
Vulkanised | Fall 2021
October 13-14 / VIRTUAL
KHRONOS GROUP
WEBINARS & MEETUPS
Vulkan

Texture compression in Vulkan

With KTX and Basis Universal

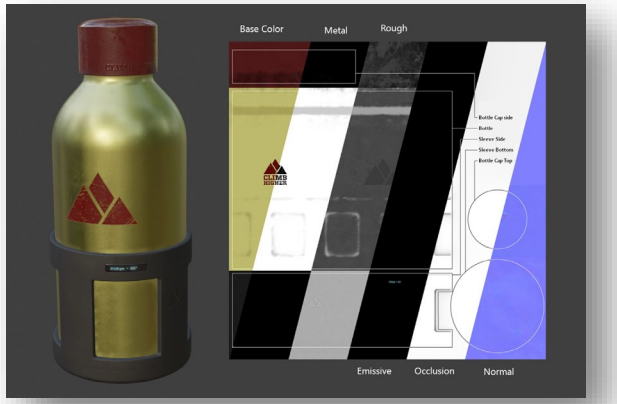
Sascha Willems

webmaster@saschawillems.de

https://twitter.com/SaschaWillems2

Texture Compression primer

- Since 1999 with S3 Texture Compression (S3TC)
- Indispensable due to
  - Larger and more detailed textures
  - Increasing number of material properties (e.g. PBR)
- Trades performance for image quality
  - Smaller file sizes (more on this later)
  - Faster download and loading times
  - Less VRAM bandwidth required
- Different formats for different use-cases
- Optimized for random access (unlike e.g. JPEG)



https://docs.blender.org/manual/en/2.80/addons/io_scene_gltf2.html

Texture Compression at the api level

- Various image formats (VK_FORMAT_...)
  - Block compression (BC1…BC7)
  - Adaptive scalable texture compression (ASTC)
  - Ericsson Texture Compression (ETC2)
  - PowerVR Texture compression format (PVRTC) as an extension
- Support varies between platforms
  - BC on desktop (except for Intel, which also has ASTC)
  - ASTC and ETC2 on mobile
  - PVRTC on PowerVR devices (VK_IMG_format_pvrtc extension)

Texture Compression coverage

https://vulkan.gpuinfo.org/listoptimaltilingformats.php

Windows

BC1_RGB_SRGB_BLOCK	99.81%
BC1_RGB_UNORM_BLOCK	99.81%
ASTC_10x10_SRGB_BLOCK	7.56%
ASTC_10x10_UNORM_BLOCK	7.56%
ETC2_R8G8B8_SRGB_BLOCK	9.11%
ETC2_R8G8B8_UNORM_BLOCK	9.11%

Android

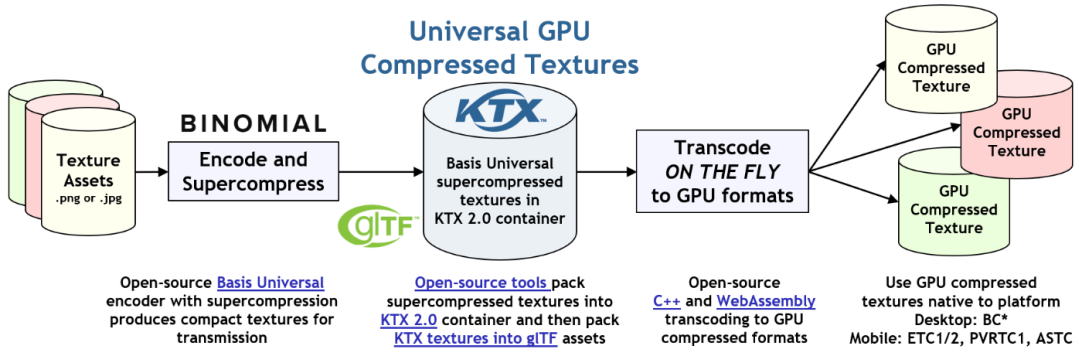
BC1_RGB_SRGB_BLOCK	3.91%
BC1_RGB_UNORM_BLOCK	3.91%
ASTC_10x10_SRGB_BLOCK	97.53%
ASTC_10x10_UNORM_BLOCK	97.53%
ETC2_R8G8B8_SRGB_BLOCK	99.48%
ETC2_R8G8B8_UNORM_BLOCK	99.48%

Texture Compression distribution options

- Distribute multiple versions, one for each compressed format
  - Increases disk space requirements
- Distribute one version and transcode
  - Transcoding from one lossy format to another (e.g. BC to ASTC) not advised
  - Need to ship multiple transcoders
  - Transcoders need to be fast
  - Need a transcode source with good quality
- Use a universal source format
  - This is where Basis Universal comes in…

Basis Universal Supercompressed Codec

- Supercompressed GPU texture data intercharge system
- Open Source, developed by Binomial
- Two universal texture modes
  - ETC1S – Optimized for small file sizes, with low/medium quality
  - UASTC – Optimized for quality, usable for all texture map types
- Serves as a source for transcoding to native formats
  - Transcoding is very fast
  - Quality maps to native formats (e.g. BC7)

