Cartography

(Int. Master's Program)

Lehrstuhl für Kartographie

Technische Universität München

Master's Thesis

Visualizing dynamic spatial height information in a dam monitoring context

Author: Amir Ashkan Baghdoust

Supervisors: Prof. Thomas H. Kolbe

MSc Kanishk Chaturvedi

MSc Juliane Cron

30 June 2017

Declaration of authorship

I hereby declare that the submitted master thesis entitled **Visualizing dynamic spatial height information in a dam monitoring context** is my own work and that, to the best of my knowledge, it contains no material previously published, or substantially overlapping with material submitted for the award of any other degree at any institution, except where acknowledgement is made in the text.

Munich, 28th of June 2017 Amir Ashkan Baghdoust

Abstract

Natural disasters constantly threaten the safety of our infrastructures including dams. The potential damages associated with dams usually stem from the failure of the structure due to overtopping, seepage water flow, and deformation. Therefore, the implementation of monitoring techniques is essential in detecting any potential damage in an early stage. In creating such monitoring systems, the Open Geospatial Consortium (OGC) Sensor Web Enablement (SWE) suite of specifications enable the gathering of real time information from heterogeneous sources by introducing a standard interface for time series. Although many researchers has implemented SWE based monitoring systems, there has been little work in combining the Sensor Observation Services and the Web Processing Services (WPS) with 3D visualization methods. Thanks to the recent advancement in WebGL and HTML5 this research uses the WebGL-based frameworks such as Three.js and Cesium Virtual Globe JavaScript Libraries to create 3D models on the browser without a need for any plug-ins. Moreover, this research mainly focuses on developing a framework for visualization of dynamic data acquired from web services such as sensor observation services and Web Processing Services. This framework will help Cesium's virtual globe access and visualize the dam model and the sensor water levels from the Sensor Observation Services. It also visualizes the dynamic height observation data and the interpolated water body acquired through Web Processing Services in a web-based application. In addition, the application will provide a further analysis tool in the form of charts that would allow in-depth examination of changes in height values.

Keywords: dam monitoring systems, 3D visualization, HTML5, WebGL, virtual globes, Sensor Web Enablement (SWE), Sensor Observation Services (SOS), Web Processing Services(WPS)

Table of Contents

List of Figures

List of Tables

1 Introduction

Natural disasters constantly threaten the safety of dams as man-made structures that are subject to direct natural exposure. Potential damages associated with dams usually stem from the failure of the structure due to overtopping, seepage water flow, and deformation("Research Project TAMIS," 2014). This makes the implementation of monitoring techniques essential to detect any potential for damage in an early stage.

Relevant data for dams' risk management involve information from hydrological measurements, geological data, and the dams' constructions standards. This information is currently collected independently and lacks the temporal and spatial consistency. Thus, preventing the integration of the measured elements for a comprehensive analysis. In this context, a dam monitoring system allows for the combination of the multiparametric data acquired from the ground sensors with geoprocessing and visualization capabilities. Numerous properties can be measured in such dam monitoring systems including temperature, precipitation and physical properties such as seepage and water level at the "Air" site.

Increased seepage may be associated with internal erosion in the dam. Internal erosion is one of the main reasons for dam failures (Sjödahl, Dahlin, & Zhou, 2006). However, it's hard to detect this internal erosion by conventional methods. Hence, new approaches should be developed to monitor seepage water with the help of its contributing factors such as dam's water level and precipitations. Considering the fact that the water level is measured in height above sea level, 3D visualization can help the user acquire more knowledge and information from these sources in contrast to 2D visualizations.

Also, by providing the variation of the water level through time, there can be a more comprehensive monitoring system for the user. All these are made possible through the introduction of Sensor Web infrastructures that enables a setup to access real-time data observed by sensors. This data can be used in combination of the 3D web-based technologies to represent a real-time visualization of the current state of water levels in the dam.

With the help of 3D visualizations, users can grasp a more realistic view of the objects in comparison to the conventional 2D plans, hence, making its application ideal for the case of water level monitoring. In addition, with the advancements in computer graphics, high computational devices and the latest trend in web technologies (such as HTML5 and WebGL), the realization of 3D models in the web environment is easier than before. The combination of HTML5 and WebGL enables the web browsers to provide tools for analyzing and representing the 3D world without the need of any plug-in.

Furthermore, the developments of the WebGL-based frameworks such as the JavaScript libraries Three.js and Cesium Virtual Globe have provided good possibilities in creating 3D content without the need for low-level programming.

1.1 Motivation

The introduction of WebGL and HTML5 enables the realization of 3D visualization directly within the browser without the need of a plugin. And various virtual globes such as Cesium Virtual Globe have been developed based on this combination. One of the main challenges in 3D Visualization using the WebGL and HTML5 combination technologies is the visualization and analysis of 3D objects. These objects can be buildings or in the case of a dam monitoring system the infrastructure models and the water surface generated from height points. These objects are usually represented using CityGML that will include the spatial and graphical aspects of the objects along with the attributes.

The Digital Earth envisions a multi-resolution, three-dimensional representation of the planet in order to find, visualize, and make sense of vast amounts of geo-referenced information on the physical and social environment (Craglia et al., 2012). To achieve this goal standardized observations infrastructures and easily available visualization technologies should be put in place. However, the lack of efficient visualization systems through the web and interoperable frameworks that allow standardizing the access to the city models, have limited the use of these datasets and various researchers try to overcome these bottlenecks. The work of (Chaturvedi, Yao, & Kolbe, 2015) defines a framework and implementation of a web-based 3D client for processing, visualization, and analysis of very large semantic 3D city models. Consequently, it represents how Cesium can be used in 3D visualization and illustrates how it can be tailored for objects with complex semantics. The efficient visualization client introduced in the mentioned paper allows for the interaction with CityGML features. Therefore, the 3D models can now be easily implemented with web based Cesium applications.

Furthermore, (Schilling, Bolling, & Nagel, 2016) also provides an alternative solution to render huge 3D city models on the web browsers. They evaluate the newly introduced glTF formats usage in combination with Cesium.js. This paper also considers the use of glTF/B3DM/3D Tiles in Cesium.js which stores the vertices separately and increases the rendering performance and provides storable attributes.

Systems with support for the OGC Sensor Web Enablement (SWE) suite of specifications are capable of gathering information from heterogeneous sources by introducing a standard interface for time series. This technology is proven to be reliable and is being used in many projects in firefighting and pollution control. In addition, the application of Sensor Web in flood warning scenarios has been demonstrated by (Spies & Heier, 2008). Geoprocessing systems provide algorithmic functionality for spatiotemporal data. This is possible when geoprocessing functionalities are published as web services and standardized interfaces.

