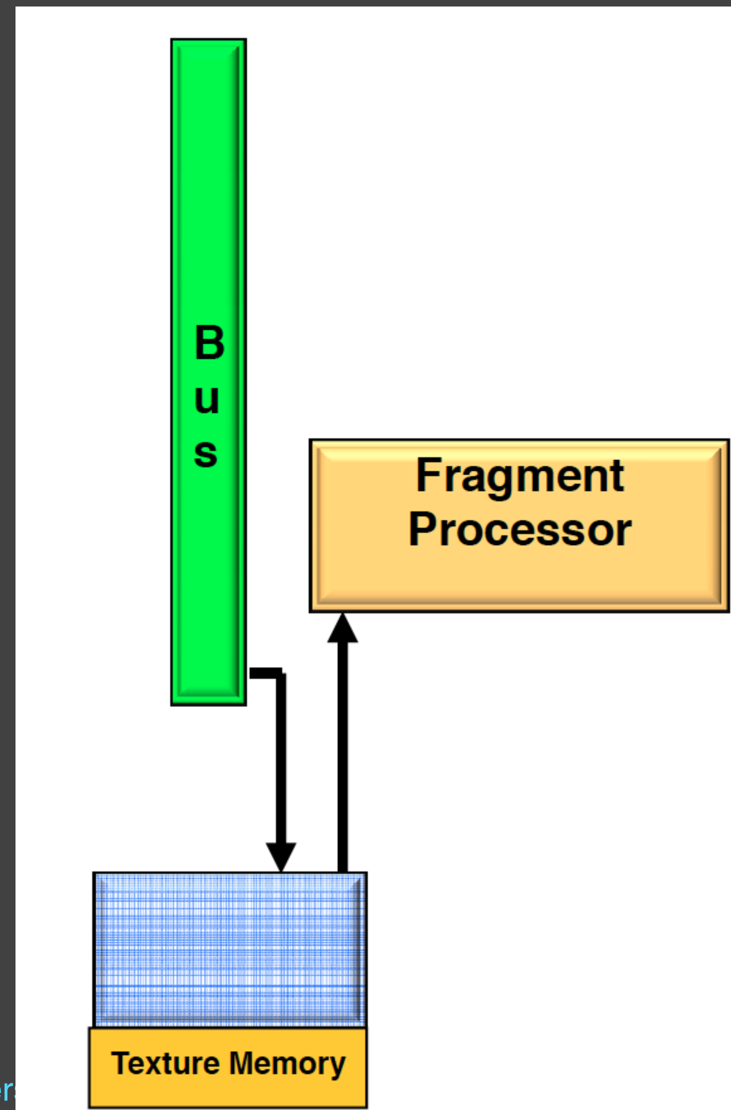
- Screen space coordinates into lines, polys
- Interpolates
  - x,y
  - RGB
  - alpha
  - z
  - intensities
  - normals
  - texture coordinates
  - custom values given by shaders



