Technical Brief

NVIDIA GeForce 8800 GPU Architecture Overview

World's First Unified DirectX 10 GPU Delivering Unparalleled Performance and Image Quality

November 2006
TB-02787-001_v01

Table of Contents

List of Figures

List of Tables

Preface

Welcome to our technical brief describing the NVIDIA® GeForce® 8800 GPU architecture.

We have structured the material so that the initial few pages discuss key GeForce 8800 architectural features, present important DirectX 10 capabilities, and describe how GeForce 8 Series GPUs and DirectX 10 work together. If you read no further, you will have a basic understanding of how GeForce 8800 GPUs enable dramatically enhanced 3D game features, performance, and visual realism.

In the next section we go much deeper, beginning with operations of the classic GPU pipeline, followed by showing how GeForce 8800 GPU architecture radically changes the way GPU pipelines operate. We describe important new design features of GeForce 8800 architecture as it applies to both the GeForce 8800 GTX and the GeForce 8800 GTS GPUs. Throughout the document, all specific GPU design and performance characteristics are related to the GeForce 8800 GTX.

Next we'll look a little closer at the new DirectX 10 pipeline, including a presentation of key DirectX 10 features and Shader Model 4.0. Refer to the NVIDIA technical brief titled *Microsoft DirectX 10: The Next-Generation Graphics API* (TP-02820-001) for a detailed discussion of DirectX 10 features.

We hope you find this information informative.

GeForce 8800 Architecture Overview

Based on the revolutionary new NVIDIA® GeForce® 8800 architecture, NVIDIA's powerful GeForce 8800 GTX graphics processing unit (GPU) is the industry's first fully unified architecture-based DirectX 10–compatible GPU that delivers incredible 3D graphics performance and image quality. Gamers will experience amazing Extreme High Definition (XHD) game performance with quality settings turned to maximum, especially with NVIDIA SLI® configurations using high-end NVIDIA nForce® 600i SLI motherboards.

Unified, Massively Parallel Shader Design

The GeForce 8800 GTX GPU implements a massively parallel, unified shader design consisting of 128 individual stream processors running at 1.35 GHz. Each processor is capable of being dynamically allocated to vertex, pixel, geometry, or physics operations for the utmost efficiency in GPU resource allocation and maximum flexibility in load balancing shader programs. Efficient power utilization and management delivers industry-leading performance per watt and performance per square millimeter.

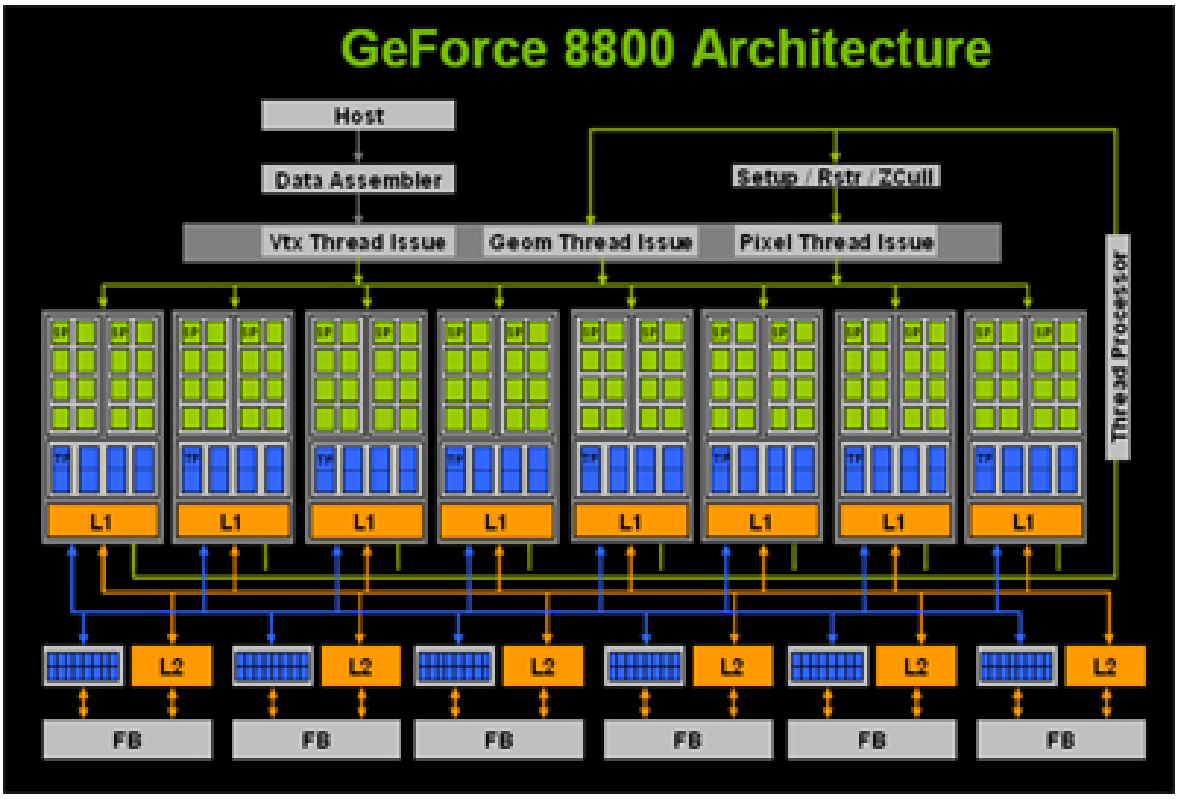


Figure 1. GeForce 8800 GTX block diagram

Don't worry—we'll describe all the gory details of Figure 1 very shortly! Compared to the GeForce 7900 GTX, a single GeForce 8800 GTX GPU delivers 2× the performance on current applications, with up to 11× scaling measured in certain shader operations. As future games become more shader intensive, we expect the GeForce 8800 GTX to surpass DirectX 9–compatible GPU architectures in performance.

In general, shader-intensive and high dynamic-range (HDR)–intensive applications shine on GeForce 8800 architecture GPUs. Teraflops of raw floating-point processing power are combined to deliver unmatched gaming performance, graphics realism, and real-time, film-quality effects.

The groundbreaking NVIDIA® GigaThread™ technology implemented in GeForce 8 Series GPUs supports thousands of independent, simultaneously executing threads, maximizing GPU utilization.

The GeForce 8800 GPU's unified shader architecture is built for extreme 3D graphics performance, industry-leading image quality, and full compatibility with DirectX 10. Not only do GeForce 8800 GPUs provide amazing DirectX 10 gaming experiences, but they also deliver the fastest and best quality DirectX 9 and OpenGL gaming experience today. (Note that Microsoft Windows Vista is required to utilize DirectX 10).

We'll briefly discuss DirectX 10 features supported by all GeForce 8800 GPUs, and then take a look at important new image quality enhancements built into every GeForce 8800 GPU. After describing other essential GeForce 8800 Series capabilities, we'll take a deep dive into the GeForce 8800 GPU architecture, followed by a closer look at the DirectX 10 pipeline and its features.

DirectX 10 Native Design

DirectX 10 represents the most significant step forward in 3D graphics APIs since the birth of programmable shaders. Completely built from the ground up, DirectX 10 features powerful geometry shaders, a new "Shader Model 4" programming model with substantially increased resources and improved performance, a highly optimized runtime, texture arrays, and numerous other features that unlock a whole new world of graphical effects (See "DirectX 10 Pipeline" later in this document).

GeForce 8 Series GPUs include all the required hardware functionality defined in Microsoft's Direct3D 10 (DirectX 10) specification and full support for the DirectX 10 unified shader instruction set and Shader Model 4 capabilities. The GeForce 8800 GTX is not only the first shipping DirectX 10 GPU, but it was also the reference GPU for DirectX 10 API development and certification. (For more details on DirectX 10, refer to *Microsoft DirectX 10: The Next-Generation Graphics API.*)

New features implemented in GeForce 8800 Series GPUs that work in concert with DirectX 10 features include geometry shader processing, stream output, improved instancing, and support for the DirectX 10 unified instruction set. GeForce 8 Series GPUs and DirectX 10 also provide the ability to reduce CPU overhead, shifting more graphics rendering load to the GPU.

Courtesy of Crytek

Figure 2. DirectX 10 game “Crysis” with both HDR lighting and antialiasing

DirectX 10 games running on GeForce 8800 GPUs deliver rich, realistic scenes; increased character detail; and more objects, vegetation, and shadow effects in addition to natural silhouettes and lifelike animations.

PC-based 3D graphics is raised to the next level with GeForce 8800 GPUs accelerating DirectX 10 games.

Lumenex Engine: Industry-Leading Image Quality

Image quality is significantly improved on GeForce 8800 GPUs with the NVIDIA Lumenex™ engine. Advanced new antialiasing technology provides up to 16× full-screen multisampled antialiasing quality at near 4× multisampled antialiasing performance using a single GPU.

High dynamic-range (HDR) lighting capability in all GeForce 8800 Series GPUs supports 128-bit precision (32-bit floating-point values per component), permitting true-to-life lighting and shadows. Dark objects can appear very dark—and bright objects can be very bright—with visible details present at both extremes, in addition to completely smooth gradients rendered in between.

HDR lighting effects can be used in concert with multisampled antialiasing on GeForce 8 Series GPUs. Plus, the addition of angle-independent anisotropic filtering, combined with considerable HDR shading horsepower, provides outstanding image quality. In fact, antialiasing can be used in conjunction with both FP16 (64-bit color) and FP32 (128-bit color) render targets.

The following image of model Adrianne Curry was rendered using a GeForce 8800 GTX GPU, and clearly illustrates the realistic effects made possible by the NVIDIA Lumenex engine.

(Image of model Adrianne Curry rendered on a GeForce 8800 GTX GPU)

Figure 3. NVIDIA Lumenex engine delivers incredible realism

An entirely new 10-bit display architecture works in concert with 10-bit DACs to deliver over a billion colors (compared to 16.7 million in the prior generation), permitting incredibly rich and vibrant photos and videos. With the next generation of 10-bit content and displays, the Lumenex engine will be able to display images of amazing depth and richness.

For more details on GeForce 8800 GPU image quality improvements, refer to *Lumenex Engine*: *The New Standard in GPU Image Quality* (TB-02824-001).

SLI Technology

NVIDIA's SLI technology is the industry's leading multi-GPU technology. It delivers up to 2× the performance of a single GPU configuration for unequaled gaming experiences by allowing two graphics cards to run in parallel on a single motherboard. The must-have feature for performance PCI Express graphics, SLI dramatically scales performance on today's hottest games. Running two GeForce 8800 GTX boards in an SLI configuration allows extremely high image-quality settings at extreme resolutions.

Figure 4. NVIDIA SLI technology

Quantum Effects GPU-Based Physics

NVIDIA Quantum Effects™ technology enables more physics effects to be simulated and rendered on the GPU. Specifically, GeForce 8800 GPU stream processors excel at physics computations, and up to 128 processors deliver a staggering floating-point computational ability that results in amazing performance and visual effects. Games can implement much more realistic smoke, fire, and explosions. Also, lifelike movement of hair, fur, and water can be completely simulated and rendered by the graphics processor. The CPU is freed up to run the game engine and artificial intelligence (AI), thus improving overall gameplay. Expect to see far more physics simulations in DirectX 10 games running on GeForce 8800 GPUs.

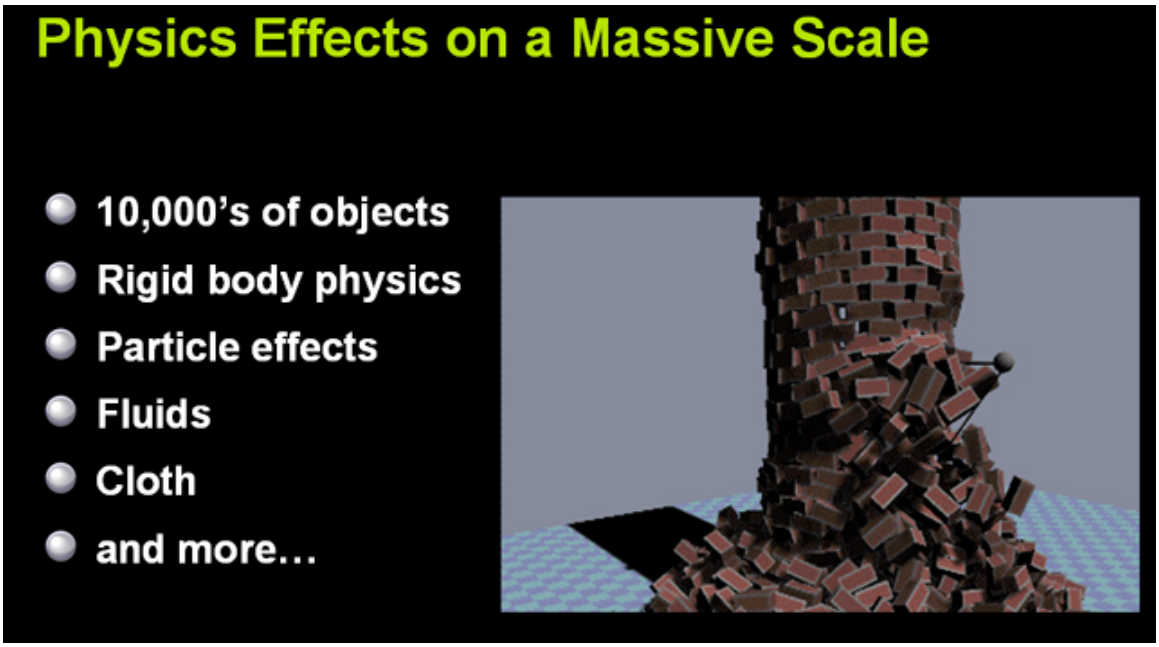


Figure 5. Quantum Effects

PureVideo and PureVideo HD

The NVIDIA PureVideo™ HD capability is built into every GeForce 8800 Series GPU and enables the ultimate HD DVD and Blu-ray viewing experience with superb picture quality, ultra-smooth movie playback, and low CPU utilization. High-precision subpixel processing enables videos to be scaled with great precision, allowing low-resolution videos to be accurately mapped to high-resolution displays.

