Employing OGC's 3D Portrayal Service to Interoperate Hierarchical Data Structures: A Case Study on Visualizing I3S in Cesium

ATHANASIOS KOUKOFIKIS[1] & VOLKER COORS[1]

Abstract: The demand of serving large 3D spatial data, mainly in urban areas, reflected the need of hierarchical 3D data structures. During the last years the OGC community standard I3S (ESRI) and 3D Tiles (Analytical Graphics), emerged in order to deal with this issue. Conceptually, hierarchical 3D structures operate in an analogous manner with web map tiles, differentiating only in the implementation. A prototype implementation focuses on the rendering of I3S in the Cesium client using the 3D Portrayal Service. As a result, the user can query a scene via the 3D Portrayal Service by specifying a spatial region, rather than a specific resource via a URI. The result can be delivered either using I3S or 3D Tiles as a data delivery format, depending on which data is available for the specified region. The Cesium client can both render both the I3S as well as the 3D Tiles content.

1 Introduction

The technological pluralism in 3D web visualization constantly increases the need of interoperability. The availability of a number of different 3D delivery formats and client consumers indicates a need for a generic approach for accessing 3D information. This paper defines an attempt to realize the interoperability between geospatial data web consumers (web globes) and OGC's 3D Portrayal Service regarding the request of hierarchical 3D data storages. 3D Tiles and I3S are the latest implementations of hierarchical data storages. Our goal is to investigate if the 3D Portrayal Service could abstract the access of 3D Tiles or I3S in a dedicated web client/consumer which is designed to be compatible with specific delivery formats (e.g. Cesium, 3D Tiles). Cesium is a web globe API designed to support 3D Tiles, whereas ESRI's ArcGIS API for JavaScript is designed to consume I3S. Our approach attempts to render I3S data in Cesium employing a request scheme that extends 3D portrayal's getScene request.

2 Related Work

Use cases of the 3D Portrayal Service appear to be scarce, since the version 1.0 of the standard was recently released. GAILLARD et al. (2015) utilized the 3D Portrayal Service View conformance class for client side rendering of tiled 3D city models. Building geometry, originating form CityGML, was converted and served as JSON using a GetScene request. GUTBELL et al. (2016) implemented a server side rendering framework to visualize 3D city models using the 3D Portrayal Service GetView request.

[1] Hochschule für Technik Stuttgart, Faculty Geomatics, Computer Science and Mathematics, Schellingstr. 24, D-70174 Stuttgart, E-Mail: [athanasios.koukofikis, volker.coors]@hft-stuttgart.de

Cesium as visualization component is used in several cases. KRÄMER & GUTBELL (2015) produced a surface model visualization using Cesium's Terrain Builder. KYOUNG-SOOK et al. (2017) presented a visual extension of Cesium to visualize moving objects in a space-time cube. LU et al. (2016) implemented an online 3D city model interactive editor using a light-weight viewer based on Cesium.
So far, use cases of Cesium utilize data formats which are compatible with this API.

3 Interoperable visualization using the 3D Portrayal Service (3DPS)

The 3D Portrayal Service is a OGC Standard that abstracts the access of 3D geospatial datasets in various client platforms via the web for visualization purposes (3D PORTRAYAL SERVICE 1.0 2017). The 3D Portrayal Service specifies three methods to access information: GetCapabilities, AbstractGetPortrayal and AbstractGetFeatureInfo. A GetCapabilities request returns information about the available request methods, the extents of the data, data layers (buildings, vegetation etc.), layer styles and streaming formats supported. The retrieval of a scene is implemented by the AbstractGetPortrayal operator. The GetScene method is used for client side rendering and the GetView method for server side rendering. The AbstractGetFeatureInfo implementations are used for requesting metadata of scene features. The supported methods are GetFeatureInfoByObjectId, GetFeatureInfoByPosition and GetFeatureInfoByRay. The 3D Portrayal Service specification does not indicate a content delivery format. The OGC community standard I3S can be used as content delivery for the GetScene request. I3S (Indexed 3d Scene Layer) is a technology for rapidly streaming and distributing large volumes of 3D content across enterprise systems that may consist of server components, cloud hosted components, and desktop, web and mobile clients (I3S-SPEC 2017). The specification is realized by various profiles that describe the behavior of a I3S layer. The spatial extent of the I3S data is split into regions, called nodes, and organized into a hierarchical data structure that allows the client to quickly discover which data is needed and the server to quickly locate the data requested by the client (INDEXED 3D SCENE LAYER FORMAT SPECIFICATION 2017).

