Follow Along

Required Specifications

- Format(s) for the image bits
- Container
- Textures usually consist of

[Diagram showing a container with header, image 1, and multiple images labeled Image n]
Image Bits - Issues

- Can use image formats defined by the GPU APIs but
- uncompressed formats too large for transmission
- block-compressed formats suboptimal for transmission
- JPEG-compressed files can’t be randomly accessed
- Compression to GPU formats unavailable on most clients and slow
- A googol\(^1\) of GPU/Platform-Specific formats. Nightmare!

---

1. \(10^{100}\)
Image Bits - Solving Transmission Size

Supercompression

Rate Distortion Optimization (Crunch RDO Mode)

Image ➔ BC[1-5], ETC[12] ➔ RDO ➔ \{LZ, tANS\} ➔ Inflated ➔ GPU

Crunch (Crunch CRN mode)

Image ➔ BC[1-5], ETC[12] ➔ Crunch ➔ Decrunch ➔ GPU
Image Bits - Solving the Googol of formats

Universal Transcodable Format

2. No transcode necessary
Quality

- Will show comparison tests later.
- ETC1S-CRN works well for natural images.
- Not so well for line drawings, icons etc.
- LZ compression often works well in these cases
  - but will be uncompressed in GPU
- Will add support for variable block size and hi-precision formats: ASTC, BC6H/BC7, ETC2 in an incremental revision.
KTX File Structure

Header  Meta data  Mip level 0  Mip level n
KTX File Structure

Mip Level Structure

Array layer 0

face / z-slice 0

Array layer n

face / z-slice n

Mip level 0

Mip level n

Meta data

Header
KTX2 File Structure

Header

Data Format Descriptor

Meta data

Super-compression Global Data

Mip level 0

Mip level n

Header Additions
supercompressionScheme
vkFormat
levelOrder

Metadata Additions
ktxSwizzle
ktxWriter
KTX2 Header Additions: Supercompression

- supercompressionScheme
- Deflate (a.k.a zlib)
- Zstandard
- Crunch CRN mode
- LZ4 - under discussion
- tANS - under discussion

✍️

- Except for CRN, use these only with Crunch RDO-mode-processed or uncompressed images.
- Use CRN for supercompression of block-compressed textures.
- Transcodable format is supercompressed using CRN.
KTX 2 Header Additions: Others

• vkFormat field
  • makes loading of Vulkan textures easier
• levelOrder field
  • lets mip levels be ordered from smallest first, enabling streaming
Data Format Descriptor

- Exact description of texel format and color space
- non-OpenGL and non-Vulkan applications can use the image data without understanding OpenGL or Vulkan enums
- can save & load multi-sample images to/from KTX files
- applications that care can do color correction

1. See https://www.khronos.org/dataformat.
DFD Color Space Information

- Enums are provided to identify
  - common primaries
    - BT709/sRGB, BT2020, Display P3, Adobe RGB & more.
  - common transfer functions
    - Linear\(^1\), sRGB\(^1\), ITU (BT 601, 709 & 2020), Adobe RGB & more
- A full ICC profile could be incorporated via a KDFS extension

---

1. D3D/OpenGL/Vulkan/WebGL support only Linear & sRGB in hardware
Metadata Additions

- KTXswizzle
  - Indicates desired component mapping for a texture
- KTXwriter
  - File writer should identify itself with this
Open KTX2 Issues

- Issues list at end of specification:
  http://github.khronos.org/KTX-Specification/
- Please look. I want your opinion.
- File suggestions and new issues at
  https://github.com/KhronosGroup/KTX-Specification
Status

- **KHR_texture_transmission v1.0 Provisional spec in Q4**
  - Universal supercompressed transcodable format
    - Target device transcodes to its own native GPU format in real-time
    - 'RDO mode' - allows target bitrate to be set (compression ratio vs quality)
    - Only lower precision formats supported for now [BC1-5, ETC1, PVRTC, ASTC (LDR subset)]

