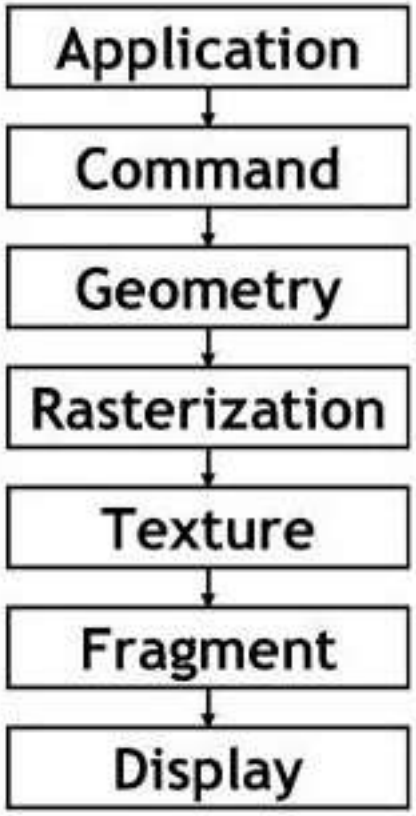
Graphics Processing Unit Architecture (GPU Arch)

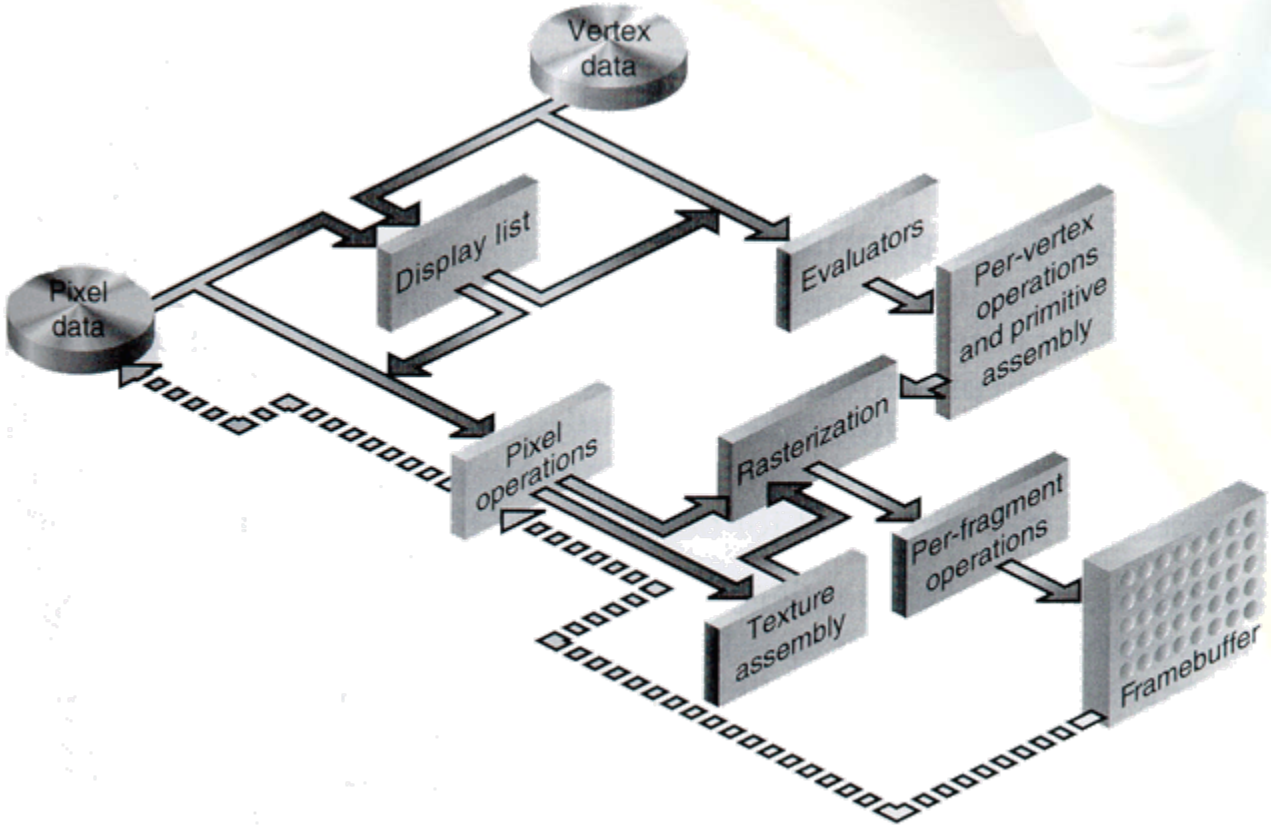
With a focus on NVIDIA GeForce 6800 GPU

What is a GPU

- From Wikipedia : A specialized processor efficient at manipulating and displaying computer graphics
- 2D primitive support – bit block transfers
- Some might have video support
- And of course 3D support (a topic at the heart of this presentation)
- GPUs are optimized for raster graphics

The Graphics pipeline





Modern graphics pipeline (left) (ref: http://graphics.stanford.edu/courses/cs448a-01-fall/lectures/lecture2/walk010.html)

OpenGL 3D pipeline (right) (ref: http://www.vorlesungen.uos.de/informatik/ifc99-00/opengl/images/pipeline.gif)

3D graphics software interfaces
OpenGL (v2.0 as of now)

- Low level
- Specification not an API
- Crossplatform implementations
- Popular with some games
- A simple seq of opengl instr (in C)

```
glClearColor(0.0,0.0,0.0,0.0);
glClear(GL_COLOR_BUFFER_BIT);
glColor3f(1.0,1.0,1.0);
glOrtho(0.0,1.0,0.0,1.0,-1.0,1.0);
glBegin(GL_POLYGON);
        glVertex(0.25,0.25,0.0);
        glVertex(0.75,0.25,0.0);
        glVertex(0.75,0.75,0.0);
        glVertex(0.25,0.75,0.0);
glEnd();
```

3D graphics software interfaces
Direct 3D (v9.0c as of now)

