Indexed 3D Scene Layer (I3S) – An Open 3D Standard

Chris Andrews | 3D Product Manager
Tamrat Belayneh | Lead Software Developer - 3D Services
Sean William Morrish | Product Engineer
Tamrat Belayneh
- Lead Software Developer for 3D services at Esri
- Over 18 years in 3D software Development
- Joined Esri in 2000
- Primary focus areas
  - 3D Services
  - Indexed 3D Scene Layer Specification (I3S)
  - 3D across the ArcGIS platform
Speaker

- **Sean Morrish**
  - 3D Product Engineer
  - 15 years AEC experience
  - 4 years with Esri as product engineer in 3D
  - Primary Focus areas
    - 3D Scene Layer Services
    - Data prep and processing (3D Objects, Points, Mesh, Point Cloud)
    - Testing and implementation of i3S Scene Layers across the platform
• **Chris Andrews**
  - Lead Product Manager for 3D at Esri
  - 9 years in 3D with GIS, AEC, Molecular Biology, Entertainment
  - Joined Esri in 2014
  - Primary focus areas
    - 3D across the ArcGIS platform
    - ArcGIS Earth
    - Evangelizing 3D with customers, partners, and internally
3D content is large, heterogeneous, and distributed.
Core 3D Capability

- Transform 2D and 3D GIS into a single GIS workflow
- Reuse dynamic services across clients
- Securely collect, manage, curate 3D data
- Conduct analysis across real-time and historical data
- Create tailored experiences for different types of users

Anywhere In Any Environment

Cross-platform  Open  Accessible

Server  Online Content and Services

Desktop  Web  Device
Standards - Enabling customers & partners through sharing and integration

Open Software, Standards and Data enable organizational resiliency
- Ensure access to data
- Guarantee interoperability
- Enable innovation
- Encourage usage and adoption

I3S - Scalable 3D scene content for visualization and distribution

LERC - Raster (imagery and elevation) compression technology for 2D and 3D

GeoREST - Esri open REST APIs for access to any kind of GIS content and services
Indexed 3D Scene (I3S) layer specification
- Open specification for 3D layers
- Shared under Creative Commons licensing
- OGC community currently considering I3S as a Community Standard
- Describes a scalable scene cache with attributes and indexing
- Multiple levels of detail
- Can be streamed over the internet
- Can be used locally on disk as a package
- Opportunity for future layer types to accommodate new data types
- Open for feedback and modification
Web Scenes, Scene Layers

• Web Scene
  - Vehicle for cross-platform 3D capability
  - Collection of layers, environment settings, slides, *animation*
  - Essential for 3D apps on any platform or experience

• Scene Layer
  - Scalable cache of graphics, styles, and attributes
  - 3D Objects, 3D Points, Integrated Meshes, Point clouds
  - Future: Enhanced support for: BIM (2017), 3D Lines, 3D Polygons

• Reuse, Share, Extend
3rd party adoption of I3S

• Integrated mesh
  - VRICON, Pix4D, and Bentley sharing packages or services in I3S format
  - Supports Drone2Map

• Additional partners coming soon

• Expanding list of layer types and layer capability
I3S has no commercial dependencies

• Any organization can:
  - Access and read the openly published spec
  - Inspect I3S layers with common tools such as JSON browsers, zip file readers, and text editors
  - Create an I3S layer according to the specification
  - Read an I3S layer created by themselves or another application

• I3S will be improved by having many interested organizations creating and consuming content
Users can
- Create I3S content in Esri and non-Esri apps
- Acquire I3S content from vendors
- Share and distribute I3S content freely

Partners can
- Build their own apps to write I3S content
- Build their own apps to read, analyze, and display I3S content
- Sell apps that can read/write I3S
- Contribute requirements and changes into the I3S community
Indexed 3D Scene Layers are supported across the ArcGIS platform.
Indexed 3D Scene Layers (I3S) – What is it?

