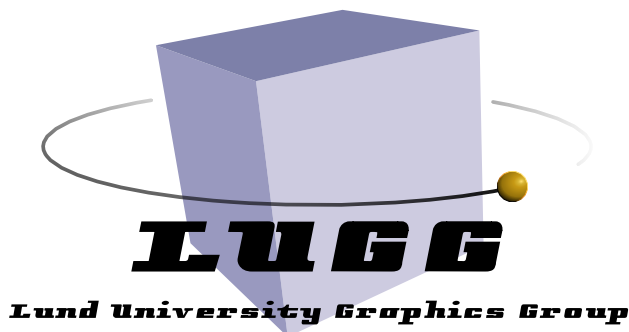




Texture Compression

slides courtesy Jacob Ström, Ericsson Research



Michael Doggett
Department of Computer Science
Lund University

Overview

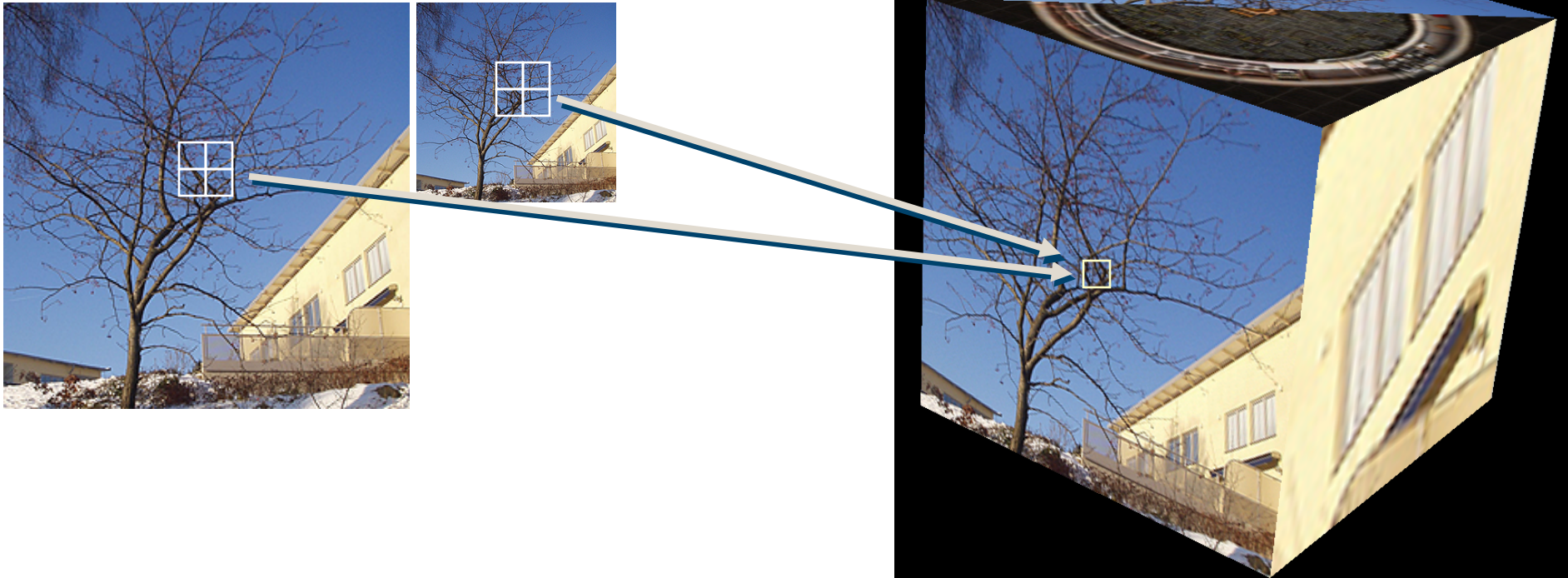
- **Benefits of texture compression**
- **Differences from ordinary image compression**
- **Texture compression algorithms**
 - **BTC - The mother of all texture compression systems**
 - **S3TC/DXTC/BC1 - used in desktops/laptops (PCs, Macs) (4bpp, old)**
 - **BC7 - High bit rate texture compression on desktops/laptops (8bpp, current)**
 - **PACKMAN, ETC - used in Android 2.2 (old)**

Texture Mapping is a Bandwidth Hog

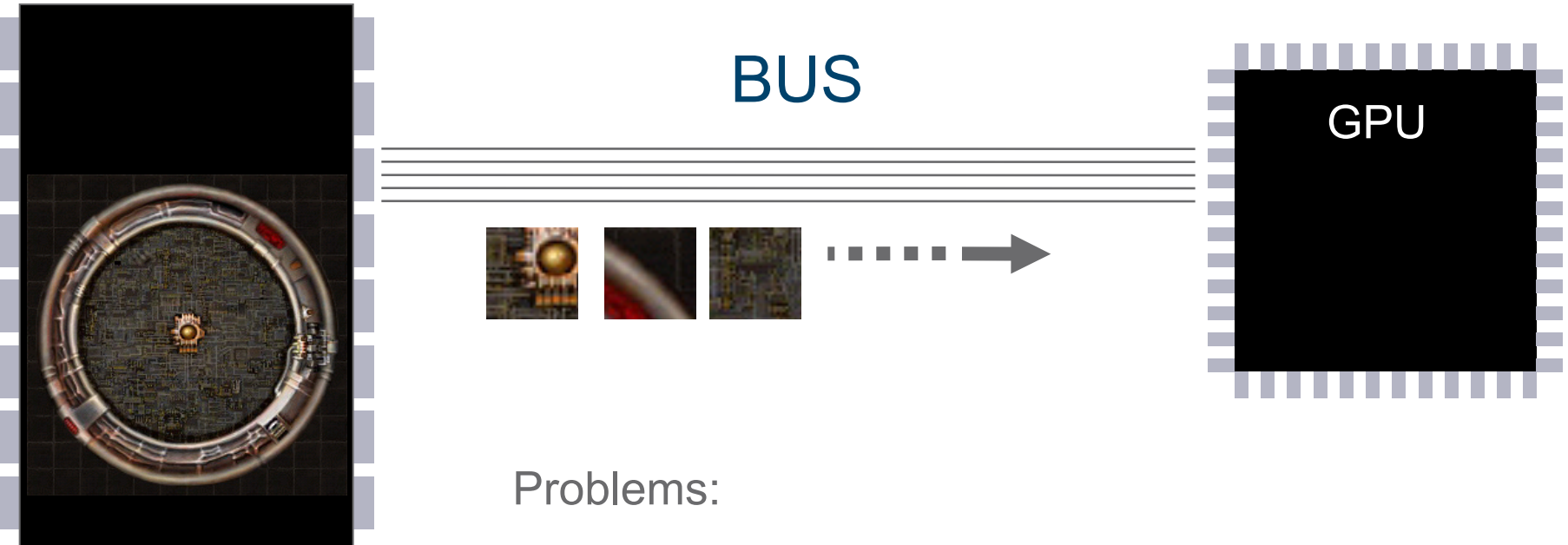
- › For each pixel drawn in the image, eight pixels from the texture (texels) are usually read.