PureVideo HD is comprised of dedicated GPU-based video processing hardware (SIMD vector processor, motion estimation engine, and HD video decoder), software drivers, and software-based players that accelerate decoding and enhance image quality of high-definition video in H.264, VC-1, WMV/WMV-HD, and MPEG-2 HD formats.

PureVideo HD can deliver 720p, 1080i, and 1080p high-definition output and support for both 3:2 and 2:2 pull-down (inverse telecine) of HD interlaced content. PureVideo HD on GeForce 8800 GPUs now provides HD noise reduction and HD edge enhancement.

PureVideo HD adjusts to any display and uses advanced techniques (found only on high-end consumer players and TVs) to make standard and high-definition video look crisp, smooth, and vibrant, regardless of whether videos are watched on an LCD, plasma, or other progressive display type.

AACS-protected Blu-ray or HD DVD movies can be played on systems with GeForce 8800 GPUs using AACS-compliant movie players from CyberLink, InterVideo, and Nero that utilize GeForce 8800 GPU PureVideo features.

All GeForce 8800 GPUs are HDCP-capable, meeting the security specifications of the Blu-ray Disc and HD DVD formats and allowing the playback of encrypted movie content on PCs when connected to HDCP-compliant displays.

GeForce 8800 Series GPUs also readily handle standard definition PureVideo formats such as WMV and MPEG-2 for high-quality playback of computer-generated video content and standard DVDs. In the popular industry-standard HQV Benchmark (www.hqv.com), which evaluates standard definition video de-interlacing, motion correction, noise reduction, film cadence detection, and detail enhancement, *all GeForce 8800 GPUs achieve an unrivaled 128 points out of 130 points!*

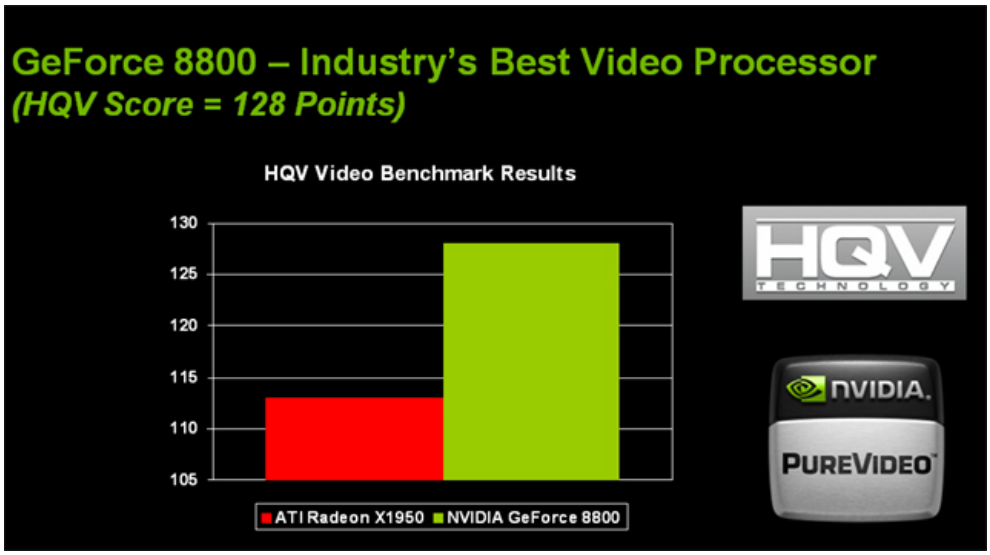


Figure 6. HQV benchmark results for GeForce 8800 GPUs

PureVideo and PureVideo HD are programmable technologies that can adapt to new video formats as they are developed, providing a future-proof video solution.

Figure 7. PureVideo vs. the competition

GeForce 8800 GPUs support various TV-out interfaces such as composite, S-video, component, and DVI. HD resolutions up to 1080p are supported depending on connection type and TV capability.

Extreme High Definition Gaming (XHD)

All GeForce 8800–based GPUs are designed for Extreme High Definition gaming (XHD), where games can be played at high widescreen resolutions up to 2560×1600. XHD has over seven times the picture clarity of native 1080i HD televisions and double the picture clarity of the 1080p HD format. XHD widescreen resolution allows users to see more of their PC games, enhance their video editing, and even add useful extra screen real estate to applications such as Google Earth. The dual-link DVI outputs on GeForce 8800 GTX boards enable XHD gaming up to 2560×1600 resolution with very playable frame rates. SLI configurations allow dialing up eye-candy to new levels of details never seen in the past, all with playable frame rates.

Figure 8. Extreme High Definition widescreen gaming

Built for Microsoft Windows Vista

GeForce 8800 GPU architecture is actually NVIDIA's fourth-generation GPU architecture built for Microsoft® Windows Vista™ technology, and gives users the best possible experience with the Windows Aero 3D graphical user interface and full DirectX 10 hardware support. GeForce 8800 GPUs support for Vista includes Windows Display Driver Model (WDDM), Vista's Desktop Windows Manager (DWM) composited desktop, the AERO interface using DX9 3D graphics, fast context switching, GPU resource virtualization support, and OpenGL Installable Client Driver (ICD) support (both older XP ICDs and newer Vista ICDs).

CUDA: Compute Unified Device Architecture

All GeForce 8800 GPUs include the revolutionary new NVIDIA CUDA™ built-in technology, which provides a unified hardware and software solution for data-intensive computing. Key highlights of CUDA technology are as follows:

- New "Thread Computing" processing model that takes advantage of massively threaded GeForce 8800 GPU architecture, delivering unmatched performance for data-intensive computations.
- Computing threads that can communicate and cooperate on the GPU.
- Standard C language interface for a simplified platform for complex computational problems.
- Architecture that complements traditional CPUs by providing additional processing capability for inherently parallel applications.
- Use of GPU resources in a different manner than graphics processing as seen in Figure 9, but both CUDA threads and graphics threads can run on the GPU concurrently if desired.

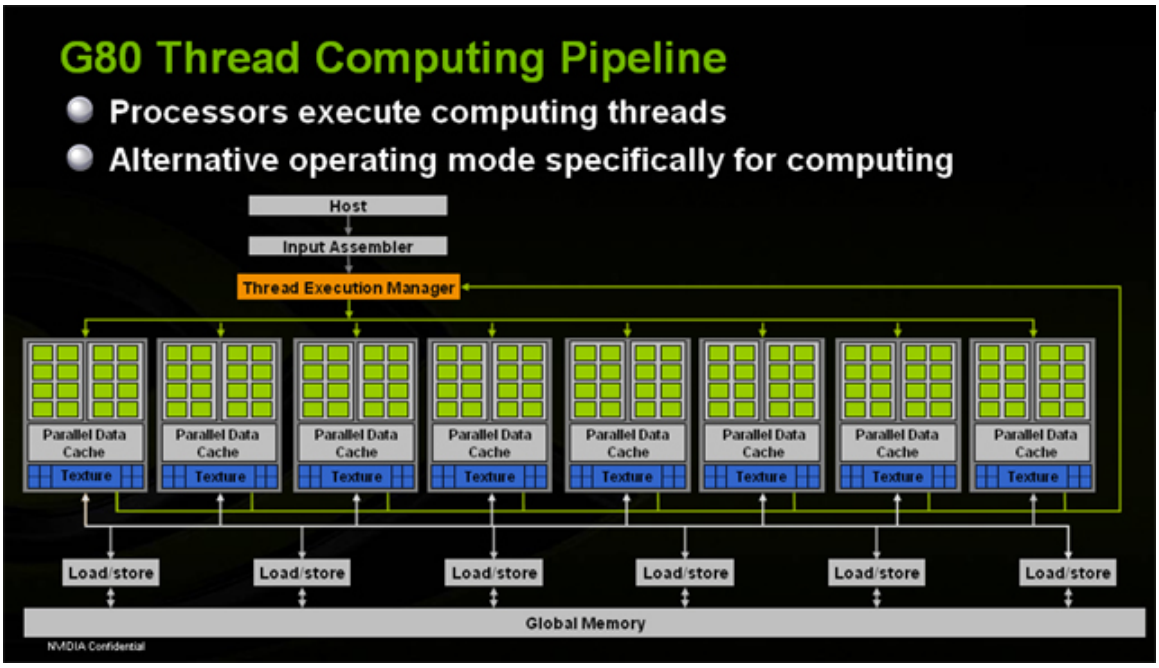


Figure 9. CUDA thread computing pipeline

CUDA enables new applications with a standard platform for extracting valuable information from vast quantities of raw data, and provides the following key benefits in this area:

- Enables high-density computing to be deployed on standard enterprise workstations and server environments for data-intensive applications.
- Divides complex computing tasks into smaller elements that are processed simultaneously in the GPU to enable real-time decision making.
- Provides a standard platform based on industry-leading NVIDIA hardware and software for a wide range of high data bandwidth, computationally intensive applications.
- Combines with multicore CPU systems to provide a flexible computing platform.
- Controls complex programs and coordinates inherently parallel computation on the GPU processed by thousands of computing threads.

CUDA's high-performance, scalable computing architecture solves complex parallel problems 100× faster than traditional CPU-based architectures:

- Up to of 128 parallel 1.35 GHz compute cores in GeForce 8800 GTX GPUs harness massive floating-point processing power, enabling maximum application performance.
- Thread computing scales across NVIDIA's complete line of next-generation GPUs—from embedded GPUs to high-performance GPUs that support hundreds of processors.

- NVIDIA SLI™ technology allows multiple GPUs to distribute computing to provide unparalleled compute density.
- Enables thread computing to be deployed in any industry-standard environment.
- Parallel Data Cache stores information on the GPU so threads can share data entirely within the GPU for dramatically increased performance and flexibility.

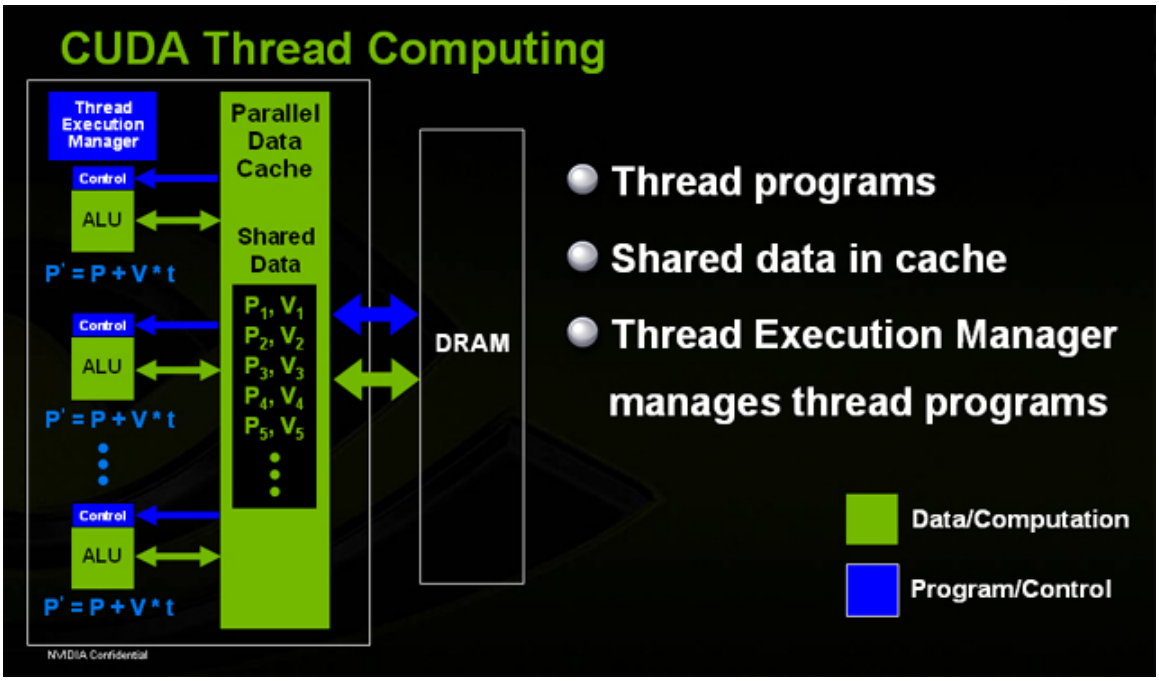


Figure 10. CUDA thread computing parallel data cache

- Thread Execution Manager efficiently schedules and coordinates the execution of thousands of computing threads for precise computational execution.