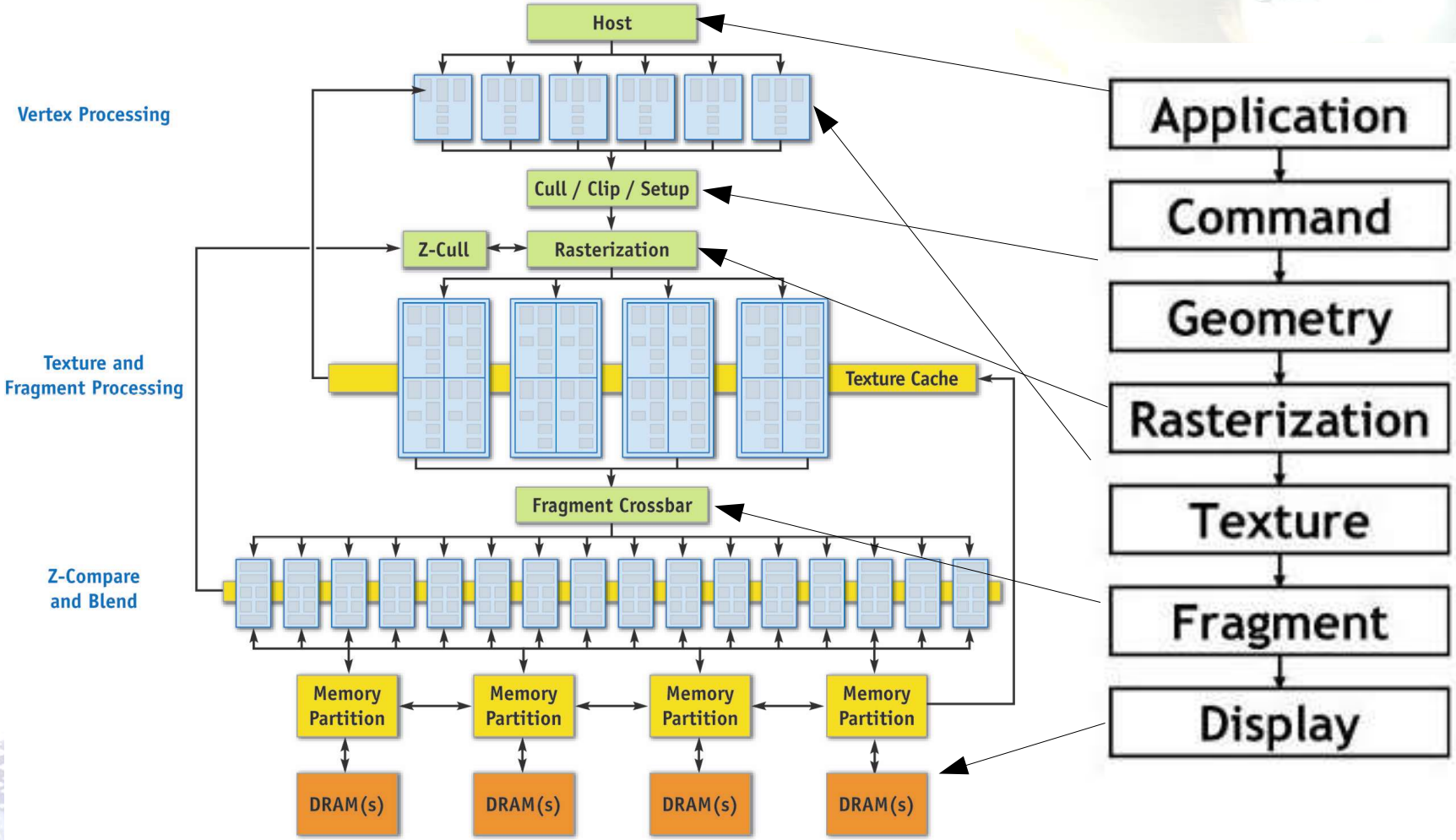
- High level
- 3D API – part of DirectX
- Very popular in the gaming industry
- Microsoft platforms only

NVIDIA GeForce 6800
General info

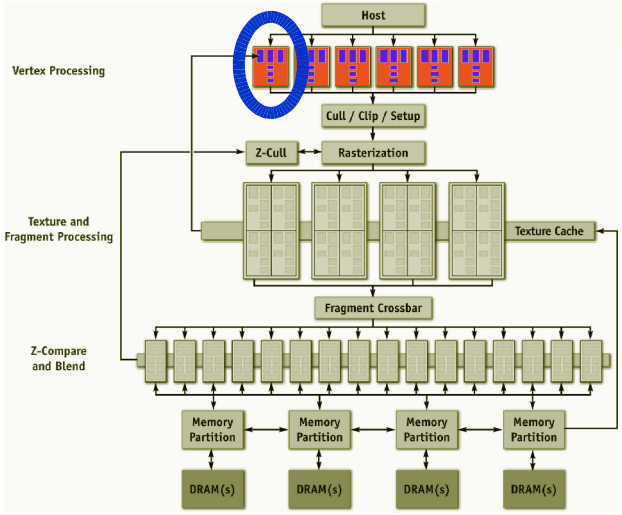
- Impressive performance stats
  - 600 Million vertices/s
  - 6.4 billion texels/s
  - 12.8 billion pixels/s rendering z/stencil only
  - 64 pixels per clock cycle early z-cull (reject rate)
- Riva series (1st DirectX compatible)
  - Riva 128, Riva TNT, Riva TNT2
- GeForce Series
  - GeForce 256, GeForce 3 (DirectX 8), GeForce FX, GeForce 6 series

