Technical Brief

GeForce3: Lightspeed Memory Architecture

GeForce3: Lightspeed Memory Architecture

Creating a real-time realistic 3D environment on a mainstream PC is the driving ambition of thousands of hardware and software developers employed in the computer graphics industry. While developments over the past years have made incredible strides towards improving the quality of real-time 3D graphics, one of the fundamental challenges in delivering interactive 3D content remains: increasing performance given limited increases in memory. Memory and graphics bus bandwidth remain as critical factors in determining graphics performance and quality. GeForce3 incorporates a number of revolutionary advances in graphics architecture that dramatically improve the GPU's efficiency with memory and bus bandwidth delivering a new level of performance and image quality.

Bandwidth Challenges

Real-time 3D graphics place a tremendous load on the entire PC architecture. The nature of the graphics problem places strain on all aspects of a PC's subsystems: the microprocessor, main memory, the graphics bus (typically AGP), the graphics processing unit (GPU) and the GPU's frame buffer memory. Different components of a graphics application stress various components, which often results in the performance of a specific application limited by one system component at a given instant in time and a different component a fraction of a second later. The three key stressors of a system are computational load, geometry computation, and pixel rendering.

The 3D Graphics Problem

Understanding how graphics applications work is key to understanding the various challenges in improving graphics performance and bandwidth efficiency. A typical graphics application such as an interactive video game has four main components: game logic, scene management, geometry calculations, and pixel rendering. Each one of these is discussed below.

Game Logic

Compelling, interactive 3D applications require several elements to keep users interested and entertained. Game logic, physics, artificial intelligence (AI), networking, interactivity, sound, and other non-graphics functions are some examples, and are all elements, of the primary game engine code. The key to great content is delivering an engaging interactive experience. In order to create this experience, developers will allocate a majority of the central processing unit's processing power to tasks that directly create these elements of the user experience. To the extent that other aspects of the graphics application can be offloaded from the CPU, more CPU power can be dedicated to those elements. With today's modern GPU's, (like the GeForce2 GTS and GeForce3) much of the graphics "problem" is offloaded from the CPU, leaving more of the CPU to create compelling interactive experiences.

Scene Management

In order for graphics to be interactively rendered, a database describing the "3D world" and every single object in that world must be created. Typically these databases are very large, sometimes containing hundreds of megabytes, or even gigabytes of data. Rendering and displaying all of this data is simply not practical, even on high-end multiprocessing graphics supercomputers, so this task must be simplified. The graphics application must calculate what portion of the "world" or database is going to be processed, or rendered, at a given time. This process of calculating what portion of the database will be rendered at a time is commonly called scene management. A variety of techniques are typically used, all of which require tradeoffs between computational cost, memory requirement, and accuracy. Algorithms that minimize the amount of scene data required to render a frame exist, but are typically too computationally expensive for practical use. Hence, most graphics processors actually end up processing many times the amount of data than is actually displayed on the computer screen. Increasing the efficiency of the scene management processing can improve the performance of some graphics processors, but often comes at the cost of increasing the CPU load for these functions, which can detract from the processing power available for the game logic.

Geometry Calculations

Once the application calculates what portion of the scene to process (or render), the application passes that subset of its database to the geometry pipeline of the graphics application. Historically the geometry processing of graphics applications took place on the CPU as well. These computations typically involved transforming the geometry and lighting it. These calculations have recently been handed over to modern graphics processing units, beginning with NVIDIA's groundbreaking GeForce 256 and continuing with subsequent NVIDIA processors. By offloading the geometry burden from the host processor, more of the CPU is made available for game logic. Simultaneously, modern GPUs such as NVIDIA's GeForce family of graphics processors, can process many times the geometry data of even the fastest CPUs. By processing more geometry data, visual quality can be dramatically improved.