4 Methodology

An agile software development approach was used in this study to implement both 3DPS server and web-based client. It follows an iterative incremental process with 4 prototypes in total. Each iteration encapsulates a prototypical objective which contributes to the goal of this case study. In order to prove interoperability, we selected a hierarchical 3D format, i.e., I3S, with a dedicated consumer on the client side and Cesium as an incompatible consumer.
The aim of the first prototype was a preliminary attempt to visualize elements with a geographic component coming from I3S in Cesium. For this, we used the minimum bounding spheres (MBS) of I3S nodes.
The aim of the second prototype was the implementation of the 3D Portrayal Service and visualization of building shells originating from I3S. During this step, it was critical to decide the role of the 3D Portrayal Service in the end to end communication. An important action is the traversal of the I3S node tree, which in the case of an ArcGIS API application is happening on the client.

A Cesium application accessing the I3S REST API for the node traversal, in order to identify the visible nodes in the view frustum, would result to a passive/obsolete 3D Portrayal Service implementation. For this reason, the 3DPS received an extra parameter, i.e., "i3sLayer" and was responsible to access and traverse the node tree of a I3S layer.
The third prototype introduced the "boundingbox" parameter from the GetScene request and replace the "i3sLayer" which was used until this point to identify the requested resource. Even though the access of 3D resources was abstracted after the "i3sLayer" parameter was removed, the "boundingbox" parameter was still a static implementation.
The aim of the fourth prototype was to define a dynamic bounding box selection in the server (3DPS) and client side in order to allow the users to fetch the data they are interested. In this case the "boundingbox" parameter is used to identify which hierarchical dataset is requested and also restrict spatially the tiles/nodes to be rendered.

5 Implementation

The 3D Portrayal Service is implemented utilizing the Play Framework. At a later stage of the application the Dojo Toolkit is introduced in the client side.

5.1 Play Framework

Play is a high-productivity Java and Scala web application framework that integrates the components and APIs needed for modern web application development. Play is based on a lightweight, stateless, web-friendly architecture for highly-scalable applications (PLAY 2.6.X DOCUMENTATION 2017). The source code is compiled into bytecode and executed in the Java Virtual Machine.

5.2 Dojo Toolkit

Dojo Toolkit is an open source JavaScript library designed to ease the rapid development of cross-platform, web applications and web sites (DOJO TOOLKIT 1.13 2017). This tool was used mainly to modularize the client side application.

5.3 First Prototype: Visualization of the minimum bounding spheres of I3S nodes

The first attempt visualizes spatial elements retrieved by a I3S REST API. For this purpose, the minimum bounding sphere (MBS) for each I3S node that lies within a cartographic bounding box is visualized using Cesium's Ellipsoid primitive. The MBS returned by the I3S REST API contains an array with the geographical position of the sphere (longitude, latitude, height) and its radius in meters (Fig. 1).