© Kavita Bala, Computer Science, Cornell University

Texture Mapping

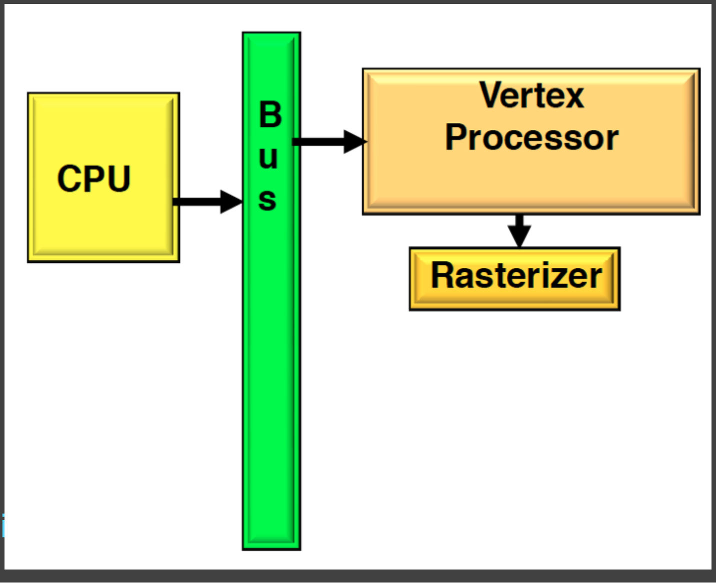
- Workhorse



© Kavita Bala, Computer Science, Cornell Univer

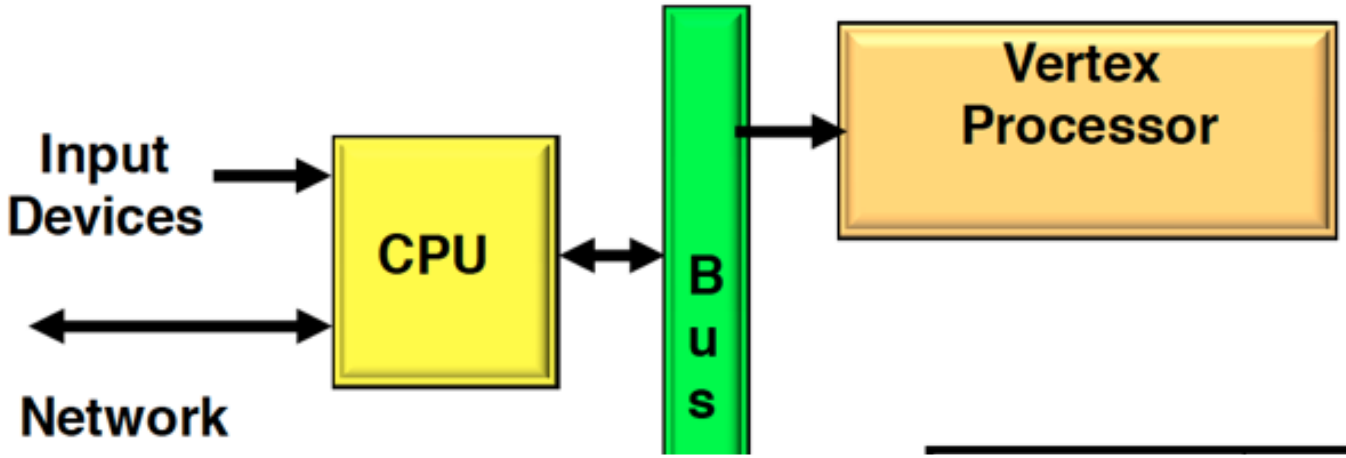
Vertex Processor

- Coordinates
  - in model units, out pixel units
- Shaders
  - Vertex transformations
  - Normal transformations, Normal normalization
  - Per-vertex lighting
- Fixed function
  - View volume clipping
  - Homogeneous division
  - Viewport mapping
  - Backface culling



