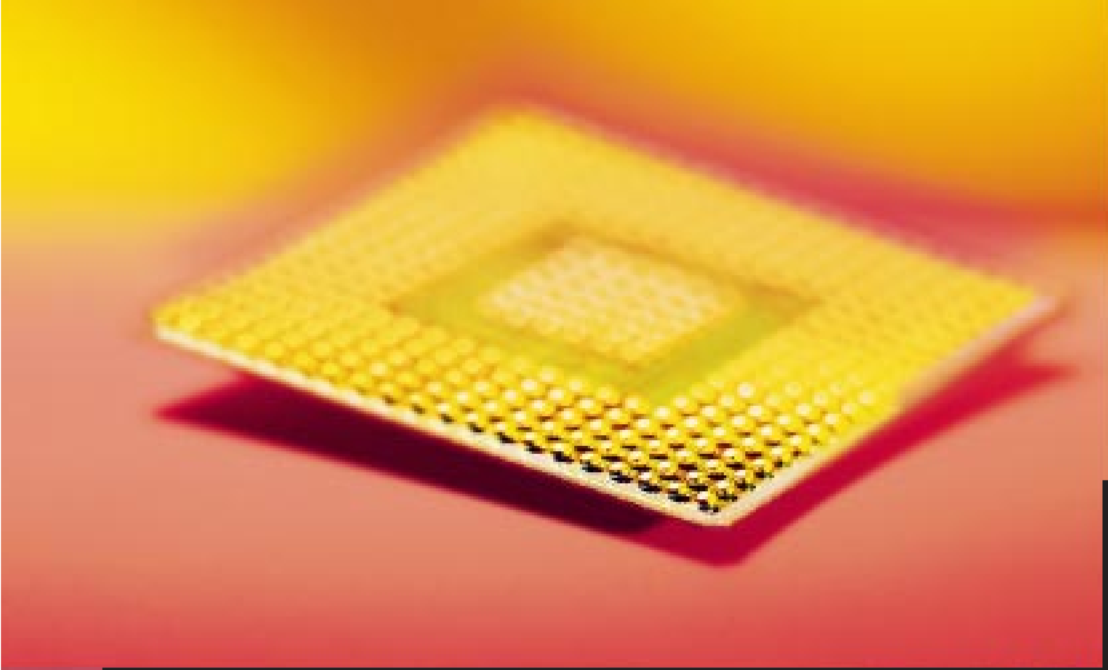
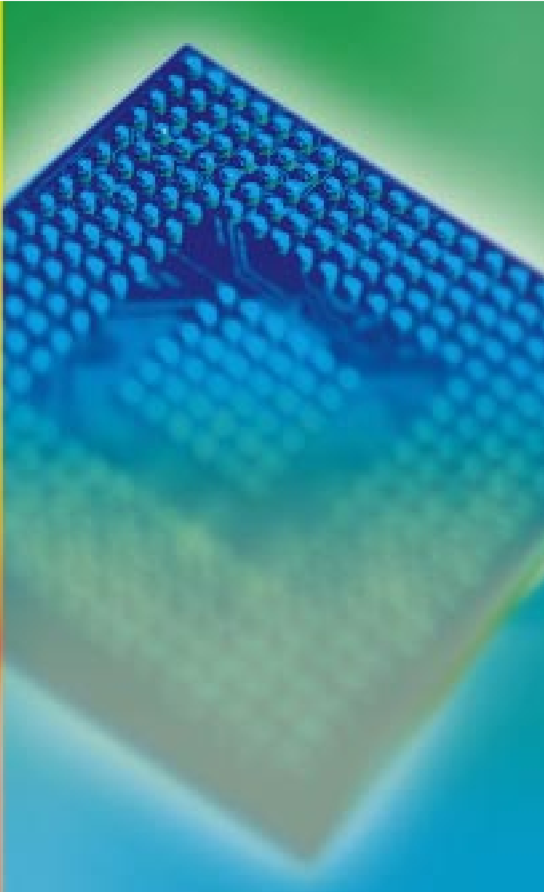