texture mipmap levels

drawn image



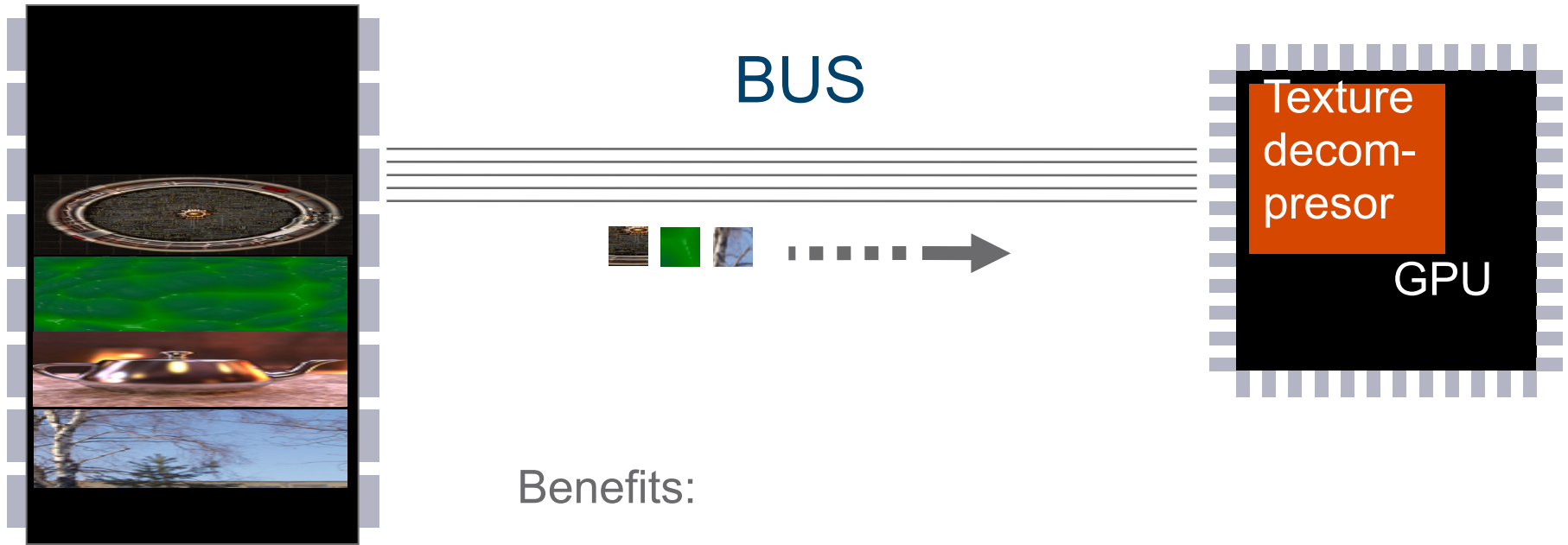
Texturing and the bus



Problems:

- Bus can get full (performance bottleneck)

Texture Compression helps

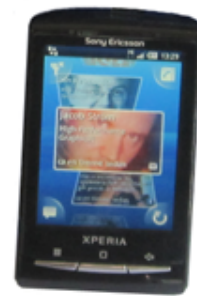


Benefits:

- Less traffic on bus = higher performance
- Less traffic on bus = lower power consumption

Power Savings

- › Especially good for mobile devices





... unless you want to carry an extra battery.

Benefits of Texture Compression

› Higher Performance

- Bandwidth is usually the factor limiting performance in rasterization-based graphics hardware.
- Texture Compression reduces texturing bandwidth with a factor of up to 6
- Spare bandwidth can be used for higher performance, or lower power consumption (mobile case)

› Higher Quality! (Yes, really...)

- Even a huge video memory gets full.
- With a compression ratio of 6, you can increase the resolution one mipmap level and still save memory

Benefits of Texture Compression

› Higher Performance

- Bandwidth is usually the factor limiting performance in rasterization-based graphics hardware.
- Texture Compression reduces texturing bandwidth with a factor of up to 6
- Spare bandwidth can be used for higher performance, or lower power consumption (mobile case)

› Higher Quality! (Yes, really...)

- Even a huge video memory gets full.
- With a compression ratio of 6, you can increase the resolution one mipmap level and still save memory

with texture compression,
128x128 pix, 8192 bytes



Benefits of Texture Compression

> Higher Performance

- Bandwidth is usually the factor limiting performance in rasterization-based graphics hardware.
- Texture Compression reduces texturing bandwidth with a factor of up to 6
- Spare bandwidth can be used for higher performance, or lower power consumption (mobile case)

> Higher Quality! (Yes, really...)

- Even a huge video memory gets full.
- With a compression ratio of 6, you can increase the resolution one mipmap level and still save memory

with texture compression,
128x128 pix, 8192 bytes

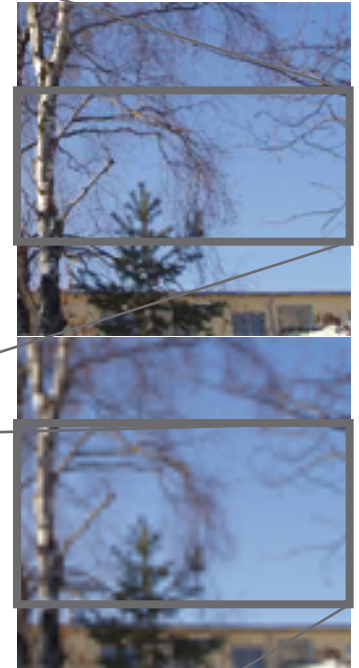


no texture compression,
downsampled to 64x64,
12288 bytes

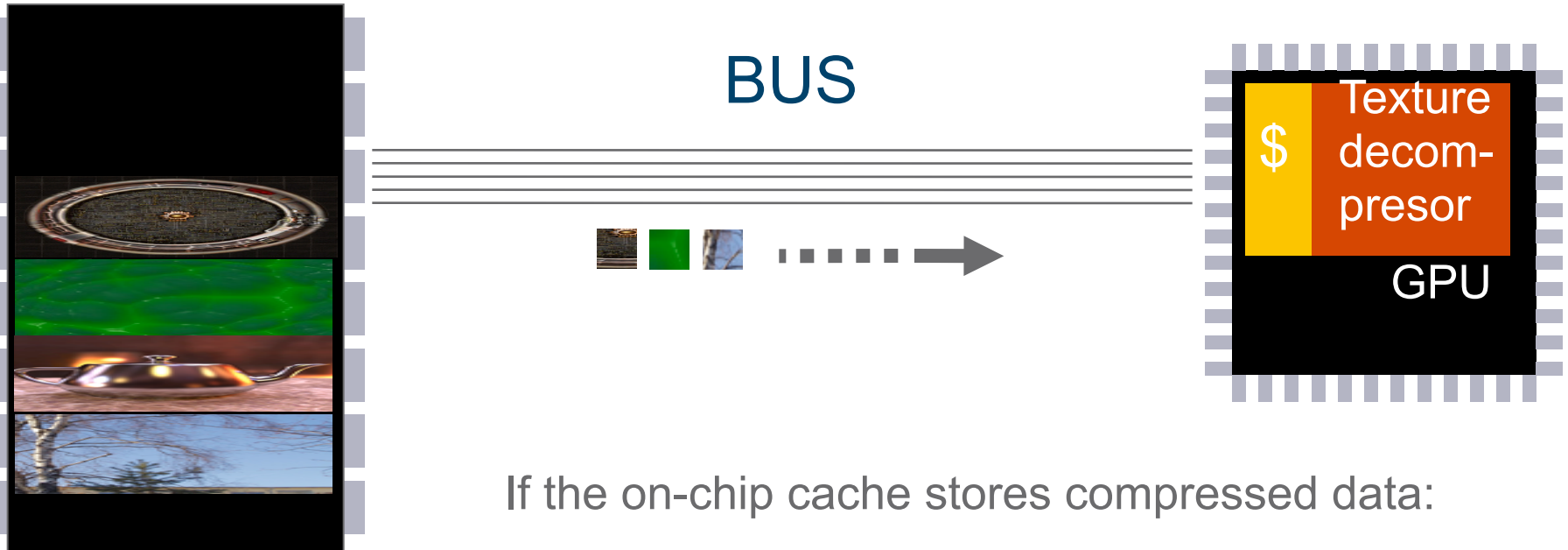
Benefits of Texture Compression



with texture compression,
128x128 pix, 8192 bytes



no texture compression,
downsampled to 64x64,
12288 bytes



If the on-chip cache stores compressed data:

- More blocks can fit in cache = better hit rate
- Mobile case: a smaller cache (less silicon) can hold the same number of blocks

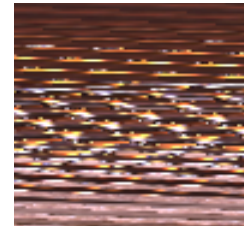
Difference to Image Compression

why not just use JPEG?

- › First a short recap on how JPEG compresses images



image



JPEG bits

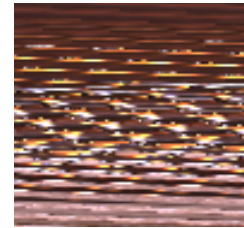
Difference to Image Compression

why not just use JPEG?