© Kavita Bala, Computer Science, Cornell Universi

CPU and Bus

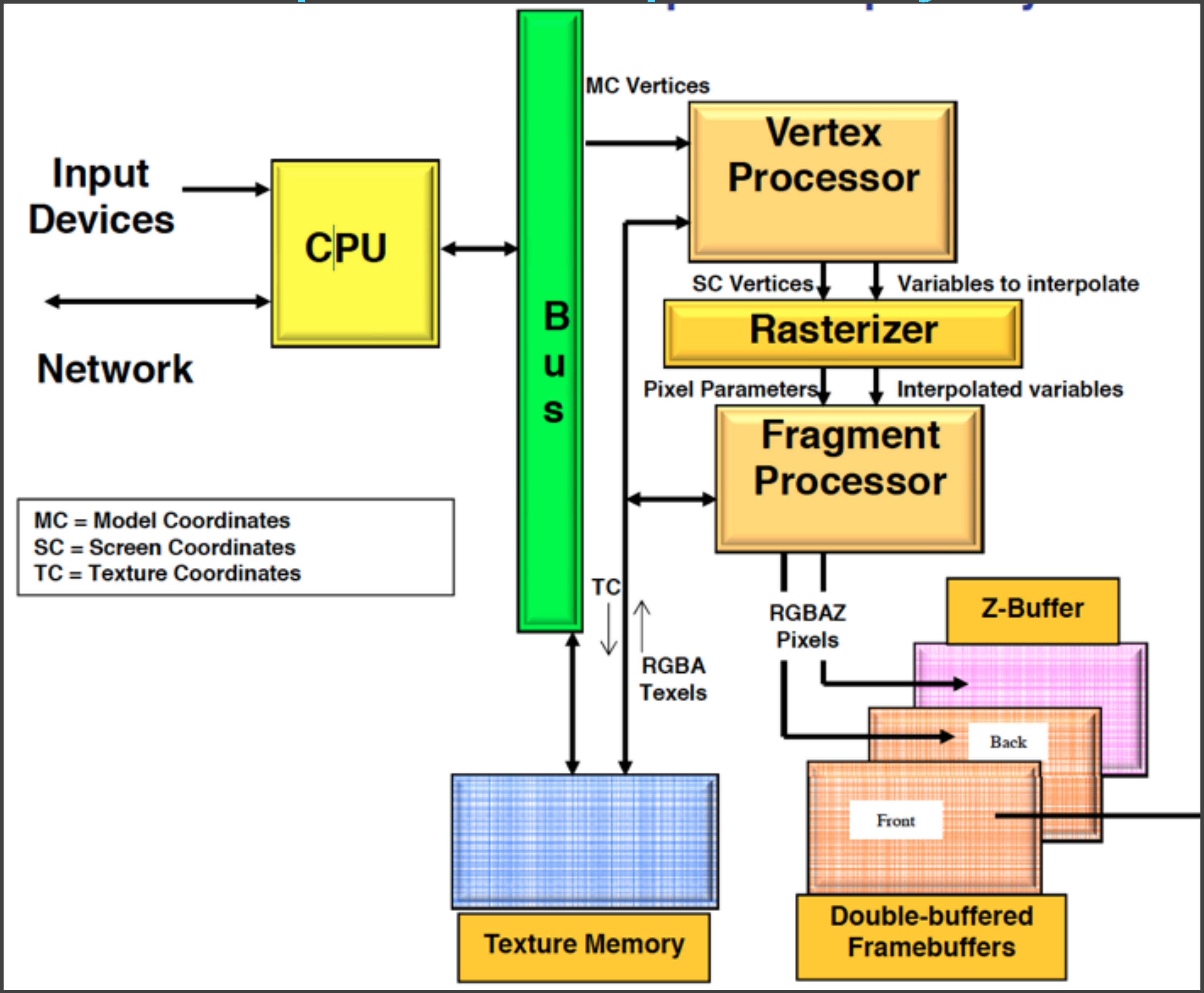


PCI Express link performance[21][22]

