GPU Architecture and Function

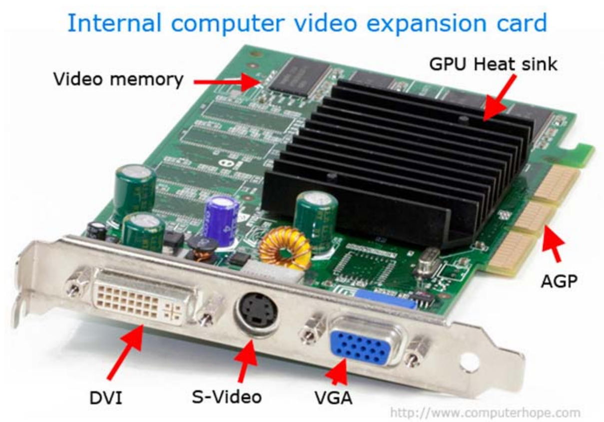
Michael Foster and Ian Frasch

Overview

- What is a GPU?
- How is a GPU different from a CPU?
- The graphics pipeline
- History of the GPU
- GPU architecture
- Optimizations
- GPU performance trends
- Current development

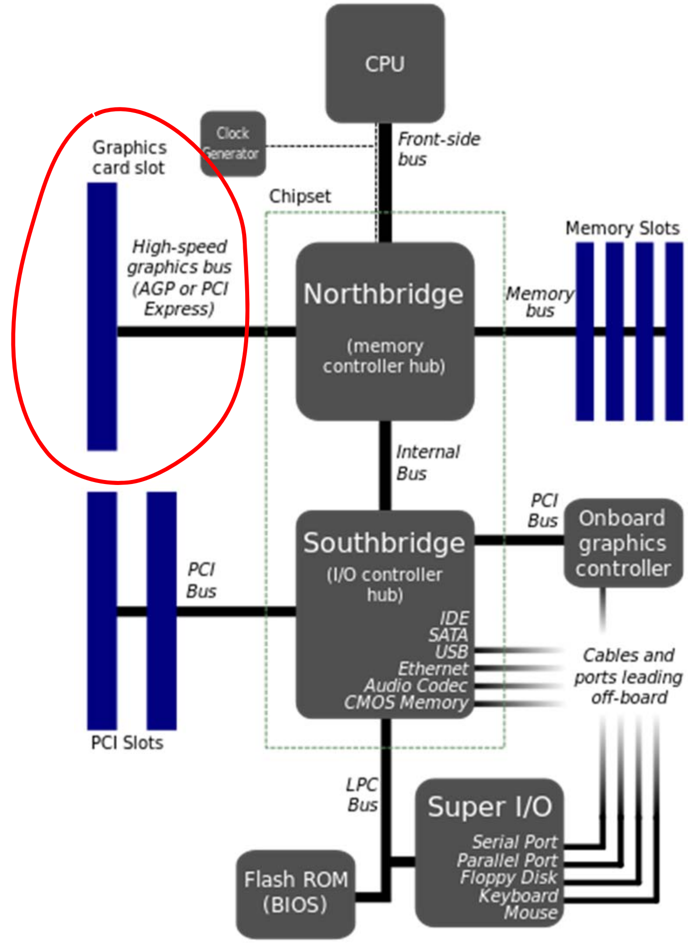
What is a GPU?

- Dedicated graphics chip that handles all processing required for rendering 3D objects on the screen
- Typically placed on a video card, which contains its own memory and display interfaces (HDMI, DVI, VGA, etc)
- Primitive GPUs were developed in the 1980s, although the first “complete” GPUs began in the mid 1990s.



Systems level view

- Video card connected to motherboard through PCI-Express or AGP (Accelerated Graphics Port)
- Northbridge chip enables data transfer between the CPU and GPU
- Graphics memory on the video card contains the pixel RGB data for each frame



How is a GPU different from a CPU?

<u>Throughput more important than latency</u>

- High throughput needed for the huge amount of computations required for graphics
- Not concerned about latency because human visual system operates on a much longer time scale
  - 16 ms maximum latency at 60 Hz refresh rate
  - Long pipelines with *many* stages; a single instruction may thousands of cycles to get through the pipeline.

CPU	Generate Frame 0	Generate Frame 1	Generate Frame 2
GPU	Idle	Render Frame 0	Render Frame 1

Latency

How is a GPU different from a CPU?

Extremely parallel

- Different pixels and elements of the image can be operated on independently
- Hundreds of cores executing at the same time to take advantage of this fundamental parallelism

Inputs and Outputs

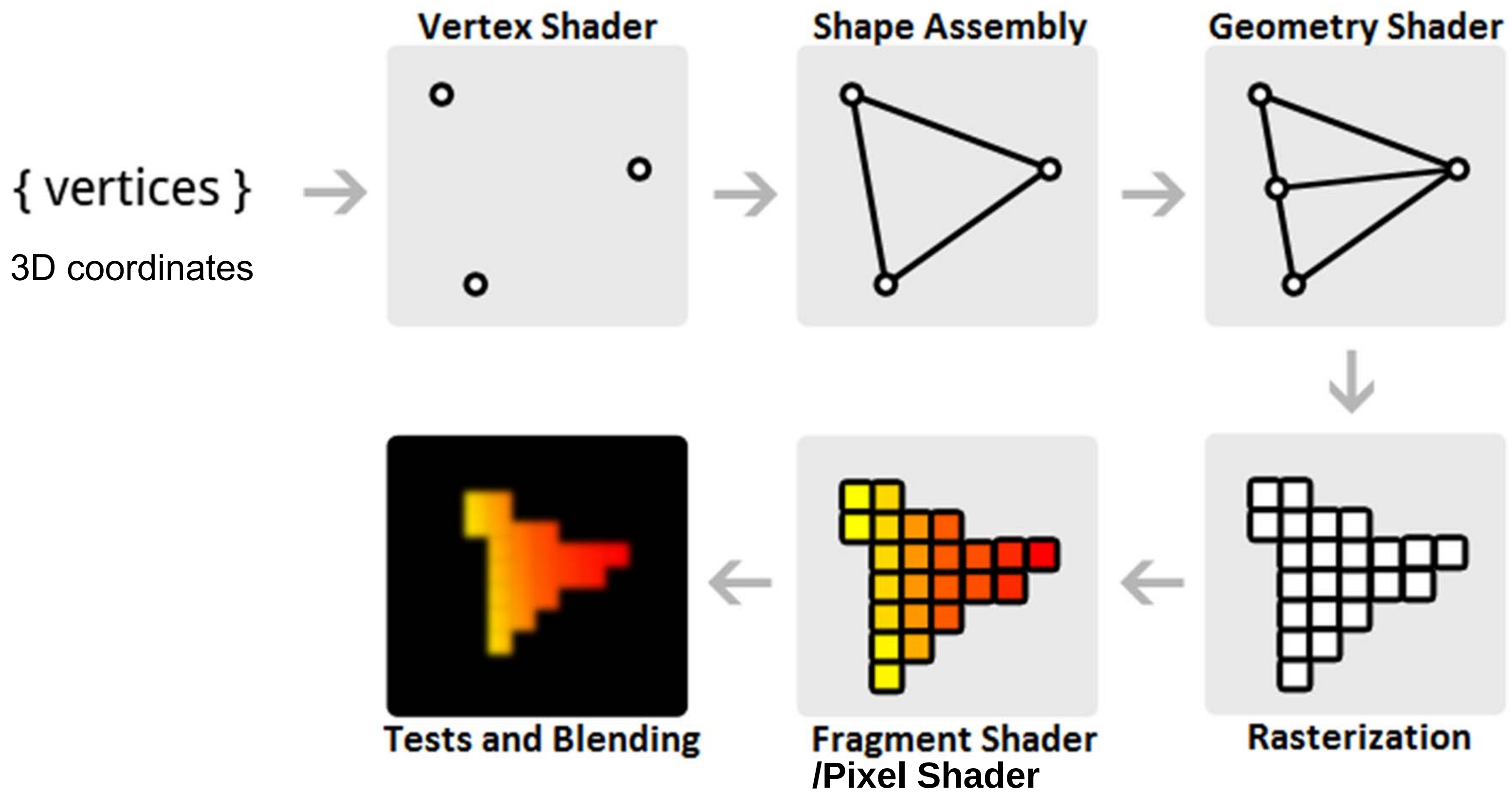
Inputs to GPU (from the CPU/memory):