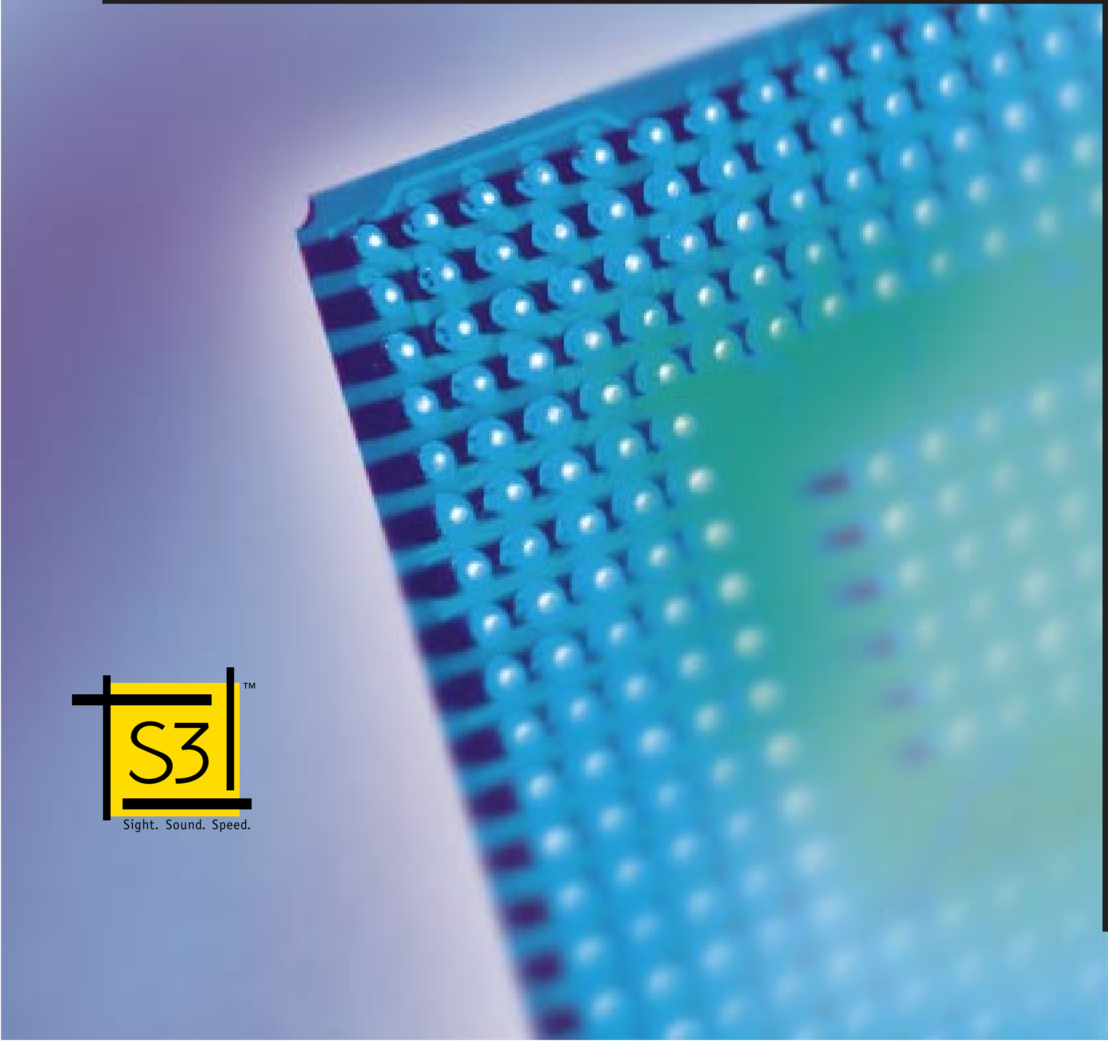
S3TC™ DirectX 6.0 Standard Texture Compression