CUDA SDK unlocks the power of the GPU using industry-standard C language:

- Industry-standard C compiler simplifies software for complex computational problems.
- Complete development solution includes an industry-standard C compiler, standard math libraries, and a dedicated driver for thread computing on either Linux or Windows.
- Full support of hardware debugging and a profiler for program optimization.
- NVIDIA "assembly for computing" (NVasc) provides lower-level access to the GPU for computer language development and research applications.

The Four Pillars

Overall, the GeForce 8800 GPU Series is best defined by these four major categories:

- Outstanding performance with a unified shader design
  NVIDIA GigaThread technology and overall thread computing capability delivers the absolute best GPU performance for 3D games.
- DirectX 10 compatibility
  GeForce 8800 Series GPUs are the first shipping GPUs that support all DirectX 10 features.
- Significantly improved image quality
  NVIDIA Lumenex engine technology provides top-quality antialiasing, anisotropic filtering, and HDR capabilities enabling rich, lifelike detail in 3D games.
- High-performance GPU physics and GPU computing capability
  NVIDIA Quantum Effects permits billions of physics operations to be performed on the GPU, enabling amazing new effects and providing massive floating-point computing power for a variety of high-end calculation-intensive applications.

That's the high-level overview. Now it is time to go deep!

In the following sections, we first review classic GPU pipeline design and compare it to the new unified pipeline and shader architecture used by GeForce 8800 GPUs. We then discuss stream processors and scalar versus vector processor design, so you'll better understand GeForce 8800 GPU technology. Next, we'll present a high-level view of the GeForce 8800 GTX GPU architecture, followed by many of the new features that apply to all GeForce 8800 GPUs. All the while we'll provide specific references to GeForce 8800 GTX design and performance characteristics. The final section looks at important aspects of the DirectX 10 pipeline and programming model, and how they relate to the GeForce 8800 GPU architecture.

This page is blank.

The Classic GPU Pipeline... A Retrospective

Before we examine architectural details of a GeForce 8800 GPU, we should explain how the classic GPU internal pipeline operations have worked over the years.

When discussing GPU hardware architectures, dataflow, and pipeline operations, it's often good to start from the top, where data is fed from the CPU to the GPU, and work our way down through multiple processing stages until a pixel is finally drawn on the screen.

To date, GPUs have utilized traditional pipelined designs, as shown in Figure 11, which broadly depicts the GPU pipeline evolution through the DX9 generation.

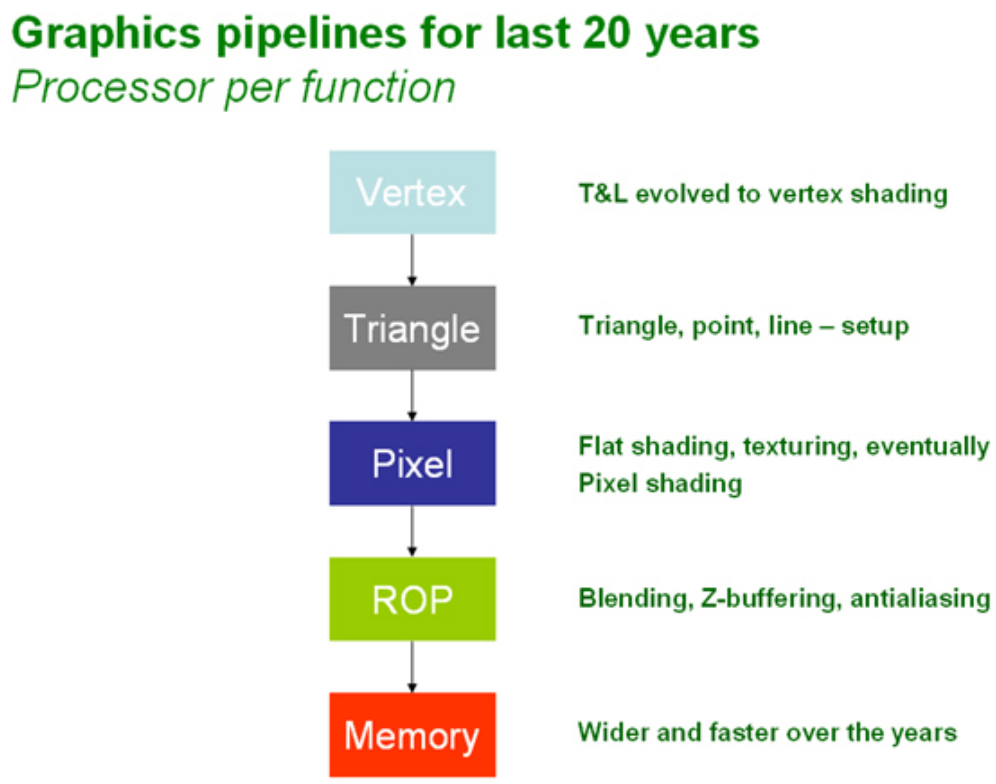


Figure 11. Classic GPU pipeline

After the GPU receives vertex data from the host CPU, the vertex stage is the first major stage. Back in the DirectX 7 timeframe, fixed-function transform and lighting hardware operated at this stage (such as with NVIDIA's GeForce 256 in 1999), and then programmable vertex shaders came along with DirectX 8. This was followed by programmable pixel shaders in DirectX 9 Shader Model 2, and dynamic flow control in DirectX 9 Shader Model 3. DirectX 10 expands programmability features much further, and shifts more graphics processing to the GPU, significantly reducing CPU overhead.

The next step in the classic pipeline is the setup, where vertices are assembled into primitives such as triangles, lines, or points. The primitives are then converted by the rasterization stage into pixel fragments (or just "fragments"), but are not considered full pixels at this stage. Fragments undergo many other operations such as shading, Z-testing, possible frame buffer blending, and antialiasing. Fragments are finally considered pixels when they have been written into the frame buffer.

As a point of confusion, the "pixel shader" stage should technically be called the "fragment shader" stage, but we'll stick with pixel shader as the more generally accepted term. In the past, the fragments may have only been flat shaded or have simple texture color values applied. Today, a GPU's programmable pixel shading capability permits numerous shading effects to be applied while working in concert with complex multitexturing methods.

Specifically, shaded fragments (with color and Z values) from the pixel stage are then sent to the ROP (Raster Operations in NVIDIA parlance). The ROP stage corresponds to the "Output Merger" stage of the DirectX 10 pipeline, where Z-buffer checking ensures only visible fragments are processed further, and visible fragments, if partially transparent, are blended with existing frame buffer pixels and antialiased. The final processed pixel is sent to the frame buffer memory for scanout and display to the monitor.

The classic GPU pipeline has basically included the same fundamental stages for the past 20 years, but with significant evolution over time. Many processing constraints and limitations exist with classic pipeline architectures, as did variations in DirectX implementations across GPUs from different vendors.

A few notable problems of pre-DirectX 10 classic pipelines include the following: limited reuse of data generated within the pipeline to be used as input to a subsequent processing step; high state change overhead; excessive variation in hardware capabilities (requiring different application code paths for different hardware); instruction set and data type limitations (such as lack of integer instructions and weakly defined floating point precision); inability to write results to memory in mid-pipeline and read them back into the top of the pipeline; and resource limitations (registers, textures, instructions per shader, render targets, and so on.)[1]

Let's proceed and see how the GeForce 8800 GPU architecture totally changes the way data is processed in a GPU with it unified pipeline and shader architecture.

---

[1] *The Direct3D® 10 System*, David Blythe, Microsoft Corporation.

GeForce 8800 Architecture in Detail

When NVIDIA's engineers started designing the GeForce 8800 GPU architecture in the summer of 2002, they set forth a number of important design goals. The top four goals were quite obvious:

- Significantly increase performance over current-generation GPUs.
- Notably improve image quality.
- Deliver powerful GPU physics and high-end floating-point computation ability.
- Provide new enhancements to the GPU pipeline (such as geometry shading and stream output), while working collaboratively with Microsoft to define features for the next major version of Direct X (DirectX 10 and Windows Vista).

In fact, many key GeForce 8800 architecture and implementation goals were specified in order to make GeForce 8800–class GPUs most efficient for DirectX 10 applications, while also providing the highest performance for existing applications using DirectX 9, OpenGL, and earlier DirectX versions.

The new GPU architecture would need to perform well on a variety of applications using different mixes of pixel, vertex, and geometry shading in addition to large amounts of high quality texturing.

The result was the GeForce 8800 GPU architecture that initially included two specific GPUs—the high-end GeForce 8800 GTX and the slightly downscaled GeForce 8800 GTS.

Figure 12 again presents the overall block diagram of the GeForce 8800 GTX for readers who would like to see the big picture up front.

But fear not, we'll start by describing the key elements of the GeForce 8800 architecture followed by looking at the GeForce 8800 GTX in more detail, where we will again display this "most excellent" diagram and discuss some of its specific features.

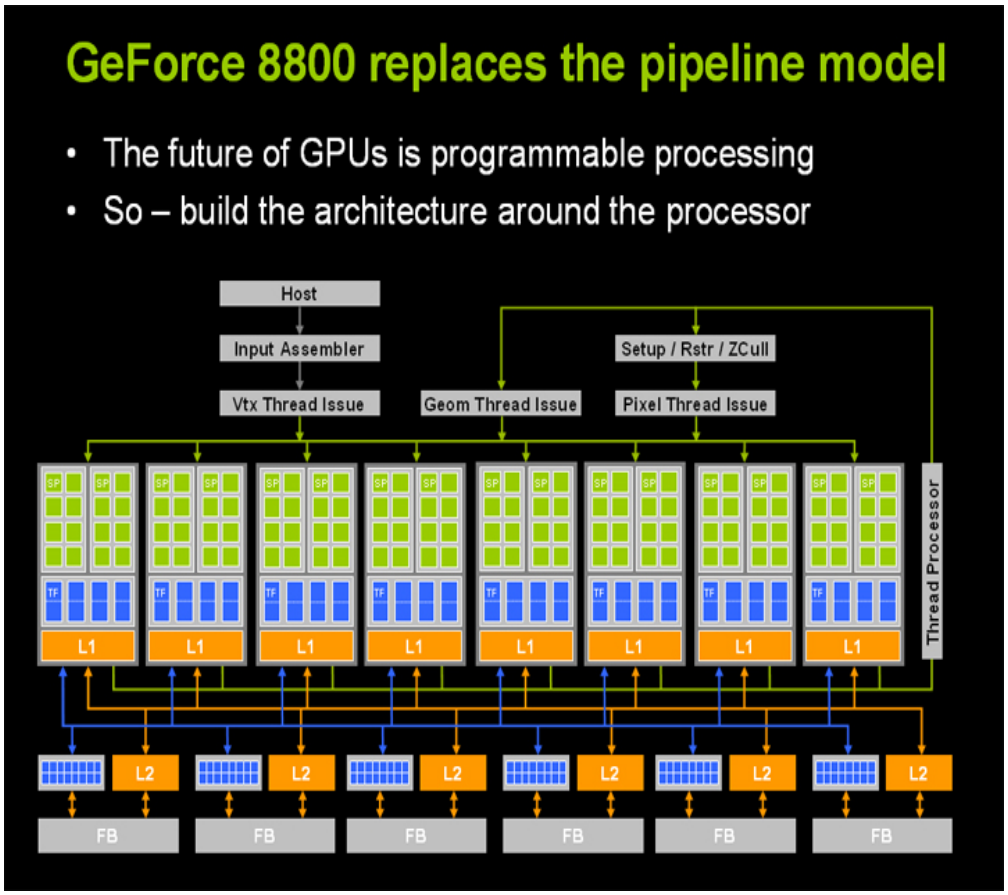


Figure 12. GeForce 8800 GTX block diagram

Unified Pipeline and Shader Design

Recall the classic pipeline model with its dataflow starting at the top, where vertices with various attributes, indices, commands, and textures are passed into the GPU from the CPU. Major processing stages follow in a fairly linear manner, including vertex shading, pixel shading, raster operations, and writing final pixels to the frame buffer. In fact, the GeForce 7 Series GPUs had many physical pipeline stages per major processing stage. Just the pixel shader stage of GeForce 7 GPUs had over 200 sequential pipeline stages!

With its unified pipeline and shader architecture, the GeForce 8800 GPU design significantly reduces the number of pipeline stages and changes the sequential flow to be more looping oriented. Inputs are fed to the top of the unified shader core, and outputs are written to registers and then fed back into the top of the shader core for the next operation.

In the generalized unified GPU diagram shown in Figure 13, the classic pipeline uses discrete shader types represented in different colors, where data flows sequentially down the pipeline through different shader types. The illustration on the right depicts a unified shader core with one or more standardized, unified shader processors.