- Vertices (3D coordinates) of objects
- Texture data
- Lighting data

Outputs from GPU:

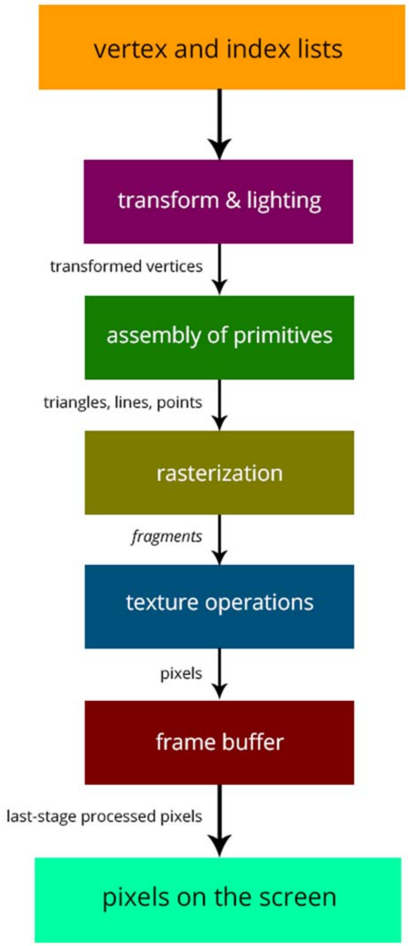
- Frame buffer
    - Placed in a specific section of graphics memory
    - Contains RGB values for each pixel on the screen
    - Data is sent directly to display

The Graphics Pipeline: A Visual



The Graphics Pipeline

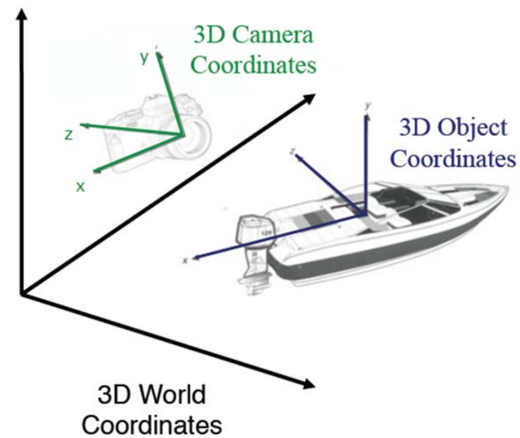
- The GPU completes every stage of this computational pipeline



Transformations

Camera transformation

- Convert vertices from 3D world coordinates to 3D camera coordinates, with the camera (user view) as the origin



Projection transformation

- Convert vertices from 3D camera coordinates to 2D screen view coordinates that the user will see

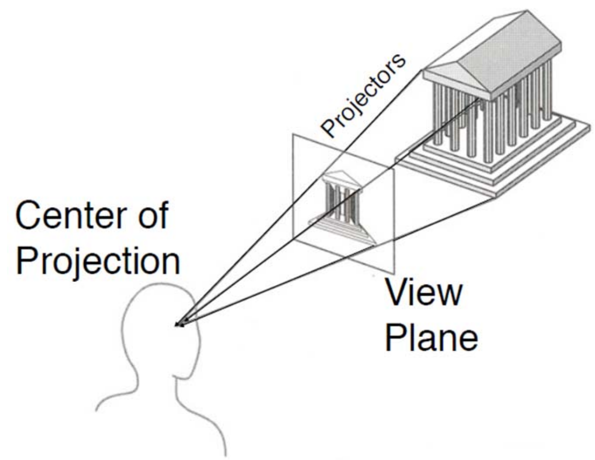
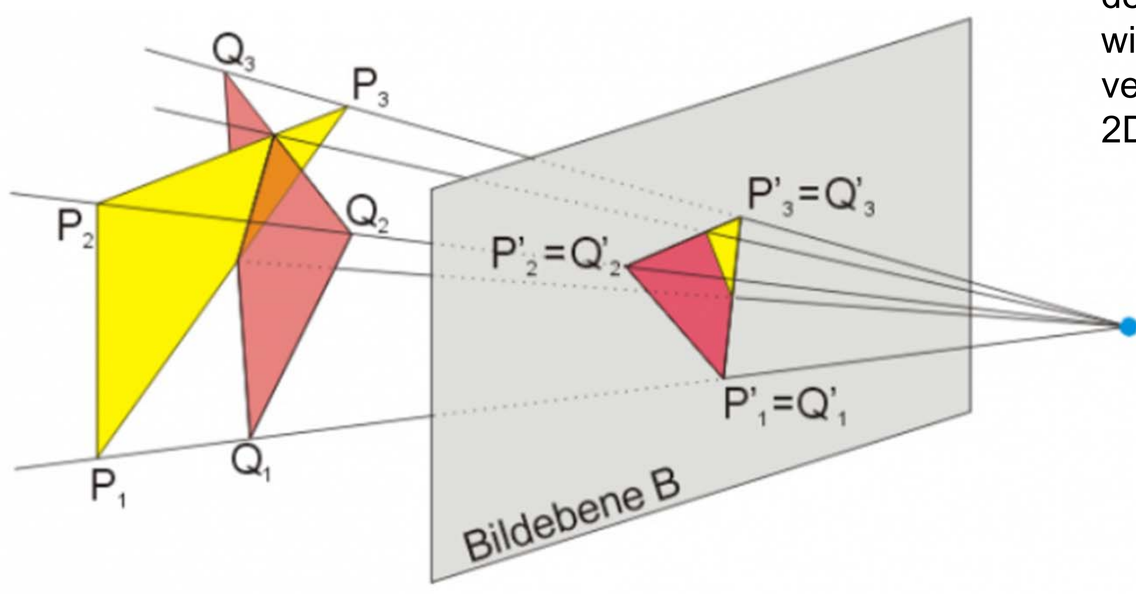


Illustration of 3D-2D Projection

(With overlapping vertices)

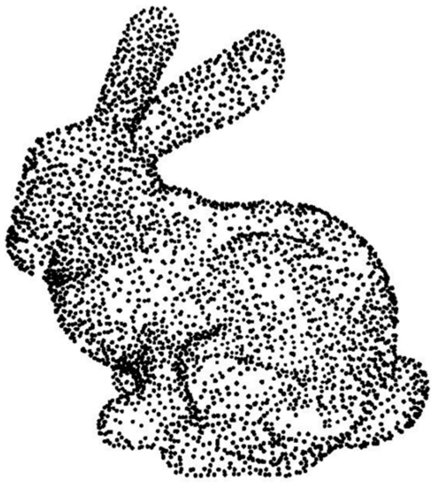
Depth values in *Z buffer* determine which triangle will be visible if two vertices map to the same 2D coordinate



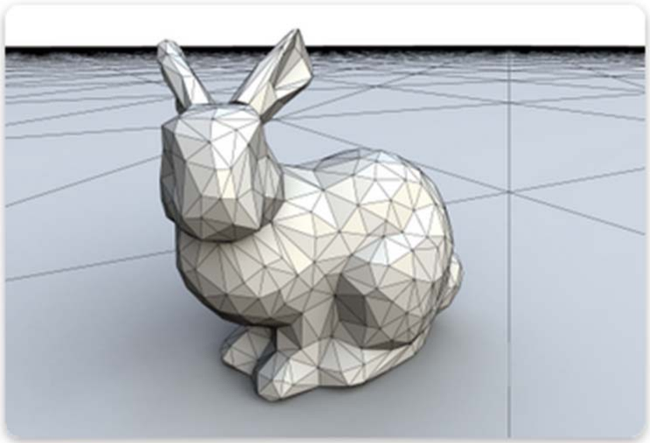
Transformations

- These transformations simply modify vertices, so they are done by *vertex shaders*
- Transform computations are heavy on matrix multiplication
- *Each vertex can be transformed <u>independently</u>*
    - Data Parallelism