SAVAGE 3d



S3™

Sight. Sound. Speed.

Introduction

Texture maps are bit map images that are applied to 3D objects. They are used to add realistic surface detail without increasing the complexity of the geometry in a 3D scene. Textures can be anything from wood grain or marble patterns to complex pictures of people, buildings, trees, etc. To simulate real life scenes, it is desirable to have access to a large number of detailed textures. However, this places significant demands on system or graphics memory (depending on where the textures are stored), forcing application developers to use fewer and less detailed textures in order to match a limited amount of memory storage and bandwidth.

Accelerated Graphics Port (AGP) has made it possible to access textures directly from system memory increasing overall available storage. However, AGP and the system memory interface are shared resources. Besides textures, AGP is also used for passing geometry data, while system memory is used to store and run the operating system and applications. So, it should not be assumed that all system memory bandwidth will be available for reading texture data. Given this, making the most of the bandwidth that is available is critical to a graphics subsystem optimized for AGP texturing.

S3TC™ texture compression helps in these two main areas by allowing more and larger (more detailed) textures to be stored in the same memory area and, at the same time, by significantly decreasing the bandwidth required to access them.

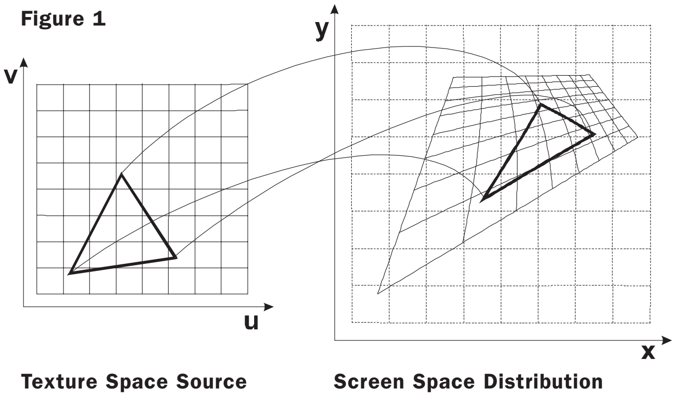
Advantages of Texture Compression

Decreased memory size/bandwidth requirements = increased performance & better image quality

The obvious benefit that texture compression provides is that a given amount of texture data can be stored using significantly less memory. This is most critical when texturing out of the local frame buffer. Equally important is the fact that the memory and bus bandwidth required to read textures is greatly reduced, translating to much improved performance over AGP.

System memory and AGP bus bandwidths are finite resources. With compression, the accelerator can take better advantage of these resources.

Figure 1



Larger and More Numerous Textures

Texture compression can allow for larger textures. While smaller, less detailed texture maps typically result in surfaces that look blurry or blocky, larger textures let the application provide more surface detail. Texture compression can also allow for a greater variety of textures to be used at any given time, permitting more varied scenes. When texturing out of local frame buffer memory, compression may free up enough memory to increase the display resolution or to perform triple buffering. A higher resolution display provides a smoother, more detailed look, while triple buffering can improve performance by allowing the rendering engine to start on a new scene without waiting for the display's vertical sync. The use of triple buffering can result in a significant increase in frame rate (typically 30%).