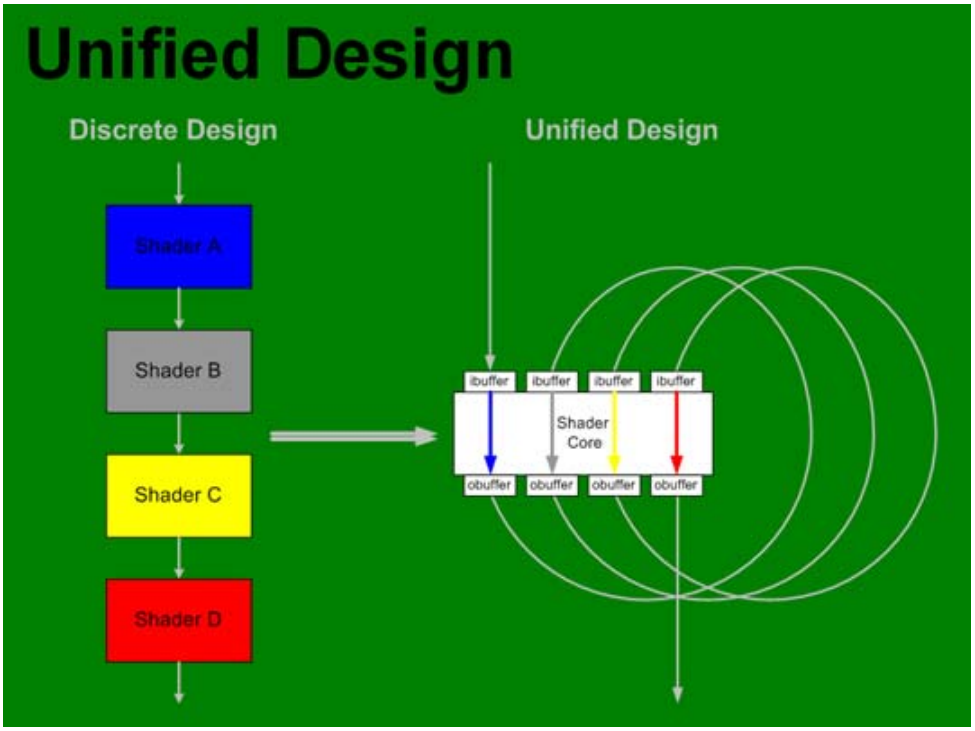


Figure 13. Classic vs. unified shader architecture

Data coming in the top left of the unified design (such as vertices), are dispatched to the shader core for processing, and results are sent back to the top of the shader core, where they are dispatched again, processed again, looped to the top, and so on until all shader operations are performed and the pixel fragment is passed on to the ROP subsystem.

Unified Shaders In-Depth

The GeForce 8800 design team realized that extreme amounts of hardware-based shading horsepower would be necessary for high-end DirectX 10 3D games. While DirectX 10 specifies a unified instruction set, it does not demand a unified GPU shader design, but NVIDIA GeForce 8800 engineers believed a unified GPU shader architecture made the most sense to allow effective DirectX 10 shader program load-balancing, efficient GPU power utilization, and significant improvement to the GPU architectural efficiency.

Note that the GeForce 8800 unified shaders can be also be used with DirectX 9, OpenGL, and older DirectX versions. No restrictions or fixed numbers of unified shading units need to be dedicated to pixel or vertex processing for any of the API programming models.

In general, numerous challenges had to be overcome with such a radical new design over the four-year GeForce 8800 GPU development timeframe.

Looking more closely at graphics programming, we can safely say that in general the number of pixels outnumbers vertices by a wide margin, which is why you saw a much larger number of pixel shader units versus vertex shader units in prior fixed-shader GPU architectures. But different applications do have different shader processing requirements at any given point in time—some scenes may be pixel-shader intensive and other scenes may be vertex-shader intensive. Figure 14 shows the variations in vertex and pixel processing over time in a particular application.

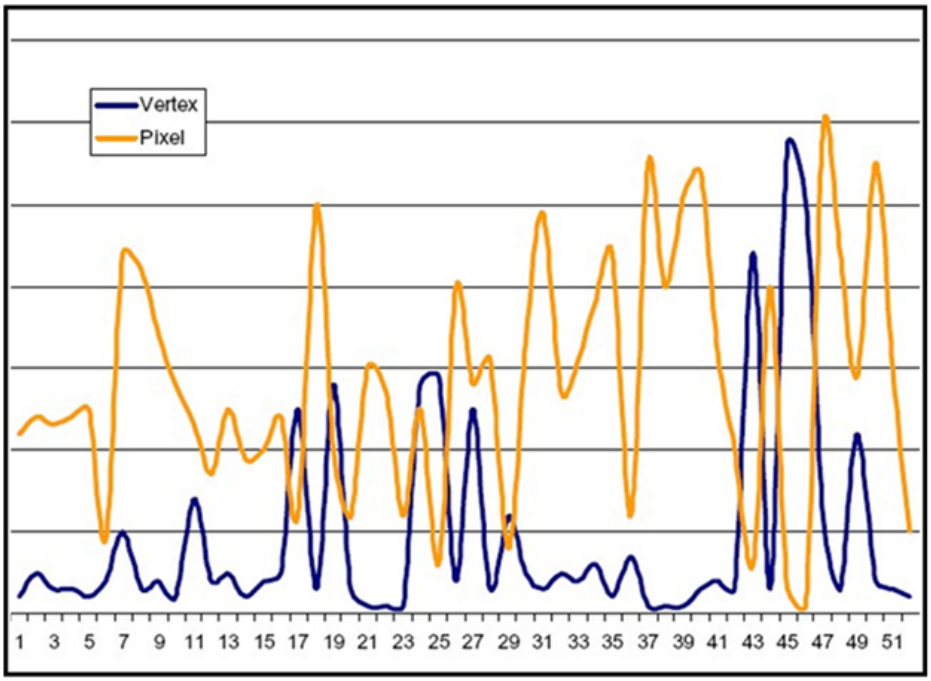


Figure 14. Characteristic pixel and vertex shader workload variation over time

In a GPU with a fixed number of specific types of shader units, restrictions are placed on operating efficiency, attainable performance, and application design.

Figure 15 shows a theoretical GPU with a fixed number of 4 vertex shader units and 8 pixel shader units, or a total of 12 shader units altogether.

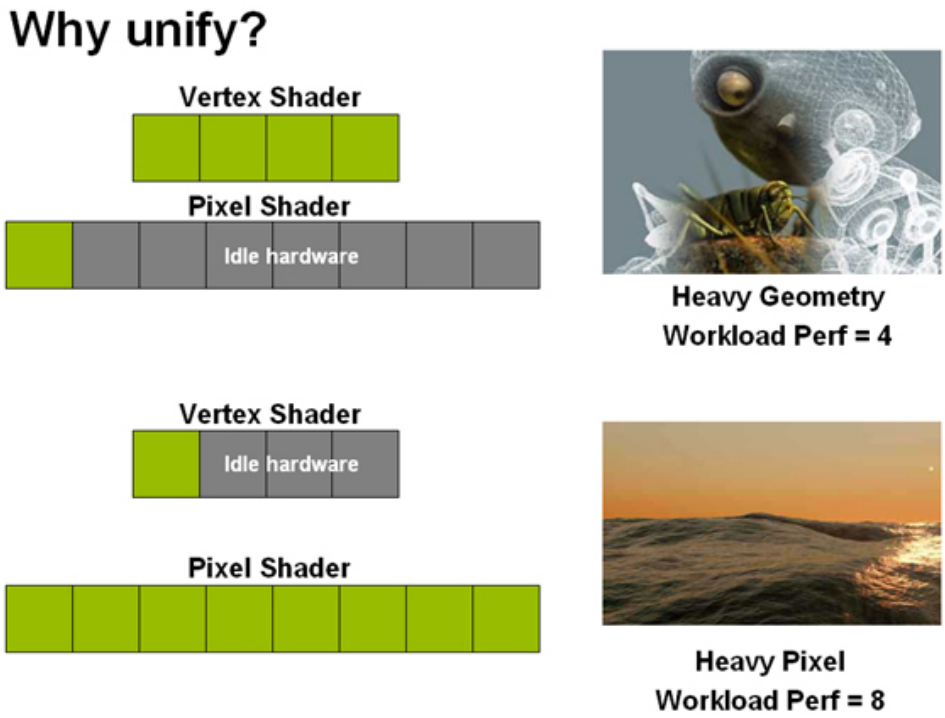


Figure 15. Fixed shader performance characteristics

In Figure 15, the top scenario shows a scene that is vertex shader-intensive, which can only attain performance as fast as the maximum number of vertex units, which in this case is "4." In the bottom scenario, the scene is pixel shader-intensive, which might be due to various complex lighting effects for the water. In this case, it is pixel-shader limited and can only attain a maximum performance of "8," equal to the number of pixel shader units, which is the bottleneck in this case. Both situations are not optimal because hardware is idle and performance is left on the table, so to speak. Also, it's not efficient from a power (performance/watt) or die size and cost (performance/sqmm) perspective.

In Figure 16, with a unified shader architecture, at any given moment when an application might be vertex-shader intensive, you can see the majority of unified shader processors are applied to processing vertex data, and in this case, the overall performance is increased to "11." Similarly, if its pixel shader heavy, the majority of unified shader units can be applied to pixel processing, also attaining a score of "11" in our example.

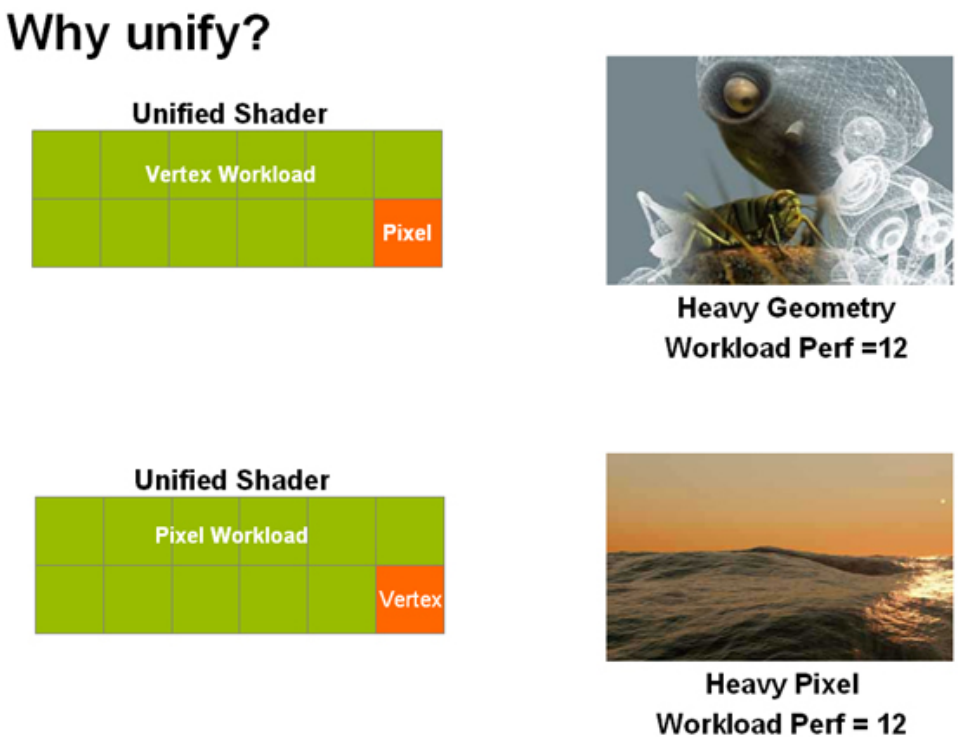


Figure 16. Unified shader performance characteristics

Unified stream processors (SPs) in GeForce 8800 GPUs can process vertices, pixels, geometry or physics—they are effectively general purpose floating-point processors. Different workloads can be *mapped* to the processors, as shown in Figure 17.

Note that geometry shading is a new feature of the DirectX 10 specification that we cover in detail later in the DirectX 10 section. The GeForce 8800 unified stream processors can process geometry shader programs, permitting a powerful new range of effects and features, while reducing dependence on the CPU for geometry processing.

The GPU dispatch and control logic can dynamically assign vertex, geometry, physics, or pixel operations to available SPs without worrying about fixed numbers of specific types of shader units. In fact, this feature is just as important to developers, who need not worry as much that certain aspects of their code might be too pixel-shader intensive or too vertex-shader intensive.

Not only does a unified shader design assist in load-balancing shader workloads, but it actually helps redefine how a graphics pipeline is organized. In the future, it is possible that other types of workloads can be run on a unified stream processor.

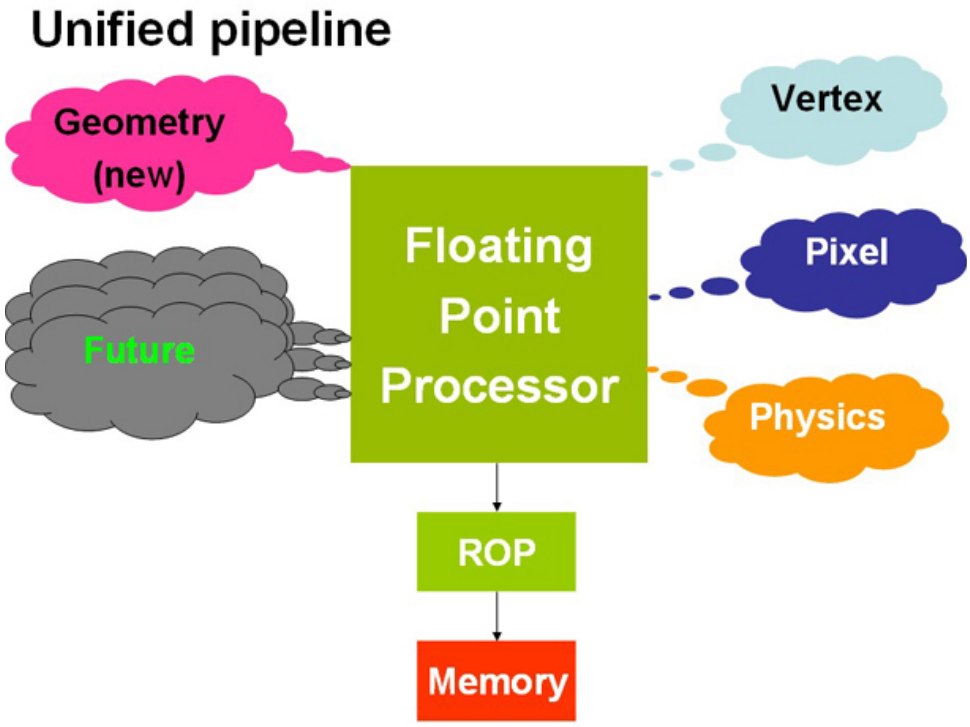


Figure 17. Conceptual unified shader execution framework

Stream Processing Architecture

Key to the GeForce 8800 architecture is the use of numerous scalar stream processors (SPs) to perform shader operations. Stream processors are highly efficient computing engines that perform calculations on an input stream, while producing an output stream that can be used by other stream processors. Stream processors can be grouped in close proximity, and in large numbers, to provide immense parallel processing power.