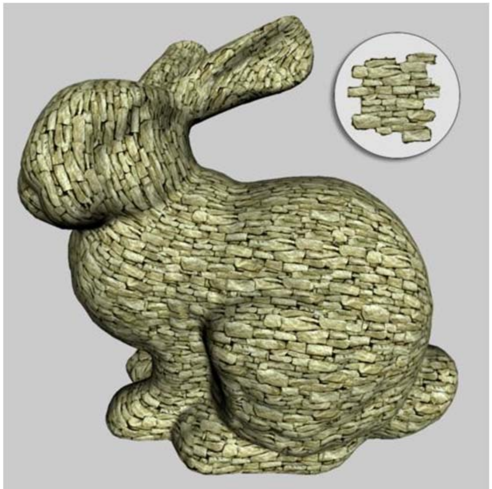
Example renderings



Vertices
(Point-cloud)

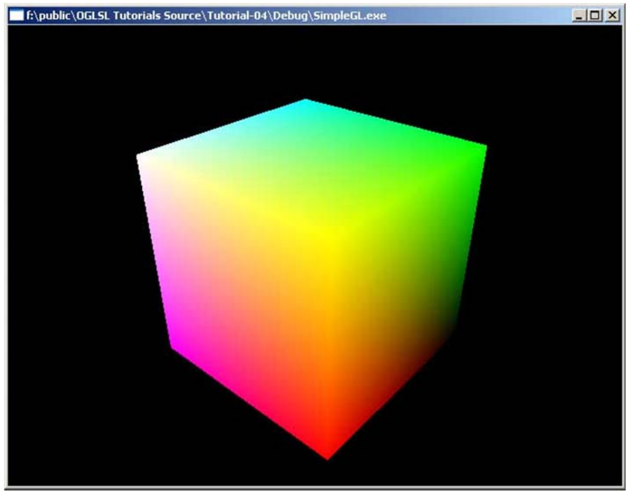


Primitives
(Triangles)



Textures

More renderings



Simple shaded 6-vertex cube; each vertex has a color associated with it, the pixel shader blends the colors.

Advanced rendering

Fixed-Function to Programmable

- Earlier GPUs were fixed-function hardware pipelines
  - Software developers could set parameters (textures, light reflection colors, blend modes) but the function was completely controlled by the hardware
- In newer GPUs, portions of the pipeline are completely programmable
  - Pipeline stages are now *programs* running on processor cores inside the GPU, instead of fixed-function ASICs
  - Vertex shaders = programs running on vertex processors, fragment shaders = programs running on fragment processors
  - However, some stages are still fixed function (e.g. rasterization)

History of the GPU

1996: 3DFX Voodoo graphics card implements texture mapping, z-buffering, and rasterization, but no vertex processing

1999: GPUs implement the full graphics pipeline in *fixed-function* hardware (Nvidia GeForce 256, ATI Radeon 7500)

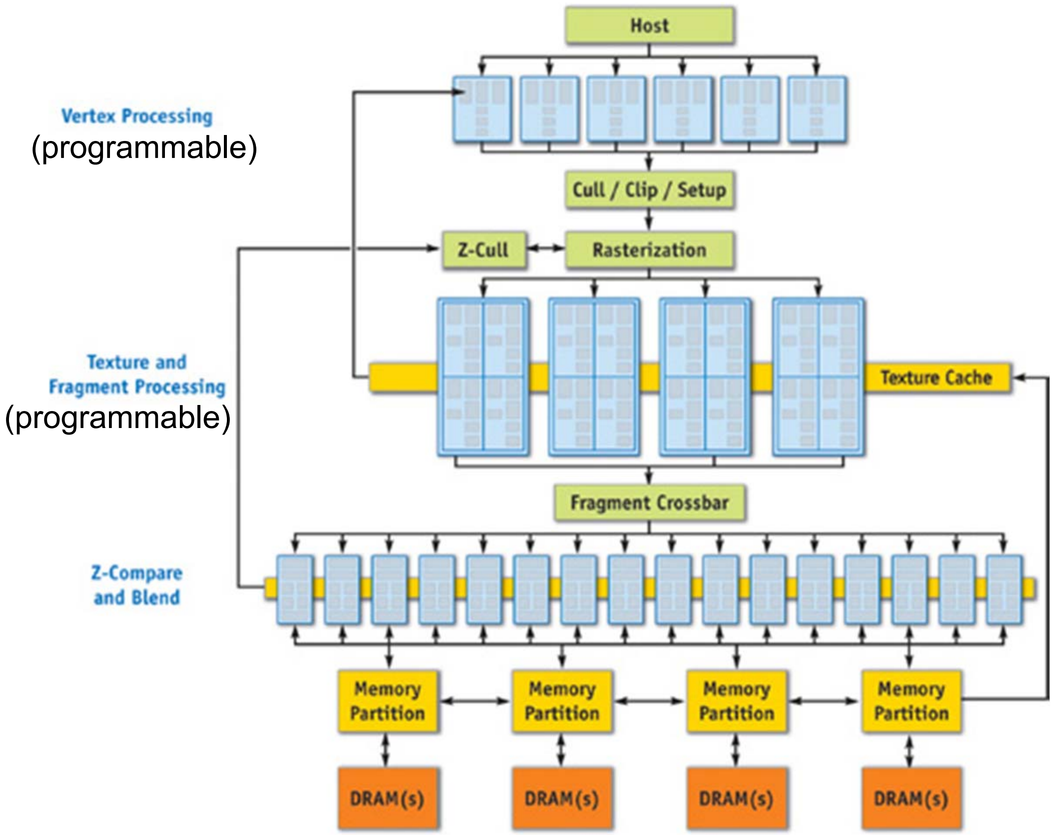
2001: *Programmable* shader pipelines (Nvidia Geforce 3)

2006: Unified shader architecture (ATI Radeon R600, Nvidia Geforce 8, Intel GMA X3000, ATI Xenos for Xbox360)

2010: General Purpose GPUs for non-graphical compute-intensive applications, Nvidia CUDA parallel programming API

2014: Unprecedented compute power
Nvidia Geforce GTX Titan Z - 8.2 TFLOPS
AMD Radeon R9 295X2 (dual GPU card) - 11.5 TFLOPS

