Zone Rendering

Whitepaper

May 2002

Document Number 298587-001

Contents

Figures

Revision History

Rev. No.	Description	Rev. Date
1.0	Initial release	May 2002

1 Introduction

With the release of the next-generation Intel® GMCH, the Intel® 845G chipset, a brand-new graphics engine is introduced. One of the most significant new features of this graphics engine is Zone Rendering. This paper introduces the zone rendering capabilities of the 845G chipset, outlines the advantages offered by zone rendering, and discusses the implications of zone rendering to software application developers.

This page is intentionally left blank.

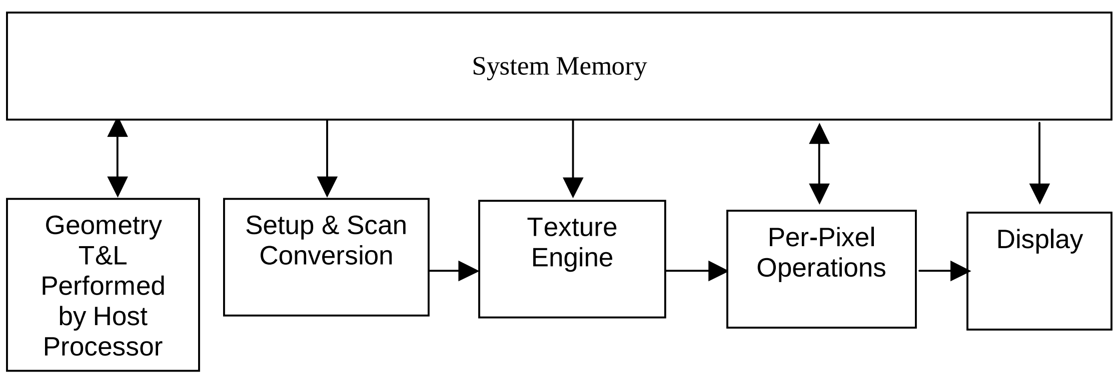
2 Overview

This section provides an overview of the traditional 3D graphics rendering engine as well as a zone rendering graphics engine.

2.1 Conventional 3D Rendering

The basic overview of the traditional 3D graphics pipeline is given in Figure 1 below. In this traditional 3D pipeline, each polygon is processed in the order that it is received by the hardware. Hence, the 3D pipeline displayed in the figure below is referred to as “immediate mode” 3D rendering, meaning that as each polygon is sent to the hardware, it is immediately processed through the pipeline and rendered into the frame buffer as appropriate.

Figure 1. Conventional 3D Graphics Pipeline



2.1.1 Geometry Transform and Lighting Performed by Host Processor

In the initial stage of the pipeline, all of the vertices progress through some number of matrix transformations in order to be converted into 3D world coordinates. The diffuse and specular lighting components are then calculated at each vertex based on its position in the 3D world. These vertices are then finally transformed into 2D screen coordinates. The host processor is used to handle the transform and lighting stage of the pipeline. The vertices may be read from and written to memory multiple times depending on the implementation.

2.1.2 Setup and Scan Conversion

Following the transform and lighting stage the pixel-level specifics of each polygon are calculated. Initially the setup engine receives as input the vertices of each polygon, and then calculates the information needed to convert that polygon into its associated pixels. Also portions of polygon that do not appear on the screen are clipped or culled by the setup engine. The scan converter then receives the information from the setup engine and identifies all the pixels that are being affected by the polygon being rendered.

2.1.3 Texture Engine

Once the associated pixels for each polygon have been calculated, the texture engine is then responsible for retrieving any texture elements, or texels, associated with each pixel. Any required texel operations, such as colorkey matching, color format conversion, or filtering, are then performed by the texture engine. As there are potentially multiple texel fetches for every pixel of every polygon, memory access for texels typically account for a large percentage of the graphics memory bandwidth usage.

2.1.4 Per-Pixel Operations

Once a pixel is fully textured it's almost ready to be put into the frame buffer. First, however, the z-buffer is read to determine whether the pixel will even be visible. If not visible, (meaning that another pixel would be covering it up in the frame buffer), then the pixel is thrown out and not used. If the pixel is nearer than what is currently in the frame buffer, or needs to be blended with the current frame buffer contents, then both the frame buffer and z-buffer are updated with the new pixels information. For every pixel, there is a required z-buffer read and potential z-buffer and frame buffer write in memory. Texel, z-buffer, and frame buffer access in memory will typically account for a large percentage of the memory bandwidth usage.

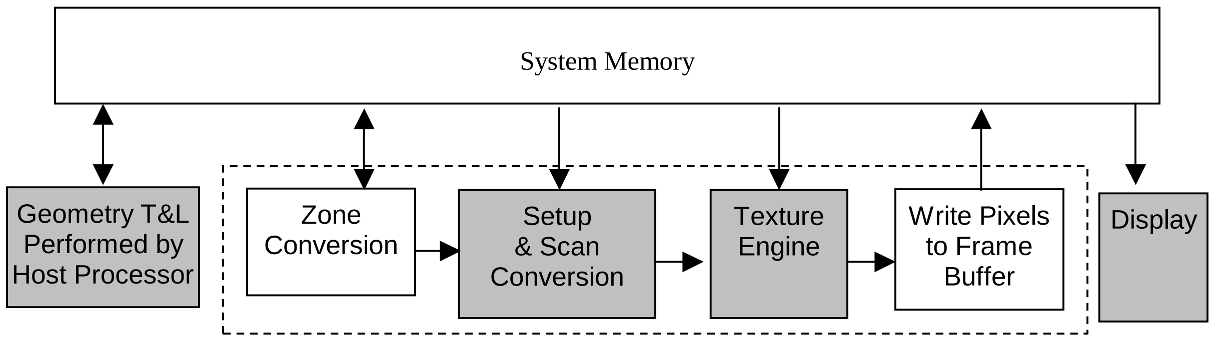
2.2 Zone Rendering

2.2.1 Concept of Zone Rendering

With immediate mode rendering, discussed in the previous section, any portion of the frame buffer may be drawn anytime during a single frame. Zone rendering, however, divides the frame buffer up into a number of rectangular zones, and then renders all of the pixels within a single zone before proceeding on to the next zone. The advantages of this method will be discussed in the next section.

