Improving Texture Compression in Games

AMD Capsaicin & Cream Event
Stephanie Hurlburt

CAPSAICIN & Cream

Introduction

I’m an owner and graphics engineer at Binomial, along with my business partner Rich Geldreich. We’re developing Basis, a texture compressor.

Basis can be useful in all kinds of apps: web applications, console games, mapping software, mobile apps, XR, light fields, video, animated textures, and more.

For example, Netflix uses Basis across its varied pipeline on both mobile and console platforms.

Overview of current texture compression pipelines

Download a JPEG, at runtime decompress and store uncompressed on GPU

- Great CPU memory usage, but poor GPU memory usage
- Processing required at runtime to decompress JPEG

GPU formats were created to replace this pipeline, so let’s take a look at pipelines that utilize GPU formats next.

Overview of current texture compression pipelines

Download a JPEG, at runtime decompress the JPEG then encode to a GPU format

- Good memory usage on GPU and CPU
- Runtime processing required to decompress the JPEG and encode to a GPU format
- Encoding to a GPU format at runtime can be slow and produces poor quality images

Overview of current texture compression pipelines

Download supercompressed texture, transcode to GPU format

- Good GPU and CPU storage
- CPU storage comparable to JPEG
- Runtime cost of transcoding, faster than lossless decompression

Download GPU format + lossless compression, decompress and send to GPU

- Good GPU memory usage, not the greatest CPU storage
- GPU formats don't losslessly compress well, and since GPU formats are lossy themselves, there isn't a strong need for lossless compression
- Runtime cost of lossless decompression

Very important note on multiplatform titles

Multiplatform games currently have to ship one GPU format per GPU they support

- DXT1-5, BC1-5: Desktop, consoles
- BC6-7: High end desktop/consoles (better quality)
- ETC1, ETC2, ASTC: Android, Metal-compatible iOS, most mobile, embedded
- PVRTC: iOS (required on devices without Metal support)

Basis (which is a type of supercompressed texture) is the only GPU-friendly solution that is also truly multiplatform.

Overview of Basis: RDO Mode

Outputs an optimized .ktx/.dds file that is then intended for further lossless compression. Lossless decompress at runtime.

Easy to fit into existing pipelines that use GPU format + lossless compression.

This is worse than .basis, but is possible to provide further optimizations if we know the lossless codec you'll use.

RDO stands for "rate distortion optimization." We provide a quality slider, allowing you to decide how much quality you're willing to give up for size reduction. (RDO is also used in .basis)

Overview of Basis: Using the .basis file format

Compress to .basis offline, then at runtime transcode to the GPU format you need.

Great for multiplatform-- just one file, even if you need to support many GPU formats.

Transcoding to the GPU format is much faster than lossless decompression, since .basis can transcode at the block level.

Better compression ratios than RDO mode because we aren’t just using a generic lossless codec, further compression can be extra optimized.

This method also has a quality slider.

Basics of Basis

We still keep the concept of GPU formats, they’re now just abstracted away from game developers.

The .basis format is still based on blocks, but is a format that is generalized enough so that it can be transcoded to any GPU format at very fast speeds. Because it’s block-based, much faster than most lossless decompression.

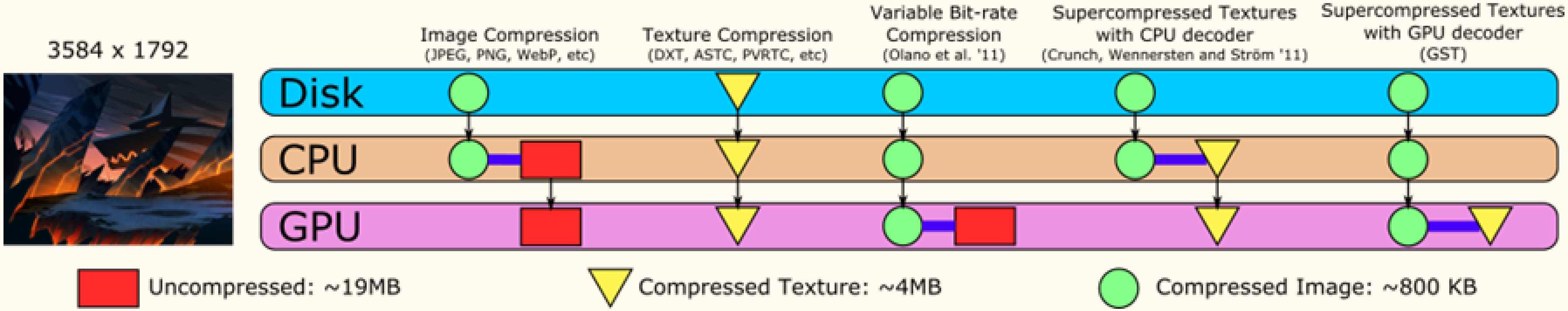
Currently supporting ETC1 and DXT. Plan is to support all relevant GPU formats.