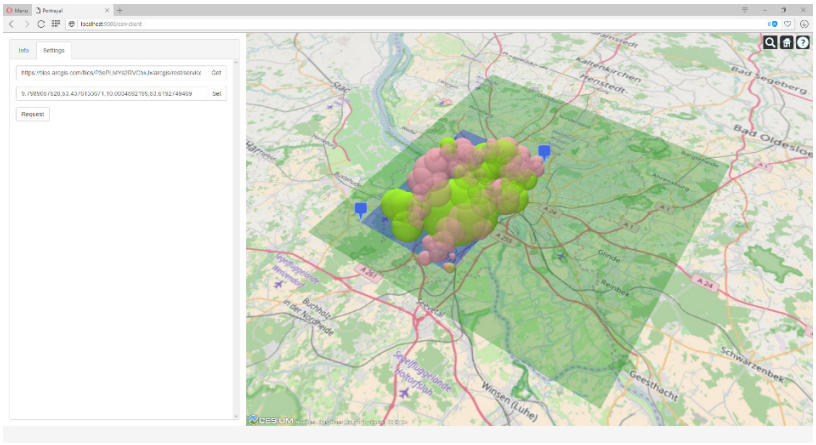

Fig. 1: Visualization of the minimum bounding spheres of I3S nodes within a user defined bounding box. Each sphere is styled by the node level. The green box defines the extents of the I3S layer

5.4 Second Prototype: 3D Portrayal Implementation

In this stage a middleware service (3DPS) is introduced using the Play Framework. This service, which plays the role of the 3D Portrayal Service, acts as a broker that negotiates the I3S nodes that should be rendered in the Cesium viewer (Fig. 2). The "boundingbox" URL parameter of the GetScene request is not present in the server endpoint, yet the I3S layer URL is passed directly as a request parameter to identify the 3D dataset that should be queried.

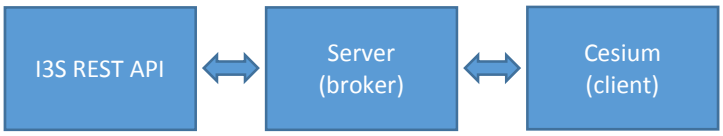

Fig. 2: Rendering I3S in Cesium overview

On the client side, a request is sent to the server when the "camera changed" event is triggered in Cesium. On the server side (broker) the implemented API is responsible to handle any requests from the client. The core action of the broker service is to apply the I3S node selection criteria, then generate a response with the nodes' description that should be rendered in Cesium. Once the broker's response is delivered, the client fetches the I3S payloads using the REST API and prepares the visualization (Fig. 3).

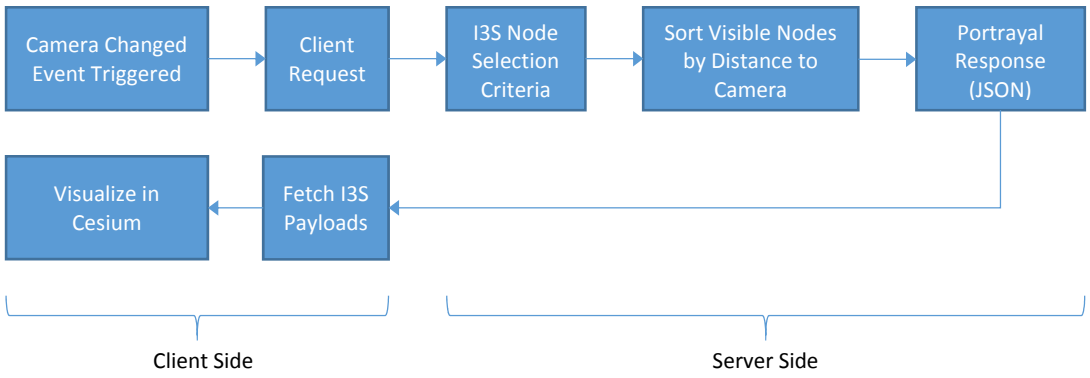

Fig. 3: Communication Synopsis

The I3S node selection is a repetitive process which accesses the root node of the I3S layer and continues by traversing descendant nodes of the tree structure utilizing the breadth-first traversal method. The main node selection criteria are: a) The node's minimum bounding sphere (MBS) visibility b) The node's MBS screen size in pixels and c) The existence of a node's children. The last iteration introduces the intersection of the MBS center with a buffered cartographic bounding box, defined by the user, as an additional criterion. The following pseudo code describes the traversal of an I3S node inside the broker service:

```
if node's mbs in not visible in client's viewport
      break traversal of the node and it's children
else if node's screen size >= node's maxScreenThreshold and node has children
      traverse node's children
else
      add node in the response for rendering
```