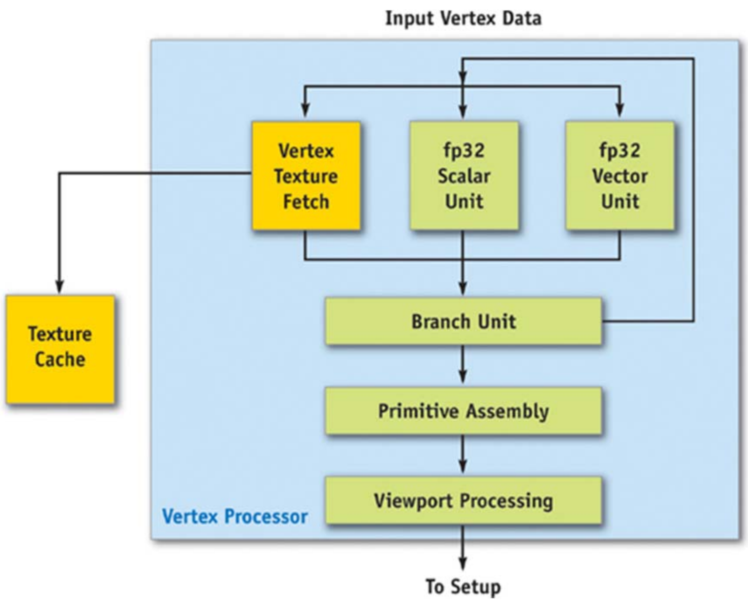
GPU Architecture



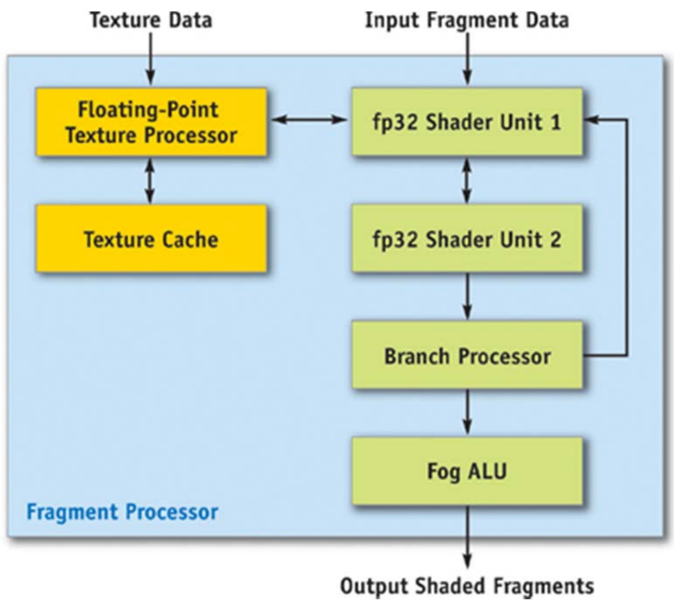
Nvidia Geforce 6 series (2004)

Shader processors

Vertex Shader core

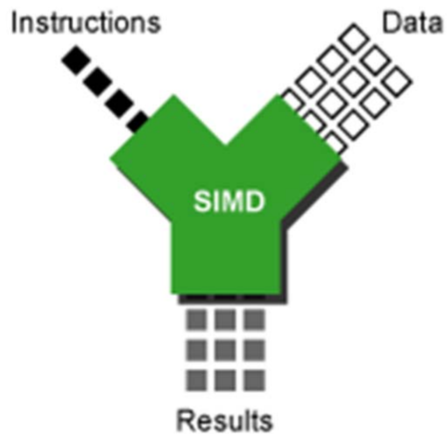


Fragment/Pixel Shader core

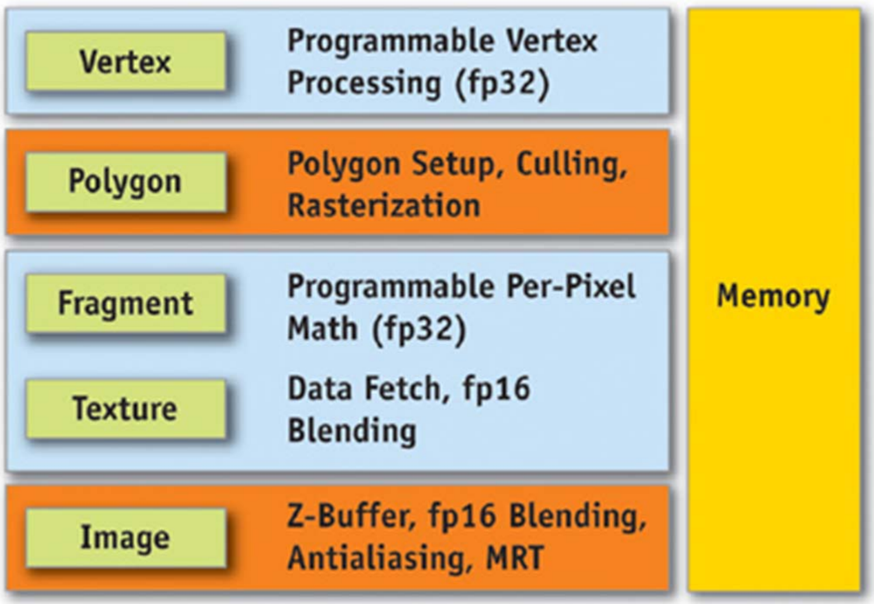


Single Instruction, Multiple Data (SIMD)

- Shader processors are generally SIMD
- A single instruction executed on *every* vertex or pixel



Functional block diagram



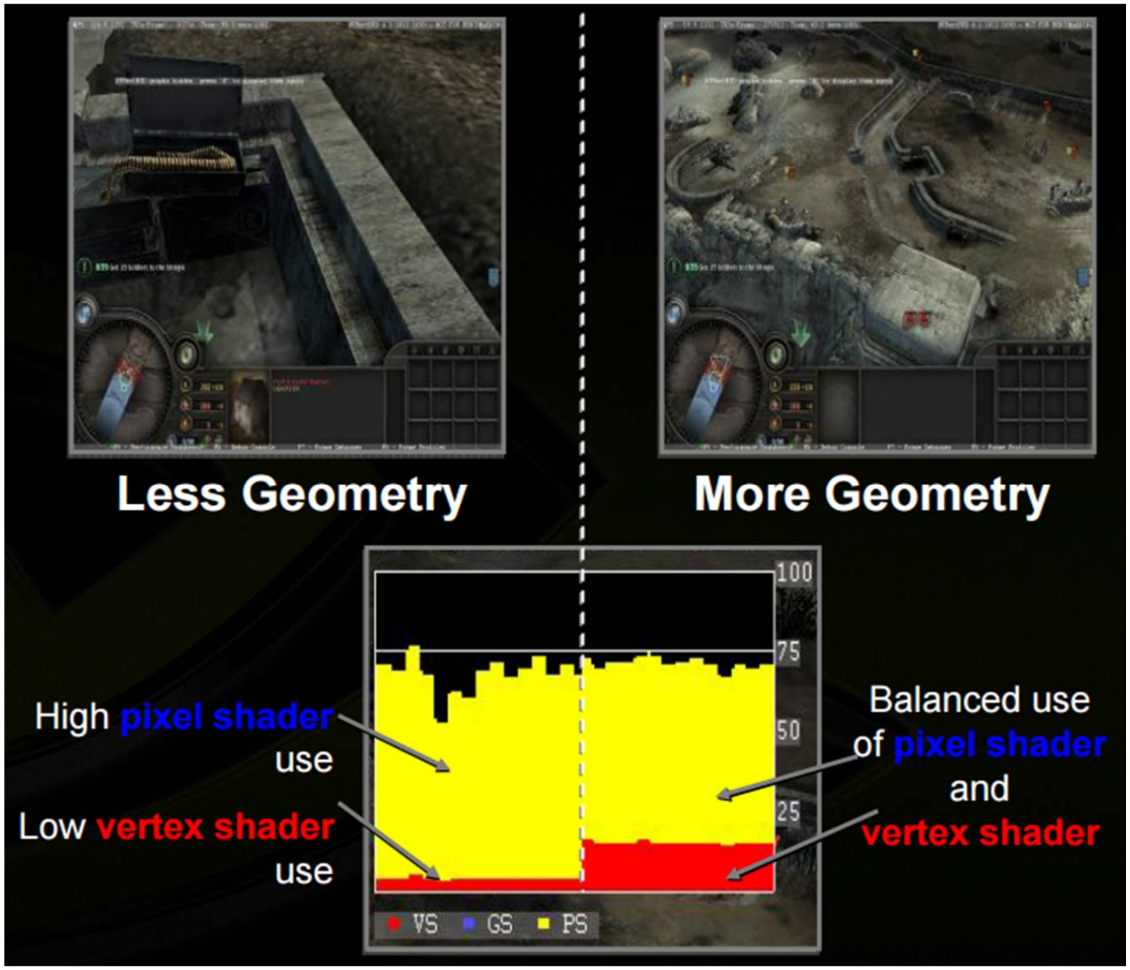
Optimizations

- Combining different types of shader cores into a single unified shader core
- Dynamic task scheduling to balance the load on all cores