The Geometry Bandwidth Problem

In order to describe scenes with richness and enough detail to create a compelling environment, content developers have been increasing the geometric detail in their scenes at an incredible rate. With the introduction of the GeForce 256 graphics processing unit, much of the computational load for this function was moved from the CPU to the GPU. This shifting of the computational load to the GPU was a key factor in allowing content developers the freedom to move from hundreds of polygons per frame to hundreds of thousands. These rich scenes, while orders of magnitude more visually compelling, are also incredibly bandwidth intensive. With a typical scene of 100,000 polygons, each composed of three vertices, 300,000 vertices per frame is common. Each vertex can typically contain 50 bytes or more of information, for things like color, position, lighting, texturing, and shading information. Hence, it is common for each frame to contain 15 Mbytes of geometric information. While an individual frame certainly would not stress even a basic PC system's bus, the requirement to run this type of load at 60 frames per second makes this challenge daunting, as the 900 Mbytes/sec of bandwidth required for such a task will push every aspect of the PC to its limits.

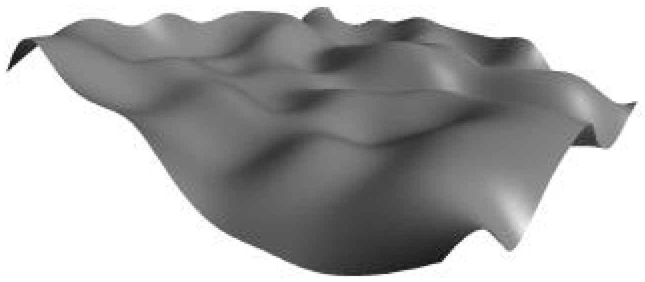
One challenge facing today's PC architectures with these types of loads is the communication between the host CPU, and today's GPUs. The link between those two systems is commonly Intel's AGP, or accelerated graphics port. PCI (peripheral component interconnect) is the second most common. The most advanced implementation of AGP today is AGP 4x. The AGP specification calls for a point-to-point connection between the host and the graphics processor; hence this interface is not shared with other devices. While a private interface certainly helps to address the issue of bandwidth between the two systems, even the 1.0GB/sec of bandwidth offered by AGP 4x is not sufficient for geometry-rich scenes.

NVIDIA has developed several unique solutions to address this geometry bandwidth problem. The first is high order surfaces.

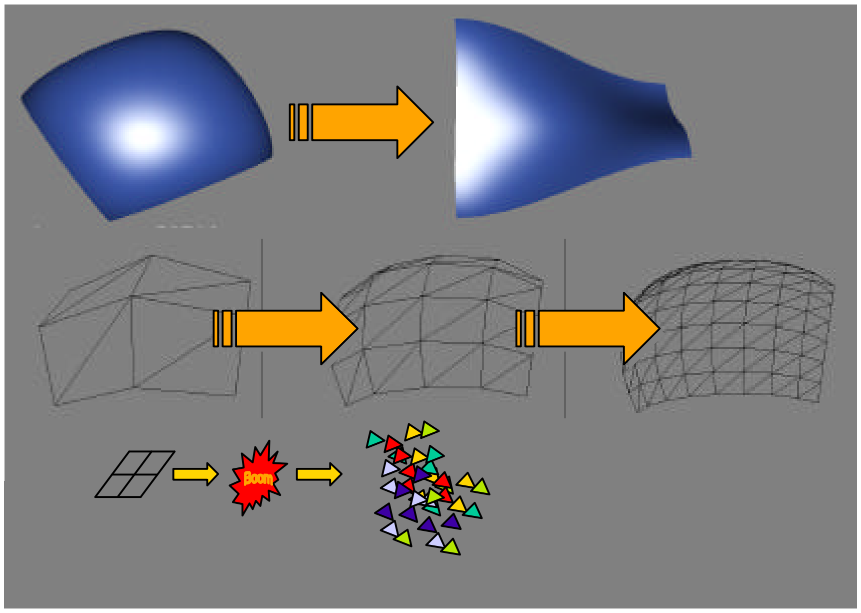
Higher Order Surfaces

Traditionally the fundamental building block of real-time 3D graphics has been triangles. Artists used collections of triangles, each built from three vertices (the corners of triangles), in order to build 3D objects. The challenge with this approach is that in order to create objects with rich detail, or smooth curves, the artist is forced to use an ever increasing number of triangles to get the fine levels of detail, or to get curved edges to appear smooth.