Considering all the work above in the field of SWE based monitoring systems, there has been little work in combining these Sensor Observation Services with 3D visualization methods. Nonetheless

(Bröring, Vial, & Reitz, 2014), tried to present an approach for processing real-time sensor data streams to enable scalable Web-based 3D visualizations while focusing on processing efficiency.

Despite all the developments in the field, many research challenges including finding the optimal information density and incorporation of 4th dimension in the form of time different elements within the 3D models have not been thoroughly discussed.

The addition of time in an interactive GIS poses interesting challenges both conceptually and regarding implementation (Arsenault et al., 2004). However, platforms such as Cesium.js provide the necessary building blocks for time varying data visualization that has not been yet fully taken advantage of. The work of (Chaturvedi & Kolbe, 2016) proposes a new concept, "Dynamizers," which allows integrating dynamic and time-dependent data with semantic 3D city models such as CityGML. This approach not only allows representing dynamic data in different and generic ways but also enhances spatial, thematic and appearance properties of static city objects by dynamic property values. In addition, the Dynamizers also establish the explicit links between sensors and the respective properties of the city model objects that are measured by them. However, the technology is yet to be implemented in 3D geodatabases.

Furthermore, as these visualization technologies become more mainstream there will be more demand for more real-time data visualization, this gives a great opportunity for integration of real-time data sources such as Sensor Observation Service with the currently available frameworks such as Cesium.js.

However, considering the challenges regarding visualizing of spatiotemporal data and taking into account the advantages of WebGL based platforms such as Cesium.js, this study is motivated toward developing a framework for visualization of dynamic data acquired from web services such as sensor observation services and Web Processing Services. This framework will help Cesium access and visualize the dam model and the sensor water levels from the Sensor Observation Services. It also visualizes the dynamic height observation data and the interpolated water body acquired through Web Processing Services in a web-based application.

1.2 Research Objectives

The core objective of this research will be to develop a framework to implement 3D visualization of water level data, the dynamic variation in the height levels and the seepage water level using the OGC's Sensor Web Enablement (SWE) framework and Sensor Observation Services data within the Cesium.js as a browser-based WebGL enabled platform. The application would provide further analysis tool in the form of charts that would allow in-depth examination of changes in height values.

1.3 Research questions

- What is the most suitable format to visualize water bodies and the surrounding building structure for the mentioned dam monitoring system in the web browser using Cesium virtual globe?
- What is the most efficient way to retrieve dynamic data from Web Processing Service and Sensor Observation Service?
- What is the most efficient way to visualize such time-dynamic variations using Cesium.js?
- How can the dynamic 3D visualization model of water level within the dam monitoring system help real-time prediction of potential natural hazards and detection of irregularities in the dam structure?
- How can the gauge readings be visualized within a line graph as a navigable feature along the timelines?

1.4 Thesis structure

The organization of this thesis is presented as follows:

Chapter 1: Introduction, this section includes the general overview concerning the topic. It discusses the background and the motivation behind the research. The research question and objective are also presented in this chapter.

Chapter 2: Literature Review, the chapter will elaborate the essential components of the research. The chapter will also include the relevant standards and technologies necessary for the study.

Chapter 3: Data preparation, this part will introduce the utilized datasets and the additional steps required for incorporating the data into the visualization pipeline.

Chapter 4: Design and Implementation, the methodology used in integrating the OGC standards and the data models in the Cesium framework is described. In this section, the high-level architecture and its essential implementation components are proposed.

Chapter 5: Results and discussions, in this part of the research, the obtained results at each implementation step is presented.

Chapter 6: Conclusion and Recommendations, as the final chapter, the section will answer the primary research questions and consequently, provide some recommendation for future studies.

2 Literature and Standards review

This section describes all sensor web technologies besides the browser components and the 3D object models necessary for the proposed study. In addition, the previous works in the field of Dam monitoring context visualization of CityGML data are presented.

2.1 3D visualization in the context of Dam Monitoring Systems

Ensuring the safety of Dams as a man-made structure is only possible if there is a comprehensive monitoring system in place. In light of this, there have been many attempts to present the dam inspecting bodies with an online monitoring system that can oversee the infrastructures condition through day and night and help the user prevent any possible damage to the dam and the surrounding environment. Such systems try to couple 2D graphs with real-time data in order to infer some understanding from the gathered data. This is visible in the works of (Zimmerman, Jordan, & Newell, 2016) where they create a portal for the users to access charts and reports and see possible alarm notifications quickly. The system enables the integration and combination of sensors and data sources from different vendors. In these systems, there is no representation of geospatial data. Other toolkits provide some level of geospatial visualization by integrating a 2D map of the area within the monitoring toolbox. (Yang, Bao, Liang, Mi, & Yang, 2009) Is an example of this approach where sensor data are visualized on the map using OpenLayers[1] library. The interface in this application helps detect any overflow of the dam.

In relation to 3D Dam data visualization, typically the researchers focus on highlighting different parts of the dams. Such studies include visualization of geometric surfaces, lithological and hydraulic level properties done by (Dominguez-Acosta, Granados-Olivas, Hibbs, Eastoe, & Hawley, 2004) and the visualization of groundwater and surface features for hydraulic erosion for various types of dams carried out by (Chen et al., 2011). These models are not browser based and are specific to one feature of the dam. Moreover, the final output is not part of a complete dam monitoring toolkit.

On the other hand, some research visualizes the dam body and its elements in 3D representation inside a web platform in order to maximize the accessibility of the tools. In these visualizations such as (Pantea, Hudson, Grauch, & Minor, 2011)The data is mostly coupled with additional information from various sources to enhance the understanding of the context. Also, (Wu, Cui, & Zhong, 2012) and (Fan et al., 2016) depicts a web-based 3D visualization of the dam based on Unity3D[2] Game engine. The first paper integrates the dams' dynamic data from a database with the 3D models on a web client to represent the current state of the dam to the construction managers. In this application, the models are created using 3ds Max before being fed into the Unity3D engine. The latter paper, however, shows the 3D visualization system for dam's

[1] https://openlayers.org/
[2] https://unity3d.com/

foundation curtain grouting. In this work, a combination of parameters measured on the site is sent to the server using a short distance wireless network. This information is then visualized using the Unity3D engine.