Workload Distribution

- Frames with many “edges” (vertices) require more vertex shaders

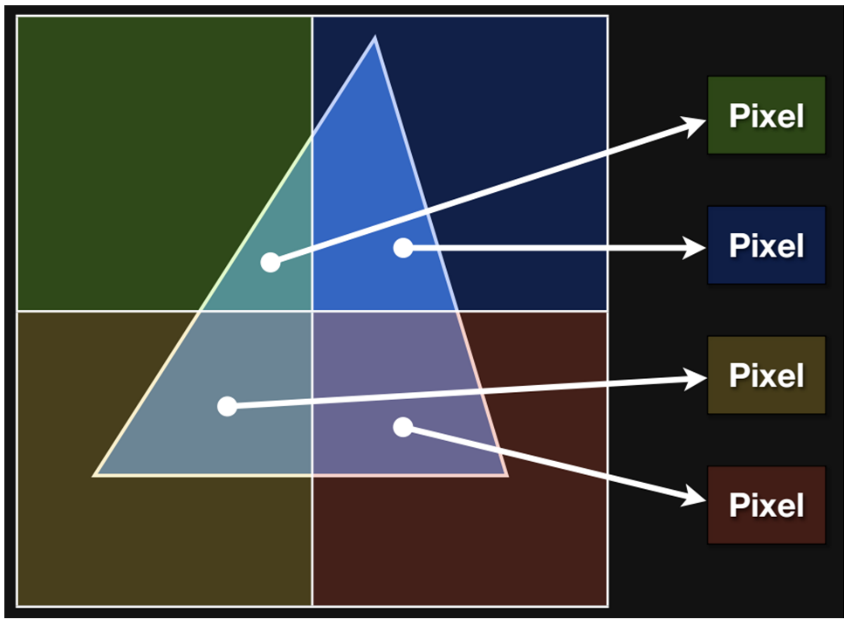
- Frames with large primitives require more pixel shaders

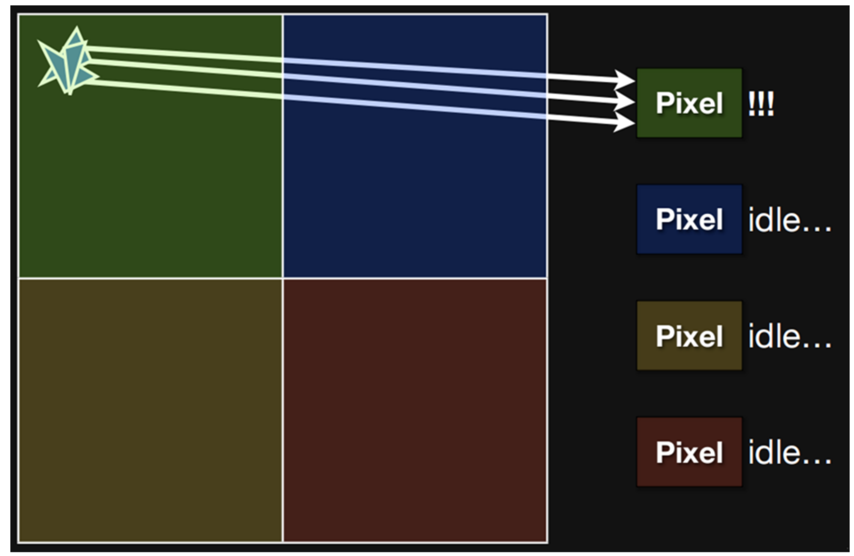


Solution: Unified Shader

- Pixel shaders, geometry shaders, and vertex shaders run on the same core - a unified shader core
  - Unified shaders limit idle shader cores
  - Instruction set shared across all shader types
  - Program determines type of shader
- Modern GPUs all use unified shader cores
- Shader cores are programmed using graphics APIs like OpenGL and Direct3D

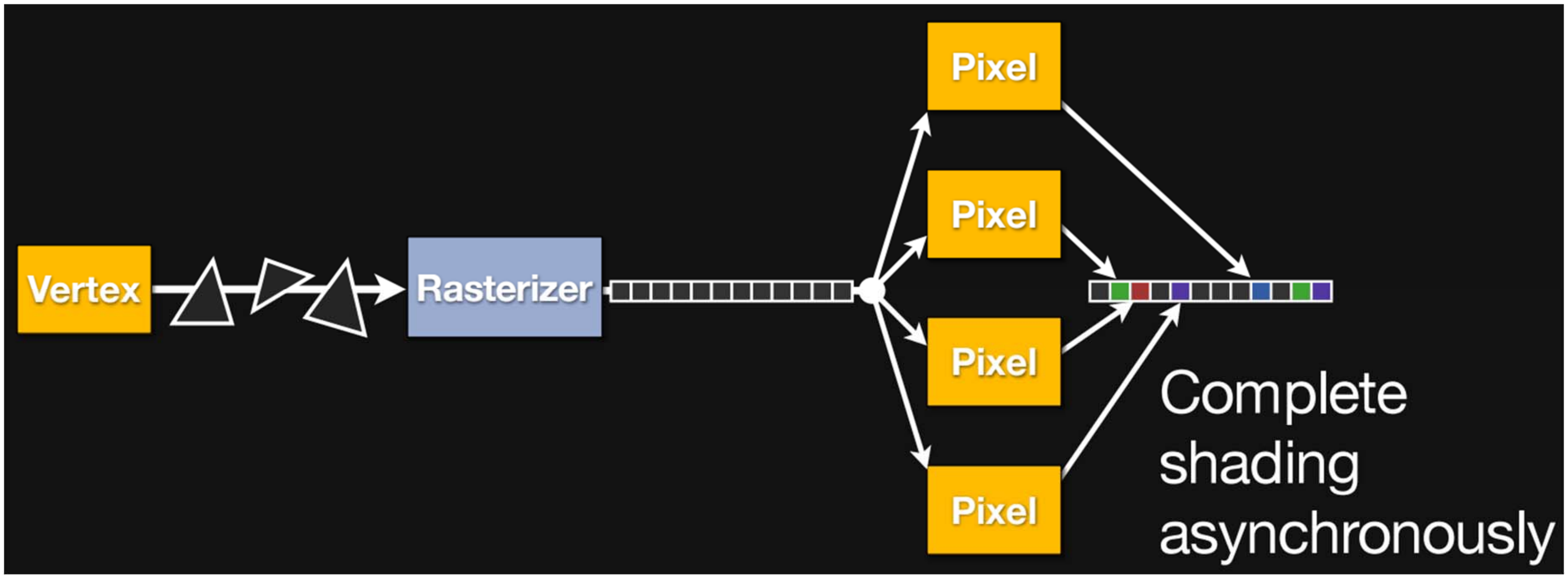
Static Task Distribution





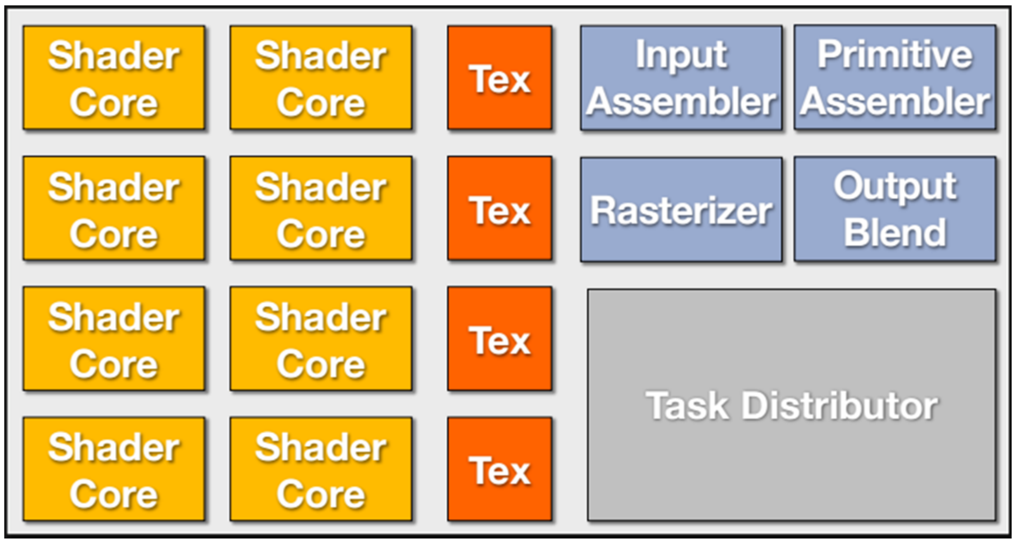
- Unequal task distribution leads to inefficient hardware usage
- Parallel processors should handle tasks of equal complexity