- › First a short recap on how JPEG compresses images
 - The image is first divided into 8x8 blocks.



image

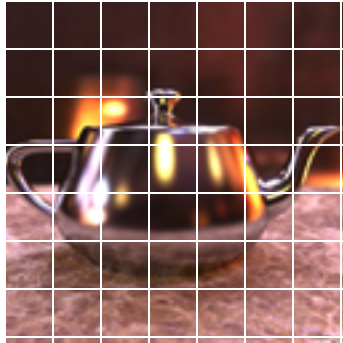


JPEG bits

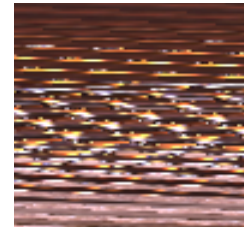
Difference to Image Compression

why not just use JPEG?

- › First a short recap on how JPEG compresses images
 - The image is first divided into 8x8 blocks.



image



JPEG bits

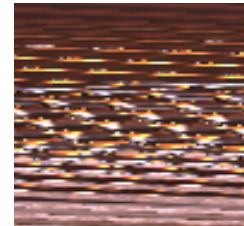
Difference to Image Compression

why not just use JPEG?

- › First a short recap on how JPEG compresses images
 - The image is first divided into 8x8 blocks.
 - Each block is then encoded and put into the file



image

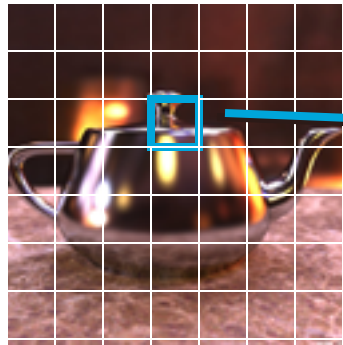


JPEG bits

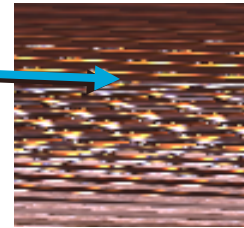
Difference to Image Compression

why not just use JPEG?

- › First a short recap on how JPEG compresses images
 - The image is first divided into 8x8 blocks.
 - Each block is then encoded and put into the file



image



JPEG bits

Difference to Image Compression

why not just use JPEG?

- › Most image compression algorithms, such as JPEG, uses variable bit length coding (VLC).
- › A block that is hard to code is allowed to occupy more bits than a block that is, for instance, just black.

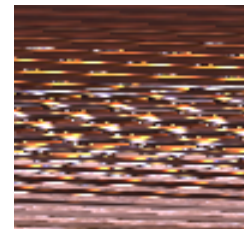
Difference to Image Compression

why not just use JPEG?

- › Most image compression algorithms, such as JPEG, uses variable bit length coding (VLC).
- › A block that is hard to code is allowed to occupy more bits than a block that is, for instance, just black.



image

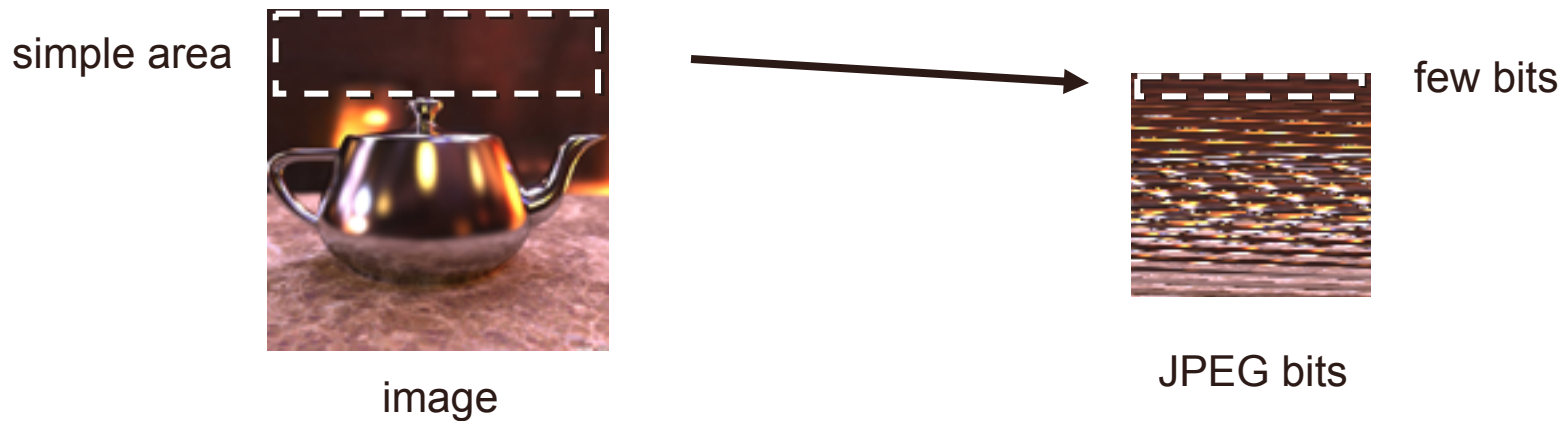


JPEG bits

Difference to Image Compression

why not just use JPEG?

- › Most image compression algorithms, such as JPEG, uses variable bit length coding (VLC).
- › A block that is hard to code is allowed to occupy more bits than a block that is, for instance, just black.



Difference to Image Compression

why not just use JPEG?

- › Most image compression algorithms, such as JPEG, uses variable bit length coding (VLC).
- › A block that is hard to code is allowed to occupy more bits than a block that is, for instance, just black.



Difference to Image Compression

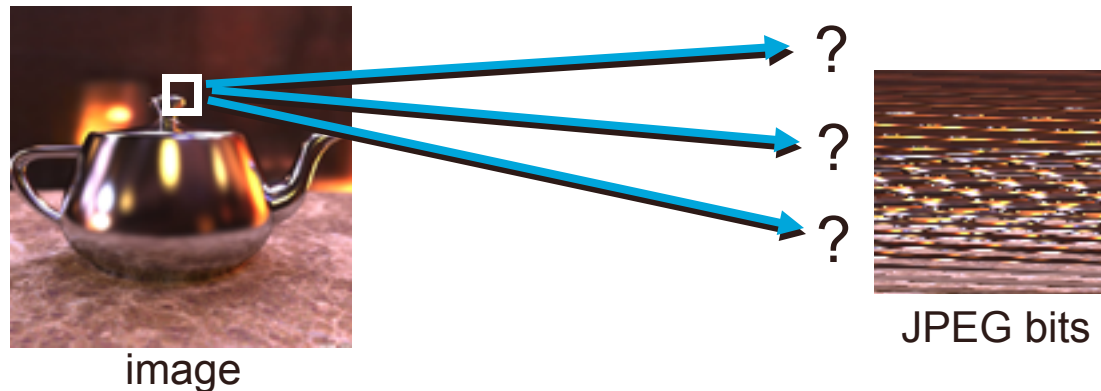
why not just use JPEG?

- › However, variable bit rate also means that you cannot calculate the address for a pixel in the JPEG bits.

Difference to Image Compression

why not just use JPEG?

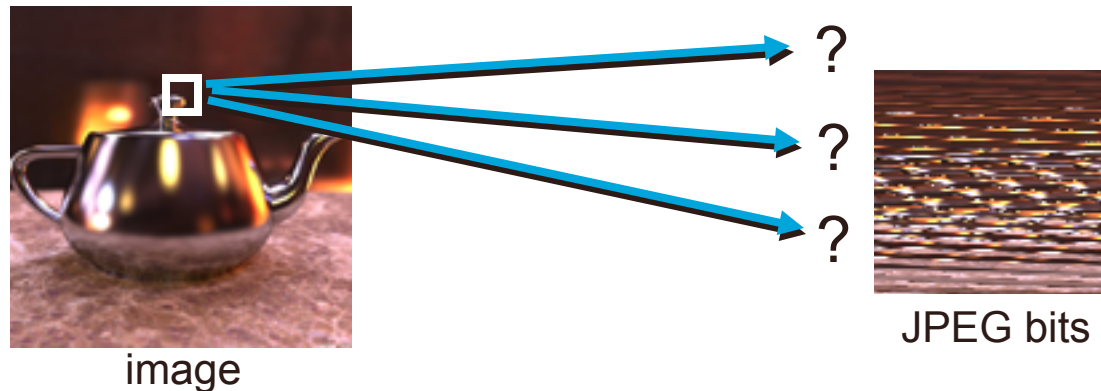
- › However, variable bit rate also means that you cannot calculate the address for a pixel in the JPEG bits.



