INDEXED 3D SCENE LAYERS (I3S) – AN EFFICIENT ENCODING AND STREAMING OGC COMMUNITY STANDARD FOR MASSIVE GEOSPATIAL CONTENT

Tamrat Belayneh
tbelayneh@esri.com
Esri, CA

KEY WORDS: I3S, 3D Objects, IntegratedMesh, BVH, Massive Mesh, OGC, Compressed Texture (Basis Universal in KTX™2.0)

ABSTRACT:

Indexed 3D Scene Layers (I3S), an OGC Community Standard for streaming and storing massive amounts of geospatial content has been rapidly evolving to capture new use cases and techniques to advance geospatial visualization and analysis. I3S enables efficient transmission of various 3D geospatial data types including discrete 3D objects with attributes, integrated surface meshes and point cloud data covering vast geographic areas as well as highly detailed BIM (Building Information Model) content, to web browsers, mobile apps and desktop.
In this paper, we will explore multiple evolutions of I3S, including the latest, OGC I3S Version 1.2, that brings dramatic improvements in performance and scalability. We will demonstrate the advantages, including its support for a paged node access pattern, a more compact geometry layout, advanced material definitions property that supports PBR, as well as its support for supercompression of texture data using the Basis Universal SuperCompressed Texture format in KTX™2.0 containers. We will also demonstrate collaborative & research work done to dramatically improve in Basis compressed texture creation in KTX™2.0 container leveraging both the CPU and GPU – contributions that benefit both the geospatial and 3d graphics communities.
The paper will conclude by highlighting and documenting various use cases and application, where formats such as I3S are pushing the envelope in geospatial technology – by enabling seamless and ubiquitous access to vast amounts of geospatial data which traditionally have required specialized hardware and software platforms.

1. GENESIS

1.1 Evolution of a Standard

Indexed 3D Scene Layers (I3S), a specification developed by Esri for streaming textured mesh and point cloud dataset, has become one of the widely used format for disseminating massive geospatial content. I3S Supports various layer types including 3D object and IntegratedMesh – allowing the streaming of millions of 3D objects and high fidelity meshes, as well as Point Cloud Scene Layer – enabling the streaming of point cloud data consisting of billions of points. I3S has also added support for Building Scene Layer – unlocking complex BIM (Building Information Model) content to be accessible in a user friendly, web stream-able standard. In short, I3S enables the streaming of massive geospatial content to web browsers, mobile devices, and desktop applications.

I3S has been evolving since it was publicly shared to the open-source community under an Apache license in early 2015. Adopted as the first 3D streaming Community Standard by the Open Geospatial Consortium (OGC) in the fall of 2017, I3S has continued to evolve, as a living, breathing specification. The OGC routinely picks updates from the GitHub version of I3S, as evidenced by I3S OGC 1.1 Community Standard update, incorporating Point Cloud Scene Layer support in early 2020, as well as the recent performance and scalability focused update for I3S 1.2 Community Standard, on which we will focus on this paper.

A note to the reader, the OGC community standard, by design, lags the Github version of I3S. The current OGC I3S Version 1.2 is compatible with 1.8 GitHub version of I3S. There is a compatibility map, https://github.com/Esri/i3s-spec#an-ogc-community-standard, located at the Github version of I3S that is regularly updated.

1.2 I3S OGC 1.2 Community Standard

The essence of the changes in OGC I3S 1.2 Community Standard include:

• Node Paging Support: Introduction of a node paging (bundling) capability significantly reduces the client-server traffic resulting in significant performance improvement.

• Draco Compression Support: Compression of I3S geometry attributes using Draco 3D Data Compression scheme creates a more compact content, which in turn provides a smaller payload, increasing performance.

• Advanced Material Support: I3S OGC 1.2 now has advanced material definitions supporting PBR materials. I3S advanced material support is feature compatible to Khronos® glTF™ standard.

• Basis Universal Texture Compression in Khronos® KTX™ 2.0 container Support: I3S 1.2 supports supercompression of texture data using the Basis Universal Texture interchange system in the Khronos® KTX™ 2.0 format.