When the I3S node traversal is completed, a response is generated which contains the essential information to render a node in Cesium. For rendering optimization, the response nodes are sorted with ascending order by distance to the camera (Fig. 4).


```
[
  + {…},
  - {
        id: "5-1-0-0-0",
        level: 6,
        lng: 9.994545253174223,
        lat: 53.568095227256954,
        radius: 455.4209899902344,
        url: "https://tiles.arcgis.com/tiles/P3ePLMYs2RVChkJx/arcgis/rest/services/Buildings_Hamburg/SceneServer/layers/0/nodes/5-1-0-0-0",
        distanceToCamera: 140.98541019527966,
        time: "1504872740063"
    },
  - {
        id: "5-0-0-2-0",
        level: 6,
        lng: 9.986759068536088,
        lat: 53.57170940098039,
        radius: 563.823974609375,
        url: "https://tiles.arcgis.com/tiles/P3ePLMYs2RVChkJx/arcgis/rest/services/Buildings_Hamburg/SceneServer/layers/0/nodes/5-0-0-2-0",
        distanceToCamera: 162.51702725550808,
        time: "1504872740063"
    },
  - {
        id: "5-1-0-0-1",
        level: 6,
        lng: 9.997446293904332,
        lat: 53.565953434999045,
        radius: 546.78515625,
        url: "https://tiles.arcgis.com/tiles/P3ePLMYs2RVChkJx/arcgis/rest/services/Buildings_Hamburg/SceneServer/layers/0/nodes/5-1-0-0-1",
        distanceToCamera: 214.9592984261286,
        time: "1504872740063"
    },
  + {…},
  + {…},
```


Fig. 4: Broker service response

I3S supports discrete levels of detail (LoD) in the geometry which correspond to the levels in the nodes' hierarchy. Leaf nodes usually contain the original representation with the highest detail, where lower node levels contain simplified geometry of the same features using edge collapse algorithms (Fig. 5).

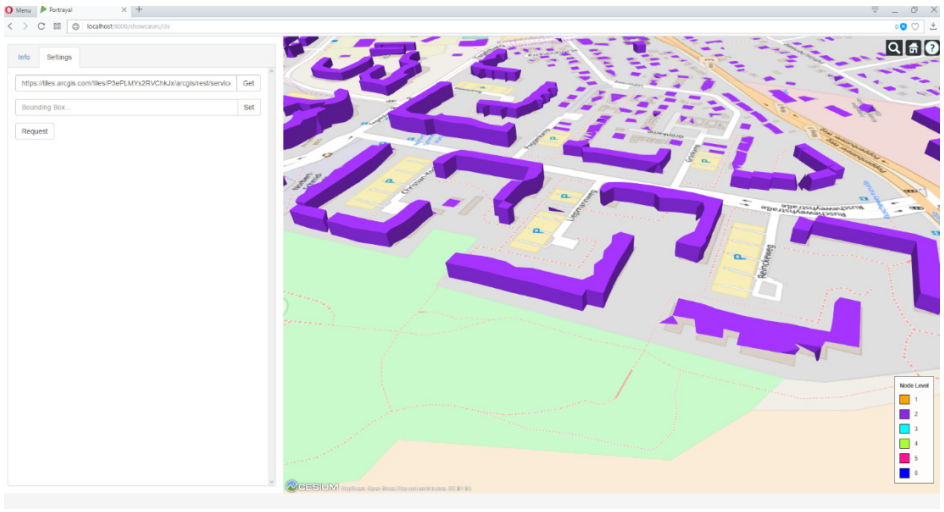

Fig. 5: Decimated geometry of I3S low level nodes rendered in Cesium

5.5 Third Prototype: The “boundingbox” parameter

In this stage of the application the I3S layer URL parameter gets replaced with a static “boundingbox” parameter. Although the user is able to select a bounding box which is available in the 3D Portrayal Service capabilities and corresponds to the extents of various 3D city models, it is not possible to dynamically define its own bounding box (Fig. 6). On the server side a lookup array is used to match the requested bounding box with the I3S layer URL.