Generally, specialized high-speed instruction decode and execution logic is built into a stream processor, and similar operations are performed on the different elements of a data stream. On-chip memory is typically used to store output of a stream processor, and the memory can be quickly read as input by other stream processors for subsequent processing. SIMD (single instruction/multiple data) instructions can be implemented across groupings of stream processors in an efficient manner, and massively parallel stream processor clusters are well-suited for processing graphics data streams.

With the GeForce 8800 architecture, the image in Figure 18 shows a collection of stream processors (SPs) with associated numbers of Texture Filtering (TF), Texture Addressing (TA), and cache units to ensure a balanced design. The ratios of unit types shown below in a subset slice of a typical GeForce 8800 GPU are maintained when scaling up to 128 SPs specifically in a GeForce 8800 GTX GPU.

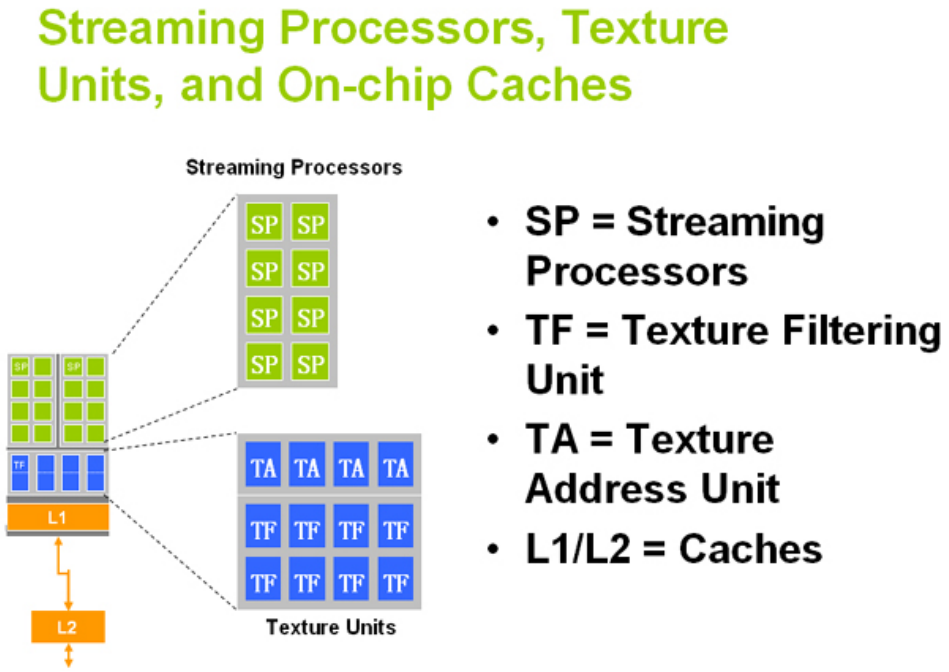


Figure 18. Streaming processors and texture units

Each GeForce 8800 GPU stream processor is fully generalized, fully decoupled, scalar (see "Scalar Processor Design Improves GPU Efficiency"), can dual-issue a MAD and a MUL, and supports IEEE 754 floating-point precision.

The stream processors are a critical component of NVIDIA GigaThread technology, where thousands of threads can be in flight within a GeForce 8800 GPU at any given instant. GigaThread technology keeps SPs fully utilized by scheduling and dispatching various types of threads (such as pixel, vertex, geometry, and physics) for execution.

All stream processors are driven by a high-speed clock domain that is separate from the core clock that drives the rest of the chip. For example, the GeForce 8800 GTX core clock is 575 MHz and its stream processors run at 1.35 GHz delivering exceptionally high shader performance.

Scalar Processor Design Improves GPU Efficiency

Leading GPUs to date have used vector processing units because many operations in graphics occur with vector data (such as R-G-B-A components operating in pixel shaders or 4×4 matrices for geometry transforms in vertex shaders). However, many scalar operations also occur. During the early GeForce 8800 architecture design phases, NVIDIA engineers analyzed hundreds of shader programs that showed an increasing use of scalar computations. They realized that with a mix of vector and scalar instructions, especially evident in longer, more complex shaders, it's hard to efficiently utilize all processing units at any given instant with a vector architecture. Scalar computations are difficult to compile and schedule efficiently on a vector pipeline.

NVIDIA and ATI vector-based GPUs have used shader hardware that permits dual instruction issue. Recent ATI GPUs use a "3+1" design, allowing a single issue of a four-element vector instruction, or a dual issue of a three-element vector instruction plus a scalar instruction. NVIDIA GeForce 6x and GeForce 7x GPUs are more efficient with 3+1 *and* 2+2 dual-issue design. But they are still not as efficient as a GeForce 8800 GPU scalar design, which can issue scalar operations to its scalar processors with 100 percent shader processor efficiency. NVIDIA engineers have estimated as much as 2× performance improvement can be realized from a scalar architecture that uses 128 scalar processors versus one that uses 32 four-component vector processors, based on architectural efficiency of the scalar design. (Note that vector-based shader program code is converted to scalar operations inside a GeForce 8800 GPU to ensure complete efficiency.)

Lumenex Engine: High-Quality Antialiasing, HDR, and Anisotropic Filtering

NVIDIA's Lumenex engine in GeForce 8800 GPUs implements entirely new and very high-quality antialiasing (AA) and anisotropic filtering (AF) technologies. The new antialiasing technology uses "coverage samples" and color and Z samples, and is Coverage Sampling Antialiasing (CSAA).

CSAA technology provides much higher levels of antialiasing than possible with present-day GPUs, with performance characteristics similar to 4× or 8× multisampled AA. New CSAA modes include 8×, 16×, and 16×Q. Each CSAA mode enhances built-in application antialiasing modes with much higher-quality antialiasing. The fourth new mode, 8×Q, is standard 8× multisampling.

Each new AA mode can be enabled from the NVIDIA driver control panel and requires the user to select an option called "Enhance the Application Setting." Users must first turn on *any* antialiasing level within the game's control panel for the new AA modes to work, since they need the game to properly allocate and enable antialiased rendering surfaces. If a game does not natively support antialiasing, a user can select an NVIDIA driver control panel option called "Override Any Applications Setting," which allows any control panel AA settings to be used with the game. Note that this setting does not always work with all applications. For

games with AA capability built in, the "Enhancing the Application Setting" is an easy way to improve overall image quality.

In many games, the new 16× high quality mode will yield frame-per-second performance results similar to standard 4× multisampled mode, but with much improved image quality. In certain cases, such as the edge of stencil shadow volumes, the new antialiasing modes will not be enabled and those portions of the scene will fall back to 4× multisampled mode.

Below is a close-up image of how the new CSAA 16× mode compares to standard 4× multisampled AA mode.

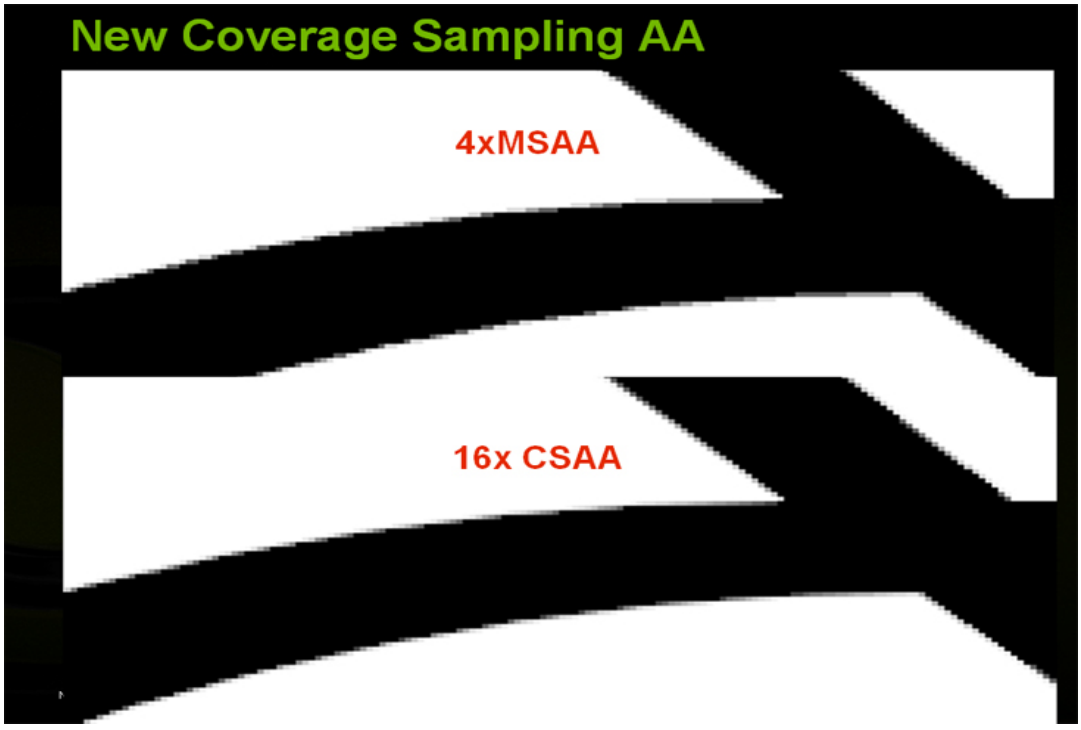
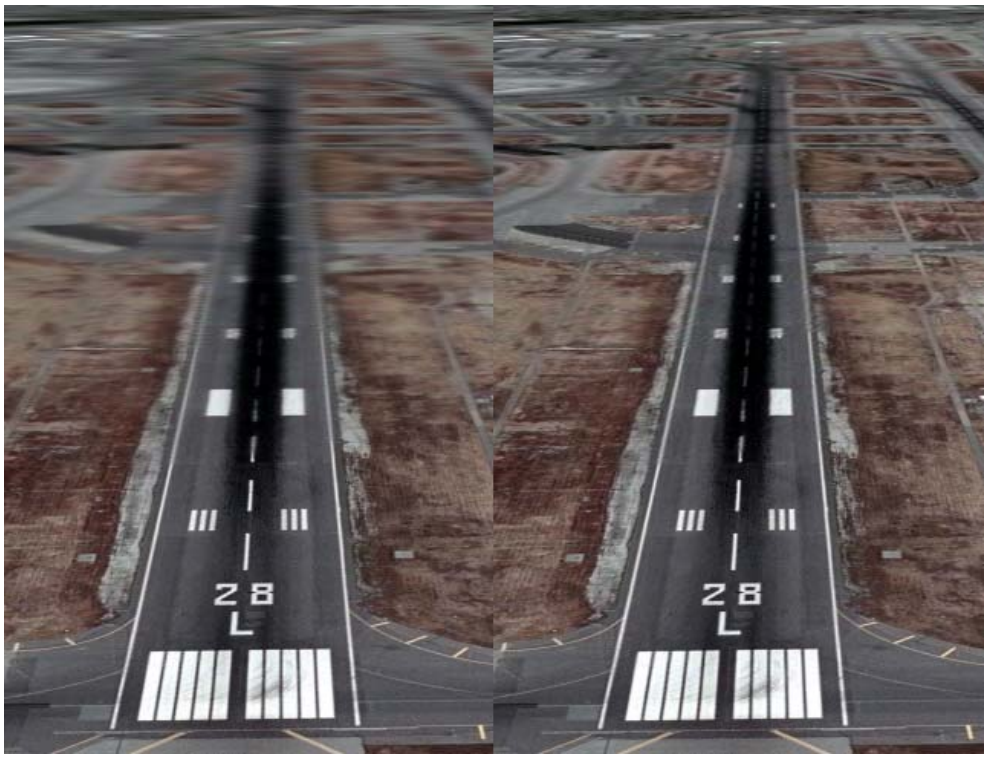


Figure 19. Coverage sampling antialiasing (4× MSAA vs. 16× CSAA)

GeForce 8800 GPUs support both the FP16 and the FP32 component for HDR rendering, which can work simultaneously with multisampled antialiasing delivering incredibly rich images and scenery.