- Open standard for storage and transmission of large, heterogeneous 3D geospatial data sets
- Cloud, Web and Mobile friendly based on JSON, REST and modern web standards
- Support 3D geospatial content, various coordinate systems along with a rich set of layer types
- An I3S data set, referred to as a Scene Layer is:
  - a container for arbitrarily large amounts of heterogeneously distributed 3D geographic data
Indexed 3D Scene Layers (I3S) – What is it?

- I3S is in process to become an OGC community standard
- The standard includes specification for Scene Layer Package (SLPK) – An archive that captures all node resources of a scene layer and allows direct access
- I3S can serve as a common tool to package and disseminate, a variety of GIS content
- Both I3S and SLPK are licensed under Creative Commons
- Available @ https://github.com/Esri/i3s-spec
Indexed 3D Scene Layers (I3S) – What is it?

I3S Design Principals for a 3D GIS visualization format

1. **Web friendly:** JSON + Typed Arrays
2. **Mobile friendly:** Works good with varying bandwidth
3. **Extensible:** Support different types of content
4. **Declarative:** Reduce required implicit knowledge
5. **Efficient:** Use spatial indexing for quick delivery
6. **Scalable:** Provide Level of Detail Support
7. **Protected:** Ensure that content is protected
8. **Open:** Full Specification publicly accessible

[https://github.com/Esri/i3s-spec](https://github.com/Esri/i3s-spec)
Scene Layer types and profiles
Support different geometry types

- 3D Objects
- Integrated Meshes
- Points
- Point Clouds
Indexed 3D Scene Layers (I3S)

- 3D Objects
  - Example: Building Exteriors
  - Sources: Derived from GIS Data, as well as 3D models in various formats
Indexed 3D Scene Layers (I3S)

- Integrated Meshes
  - Examples: Mesh surface representing the skin of the Earth, including vegetation, buildings and roads
  - Sources: Derived from satellite, aerial or drone imagery via dense matching photogrammetry, or calculated
Indexed 3D Scene Layers (I3S)

- Points
  - Examples: Hospitals, schools, trees, cars
  - Sources: Feature locations combined with Instanced 3D models generated by hand
Indexed 3D Scene Layers (I3S)

- **Point Clouds**
  - Example: LiDAR data sets
  - Sources: Typically sensor-collected or Photogrammetrically derived
I3S: Organization and structure

- Organizes geospatial data using a hierarchical, node-based spatial index structure
I3S: Organization and structure

The physical organization of information within node:
I3S: LoD Models, Selection Metrics

• I3S promotes the concept of discrete levels of details with multiple discrete representations of features and nodes.

• An example lod selection metric is the maximum screen size that the node may occupy before it must be replaced with data from more detailed nodes.

• This model of discrete LOD rendering is referred to in I3S as node switching.
I3S: Consumption

- As a service (via a REST API) or locally as a file system (SLPK)

  - As RESTful interfaces/services:
    - Via a RESTful interface that exposes the scene layer, its nodes and their associated resources (geometries, attributes, textures) as web addressable resources.
    - I3S resources are designed for direct access (via a unique key) from key value based cloud blob stores such as Windows Azure Blob Storage or Amazon Simple Storage (S3) using built in REST APIs of such infrastructures

  - As a single large Scene Layer Package (SLPK):
    - A single file that packages the complete node tree and its resources into an archive that supports direct access to the individual nodes and resources within it.
I3S: Flexibility

- Examples of I3S flexibility:

- Minimum Bounding Volume (MBV):
  - Minimum Bounding Sphere (MBS)
  - Oriented Bounding Box (OBB)

- The node structure may be
  - ‘expanded’ - with complete meta-information about node’s position and BVH topology
  - ‘fixed-size’ - in support of ‘paged’ access pattern
I3S: Flexibility (cont’d)

• Nodes may have “embedded” vs “binary” geometry/attribute content format
  - Embedded geometry: as JSON in-lined with additional metadata
  - Binary format: as typed array buffer views