- **Incremental spec revisions**
  - Universal format for precision encoding and high dynamic range
    - Will support variable block size and hi-precision formats: ASTC, BC6H/BC7, HDR
    - New minor revisions will not introduce breaking changes

- **Call for Industry Collaboration!**
  - Actively seeking industry feedback and contributions
  - Development is open to the community - github links to specs, libs & samples to follow
  - Consider participating directly in the Khronos 3D Formats WG!
Reference DXT1 vs. CTTF Comparison Tests

- Quality comparisons of a reference DXT1 image vs equivalent CTTF encodings

  - Reference DXT1 image created with Crunch encoder
  - CTTF bitrates set to best match reference DXT1 quality
  - Test 1: Reference DXT1 vs CTTF ETC1S-CRN transcoder
  - Test 2: Reference DXT1 vs CTTF RDO+LZMA
  - Test 3: CTTF mode comparison: ETC1S-CRN vs RDO+LZMA
## DXT1 vs. CTTF ETC1S-CRN Transcoder

<table>
<thead>
<tr>
<th>Format</th>
<th>DXT1</th>
<th>CTTF ETC1S-CRN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmitted bits/pixel</td>
<td>4.0</td>
<td>1.8*</td>
</tr>
<tr>
<td>Stream Compression</td>
<td>6:1</td>
<td>13:1</td>
</tr>
<tr>
<td>PSNR (dB)</td>
<td>37.78</td>
<td>36.32</td>
</tr>
<tr>
<td>MSE¹</td>
<td>10.83</td>
<td>15.19</td>
</tr>
<tr>
<td>SSIM²</td>
<td>0.97</td>
<td>0.96</td>
</tr>
</tbody>
</table>

*Best bitrate setting to approx ref DXT1 PSNR

¹ Mean Squared Error
² Structured Similarity Index
DXT1 vs. CTTF RDO+LZMA (DXT1)

<table>
<thead>
<tr>
<th>Format</th>
<th>DXT1</th>
<th>CTTF RDO+LZMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmitted bits/pixel</td>
<td>4.0</td>
<td>2.1*</td>
</tr>
<tr>
<td>Stream Compression</td>
<td>6:1</td>
<td>11:1</td>
</tr>
<tr>
<td>PSNR (dB)</td>
<td>37.78</td>
<td>36.45</td>
</tr>
<tr>
<td>MSE</td>
<td>10.83</td>
<td>14.71</td>
</tr>
<tr>
<td>SSIM</td>
<td>0.97</td>
<td>0.95</td>
</tr>
</tbody>
</table>

*Best bitrate setting to approx ref DXT1 PSNR
CTTF Mode Comparison - ETC1S-CRN vs. RDO+LZMA (DXT1)

CTTF RDO mode + LZMA does not give as high compression perf as ETC1S-CRN. However, given that it is just an optimizing DXT encoder, the resulting texture can be used by existing workflows without modification.

ETC1S-CRN has better compression with no extra encoder stage needed, but requires work to integrate universal format support into existing workflows.

<table>
<thead>
<tr>
<th>Format</th>
<th>CTTF ETC1S-CRN</th>
<th>CTTF RDO+LZMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmitted bits/pixel</td>
<td>1.8</td>
<td>2.1</td>
</tr>
<tr>
<td>Stream Compression</td>
<td>13:1</td>
<td>11:1</td>
</tr>
<tr>
<td>PSNR (dB)</td>
<td>36.32</td>
<td>36.45</td>
</tr>
<tr>
<td>MSE</td>
<td>15.19</td>
<td>14.71</td>
</tr>
<tr>
<td>SSIM</td>
<td>0.96</td>
<td>0.95</td>
</tr>
</tbody>
</table>
Acknowledgements


Rich Geldreich, Binomial Core Technology Creator. Masochist.* 😁
* How else to describe someone with the willingness & stamina to wade through enormously long compressed texture format specs looking for commonalities? 😢

Alexander Suvorov, Unity Crunch support for ETC{,2} & performance improvements.


