TEXTURE COMPRESSION ON QUANTUM3D PCIGs

Methods and Benefits

Author: Brian Dickens

Date: 7/24/2002

INTRODUCTION

In the quest to achieve greater visual realism, visual simulation applications employ 3D content databases that make use of many high-resolution texture maps. These textures increase the realism of 3D objects by using photo-specific texture maps and super high-resolution geo-specific satellite imagery mapped to the terrain. In doing so, these applications many times use texture maps whose total size exceeds the amount texture memory available in standard graphics subsystems. In some cases, these databases and textures even exceed PCIG system memory size. In the case of databases that cover large geographic portions of the world with satellite imagery, the data can represent many gigabytes of information.

Physical memory and system bandwidths limit applications that use only conventional non-compressed textures. Until recently, the only method of dealing with such large amounts of data is a database and texture paging strategy that swaps textures into graphics memory when needed for rendering, and makes them available for discarding when not needed for rendering. Unfortunately conventional texture storage formats are not efficient methods for storing textures on disk, in memory or for efficient transfer into of system and graphics memory.

Overcoming the limitations of these envelope-pushing applications has driven the implementation of hardware supported texture compression capabilities. Texture compression is a method of reducing the overall storage requirements for textured rendering; this means more texture may be stored in a given amount of texture memory. The result is the ability to use more textures with higher accuracy to increase rendering quality. Texture compression also reduces the amount of bandwidth required to transfer and process textures for rendering. This increases fill-rate and frame rate creating a more realistic, responsive and accurate simulation.

WHAT IS TEXTURE COMPRESSION?

Although there are many texture compression technologies, the two adopted by the leading graphics chip manufacturers for use in graphics applications are FXT1 and DXTn. FXT1 texture compression was the first texture compression format available to application developers. It was developed by 3dfx and is now an open source project. Slightly later, S3 developed a nearly identical texture compression algorithm called S3 Texture Compression (S3TC). Microsoft licensed the S3TC technology and incorporated it into DXTn texture compression; this is now the mechanism for compressing textures in DirectX. Texture compression is also available to OpenGL applications via the `GL_ARB_texture_compression` extension; the formats offered are implementation dependent. Many applications programs use DXTn texture compression and many of these are OpenGL based applications. Current PCIG products from Quantum3D support either FXT1 or DXTn texture compression.

Texture compression reduces the amount of data required to store texture map images. Both FXT1 and DXTn result in very little loss of texture image quality when compared to the original uncompressed image as shown in **Figure 1** and **Figure 2** on page 3. FXT1 and DXTn use a similar strategy for compressing the texture, this operation is termed encoding. The Quantum3D graphic subsystem performs compressed texture decoding automatically.

