open Source days '23

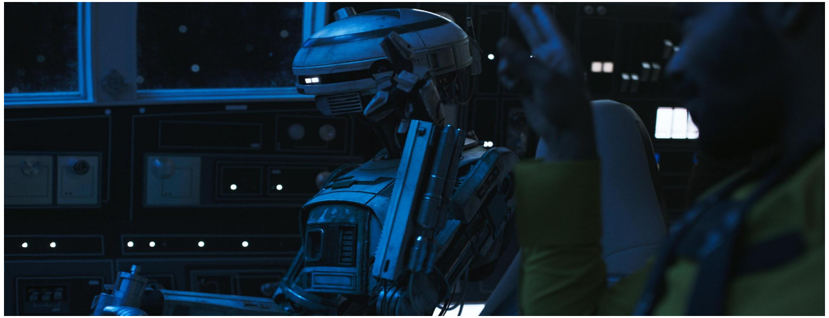
/* ACADEMY SOFTWARE FOUNDATION

Virtual Town Hall Series

August 1, 2023

OpenEXR

OpenEXR



Images from *Solo: A Star Wars Story.* © and TM Lucasfilm Ltd. All Rights Reserved.

- Professional-grade High Dynamic Range image storage format of the motion picture industry, first released in 2003.
- Key contributors: ILM, Weta Digital, DreamWorks, lots of other
- Joined ASWF in 2019

https://openexr.com

https://github.com/AcademySoftwareFoundation/openexr

https://github.com/AcademySoftwareFoundation/Imath

https://github.com/AcademySoftwareFoundation/openexr-images

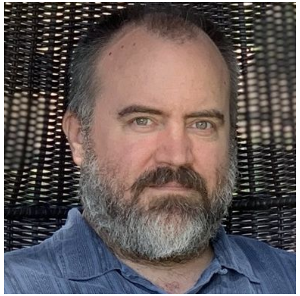
OpenEXR

OpenEXR Technical Steering Committee

Cary Phillips
Industrial Light & Magic

Christina Tempelaar-Lietz
Industrial Light & Magic

Joseph Goldstone
ARRI, Inc



Kimball Thurston
Wētā FX

Larry Gritz
Sony Pictures Imageworks

Nick Porcino
Pixar Animation Studios

Peter Hillman
Wētā Digital x Unity

Rod Bogart
Epic Games

Agenda

- OpenEXR v3.1 Review
- OpenEXR v3.2 Preview
- Discussion topics:
    - Experiments in GPU Decompression & Real-time Streaming

OpenEXR v3.1

/* ACADEMY SOFTWARE FOUNDATION
#ASWF

- OpenEXRCore, now with DWAA/B compression support
- Performance optimizations (zip, neon, huf decoder, SSE4)
- Bug/build fixes
- New https://openexr.com website, with test images

Other developments…

- MacPorts is now up to date (v3.1.9)
- PyPI python bindings (a.k.a. “pip install openexr”)
- OpenEXR & USDZ
- Coalition for Content Provenance and Authenticity (C2PA)

OpenEXR v3.2 (VFX Ref. Platform 2024)

- Lossless compression via libdeflate (replaces zlib)
  - New dependency
- ABI compatibility
- New standard optional camera metadata

v3.2: New Optional Standard Metadata

`sensorCenterOffset`

Horizontal and vertical distances, in microns, of the center of the light-sensitive area of the camera's sensor from a point on that sensor where a sensor surface normal would intersect the center of the lens mount.

`sensorOverallDimensions`

Dimensions of the light-sensitive area of the sensor, in millimeters, independent of the subset of that region from which image data are obtained.

`sensorPhotositePitch`

Distance between centers of sensor photosites, in microns.

`sensorAcquisitionRectangle`

The rectangular area of the sensor containing photosites the contents of which are in one-to-one correspondence with the captured sensels, for a monochrome sensor, or with the reconstructed RGB pixels, for a sensor covered with color filter array material in a Bayer or a similar pattern.

v3.2: New Optional Standard Metadata

Support automated editorial workflow

`reelName`, `imageCounter`, `ascFramingDecisionList`

Support forensics ("which other shots used that camera and lens before the camera firmware was updated?")