Anisotropic Filtering (AF) improves the clarity and sharpness of various scene objects that are viewed at sharp angles and/or recede into the distance (Figure 20). One example is a roadway billboard with text that looks skewed and blurred when viewed at a sharp angle (with respect to the camera) when standard bilinear and trilinear isotropic texture filtering methods are applied. Anisotropic filtering (combined with trilinear mipmapping) allows the skewed text to look much sharper. Similarly, a cobblestone roadway that fades into the distance can be sharpened with anisotropic filtering.



(Photo courtesy of Wikipedia for public domain use)

Figure 20. Isotropic trilinear mipmapping (left) vs. anisotropic trilinear mipmapping (right)

Anisotropic filtering is memory bandwidth intensive, particularly at high AF levels. For example, 16×AF means that up to 16 bilinear reads per each of two adjacent mipmap levels (128 memory reads total) are performed, and a weighted average is used to derive final texture color to apply to a pixel.

GeForce 8800 GPU's new default high-quality anisotropic filtering removes angle optimizations and yields near perfect anisotropic filtering, as shown in Figure 21.

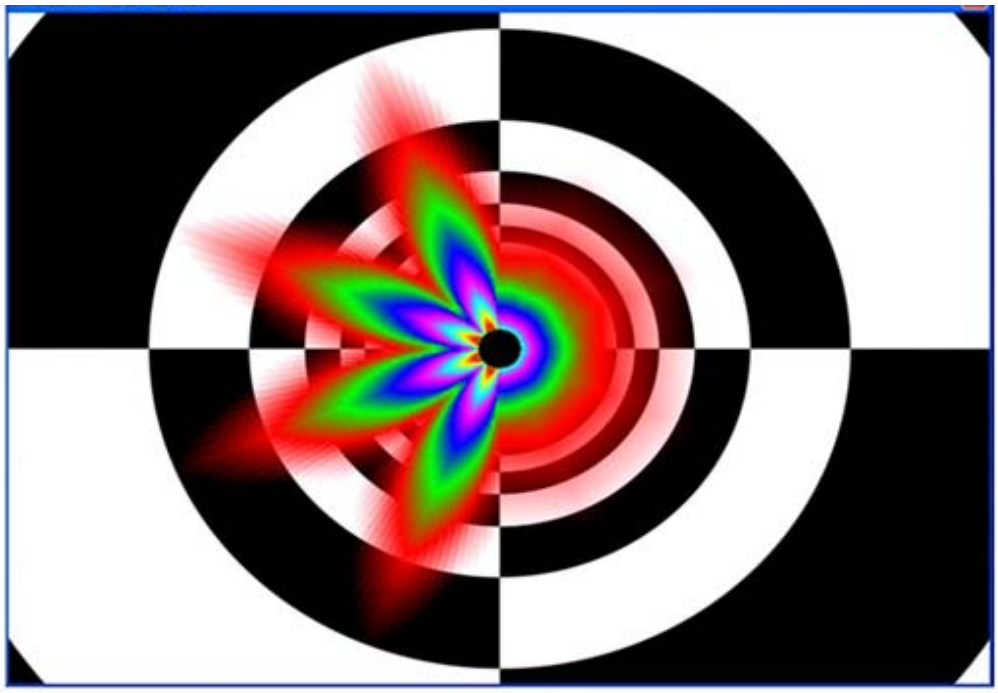


Figure 21. Anisotropic filtering comparison
(GeForce 7 Series on left, and GeForce 8 Series or right using default anisotropic Texture Filtering)

Refer to *Lumenex Engine: The New Standard in GPU Image Quality* (TB-02824-001) for more details.

Following is a discussion of three important architectural enhancements that permit better overall GPU performance.

Decoupled Shader/Math, Branching, and Early-Z

Decoupled Shader Math and Texture Operations

Texture addressing, fetching, and filtering can take many GPU core clock cycles. If an architecture requires a texture to be fetched and filtered before performing the next math operation in a particular shader, the considerable texture fetch and filtering (such as 16× anisotropic filtering) latencies can really slow down a GPU. GeForce 8800 GPUs can do a great job tolerating and essentially "hiding" texture fetch latency by performing a number of useful independent math operations concurrently.

For comparison, a GeForce 7 Series GPU texture address calculation was interleaved with shader floating-point math operations in Shader Unit 1 of a pixel pipeline. Although this design was chosen to optimize die size, power, and performance, it could cause some shader math bottlenecks when textures were fetched, preventing use of a shader processor until the texture was retrieved. GeForce 8800 GPUs attacks the shader math and texture processing efficiency problem by decoupling shader and texture operations so that texture operations can be performed independent of shader math operations.

Figure 22 illustrates math operations (not dependent on specific texture data being fetched) that can be executed while one or more textures are being fetched from frame buffer memory, or in the worst case, from system memory. Shader processor utilization is improved. In effect, while a thread fetching a texture is executing, the GeForce 8800 GPU's GigaThread technology can swap in other threads to execute, ensuring that shader processors are never idle when other work needs to done.

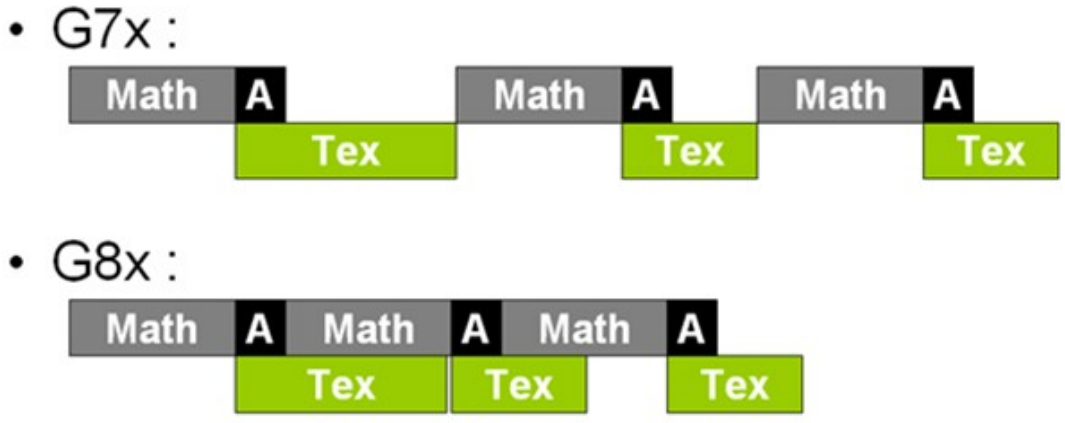


Figure 22. Decoupling texture and math operations

Branching Efficiency Improvements

An important aspect of overall GPU performance in processing complex DirectX 10 shader workloads is branch efficiency. For background and comparison, GeForce 7 Series GPUs were designed to be efficient when processing typical DirectX 9 shaders. When an *if-then-else* statement was encountered in pixel shader code, a batch of 880 pixels was processed at once. Some of the pixels in the batch would generally require pixel shading effects applied based on the "then" code path, while the other pixels in the batch just went along the same code path for the ride, but were effectively masked out of any operations. Then the whole batch would take the "else" code path, where just the opposite would occur, and the other set of pixels would respond to the "else" code, while the rest went along for the ride.

GeForce 8800 Series GPUs are designed to process complex DX10 shaders. Programmers will enjoy as fine 16-pixel branching granularity up to 32 pixels in some cases. Compared to the ATI X1900 series, which uses 48 pixel granularity, the GeForce 8800 architecture is far more efficient with 32 pixel granularity for pixel shader programs. The chart in Figure 23 shows perfect branch efficiency for even numbers of coherent 4×4 pixel tiles.

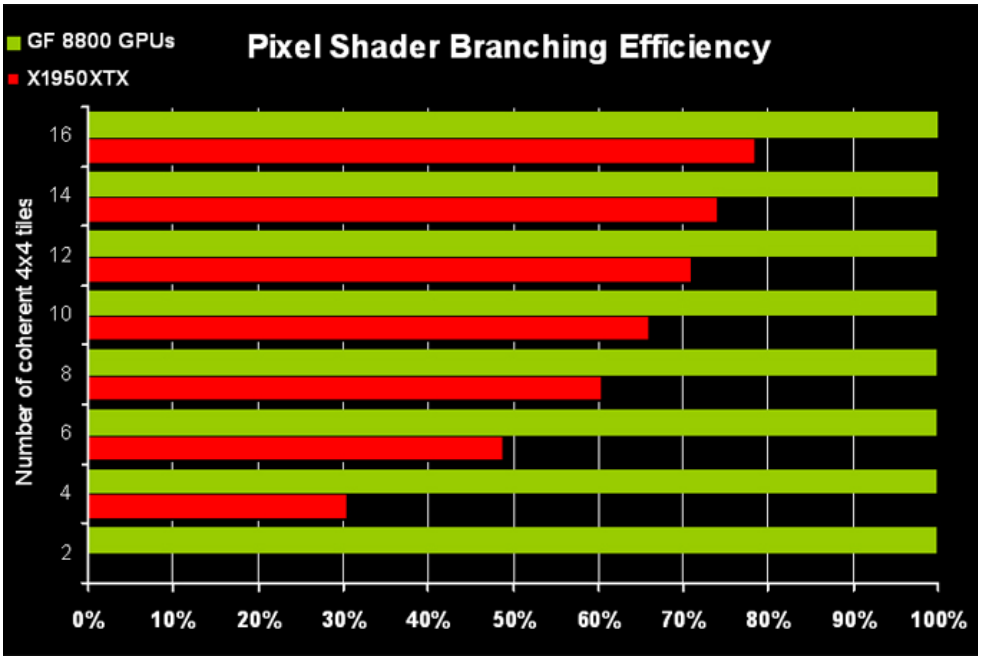


Figure 23. GeForce 8800 GPU pixel shader branching efficiency

Early-Z Comparison Checking

Modern GPUs use a Z-buffer (also known as depth buffer) to track which pixels in a scene are visible to the eye, and which do not need to be displayed because they are occluded by other pixels. Every pixel has corresponding Z information in the Z-buffer.

For background, a single 3D frame is processed and converted to a 2D image for display on a monitor. The frame is constructed from a sequential stream of vertices sent from the host to the GPU. Polygons are assembled from the vertex stream, and 2D screen-space pixels are generated and rendered.

In the course of constructing a single 2D frame in a given unit of time, such as 1/60th of s second, multiple polygons and their corresponding pixels may overlay the same 2D screen-based pixel locations. This is often called *depth complexity*, and modern games might have depth complexities of three or four, where three or four pixels rendered in a frame overlay the same 2D screen location.

Imagine polygons (and resulting pixels) for a wall being processed first in the flow of vertices to build a scene. Next, polygons and pixels for a chair in front of the wall are processed. For a given 2D pixel location onscreen, only one of the pixels can be visible to the viewer—a pixel for the chair or a pixel for the wall. The chair is closer to the viewer, so its pixels are displayed. (Note that some objects may be transparent, and pixels for transparent objects can be blended with opaque or transparent pixels already in the background or with pixels already in the frame buffer from a prior frame).

Figure 24 shows a simple Z-buffering example for a single pixel location. Note that we did not include actual Z-buffer data in the Z-buffer location.

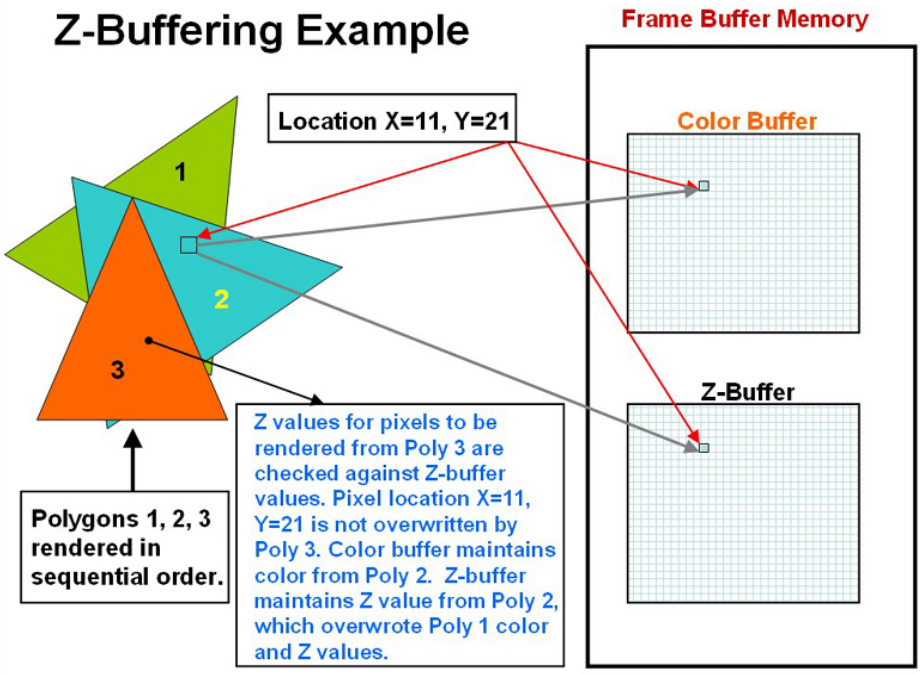


Figure 24. Example of Z-buffering

A few methods use Z-buffer information to help cull or prevent pixels from being rendered if they are occluded. Z-cull is a method to remove pixels from the pipeline during the rasterization stage, and can examine and remove groups of occluded pixels very swiftly.