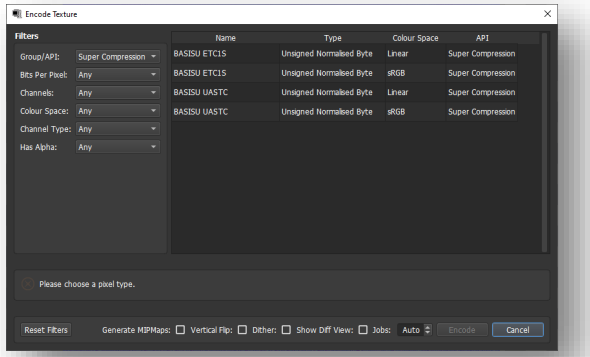


Basis Universal numbers

- Full 1k x 1k Crytek Sponza Texture set (color and normal maps)
- File sizes (roughly matches VRAM consumption)
  - Uncompressed RGBA ~362 Mbytes
  - UASTC ~91 Mbytes (optimized for image quality)
  - ETC1s ~9 Mbytes (optimized for small file sizes)
- Speed
  - UASTC to BC7 transcoding of full set
  - ~160ms on an AMD Ryzen 5 3600 (12 Threads)
  - Speed important for background streaming

KTX GPU Texture Container Format

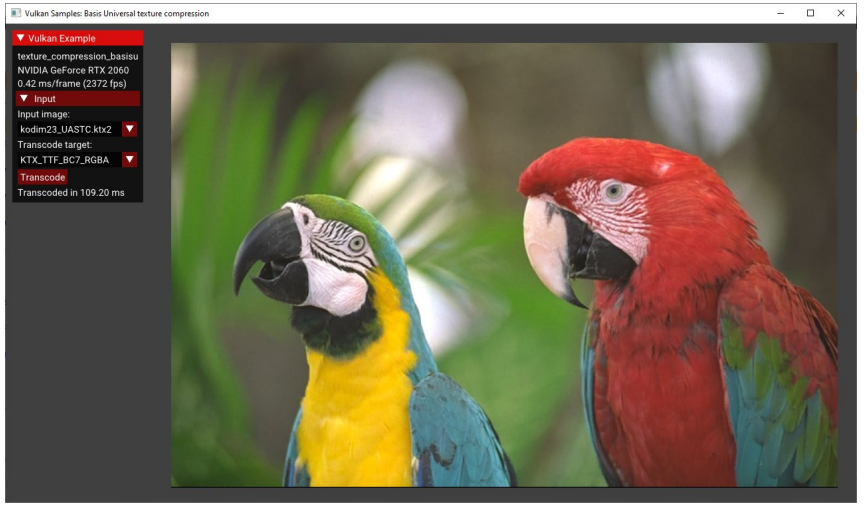
- https://www.khronos.org/ktx/
- Lightweight container format for textures
- Can store all texture types (2D, 3D, cubemaps, arrays)
- Supports compressed and uncompressed formats
- KTX 2.0 natively supports Basis Universal supercompressed textures
- Can be included into your application as a library (KTX-Software)
- Content creation tools
  - PVRTexTool - Large feature set, incl. BasisU
  - NVIDIA Texture Tools Exporter (no BasisU yet)



Transcoding in C++ using KTX-Software

```
#include <ktx.h>

// Load
ktxTexture2 *ktx_texture;
KTX_error_code result = ktxTexture_CreateFromNamedFile(file_name, KTX_TEXTURE_CREATE_LOAD_IMAGE_DATA_BIT, (ktxTexture**)&ktx_texture);

// Select format depending on support
ktx_transcode_fmt_e target_format = KTX_TTF_RGBA32;
if ((device_features.textureCompressionBC) && (fmt_supported(VK_FORMAT_BC7_SRGB_BLOCK)))
{
  target_format = KTX_TTF_BC7_RGBA;
}


// Transcode
if (ktxTexture2_NeedsTranscoding(ktx_texture))
{
  ktxTexture2_TranscodeBasis(ktx_texture, target_format, 0);
}

// Vulkan image format
VkFormat format = (VkFormat) ktx_texture->vkFormat;

// Upload
VK_CHECK(vkMapMemory(device, staging_memory, 0, memory_requirements.size, 0, (void **) &data));
memcpy(data, ktx_texture->pData, ktx_texture->dataSize);
```

Vulkan Sample



- https://github.com/KhronosGroup/Vulkan-Samples
- Cross-platform C++ sample using KTX-Software
- Transcode from UASTC or ETCS1 to native formats
- Compare quality and performance differences
- Includes a readme with a small tutorial on how to transcode and upload data to the GPU

Closing words

- Make use of texture compression
- Small effort with big gains
- The inner workings of texture compression is a large area of research
- But luckily, no knowledge is required if you just want to use it
- Pretend that JPG and PNG don’t exist ;
  - Decoding slow
  - Only 2D images
  - No mip maps

Links

- https://github.com/KhronosGroup/3D-Formats-Guidelines/blob/main/KTXDeveloperGuide.md
- https://github.com/KhronosGroup/KTX-Software
- https://github.com/BinomialLLC/basis_universal
- https://github.com/BinomialLLC/basis_universal/wiki/How-to-Use-and-Configure-the-Transcoder
- https://developer.imaginationtech.com/downloads/
- https://developer.nvidia.com/nvidia-texture-tools-exporter
- https://www.binomial.info/
- https://sv-journal.org/2014-1/06/en/index.php?lang=en
- https://www.reedbeta.com/blog/understanding-bcn-texture-compression-formats/#bc4
- https://aras-p.info/blog/2020/12/08/Texture-Compression-in-2020/