`cameraMake`, `cameraModel`, `cameraSerialNumber`, `cameraFirmware`, `cameraUuid`, `cameraLabel`

`lensMake`, `lensModel`, `lensSerialNumber`, `lensFirmware`

`cameraColorBalance`

Support pickup shots (reproduce critical camera settings)

`shutterAngle`, `cameraCCTSetting`, `cameraTintSetting`

Support metadata-driven match move

`sensorCenterOffset`, `sensorOverallDimensions`, `sensorPhotositePitch`, `sensorAcquisitionRectangle`

`nominalFocalLength`, `effectiveFocalLength`, `pinholeFocalLength`, `entrancePupilOffset`

`tStop` (complementing existing `aperture`)

v3.2: New Optional Standard Metadata

Tracking current work in SMPTE

SMPTE is trying to be more agile, and virtual production makes a good test case for their efforts; there's an overview blog post on their site:

Update on SMPTE's Rapid Industry Solutions (RIS) On-Set Virtual Production (OSVP) Initiative

SMPTE Content
September 19, 2022

SMPTE's Rapid Industry Solutions (RIS) program was designed to give the Society an agile, responsive framework through which to address emerging industry needs. Working with partners across the industry, SMPTE identified on-set virtual production (OSVP) as a suitable subject for its first RIS initiative, the SMPTE RIS OSVP initiative, which launched one year ago

… that you can find at

https://www.smpte.org/blog/update-on-smptes-rapid-industry-solutions-ris-on-set-virtual-production-osvp-initiative

v3.2: New Optional Standard Metadata

Real-world camera and lens metadata is messy

- Camera vendors provide metadata with their own names, their choice of data type, their choice of unit, and perhaps most importantly their own semantics

SMPTE offers metadata normalization definitions

- Anyone with a GitHub account can comment on them: https://github.com/SMPTE/ris-osvp-metadata/

SMPTE works with camera vendors to show how their metadata is mapped to SMPTE RIS OSVP metadata

- Anyone with a GitHub account can file an Issue or submit a PR: https://github.com/SMPTE/ris-osvp-metadata-camdkit
- ARRI, Blackmagic Design, Canon, RED, Sony contributions so far

v3.2: New Optional Standard Metadata

The new metadata leverage a very good white paper from Cooke Optics establishing lens terminology for VFX

The definitions in this paper are used in the SMPTE RIS for OSVP `camdkit` project just mentioned

Camera and lens definitions for VFX 21/07/23

Contents

Introduction ... 3
2.0 There are three different definitions of focal length ... 4
2.1 Pinhole focal length ... 5
2.2 Optical models of lenses ... 6
2.3 Talking to optical designers about pinhole focal length ... 9
2.4 Practical implications ... 10
3.0 The mount is the best physical reference point ... 11
4.0 Definitions ... 12
5.0 References ... 13

… that you can find at

https://cookeoptics.com/wp-content/uploads/2023/07/Cooke-Camera-Lens-Definitions-for-VFX-210723.pdf

v3.2: New Optional Standard Metadata

What's next for optional standard metadata?

OpenEXR will continue to track SMPTE's work in support of virtual production

2023 and early 2024 will likely see some recommendations in support of multiple lens distortion models

Example code relating a well-characterized physical camera on a well-described set to a CG camera described by the existing Renderman-inspired `worldToCamera` and `worldToNDC` attributes

Clear out two optional standard attributes we've never seen in the wild, as a better alternative is available now: Use the ACES Metadata File (AMF) instead of `renderingTransform` and `lookModTransform`

Discussion

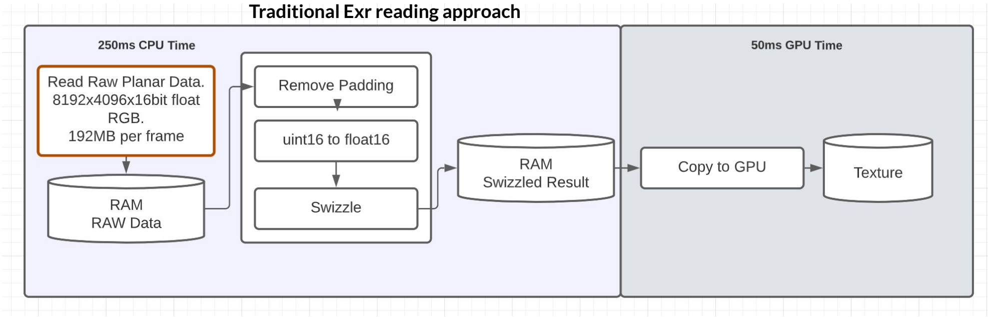
- Experiments in GPU Decompression & Real-time Streaming