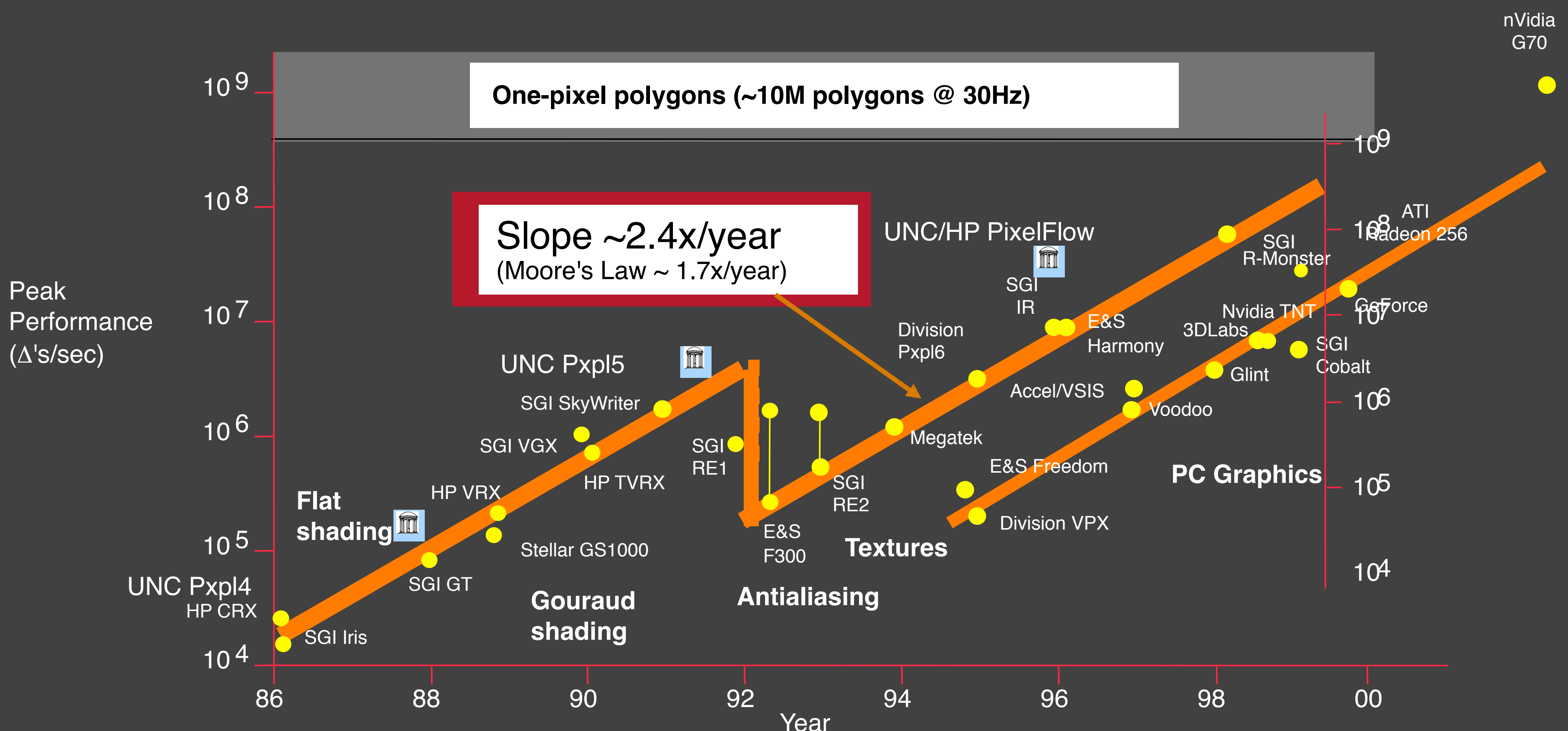
PCI Express version	Line code	Transfer rate[a]	Bandwidth	
			Per lane[a]	In a ×16 (16-lane) slot[a]
1.0	8b/10b	2.5 GT/s	2 Gbit/s (250 MB/s)	32 Gbit/s (4 GB/s)
2.0	8b/10b	5 GT/s	4 Gbit/s (500 MB/s)	64 Gbit/s (8 GB/s)
3.0	128b/130b	8 GT/s	7.877 Gbit/s (984.6 MB/s)	126.032 Gbit/s (15.754 GB/s)
4.0	128b/130b	16 GT/s	15.754 Gbit/s (1969.2 MB/s)	252.064 Gbit/s (31.508 GB/s)

© Kavita Bala, Computer Science, Cornell University

Computer Graphics System



GPUs Faster than Moore's Law



Graph courtesy of Professor John Poulton (from Eric Haines)

© Kavita Bala, Computer Science, Cornell University

GPU Parallelism

- GPUs are SIMD machines

- They exploit 2 types of parallelism
  - Data: (vertex, triangle, fragment) parallelism
    - Process k triangles in parallel, m fragments in parallel
  - Task: pipeline
    - Pipeline in GPUs up to 800-1000 clocks long (compare to 10-20 on CPUs)

© Kavita Bala, Computer Science, Cornell University

Multi-Threaded SIMD

- Very fine grain threads

- Latency
  - Hide latency by switching to other threads
  - Shared register file (very large, 65k 32-bit registers now)
  - Also prefetching