The basic overview of the zone rendering 3D graphics pipeline featured in the Intel 845G chipset is given in Figure 2 below.

Figure 2: Zone Rendering 3D Graphics Pipeline



intel

2.2.2 Zone Conversion

As can be seen in Figure 2 above, one major difference between the zone rendering capable 3D graphics pipeline and the conventional 3D graphics pipeline is the inclusion of the zone conversion stage of the pipeline.

In the zone conversion stage, triangles are sorted into memory by zone. Hence all triangles associated with a respective zone are placed into the same zone in memory. Then, these zones are processed one at a time by the remainder of the 3D graphics pipeline, causing only a single zone to be rendered at a time.

2.2.3 Per-Pixel Operations

The second major difference between the zone rendering 3D graphics pipeline and the conventional pipeline occurs as the pixels are written out to the frame buffer. Because only a single zone is being rendered at a time, all depth calculations can be done on-chip, eliminating the need for a separate depth buffer. Likewise all pixel blend operations are done on-chip. Hence each pixel of a particular zone will only be written once. The need for depth buffer reads and writes, as well as frame buffer reads, has been eliminated.

This page is intentionally left blank.

intel®

3 Advantages of a Zone Rendering 3D Graphics Pipeline

The 845G chipset represents a class of chipset with integrated graphics employing a unified memory architecture (UMA). Integrated graphics chipsets that employ UMA, share system memory resources with the rest of the system, utilizing system memory resources for graphics.

The 845G chipset implements 32 bits per pixel (bpp) graphics allowing for true transparent and translucent surfaces – more realistic shadow effects, smoke effects, clouds, etc. Delivery of these capabilities puts a lot of demands on the memory subsystem.

Zone rendering is a unique mechanism that addresses memory bandwidth limitations by reducing the required memory bandwidth for graphics. By processing only a single zone of the frame buffer at a time, the use of the render cache is highly optimized. The high level of spatial coherency in the render cache for any individual zone allows on-chip access to all of the significant color and depth information for a particular frame. As stated previously, this eliminates the need for depth buffer reads and writes, as well as color buffer reads. By eliminating the need for these memory accesses, zone rendering reduces the maximum theoretical required graphics memory bandwidth.

The improved memory bandwidth efficiency provides some secondary memory benefits as well. Lower memory bandwidth requirements simplify the memory interface and memory technology used by a zone rendering graphics device. This offers a potentially large saving in graphics memory cost.

One final benefit of the zone rendering architecture is that the fill rate requirements placed on the graphics device is not reduced. Because pixels are not overdrawn in the frame buffer, the fill rate that is required to draw any scene is equal to the total number of pixels in the scene (depth complexity = 1). By contrast, traditional 3D graphics architecture may have to redraw each pixel in a scene anywhere that triangles overlap. This means that if the average depth complexity in a scene is 3, meaning each pixel must be drawn 3 times, the fill rate required by a conventional 3D graphics architecture is 3 times the number of pixels in the scene. Thus the fill rate requirements for zone rendering will always be the same as the current resolutions, whereas the fill rate requirements for a conventional architecture will increase as a function of the depth complexity.

The scene below was rendered on a 845G chipset based-system running 3D WinBench* 2000. This particular scene has a depth complexity of 2.72. Thus, the memory bandwidth required to render this scene with a conventional 3D graphics pipeline is 2.72 times more than with a Zone Rendering 3D graphics pipeline.

this scene with a conventional 3D graphics pipeline is 2.72 times more than with a Zone Rendering 3D graphics pipeline.

Figure 3: 3D WinBench* 2000 Scene with Average Depth Complexity of 2.72

4 Software Considerations

Zone Rendering is designed to be completely transparent from an application software point of view. The 845G chipset graphics drivers will feature both a complete OpenGL* ICD and full support for Microsoft DirectDraw* and Direct3D* APIs. Furthermore, Zone Rendering will be handled transparently within the driver. Therefore, any application written for OpenGL or the Microsoft DirectX* graphics APIs will run on the 845G chipset and make use of 845G chipset hardware capabilities wherever possible.

This page is intentionally left blank.

5 Summary

5.1 What is Zone Rendering?

Zone Rendering is a process in which the screen is divided into several zones. Each zone is completely cached and rendered on chip before being written to the frame buffer.

5.2 What are the benefits of Zone Rendering?

Zone Rendering has several benefits:

- Reduction in Depth and Color Bandwidth associated with conventional rendering
- Increased memory efficiency via better localization of data
- Increased on-chip processing speed due to decreased wait time for data
- Reduced power as a result of decreased memory bandwidth
- Increased effective pixel fill rates
- Increased headroom for larger resolution and color depth

5.3 Conclusions

The Zone Rendering architecture featured in the next-generation Intel GMCH, the Intel 845G chipset, provides improved performance above conventional 3D architectures through efficient memory bandwidth usage and optimal utilization of the render cache. The 845G chipset drivers provide software support for this new architecture for both Direct3D and OpenGL, allowing both existing and future 3D applications using these APIs to utilize Zone Rendering without the need for modification.