Mip-mapping

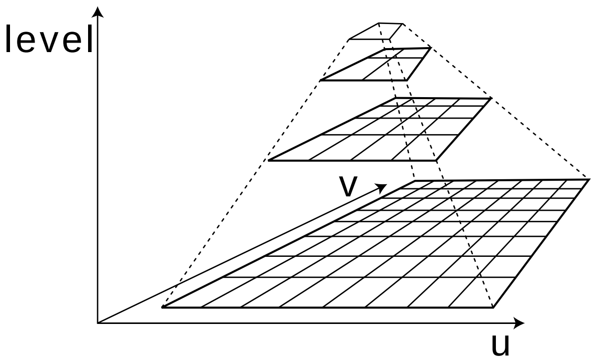
The extra memory available with compressed textures allows for the use of Mip-maps even with the added memory storage required (30%) over the base texture map level. Mip-maps help to reduce aliasing artifacts visible on textured surfaces that span significant distances. Without Mip-maps, a pixel on an object far away may be associated with several texels from the original texture map. Low-pass filtering is used to retain the information, while not introducing unwanted artifacts (shimmering, crawling pixels). Real time filtering is expensive, so staged Mip-map levels can be used that are pre-computed filtered images, dramatically lowering the filtering complexity. Figure 2, below shows four simple Mip-map levels, each one quarter the size of the last.

The use of Mip-maps can also improve performance. Mip-maps help keep memory accesses sequential and allow longer bursts. Otherwise, as the object moves farther away, pixels are sampled less frequently causing memory accesses to become more random and increasing the likelihood of shorter (less efficient) bursts and page break penalties. For more information on Mip-mapping, please see S3's recently released white paper entitled *Trilinear Filtering: A Better Way to Texture 3D Objects*.

S3TC™ Overview

The S3 texture compression scheme (S3TC™) was developed specifically for texture maps. Textures are compressed to a fixed size equal to four bits per telel for opaque textures (or textures with simple transparency effects) or eight bits per texel for complex transparent textures. The quality of the textures even after compression is very good. (Figure 3)

Figure 2



Brief Description of S3TC™

S3TC™ breaks a texture map into 4 x 4 blocks of texels. For opaque texture maps, each of these texels is represented by two bits in a bitmap, for a total of 32 bits. In addition to the bitmap, each block also has two representative 16 bit colors in RGB565 format associated with it. These two explicitly encoded colors, plus two additional colors that are derived by uniformly interpolating the explicitly encoded colors, form a four color lookup table. This lookup table is used to determine the actual color at any texel in the block. In total, the 16 texels are encoded using 64 bits, or an average of four bits per texel. Simple transparencies (single bit or color keys) are performed by reserving one of the four colors to indicate that the texel is transparent and the third color is just the average of the explicitly encoded colors. The order of the two encoded colors determines whether the block is completely opaque or whether it has transparent texels. When the block is determined to be one that has transparent texels, the fourth bit encoding (11) indicates a transparent texel, while the other derived value is just the average of the two encoded color values.

Figure 4 shows the original block of texels along side the color look-up table and 2-bit indexed matrix that make up the final compressed block. As can be seen, each texel in the compressed block was assigned one of the four colors from the look-up table closest to that of the original block.

S3TC™ also provides an additional 64 bits to encode more sophisticated transparency effects if desired. Two different mechanisms are provided but not described here (consult the DirectX 6.0 documentation for full details).