Dynamic Task Distribution

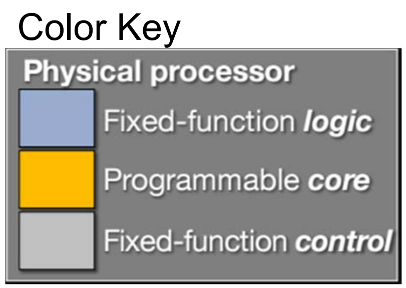


- Tasks are dynamically distributed among pixel shaders
- Slots for output are pre-allocated in output FIFO

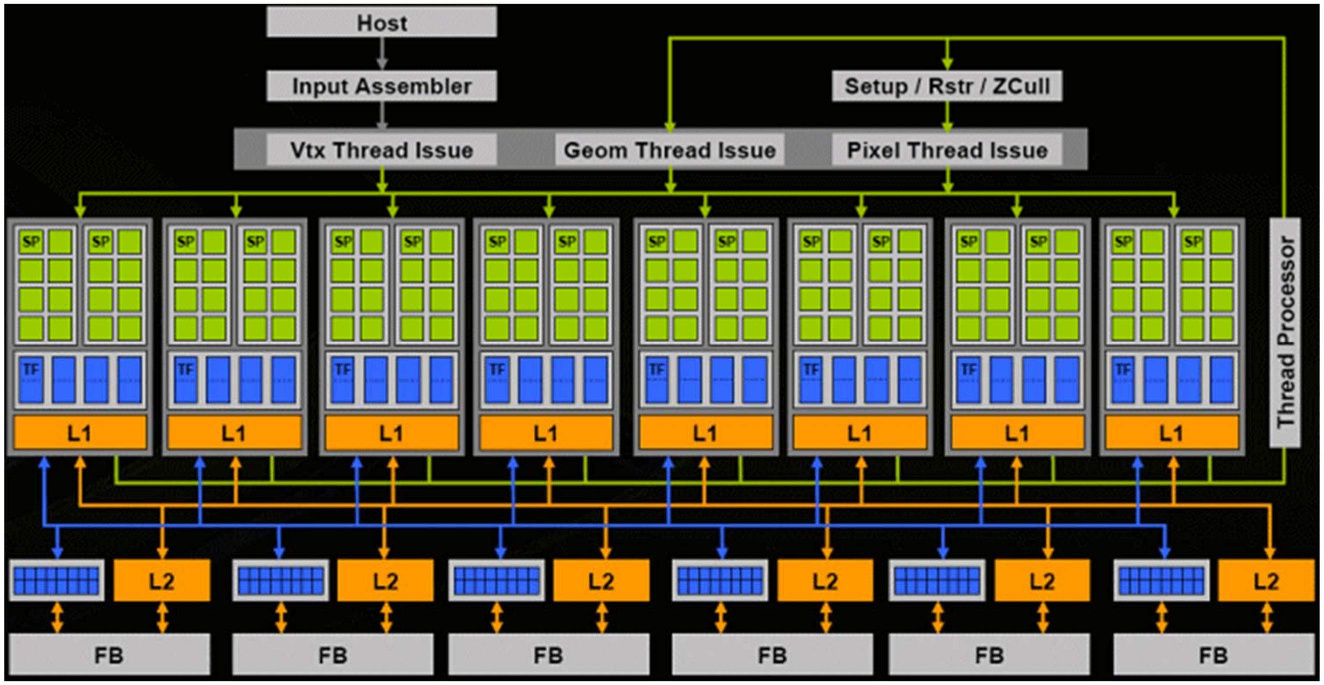
Modern GPU Hardware