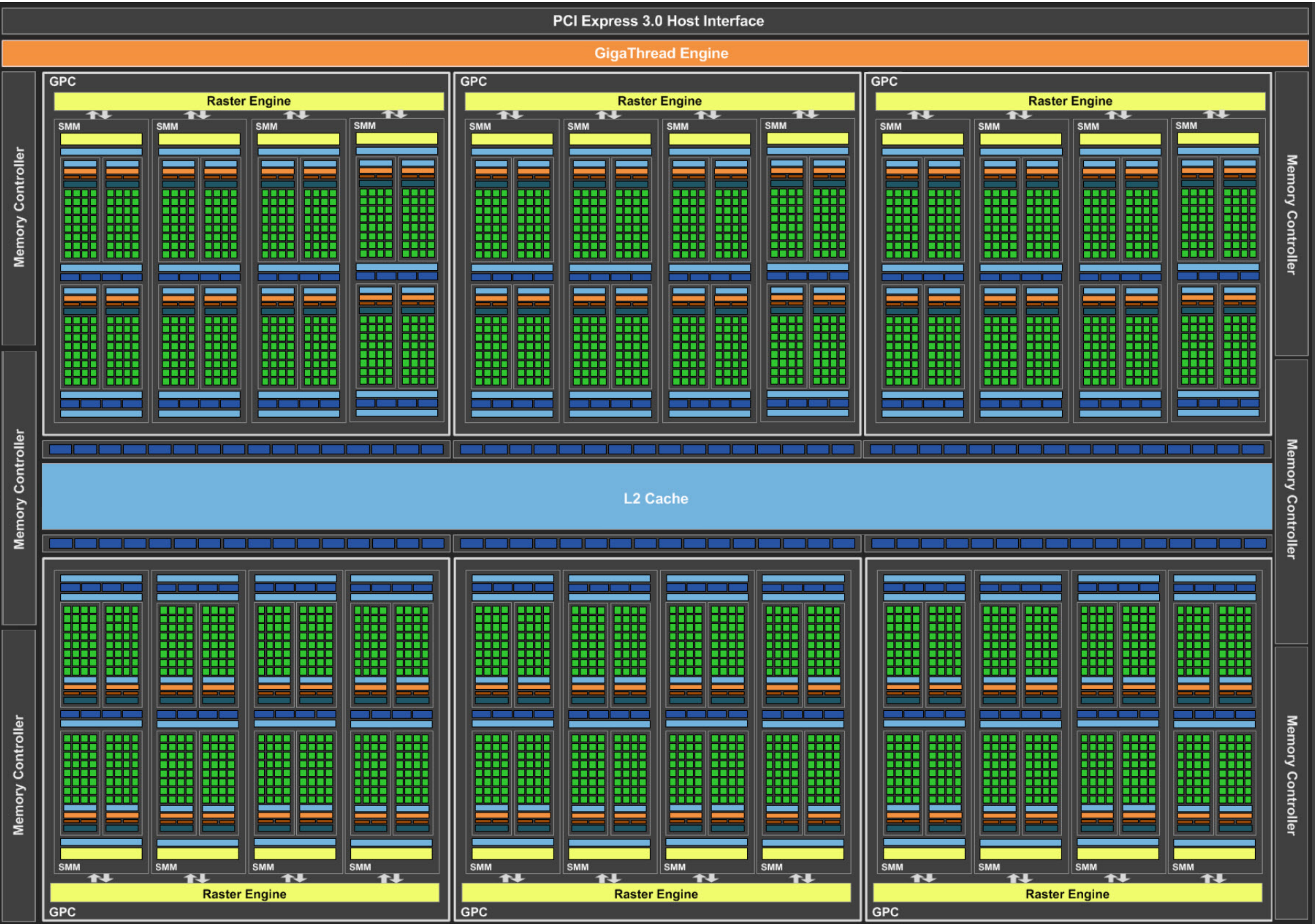
© Kavita Bala, Computer Science, Cornell University

Architectural Trends

- More general purpose
- More shaders: vertex, pixel, geometry, tesselation

- Longer shaders
  - Length of shaders: 16, 128, … unbounded
- More bits
  - More texturing: more, bigger, and greater precision
  - Better floating point
  - Better HDR support