2. I3S EVOLUTION

The I3S standard has been evolving over the last decade and has been adding features and improvements, in backward compatible fashion. As described in the introduction section above, I3S OGC 1.2 also adheres to this fact, and we will explore in further detail the additions at this version.

2.1 Paged Nodes

In addition to geometry, texture, feature and attribute data, all represented in binary forms, I3S also includes the bounding volume hierarchy (BVH) and selection criteria in a textual format represented with a JSON encoding. Pre OGC I3S 1.2, this information existed in an expanded notation format, where each node (referenced as a tile in other data partitioning schemes) was represented as an individual asset and required being accessed individually, making the server-client request pattern very chatty. OGC I3S 1.2 introduces node paging capability, significantly reducing the server-client traffic by bundling a group of nodes into a page (the nodes in the bundle have spatial proximity). By default, there are 64 nodes in each node page, thereby significantly reducing client requests and allowing local caching of the resource for further performance improvement.

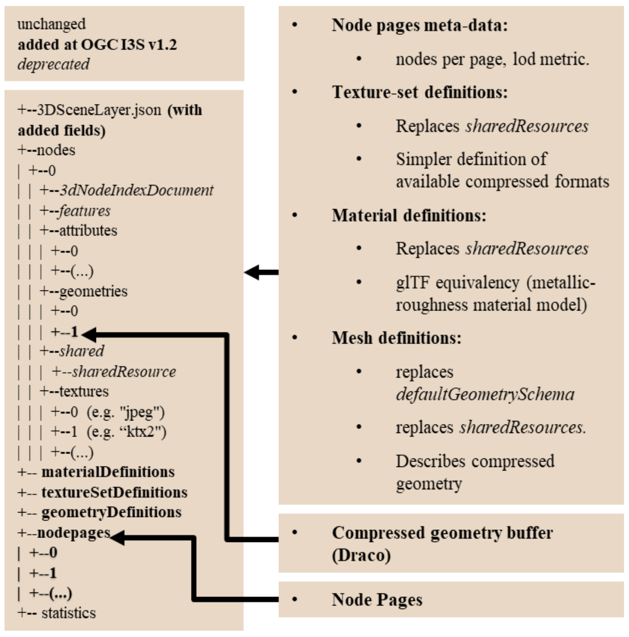
unchanged
added at OGC I3S v1.2
deprecated
+--3DSceneLayer.json **(with added fields)**
+--nodes
| +--0
| | +--*3dNodeIndexDocument*
| | +--*features*
| | +--attributes
| | | +--0
| | | +--(...)
| | +--geometries
| | | +--0
| | | +--**1**
| | +--*shared*
| | | +--*sharedResource*
| | +--textures
| | | +--0 (e.g. "jpeg")
| | | +--1 (e.g. "ktx2")
| | | +--(...)
+-- **materialDefinitions**
+-- **textureSetDefinitions**
+-- **geometryDefinitions**
+--**nodepages**
| **+--0**
| **+--1**
| **+--(...)**
+-- statistics
Node pages meta-data:
nodes per page, lod metric.
Texture-set definitions:
Replaces *sharedResources*
Simpler definition of available compressed formats
Material definitions:
Replaces *sharedResources*
glTF equivalency (metallic-roughness material model)
Mesh definitions:
replaces *defaultGeometrySchema*
replaces *sharedResources.*
Describes compressed geometry
Compressed geometry buffer (Draco)
Node Pages

Figure 1. Figure showing evolution of OGC I3S V1.2.

2.2 Support For Geometry Compression Using Draco

Previous versions of I3S supported compression of the components of an I3S geometry attribute using both position quantization and a non-lossy compression scheme, Gzip.

The primary components of an I3S geometry attribute include vertex positions (x,y,z) – expressed as offsets from the centre of bounding volume (BV), normals (dx, dy, dz), texture coordinates (uv0), color (RGBA) and an optional sub-image UV region, defined as a per vertex, four-component array [u_min, v_min, u_max, v_max].

Note that vertex positions are recorded not as absolute values but rather, as offsets from the BV centre of the node – captured as a Float64 value type, there by retaining the high precision required in geospatial applications, while still benefiting from the quantization gained by storing Float32 vertex offset values.