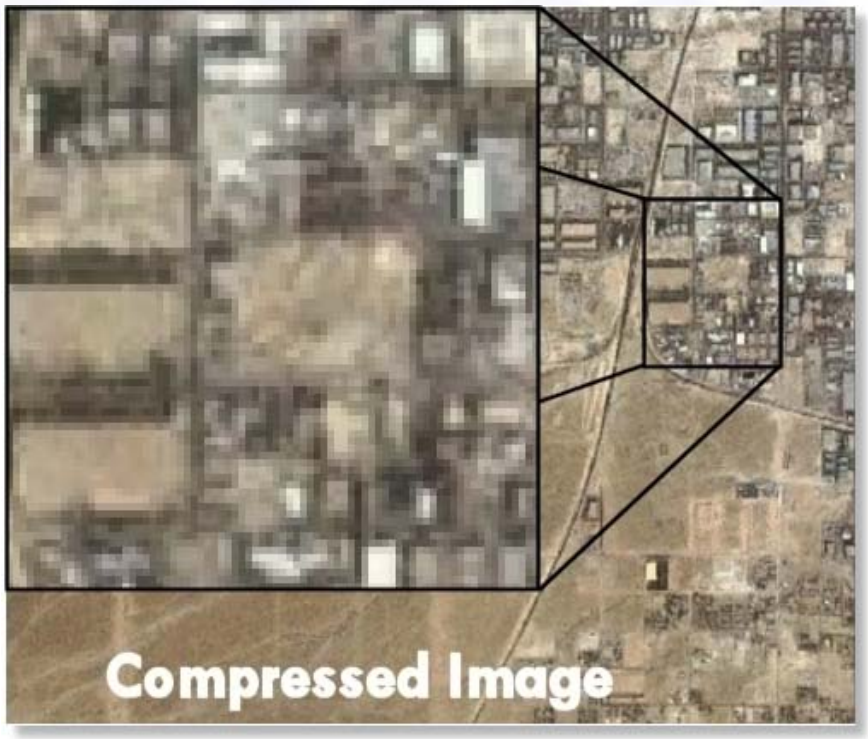

Figure 1: FXT1 compressed texture, image data courtesy of Terrain Experts, Inc.

Figure 2: Non-compressed texture, image data courtesy of Terrain Experts, Inc.

Both FXT1 and DXTn use similar schemes for encoding. The texture is broken into blocks of 4x4 texels, resulting in 16 texels per block. The 16 texels are examined, and two colors are chosen, which are typically the extremes of the 16 texel colors. The encoding/decoding process calculates two additional colors by interpolating between the two first colors; these two interpolated colors are not stored in the compressed format. The two stored color values and the two interpolated color values define the colors in the 4x4 texel block. Because there are only four colors for each 4x4 block, each color in the block can be represented using only 2 bits. The total storage required for the 4x4 texel block is 64 bits (16x2 bit colors and 2x16 bit colors), resulting in an average of 4 bits per texel which is a compression ratio of 6:1 **Figure 3** is a diagram of the storage layout for a 4x4 block of a compressed texture.

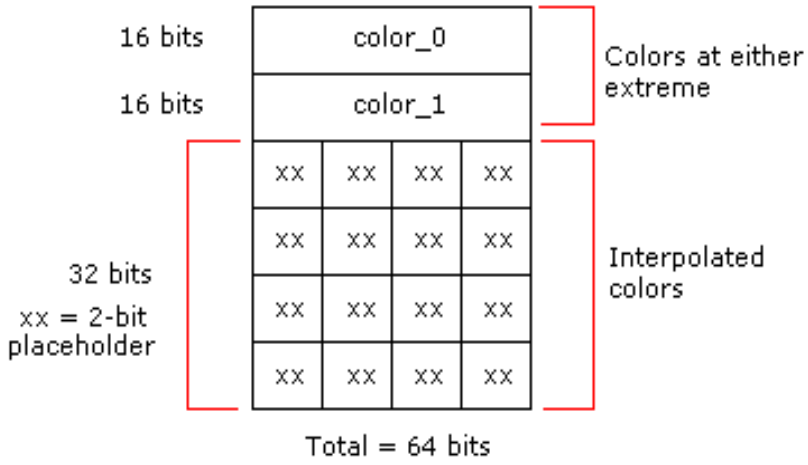

Figure 3: Basic DXTn pixel block layout, courtesy of Microsoft MSDN

The FXT1 format offers four different methods for compression, some of which results in slightly lower compression ratios, but can result in better image quality. One of these algorithms is *CC_MIXED* and is the equivalent of DXTn compression. Other compression algorithms supported in FXT1 are CC_HI, which is used for better spatial resolution, CC_CHROMA (best for complex color areas), and CC_ALPHA (best control for complex alpha values).

Once the texture has been encoded, it is stored to disk and may be used in an application just like any conventional texture. Supporting compressed texture in an application is very straightforward since hardware in the graphics subsystem performs the decoding.

THE BENEFITS OF TEXTURE COMPRESSION

COTS tools support

A wide variety of products and tools support FXT1 texture compression; some of the COTS product offerings are Quantum3D OpenGVS, Terrain Experts, Inc TerraVista and CG2 Vtree and Mantis. In addition, Quantum3D supplies conversion utilities used to convert most popular image formats (both compressed and uncompressed texture image file formats).

Increased effective amount of texture memory

In order to understand the drastic effect of using texture compression, consider the case of a 24 bit 2048x2048, 3-component texture. Using a format that does not use compression the number of bytes required to store the texture is 2048x2048x24/8 Bytes or 12 MB For a 24-bit texture, the compression ratio is 6:1, and this12 MB texture reduces to 2 MB -- a significant improvement over conventional storage formats. Compression of 32 bit, 4-component textures (RGBA) is 8:1. **Figure 4** illustrates the savings when using a texture compression method.