A GeForce 8800 GTX GPU can cull pixels at four times the speed of GeForce 7900 GTX, but neither GPU catches all occlusion situations at the individual pixel level.

Z comparisons for individual pixel data have generally occurred late in the graphics pipeline in the ROP (raster operations) unit. The problem with evaluating individual pixels in the ROP is that pixels must traverse nearly the entire pipeline to ultimately discover some are occluded and will be discarded. With complex shader programs that have hundreds or thousands of processing steps, all the processing is wasted on pixels that will never be displayed!

What if an Early-Z technique could be employed to test Z values of pixels before they entered the pixel shading pipeline? Much useless work could be avoided, improving performance and conserving power.

GeForce 8800 Series GPUs implement an Early-Z technology, as depicted in Figure 25, to increase performance noticeably.

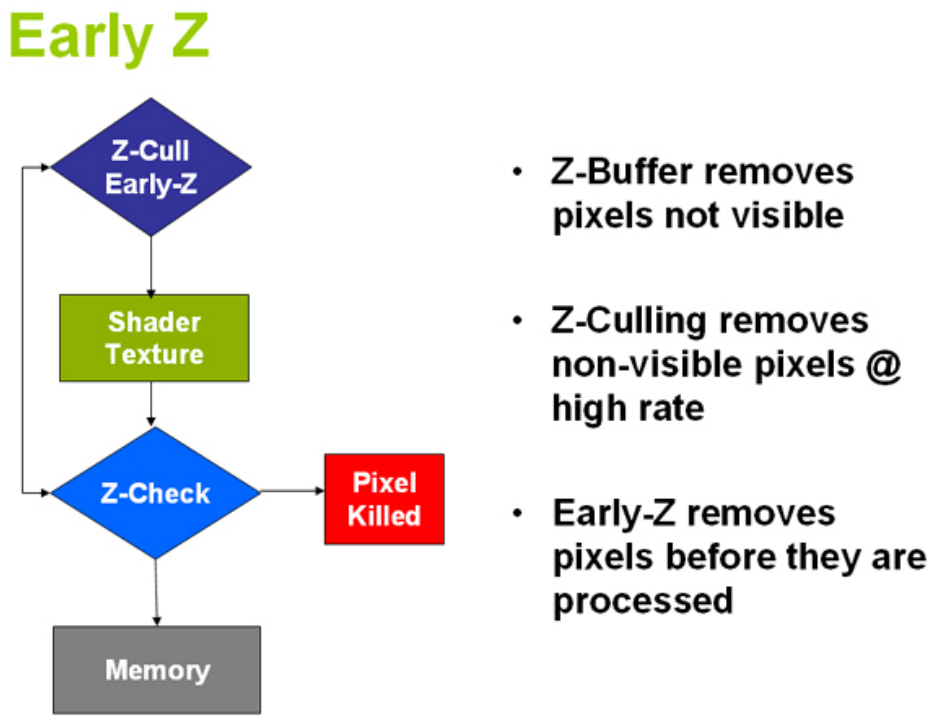


Figure 25. Example of early-Z technology

Next, we'll look at how the GeForce 8800 GPU architecture redefines the classic GPU pipeline and implements DirectX 10–compatible features. Later in this document, we describe key DirectX 10 features in more detail.

W have already covered a lot of the basics, so now we can look at the specifics of the GeForce 8800 GTX architecture without intimidation. The block diagram shown in Figure 26 should now look less threatening if you've read the prior sections.

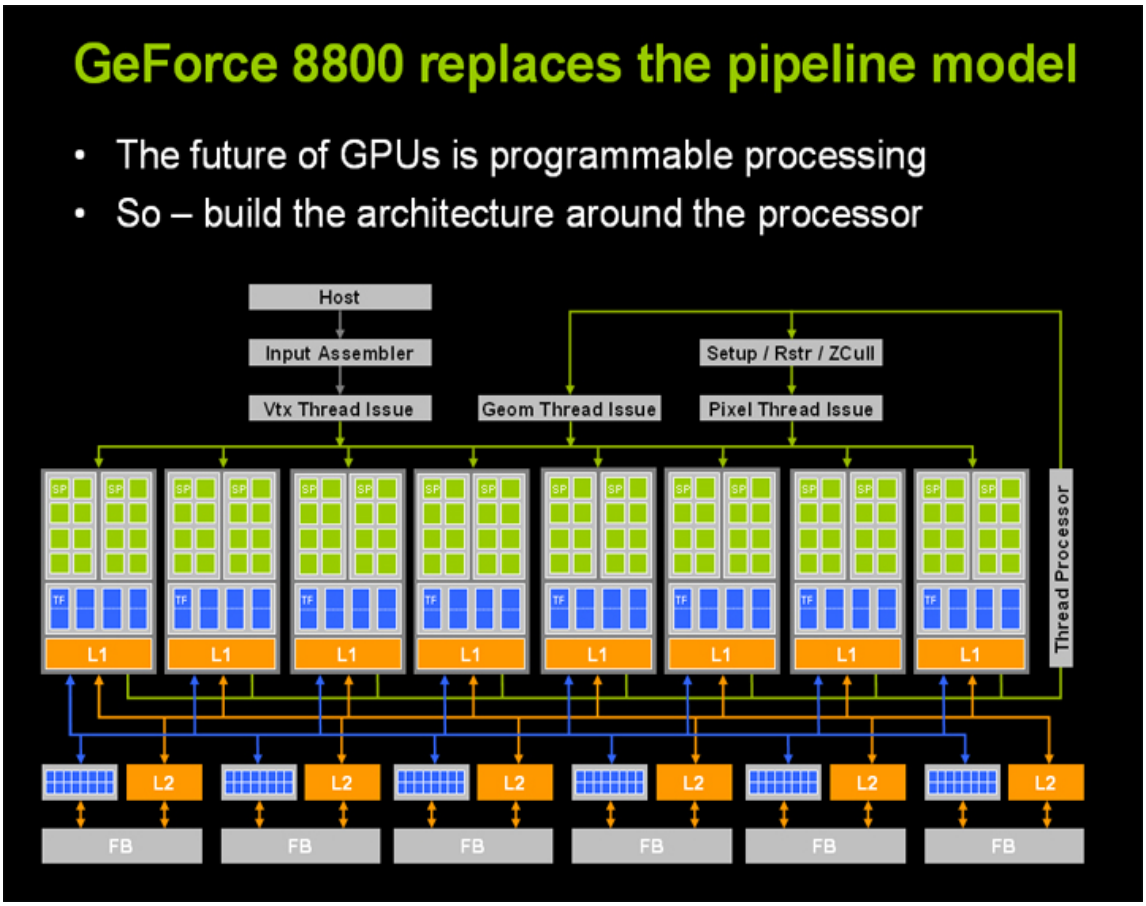


Figure 26. GeForce 8800 GTX block diagram

Host Interface and Stream Processors

Starting from the top of Figure 26 you see the Host interface block, which includes buffers to receive commands, vertex data, and textures sent to the GPU from the graphics driver running on the CPU. Next is the input assembler, which gathers vertex data from buffers and converts to FP 32 format, while also generating various index IDs that are helpful for performing various repeated operations on vertices and primitives, and for enabling instancing.

The GeForce 8800 GTX GPU includes 128 efficient stream processors (SPs) depicted in the diagram, and each SP is able to be assigned to any specific shader operation. We've covered a lot about unified shader architecture and stream processor characteristics earlier in this paper, so now you can better understand how the SPs are grouped inside the GeForce 8800 GTX chip. This grouping allows the most efficient mapping of resources to the processors, such as the L1 caches and the texture filtering units.

Data can be moved quickly from the output of a stream processor to the input of another stream processor. For example, vertex data processed and output by a stream processor can be routed as input to the Geometry Thread issue logic very rapidly.

It's no secret that shader complexity and program length continue to grow at a rapid rate. Many game developers are taking advantage of new DirectX 10 API features such as stream output, geometry shaders, and improved instancing supported by GeForce 8800 GPU architecture. These features will add richness to their 3D game worlds and characters, while shifting more of the graphics and physics processing burden to the GPU, allowing the CPU to perform more artificial intelligence (AI) processing.

Raw Processing and Texturing Filtering Power

Each stream processor on a GeForce 8800 GTX operates at 1.35 GHz and supports the dual issue of a scalar MAD and a scalar MUL operation, for a total of roughly 520 gigaflops of raw shader horsepower. But raw gigaflops do not tell the whole performance story. Instruction issue is 100 percent efficient with scalar shader units, and the mixed scalar and vector shader program code will perform much better compared to vector-based GPU hardware shader units that have instruction issue limitations (such as 3+1 and 2+2).

Texture filtering units are fully decoupled from the stream processors and deliver 64 pixels per clock worth of raw texture filtering horsepower (versus 24 pixels in the GeForce 7900 GTX); 32 pixels per clock worth of texture addressing; 32 pixels per clock of 2× anisotropic filtering; and 32-bilinear-filtered pixels per clock.

In essence, full-speed bilinear anisotropic filtering is nearly free on GeForce 8800 GPUs. FP16 bilinear texture filtering is also performed at 32 pixels per clock (about 5× faster than GeForce 7x GPUs), and FP16 2:1 anisotropic filtering is done at 16 pixels per clock. Note that the texture units run at the core clock, which is 575 MHz on the GeForce 8800 GTX.

At the core clock rate of 575 MHz, texture fill rate for both bilinear filtered texels and 2:1 bilinear anisotropic filtered texels is 575 MHz × 32 = 18.4 billion texels/second. However, 2:1 bilinear anisotropic filtering uses two bilinear samples to derive a final filtered texel to apply to a pixel. Therefore, GeForce 8800 GPUs have an effective 36.8 billion texel/second fill rate when equated to raw bilinear texture filtering horsepower.

You can see the tremendous improvement of the GeForce 8800 GTX over the GeForce 7900 GTX in relative filtering speed in Figure 27.

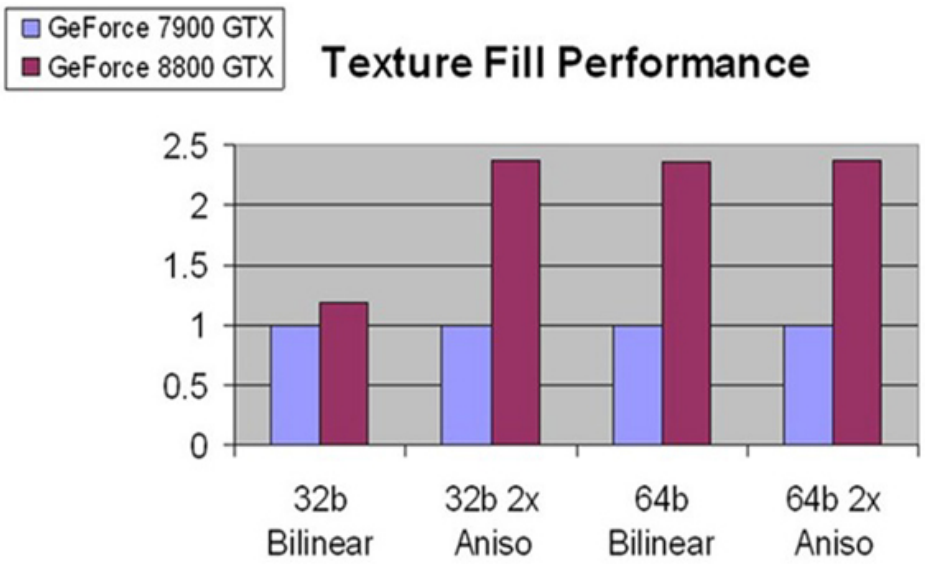


Figure 27. Texture fill performance of GeForce 8800 GTX

ROP and Memory Subsystems

The GeForce 8800 GTX has six Raster Operation (ROP) partitions, and each partition can process 4 pixels (16 subpixel samples, as shown in the diagram) for a total of 24 pixel/clock output capability with color and Z processing. For Z-only processing, an advanced new technique allows up to 192 samples/clock to be processed when a single sample is used per pixel. If 4× multisampled antialiasing is enabled, then 48 pixels per clock Z-only processing is possible.

The GeForce 8800 ROP subsystem supports multisampled, supersampled, and transparency adaptive antialiasing. Most important is the addition of four new single-GPU antialiasing modes (8×, 8×Q, 16×, and 16×Q), which provide the absolute best antialiasing quality for a single GPU in the market today.

The ROPs also support frame buffer blending of FP16 AND FP32 render targets, and either type of FP surface can be used in conjunction with multisampled antialiasing for outstanding HDR rendering quality. Eight MRTs (Multiple Render Targets) can be utilized, which is also supported by DX10. Each of the MRTs can define different color formats. New high-performance, more efficient compression technology is implemented in the ROP subsystem to accelerate color and Z processing.

As shown in Figure 26 six memory partitions exist on a GeForce 8800 GTX GPU, and each partition provides a 64-bit interface to memory, yielding a 384-bit combined interface width. The 768 MB memory subsystem implements a high-speed crossbar design, similar to GeForce 7x GPUs, and supports DDR1, DDR2, DDR3, GDDR3, and GDDR4 memory. The GeForce 8800 GTX uses GDDR3 memory default clocked at 900 MHz. With a 384-bit (48 byte wide) memory interface running at 900 MHz (1800 MHz DDR data rate), frame buffer memory bandwidth is very high at 86.4 GBps. With 768 MB of frame buffer memory, far more complex models and textures can be supported at high resolutions and image quality settings.