Basic blocks in a modern GPU

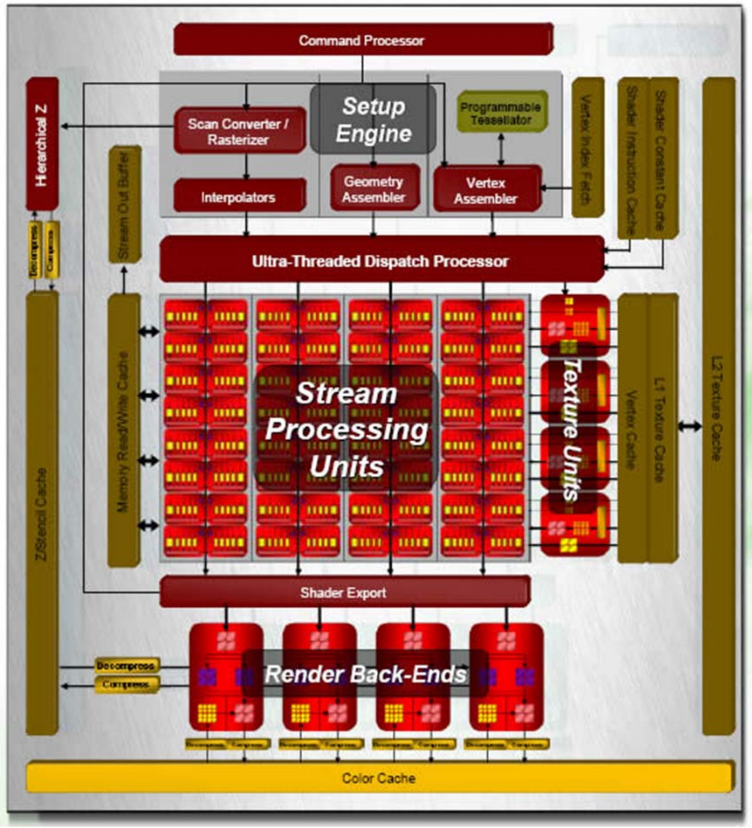


Unified Architecture example



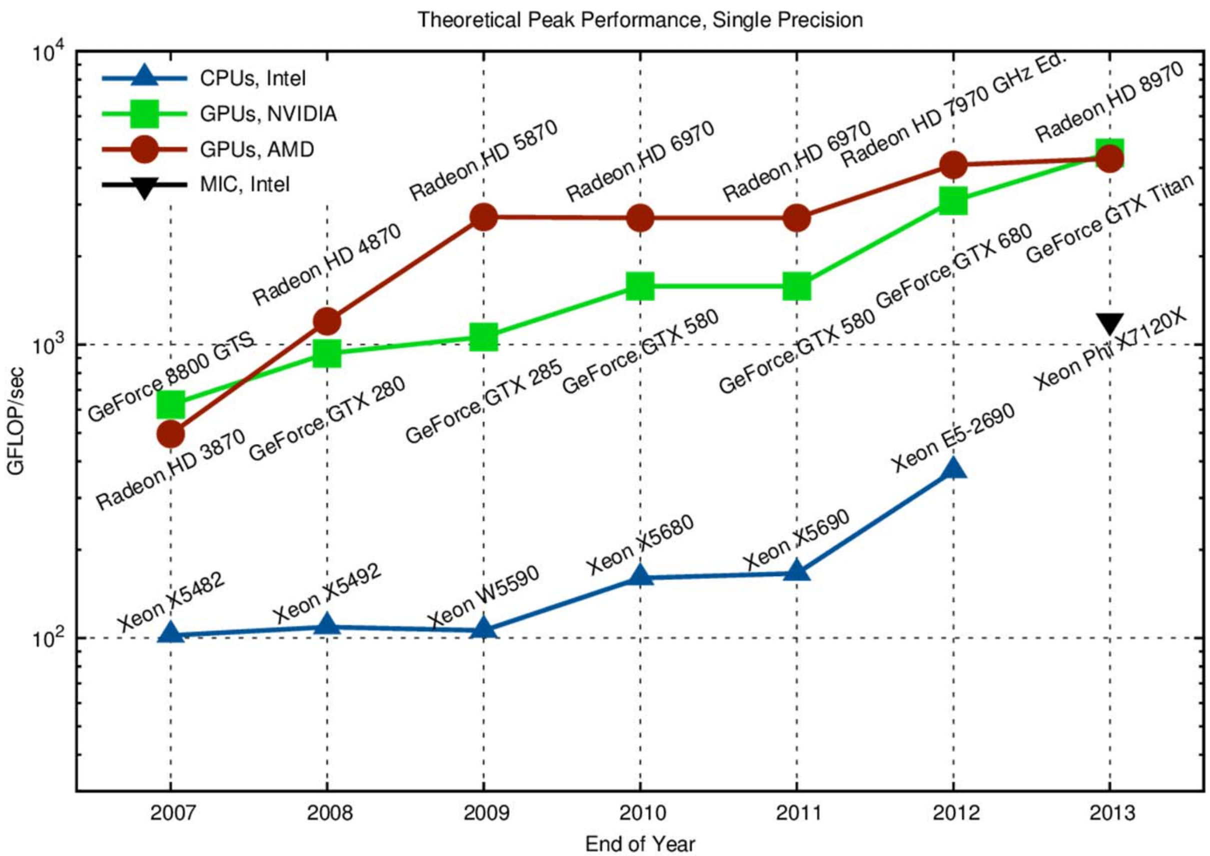
Nvidia Geforce 8 (2006)

Unified Architecture example

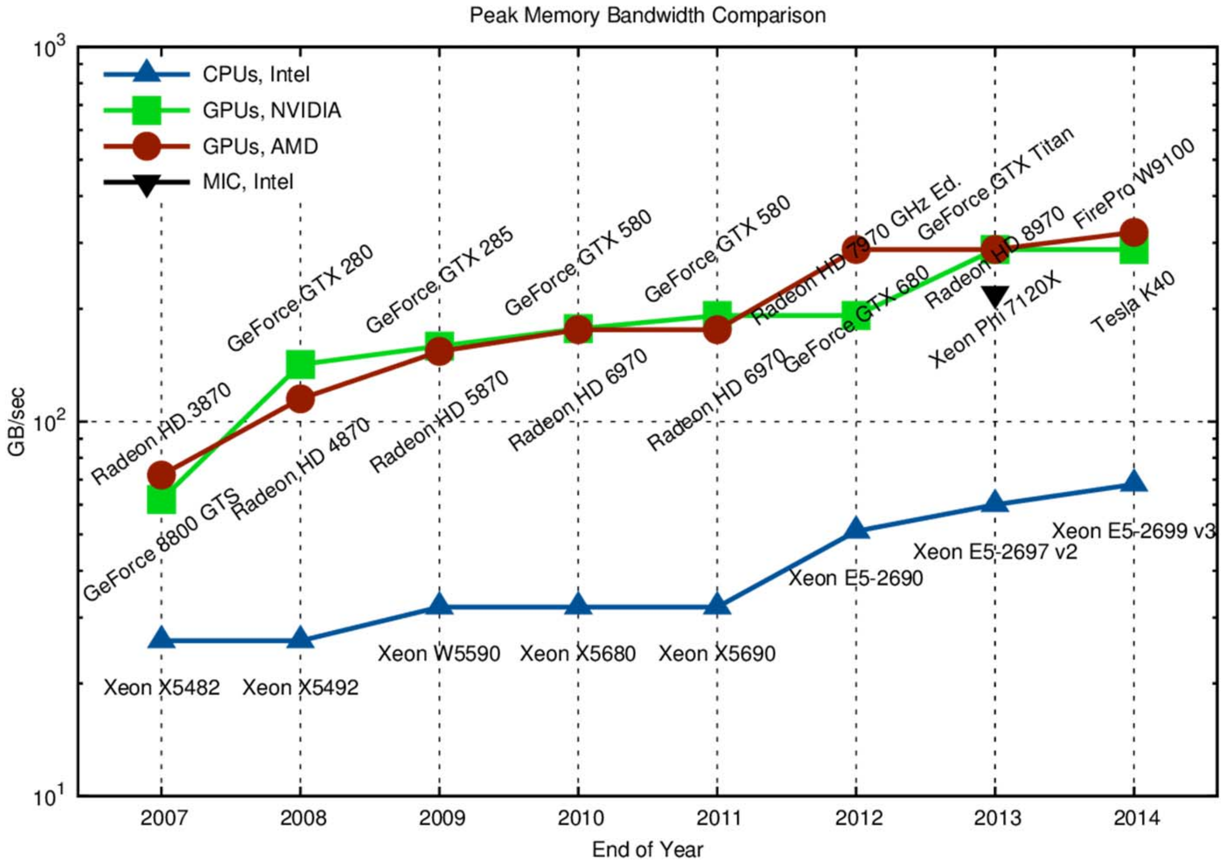


AMD Radeon R600 (2006)