Discussion: GPU-based decompression

EPIC GAMES

The motivation

- Uncompressed EXRs occupy a large portion of disk space.
- Unreal Engine utilizes EXR to play high quality HDR sequences. 30 seconds of uncompressed 8k RGB roughly equates to 140GB.
- Alternatively compressed EXRs require CPU decompression and buffer juggling. CPU Decompression is slow.
- Uncompressed data has large PCIe bandwidth requirements. 8192x4096 RGB 16 bit per channel takes ~45ms to copy from RAM to vRAM on Nvidia's A6000 GPU.
- In addition to substantial PCIe throughput users need high disk read speeds. To stream 8k Uncompressed EXR at 24fps users need ~4.8 GB/s read speed.
- These performance requirements stack with the rendering requirements of large Unreal Engine scenes.

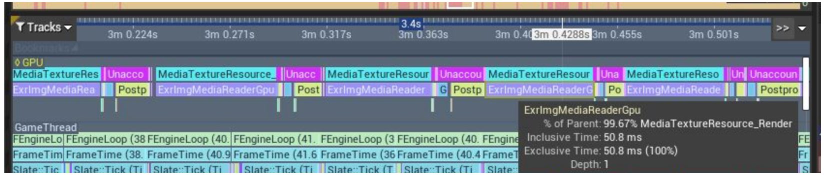


Bare-bones Uncompressed EXR

- Read into staging buffer in shared memory (RAM).
- Utilize directx 12 copy queue to asynchronously transfer from shared memory to vRAM Structured buffer.
- Process data into texture as before
- 15fps full texture read if copy time is taken into account. 8192x4096 16 bit RGB
- We increase fps further by loading only the required tiles.

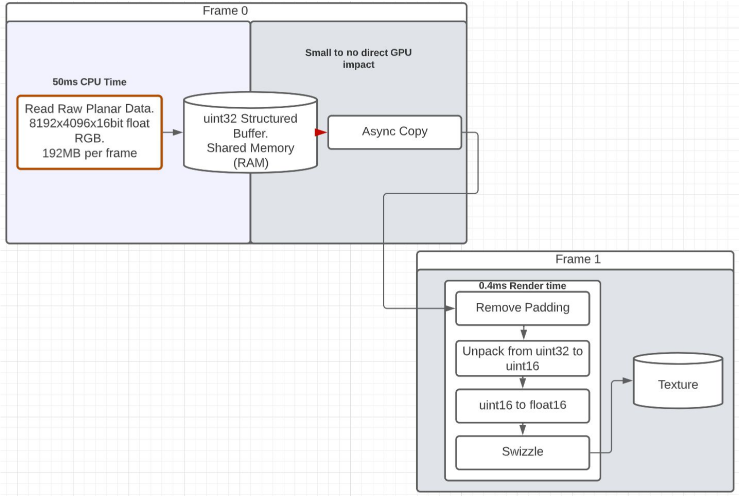
Remaining Problem:

- Disk space and disk load. 30 second 8k 16bit sequence takes up 140GB of the hard drive.
- Slow throughput (Hopes for PCIe 5.0 and 6.0)



A large portion of each frame is consumed by reading and delivering data to GPU

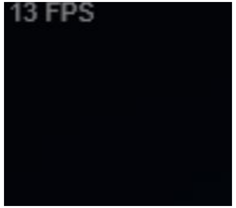
Current UE5 approach.



