Texture Compression

Philipp Klaus Krause

November 24, 2007

Contents

1 Abstract

Texture compression is widely used in computer graphics today. I give an overview of the field starting by the reasons that make texture compression desirable, the special requirements on texture compression systems, which distinguish them from general image compression systems.

Today's three most important texture compression systems are extensively presented and compared.

Other systems of some significance are mentioned with some information about them.

Appendix A shows how texture compression interacts with OpenGL.

2 Introduction

2.1 Motivation

In computer graphics achieving high visual quality typically requires high-resolution textures. However the desire for increasing texture resolution conflicts with the limited amount of graphics memory available.

Memory bandwidth is the most important aspect of graphics system performance today. Most bandwidth is consumed by texture accesses, not accesses to vertex data or framebuffer updates. One reason for this is texture filtering, which increases visual quality: The common trilinear texture filtering needs to access 8 texels during each texture read.

Increasing memory bandwidth is expensive. Increasing memory clock frequency requires more expensive and power-consuming memory chips. Increasing the number of data lines often results in increasing the number of memory chips increasing cost and power consumption, too. Especially for embedded systems increasing the memory bandwidth may not be an option.

Texture compression can help to achieve higher graphics quality with given memory and bandwidth or reduce memory and bandwidth consumption without degrading quality too much.

2.2 Texture Compression system requirements

Texture compression systems must allow fast random access to texels.

A fixed compression ration eases address computations. While lossless texture compression systems have been proposed [11] they require additional hardware to implement. Only texture compression systems with a fixed compression ratio (and thus only lossy systems) will be considered herein.

Since computer graphics systems are typically highly pipelined it is desirable to keep the number of indirections during texture lookup low. Most of the discussed texture compression systems have no indirections.

Texture caches are used in graphics hardware to speed up texture accesses. A texture compression system should work well with existing caches, thus it is important that it preserves the locality of reference on which caches are based.

The decompression algorithm should be simple, fast, and easy to implement in hardware. There is no such requirement on the compression algorithm though.

3 Important texture compression systems

This section presents three texture compresion systems used today. Each has been used to compress the lorikeet image at a 6:1 compression ratio, at the end of each subsection the compressed image (and a closeup of the lorikeet's beak) can be seen side-by-side with the compressed image. At the section's end the three systems will be compared, their respective advantages and drawbacks emphasized.

3.1 Indexed color

Indexed color is the first texture compression algorithm proposed [7] as well as the first to be considered for implementation in hardware [10]. While common on older graphics hardware and being part of the OpenGL ES 1.1 common profile [3] today's consumer-level hardware no longer supports it except for embedded systems such as mobile phones, where OpenGL ES 1.x is used. A texture compressed using a texture compression system based on indexed color is called a palettized textures.

A palettized texture has a color table, which contains a number of colors stored at high precision. For each texel an index into the color table is stored. This results in one indirection per texel lookup which is the reason palettized textures are rarely used today.

Indexed color can be considered a special case of vector quantization by treating the individual texels as vectors in color space.