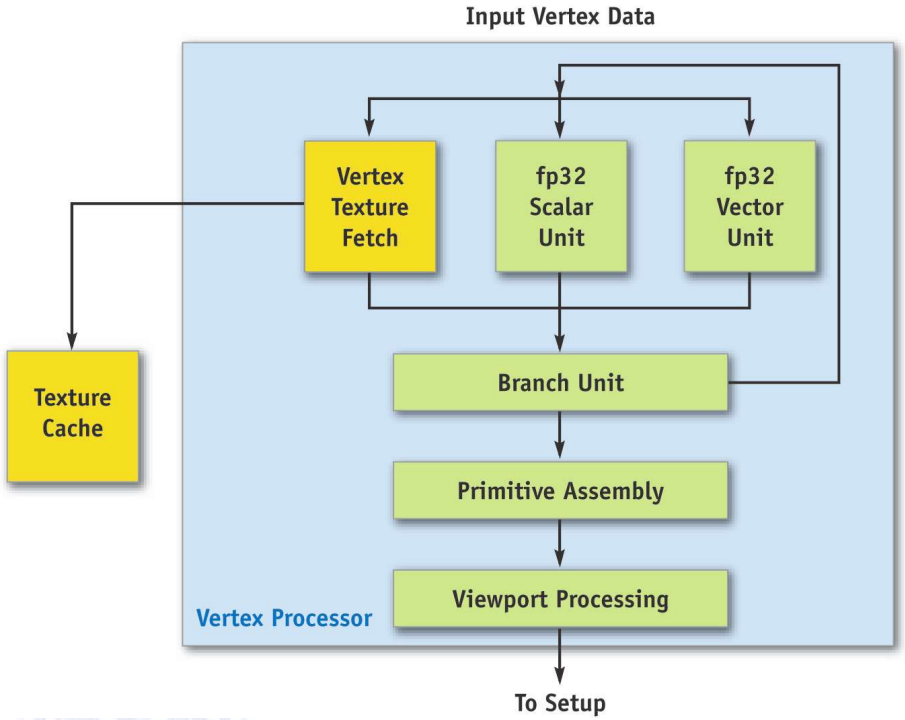
NVIDIA GeForce 6800

Block Diagram



NVIDIA GeForce 6800



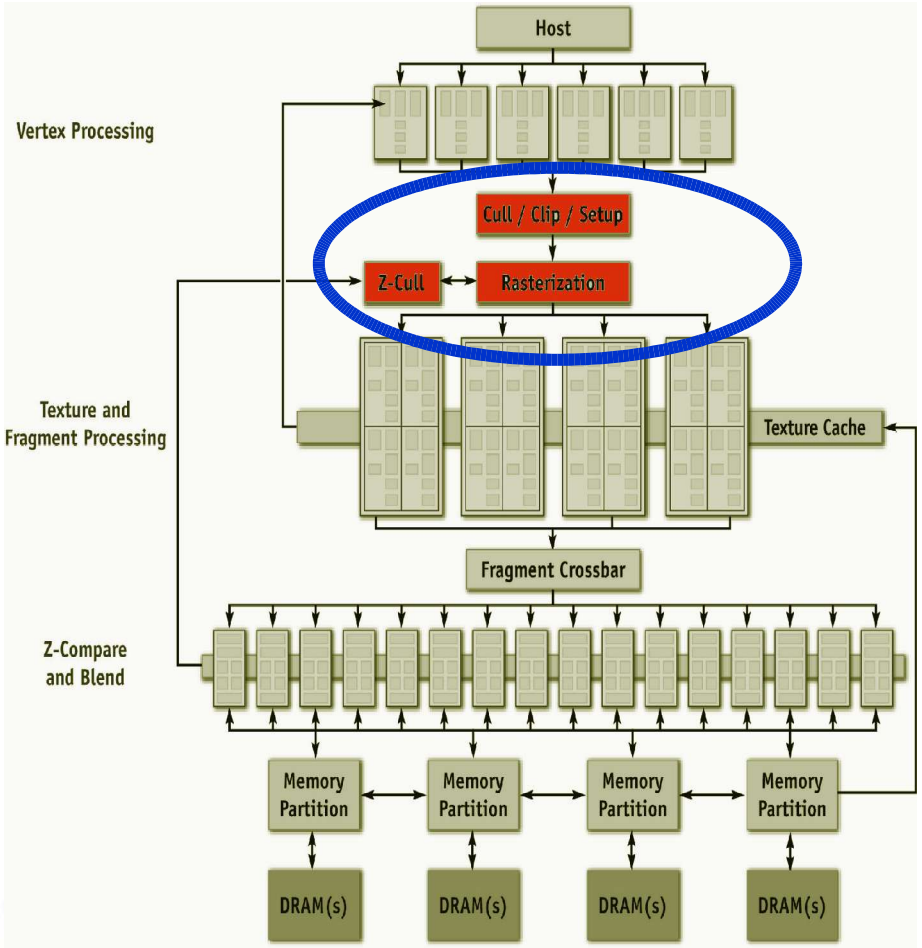


Vertex Processor (or vertex shader)

- Allow shader to be applied to each vertex
- Transformation and other per vertex ops
- Allow vertex shader to fetch texture data (6 series only)