The available approaches in using the Unity3D engine in creating the visualizations can be extremely cumbersome. This is because Unity3D doesn't have access to the DOM elements when creating interactive features. Moreover, the Unity3D engine is mainly a game engine and doesn't only focus of web based visualizations. The proprietary nature of the product also discourages the implementation of the tool in a commercial Dam monitoring context.

Considering all the mentioned literature, it is evident that little or no work has been carried out to visualize OGC Sensor Web Enablement data as part of dynamic 3D visualization toolkit that would enable in-depth analysis of the dam's conditions in a browser environment using open source libraries.

2.2 Relevant standards

The standards used in conducting this research is disrobed in this section.

2.2.1 CityGML

City Geography Markup Language (CityGML) is an open data model and XML-based format for the storage and exchange of virtual 3D city models. CityGML models both complex and georeferenced 3D vector data along with the semantics associated with the data. CityGML allows defining different thematic modules such as buildings, streets, vegetation as well as water bodies. Additionally, it provides functionality to represent the scale of the specific object with the help of five consecutive Levels of Detail (LOD) (Gröger, Kolbe, Nagel, & Häfele, 2012). The 3D objects become more detailed with increasing LOD.

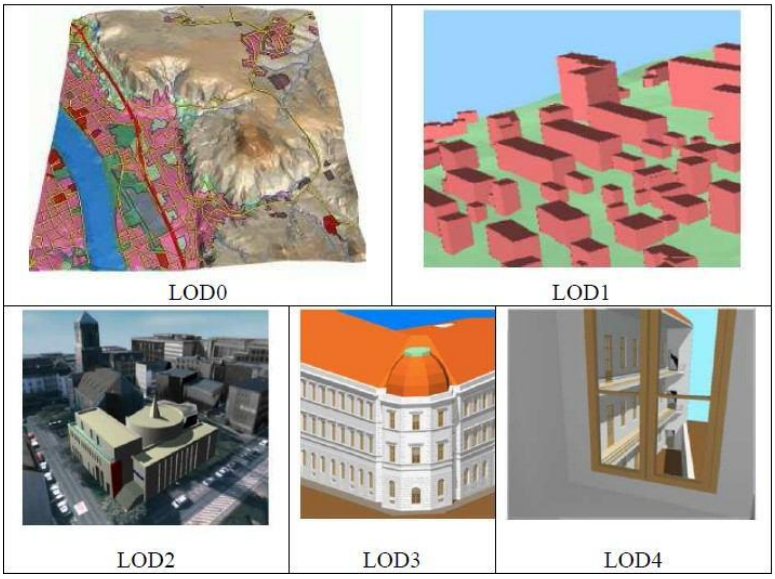

Figure 2-1: The five LODs defined by CityGML (Gröger et al., 2012).

Multiple modules represent the version 2.0.0 of CityGML (based on OGC 12-019) (Figure 2-1). The vertical modules provide the definitions of the different thematic models like building, relief (i.e. di,digital terrain model), city furniture, land use, water body, and transportation, etc. The waterbody module (based on OGC 12-019, cityGML V 2.0.0) is of great importance in this research. The module will represent the three-dimensional geometry of the underground waters, and it also includes a dynamic element of *WaterSurface* to represent temporarily changing situations of tidal flats. The module can also include the optional WaterGroundSurface and WaterClosureSurfaces which represent the basin and the boundaries between waterbodies respectively.

The Figure 2-2 depicts the UML diagram of the Waterbody module

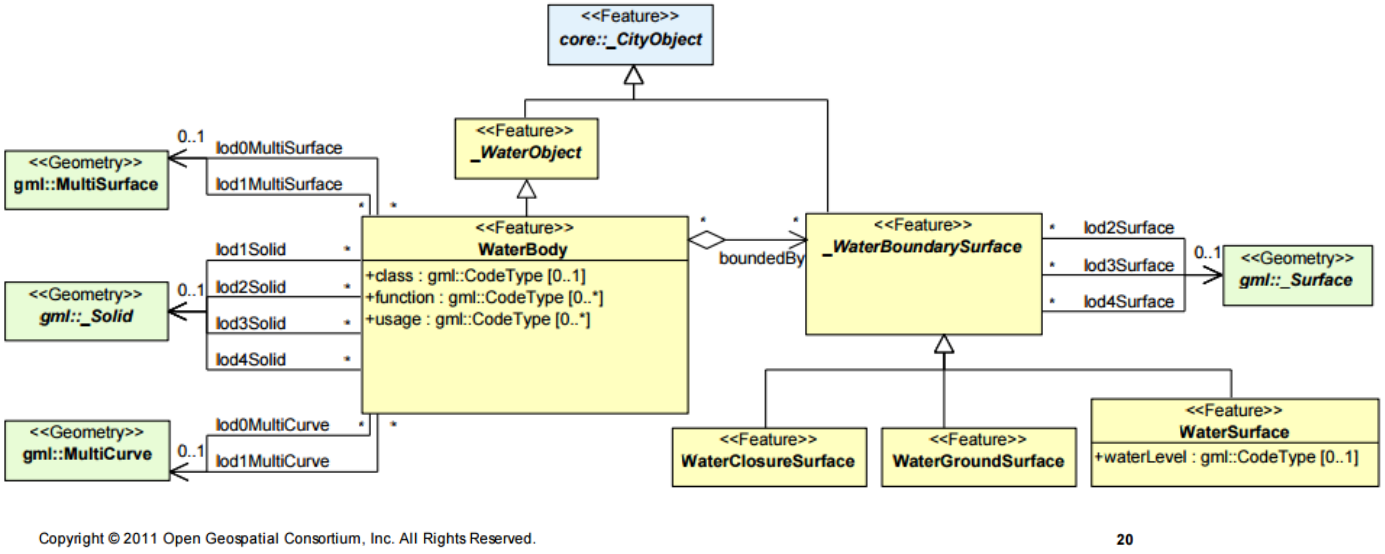

Figure 2-2: UML structure of the Waterbody CityGML standard(Gröger et al., 2012)

The LOD1 mainly used in this research will include a highly generalized surface and is represented as *MultiSurface*. Based on the diagram this attribute can be assigned to a combination of different geometry types. However, for this research, the polygon representation of the geometry is selected.

The other utilized cityGML modules include the Transportation, Tunnel and Generic cityGML elements.