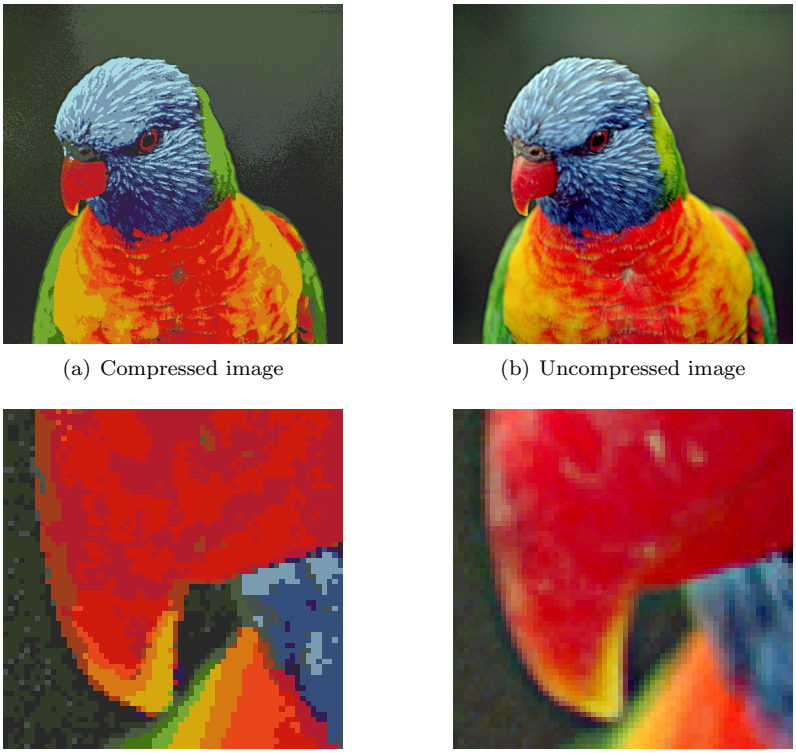
3.2 S3TC

S3TC [19] is the most common texture compression system. It is available on all consumer-level hardware. S3TC has been integrated into Microsoft's DirectX and is considered for inclusion in OpenGL 3 [5]. S3TC is, as most texture compression systems, patented, so this would make free OpenGL implementations illegal in large parts of the world.

S3TC consists of 5 different formats, which differ in their handling of the alpha channel. DXT2 and DXT4 are rarely used and will not be discussed here. A variant called VTC [16] has been proposed by Nvidia to extend S3TC to allow textures of width and height other than multiples of 4.

DXT1 [20] stores a 64 bit value per 4x4 texel block. The block's first half is used to store two 16 bit RGB565 colors. The second half is used to store a 2 bit control code per texel. It has only very primitive alpha channel support.

If the first color as a 16 bit unsigned integer is greater than the second one the color for a given texel is the first color for code 0, the second color for code 1.



(a) Compressed image

(b) Uncompressed image

(c) Compressed image

(d) Uncompressed image

Figure 1: Lorikeet compressed using indexed color

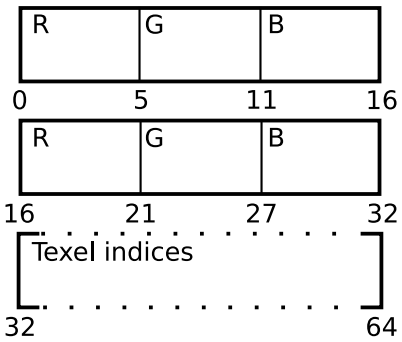
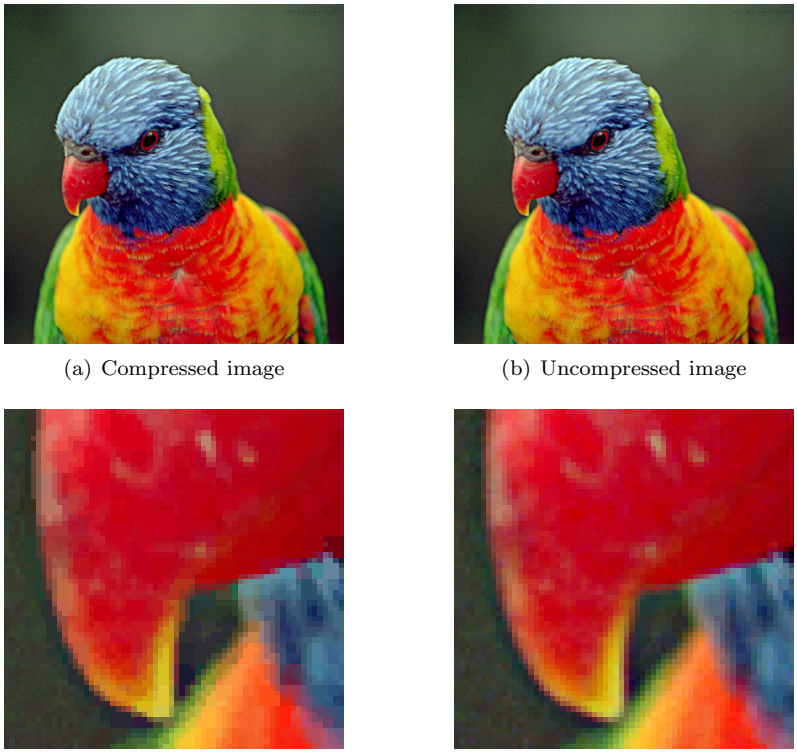


Figure 2: DXT1



(a) Compressed image

(b) Uncompressed image

(c) Compressed image

(d) Uncompressed image

Figure 3: Lorikeet compressed using DXT1

A linear interpolation is done to get the colors for control codes 2 (the first color times two thirds plus the second color times one third) and 3 (the first color time one third plus the second color times two thirds.). The block is opaque.