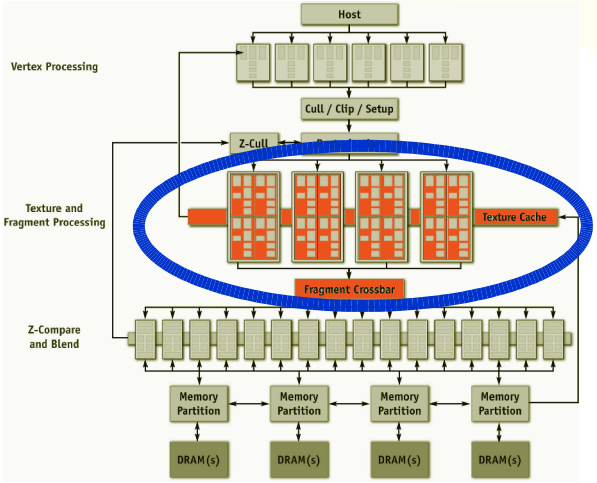
NVIDIA GeForce 6800

Clipping, Z Culling and Rasterization



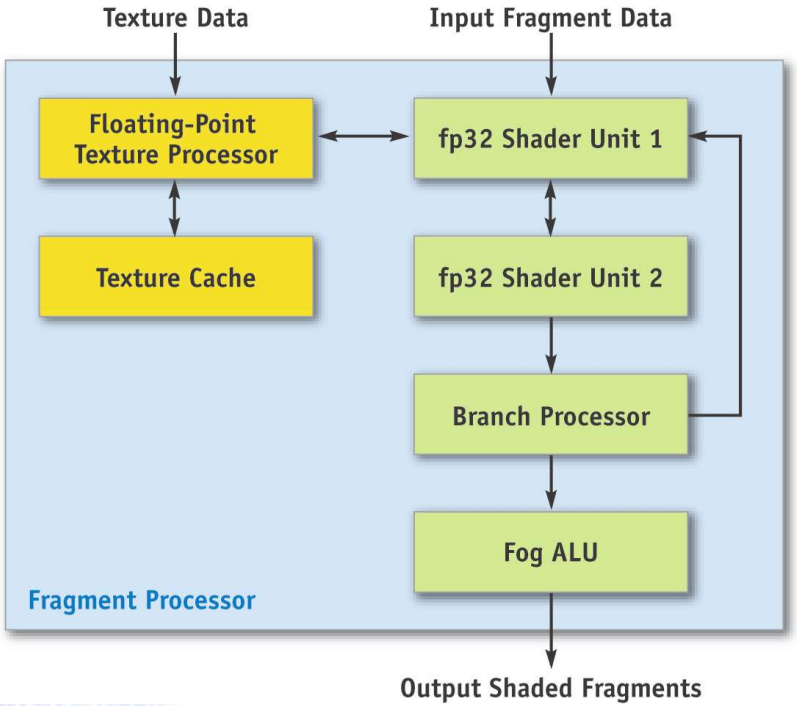
- Cull/clip – per primitive operation and data preparation for rasterization
- Rasterization: primitive to pixel mapping
- Z culling : quick pixel elimination based on depth