Difference to Image Compression

why not just use JPEG?

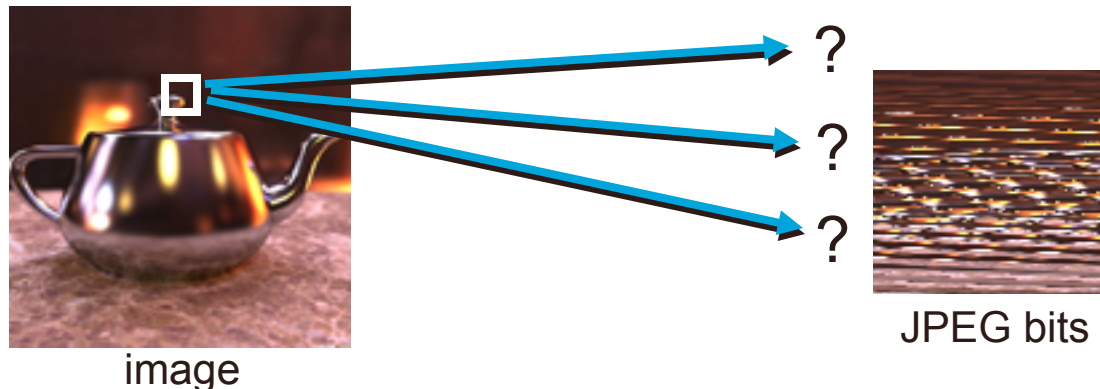
- › However, variable bit rate also means that you cannot calculate the address for a pixel in the JPEG bits.
- › In order to know the address for a particular pixel, you have to parse the entire file.



Difference to Image Compression

why not just use JPEG?

- › However, variable bit rate also means that you cannot calculate the address for a pixel in the JPEG bits.
- › In order to know the address for a particular pixel, you have to parse the entire file.
- › During rendering, you would have to parse the **entire file for every texel!** Not feasible.

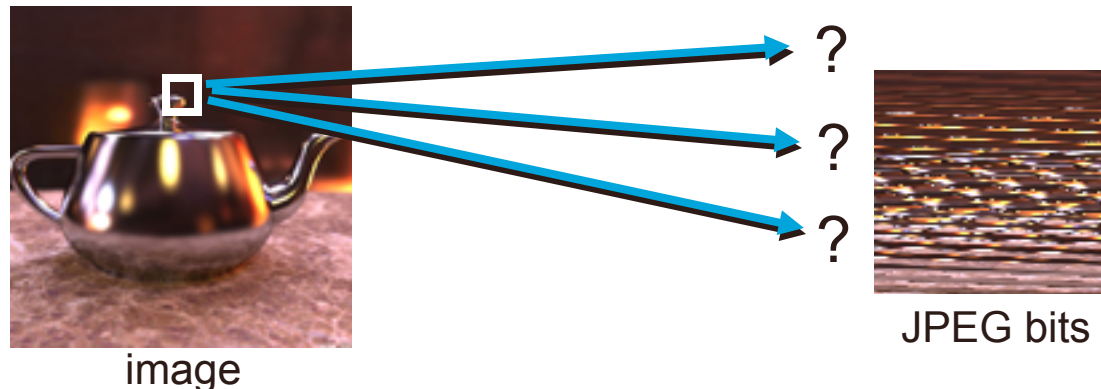


left image courtesy of Henrik Wann Jensen

Difference to Image Compression

why not just use JPEG?

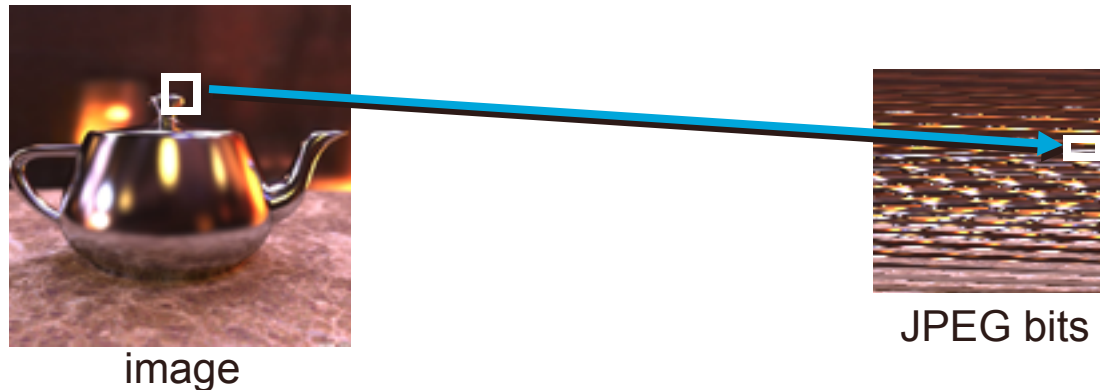
- › Therefore, most texture compression coders are **fixed rate** coders, which means that each block in the image occupies the same number of bits, for instance 64 bits per 4x4 block.
- › In this way, it is simple to calculate the address for a particular block in the compressed texture bit stream.



Difference to Image Compression

why not just use JPEG?

- › Therefore, most texture compression coders are **fixed rate** coders, which means that each block in the image occupies the same number of bits, for instance 64 bits per 4x4 block.
- › In this way, it is simple to calculate the address for a particular block in the compressed texture bit stream.



Difference to Image Compression

why not just use JPEG?

- › Note, that there is only one rate that will guarantee error free coding, and that is to have no compression at all!
- › Thus, for fixed rate coding, one has to allow error (distortion) in the image. The goal is to make this error as small as possible.



original

≈



compressed and decompressed

Difference to Image Compression

why not just use JPEG?

- › Decompression should be of low complexity.
- › Up to eight texels must be decompressed for each pixel.
- › If we are unlucky, all eight texels can be in different blocks.

texture mipmap levels

drawn image

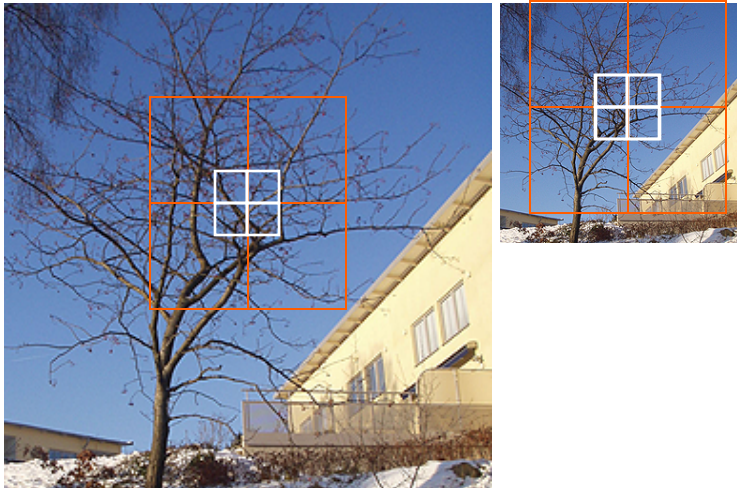


Difference to Image Compression

why not just use JPEG?

- › Decompression should be of low complexity.
- › Up to eight texels must be decompressed for each pixel.
- › If we are unlucky, all eight texels can be in different blocks.

texture mipmap levels



- › This means that we have to have eight parallel block decompressors on the chip to deliver one filtered pixel per clock.

Differences to Image Compression

Summary

1. Random access is needed – a fixed rate coder makes this possible.
2. Several parallel units needed – low hardware decompression complexity necessary. (Long compression times OK though!)
3. Indirect addressing due to use of palettes or other global, texture depending data should be avoided.