If the first color is not greater than the second one texel color is given as above for control codes 0 and 1. For control code 2 the average between the two colors is used, 3 means black. The texel is opaque for control codes 0, 1 and 2. It is transparent for control code 3.

DXT3 and DXT5 store a 128 bit value per 4x4 texel block. 64 bits are used to store RGB information as in DXT1. DXT3 contains a 4 bit alpha value per texel. DXT5 stores two 8 bit alpha values and a 3 bit index per texel. If the first alpha value is greater than the second one the index value is used to smoothly interpolate between the two alpha values. If the first alpha value is equal or less than the second one indices 0 to 5 are used to interpolate between the alpha values, while indices 6 and 7 have special meanings of completely transparent and completely opaque.

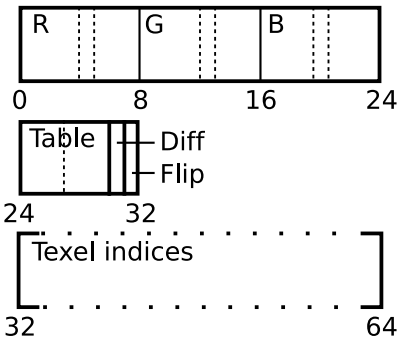


Figure 4: ETC

3.3 ETC

The idea behind ETC [21] (Ericsson Texture Compression, originally called iPACKMAN) is that humans are more sensitive to changes in luminance than in chrominance. ETC varies the luminance per texel while the chrominance can be changed for multiple texels at a time only.

ETC is used in high-end smartphones and was considered for inclusion into OpenGL ES 2.

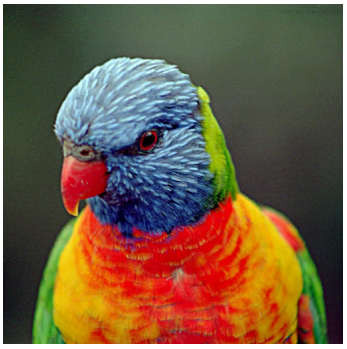
As DXT1 ETC stores a 64 bit value per 4x4 texel block. The block is divided into two 4x2 or 2x4 subblocks, controlled by a flip bit. Each subblock has a base color. These base colors are either stored as independent RGB444 colors or as one RGB555 color and a 3x3 bit difference vector (this differential mode is the main difference from the older PACKMAN [22] algorithm; it is useful to accuracy in blocks where color does not vary much between subblocks). For each subblock one of 16 hardcoded luminance tables is selected. For each texel a 2 bit index into the subblock's luminance table is stored.

The luminance tables vary from $(-8, -2, 2, 8)$ to $(-127, -42, 42, 127)$. Since they contain nonzero entries the base color is not used unmodified for any texel. Since the luminance variation is applied after expanding the base colors to 24 bit colors and there are luminance tables with small entries ETC can represent fine luminance variations.

3.4 Comparison

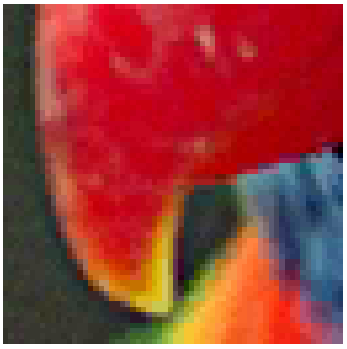
Indexed colors are good at compressing images that contain a low number of colors such as drawings or comics. For these type of images indexed colors are the best compression system, while they are the worst for everything else.

S3TC is good at chrominance variations. The low number (4) of colors per block and the color's limited precision result in visible artifacts all over the

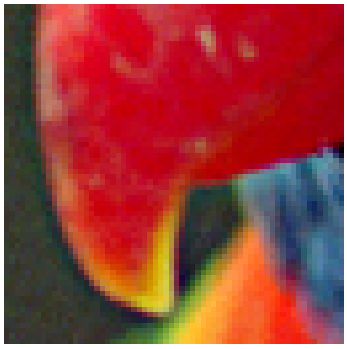


(a) Compressed image

(b) Uncompressed image



(c) Compressed image



(d) Uncompressed image

Figure 5: Lorikeet compressed using ETC

compressed texture.

ETC is good at areas of uniform chromacity where only luminance varies. It cannot represent smooth chrominance changes or hard lines between colors of the same luminance though.

4 Other texture compression systems

Other texture compression systems have been proposed over time; some like FXT1 or vector quantization have seen some use in the past, others like PVRTC or ATC are rather new compression systems that have not seen widespread use yet. With the exception of PACKMAN and ETC2 (both related to the ETC system discussed above) only compression systems where known hardware implementations exist have been included in this section.