X Res	Y Res	8bit	16bit	24bit	32bit	FXT1/DXTn
64	64	4 KB	8KB	12 KB	16 KB	2 KB
128	128	16 KB	32 KB	48 KB	64 KB	8 KB
256	256	64 KB	128 KB	192 KB	256 KB	32 KB
512	512	256 KB	512 KB	768 KB	1 MB	128 KB
1024	1024	1 MB	2 MB	3 MB	4 MB	512 KB
2048	2048	4 MB	8 MB	12 MB	16 MB	2 MB

Figure 4: Storage Requirements for textures of varying color depth and size

(Please note that when calculating texture memory allocation you must add 1/3 of the texture storage requirements in order to compensate for MIPMAP storage requirements. The calculations above as well as those in Figure 4 do not contain the extra 1/3 memory impact.) Reducing the amount of memory needed to store textures, increases the amount of effective texture memory available to the application. For example, an application that utilizes texture compression of RGB textures will effectively have 6 times more texture memory available to the application.

Applications may use more textures with increased resolution

Considering the case of a 256x256 RGBA image a conventional texture will require 256 KB of memory while a compressed texture only requires 32 KB. The result is that the application could use a total of 8 256x256 compressed textures in the same 256KB, which would significantly increase the visual realism of the 3D scene. If increased resolution is desired the same 256 KB of could be used to store a much higher resolution 1024x512 texture. The increase in effective texture memory also opens the door for using more texture techniques such as detail texture, texture light maps and other advanced texture techniques.

Decreased bandwidth requirement increases fill-rate

The amount of texture memory bandwidth available helps determine ill-rate performance. Using textures in a compressed format allows for more efficient graphics subsystem operations. Using the example of a 32-bit texture compressed to a 4-bit format, the compressed texture is 8 times more efficient to transfer within the graphics system when compared to a conventional texture. The decreased requirement for texture memory bandwidth results in higher fill-rates and frame rates.

Decreased bandwidth improves real-time database paging performance

Textures used in large area database paging applications must travel across numerous interfaces during the rendering process. Typically, textures are stored on a hard disk, because of this; textures are transferred over a PCI bus into system memory. Once the textures are in system memory, they are then transferred over the PCI or AGP bus to the graphics subsystem. Because database paging can involve each of these transfer operations in real-time during each frame of the simulation, paging will benefit significantly from using texture compression. In the case of real-time database paging, each of these transfers receives a significant improvement in performance. Because of the decreased bandwidth needed for each texture, texture paging algorithm's transfer rates are up to 6 times more efficient when using compressed textures (compared to 24-bit RGB uncompressed textures). **Figure 5** shows where texture compression improves efficiency of transfer operations.