Texture Compression Formats

Block Truncation Coding

Predecessor to texture compression techniques

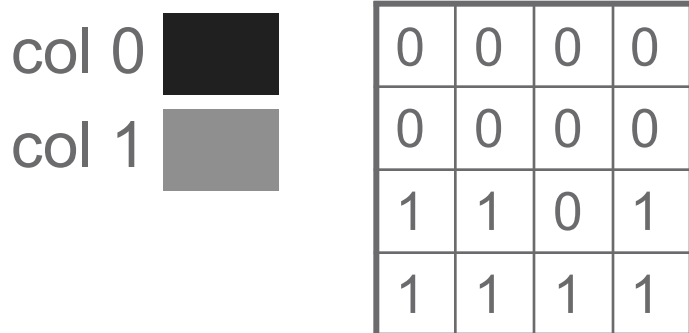
BTC – Block Truncation Coding

- › Image is divided into 4x4 blocks
- › Two 8-bit gray shades are encoded per block



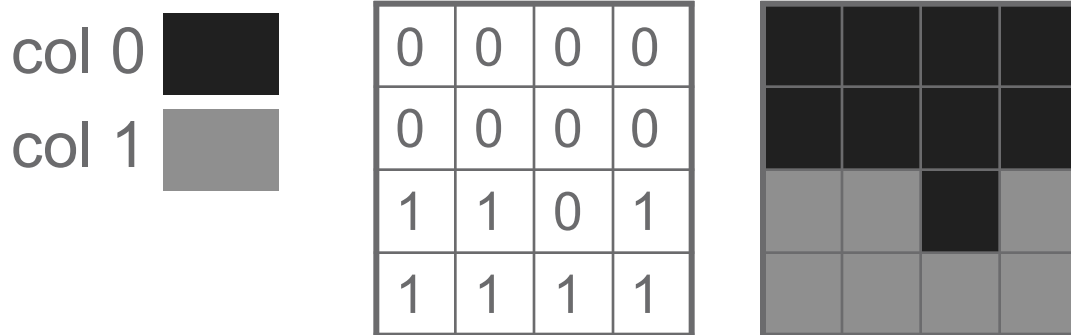
BTC – Block Truncation Coding

- › Image is divided into 4x4 blocks
- › Two 8-bit gray shades are encoded per block
- › A bit mask of 16 bits is also used.



BTC – Block Truncation Coding

- › Image is divided into 4x4 blocks
- › Two 8-bit gray shades are encoded per block
- › A bit mask of 16 bits is also used.

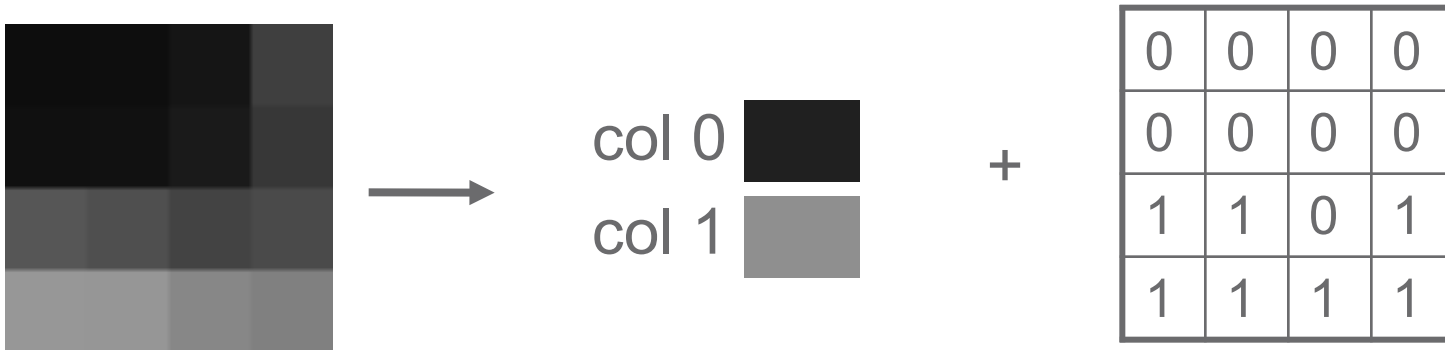


- › Bit rate equals $8+8+16 = 32$ bits per block, i.e., 2 bits per pixel (bpp).
 - Compared to 32 bits per pixel
- › Everything is contained in the codeword, no “global data” or color palette needs to be read.

BTC – Block Truncation Coding

Compression

- › To compress a block, you need to find a pair of base colors and the bitmask.
- › If you have the base colors, it is simple to get the bitmask: Just try each pixel and see which color is best.
- › Since there are only 256×256 such pairs, you can try all pairs.



BTC – Block Truncation Coding

Quality

- › However, having only two shades of gray gives rise to banding artifacts.



S3TC

Used on PCs (Windows, Mac)

S3TC – S3 Texture Compression

also called BC1-3 (used to be DXT1)

- › S3TC can be seen as an extension of BTC.
- › Instead of two gray scales, two colors are stored per block in RGB565.

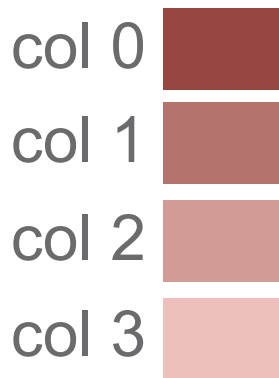
col 0 

col 3 

S3TC – S3 Texture Compression

also called DXT1









- › S3TC can be seen as an extension of BTC.
- › Instead of two gray scales, two colors are stored per block in RGB565.
- › Two more colors are interpolated between the stored ones.



S3TC – S3 Texture Compression

also called DXT1

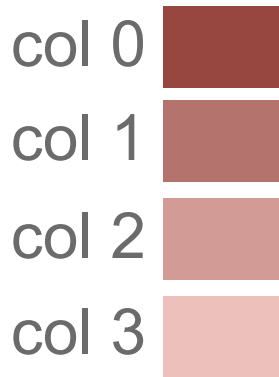
- › S3TC can be seen as an extension of BTC.
- › Instead of two gray scales, two colors are stored per block in RGB565.
- › Two more colors are interpolated between the stored ones.

| | | | | |
|-------|---|----------------|--|---|
| col 0 |  | | | |
| col 1 |  | = 2/3 * (col 0 | ) + 1/3 * (col 3 | ) |
| col 2 |  | = 1/3 * (col 0 | ) + 2/3 * (col 3 | ) |
| col 3 |  | | | |

S3TC – S3 Texture Compression

also called DXT1

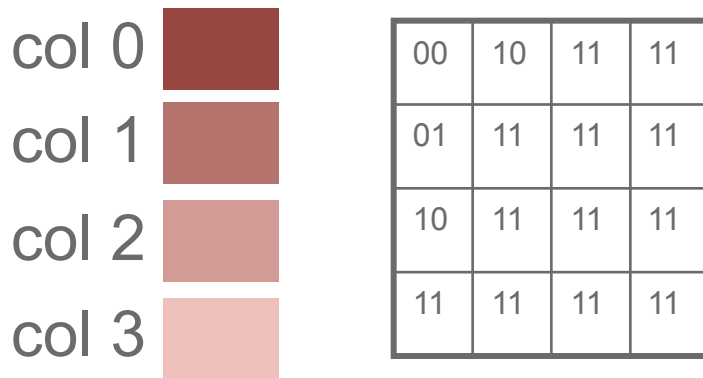
- › S3TC can be seen as an extension of BTC.
- › Instead of two gray scales, two colors are stored per block in RGB565.
- › Two more colors are interpolated between the stored ones.
- › Bit mask must now be 2 bits per pixel.



S3TC – S3 Texture Compression

also called DXT1

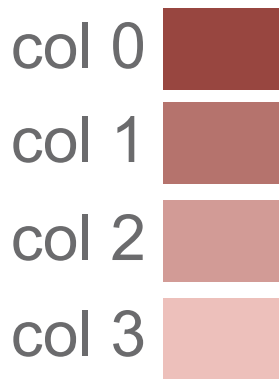
- › S3TC can be seen as an extension of BTC.
- › Instead of two gray scales, two colors are stored per block in RGB565.
- › Two more colors are interpolated between the stored ones.
- › Bit mask must now be 2 bits per pixel.



S3TC – S3 Texture Compression

also called DXT1

- › S3TC can be seen as an extension of BTC.
- › Instead of two gray scales, two colors are stored per block in RGB565.
- › Two more colors are interpolated between the stored ones.
- › Bit mask must now be 2 bits per pixel.