- More SIMD cores
  - More parallelism



© Kavita Bala, Computer Science, Cornell University

GTX TITAN GPU Engine Specs:

CUDA Cores	2688
Base Clock (MHz)	837
Boost Clock (MHz)	876
Texture Fill Rate (billion/sec)	187.5

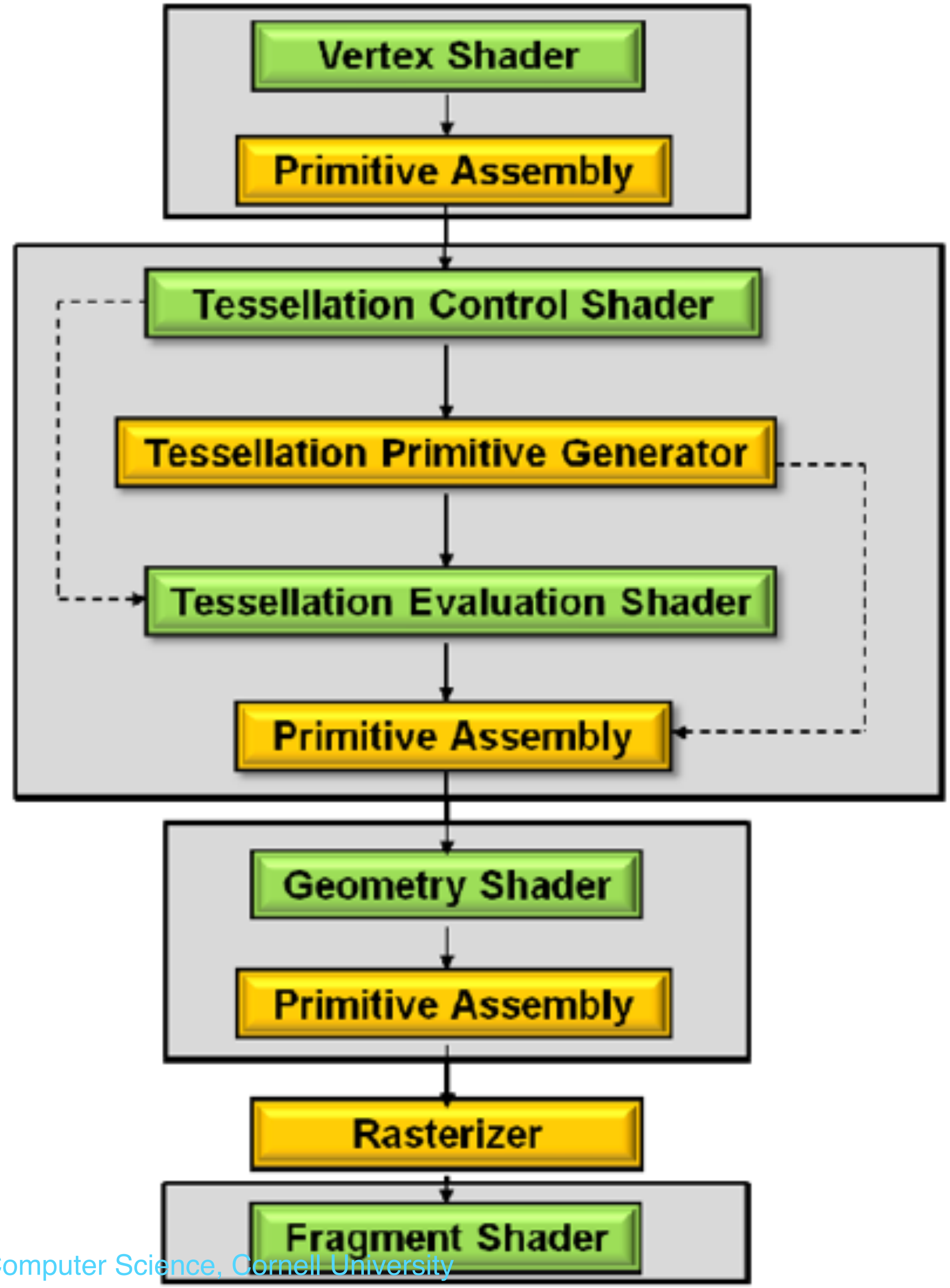
GTX TITAN Memory Specs:

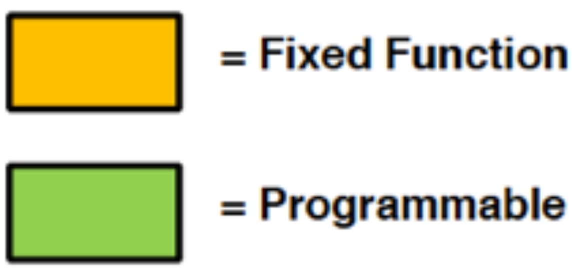
Memory Clock	6.0 Gbps
Standard Memory Config	6144 MB
Memory Interface	GDDR5
Memory Interface Width	384-bit GDDR5
Memory Bandwidth (GB/sec)	288.4

GTX TITAN Support:

Important Technologies	GPU Boost 2.0, PhysX, TXAA, NVIDIA G-SYNC-ready, SHIELD-ready
Other Supported Technologies	3D Vision, CUDA, Adaptive VSync, FXAA, NVIDIA Surround, SLI-ready
OpenGL	4.4
Bus Support	PCI Express 3.0
Certified for Windows 7, Windows 8, Windows Vista, or Windows XP	Yes
3D Vision Ready	Yes
Microsoft DirectX	12 API
Blu Ray 3D	Yes
3D Gaming	Yes
3D Vision Live (Photos and Videos)	Yes

OpenGL 4.2+





© Kavita Bala, Computer Science, Cornell University

GPU Pipeline

- Vertex shader
  - Model and View Transform
  - Vertex Shading

- Tessellation Shader
  - Create subdivision surfaces
- Geometry Shader
  - Create/destroy primitives
- Fragment Shader
  - Fully general and really powerful

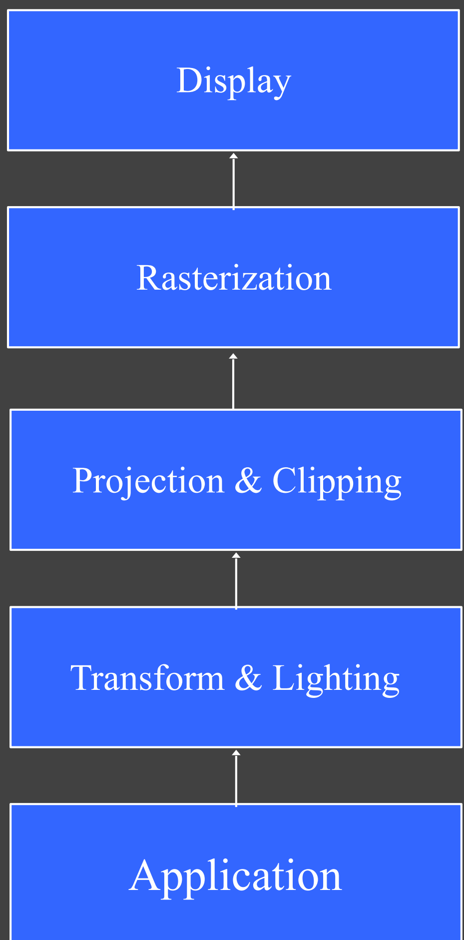
© Kavita Bala, Computer Science, Cornell University

Tessellation Shaders

- Adaptive subdivision
  - Based on size, curvature, screen space extent
- Coarse models with
  - GPU compression
  - detailed displacement maps w/o detailed geometry
  - subdivision rules
  - adapt quality to level of detail
    - smoother silhouettes
  - Terrain proof of concept, Demo

© Kavita Bala, Computer Science, Cornell University

Brief History



The dark ages (early-mid 1990's), when there were only frame buffers for normal PC's.