• LOD Selection based on different metricTypes:
  - **maxScreenThreshold** - LOD switching based on screen ‘size’ of the node’s MBV
  - **screenSpaceRelative** - LOD switching based on screen ‘scale’ of the node’s MBV
  - **distancRangeFromDefaultCamera** - LOD switching based on normalized distance of the node’s MBV from the camera
  - **effectiveDensity** - estimation of the point density covered by the node
The Key to Scalability: Indexing

- Adapt Index type to data
  - R-Tree
  - Quadtree, Octtree
  - Standard Tiling Scheme

- Load-Balanced
  - Near-Constant Data Volume per Node

Legend

3 Node ID
5 Representation Object ID
• Node Switching LoD
  - *Full Representation* Pyramid with a Node-based alternate representation
  - Use for:
    - Feature data lacking authored LODs
    - Homogeneous contextual data (Integrated Meshes)
Scene Layer Package (SLPK) and Scene Service REST API

Archive.slpk

<table>
<thead>
<tr>
<th>metadata.json</th>
<th>3dScenelayer.json.gz</th>
<th>Zip64 container</th>
</tr>
</thead>
<tbody>
<tr>
<td>/nodes/</td>
<td></td>
<td>gzip'ed resources</td>
</tr>
<tr>
<td>/nodes/root/</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3dNodeIndexDocument.json.gz</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/nodes/1-4-2-0/</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3dNodeIndexDocument.json</td>
<td></td>
<td></td>
</tr>
<tr>
<td>geometries/*</td>
<td>textures/*</td>
<td></td>
</tr>
<tr>
<td>shared/*</td>
<td>features/*</td>
<td></td>
</tr>
<tr>
<td>attributes/*</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
File System Folder Layout
Direct mapping of the REST API

- All resources reside in the file system as individual files.
- These files are organized in folders in the following schema:

```
/3dSceneLayer.json
/nodes/<node-id>/3dNodeIndexDocument.json
/nodes/<node-id>/features/0.json ... n.json
/nodes/<node-id>/geometries/0.bin ... n.bin
/nodes/<node-id>/shared/SharedResource.json
/nodes/<node-id>/textures/0_0.bin ... n_m.bin
/nodes/<node-id>/attributes/0.bin ... n.bin
```
Declarative: Geometry buffer metadata

```
"defaultGeometrySchema": { // geometry resource layout for nodes that declare the use of defaultGeometrySchema in the node index.
  "header": [ // header fields that precede the vertex data
    {
      "property": "vertexCount", // vertex count
      "type": "UInt32"
    }
  ],
  "topology": "PerAttributeArray", // one of ["PerAttributeArray", "InterleavedArray", "Indexed"]. When "Indexed", the indices must also be declared in the geometry.
  "vertexAttributeOrder": ["position", "normal", "uv0", "region"], // provides the order of the keys in vertexAttributes.
  "vertexAttributes": { // the vertex attributes must appear in the order that they are declared here.
    "position": { // the name of the vertex attribute; here: vertex positions
      "valueType": "Float32", // the element type, either UInt8, UInt16, UInt32, Int16, Int32, Int64 or *Float32*, Float64
      "valuesPerElement": 3 // number of (Float32) values need to make a valid element (here a xyz position)
    },
    "normal": { // the name of the vertex attribute; here: vertex normals
      "valueType": "Float32", // the element type, either UInt8, UInt16, UInt32, Int16, Int32, Int64 or *Float32*, Float64
      "valuesPerElement": 3 // number of (Float32) values need to make a valid element (here a xyz position)
    }
  } // face attributes
  "faceCount": 0, // face count
  "featureCount": 0, // feature count
}
```

**Header**
- vertexCount
- faceCount
- featureCount

**Body**
- vertexAttributes
  - position
  - uv0
  - normal
  - color
- faceAttributes
  - position
  - uv0
  - normal
- featureAttributes
  - id
  - faceRange