DirectX 10.1 Architecture

for

Chrome 400/500

Discrete Graphics Processors

A

S3 Graphics

White Paper

www.s3graphics.com

WP020-B.0
11/11/2008

Revision History

B.0	11/11/2008	Added Chrome 500 Series GPU Support	BT/KG
A.0	01/29/2008	Initial Version	BT/KG

Introduction

The introduction of Microsoft's DirectX® 10 (DX10) application programming interface (API) provided a radical change to the underlying hardware and software architecture with the introduction of unified shaders with geometry, vertex, and pixel processing capabilities. Additional features and instruction set enhancements were added to the specification to provide real-life, cinematic visual quality for all 3D related applications.

The latest refresh to the DX10 API is DX 10.1, which was released in early 2008 with Windows Vista™ Service Pack 1 (SP1). DX10.1 is based on the DX10 architecture and includes additional features and instructions to enhance rendering quality by providing improved filtering and anti-aliasing support. Some features that were optional in DX10 are now a mandatory part of DX10.1, including finer granularity of data access to improve data throughput and usage, and better standardization of the specification to improve third party application compatibility.

S3 Graphics works on technological advancements in close collaboration with Microsoft, to support not only the latest upgrades to the Windows Vista and future operating systems, but also to support DX10.1 applications from third party application vendors. The Chrome 400/500 Series of graphics processors (GPUs) enable DX10.1 support for the mainstream market, and also incorporate support for technologies, such as second generation PCI Express® bus (PCIe Gen2) and an enhanced HD video processor for Blu-ray™ playback.

Please refer to the S3 Graphics' White Papers WP016 and WP019 for more background information on the DirectX 10 architecture and the ChromotionHD video processor of the Chrome 400/500 Series GPUs.

DirectX 10.1: Features and Benefits

DX10.1 is based on the Shader Model 4.1 (SM4.1) specification, which is an incremental upgrade to the Shader Model 4.0 specification of DX10. Features that were optional in DX10, such as 32-bit floating point filtering and 4x MSAA (Multisampling Anti-Aliasing), are requirements for DX10.1 graphics hardware. Key DX10.1 benefits can be broken down into the following categories:

- anti-aliasing enhancements,
- image and texture improvements, and
- stringent specification features to standardize the API.

Anti-Aliasing Enhancements:

In high resolution images and photo-realistic rendering, the aliasing effects of jagged lines and blocky artifacts can reduce image quality as resolution and rendering frame rates scale higher. In addition, aliasing can happen during texture filtering when texture samples have very fine detail that do not map one-to-one to (coarse) texture coordinates onscreen.

To mitigate these unwanted artifacts, different anti-aliasing methods can be implemented to make edges and surface transitions appear smooth and continuous. The most common method blends multiple samples for each pixel to create an ideal final pixel value (MSAA). In Chrome 400/500 Series processors, the filtering algorithm follows the MSAA method by using the Z-buffer to correctly determine the position of the current pixel relative to a rendered object and giving it the correct value. If the pixel is completely covered by the object, the pixel will take the full value of the object. If the pixel is partially covered by the object, then the pixel's final value will be a weighted value based on the blending factor of the object and pixel (background color). The Chrome GPUs also have proprietary technology to effectively sample "ideal" points with the correct blending factors to create smooth and appealing images, along with a customizable AA filter using the Pixel Shader (PS) and new PS instructions.

- **4x MSAA Support Minimum:** In DX10, 4x MSAA was an optional feature. In DX10.1, this AA implementation is mandatory. All graphics hardware is required to support four samples per pixel for all surface formats except for cases where the bits per pixel times the numbers of samples is greater than 256. For example, hardware is not required to support 128bpe at 4x AA mode. However, hardware support for single channel float, such as R16_FLOAT or R8_UNORM, is required for 4x AA mode.

- **S3G Defined AA Sample Patterns:** Microsoft has standardized two sample patterns that are required by DX 10.1 hardware as 1x and 4x samples. Hardware vendors can support additional sample counts beyond 4x, and standard patterns for 2x, 4x, 8x, and 16x AA modes are defined in the Microsoft specification. These sample patterns provide a quick and clean way of querying the current image or scene to get the best approximation of sample positions. Chrome 400/500 processors can implement six additional AA sample patterns, giving developers a total of eight patterns to utilize. These additional patterns allow third party software developers more control over sample locations for individual pixels and also resolve temporal anti-aliasing side effects of the rendering process.