Geometry Component	*Value Type*	*description*
vertexCount	UInt32	Number of vertices
featureCount	UInt32	Number of features (could be 0)
position	Float32	Vertex positions as x,y,z [3*vertex count]
normal	Float32	Normal vectors as dx, dy, dz [3*vertex count]
uv0	Float32	Texture coordinates [2*vertex count]
color	UInt8	Geometry color as R,G,B,A [4*vertex count]
region	UINT16	sub-image UV region for repeated textures [4*vertex count].
id	UInt64	Feature IDs [feature count]
faceRange	UInt32	Inclusive range of the mesh triangles belonging to each feature in the featureID array. [2*feature count]

Table 1. Components of an I3S Geometry buffer.

Though the I3S geometry buffer had a compact layout, the advent of superior compression schemes such as Draco 3D Data Compression scheme, specifically designed to improve storage and transmission of 3D graphics, allow creating a more compact I3S geometry buffer (which in turn provides a smaller payload increasing performance). As a result, OGC I3S 1.2 opted to standardize on compressing I3S geometry buffer using Draco, https://google.github.io/draco/spec, in lieu of Gzip. Draco, a library for compressing and decompressing 3D geometric meshes and point clouds supports compressing points, connectivity information, texture coordinates, color information, normals, and any other generic attributes associated with geometry, allowing for significantly smaller payloads without compromising visual fidelity [Galligan, 2017].

By using Draco compression, OGC I3S 1.2 Community Standard geometry payload is ~85% smaller compared to OGC I3S 1.1 which uses Gzip for compression. Such a reduction in the geometry payloads enables I3S 1.2 to stream content across the wire more efficiently compared to previous versions.

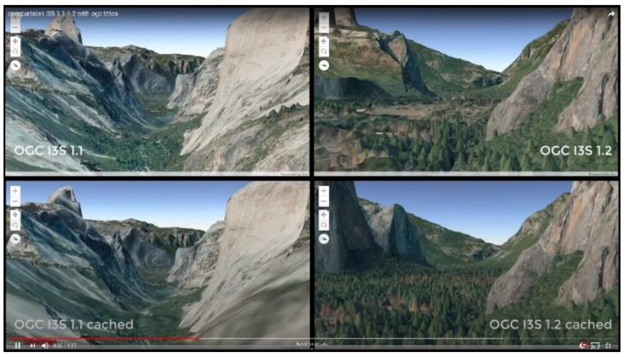

Figure 2. Snapshot showing loading performance of I3S OGC 1.1 community standard compared to OGC I3S 1.2. As shown in the video, https://youtu.be/chRpgjIDkzc, the scenes on the left column correspond to I3S OGC 1.1 and take approximately 75 and 60 secs to completely render, whereas the scenes on the right column corresponding to OGC I3S 1.2 and take 30 and 25 seconds, with and without any client-side browser caching, respectively in both cases. Data provided by Vricon.

Test case	***Loading Improvement factor***			***Layer Type***	***Layer Size[1]***	***Textured***
	Min	**Avg**	Max			
Spain	2.0	**3.0**	3.6	Integrated Mesh	Large	Yes
Washington	1.8	**3.4**	5.6	Integrated Mesh	Medium	Yes
San Francisco	1.7	**2.5**	3.4	3D Object	Large	Yes
San Diego	2.0	**2.5**	3.2	3D Object	Medium	Yes
New York	1.9	**2.6**	3.7	3D Object	Large	No
Zurich	1.5	**1.9**	2.3	3D Object	Medium	No
Vancouver	1.7	**2.5**	4.3	3D Object	Medium	No
Manhattan	1.3	**2.7**	5.9	3D Object	Small	No
France	2.5	**3.2**	4.6	3D Object	Huge	No

Table 2. Loading Improvement factors of OGC I3S 1.2 over I3S OGC 1.1 Community Standard. These improvements were measured using the ArcGIS JS API client application.