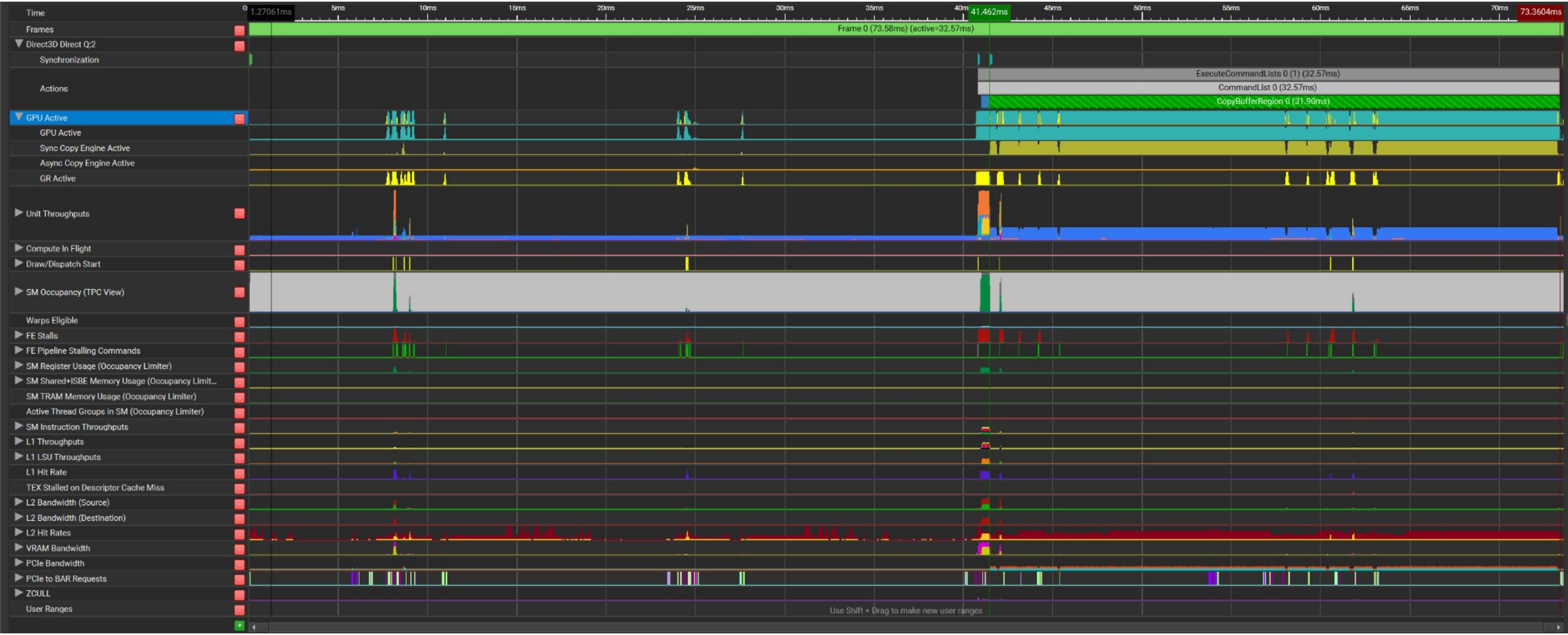
How it looks

8192x4096 RGB 16 bit per channel (192MB per frame) @13-15fps 2.4~3GB/s

- 35-50ms to read into shared memory (system memory).
- 40ms to copy to vRAM.
- 0.4ms to re-arrange encoded EXR Into a texture
- Total frame time is roughly 75-80ms or 13fps
- With async copy frame time can be improved.



How it looks



Time
Frames
Direct3D Direct Q:2
Synchronization
Actions
GPU Active
GPU Active
Sync Copy Engine Active
Async Copy Engine Active
GR Active
Unit Throughputs
Compute In Flight
Draw/Dispatch Start
SM Occupancy (TPC View)
Warps Eligible
FE Stalls
FE Pipeline Stalling Commands
SM Register Usage (Occupancy Limiter)
SM Shared+ISBE Memory Usage (Occupancy Limit...
SM TRAM Memory Usage (Occupancy Limiter)
Active Thread Groups in SM (Occupancy Limiter)
SM Instruction Throughputs
L1 Throughputs
L1 LSU Throughputs
L1 Hit Rate
TEX Stalled on Descriptor Cache Miss
L2 Bandwidth (Source)
L2 Bandwidth (Destination)
L2 Hit Rates
VRAM Bandwidth
PCIe Bandwidth
PCIe to BAR Requests
ZCULL
User Ranges
1.27061ms
41.462ms
73.3604ms
Frame 0 (73.58ms) (active=32.57ms)
ExecuteCommandLists 0 (1) (32.57ms)
CommandList 0 (32.57ms)
CopyBufferRegion 0 (31.90ms)
Use Shift + Drag to make new user ranges