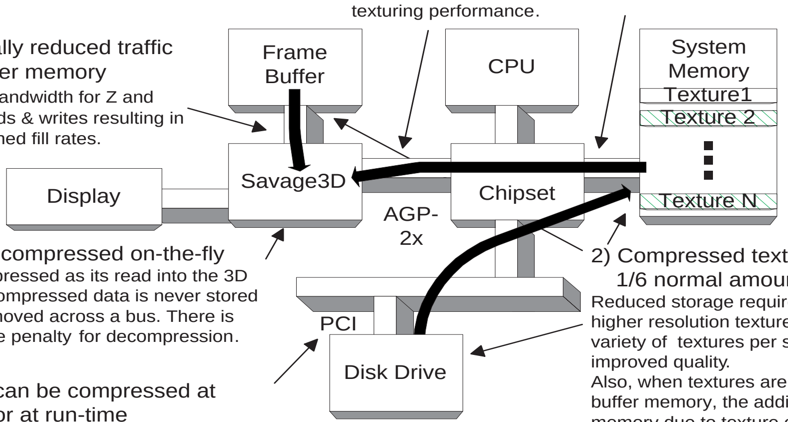
Simple Decoder

Decoding blocks compressed in S3TC™ format is straightforward. A two-bit index is assigned to each of the 16 texels. A four color lookup table is then used to determine which 16-bit color value should be used for each texel. The decoder requires relatively little logic, which can be operated at very high speeds and replicated to allow parallel decoding for very high performance solutions. The simplicity of the implementation will result in its universal adoption throughout the graphics industry.

Figure 3

3) Reduced traffic on AGP bus and in system memory
Texture read bandwidth equivalent to AGP-12x yields superior AGP texturing performance.

4) Dramatically reduced traffic in frame buffer memory
Leaves extra bandwidth for Z and destination reads & writes resulting in superior sustained fill rates.



5) Data is decompressed on-the-fly
Data is decompressed as its read into the 3D engine, so decompressed data is never stored in memory or moved across a bus. There is no performance penalty for decompression.

2) Compressed textures require 1/6 normal amount of memory
Reduced storage requirements allow for higher resolution textures and a greater variety of textures per scene for improved quality.
Also, when textures are stored in frame buffer memory, the additional available memory due to texture compression can be used to perform triple buffering for improved performance.

1) Textures can be compressed at author-time or at run-time
Normally, 3D applications ship with compressed textures. Compression can also be done at run-time using the DX6 API as textures are loaded into memory (AGP or frame buffer). Typically, compressed images suffer no perceptible loss of image quality.

S3 Texture Compression (S3TC™) is the DirectX 6.0 standard

Image Quality Compared to Other Methods

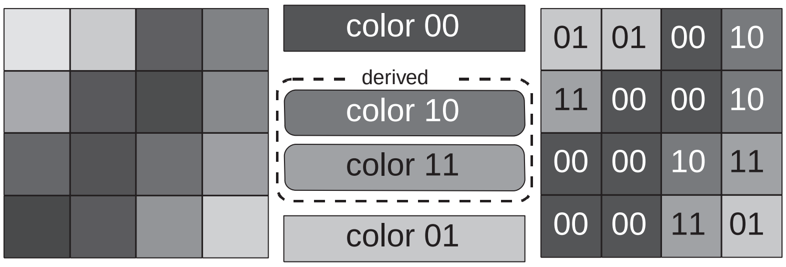
While other compression techniques exist, many “simple” schemes with inexpensive decoders have inferior quality, or a smaller compression factor, or both. Vector quantization techniques produce inferior quality and, because they rely on using a code book, the decoder needs to do two memory accesses to decode each texel block. The first memory read is needed to read the block code - the index into the code book. The second access is needed to look up the texel values associated with the block code, by using the index to look that up in the code book. The code book could theoretically be stored on-chip to avoid the second memory reference, but since this would be costly, it’s not done. Plus, even if the code book were stored on-chip, it would need to be loaded and that would have a significant performance impact as well. For these reasons, VQ compression is not only much slower than S3TC™, but – equally important – the quality is worse.

Palletizing is one well-known form of vector quantization. Palletized texture image quality suffers when a large variety of colors are used. While a palletized image is limited to 256 colors for the whole image, S3TC™ does not impose a limit on the overall number of colors available. Palletized textures also require that a new palette be downloaded for each new texture (the palette being equivalent to a code book).

Other “standard” compression techniques like JPEG (TREC) are expensive to implement and the quality of many images will still look better with S3TC™. The cost is not just the large amount of logic required to do the decoding. Because of the large latency incurred in decoding a block of TREC or JPEG texture, a significant amount of latency-compensating buffering is required as well, further increasing the cost and complexity of the logic around the decoder. Also, DCT-based compression introduces low frequency artifacts - ringing or blocky artifacts - that are not easily removed with standard texture filtering algorithms like trilinear filtering. Texture images with smooth gradients will even look better with S3TC™ than uncompressed 16-bit RGB565.