[1] Layer Size: Indicates the total I3S dataset size where Small is < 5GB, Medium 5 – 20GB, Large 20 – 50 GB and Huge > 50G

2.3 Support for Advanced Material Definitions

Physically Based Rendering (PBR) materials have significantly changed the fidelity and realism of geospatial asset representations.

I3S's Advanced Material Definition support adds realism to geospatial assets and is feature compatible with Khronos® glTF™ standard with the following exceptions:

• I3S material color properties (baseColorFactor, emissiveFactor etc.) are assumed to be in the same color space as the textures, most commonly sRGB while in glTF they are interpreted as linear.
• glTF has separate definitions for properties like strength for occlusionTextureInfo and scale for normalTextureInfo. Further I3S has only one texture definition with factor that replaces strength and scale.

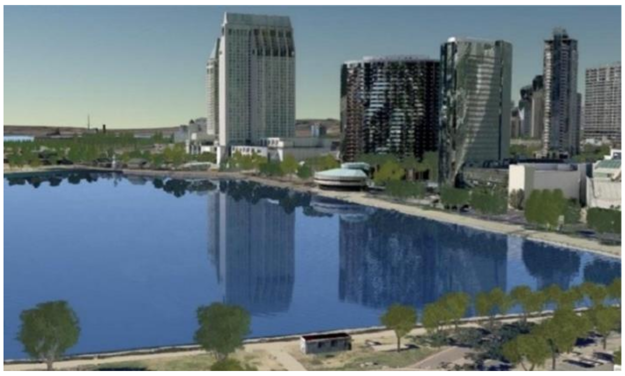

Figure 3. A snapshot of various I3S layer types showing advanced material support added at OGC Version 1.2.

2.4 Support For Basis Universal Supercompressed Texture in KTX™2.0

One of the key features of IntegratedMesh and 3D Object Scene Layers in I3S include support for compressed texture formats such as DXT and ETC2, as GPU native texture asset resources. Compressed textures bring massive reduction in client application memory since they are directly loaded in the GPU without having to be uncompressed to RGB/A. This affords the consuming application much-less memory utilization and avoid the CPU cycles required to decompress highly compressed image formats such as JPEG and PNG (Belayneh, 2021).

The advantage of a compressed texture is clear. Compressed textures considerably lower memory footprint of an application (particularly important in GPUs with shared graphics memory), more compressed images can fit into the cache of the processor and using compressed texture can lower battery usage on mobile devices by avoiding expensive decompression from highly compact formats such as JPEG/PNG. However, generating and using compressed textures comes at a cost, including, compressed textures are significantly larger compared to the same image quality in JPEG/PNG formats and they tend to be hardware and platform specific (different platforms require different type of compressed textures) – creating challenges in storage and transmission. Lastly, creating compressed textures is also prohibitively slow even when using state of the art compression software. For example, ETC2 (a compressed texture format native to GPUs on mobile platforms) and PVRTC (GPU native compressed texture supported on Apple Ax/M1 chipsets) are on average about 100x magnitude order slower compared to the generation of DXT (using Intel SSE compressor) which is a GPU native compressed texture format usable on desktop platforms (Belayneh, T., Khronos 3D Formats Working Group, 2022).

	AMD	Apple Ax	Apple M1	ARM	Intel	NVIDIA Desktop	NVIDIA Tegra	Qualcomm
ASTC	✗	A8 and newer	✓	Mali-T620 and newer	Gen9 and newer	✗	✓	Adreno 3xx series and newer
BC1, BC3	✓	✗	✓	✗	✓	✓	✓	✗
BC7	Radeon HD 5000 series and newer	✗	✓	✗	Gen7 and newer	GeForce 400 and newer	✓	✗
ETC1	✗	A7 and newer	✓	Mali-300 and newer	Gen8 and newer	✗	✓	Adreno 2xx series and newer
ETC2	✗	A7 and newer	✓	Mali-T6xx and newer	Gen8 and newer	✗	✓	Adreno 3xx series and newer
PVRTC1	✗	✓	✓	✗	✗	✗	✗	✗

Table 3. GPU Compressed Texture Format Fragmentation. Multiple GPU Compressed Texture formats with varying levels of support across diverse platforms. (Khronos Group, 2021).