The tunnel module(based on OGC 12-019, cityGML V 2.0.0) used in this research represents the control tunnel under the Dam structures. The model supports the representation of thematic and spatial aspects of tunnels and tunnel parts in four levels of detail(Gröger et al., 2012) . The chosen level of detail for this project was the LOD3 without the walls to better depict the underground infrastructures.

Figure 2-3 shows the UML diagram of the Tunnel module. In this module, the *_AbstractTunnel* is the key class which is a subclass of the thematic class _Site. This class can either be

specialized to a *Tunnel* or to a *TunnelPart*. A *_AbstractTunnel* usually consists of *TunnelParts*, which again are *_AbstractTunnels*.

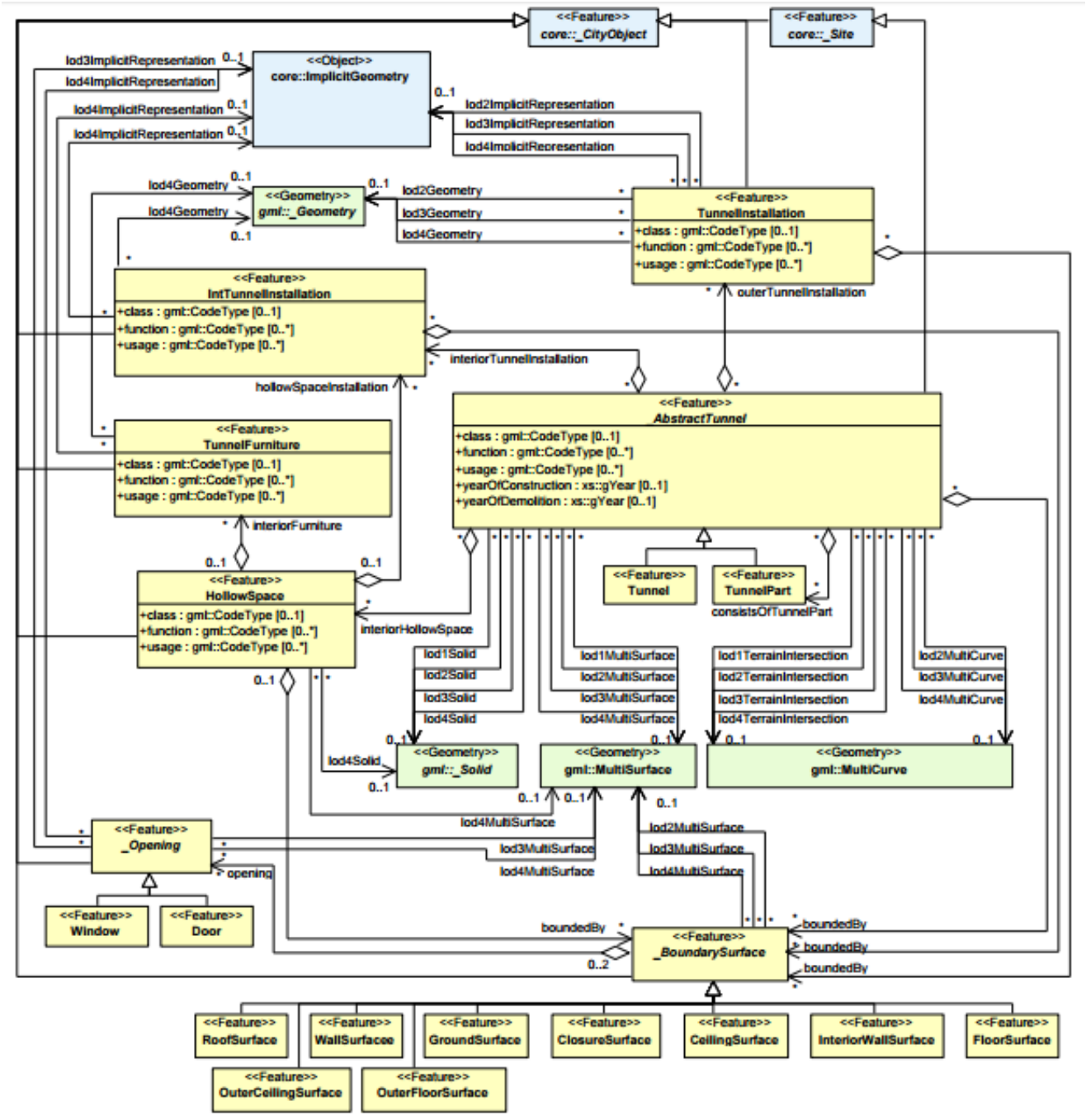

Figure 2-3: UML structure of the Tunnel CityGML standard(Gröger et al., 2012)

Moreover, the transportation objects represent the thematic and special aspects of the road data; these objects are described by 3D surfaces in the Transportation CityGML module (based on OGC 12-019, cityGML V 2.0.0). The LOD 1 illustration of the data is chosen to visualize the surrounding dam area. Figure 2-4 depicts the UML diagram of the CityGML Transportation

model. Based on the OGC standards description (Gröger et al., 2012) the main class in this model is *TransportationComplex*, which represents a road or any other major transportation feature. In the chosen LOD1 this class provides a surface geometry for describing the shape of the road. This can be broken down into different sections of the road network as various *TrafficAreas* however in this research only the general shape of the roads is considered.