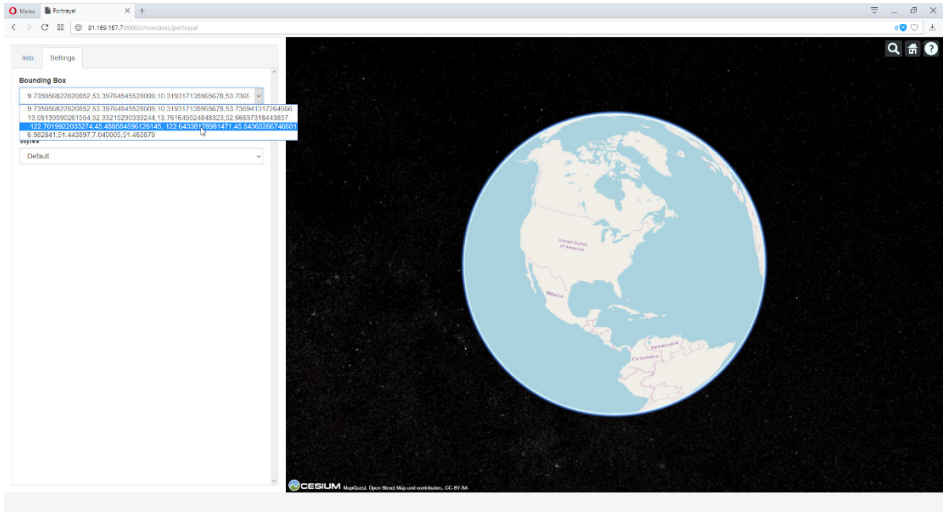

Fig. 6: The user is able to select a bounding box the defines the extents of a 3D city model stored as I3S

5.6 Fourth Prototype: Dynamic bounding box selection

In the code base of the application the Dojo Toolkit is introduced in order to modularize the client application. Even though the client appears the same, many parts of the client were rewritten. The user is able to dynamically define a cartographic bounding box which is spatially intersected by the 3D Portrayal Service available extents in order to identify the requested resource, either

3D Tiles or I3S. Additionally, a cartographic buffer is generated around the requested bounding box to include neighboring I3S nodes (Fig. 7).

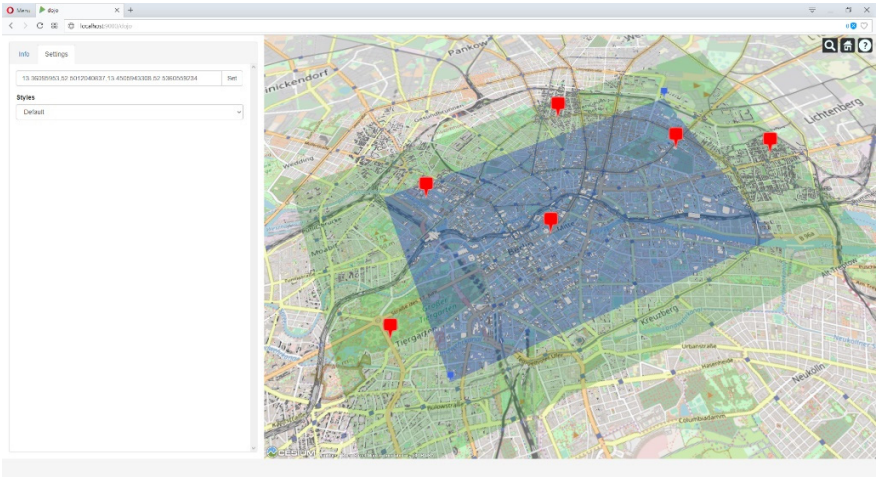

Fig. 7: I3S nodes rendered in Cesium. Blue rectangle: requested bounding box. Green rectangle: buffer. Red pins: centers of the minimum bounding spheres

The 3D Portrayal request is extended with URL parameters which are not defined in the request scheme of the standard. These parameters, originating from Cesium, are used to apply the I3S node selection criteria and to reject outdated responses by the 3D Portrayal Service (Fig. 8).