Until the introduction of GSTs – GPU-decodable Supercompressed Textures (Krajcevski et al., 2016), GPU native compressed image generation was extremely specialized and was tightly coupled to specific hardware/platforms (see Table 3). Before GSTs, there wasn't much option other than redundantly generating the same texture and distributing it in the various compressed texture format flavors, targeting specific platforms.

GSTs allowed creating a supercompressed texture format once and being able to transcode it on the GPU to the native compressed texture format supported on the target platform. The general idea behind GSTs is to further compress endpoint texture compression formats such as DXT, ETC2 and PVRTC – significantly reducing the payload size (Krajcevski et al., 2016) (which are typically 3X larger compared to JPEG/PNG – even after further lossless LZ77 re-compression). The reduction in compressed texture size coupled with just having to deal with a single texture asset that works everywhere, allows asset creators the ability to create and transmit GPU native textures to any desired platform. GSTs came to further widespread usage with the introduction of Basis Universal Texture format by Binomial (Geldreich, R., 2022).

Performant 3D systems routinely utilize compressed textures, as an essential optimization technique to support the loading and display of massive amounts of textures in 3D applications.

The addition of Basis Universal compressed texture in KTX™2.0 containers in I3S OGC 1.2 brings excellent decoding speed, great compression rates while keeping the visual quality and fidelity intact. Basis Universal, defines a 'universal' compressed texture format that can be efficiently transcoded at run-time into a natively supported GPU format on the target device, allowing the compressed texture files to operate cross-platform. The resulting texture files are comparable to JPEG/PNG in quality and transmission size but consume only a fraction of the GPU memory

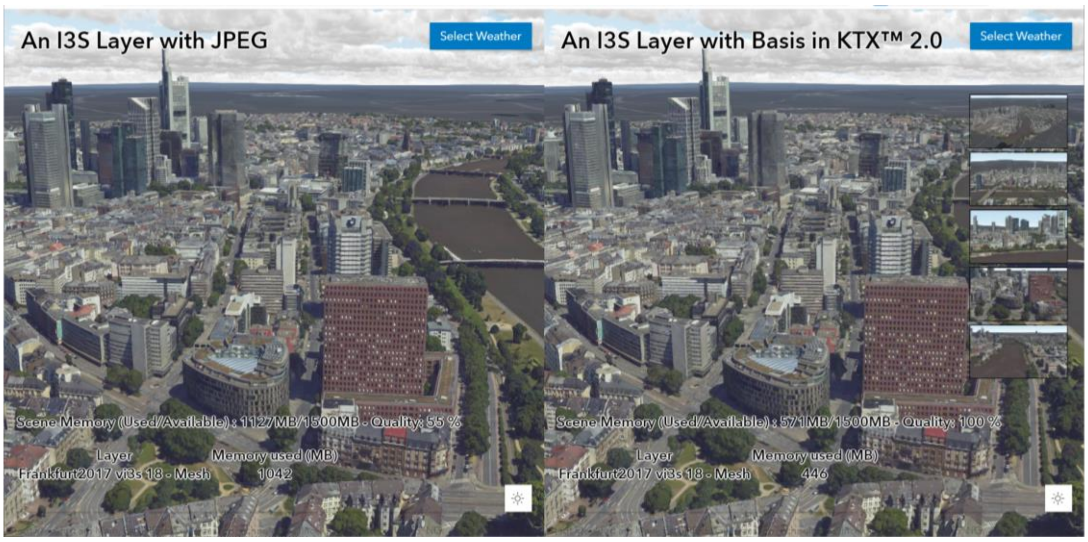
An I3S Layer with JPEG
Select Weather
Scene Memory (Used/Available): 1127MB/1500MB - Quality: 55 %
Layer
Memory used (MB)
Frankfurt2017 vi3s 18 - Mesh
1042
An I3S Layer with Basis in KTX™ 2.0
Select Weather
Scene Memory (Used/Available): 571MB/1500MB - Quality: 100 %
Layer
Memory used (MB)
Frankfurt2017 vi3s 18 - Mesh
446

Figure 4. Figure showing evolution of OGC I3S V1.2. An ArcGIS JSAPI based side-by side compare application, https://tamrat-b.github.io/i3sBasisKTX20, demonstrating OGC I3S 1.2 client-side memory reduction improvements. The app has various *bookmarks* where on average I3S 1.2 shows over 60% client-side memory savings when the input I3S layer is JPEG vs Basis Universal SuperCompressed texture in KTX™2.0 container.