Higher order surfaces allow developers to create objects using curves defined by control points. A curve or surface defined with a set of control points is called a *spline*. While there are a number of different types of splines, the important thing is that by using a spline and a few control points, you can create fairly complex, smooth curved surfaces. Joining splines together allows a designer to create complex curved surfaces that are difficult to create by just using triangles.


Complex object created using higher order surfaces.

GeForce3 supports these curved surfaces in hardware, allowing for a much more efficient description of complex geometry without an every increasing number of triangles. By describing surfaces with control vertices, effectively the surface is being described by a formula, instead of thousands or millions of discreet values. The GeForce3 graphics processor is capable of processing these high order surfaces in hardware and in real time, essentially accepting these small, highly efficient formulas describing the geometry from the AGP bus, and then processing those formulas to “create” the geometry on the graphics processor.



The benefits of high order surfaces are clear. Much higher quality scenes, particularly scenes requiring smooth curves (such as columns to support the roof of a temple), and much more efficient use of the graphics bus. By transmitting only 16 control points, the GeForce3 graphics processor is able to generate the equivalent of hundreds of thousands of polygons worth of geometric data, effectively offering hundreds, or even thousands of times the efficiency of transmitting that triangular geometry data across the bus. The result. Better performance more of the time.

The Pixel Bandwidth Problem

To fully grasp the challenges of rendering a realistic 3D world requires an understanding of some basic 3D graphics concepts and terminology. A reference of useful terms can be found in the appendix at the end of this paper.

Calculating Pixel Memory Bandwidth

Traditional graphics architectures render pixels by reading from and writing to color and Z-buffers, and accessing texture data. They do this for every pixel they render, regardless of the pixel's visibility. Most graphics applications today actually render each pixel two to three times per frame, as objects often "occlude" or hide, other objects. A simple example would be a game with a background and a character in the foreground. Taken in its most basic form, such a scene would have a depth complexity of two, with the pixels of the background being hidden by the character in the foreground.

Rendering a single pixel once requires the graphics processor to read the color buffer, to discover the previous value, to read the Z-value to determine the depth in the scene for the pixel, and to read the texture data necessary to texture map that pixel. Once the pixel is generated it requires writing the new (potentially blended) color value to the color buffer, and potentially writing the new Z-buffer value. In the 32-bit depth rendering case, each of these operations requires 32-bits, or 4-bytes of data per access. So:

Color Read + Z-Read + Texture Read + Color Write + Z-Write

4 bytes + 4 bytes + 4 bytes + 4 bytes + 4 bytes = 20 bytes

This calculation assumes that the graphics processor is fetching one 32-bit texel per pixel, which makes the assumption that the remainder of texels (necessary to perform bilinear filtering) are already resident on-chip in the texture cache. 20 bytes may not seem like a lot of data, but when the complete frame is rendered 2.5 times per pixel (the average depth complexity) a more bandwidth intensive picture begins to emerge. Assume a resolution of 1024 pixels by 768 pixels.

Horizontal Resolution	x Vertical Resolution	x Depth Complexity	x 20 bytes/pixel
1024	x 768	x 2.5	x 20 = 39,321,600 bytes/frame

39.3Mbytes per frame x 60 frames per second = 2.4GB/sec

Rendering higher resolutions, higher frame rates, or higher depth complexity can have a dramatic impact on memory bandwidth requirements. Moving the resolution up to 1600 x 1200 pixels:

Horizontal Resolution x Vertical Resolution x Depth Complexity x 20 bytes/pixel

1600 x 1200 x 2.5 x 20 = 96,000,000 bytes/frame

96Mbytes per frame x 60 frames per second = 5.8GB/sec

Such tremendous amounts of memory bandwidth can only be accomplished with wide memory systems (128-bits) and high-speed memories. Today, typically double data rate, or DDR, memory is used. Even with such advanced memory subsystems, frame buffer bandwidth is one of the key limiters to increasing the resolution and/or frame rate of graphics applications. By increasing the efficiency in which the graphics processor renders pixels, dramatic improvements in performance can be achieved without increasing the memory bandwidth of the frame buffer. By improving the speed of the frame buffer (and hence memory bandwidth) and improving the efficiency in which pixels are rendered, dramatic breakthroughs in performance and visual quality become possible. GeForce3 implements both techniques to deliver such a breakthrough.