Alexey Knyazev, Independent glTF Integration Expert.

Patrick Cozzi, Cesium 3D Formats WG Chairman.
Watch these places for progress

Texture Compression GUI: https://github.com/KhronosGroup/gltf-Compressionator.
Samples & Test Data: https://github.com/KhronosGroup/gltf-Texture-Transmission-Samples.
Readable KTX2 specification: https://github.khronos.org/KTX-Specification/.
KTX software (currently only supports KTX1): https://github.com/KhronosGroup/KTX-Software.
Original Crunch GitHub Repo: https://github.com/BinomialLLC/crunch. Use the fork above.

Blogs

Alexander Suvorov on his improvements to Crunch to support ETC:

Rich Geldreich on the “universal format” and the transcoders
http://richg42.blogspot.com/2018/06/etc1s-texture-format-encoding.html
http://richg42.blogspot.com/2018/05/some-basis-baseline-universal-format.html
Supplemental Material
DXT1 vs. CTTF ETC1S-CRN Transcoding

- Quality and Risk Analysis

- Note how the ETC1S-CRN transcodable format compresses better than an RDO+LZMA DXT
- Highly compact - potentially up to 4x smaller than equivalent GPU format encoding
- Supercompression is gained through a set of entropy reduction stages + clusterization +VQ
- ETC1S-CRN also features an implicit RDO stage - so quality vs bitrate can be controlled
- Supporting ETC1S-CRN requires changes to existing asset pipelines and app code - some risk
- Unlike CTTF RDO mode, where the optimized DXT textures can be used as-is - no risk
DXT1 vs. CTTF RDO+LZMA mode

- Quality and Risk Analysis

  - Note performance slides showed RDO+LZMA mode performing WORSE than ETC1S-CRN!
  - ETC1S-CRN applies supercompression on encode, so no need for a lossless encoding stage
  - RDO mode adjusts bitrate of encoded DXT to desired level, for use with a lossless encoder
  - Works with existing workflows - no changes to asset pipelines or app code, zero risk
  - Unlike the ETC1S-CRN universal format, which would require integration into tools and apps
Bitstream Compression Performance

Transmission with standard DXT1 Encoding

Original

DXT Encode

DXT1

Diff

Size: 512 x 768
Format: RGB8 RAW
Buffer Size: 1152KB

Size: 512 x 768
Format: DXT1 RGBA
Buffer Size: 192 KB

Size: 512 x 768
Format: DXT1 RGBA
Buffer Size: 192 KB

MSE: 10.8  PSNR: 37.8dB
Stream Compress: 6.1
Bitstream Compression Performance

CTTF ETC1S-CRN bitstream with DXT1 transcode on receiver

Original

ETC1S-CRN

DXT1

Diff

Size: 512 x 768
Format: RGB8 RAW
Buffer Size: 1152KB

Size: 512 x 768
Format: ETC1S-CRN
Buffer Size: 67 KB

Size: 512 x 768
Format: DXT1 RGBA
Buffer Size: 192 KB

MSE: 15.19  PSNR: 36.32dB
Stream Compress: 13:1
Bitstream Compression Performance

CTTF RDO+LZMA lossless DXT1 bitstream with unpack on receiver

Original

Size: 512 x 768
Format: RGB8 RAW
Buffer Size: 1152KB

RDO+LZMA

Rate-Dist Clusterization
Generate DXT1 Texture
Izma pack
Transmission
Izma unpack
DXT1 Texture Copy

DXT1

Size: 512 x 768
Format: LZMA
Packed Size: 100K
Buffer Size: 192 KB

Diff

MSE: 0.95
PSNR: 36.45dB
Stream Compress: 11:1