3. BASIS UNIVERSAL SUPERCOMPRESSED TEXTURE ENCODER OPTIMIZATION

In the 3D graphics industry, when it comes to compressed texture assets, there has been much focus in maintaining quality and size as optimal as possible. However, less emphasis and effort has gone into improving the encoding rates of assets. In other words, the encoding codec was relegated to a back-office task that must be done once to get the many runtime advantages it brings to the fore - which was also the case for the Basis Universal Texture codec as well.

3.1 Phase 1 Optimization

Geospatial users, typically generate content and have a need to share, modify and republish content frequently, typically modelling real world scenarios that change based on various factors. As a result, it is critical for geospatial users to be able to generate supper compressed textures in reasonable amount of time. In early 2020, Esri collaborated with Binomial, to improve the Basis Universal encoder speed by a factor of at least 3X (Belayneh, 2021).

In fact, as it can be seen from Table 1, there is about ~4X improvement in performance for larger textures (2k-4k) brought by Basis 1.13 encoder for the same quality (measured as a function of Y-PSNR) and file sizes across various I3S dataset types. This was very encouraging result especially as the optimization was brought about with very minimal loss in quality (less than 3db) or significant increment in file size (less than 5%).

Image	*Opt* [1]	*Opt*[2]	*Pre-Opt*[3]	*Factor* [4]	*Factor* [5]
FrankFurt_4k	5.12	4.62	14.23	2.78	3.08
Mesh_1k	0.3	0.26	0.91	3.01	3.57
NYC_2k	1.48	1.33	5.22	3.51	3.91
SanFran_4k	8.68	6.45	23.46	2.7	3.64
chateau_1k	0.72	0.63	1.96	2.72	3.13
Rancho_4k	6.46	5.82	23.23	3.6	3.99
Singapore_2k	1.08	0.89	3.36	3.11	3.78

Table. 4 Compares the encoding optimization gained for various I3S datasets by using Compression Level 1 and the new geospatial focused compression level 0 settings (using single multithreading). Less values in the *Opt* columns means the task completed faster.

[1] Post-Optimization (Using SSE 4.1, comp_level 1) encoding time (Secs)
[2] Post-Optimization (Using SSE 4.1, comp_level 0) encoding time (Secs)
[3] Pre-Optimization encoding time (Secs)
[4] Improvement Factor (Using Comp_level_1)
[5] Improvement Factor (Using Comp_level_0)

However, even with all the improvements that went into improving Basis Universal encoder, the wall-clock times for encoding Basis Universal compressed texture still lagged, by order of magnitudes, when compared to the encoding times of other compressed texture formats such as DXT1-5/BC3-5 – though it needs to be pointed out, most of the other formats are device and platform specific and result typically in 6X or more bigger payload than Basis Universal compressed texture files.

As encouraging and transformative those results were, we knew we were not done and work on further improving it began soon after, especially to leverage GPUs to dramatically improve the encoding times.

3.2 Phase 2 Optimization

The second collaboration with Binomial, added performance improvement to CPU based encoding and introduced the ability to use GPU to generate Basis Universal SupperCompressed texture and was released in late 2021.

Compressed texture asset creation in geospatial applications tends to be a batch process, where many of the texture assets to be converted will be generated a-prior, during the asset creation phase and are typically queued up for conversion later during the optimization phase. For example, 3D meshes produced by the likes of SURE™ for ArcGIS, a surface reconstruction software that empowers you to create photo-realistic models using datasets captured via both large-frame nadir and oblique cameras as well as hybrid systems with lidar sensors, are good examples of such a use case. In geospatial applications its common to generate very large datasets covering a cityscape or a site resulting in an I3S SLPK (Scene Layer Package) layer with tens of thousands of textures, suitable for batch submission to the GPU to be converted in parallel, taking advantage of powerful GPU devices that might otherwise be idle.