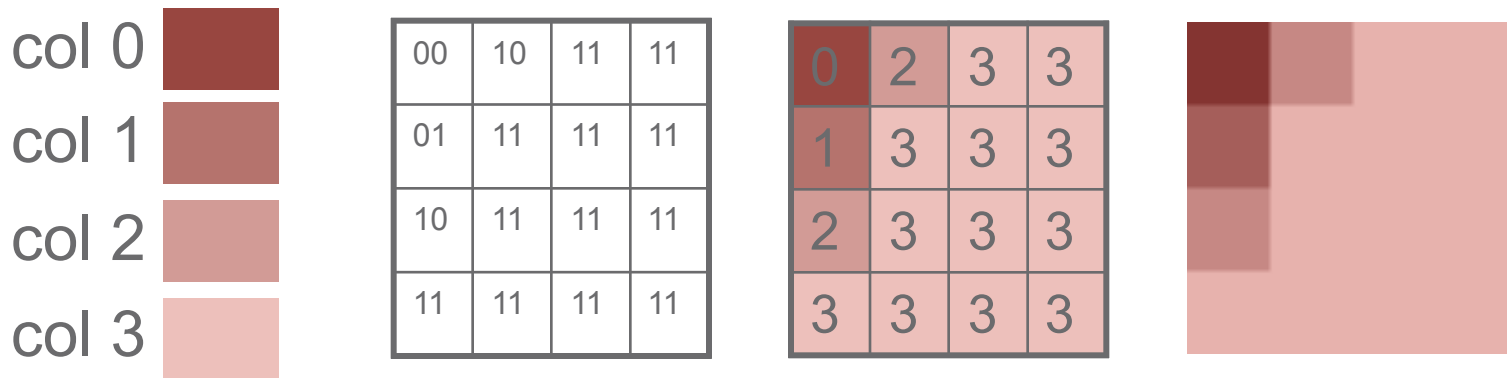
| | | | |
|----|----|----|----|
| 00 | 10 | 11 | 11 |
| 01 | 11 | 11 | 11 |
| 10 | 11 | 11 | 11 |
| 11 | 11 | 11 | 11 |

| | | | |
|---|---|---|---|
| 0 | 2 | 3 | 3 |
| 1 | 3 | 3 | 3 |
| 2 | 3 | 3 | 3 |
| 3 | 3 | 3 | 3 |

S3TC – S3 Texture Compression

also called DXT1

- › S3TC can be seen as an extension of BTC.
- › Instead of two gray scales, two colors are stored per block in RGB565.
- › Two more colors are interpolated between the stored ones.
- › Bit mask must now be 2 bits per pixel.



S3TC – S3 Texture Compression

quality

- › In this way, four colors per 4x4 block can be used instead of two gray scales – quality increases tremendously.

S3TC compression quality

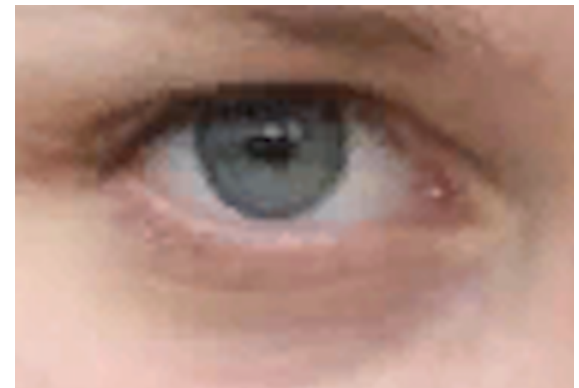
- Each 4x4 block of pixels uses 4 colours instead of 2 grey scale values
 - Big increase in quality
- S3TC was included into Direct3D as DXT1, and became an industry standard
- In DirectX11, BC1,2,3 are introduced replacing DXT1-5



Original



BC3



S3TC

Original



BC3

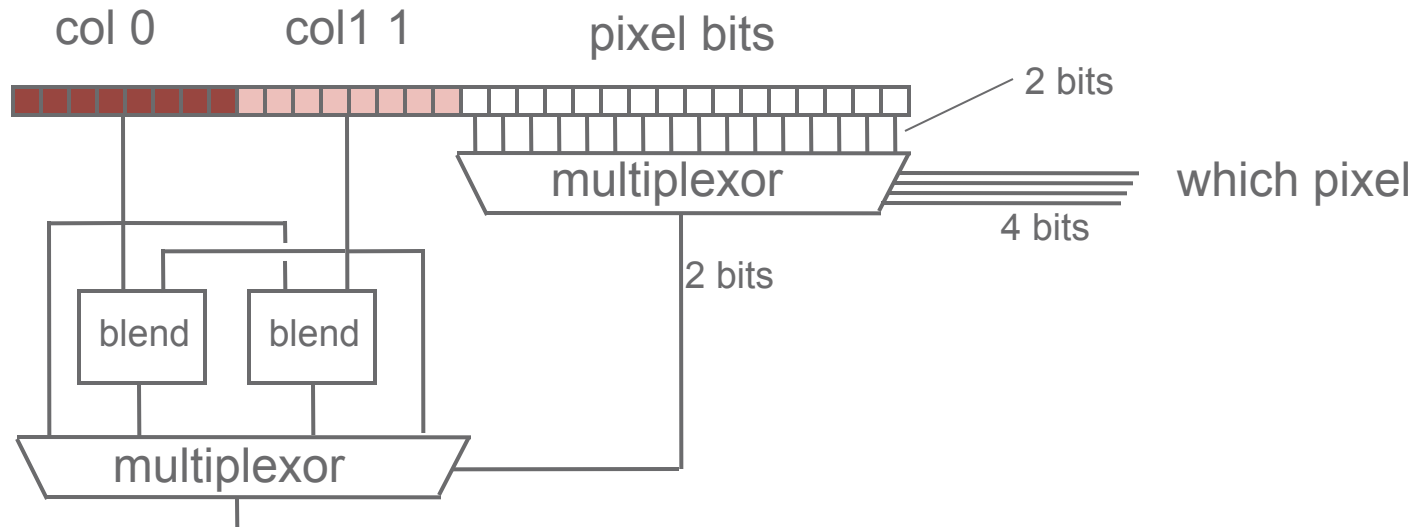


S3TC – S3 Texture Compression

- › The two base colors are stored in RGB565 (16 bits). Together with the 32 bits of pixel bits we get 64 bits per block, or 4 bpp. Compression ratio is thus 6:1.

S3TC – S3 Texture Compression

- > The two base colors are stored in RGB565 (16 bits). Together with the 32 bits of pixel bits we get 64 bits per block, or 4 bpp. Compression ratio is thus 6:1.
- > Decompression includes multiplication of 1/3 and 2/3.

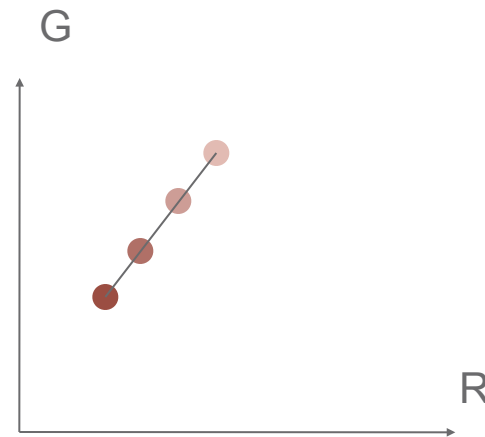


S3TC – S3 Texture Compression

- › Due to the way the intermediate colors are interpolated, the four colors of the block will lie on a straight line in RGB space.

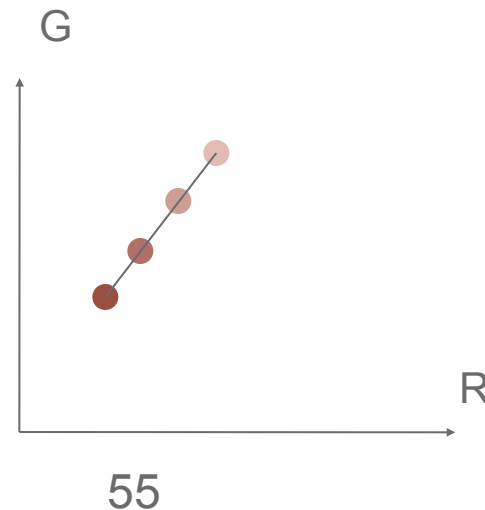
S3TC – S3 Texture Compression

- › Due to the way the intermediate colors are interpolated, the four colors of the block will lie on a straight line in RGB space.