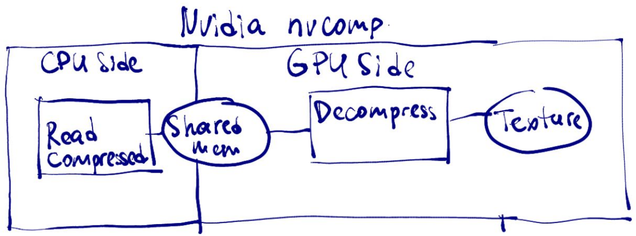
NVIDIA's GDeflate

- NVIDIA introduced GDeflate compression/decompression method that is a GPU friendly extension of deflate (zip).
- It is accessible via nvcomp library.

- CPU side fallback.
- NVIDIA has a fork of libdeflate library with addition of GDeflate (https://github.com/NVIDIA/libdeflate).
- It is cross platform (Unix, MacOS, Windows).

Main Advantages:

- Utilizes GPU for decompression and reduces PCIe bandwidth requirements. (it takes ~50ms to transfer 200MB of data to GPU via PCIe 5.0 Official transfer rate is ~4GB/s per lane. PCIe 6 is supposedly better.)
- Reduces disk speed and storage requirements.



Direct Storage with GDeflate

- Microsoft's Direct Storage API allows to read directly from nvme to vRAM.
- Allows to bypass slow system memory.
- It implements Nvidia’s GDeflate compression/decompression on GPU (with caveats).

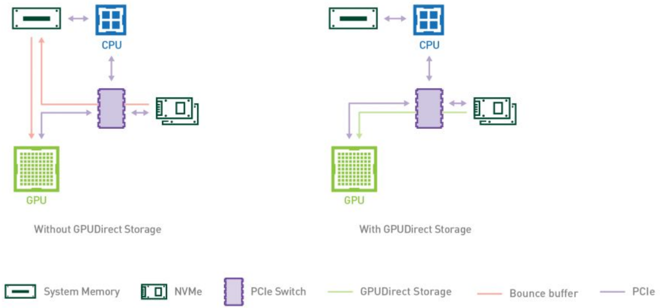
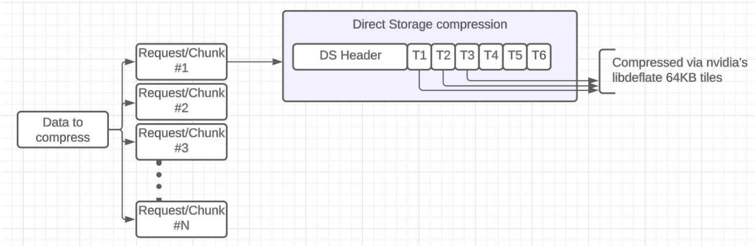


Image Source: Nvidia

Required changes in EXR storage

- To bypass system memory and utilize GPU decompression via Direct Storage (DS) EXRs will need to be compressed and stored in a certain way.
- DS is most efficient when there are multiple read requests issued (there could be too little or too many requests).
- DS compression has additional steps above nvidia's.
- Adds its own header with compression information per each request.
- Compresses data as 64KB tiles.