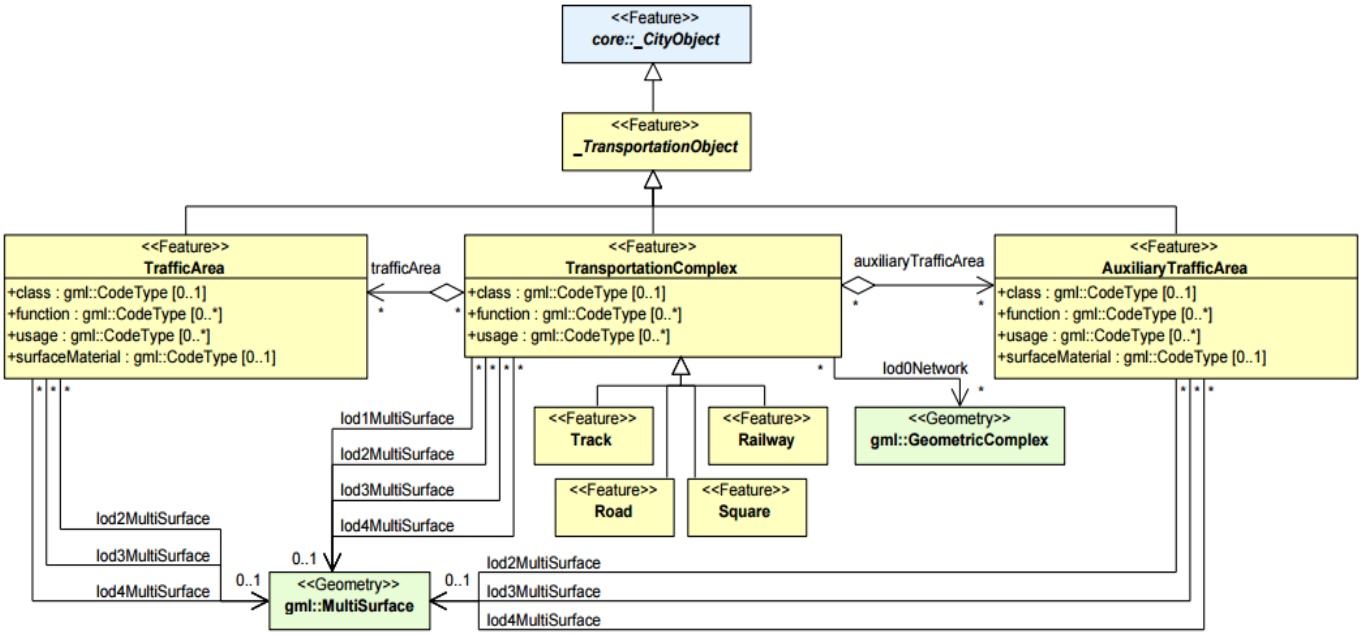

Figure 2-4:UML structure of the Transportation CityGML standard (Gröger et al., 2012)

Other dam facilities such as waterways and the integral dam structures also need to have some standardized 3D representation. Since none of the existing cityGML models represent these facilities, they are visualized using the Generic CityGML Module. These generic extensions to the CityGML data model are realized by the classes *GenericCityObject* and *_genericAttribute* defined within the thematic extension module *Generics(Gröger et al., 2012).*

To represent the geometry of the *GenericCityObject*, an arbitrary 3D GML geometry object GML3 geometry is used as an explicit representation. The following Figure 2-5 denotes the UML diagram of the generic CityGML modules.

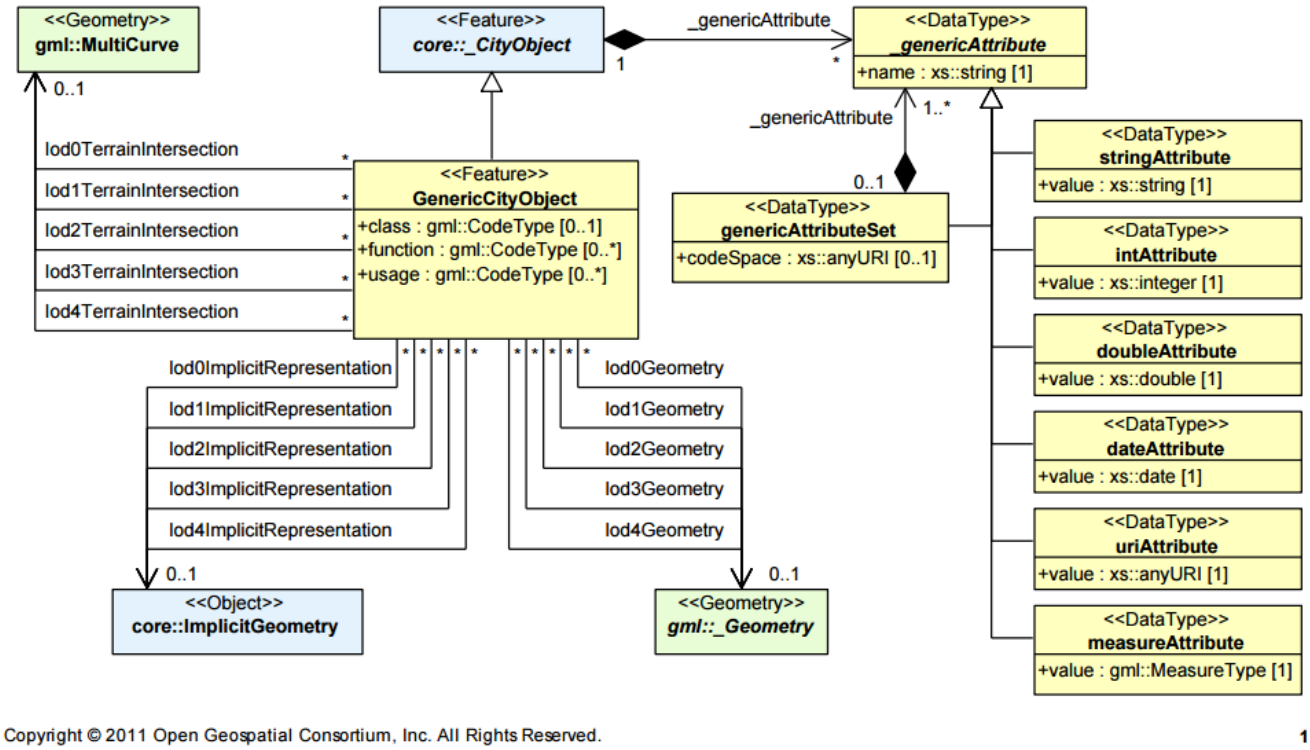

Figure 2-5:UML structure of the Generic object CityGML standard (Gröger et al., 2012)

2.2.2 OGC Web Processing Service

The OGC Web Processing Service (Version 1.0.0, OGC 05-007r7) Standard describes how to access geospatial processes from a web interface. The service provides client access to pre-programmed calculations and/or computation models that operate on spatially referenced data(Schut, 2007). The geoprocessing offered by this service based on the definition of online geoprocessing presented by (Hofer, 2015), can be described as the manipulation of the geospatial data for generation of novel web based outputs. This involves numerous operations from intersection to interpolation models.

The main operations within the WPS standard are threefold. The operation to obtain the service metadata (*GetCapabilities*), The operation to obtain the process metadata (*DescribeProcess*) and the operation to run processes (*Execute*) (Mueller & Pross, 2015). Creating a complete request requires adding of parameter identifiers and values to these operations. This result can be sent to the WPS server using HTTP Get, where the server will respond by sending an XML file or using HTTP Post where an XML file is sent from the client to the server. The HTTP get is used mainly for the GetCapabilities, and DescribeProcess operations and the HTTP Post is primarily used for the Execute operation. The execution process can be run synchronously or asynchronously. The asynchronous execution is favored for the more time-consuming processes. Therefore, in this type

of execution, an immediate status information response is sent to the user after the request has been sent. The response will also include a result location that can be accessed as soon as the process is finished.