S3TC – S3 Texture Compression

- › Due to the way the intermediate colors are interpolated, the four colors of the block will lie on a straight line in RGB space.
- › For many natural images, this is a rather good approximation.



RGBA TEXTURES

- › The S3TC/DXT1 method only encodes **RGB** textures using 64 bits per 4x4 block.
- › For RGBA textures, the alpha channel is encoded separately using another 64 bits in a similar manner.

RGBA TEXTURES

- › The S3TC/DXT1 method only encodes **RGB** textures using 64 bits per 4x4 block.
- › For RGBA textures, the alpha channel is encoded separately using another 64 bits in a similar manner.
- › Two gray colors are stored using 8 bit per gray level.
- › Six more are interpolated in between.

col 0 

col 7 

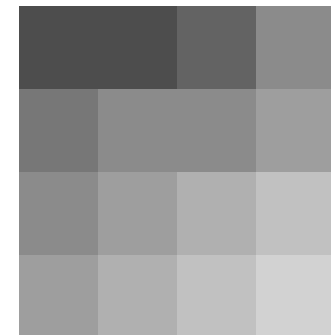
RGBA TEXTURES

- › The S3TC/DXT1 method only encodes RGB textures using 64 bits per 4x4 block.
- › For RGBA textures, the alpha channel is encoded separately using another 64 bits in a similar manner.
- › Two gray color are stored using 8 bit per gray level.
- › Six more are interpolated in between.



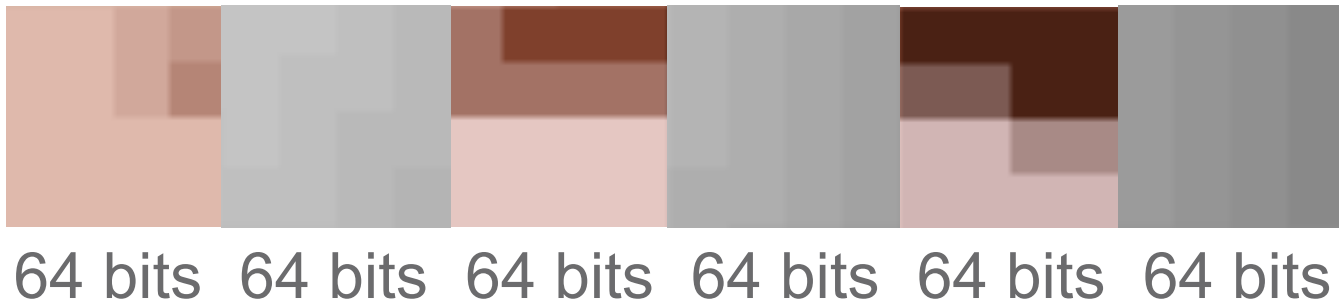
| | | | |
|-----|-----|-----|-----|
| 000 | 000 | 001 | 011 |
| 010 | 011 | 011 | 100 |
| 011 | 100 | 101 | 110 |
| 100 | 101 | 110 | 111 |

| | | | |
|---|---|---|---|
| 0 | 0 | 1 | 3 |
| 2 | 3 | 3 | 4 |
| 3 | 4 | 5 | 6 |
| 4 | 5 | 6 | 7 |



RGBA TEXTURES

- › In DirectX, RGBA textures are stored by interleaving 64 bits of RGB data with 64 bits of alpha-data.
- › This format is called BC3 (DXT5).





BC7 and BC6h

High bit rate texture compression for PCs

In OpenGL 4.2 they are called **BPTC**

BC7

- › In DXT5, 128 bits are spent per 4x4 pixels: 8 bits per pixel.
- › This is unevenly distributed per channel: 4 bpp for RGB (compression rate 6:1) and 4 bpp for alpha (compression rate only 2:1)
- › In BC7, 8 modes make it possible to change this balance:
 - Mode 4: 50% of bits RGB, 50% alpha (similar to BC3/DXT5)
 - Mode 5: 60% of bits RGB, 40% alpha (better color quality)
- › Sometimes alpha and color are correlated
 - Mode 6: 16 interpolated RGBA colors to choose from

BC7

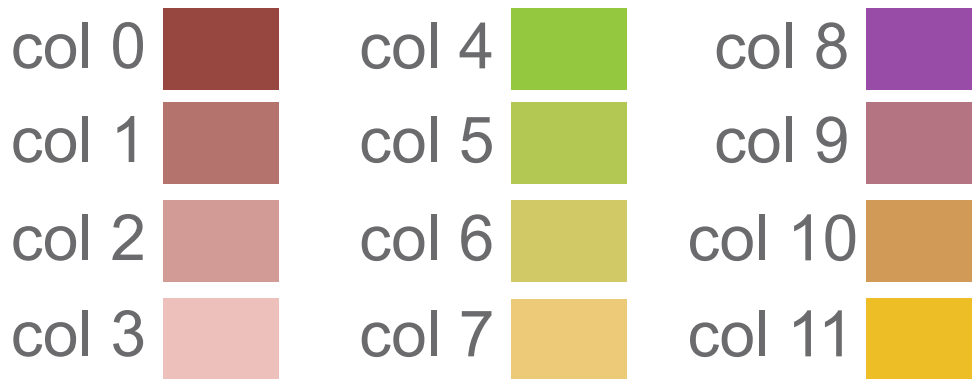
- › Some blocks in the texture have no alpha, then 100% of the bits can be used for RGB!
- › Instead of having just 1 interpolated color sequence, up to 3 are possible in BC7:

col 0  col 4  col 8 

col 3  col 7  col 11 

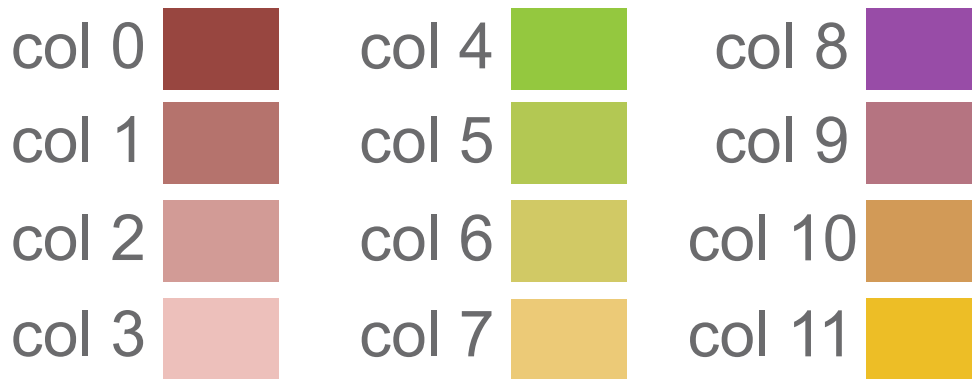
BC7

- › Some blocks in the texture have no alpha, then 100% of the bits can be used for RGB!
- › Instead of having just 1 interpolated color sequence, up to 3 are possible in BC7:



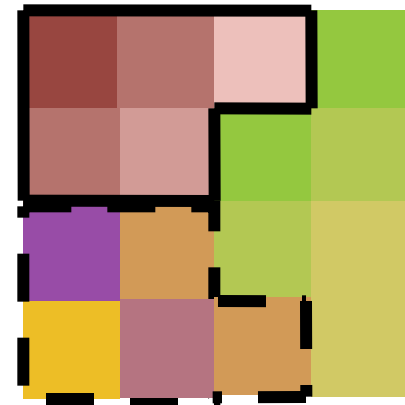
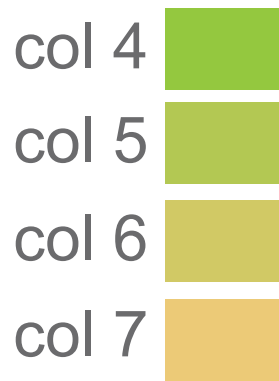
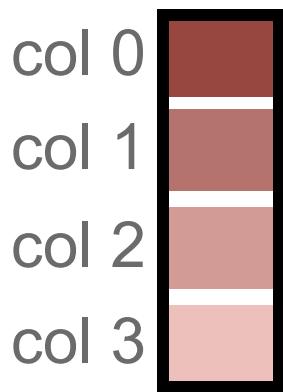
BC7