GeForce3 Pixel Memory Bandwidth Breakthrough

GeForce3 implements many patent pending technologies to improve the efficiency at which it renders pixels. Three of these key technologies are a crossbar-based memory controller to improve the efficiency of access to the frame buffer, lossless Z-buffer compression, and Z-Occlusion Culling to reduce the drawn depth complexity, and thus reduce the number of pixels that must actually read from and write to the frame buffer.

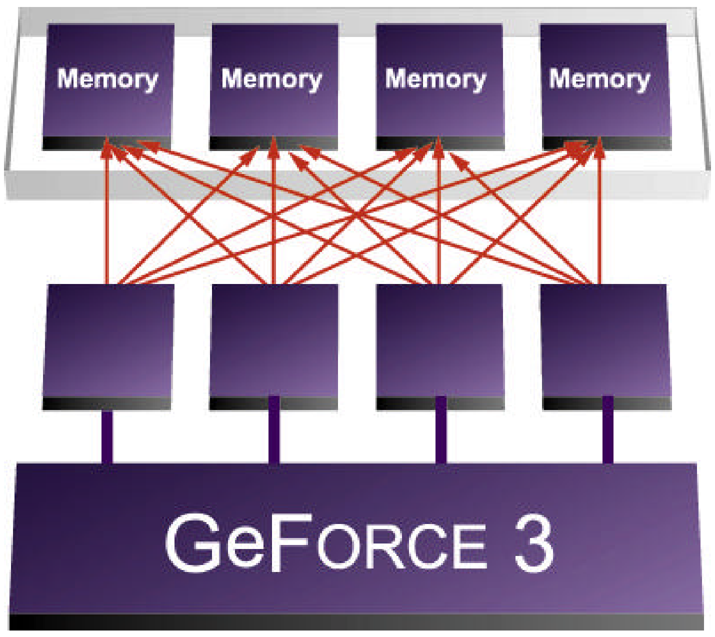
Crossbar Memory Controller

The memory controller is perhaps one of the most critical components of any graphics system. Because 3D graphics is so dependent upon memory bandwidth, the memory controller is at the crux of the bottleneck for improving performance. Besides the GPU, the other major component of a graphics system is the frame buffer. The frame buffer, which is the memory attached directly to the graphics processor, holds information such as color, depth values, textures and geometry, and is typically the highest bandwidth memory system in a personal computer. Unfortunately it is the most expensive part of a typical graphics system, often accounting for 50% of the cost of the product or more. Hence, it is critical to make the most efficient use of this expensive resource that is possible.

Traditional memory controllers have reached the point where they are reasonably efficient with basic loads, getting greater than 50% of the peak memory bandwidth from the frame buffer under most conditions. In today’s double data rate (DDR) based designs, a typical 128-bit memory controller will actually access information in 256-bit “chunks”. (Since DDR transfers twice the information in a single access) While it would seem that transferring large amounts of data in large blocks is generally optimal, in fact, with complex scenes with hundreds of thousands of polygons per frame, the reality is actually quite different.

Under conditions common in the latest generation of interactive content the size of the average triangle (again, the fundamental building block of all real-time graphics) can be very small, sometimes only a few pixels. If a triangle is perhaps 2 pixels in size, and is composed of 32-bits of color or Z for each pixel, the total amount of data for that triangle would be 32-bits x 2 pixels, or 64 bits. If memory controllers access information only in 256-bit “chunks” then much of this access would be wasted, as this “payload” or amount of data being transferred would essentially waste much of the frame buffer’s potential bandwidth. In this example, a traditional 128-bit memory controller would be only 25% efficient, "wasting" 75% of the memory bandwidth.

GeForce3 implements a radical crossbar memory controller that is optimized for accessing the frame buffer with a fine granularity access pattern, with up to 64-bits of individual access, ensuring that each individual access is completely efficient, thus ensuring that no fraction of the frame buffer's bandwidth is wasted. While the memory controller itself is still capable of accessing 256-bits of information in an individual clock cycle, the efficiency of each of those accesses is nearly perfect, keeping all aspects of the graphics processor and it's frame buffer fully utilized for maximum performance.