NVIDIA GeForce 6800

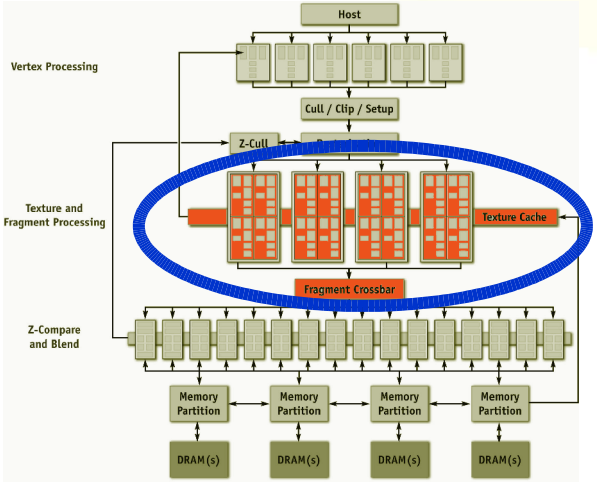


Fragment processor and Texel pipeline

- Fragment : a candidate pixel
- Varying number of pixel pipelines
- Operates on quads – for texture LOD
- SIMD processing hides texture fetch latency
- Texture caches

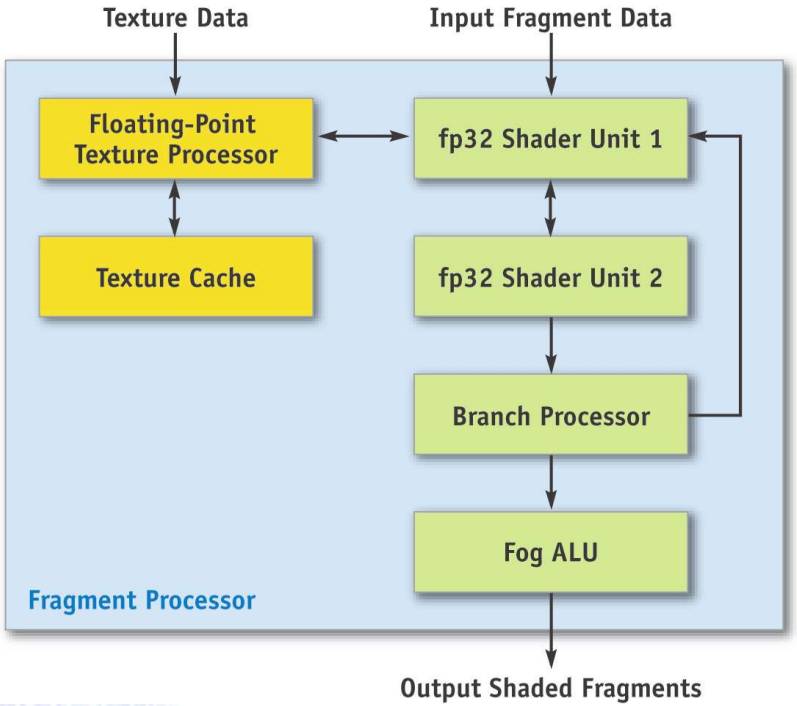


NVIDIA GeForce 6800



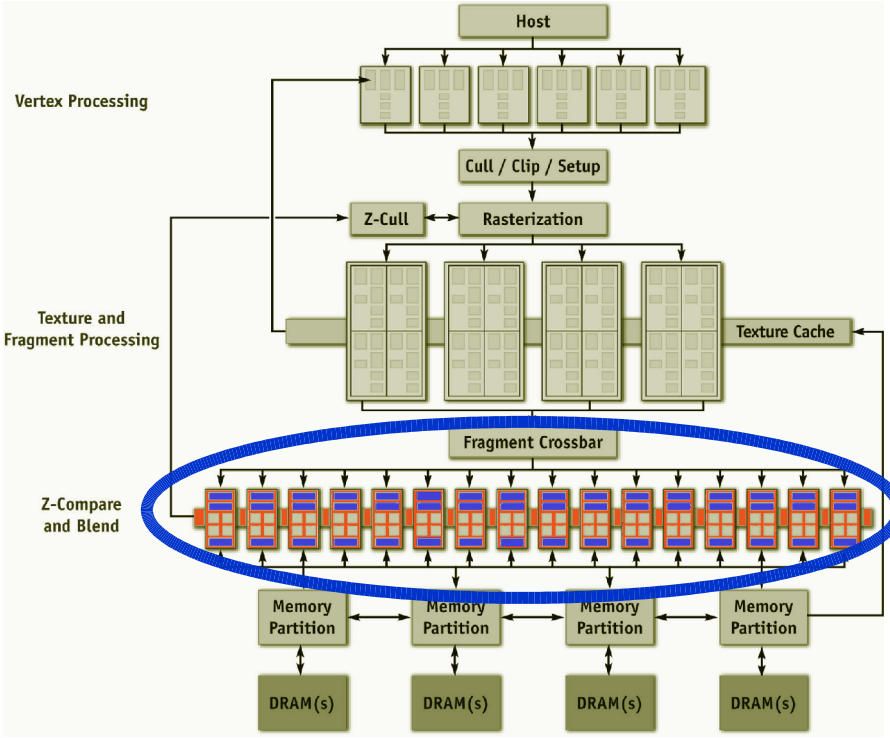
Fragment processor and Texel pipeline

- Texture unit can apply filters.
- Shader units can perform 8 math ops (w/o texture load) or 4 math ops (with texture load) in a clock
- Fog calculation done in the end
- Pixels almost ready for framebuffer



NVIDIA GeForce 6800

Z compare and blend



- Depth testing
- Stencil tests
- Alpha operations
- Render final color to target buffer

NVIDIA GeForce 6800
Features – Geometry Instancing

- Vertex stream frequency
  - hardware support for looping over a subset of vertices