- **Multi Frequency AA:** When performing AA on primitives, developers can selectively choose to run Chrome processors at sample or pixel frequency. The pixel shader can operate on pixel frequency inputs – in which case the Chrome processors execute the shader once per pixel instead of once per sample and the result will be replicated to all samples. The pixel shader can also operate in sample frequency mode – in which case the Chrome processors can compute separate colors per sample, i.e., the shaders execute once per sample for four samples in 4x mode. This mixed interpolation mode allows Chrome processors to save bandwidth on some objects instead of always being required to perform full scene AA.

- **MSAA Depth (Z) Buffer Read/Write:** In DX10 shader units could only access multi-sampled color buffers, preventing the shaders from reading / writing depth buffer data for each individual sample. With DX10.1, all color and depth samples stored in the multi-sample buffer can be accessed directly by the shader units. This helps decrease jagged edges on shadow maps when smoothing calculations are performed on object / polygon edges.

- **Pixel (Sample) Coverage Masking:** This feature allows the Chrome 400/500 Series processors to selectively control which samples to output from the Pixel Shader, allowing per pixel AA control and precision by approximating sampling over an area of pixels.

- **SampleInfo and SamplePos:** These instructions help enhance control over the MSAA algorithm inside the pixel shader. DX10.1 applications can retrieve the number of samples in a given shader via SampleInfo or query the position of a sample via SamplePos. when performing high-level rendering such as deferred or High Dynamic Range (HDR) rendering, new AA shader instructions provide more precise control of sample positions which reduce or eliminate most of the side effects of aliasing.

Image and Texture Improvements:

Improved image quality is a major benefit of DX10.1. The technological breakthroughs of DX 10.1 provide support for higher levels of detail per scene, for increased density of application content, and for higher resolution monitors, such as widescreen and HD monitors which are increasingly prevalent in the marketplace. With HD image quality now approaching the norm, the upgrades to the DirectX specification help make significant progress towards the goal of life-like image quality. Some of the major updates from DX10 to DX10.1 are listed below.

- **Per Render Target Blending:** DX10 allowed a single blend factor for each render target stored in the buffer. For multiple render target (MRT) implementations in DX10, each object in an image would have the same blending factor. With the introduction of per render target blending in DX10.1, each object in a scene can have its own distinct “look and feel”, giving each image (render target) a more realistic appearance.

- **Realistic Lighting / Shadow Rendering:** Some methods for rendering realistic scenes with lighting and shadow effects include light / shadow mapping, ray tracing, and the use of global illumination algorithms. Light / shadow mapping provides direct lighting only, while the other two methods use both direct and indirect lighting to generate more realistic effects.

The first two methods have both advantages and disadvantages in their computational speed and resulting image quality. Chrome 400/500 Series processor hardware accelerates the execution of global illumination algorithms, the method that overcomes the shortcomings of the other two methods and are clear forerunners in speed and quality. DX10.1 introduces new features, such as Texture Cube Map Arrays, which allow global illumination to expand on the benefits of light / shadow mapping and ray tracing. Together these advancements help achieve an excellent balance between computational complexity and the production of high quality graphics images.

- **Light / Shadow mapping** (direct lighting) renders each object in a scene based on the view point of each light source. Each pixel is evaluated based on light intensity, color, visibility, and texture (surface shape and material) to determine the pixel's final value. Pixels not directly in the path of the light source are considered shadowed. This method is effective for speedy rendering of any scene containing a minimal number of light sources, but scene details (reflections and shadows) suffer because the GPU needs to render separate cube maps for reflections. With this method indirect light sources are not captured and therefore are not available to give the scene additional photorealistic effects.

- **Ray tracing** is a method that utilizes a direct path (ray) from a viewpoint to each pixel, and takes into account the accumulation of all the light sources in the scene, in order to get the final pixel value. Each ray is also tested for its intersection with any objects in the scene. If there is a collision, the pixel value is updated and new rays are recast from the collision point. The value of the recast ray is determined by that material's property, color, light, and other neighboring ray values. This method produces the effect of bouncing light (indirect rays) with real refraction and shadow effects. This process provides high quality lighting and shadow effects, but the computational complexity, high shader usage per pixel, and data structures required to store intermediate pixel values and locations make ray tracing ideal only for basic scenes with minimal detail. Use of ray tracing for dynamic scenes with high refresh rates, resolutions, and greater detail would burden and slow the GPU. The extent of slowdown is so great that, in order to increase frame rate, software vendors often use rough approximations derived from uniform illumination information to represent shadows and reflections of indirect light sources. These approximations increase performance, but come at the cost of scene quality and detail.