It does this by effectively implementing four independent memory controllers, each of which communicate with each other and the rest of the graphics processor. This complex system continuously load balances itself to ensure that every aspect of the memory system is balanced and that all memory requests by the graphics processor are handled properly. Under complex loads, typical of next generation content, the GeForce3 crossbar memory controller can be up to four times as efficient as previous less intelligent designs.

Lossless Z Compression

The Z-Buffer represents the depth, or visibility information for the pixels ultimately to be displayed after being rendered. Traditional graphics processors read and potentially write Z data for every pixel they render, making Z-buffer traffic one of the largest "consumers" of memory bandwidth in a graphics system. By implementing an advanced form of 4:1 lossless data compression the memory bandwidth consumed by Z-buffer traffic is reduced by a factor of four. This Z compression is implemented in hardware transparently to applications, with both compression and decompression taking place in real time by the Lightspeed Memory Architecture's Z compression/decompression engines. Because this compression is completely lossless there is no reduction in image quality or precision. The result of this technology is dramatically more efficient use of memory bandwidth for dramatically improved performance with no compromise in image quality.

Visibility Subsystem: Z-Occlusion Culling

As previously discussed, traditional graphics architectures render every pixel of every triangle as it receives them, accessing the frame buffer with each pixel to determine the appropriate values for the color and Z (or depth) for each of those pixels. This method produces correct results, but requires all of the pixels to be rendered, regardless of their visibility or not. Typical content today has an average depth complexity of 2, which means that for every pixel that ends up being visible, two pixels have to be rendered (on average) to come up with that result. This means that for every visible pixel, the graphics processor is forced to access the frame buffer twice, spending valuable frame buffer bandwidth essentially rendering pixels that the viewer will never see.

GeForce3 implements a sophisticated Z-Occlusion Culling technology, whereby it attempts to determine early if a pixel is going to end up being visible. If a pixel is going to be occluded and the Z-Occlusion Culling unit determines this, the pixel is not rendered, the frame buffer is not accessed, and the frame buffer bandwidth is saved. Depending on the depth complexity of the scene this can mean tremendous improvements in efficiency. With today's content, averaging a depth complexity of approximately 2, this technique could reduce bandwidth requirements by 50%. With next generation content approaching depth complexities of 4 or more, the benefits can be tremendous, with up to a four times improvement in memory bandwidth efficiency.

An additional technique, which developers can employ, is an "Occlusion Query". Essentially, the application makes a request of the graphics processor to render a bounding box or region to test for visibility. If the GPU determines that the region is going to be occluded, then all the representative geometry and rendering representing that region can be skipped over, potentially offering an order of magnitude increase in fill rate, as characters behind walls, or scenery outside of a tunnel can simply be "occlusion queried" and skipped over, without spending precious memory bandwidth or GPU processing time to render them.

These two key technologies effectively amplify the bandwidth of an GeForce3 graphics processor, both by getting dramatically more efficiency from the memory bandwidth offered by the frame buffer, and by making more efficient use of the frame buffer by avoiding having to access it for pixels that would not be visible. In some cases each of these benefits can demonstrate as much as four times the performance of previous architectures, while in practice the typical benefit of these memory bandwidth amplification technologies averages a 50%-100% improvement.

GeForce 3 Lightspeed Memory Architecture

GeForce 3 brings an array of technology to bear on the challenge of memory bandwidth. By representing complex geometry as a high order surface and performing those surface calculations entirely on the GPU, GeForce3 is able to avoid transmitting tremendous amounts of triangle data across the AGP bus, ensuring that communication between the host and the GPU can continue in an efficient and high performance manner. By attacking the pixel bandwidth problems in a variety of ways, GeForce3 brings a tremendous leap in memory bandwidth efficiency to PC graphics. The combination of the most efficient and sophisticated crossbar-based memory controller ever built for PC graphics, advanced lossless Z compression for reduced bandwidth consumption, and a highly advanced Z-Occlusion Culling method to avoid rendering and spending bandwidth on non-visible pixels means that GeForce3 makes twice the use of memory bandwidth than any previous traditional architecture.