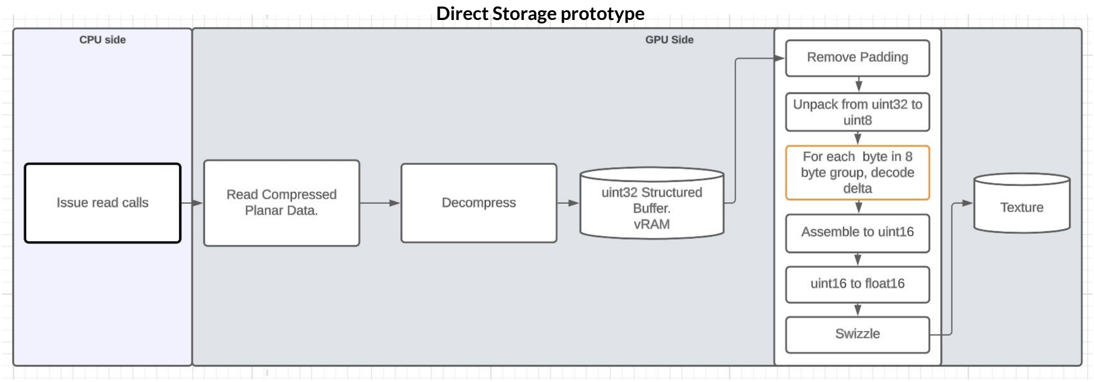
Most efficient approach to DS read



DS Header:
id
magic
numTiles
fileSizeIdx
lastTileSize
padding

GDeflate Image via Direct Storage

- Performance went from 15 to 23fps. 8192x4096 16bit RGB.
- Less system memory usage.
- Less CPU work.
- Reduced PCIe bandwidth.
- Less storage occupied, storage drive utilization is reduced allowing for more reads.