4.1 Vector quantization

Vector quantization [6] is a generic lossy compression method. It is used in many areas like audio and image compression. Vector quantization has been used for texture compression in early ATI cards and the Sega Dreamcast video game console.

Vector quantization tries to find a small number of vectors to approximate given distribution of vectors keeping the error small. In texture compression the vectors are typically 2x2 or 4x4 texel blocks. Indexed color can be seen as vector quantization using 1x1 blocks. Vector quantization is no longer used today since it requires an indirection (accessing the table of tiles from which the texture is composed).

A similar system has been proposed to create large non-repeating textures from a small sample texture [23].

4.2 FXT1

FXT1 [1] was a competitor to S3TC created by 3Dfx. Initially FXT1-compressed texture were of higher quality than those compressed using S3TC, but this changed with more advanced S3TC compression algorithms. Since the end of 3Dfx only Intel still implements it in their graphics chips. A royality-free FXT1 license was available from 3Dfx [8].

4.3 latc, rgtc

latc [13] and rgtc [14] are texture compression systems optimized for two-channel textures (greyscale images with alpha channel for latc, red-green for rgtc). They store a 64 or 128 bit value for each 4x4 texel block.

4.4 3Dc

3Dc [4] is a texture compression algorithm optimized for compressing normal maps. ATI created it due to the low quality of S3TC-compressed normal maps.

4.5 PVRTC

PVRTC [9] is Imagination Technologies' texture compression algorithm for embedded systems. It tries to take advantage of the correlation of texel position and color. While it stores a 32 or 64 bit value per 4x4 texel block reading a texel requires access to the values of the nearest three neighbouring blocks, too. It is implemented in PowerVR graphics chips used in PDAs and smartphones.

4.6 PACKMAN

PACKMAN [22] is the predecessor to ETC. It is based on the same idea as ETC and has been created as a texture compression algorithm for mobile phones.

4.7 ETC2

ETC2 [12] is the successor to ETC. Not all 64 bit values represent valid ETC blocks. ETC2 uses these to introduce three new modes targeted at blocks that are problematic for ETC: The T and H-modes, which are similar to ETC, but allow for some per-texel variation in chrominance at the cost of less luminance variation and one mode for smoothly varying chrominance.

4.8 ATC

ATC is AMD's new texture compression algorithm for embedded systems. Technical detail is unknown, though it probably is based on the patent [17] registered by the person that wrote ATC's OpenGL ES extension specification [2] (which states "The details of these formats is not disclosed").

A Texture compression in OpenGL

This appendix will give a brief overview of texture compression in OpenGL starting with some introductionary information on OpenGL and OpenGL extensions, describing the way texture compression is used, ending with OpenGL ES, an OpenGL variant for embedded systems.

A.1 Introduction

OpenGL is one of the two remaining important 3D graphics API, the other being Microsoft's Direct3D (part of DirectX). It is governed by the OpenGL Architecture Review Board (ARB), once an independent consortium formed in 1992, now a working group within the Khronos Group. The OpenGL API evolves slowly

with new functionality tried in OpenGL extensions first. These extensions can be vendor-specific (extension name starting in vendor acronym), multi-vendor (extension name starting in EXT) or approved by the ARB (extension name starting in GL).

Texture compression in the form of the S3TC system was first exposed in OpenGL through the S3_s3tc extension before the ARB agreed on a common interface.

A.2 Interface

The GL_ARB_texture_compression extension [18] provides an generic texture compression interface without requiring a specific texture compression system. It provides functions for querying the list of supported texture compression systems, supplying compressed one to three-dimensional textures to the graphics hardware (although so far texture compression systems focus on two-dimensional textures only). Applications can also supply uncompressed textures to the OpenGL implementation requesting that they be compressed by the OpenGL implementation. The GL_ARB_texture_compressionfunctionality was promoted to a core feature in OpenGL 1.3 [15].

Specific texture compression systems have their own extensions (i.e. FXT1 has 3DFX_texture_compression_FXT1).

A.3 OpenGL ES

OpenGL ES is a variant of OpenGL optimized for embedded systems. It removes large parts of legacy functionality and features that are difficult to implement on embedded systems. OpenGL 1.1 has palettized textures, some implementations support PVRTC, too; ETC was considered for OpenGL ES 2 and most implementations support it.