The output of a WPS process is always an XML document. And just like the input requests, the output XML's structure is defined in the OGC WPS 2.0 Interface Standard (14-065). The existence of a prescribed XML structure enables the developers to create applications that can read the WPS output documents.

2.2.3 OGC Sensor Web Enablement initiative

The OGC Sensor Web Enablement (SWE) standards suite specifies interfaces and data encodings to enable real-time integration of heterogeneous sensor networks. In this way, most types of sensors can be discovered, accessed and reused for creating web-accessible sensor applications and services. It focusses mainly on geo-sensors; whose georeferenced location is an important factor and produces georeferenced observation data. SWE contains two important information models (Mike Botts, Reichardt, & Outreach, 2006):

- Sensor Model Language (SensorML) (version 2.0, OGC 12-000), which defines an XML schema for describing the processes within sensor and observation processing systems, and provides information needed for discovery, geo-referencing, and processing of observations(M Botts, Robin, Greenwood, & Wesloh, 2014).
- Observations & Measurements (O&M) (Version 2.0.0, OGC 10-025r1) ,which is a generic information model for describing observations(Cox, 2011).

Sensor Modelling Language details the sensing procedure attribute defining a skeletal framework to model sensing devices(Mike Botts, 2007) .According to O&M, SensorML models an entity that performs observations(Cox, 2011). It models physical sensing devices as processes, enabling the transformation of input into an output. Although its focus lies on modeling physical sensor systems and processing of sensor observations, it can be applied in a broader way for modeling any type of process and process chains(Mike Botts, 2007).

O&M, on the other hand, provides a model for observations, their results, and supplementary attributes. It has been approved as an ISO standard (ISO/TC211, 2010). The second version of this implementation is currently available. An observation herein is defined as an act performed by a procedure, such as a sensor, over time or instant. Its result is an estimation of the value of a property of some feature. Besides that additional information, such as observation time, spatial location, the feature of interest or the sensing procedure can be listed (Cox, 2011).

In addition, SWE provides different interface models and web services. The most important service within the scope of this research work is the Sensor Observation Service (SOS) (Version 1.0.0, OGC 06-009r6) (Arthur Na & Priest, 2007). It defines an open interface by which a client can

obtain observation data and sensor and platform descriptions from one or more sensors. The response of the SOS is encoded in O&M and uses the SensorML specifications to provide an interface to make sensors and sensor data archives accessible via an interoperable web-based interface.

SOS aims to carefully model sensors, sensor systems, and observations in order to cover all different kinds of sensors and to support the different requirements of users in the case of using sensor data in an interoperable way(Arthur Na & Priest, 2007). The SOS response represents the aggregate readings from live, in-situ and remote sensors. SOS allows a user to send requests based on spatial, temporal and thematic criteria (Bröring, Echterhoff, Jirka, Simonis, & Lemmens, 2011). Moreover, implementing this system helps increase the accessibility to the different environmental data in a critical situation. In the SOS used for this project, the values of the reading are shown as time series for each sensor station and enable the querying of sensor reading for any desirable timespan.

2.2.4 CityGML Dynamizer ADE

While CityGML is a useful tool in simulating 3D contents, it currently lacks the support for time-varying properties. Dynamizers can be described as a mechanism for storing dynamic values separately from the original attributes in CityGML. This feature is an extension to CityGML which stores dynamic variations and overrides the specific properties of the CityGML feature property (Chaturvedi & Kolbe, 2016). The proposed schema of the output CityGML contains dynamic values in special types of features. These values are considered as ‘modifiers’ to the static values of the CityGML feature attributes. The dynamizers are defined as feature types consisting of attributes *attributeRef*, *startPoint*, and *endPoint* (Chaturvedi & Kolbe, 2016). The Figure 2-6 depicts the nature of the Dynamizer feature as a bridge between the dynamic data sources and the city object models.

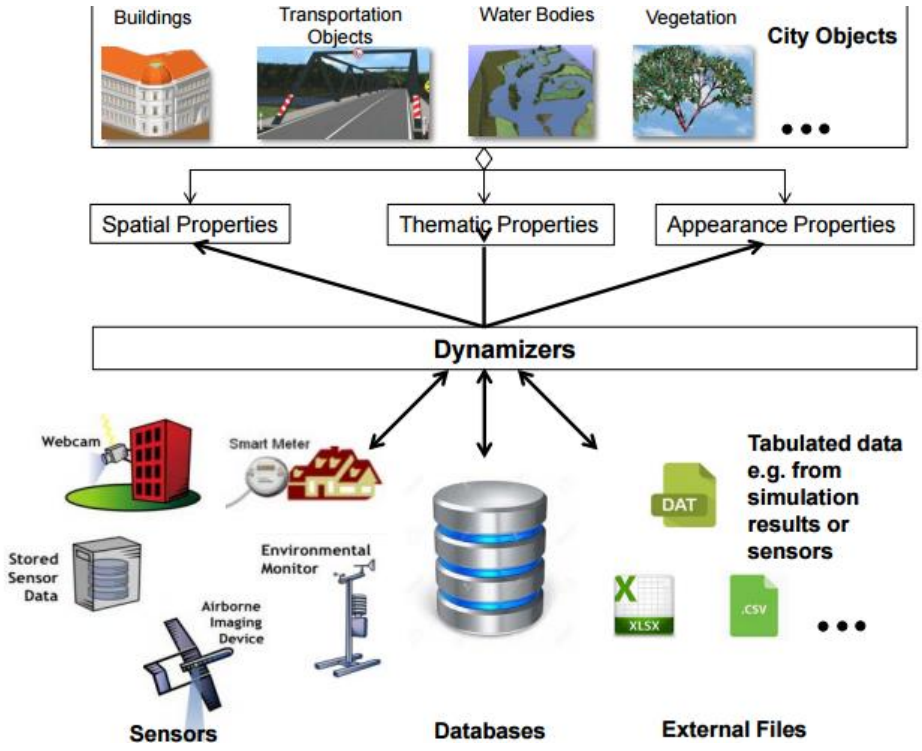

Figure 2-6: The relation of Dynamizers with the input and output sources(Chaturvedi & Kolbe, 2016)

For integrating sensor data inside cityGML, allows for the explicit linkage to sensors. This is done by linking of sensor observations with the respective city model objects. Hence, Dynamizers can enhance static waterbody models used in this research by introducing the dynamic property values.