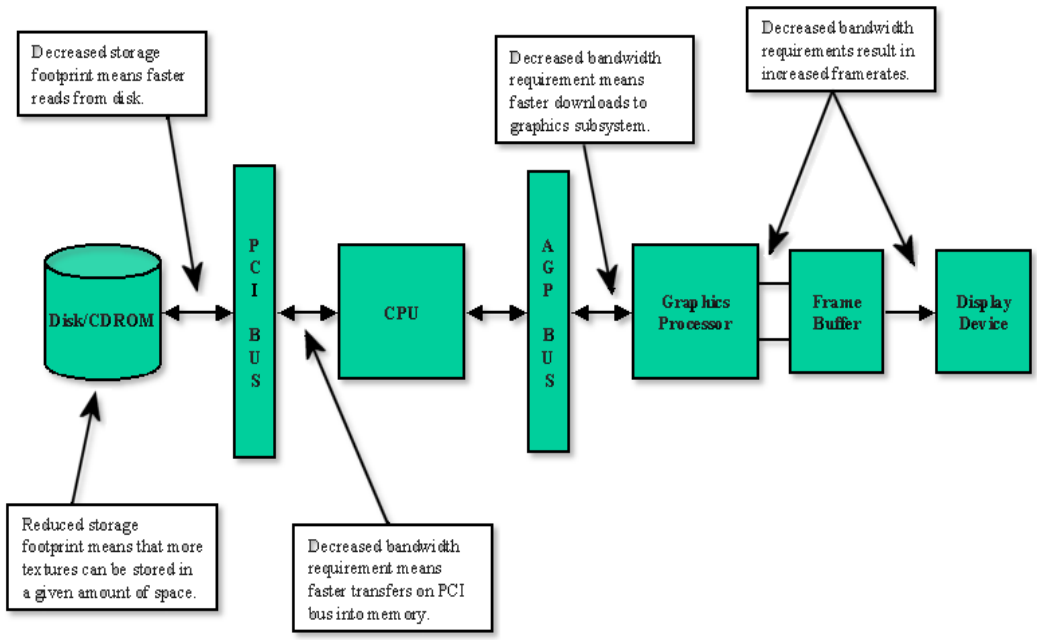

Figure 5: System diagram, improvements in texture transfer rates.

Texture compression reduces disk storage requirements

Another benefit of texture compression is reduced disk storage requirements. This is especially important for the case of large area databases. Consider an example of a database that covers an area of 1000 Km2. If the database uses texture maps where each texel represents 5m, then the resulting texture (or set of textures) would be 200,000 x 200,000 texels. If this were a conventional texture, it would require 37.25 GB of storage (on disk and in memory). The same texture compressed would only require 6.2 GB of storage.

CONCLUSION

Texture compression has a wealth of benefits to applications that make use of this technology. Using texture compression gives an application effective texture memory of 288 Megatexels when using 4 bit compressed textures (assuming that there is 64 MB graphics memory and a 1280x1024 frame buffer). Increased effective texture memory means that the application is able to use more, higher resolution texture maps for increased accuracy and realism. In addition, the application is more flexible at using these textures for more advanced techniques such as detail textures and lighting maps. Perhaps the most significant improvement in overall performance appears in large area database applications that make heavy use of texture and terrain database paging. For these applications, texture compression improves disk storage requirements, paging efficiency as well as increases fill-rate.

Because there are COTS software packages available that support texture compression, visual simulation application developers can make use of the texture compression tools appropriate for their task. For the OpenGL or scene manager developer, tools and APIs are available that make developing applications that make use of compressed textures easy. People using turnkey solutions from PCIG vendors like Quantum3D or CG2 will also benefit since these are standard features in these COTS products. Texture compression is now a standard offering on Quantum3D PCIG systems making use of this technology will not only make applications more realistic, but will also improve performance.

GLOSSARY

4-component textures: Textures that contain information about Red, Green, Blue and Alpha (RGBA) color components. These are typically 32-bit textures when stored using conventional texture storage methods. There are also 16-bit formats for 4-component texture, for example, RGB each use 5 bits and Alpha uses only a single bit.

COTS: Commercial off-the-shelf

Frame rate: The frequency at which a graphics system and/or application redraws the screen. Typically, frame rate is measured in milliseconds or Hz. For example, an application drawing the screen 60 times per second has a frame rate of 60 Hz or 16.67ms. When measuring pure graphics performance it is important to disable synchronization to the monitor vertical refresh frequency so that you are measuring drawing performance and not fixed to the refresh rate of the monitor. This will introduce some artifacts, but will yield accurate drawing performance numbers.

Geo-specific texture maps: Texture maps derived from photographs of a specific geographic area of the world. The original source of these textures is typically satellite imagery. After acquiring this imagery image processing is applied to the images before they are suitable for mapping to the database. Texture compression is performed during the mapping process of database generation.

Open source: Open source promotes software reliability and quality by supporting independent peer review and rapid evolution of source code. To be OSI certified, the software must be distributed under a license that guarantees the right to read, redistribute, modify, and use the software freely. For a more complete definition, please refer to http://www.opensource.org/docs/definition_plain.html.

Photo-specific texture maps: Texture maps derived from digital or scanned photographs of real objects. These are then mapped to 3D objects in a database. Texture compression is typically performed after the mapping process.

REFERENCES

Microsoft MSDN

3dfx, Inc, FXT1 Texture Compression Technology White Paper