- Example: rendering the same object multiple times at diff locations (grass, soldiers, people in stadium)

NVIDIA GeForce 6800
Features - continued

- Early culling and clipping;
  - cull nonvisible primitives at high rate
- Rasterization
  - supports Point Sprite, Aliased and anti-aliasing and triangles, etc
- Z-Cull
  - Allows high-speed removal of hidden surfaces
- Occlusion Query
  - Keeps a record of the number of fragments passing or failing the depth test and reports it to the CPU

NVIDIA GeForce 6800
Features Continued

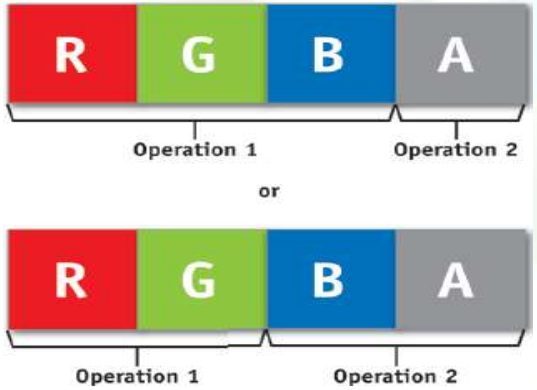
- Texturing
  - Extended support for non power of two textures to match support for power of two textures - Mipmapping, Wrapping and clamping, Cube map and 3D textures.
- Shadow Buffer Support
  - Fetches shadow buffer as a projective texture and performs z-compares of the shadow buffer data to distance from light.