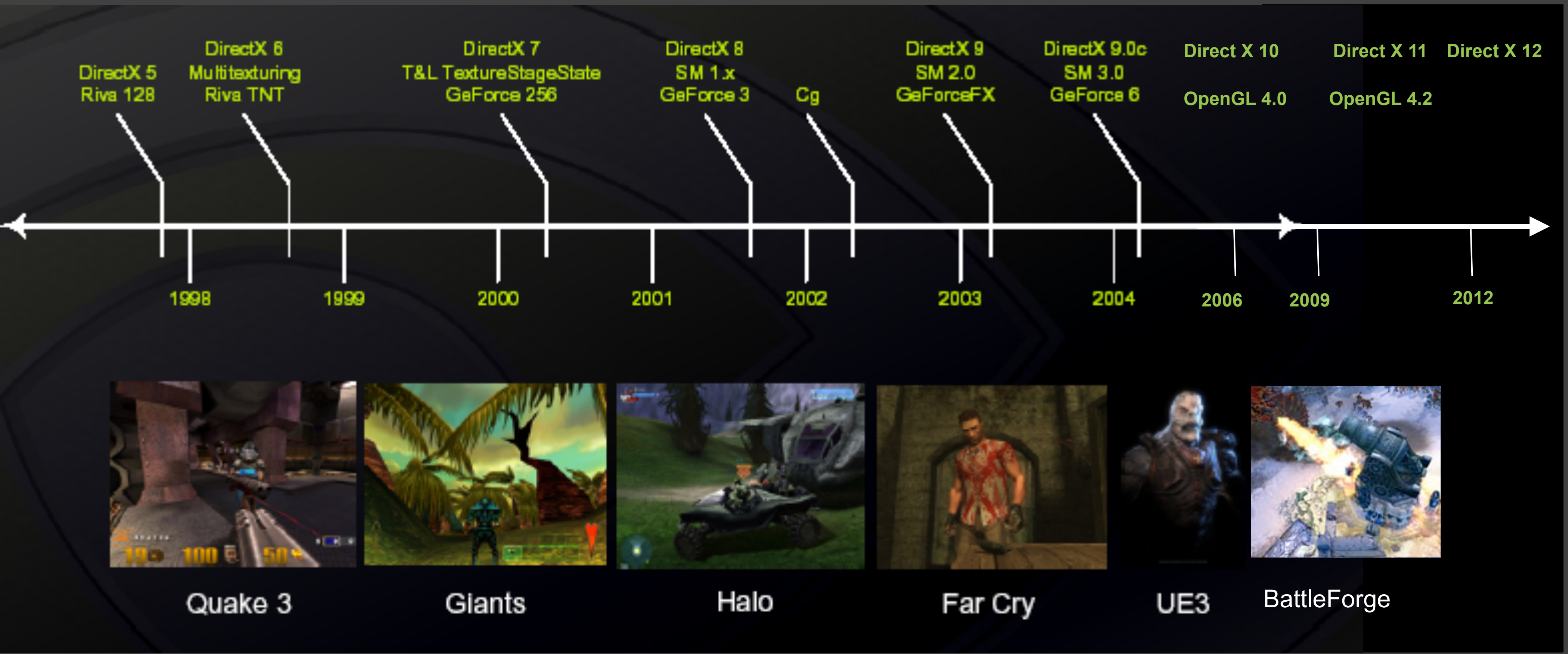
Some accelerators were no more than a simple chip that sped up linear interpolation along a single span, so increasing fill rate.

This is where pipelines start for PC commodity graphics, prior to Fall of 1999.

This part of the pipeline reaches the consumer level with the introduction of the NVIDIA GeForce256.

Hardware today has moved traditional application processing into the graphics accelerator.

© Kavita Bala, Computer Science, Cornell University



© Kavita Bala, Computer Science, Cornell University

1997

© Kavita Bala, Computer Science, Cornell University

Era of GPUs

Nvidia's GeForce 256 was the first graphics chip to actually be called a GPU, based on the addition of a hardware-based transformation and lighting engine (T&L).

This engine allowed the graphics chip to undertake the heavily floating-point intensive calculations of transforming the 3D objects and scenes – and their associated lighting – into the 2D representation of the rendered image. Previously, this computation was undertaken by the CPU, which could easily bottleneck with the workload, and tended to limit available detail.

Nvidia Grass Demo (GeForce 256)