There are plenty of applications that bring about dramatic performance improvements to traditionally CPU bound intensive tasks by leveraging GPUs. With ever improving GPU texture bandwidth and faster ROP operations, using the GPU to encode textures is a well-researched and established arena.

This is also true in geospatial applications which tend to have further requirements: on one hand geospatial users, like online video games etc… would benefit from readily available, highly optimized GPU friendly compressed texture resources, especially when consuming large city/nationwide 3d mesh data.
But their needs do not end just at the consumption level, as they also generate, alter, modify, and publish 3D content using various tool sets available at their disposal. As a result, access to a performant texture encoder library is key in being able to create multiple iterations/scenarios, typically done multiple times in modelling scenarios (Belayneh, T., Khronos 3D Formats Working Group, 2022).

Another key requirement in this workflow is that any optimization brought about by leveraging GPUs needs to universally work as most geospatial applications tend to be deployed in various platforms with varying capabilities and hardware characteristics. As a result, we opted to standardize GPU encoding of Basis Universal not to be specific to any brand of GPU or API, but to be based on Khrono's OpenCL API which is supported by all the major GPUs including from vendors such as NVDIA™, AMD™ & Intel™, covering both discrete and integrated GPUs.

Furthermore, we standardized on OpenCL 1.2 version API as it is supported by many types and generations of GPUs. OpenCL has a well specified computation environment capable of coordinating parallel computation across processors in a cross-platform programming language, very fitting to the central premise of Basis Universal compressed texture format as the cross-platform GST format that works across a variety of devices and operating systems.

Performance Improvement Factor of Basis Universal Encoder at Version 1.16

GPU	Texture assets (count)	Factor over **v1.15** using **parallel GPU**	Encoding time (Secs.)	Factor over **v1.15** using **sequential GPU**	Encoding time (Secs.)	Factor over **v1.15** using **parallel CPU**	Encoding time (Secs.)	**v1.15** encoding time using **parallel CPU** (Secs.)	CPU
Quadro RTX 5000[1]	1,126	**9.9X**	153	**2.0X**	726	**1.8X**	852	1,515	Intel(R) Xeon(R) W-10885M[3]
RTX A6000[2]	10,121	**4.5X**	3,496	**2.2X**	7,238	**2.2X**	7,010	15,648	AMD Ryzen Thread ripper PRO 3995W[4]

Table. 4 Compares the encoding optimization gained for the texture resources of an I3S dataset by leveraging the GPU using OpenCL™ to orchestrate parallel computation as well as CPU improvements gained in version 1.16 vs. 1.15 of the Basis Universal encoder libraries.

[1] Quadro RTX 5000 @ 1.545 GHz, 1545 Mhz, 3072 Cuda Cores, Memory data rate: 14.00 Gbps, Memory interface: 256-bit, Memory bandwidth: 448.06 GB/s, Total available graphics memory: 81779 MB, Dedicated video memory: 16384 MB GDDR6, Driver version: 472.42

[2] Quadro RTX A6000 @ 1.800 GHz, 1800 Mhz, 10752 Cuda Cores, Memory data rate: 16.00 Gbps, Memory interface: 384-bit, Memory bandwidth: 768.106 GB/s, Total available graphics memory: 81826 MB, Dedicated video memory: 49140 MB GDDR6, Driver version: 462.31

[3] Intel(R) Xeon(R) W-10885M CPU @ 2.40GHz, 2400 Mhz, 8 Core(s), 16 Logical Processor(s), Total Physical Memory: 128 GB, running Windows 10 Pro for Workstations 64-bit

[4] AMD Ryzen Threadripper PRO 3995WX 64-Cores, 2695 Mhz, 64 Core(s), 128 Logical Processor(s), Total Physical Memory: 128 GB, running Windows 10 Pro for Workstations 64-bit.