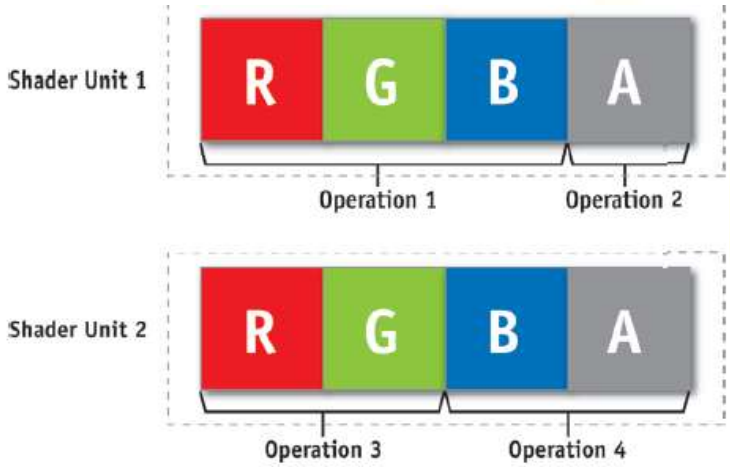
NVIDIA GeForce 6800
Features – Shader Support

- Increased instruction count (upto 65535 instructions.)
- Fragment processor; multiple render targets.
- Dynamic flow control branching
- Vertex texturing
- More temporary registers.

NVIDIA GeForce 6800
Features – Co-issue and Dual Issue

- Co-issue:
  - Each four-component-wide vector unit is capable of executing two independent instructions in parallel
  - More scalar computations done in less time.
- Dual issue:
  - two independent instructions can be executed on different parts of the shader pipeline
  - Makes scheduling easy and more efficient.





GPGPU

- Look at GPU as a fast SIMD processor
- It is a specialized processor, so not all programs can be run
- Example computational programs – FFT, Cryptography, Ray Tracing, Segmentation and even sound processing!

GPU from comp arch perspective
Processing units

- Focus on Floating point math
- fp32 and fp16 precision support for intermediate calculations
- 6 four-wide fp32 vector MADs/clock in shaders and 1 scalar multifunction op
- 16 four-wide fp32 vector MADs/clock in frag-proc plus 16 four-wide fp32 MULs
- Dedicated fp16 normalization hardware

GPU from comp arch perspective
Memory

- Use dedicated but standard memory architectures (eg DRAM)
- Multiple small independent memory partitions for improved latency
- Memory used to store buffers and optionally textures
- In low-end system (Intel 855GM) system memory is shared as the Graphics memory

GPU from comp arch perspective
System Interface

- GPU interfaces with the CPU using fast buses like AGP and PCI Express
- Port speeds
  - PCI express upto 8GB/sec (4 + 4)
    Practically upto (3.2 + 3.2)
  - AGP upto 2 GB/sec (for 8x AGP)
- Such bus speeds are important because textures and vertex data needs to come from CPU to GPU (after that it's the internal GPU bandwidth that matters)

GPU from comp arch perspective
Caches

- Texture caches (2 level)
  - Shared between vertex procs and fragment procs
  - Cache processed/filtered textures
- Vertex caches
  - cache processed and unprocessed vertexes
  - improve computation and fetch performance
- Z and buffer cache and write queues

Demo

- http://download.nvidia.com/downloads/nZone/videos/nvidia/nalu.wmv

References

- Nvidia 6800 chapter from GPU Gems 2
  http://download.nvidia.com/developer/GPU_Gems_2/GPU_Gems2_ch30.pdf
- OpenGL design
  http://graphics.stanford.edu/courses/cs448a-01-fall/design_opengl.pdf
- OpenGL programming guide (ISBN: 0201604582)
- Real time graphics architectures lecture notes
  http://graphics.stanford.edu/courses/cs448a-01-fall/
- GeForce 256 overview
  http://www.nvnews.net/reviews/geforce_256/gpu_overview.shtml
- NVIDIA website
  http://nvidia.com

So long and thanks for all the fish

(Oh yeah ... any questions?)