- › Some blocks in the texture have no alpha, then 100% of the bits can be used for RGB!
- › Instead of having just 1 interpolated color sequence, up to 3 are possible in BC7:



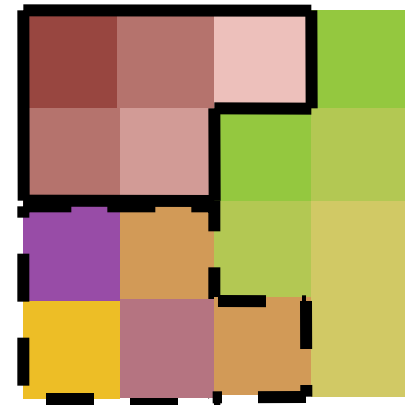
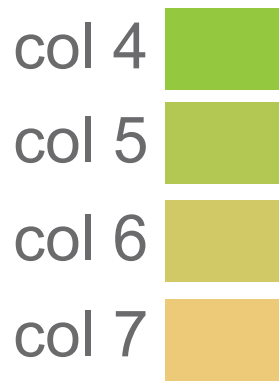
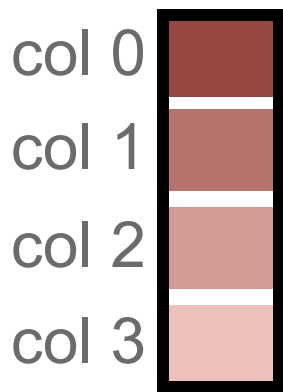
BC7

- › We need some way of telling from which color subset the pixel should fetch its color.

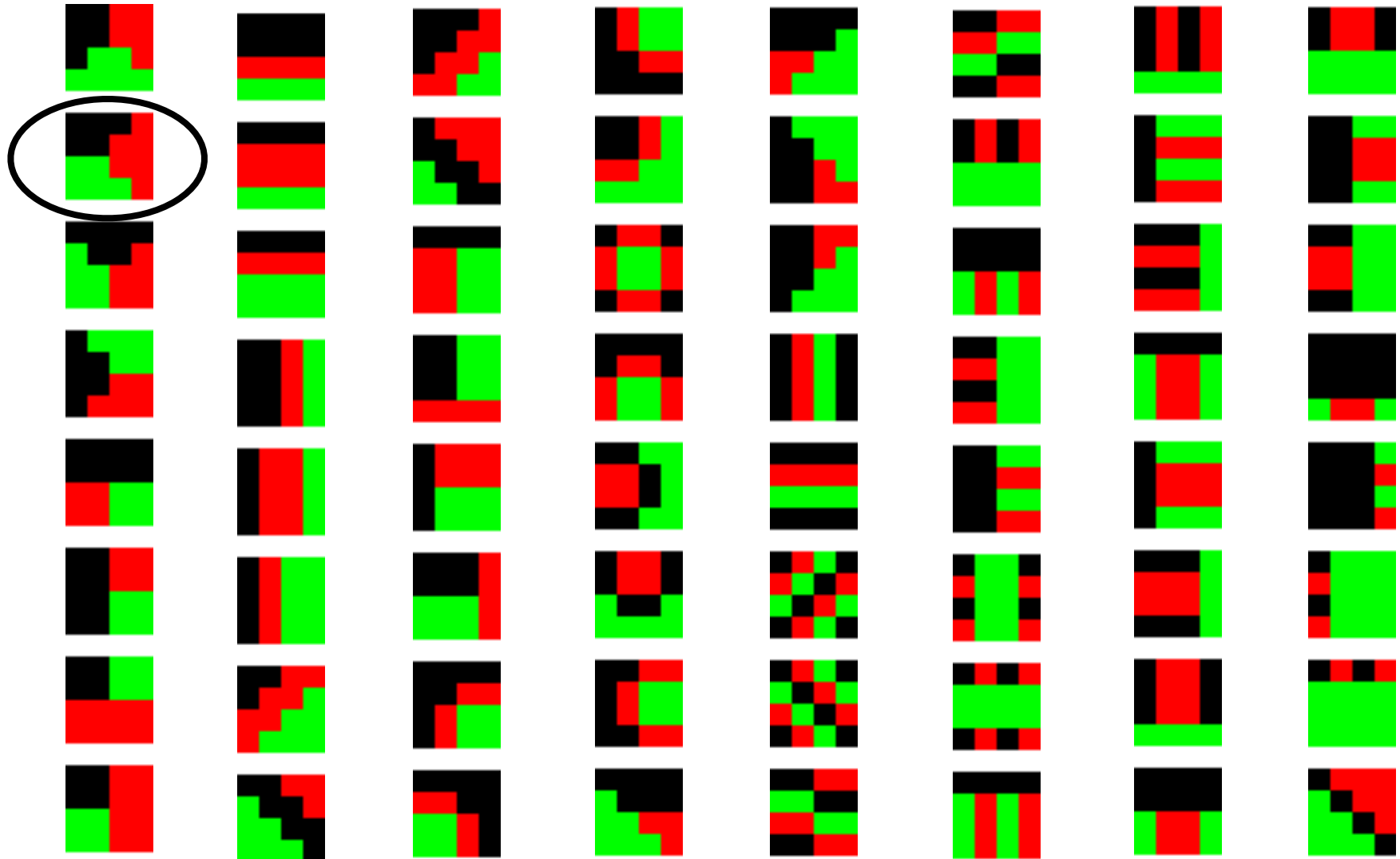


BC7

- › We need some way of telling from which color subset the pixel should fetch its color.
- › There are $3^{16} = 43\,046\,721$ possible block partitionings, which would require 25 bits to describe ($2^{25} > 3^{16}$)
- › BC7 chooses the 64 most common and selects from those using 6 bits.



64 most common patterns

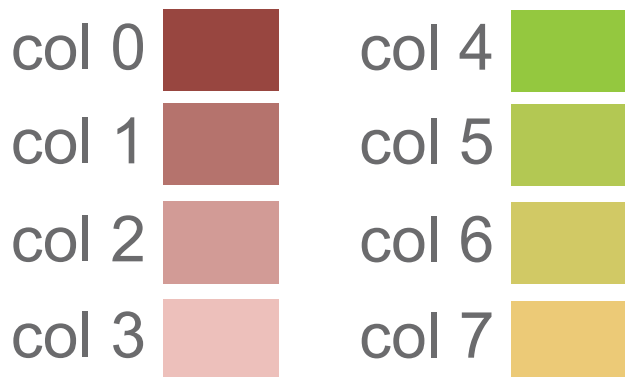


Patterns

- › The different modes have different number of color subsets:
 - Mode 0: 3 color subsets, 16 patterns
 - Mode 1: 2 color subsets, 64 patterns
 - Mode 2: 3 color subsets, 64 patterns
 - Mode 3: 2 color subsets, 64 patterns
 - Mode 4-7 color + alpha
- › Mode can be selected per block

BC6H

- › BC7 is a RGBA8888 format, but with 128 bits per 4x4 pixels, it is possible to also compress high dynamic range data (HDR)
- › BC6H decompresses to 16-bit floats (halves) using the same principles as BC7:
 - One or two color sequences
 - 32 patterns to tell which color sequence to be used
 - Only RGB (no alpha).



Summary

- **Texture compression requires :**
 - **Random access**
 - **Leads to fixed rate coding and lossy compression**
 - **Low complexity decompression for hardware**
 - **No indirect addressing**
- **Benefits**
 - **Lower bandwidth**
 - **Higher performance and lower power consumption**
 - **Less memory**
 - **Enables high resolution textures and improved cache performance**

Next week ...

- **Start working on your project !!**
 - **Description Due Monday**
- **Next week**
 - **GPU Architecture**
 - **Graphics Architecture and OpenCL**