The mentioned concept is intended to be proposed to become part of the next version of CityGML (version 3.0) therefore it is only considered at the conceptual level in this research. And a practical implementation is not considered.

2.3 Web-based 3D visualization

While CityGML enables better information sharing in the context of 3D models, it also enables the various analysis on the 3D models. However, the sheer size and complexity of the CityGML models hinder the effective browser based visualization of such file. As a result, visualization of CityGML files on the web has become an essential area of research today (Prieto, Izkara, & Delgado del Hoyo, 2012). In order to achieve the plugin-independent visualization of the CityGML data on the browser, the browser friendly 3D formats have to be utilized. These formats and the underlying HTML5 and WebGL requirements for achieving the research goal are described in this section.

2.3.1 3D modeling standards

CityGML can be considered the best suitable standard to represent 3D data's geometry and semantic information. However, the complexity and the large size of the CityGML files hinders their web-based visualizations. Therefore, several 3D standards such as glTF and COLLADA are

introduced in order to visualize 3D data inside a web browser better. This section describes these data standards.

- glTF:

  GlTF(GL Transmission Format) is a royalty-free specification for the efficient transmission and loading of 3D scenes and models by applications. glTF minimizes both the size of 3D assets and the runtime processing needed to unpack and use those assets. It has also been designed with the modern graphics card and web technologies, especially WebGL in mind. The format combines an easily parseable JSON scene description with one or more binary files representing geometry, animations, and other rich data(Khronos, 2016).

  Khronos is promoting glTF as the standard 3D format for the web. glTF is created using a COLLADA digital asset exchange (dae) files. These files were established as an ISO standard in 2013. The parent COLLADA format is widely supported as an export file type option across many 3D software. However, while the collada DAE is a single file, the collada2gltf converter outputs multiple files. Khronos supports both glTF and Collada, besides managing the OpenGL and WebGL standards.

  In glTF rather than capturing the full fidelity of the entire scene data only the essential scene elements necessary for the visualization are kept. Moreover, collada2gltf then optimizes the kept data elements in multiple ways. This process makes the files more readily consumable by WebGL. Depicted in Figure 2-7 is the workflow of 3D models' conversion to glTF format.

Figure 2-7:glTF pipeline progression of content authoring, conversion, delivery, rendering(Trevett, 2013)

- COLLADA:

COLLADA (COLLAborative Design Activity), is an open Digital Asset Exchange Schema for the interactive 3D industry. COLLADA is a standard of the Khronos group[3].The format defines an open standard XML schema from which digital contents of assets can be easily retrieved. COLLADA documents are XML files, usually identified with a '.dae.' (digital asset exchange) filename extension(Khronos, 2016). COLLADA is an intermediate language for transporting data among various interactive 3D applications this means that the file type tries to be as detailed and

[3] http://www.khronos.org/

explicit as possible to represent a complete picture of the visualization. This means that COLLADA will provide comprehensive encoding for geometry, shaders, physics, and kinematics. The high level of details in the COLLADA files undermines their effectiveness on browser-based visualization.

In this research, the two file formats' size and performance are compared in the data preparation chapter to select the most appropriate file format for the visualization.

2.3.2 HTML5 and WebGL

As the GIS applications move from the conventional desktop versions toward web platforms, the need for a platform-independent solution is evident. HTML as the enabler of this visualizations has come a long way from the static pages. The latest revision of HTML known as HTML5 is an extremely powerful platform for running sophisticated applications. The available advanced graphical technologies such as the Canvas element, WebGL and CSS3 3D and scalable vector graphics (SVG) enable the interactive 3D experience on the browser without the need for external plugins.

WebGL as an extension of the HTML5 Canvas element is the standard 3D graphics API for the Web written in a low-level language and is based on OpenGL ES 2.0. However, there are several open source JavaScript toolkits that provide higher-level access to the API to make it look more like a traditional drawing library (Parisi, 2014). Some of the notable frameworks in the context of visualizing geographic data worth mentioning include:

- Three.js: Three.js is a JavaScript based library, which creates 3D contents on the web browser with a very low level of complexity. It is lightweight in nature and can perform rendering with the help of HTML5 canvas, SVG and WebGL (Mrdoob, 2013). The built-in file format support available in Three.js permits the parsing of JSON or COLLADA file formats. In addition, the library provides the necessary interaction by enabling object picking which makes it easy to add interactivity to the applications.

Another solution for visualizing GIS data on the web is to utilize the existing Virtual Globes. These globes enable the visualization of global geospatial data and allow for the interaction between the data and the user. The virtual Globes not only reduces the effort of manually accessing archives of satellite imageries but also allows users to interact and extract content from the globe in real time on the web (Elvidge & Tuttle, 2008). Among the available Virtual Globes such as WebGL Earth (Klokan Technologies, 2011), OpenWebGlobe (Christen & Nebiker, 2011) and Cesium(Analytics Graphics Inc, 2016), Cesium is the only open source solution that has good maintenance by its user community and enable the integration of numerous data sources, creation of cameras and geometry objects. Therefore, this library is chosen and elaborated in following

- Cesium.js: Cesium (Analytics Graphics Inc, 2016)is an open sourced JavaScript Library that enables the creation of 3D globes or 2D maps with only a few lines of code on the web browser. This library has the following features that can help enhance the visualizations:

- Cesium is open source code under the Apache 2.0 license, which means, it is free for commercial and non-commercial use.
- Cesium supports imagery layers using Bing, OpenStreetMaps, ESRI standards and it also supports the integration of imagery from external TMS.
- It also shows vector data from various sources such as KML, TopoJSON, GeoJSON and ESRI shapefiles.
- Cesium provides Cesium a material system to change the objects' appearance to adapt to the user needs.
- It supports math libraries that include the major reference frames such as World Geodetic System (WGS84) and International Celestial Reference Frame (ICRF). The libraries have built-in functions to support the coordinates and Cartesian conversions.
- The Cesium Virtual Globe allows for the visualization of dynamic time dependent elements with the help of Cesium language (CZML).