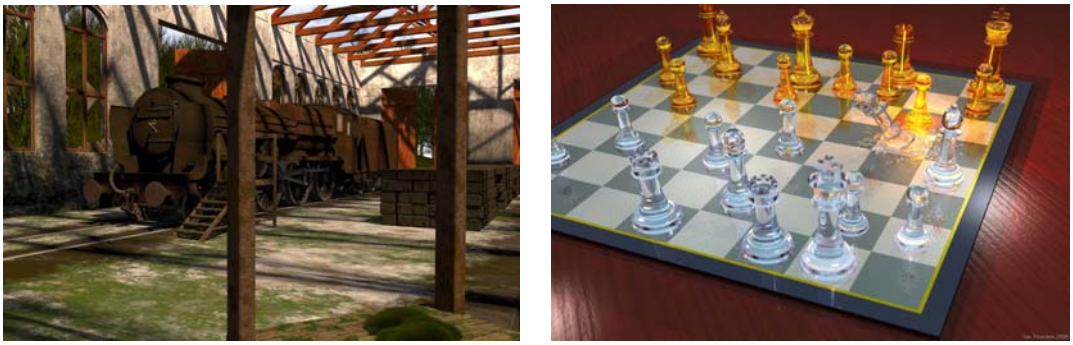

Figure 1. Ray Tracing Examples
(from http://irtc.org)

- **Global Illumination** is a rendering method that produces realistic lighting effects. It is similar to the preceding methods in that it includes both the direct light sources used for light / shadow mapping but also renders the indirect (reflected) light sources, as does ray tracing. The introduction of Texture Cube Map Arrays overcomes the significant hardware load of ray tracing by allowing efficient real time rendering. Each scene is separated into a 3D array of cubes where each cube face is rendered at low resolutions and level of detail for ease of storage. The texture cube maps are then translated into spherical representations that make it easy to determine the direction(s), intensity, and color of each surface using fewer calculations. This implementation also allows edge smoothing ambient occlusion based on the attenuation of light as objects approach one another, and life like reflections on polished surfaces. The quality level, detail, individual cube map size and total size of the array can be scaled to meet performance requirements, visual quality, and display settings. In DX10, only one cube map could be worked on at a time. In DX10.1, an entire array composed of multiple cube maps can be indexed into and worked on in parallel. This keeps the graphics processors' shader hardware units busy and allows for greater rendering efficiency.

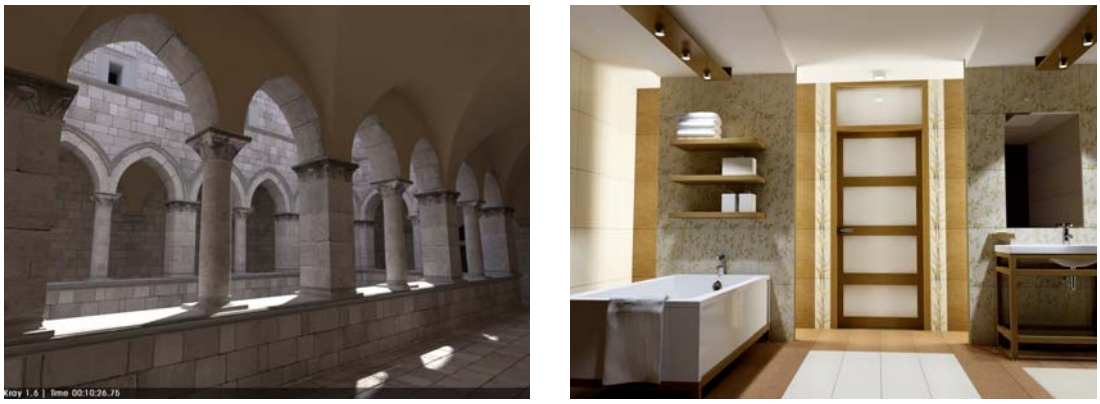

Figure 2. Global Illumination Examples (Left: KRay Global Illuminator Renderer)

- **Vertex Shader Input/Output Size Increase:** With the new Microsoft DX10.1 API each vertex shader can support up to 32 128-bit values, increasing the throughput of and performance for vertex processing. In DX10, only 16 128-bit values could be transferred to / from the vertex shader.

- **New Level-of-Detail (LOD) Instruction:** A new LOD shader instruction has been added for the unified DX10.1 architecture to allow parsing of information about the textures in a scene, based on the texture's size, distance, and importance. For different Level of Detail textures, a Chrome graphics processor can alter the number of rendering passes to achieve a balanced combination of visual quality and rendering performance. The LOD shader instruction returns a LOD value that can be used for texture filtering and, based on this returned value, viewing conditions, and shader resources, the graphics processor allocates shaders in the most optimal arrangement to render high quality, detailed textures and at the same time accommodating customizable texture filtering. Application developers now have more control over the hardware and the visual outcome of each frame than they ever did in previous generations of hardware.