With RDO mode, you can take advantage of certain optimizations in the .basis encoder for better GPU format output, and we can optimize further if we know the lossless codec you’ll be using.

What data does Basis work on?

- Single textures
  - Photographs, UI, normal maps, depth data, and more
- Mipmaps
- Cubemaps
- Texture arrays
  - Video
  - Animated textures
  - Light fields

Supercompressed texture research



GST: GPU-decodable Supercompressed Textures

Pavel Krajcevski, Srihari Pratapa, Dinesh Manocha

http://gamma.cs.unc.edu/GST/

Supercompressed textures: Crunch

An open source supercompressed texture library for DXT (and ETC in limited capacity) formats, written by Rich and maintained by Binomial.

https://github.com/BinomialLLC/crunch

Used extremely widely in games, video, map applications, and beyond.

Binomial will also be supporting the LZHAM codec, also written by Rich and often paired with crunch's RDO mode:

https://github.com/richgel999/lzham_codec

Supercompressed textures: Basis

Basis takes the concept of supercompressed textures and creates a standardized format that can be transcoded to any GPU format.

For .basis file format use, transcoding happens CPU-side now. A GPU transcoder can be created. We are currently exploring other GPU solutions too, potentially in drivers and hardware.

Efforts to standardize supercompressed textures

It’s worth mentioning that we are actively working with the 3D Formats Working Group within the Khronos Group

We’re planning to standardize the .basis file format and open up the transcoder so that anyone can target Basis. We would love to see transcoder implementations from vendors, especially GPU driver and hardware implementations.

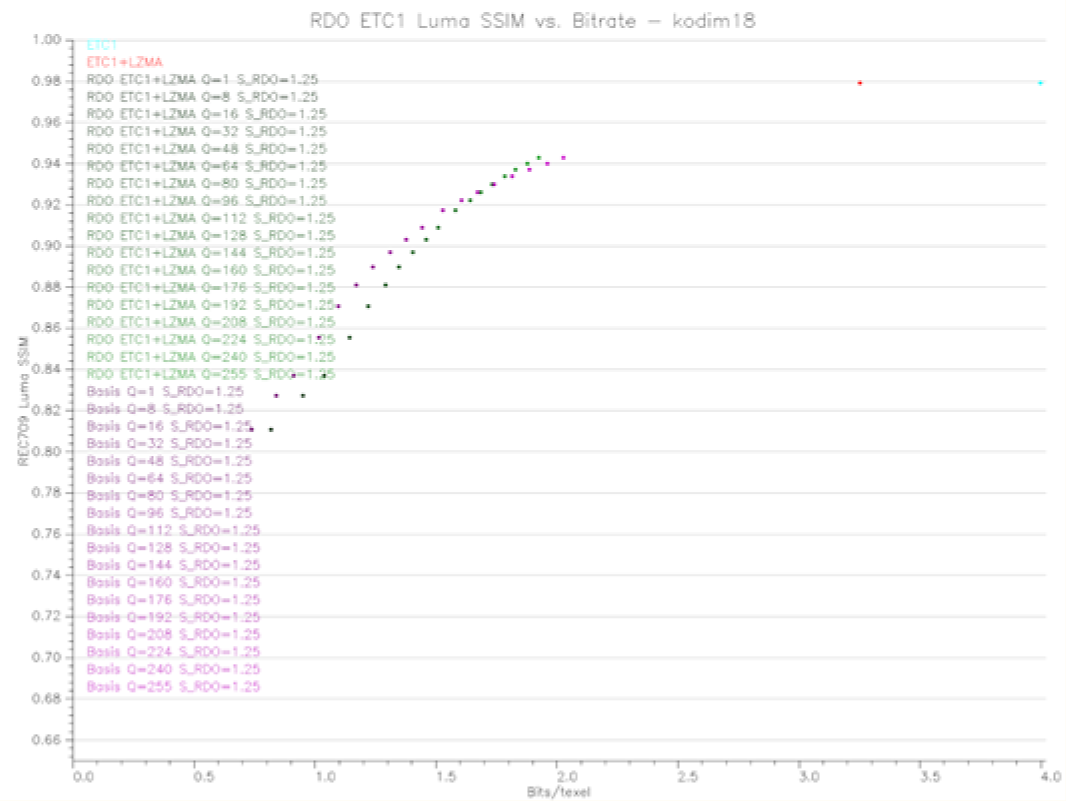
We hope this standardization will allow vendors to dream up innovative new GPU formats while maintaining one optimal texture standard (Basis) for game developers to target.