There are some special requirements on texture compression systems for embedded systems. Due to limited memory bus width texture compression systems should have low block size (typically 64 bits). Of the popular S3TC compression system only the DXT1 mode has a block size of 64 bits, DXT2 to DXT5 use 128 bits. PACKMAN, ETC, ETC2 and PVRTC have been created with embedded systems in mind and a block size of 64 bits.

B Bibliography

References

[1] 3dfx Interactive. 3dfx FXT1 Texture Compression White Paper. `http://www3.sharkyextreme.com/hardware/articles/99/3dfx_fxt/wp/`.

[2] Aaftab Munshi. AMD_compressed_ATC_texture. `http://khronos.org/registry/gles/extensions/AMD/AMD_compressed_ATC_texture.txt`.

[3] Aaftab Munshi, Jon Leech. OpenGL ES Common/Common-Lite Profile Specification Version 1.1.10. http://www.khronos.org/registry/gles/specs/1.1/es_full_spec.1.1.10.pdf.

[4] ATI Technologies. ATI Radeon X800 3Dc White Paper. http://www.ati.com/products/radeonx800/3DcWhitePaper.pdf.

[5] Barthold Lichtenbelt (OpenGL ARB Working Group chair). OpenGL 3 Updates. http://www.opengl.org/discussion_boards/ubbthreads.php?ubb=showflat&Number=229374#Post229374.

[6] Andrew C. Beers, Maneesh Agrawala, and Navin Chaddha. Rendering from compressed textures. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 373–378, New York, NY, USA, 1996. ACM.

[7] Graham Campbell, Thomas A. DeFanti, Jeff Frederiksen, Stephen A. Joyce, and Lawrence A. Leske. Two bit/pixel full color encoding. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 215–223, New York, NY, USA, 1986. ACM.

[8] Don Mullis. 3DFX_texture_compression_FXT1. http://opengl.org/registry/specs/3DFX/texture_compression_FXT1.txt.

[9] Simon Fenney. Texture compression using low-frequency signal modulation. In *HWWS '03: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 84–91, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[10] G. Knittel, A. Schilling, A. Kugler, W. Straßer. Hardware for Superior Texture Performance. In *Computers & Graphics 20*, pages 475–481, 1996.

[11] T. Inada and M. D. McCool. Compressed lossless texture representation and caching. In *GH '06: Proceedings of the 21st ACM SIGGRAPH/Eurographics symposium on Graphics hardware*, pages 111–120, New York, NY, USA, 2006. ACM.

[12] Jacob Ström and Martin Pettersson. ETC2: texture compression using invalid combinations. In *GH '07: Proceedings of the 22nd ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics hardware*, pages 49–54, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[13] Mark J. Kilgard, Pat Brown, Yanjun Zhang. EXT_texture_compression_latc. http://opengl.org/registry/specs/EXT/texture_compression_latc.txt.

[14] Mark J. Kilgard, Pat Brown, Yanjun Zhang. EXT_texture_compression_rgtc. http://opengl.org/registry/specs/EXT/texture_compression_rgtc.txt.

[15] Mark Segal, Kurt Akeley. The OpenGL Graphics System: A Specification (Version 1.3). http://www.opengl.org/documentation/specs/version1.3/glspec13.pdf.

[16] Matt Craighead. NV_texture_compression_vtc. http://opengl.org/registry/specs/NV/texture_compression_vtc.txt.

[17] Aaftab Munshi. Block-based image compression method and apparatus. US Patent 20060215914.

[18] Pat Brown. ARB_texture_compression. http://opengl.org/registry/specs/ARB/texture_compression.txt.

[19] Pat Brown. EXT_texture_compression_s3tc. http://opengl.org/registry/specs/EXT/texture_compression_s3tc.txt.

[20] Pat Brown, Mathias Agopian. EXT_texture_compression_dxt1. http://opengl.org/registry/specs/EXT/texture_compression_dxt1.txt.

[21] Jacob Ström and Tomas Akenine-Möller. iPACKMAN: high-quality, low-complexity texture compression for mobile phones. In *HWWS '05: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 63–70, New York, NY, USA, 2005. ACM.

[22] Jacob Ström and Tomas Akenine-Möller. Packman: Texture compression for mobile phones. http://www.cs.lth.se/home/Tomas_Akenine_Moller/pubs/packman_sketch.pdf.

[23] Li-Yi Wei. Tile-based texture mapping on graphics hardware. In *HWWS '04: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 55–63, New York, NY, USA, 2004. ACM.