Tools are available on our web site *(http://www.s3.com/s3tc)* for compressing images in the S3TC™ format. This will allow you to compress your own images and compare them side by side with the originals. We believe you will find it difficult to see a difference.

Figure 4



01	01	00	10
11	00	00	10
00	00	10	11
00	00	11	01

Original Block **Look-up Table** **Compressed Block**

DirectX Standard

S3TC™ is the basis for the compressed texture formats used in DirectX 6.0. Microsoft has licensed S3TC™ from S3 and is making it available to both the ISV and IHV communities through the forthcoming DirectX 6.0 API. Software developers can expect a broad level of support for texture maps compressed in this format. Coupled with Microsoft's support for this standard is the fact that it is also easy to build into any graphics hardware. So expect it to be universally available before too long. Software developers should ship with all of their textures compressed, since a DirectX API will always be available to decompress these in software when the target hardware does not support S3TC™. By shipping with compressed textures, ISV's can use the highest quality S3TC™ encoder offline, and not incur any start-up penalty due to slow texture loads from compressing textures on-the-fly. Shipping with compressed textures should also ease any storage problems on the CD-ROM or diskettes used to ship the game.

Relative Texture Quantities.

To give an idea of how S3TC™ can help to improve the quality of 3D applications, the table below (Figure 5) shows the number of textures of differing sizes that can be stored in a limited memory area. For this example, 8MB of texture storage is available. The numbers below assume Mip-mapping. Color formats for uncompressed and palletized textures are 24-bits and 8-bits per texel respectively.

Another way to look at it: In an 8MB frame buffer, an application running at 800x600x16 double-buffered with a 16-bit z-buffer will have 5.25MBs of memory left over for texture storage. If S3TC™ is used, those 5.25MBs will be able to store the equivalent of 31.5MBs of texture. Furthermore, you could switch to triple buffering and still have the equivalent of 26MBs of texture storage. Or you could quadruple the resolution of all the textures, convert them all to Mip-maps and still have the equivalent of 3.5MBs of texture storage left over.

With S3TC™ compressed textures, applications can use a broader variety of textures, use higher resolution textures, increase performance by cutting the read bandwidth required, increase performance by switching to triple buffering and Mip-mapping, or use a combination of all of these. This will increase the realism of computer generated 3D scenes significantly.

Figure 5

Texture Size	Uncompressed	Palletized	S3TC Compression
64 x 64	512	1536	3072
256 x 256	32	96	192
1024 x 1024	2	6	12

With S3TC™ compressed textures, the application can use a broader variety of textures, higher resolution textures or both. This will increase the realism of computer generated 3D scenes significantly.

Conclusion

S3TC™ saves memory storage space and lowers system bandwidth requirements. S3TC™ provides a compression scheme that delivers high image quality with a simple decoder, easily implemented in hardware. Here is a summary of the benefits:

- Adopted by Microsoft as a DirectX 6.0 industry standard, it's easy for software developers to use
- Reduction in memory storage allows for more and larger textures yielding better image quality
- Reduced texture read bandwidth requirements improves overall system performance
- Good compression ratio of up to 6:1
- Block based, fixed compression rate with random access – easily implemented in hardware.
- Decompression in hardware is simple and fast
- Excellent quality – often better than palletized, JPEG or uncompressed RGB 5:6:5
- Simple transparent pixels can be encoded at no extra cost
- Complex transparencies and smooth alpha are also supported
- High quality, simple to use tools for encoding are available

SAVAGE
3d

S3 Incorporated
2801 Mission College Boulevard
P.O. Box 58058
Santa Clara, California
95052-8058
408.588.8000 *phone*
408.980.5444 *fax*
www.s3.com *website*

S3™
Sight. Sound. Speed.