These advances pave the way for an increasingly dynamic and visually rich real time 3D graphics experience. By improving the efficiency of communication between the host and graphics content developers can continue to increase the geometric richness and visual complexity of their scenes to new levels, unbound by the limits of the AGP bus. Rendering at high resolutions, with high frame rates becomes the standard with GeForce3, as its advances in pixel rendering and memory efficiency mean that frame buffer bandwidth boundaries have been broken, paving the way for the first time for high resolution, 32-bit rendering without substantial performance penalties.

Appendix

Resolution

Resolution is the number of pixels on a screen. Higher resolutions can create a more realistic 3D environment because more scene detail can be displayed. Most modern displays are capable of at least 1280 horizontal pixels by 1024 vertical pixels, while many larger or more expensive displays are capable of 1600x1200 or even 2048x1536 pixels. Most graphics applications support a variety of resolutions, allowing the end user to run at higher resolution (and hence higher level of detail) with the tradeoff being increased load on the graphics processing system.

Bit Depth

The bit depth refers to the number of bits of precision for the color and Z-values associated with each pixel on the screen. More bits of precision improve the visual realism and accuracy of the rendered frame. The two most common bit depths in modern graphics hardware are 16-bits and 32-bits. Each of these values can be associated with color or Z-values. 32-bit color (for example) typically is used to represent red, green, blue and alpha (or transparency) values with up to 8-bits per component, or 256 “values” for each of those components. A 32-bit Z-value is typically allocated as 24-bits of Z precision (or depth precision) and 8-bits of stencil or “mask” precision.

Frames per Second

Frames per second (fps), or frame rate, refers to how many times per second the scene is updated by the graphics processor. Higher frame rates yield smoother, more realistic animation. It is generally accepted that 30 fps provides an acceptable level of animation, but increasing the performance to 60 fps results in significantly improved interaction and realism. Beyond 75 fps it is difficult to detect any performance improvement. Displaying images faster than the refresh rate of the monitor results in wasted graphics computing power, as the monitor is unable to update it’s phosphors (or display) that fast, essentially wasting frame rate beyond its refresh rate.

Depth Complexity

Depth complexity is a measure of the complexity of a scene. It refers to the number of times any given pixel must be rendered before the frame is done. For example, a rendered image of a wall has a depth complexity of one. An image of a person standing in front of a wall has a depth complexity of two. An image of a dog behind the person but in front of the wall has a depth complexity of three, and so on. As depth complexity increases, more rendering horsepower and bandwidth is needed to render each pixel or scene. The average depth complexity of today’s graphics applications is two to three, meaning that for every pixel you end up seeing, it gets rendered two or three times by the graphics processor.

Texture Mapping

Texture mapping is the technique of projecting a 2D image (typically a bitmap) onto a 3D object. Texture mapping allows substantial increases in visual detail without significant increases in polygon count. Because of the improved realism that can be obtained with a very small increase in computational cost, texture mapping is one of the most common techniques for displaying realistic 3D objects. In order to render a texture-mapped pixel, the texture data for that pixel needs to be read into the graphics processor, consuming memory bandwidth.

Fill Rate

Fill rate is the rate at which pixels are drawn into the screen memory. Fill rate is a common measure used to illustrate the pixel processing capabilities of today’s 3D graphics processors. Fill rate is usually measured in millions of pixels/second (Mpixels/sec). In 1997, 50-70 Mpixels/sec was considered state-of-the-art. In 2001, the leading 3D graphics processors will be capable of 800-1000 Mpixels/sec. While this improvement is an incredible achievement, it is barely enough to create a compelling 3D environment. Rendering pixels at such a high rate can consume enormous amounts of memory bandwidth.

Memory Bandwidth

Memory bandwidth refers to the rate at which data is transferred between the graphics processor and graphics memory. Memory bandwidth limitations are one of the key bottlenecks that must be overcome to deliver truly realistic 3D environments. To deliver truly stunning 3D requires high resolution, 32-bit color depth at high frame rates, with rich geometry, sophisticated texture mapping, and complex vertex and pixel shading.