GPU Compute Power: Recent History



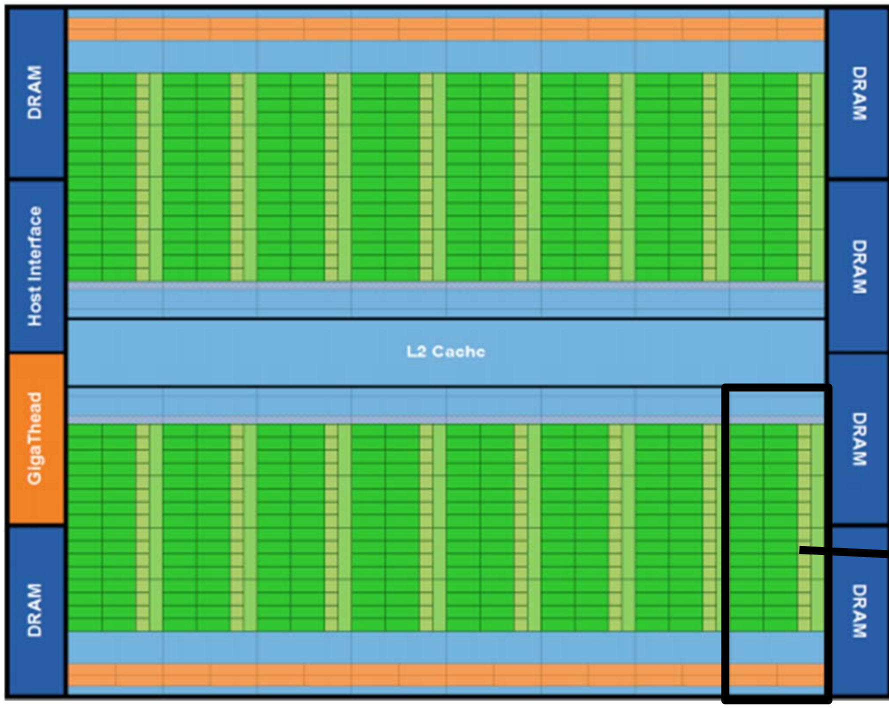
GPU Memory Bandwidth: Recent History

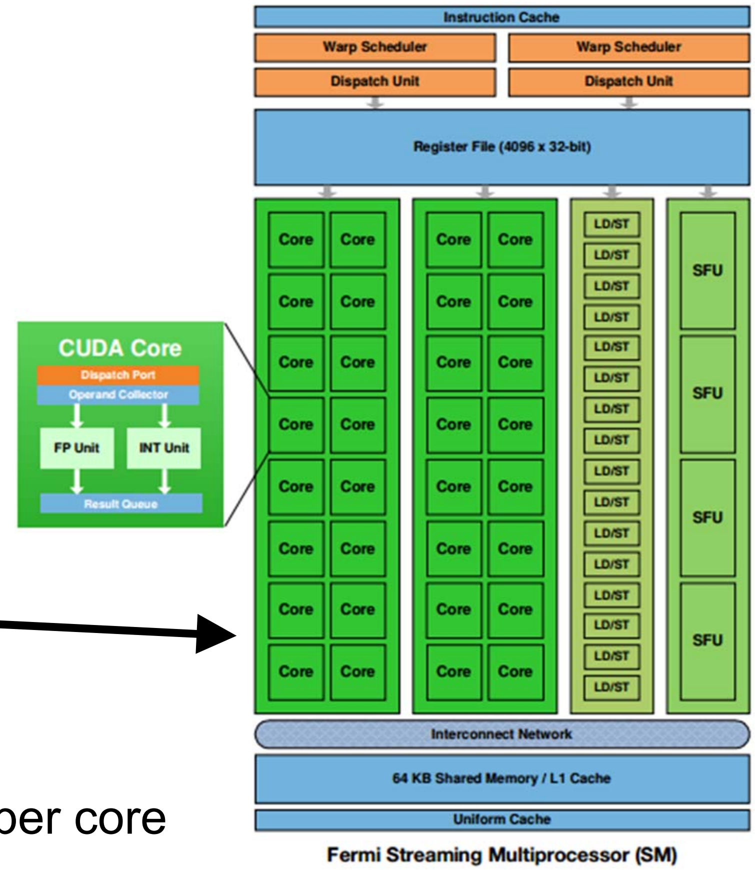


Current Development and Future

- GPU fixed-function units are being abstracted away
- Newest versions of CUDA and OpenGL include instructions for general-purpose computing
- Future GPUs will resemble multi-core CPUs with hyper-threading

Nvidia Fermi (2010)





- 16-cores with 32-way hyper-threading per core
- 1.5 TFLOPS (peak)

Sources

D. Hower. (2013, May 21). GPU Architectures: A CPU Perspective. [Online]. Available: http://courses.cs.washington.edu/courses/cse471/13sp/lectures/GPUsStudents.pdf

C. M. Wittenbrink, E. Kilgariff, and A. Prabhu. (2011, April). *IEEE Micro*. [Online]. 31(2), pp. 50 - 59. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5751939

D. Luebke and G. Humphreys, "How GPUs Work," *IEEE Computer*. [Online]. vol. 40, no. 2, pp. 96-100, Feb. 2007. Available: http://www.cs.virginia.edu/~gfx/papers/pdfs/59_HowThingsWork.pdf

B. Mederos, L. Velho, and L. H. de Figueiredo, "Moving least squares multiresolution surface approximation," *Proc. XVI Brazilian Symp. Computer Graphics and Image Processing*, [Online]. Oct. 2003, pp 19-26. Available: http://w3.impa.br/~boris/mederosb_moving.pdf

K. Hagen. (2014, July 23). Introduction to Real-Time Rendering. [Online]. Available: http://www.slideshare.net/korayhagen/introduction-to-real-time-rendering

G. Turk. (2000, Aug.). "The Stanford Bunny," [Online]. Available: http://www.cc.gatech.edu/~turk/bunny/bunny.html.old01

"Drawing Polygons," *OpenGL*. [Online]. Available: https://open.gl/drawing

K. Fatahalian. (2011). How a GPU Works. [Online]. Available: http://www.cs.cmu.edu/afs/cs/academic/class/15462-f11/www/lec_slides/lec19.pdf

D. Luebke. (2007). GPU Architecture: Implications & Trends. [Online]. Available: http://s08.idav.ucdavis.edu/luebke-nvidia-gpu-architecture.pdf

Sources (continued)

M. Houston and A. Lefohn. (2011). GPU architecture II: Scheduling the graphics pipeline. [Online]. Available: https://courses.cs.washington.edu/courses/cse558/11wi/lectures/08-GPU-architecture-II_BPS-2011.pdf

J. Ragan-Kelley. (2010, July 29). Keeping Many Cores Busy: Scheduling the Graphics Pipeline. [Online]. Available: http://bps10.idav.ucdavis.edu/talks/09-raganKelley_SchedulingRenderingPipeline_BPS_SIGGRAPH2010.pdf

B. C. Johnstone, "Bandwidth Requirements of GPU Architectures," M. S. thesis, Dept. Comp. Eng., Rochester Institute of Technology, Rochester, NY, 2014.

J. D. Owens, M. Houston, D. Luebke, and S. Green, (2008). "GPU Computing," *Proc. IEEE*, vol. 96, no. 5, pp. 879-899, [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4490127&tag=1

T. Dalling. (2014, Feb. 24). "Explaining Homogeneous Coordinates & Projective Geometry," [Online]. Available: http://www.tomdalling.com/blog/modern-opengl/explaining-homogenous-coordinates-and-projective-geometry/

NVIDIA, (2009). "Whitepaper: NVIDIA's next generation CUDA compute architecture: Fermi," [Online]. Available: http://www.nvidia.com/content/pdf/fermi_white_papers/nvidiafermicomputearchitecturewhitepaper.pdf

C. McClanahan. (2010). History and Evolution of GPU Architecture: A Paper Survey. [Online]. Available: http://mcclanahoochie.com/blog/wp-content/uploads/2011/03/gpu-hist-paper.pdf

J. Bikker. (2014). Graphics: Universiteit Utrecht - Information and Computing Sciences. [Online]. Available: http://www.cs.uu.nl/docs/vakken/gr/2015/index.html

Sources (continued)

K. Rupp. (2014, June 21). “CPU, GPU and MIC Hardware Characteristics over Time,” [Online]. Available: http://www.karlrupp.net/2013/06/cpu-gpu-and-mic-hardware-characteristics-over-time/

J. Lawrence. (2012, Oct. 22). 3D Polygon Rendering Pipeline. [Online]. Available: http://www.cs.virginia.edu/~gfx/Courses/2012/IntroGraphics/lectures/13-Pipeline.pdf

M. Christen. (2003). Tutorial 4: Varying Variables. [Online]. Available: http://www.clockworkcoders.com/oglsl/tutorial4.htm

T. S. Crow, “Evolution of the Graphical Processing Unit,” M. S. thesis, Dept. Comp. Science, University of Nevada, Reno, NV, 2004.

“Utah teapot,” *Wikipedia*. [Online]. Available: http://en.wikipedia.org/wiki/Utah_teapot

A. Rege. (2008). An Introduction to Modern GPU Architecture. [Online]. Available: http://http.download.nvidia.com/developer/cuda/seminar/TDCI_Arch.pdf

HD2000 - The First GPU’s under the AMD Name. (2007, May 14) [Online]. Available: http://www.bjorn3d.com/2007/05/hd2000-the-first-gpus-under-the-amd-name-2/

E. Kelgariff and R. Fernando. (2005). “Chapter 30. The GeForce 6 Series GPU Architecture,” *GPU Gems 2*. [Online]. Available: http://http.developer.nvidia.com/GPUGems2/gpugems2_chapter30.html

P. N. Glaskowsky. (2009, Sept.). NVIDIA’s Fermi: The First Complete GPU Computing Architecture. [Online]. Available: http://www.nvidia.com/content/PDF/fermi_white_papers/P.Glaskowsky_NVIDIA%27s_Fermi-The_First_Complete_GPU_Architecture.pdf