- **Updates to Existing Formats:** SNORM formats can be directly blended instead of being converted to other formats for blending. This helps improve the accuracy of data during the conversion process. The DX10.1 API also allows a bitwise copy from uncompressed to block compressed format, which helps increase data throughput and compression performance.

Strict API Standardization:

Starting with DX10, the Microsoft specification binds the 3D hardware feature set with the DirectX version number, so that consistency across all hardware vendors exists and a set of basic features is identified as supported by all graphics vendors. In DX10.1, this standardization has been taken this one step further by normalizing some of the key optional features. In the past, the specification allowed hardware vendors to interpret the DirectX specification to match their specific hardware, causing problems such as performance degradation when there was no alignment between hardware and software application vendors. In the end, software developers would workaround this problem by using only the most common specification features and not incorporating any enhanced and optional elements. This approach limited the true range of 3D features found in the Microsoft standard. The stricter standardization introduced with DX10.1 was necessary to resolve these hardware / software compatibility issues, and enhances the user experience for all by accelerating the adoption of advanced 3D graphics in a greater number of applications.

- **Higher Precision Texture Formats:** Shader aliasing caused by rounding errors during math operations can cause loss of detail or texture errors based on the bit resolution. 32-bit floating point filtering (FP32) is now a requirement for the filtering of 128-bit formats, an improvement over the FP16 filtering standard used in DX10. This increase in the number of bits helps mitigate the shader AA issues found in DX10. For 64-bit pixels, 16-bit integer blending (Int16) is standard. Both formats introduce a higher level of precision, and help to provide realistic, cinematic HDR rendering and ensure all hardware is compatible with these new formats.

- **Increased Floating Point Operation Precision:** When performing complex floating point operations, the accuracy of the calculation is dependent on the number of bits. If the precision or number of bits is not sufficient for the calculation, then rounding errors occur because bits at the least significant bit positions are extrapolated. When performing multiple passes through the graphics pipeline and re-using the rounded data, a rounding error can grow exponentially. To avoid rounding errors the DX10.1 specification adopts the IEEE standard for 32-bit floating point precision. DX10.1 also features a ULP (Unit in the Last Place) of 0.5 for all add, subtract, multiply, divide, and blending operations.

- **Multi-Sample Anti-Aliasing Improvements:** Two pre-defined sample patterns and 4x MSAA are part of the DX 10.1 specification. These have been added to prevent or reduce the appearance of jagged edges, wavy lines, and sparkling dots when rendering high-quality, high-resolution images.

Enhancing the Windows Vista User Experience

DX10.1 provides significant improvements to build on the Windows Vista user experience.

- **Multi-Core System Performance Improvements:** The underlying software architecture of DX10.1 is aligned with the growth of upcoming multi-core CPUs to take advantage of multiple threads and processing units for higher performance over first generation multi-core CPUs.

- **API Calls and Overhead Reduced:** DX10 was a major overhaul in the DirectX architecture and included major changes in communication channels between GPU and CPU. API overhead, state changes, run-time efficiency, and CPU-heavy rendering calls were significantly reduced in DX10 compared to previous iterations of DirectX APIs. With the introduction of DX10.1, additional mechanisms to further improve efficiency and reduce API overhead and CPU calls/usage have been implemented, and should result in improved application performance. For example, two common features in today's games that utilize heavy GPU-CPU interaction (API calls) are the rendering of reflections and refractions. In DX10.1 visual effects take significantly less API overhead, and the GPU has more control. Both performance and visual quality benefit from these refinements.

- **Visual Quality Improvements:** The overall quality and feel of Microsoft Windows Vista applications, when viewed in a DX10.1 environment which takes advantage of the many advanced features of the specification, is enhanced, giving users the benefits of life-like 3D graphics.

Conclusion

The introduction of DirectX 10.1 into the market brings a new level of 3D experience to the user. Image quality improvements derived from the implementation of AA and global illumination make photo-realistic graphics available on all systems which feature S3 Graphics' Chrome 400/500 Series DX10.1-compliant graphics processors. Additional improvements in system, Windows Vista Aero, and 3D rendering application performance, produce a noticeably enhanced user experience. Moving forward, possible extensions to the Windows Vista SP1 release, such as advanced display management capabilities, improved imaging processing through texture mapping and shading, interactive animated windows, and support for ultra higher resolution digital (high DPI) monitors, will be supported by S3 Graphics through its product offerings.