Performance statistics

GPU Format: 4.0 bits/texel

GPU Format + lossless: 3.4 bits/texel

Basis: 0.6-2.4 bits/texel

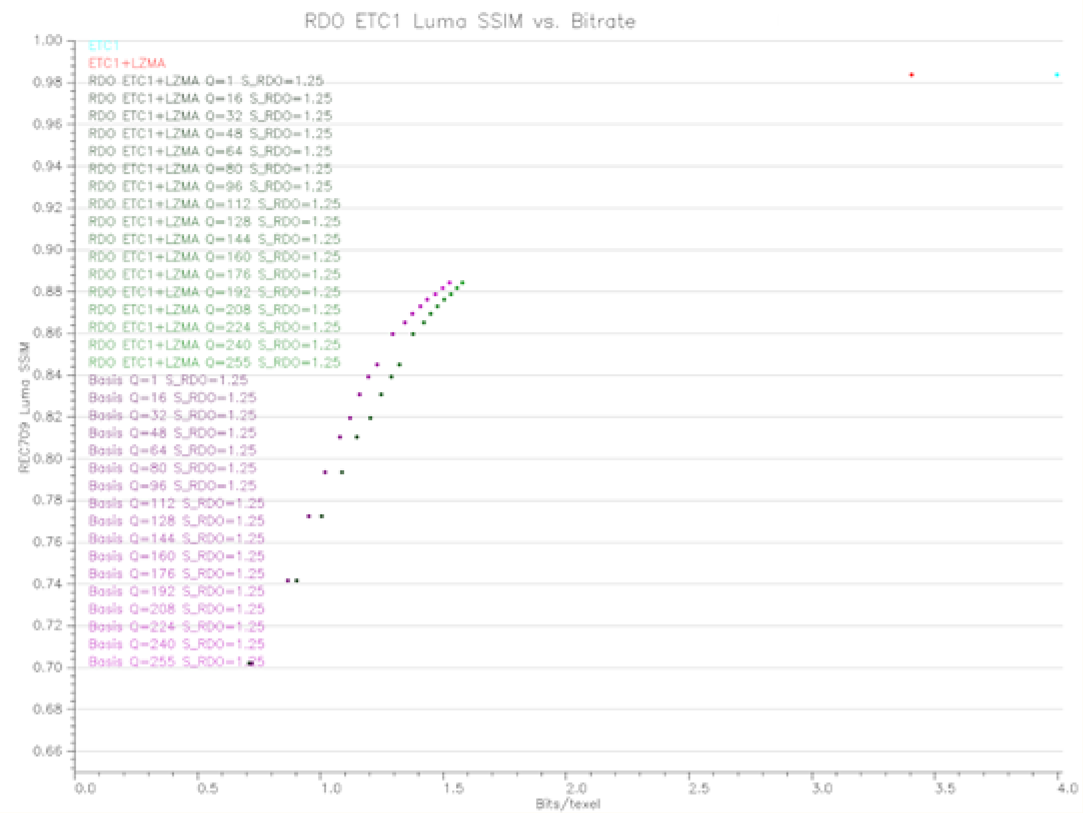


Performance statistics

GPU Format: 4.0 bits/texel

GPU Format + lossless: 3.6 bits/texel

Basis: 0.7-1.8 bits/texel

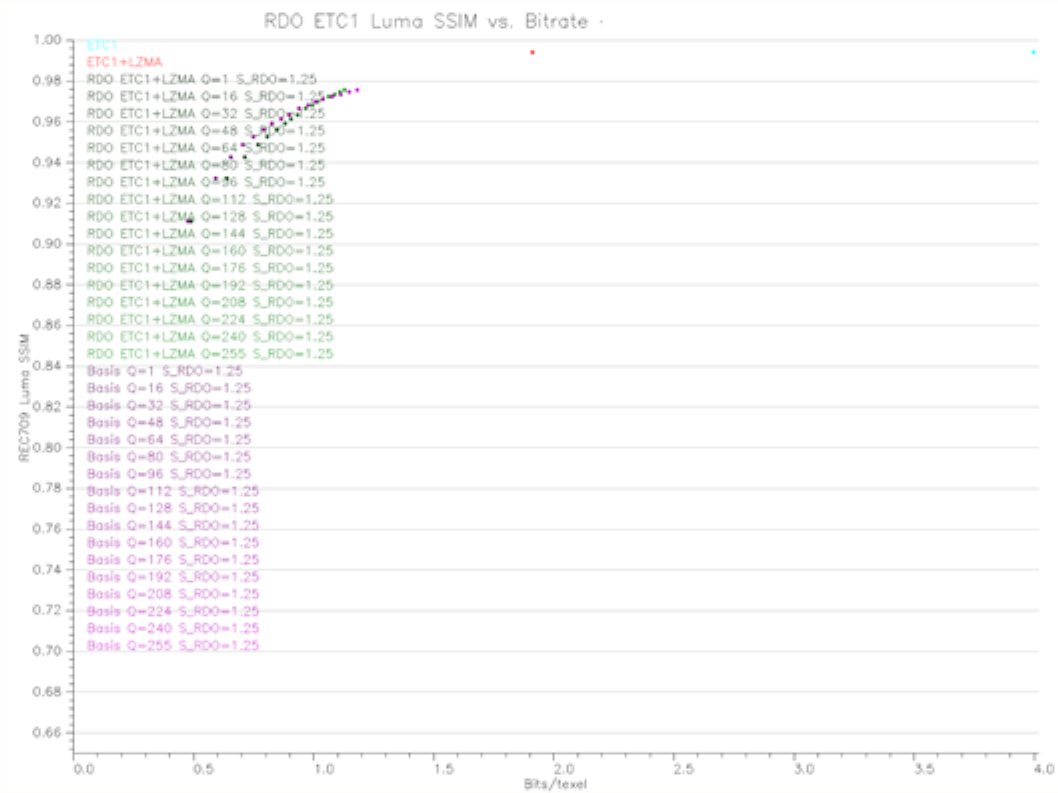