Balanced Architecture

NVIDIA engineers spent a great deal of time ensuring the GeForce 8800 GPU Series was a balanced architecture. It wouldn't make sense to have 128 streaming processors or 64 pixels worth of texture filtering power if the memory subsystem weren't able to deliver enough data, or if the ROPs were a bottleneck processing pixels, or if the clocking of different subsystems was mismatched. Also, the GPUs must be built in a manner that makes them power efficient and die-size efficient with optimal performance. The graphics board must be able to be integrated into mainstream computing systems without extravagant power and cooling.

Each of the unified streaming processors can handle different types of shader programs to allow instantaneous balancing of processor resources based on demand. Internal caches are designed for extremely high performance and hit rates, and combined with the high-speed and large frame buffer memory subsystem, the streaming processors are not starved for data.

During periods of texture fetch and filtering latency, GigaThread technology can immediately dispatch useful work to a processor that, in past architectures, may have needed to wait for the texture operation to complete. With more vertex and pixel shader program complexity, many more cycles will be spent processing in the shader complex, and the ROP subsystem capacity was built to be balanced with shader processor output. And the 900 MHz memory subsystem ensures even the highest-end resolutions with high-quality filtering can be processed effectively.

We have talked a lot about hardware and cannot forget that drivers play a large part in balancing overall performance. NVIDIA ForceWare® drivers work hand-in-hand with the GPU to ensure superior GPU utilization with minimal CPU impact.

Now that you have a good understanding of GeForce 8800 GPU architecture, let's look at DirectX 10 features in more detail. You will then be able to relate the DirectX 10 pipeline improvements to the GeForce 8800 GPU architecture.

The DirectX 10 specification, combined with DX10-capable hardware, has relieved many of the constraints and problems of pre-DirectX 10 classic graphics pipelines. In addition to a new unified instruction set and increases in resources, two of the more visible additions are an entirely new programmable pipeline stage called the *geometry shader*, and the stream output feature.

DirectX 10 and prior versions (with programmable pipeline capabilities) were designed to operate like a virtual machine, where the GPU is virtualized, and device-independent shader code is compiled to specific GPU machine code at runtime by the GPU driver's built-in Just-In-Time (JIT) compiler.

DirectX 9 Shader Model 3 used different virtual machine models with different instructions and different resources for each of the programmable pipeline stages. DirectX 10's Shader Model 4 virtual machine model provides a "common core" of resources for each programmable shader stage (vertex, pixel, geometry) with many more hardware resources available to shader programs. Let's look at the new virtualization model and Shader Model 4 a bit more closely.

Virtualization and Shader Model 4

You are likely familiar with the concept of virtualization of computing resources, such as virtual memory, virtual machines (Java VMs, for example), virtual I/O resources, operating system virtualization, and so forth.

Direct X shader assembly language is similar to Java VM language because both are a machine-independent intermediate language (IL) compiled to a specific machine language by a Just-In-Time (JIT) compiler. As mentioned above, shader assembly code gets converted at runtime by the GPU driver into GPU-specific machine instructions using a JIT complier built into the driver. (Microsoft high-level shader language (HLSL) and NVIDIA Cg high-level shader programming languages both get compiled down to the shader assembly IL format.)

While similar in many respects to Shader Model 3, new features added with Shader Model 4 include a new unified instruction set; many more registers and constants; integer computation; unlimited program length; fewer state changes (less CPU intervention); 8 multiple render target regions instead of 4; more flexible vertex input via the input assembler; the ability of all pipeline stages to access buffers, textures, and render targets with few restrictions; and the capability of data to be recirculated through pipeline stages (stream out).

Shader Model 4 also includes a very different render state model, where application state is batched more efficiently, and more work can be pushed to the GPU with less CPU involvement. Table 1 shows DirectX 10 Shader Model 4 versus prior shader models.

Table 1. Shader Model progression

	DX8 SM1.X	DX9 SM2	DX9 SM3	DX10
Vertex instructions	128	256	512	64 K
Pixel instructions	4+8	32+64	512	
Vertex constants	96	256	256	16 × 4096
Pixel constants	8	32	224	
Vertex temps	16	16	16	4096
Pixel temps	2	12	32	
Vertex inputs	16	16	16	16
Pixel inputs	4+2	8+2	10	32
Render targets	1	4	4	8
Vertex textures	N/A	N/A	4	128
Pixel textures	8	16	16	
2D tex size	–	–	2 K × 2 K	8 K × 8 K
Int ops	–	–	–	Yes
Load ops	–	–	–	Yes
Derivatives	–	–	Yes	Yes
Vertex flow control	N/A	Static	Static/Dyn	Dynamic
Pixel flow control	N/A	N/A	Static/Dyn	

Stream Output

Stream output is a very important and useful new DirectX 10 feature supported in GeForce 8800 GPUs. Stream output enables data generated from geometry shaders (or vertex shaders if geometry shaders are not used) to be sent to memory buffers and subsequently forwarded back into the top of the GPU pipeline to be processed again (Figure 28). Such dataflow permits more complex geometry processing, advanced lighting calculations, and GPU-based physical simulations with little CPU involvement.

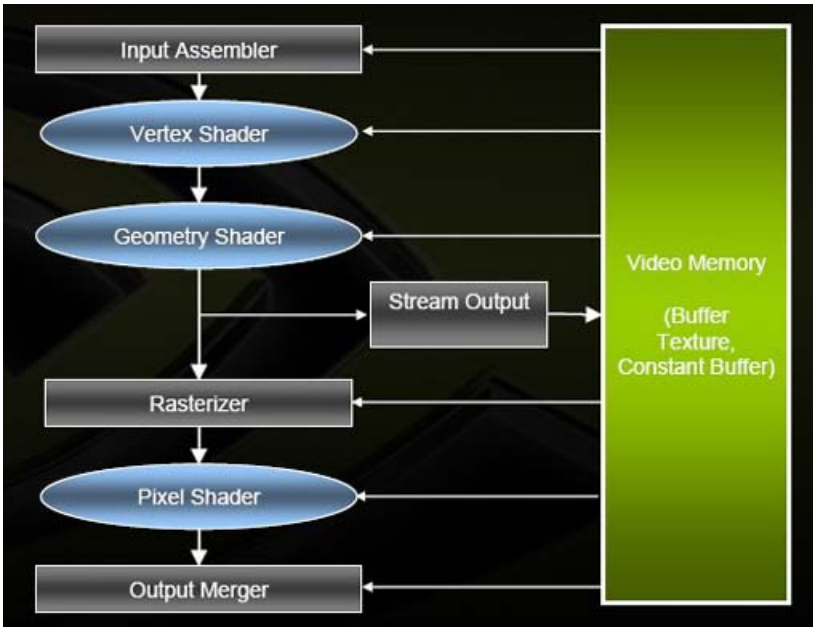


Figure 28. Direct3D 10 pipeline

Stream output is a more generalized version of the older "render to vertex buffer" feature that permits data generated from geometry shaders (or from vertex shaders if geometry shaders are not used) to be sent to "stream buffers" and subsequently forwarded back to the top of the pipeline to be processed again. (See "The Hair Challenge" for an example of usage.)

Geometry Shaders

High polygon–count characters with realistic animation and facial expressions are now possible with DirectX 10 geometry shading, as are natural shadow volumes, physical simulations, faster character skinning, and a variety of other geometry operations.

Geometry shaders can process entire primitives as inputs and generate entire primitives as output, rather than processing just one vertex at a time, as with a vertex shader. Input primitives can be comprised of multiple vertices, such as point lists, line lists or strips, triangle lists or strips, a line list or strip with adjacency info, or a triangle list or strip with adjacency info. Output primitives can be point lists, line strips, or triangle strips.

Limited forms of tessellation—breaking down primitives such as triangles into a number of smaller triangles to permit smoother edges and more detailed objects—are possible with geometry shaders. Examples could include tessellation of water surfaces, point sprites, fins, and shells. Geometry shaders can also control objects and create and destroy geometry (they can read a primitive in and generate more primitives, or not emit any primitives as output).

Geometry shaders also can extrude silhouette edges, expand points, assist with render to cube maps, render multiple shadow maps, perform character skinning operations, and enable complex physics and hair simulations. And, among other things, they can generate single-pass environment maps, motion blur, and stencil shadow polygons, plus enable fully GPU-based particle systems with random variations in position, velocity, and particle lifespan.

Note that software-based rendering techniques in existence for years can provide many of these capabilities, but they are much slower, and this is the first time such geometry processing features are implemented in the hardware 3D pipeline.

A key advantage of hardware-based geometry shading is the ability to move certain geometry processing functions from the CPU to the GPU for much better performance. Characters can be animated without having the CPU intervene, and true displacement mapping is possible, permitting vertices to be moved around to create undulating surfacing and other cool effects.

Improved Instancing

DirectX 9 introduced the concept of object instancing, where a single API draw call would send a single object to the GPU, followed by a small amount of "instance data" that can vary object attributes such as position and color. By applying the varying attributes to the original object, tens or hundreds of variations of an object could be created without CPU involvement (such as leaves on tree or an army of soldiers).

DirectX 10 adds much more powerful instancing by permitting index values of texture arrays, render targets, and even indices for different shader programs to be used as the instance data that can vary attributes of the original object to create different-looking versions of the object. And it does all this with fewer state changes and less CPU intervention.

Figure 29. Instancing at work—numerous characters rendered

In general, GeForce 8800 Series GPUs work with the DX10 API to provide extremely efficient instancing and batch processing of game objects and data to allow for richer and more immersive game environments.

Vertex Texturing

Vertex texturing was possible in DirectX 9 and is now a major feature of the DirectX 10 API and able to be used with both vertex shaders and geometry shaders. With vertex texturing, displacement maps or height fields are read from memory and their "texels" are actually displacement (or height) values, rather than color values. The displacements are used to modify vertex positions of objects, creating new shapes, forms, and geometry-based animations.

The *Hair* Challenge

The Broadway musical *Hair* said it best: "Long, straight, curly, fuzzy, snaggy, shaggy, ratty, matty, oily, greasy, fleecy, shining, streaming, flaxen, waxen, knotted, polka-dotted, twisted, beaded, braided, powdered, flowered, confettied, bangled, tangled, spangled, spaghettied, and a real pain to render realistically in a 3D game." OK, maybe not the last clause, but it's true!

A good example of the benefit of DirectX 10 and GeForce 8800 GPUs is in creating and animating complex and realistic-looking hair. Rendering natural-looking hair is both a challenging rendering problem and a difficult physics simulation problem.

Table 2. *Hair* algorithm comparison of DirectX 9 and DirectX 10

Algorithm	GeForce 7 Series	GeForce 8 Series
Physical simulation on control points	CPU	GPU
Interpolate and tessellate control points	CPU	GPU – GS
Save tessellated hairs to memory	CPU	GPU – SO
Render hair to deep shadow map	GPU	GPU
Render hair to back buffer	GPU	GPU

With DirectX 9, the physics simulation of the hair is performed on the CPU. Interpolation and tessellation of the control points of the individual hairs in the physics simulation is also performed by the CPU. Next, the hairs must be written to memory and copied to the GPU, where they can finally be rendered. The reason we don't see very realistic hair with DirectX 9 games is that it's simply too CPU-intensive to create, and developers can't afford to spend huge amounts of CPU cycles just creating and animating hair at the expense of other more important game play objects and functions.

With DirectX 10, the physics simulation of the hair is performed on the GPU, and interpolation and tessellation of control points is performed by the geometry shader. The output from the geometry shader is transferred to memory using stream output, and read back into the pipeline to actually render the hair.

Expect to see far more realistic hair in DX10 games that take advantage of the power of GeForce 8800 Series GPUs.

Conclusion

As you are now aware, the GeForce 8800 GPU architecture is a radical departure from prior GPU designs. Its massively parallel unified shader design delivers tremendous processing horsepower for high-end 3D gaming at extreme resolutions, with all quality knobs set to the max. New antialiasing technology permits 16× AA quality at the performance of 4× multisampling, and 128-bit HDR rendering is now available and can be used in conjunction with antialiasing.

Full DirectX10 compatibility with hardware implementations of geometry shaders, stream out, improved instancing, and Shader Model 4 assure users they can run their DirectX 10 titles with high performance and image quality. All DirectX 9, OpenGL, and prior DirectX titles are fully compatible with the GeForce 8800 GPU unified design and will attain the best performance possible.

PureVideo functionality built in to all GeForce 8800–class GPUs ensures flawless SD and HD video playback with minimal CPU utilization. Efficient power utilization and management delivers outstanding performance per watt and performance per square millimeter.

Teraflops of floating-point processing power, SLI capability, support for thousands of threads in flight, Early-Z, decoupled shader and math processing, high-quality anisotropic filtering, significantly increased texture filtering horsepower and memory bandwidth, fine levels of branching granularity, plus the 10-bit display pipeline and PureVideo feature set—all these features contribute to making the GeForce 8800 GPU Series the best GPU architecture for 3D gaming and video playback developed to date.

Notice

Trademarks

Copyright

NVIDIA Corporation
2701 San Tomas Expressway
Santa Clara, CA 95050
www.nvidia.com