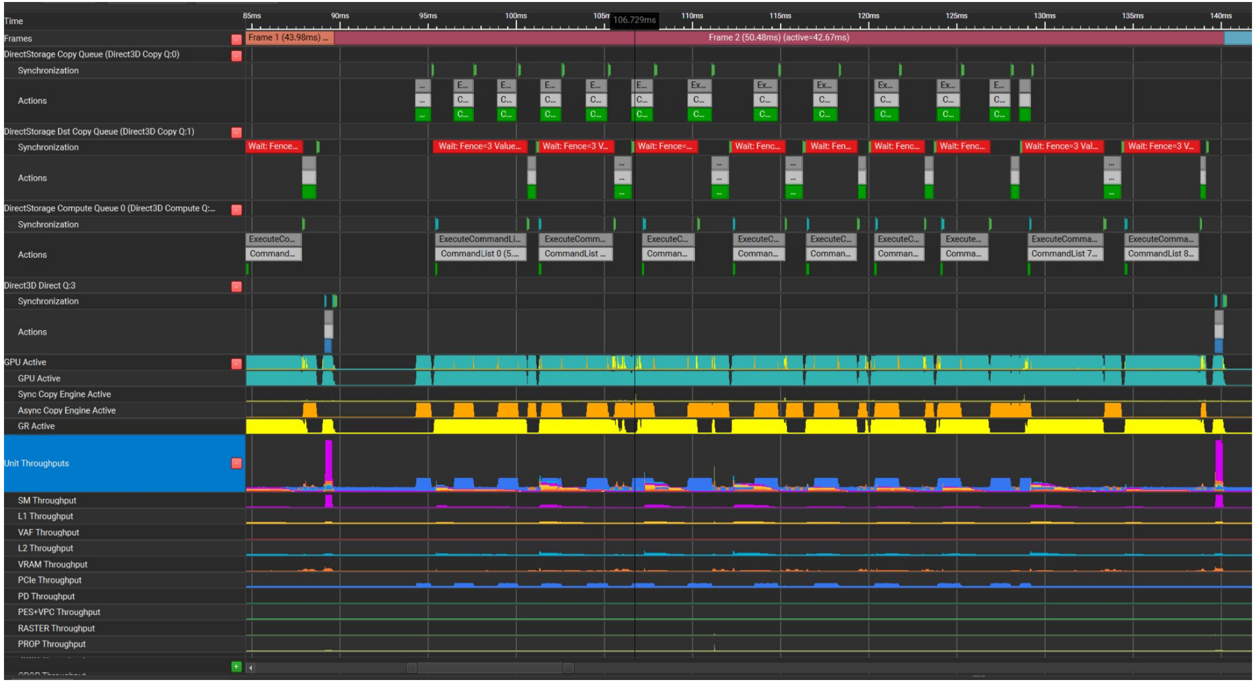


GDeflate Benchmark

/* ACADEMY SOFTWARE FOUNDATION

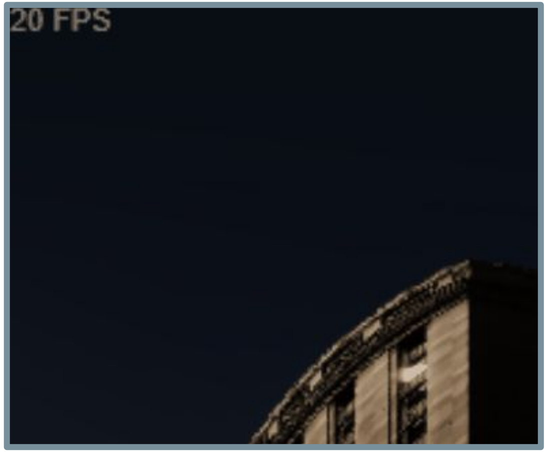
#ASWF

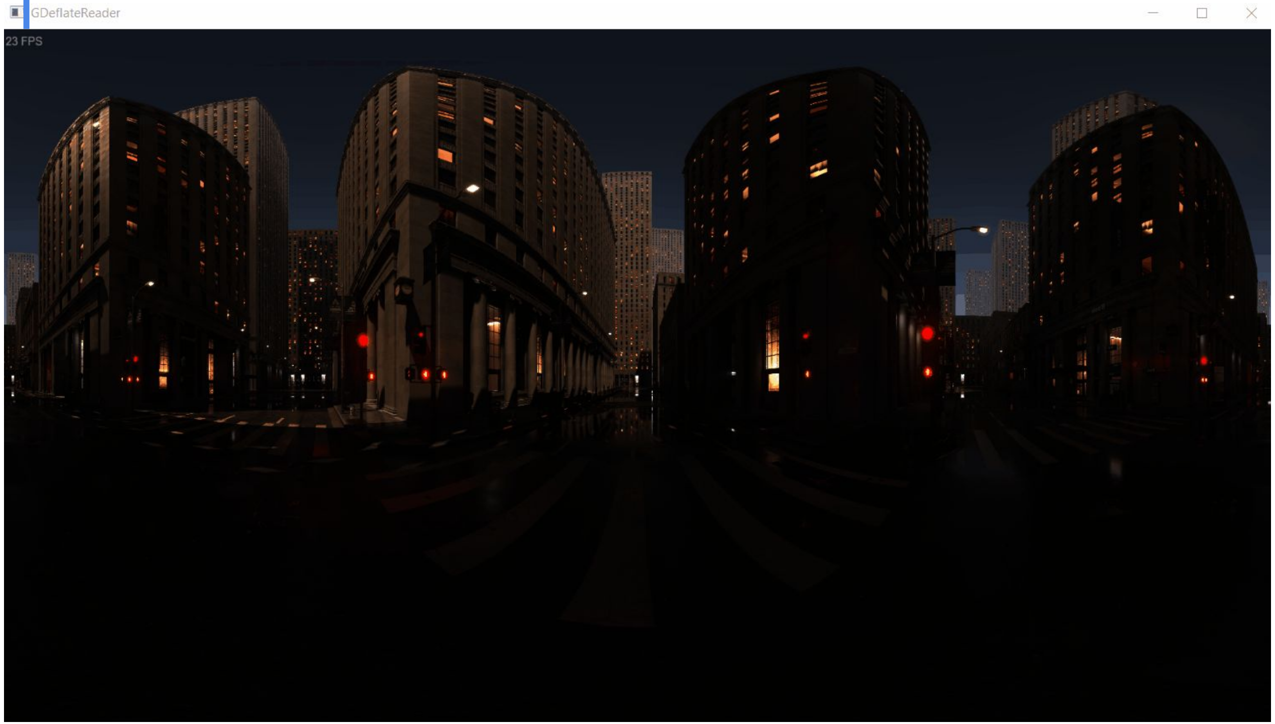
- Best compression ratio 1:2
  Reduces storage requirements.
- PCIe throughput requirements are halved.
- CPU time is negligible.
- Compared to simplest uncompressed exr approach total frame time is reduced from 70ms to 50ms (for 8192x4096 RGB 16 bit per channel exr). 15 to 23fps.



The final result

GDeflate 8192x4096xRGB 16bit per channel EXR





GPU Direct + GDeflate (Direct Storage) Benefits Summary

- Reduced file size, therefore reduced drive speed and storage requirements by a factor of 2.
- Reduced data needed to be transferred to GPU, lessens PCIe throughput requirements.
- GPU decompression.
- Little CPU time required.
- Bypassing System memory and reducing number of buffers allocated and utilized.

Call for Community Contributions

- Explore new compression schemes
- Performance metrics
- Example images:
  https://github.com/AcademySoftwareFoundation/openexr-images
- Support for bfloat16
- PyBind11 for Imath (retire boost dependency)
- Windows build support!

Thank you!

https://openexr.com

#openexr

openexr-dev@lists.aswf.io