Performance statistics

GPU Format: 4.0 bits/texel

GPU Format + lossless: 2.1 bits/texel

Basis: 0.6-1.4 bits/texel

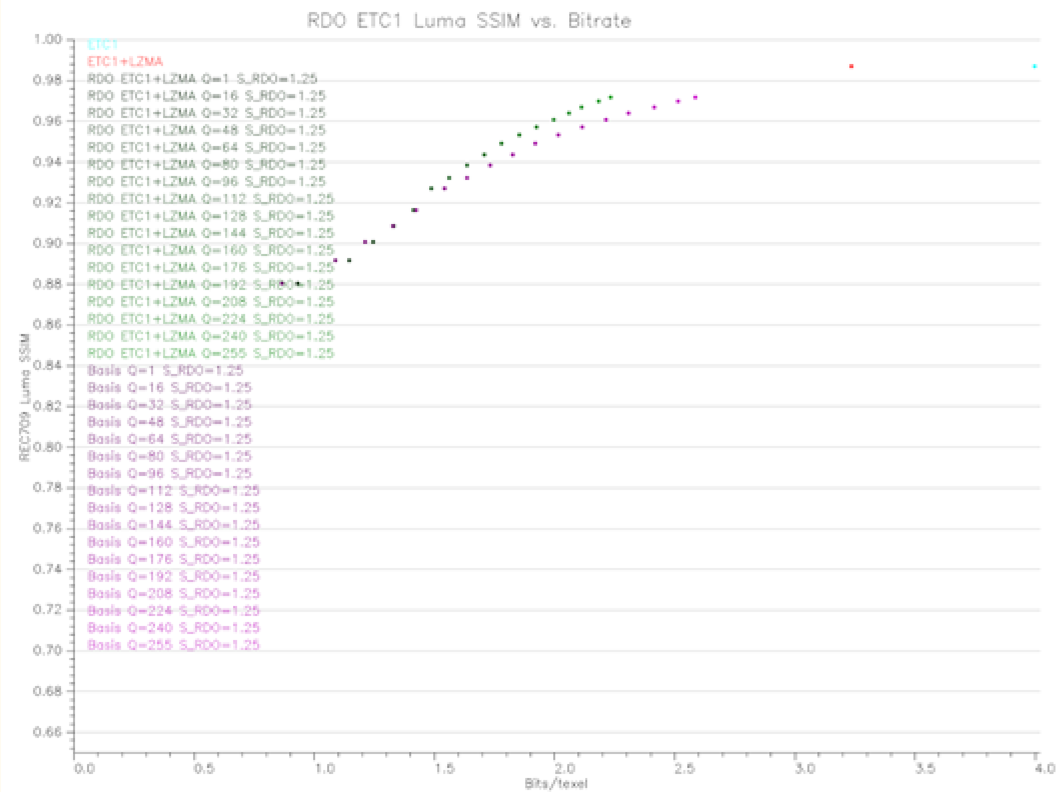


Performance statistics

GPU Format: 4.0 bits/texel

GPU Format + lossless: 3.4 bits/texel

Basis: 0.4-2.8 bits/texel



Thank you!

@sehurlburt

binomial.info