service: 3DPS
acceptversions: 1.0
request: GetScene
boundingbox: 9.7350568228,53.397645455,10.3193171359,53.7369413172
cullingvolume: -0.489520008514,0.87030950429,-0.054143586849,1533027.01880,-0.161886420120,-0.9853188378,-0.054217845856,1530930.62031,-0.62377256088,-0.110165543187,0.77380316966,-1550108.68011,-0.3480433034,-0.061413811752,-0.9354646987,6121145.3019,-0.97181586470,-0.171579355008,-0.161661529262,4571035.6240,0.97181586470,0.171579355008,0.161661529262,495428963.37
camera: 3737336.29540,658537.61433,5109703.3728,-0.97181586470,-0.171579355008,-0.161661529262
frustum: 1,0.57735026918,0.35572226262
drawingbuffer: 573,930
time: 1516234129317

Fig. 8: Extended 3D Portrayal request with additional parameters highlighted in blue

6 Results & Summary

The 3D Portrayal Service uses a bounding box oriented approach to deliver 3D content to clients while hierarchical 3D datasets are suitable for viewport oriented applications. In this paper, the 3DPS was used as a broker to simplify data access (Fig. 9).

Fig. 9: I3S layer rendered in Cesium via the 3D Portrayal Service. On the left side the default style is used, on the right side the node level is used for styling

It basically translated user bounding box queries to a URI of the relevant 3D data set in I3S. Of course, 3D Tiles data sets can be handled the same way, but the focus was on rendering I3S in Cesium. The cost of using the 3DPS as a broker intermediary is O(log n) if a spatial index is used on server side for searching the relevant data set.

7 Acknowledgements

We would like to thank Tamrat Belayneh for his invaluable experience and help.

8 References

COORS, V., HAGEDORN, B., THUM, S., REITZ, T. & GUTBELL, R., 2017: 3D PORTRAYAL SERVICE 1.0. (15-001r4).

DOJO TOOLKIT 1.13, 2017: dojotoolkit.org. Retrieved 14 January 2018, from https://dojotoolkit.org

GAILLARD, J., VIENNE, A., BAUME, R., PEDRINIS, F., PEYTAVIE, A. & GESQUIÈRE, G., 2015: Urban data visualisation in a web browser. Proceedings of the 20th International Conference on 3D Web Technology - Web3d '15. http://dx.doi.org/10.1145/2775292.2775302

GUTBELL, R., PANDIKOW, L., COORS, V. & KAMMEYER, Y., 2016: A framework for server side rendering using OGC's 3D portrayal service. Proceedings of the 21st International Conference on Web3d Technology - Web3d '16. http://dx.doi.org/10.1145/2945292.2945306

I3S-SPEC, 2017: GitHub. Retrieved 12 January 2018, from https://github.com/Esri/i3s-spec

INDEXED 3D SCENE LAYER FORMAT SPECIFICATION, 2017: GitHub. Retrieved 12 January 2018, from https://github.com/Esri/i3s-spec/blob/master/format/Indexed%203d%20Scene%20Layer%20Format%20Specification.md

KRÄMER, M. & GUTBELL, R., 2015: A case study on 3D geospatial applications in the web using state-of-the-art WebGL frameworks. Proceedings of the 20th International Conference on 3D Web Technology - Web3d '15. http://dx.doi.org/10.1145/2775292.2775303

KYOUNG-SOOK, K., DONGMIN, K., HYEMI, J. & HIROTAKA, O., 2017: Stinuum: A Holistic Visual Analysis of Moving Objects with Open Source Software. Proceedings of the 25th ACM

SIGSPATIAL International Conference on Advances in Geographic Information Systems, 1-4.

LU, Z., GUERRERO, P., MITRA, N., & STEED, A., 2016: Open3D: crowd-sourced distributed curation of city models. Proceedings of the 21st International Conference on Web3d Technology - Web3d '16. http://dx.doi.org/10.1145/2945292.2945302

PLAY 2.6.X DOCUMENTATION, 2017: Playframework.com. Retrieved 12 January 2018, from https://www.playframework.com/documentation/2.6.x/Home