The encoding rate improvement factor of Basis Universal encoder 1.16 could be as much as 10x when using GPU in parallel compression mode to compress over a thousand texture resources with varying texture sizes (ranging from 512 – 4k pixels). As expected, the performance gain decreases when queuing up more textures (in this case we were compressing ~10x (10,121) more textures in parallel), but nevertheless, parallel GPU compression in this use case still yields a 4.5X faster rate than version 1.15 using CPU encoding (running in multi-thread mode with CPU compression only (16 threads used per hardware concurrency).

In this optimization phase, not only GPU based encoding was introduced but CPU based encoding was also improved for an average of 2X. These improvement numbers (up to ~10x factor improvement as shown in Table 2) are on top of the existing 3X improvement introduced at Basis Universal encoder version 1.13/1.14 (Basis Universal 1.15 added ability to encode Basis Universal in KTX™ 2.0 containers), purely focusing on CPU optimizations.

4. CONCLUSION AND FUTURE WORK

It is evident Geospatial standards such as I3S OGC 1.2 have an evolving need to adopt, implement and improve general 3D graphics optimizations and usage patterns.

I3S OGC 1.2 introduced 4 areas of improvements, namely, node paging, better geometry compression using Draco, advanced material support compatible with Khronos® glTF™ standard, and reduced client-side memory usage using Basis Universal Supercompressed Texture in Khronos® KTX™ 2.0 format, all working in concert to bring increased client application performance and scalability as shown in Figures 2 and 4 and Table 2, respectively.

We also described in detail and demonstrated collaborative research that culminated in the improvement of Basis Universal Texture encoder (Basis Universal library is freely available, https://github.com/BinomialLLC/basis_universal, delivered in a well-established container format Khronos KTX™ 2.0), further paving the way for it to become the de facto format for delivering compressed texture assets for 3D geospatial content.

The release of Basis Universal 1.16 culminated in bringing up to 10x increase in the encoder codec performance over Basis Universal 1.15 version when using GPU encoding as shown on Table 4. With this release and via the introduction of GPUs to complement the texture conversion process, the Basis Universal texture format is now creatable at a rate similar to DXT1-5/BC1,3,7 encoding times, while still keeping its advantage of being cross platform and as compact as input source (whereas the latter is 3x bigger than input source typically and is platform specific. See Table 3).

These improvements are made accessible to geospatial users as they are incorporated in the freely available I3S Converter tool, https://github.com/Esri/i3s-spec. The current version of the tool has incorporated Phase 1 optimizations of the encoder, enabling the upgrade of I3S datasets to the latest OGC I3S 1.2 community standard version supporting Basis Universal compressed textures in KTX™ 2.0 containers. Work is ongoing to incorporate results from Phase 2 optimization bringing GPU capabilities as well as much faster encode times.

ACKNOWLEDGEMENTS

The author and Esri would like to thank Stephanie Hurlburt & Richard Geldreich of Binomial for the opportunity to collaborate on an exciting area of optimization of Basis Universal encoder. The optimized encoder code of Basis Universal (Version 1.16) is now publicly available under Apache 2.0 license.

REFERENCES

Belayneh, T., 2021. Esri collaborates with Binomial to improve Basis Universal Supercompressed GPU Texture Codec speed. https://www.esri.com/arcgis-blog/products/arcgis/3d-gis/esri-collaborates-with-binomial-to-improve-basis-universal-texture-compression-speeds.

Belayneh, T., Khronos 3D Formats Working Group, 2022. https://www.khronos.org/blog/ktx2.0-support-in-i3s-v1.2-puts-the-whole-world-in-your-hands.

Galligan, F., 2017. Draco Bitstream Specification, Version 2.2. https://google.github.io/draco/spec.

Geldreich, R., 2022. Basis Universal Supercompressed GPU Texture Codec, Version 1.16. https://github.com/BinomialLLC/basis_universal.

Khronos Group, 2021. Universal GPU Compressed Textures for glTF using KTX 2.0, April., 2021. https://www.khronos.org/assets/uploads/apis/KTX-2.0-Launch-Overview-Apr21_.pdf.

Krajcevski., P., Pratapa, S., Manocha, D., 2016. ACM Transactions on Graphics, Volume 35 Issue 6, November 2016, Article No.: 230pp 1–10, http://gamma.cs.unc.edu/GST/gst.pdf.