  CZML is a JSON format for describing a time-dynamic graphical scene, primarily for display in a web browser running Cesium. While Cesium has a rich client-side API, CZML enables Cesium to be data-driven. This gives the generic Cesium viewer the possibility to show a rich 3D scene without the need for any custom code. In many ways, the relationship between Cesium and CZML is similar to the relationship between Google Earth and KML(AnalyticalGraphics Inc, 2016). The easy to parse JSON structure of the CZML files makes way for incremental streaming of data to the client. This means the entire document doesn't need to be present before the scene can be displayed. The most important feature of the CZML format is the accurate description of properties that change value over time. Clients are also expected to be able to interpolate over time-tagged samples. Within CZML every property can be time-dynamic. Figure 2-8 shows an example CZML file structure where a sample property is represented as dynamic values.

```
[
    // packet one
    {
        "id": "GroundControlStation"
        "position": { "cartographicDegrees": [-75.5, 40.0, 0.0] },
        "point": {
            "color": { "rgba": [0, 0, 255, 255] },
        },

         "someProperty": [
        {
            "interval": "2012-04-30T12:00:00Z/13:00:00Z",
            "number": 5
        },
        {
            "interval": "2012-04-30T13:00:00Z/14:00:00Z",
            "number": 6
        },

    },
    // packet two
    {
        "id": "PredatorUAV",
        // ...
    }
```

Figure 2-8:Example CZML file structure (Source: Analytics Graphics, Inc., 2011)

Cesium architecture as a client-side virtual globe is organized in four layers shown in Figure 2-9 below.

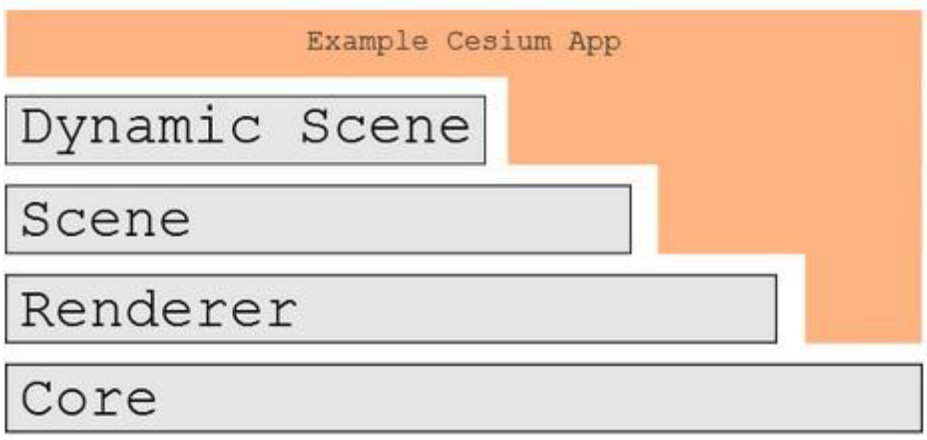

Figure 2-9:Cesium architecture (Source: Analytics Graphics, Inc., 2011)

The image shows the level that each layer is used by the applications. Generally, each layer stacks functionality over the previous layer and raises the level of abstraction. The layers are:

- Core – Contains low-level functions such as the number crunching like linear algebra, intersection tests, and projections.
- Renderer – This layer is a thin abstraction over WebGL. It comprises the already available GLSL functions to provide, textures and shader programs.

- Scene - Scene is mainly built on Core and Renderer to provide relativity high-level map and globe constructs like imagery layers, polylines, labels, and cameras.
- Dynamic Scene – As the top layer of abstraction, this layer handles the time-dynamic visualization constructs including CZML rendering. Instead of frame by frame rendering, this layer enables the storage, loading, and rendering of the data in dynamic objects altogether.

Furthermore, an extensive comparison of the two mentioned libraries is provided in the result section of this thesis.

3 Data preparation

The following section includes two main parts. The first one identifies the study area, the available datasets and the data preparation steps taken for obtaining the initial the necessary preliminary data for the visualization. The second part describes the hardware and software tools required for execution of this research.

3.1 Study area and data

The chosen site for this research is the Bever river dam located in the catchment area of the river Wupper, a tributary of the river Rhine in Western Germany. This area is a subsection of the TAMIS[4] research projects area of interest which is known as the Bever-Block. The Wupperverband [5](Wupper Association) as a responsible body for management of the water volume and water of the Wupper river has established a web-based system for visualization and analysis of the sensor data located along the watershed. The existence of the extensive sensor technologies on the dam and the already implemented web interface has led us to choose this area for the implementation of this research project. The Figure 3-1 below illustrates the relative position of the Bever block area within Germany besides the network clusters of the Bever-Block reservoir system.

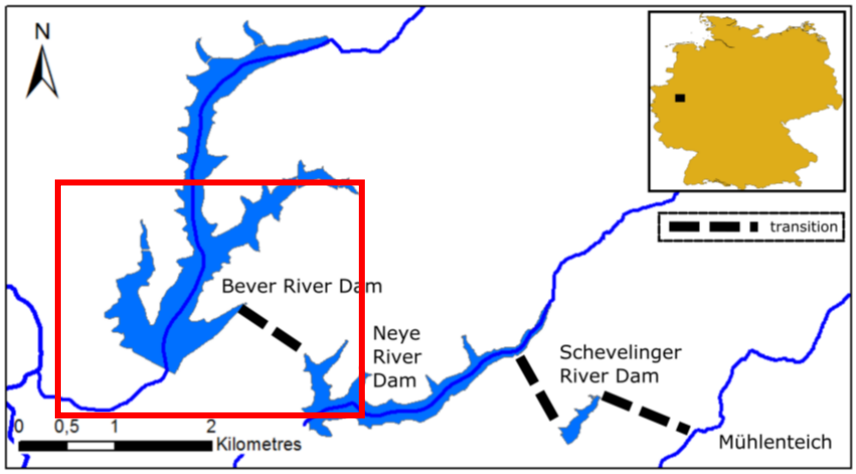

Figure 3-1 : Location of Bever river dam in Bever Block

The Figure 3-2 and Figure 3-3 denote the Bever River dam's aerial view and the plan view of the area respectively. Also, the plan view includes the control tunnel of the dam, the location of the water level sensors and an example of the sensor types with their relative positions. These series

[4] http://tamis.kn.e-technik.tu-dortmund.de
[5] http://www.wupperverband.de

of water level sensors on the ground and seepage sensors inside the control tunnel below the surface continuously monitor the dam's behavior. These sensors are part of various sensor technologies that track the changes on the dam. The control tunnel shown in the image stretches along the dam wall and provides access to the essential dam's infrastructures.

Figure 3-2: Areal View of Bever River dam

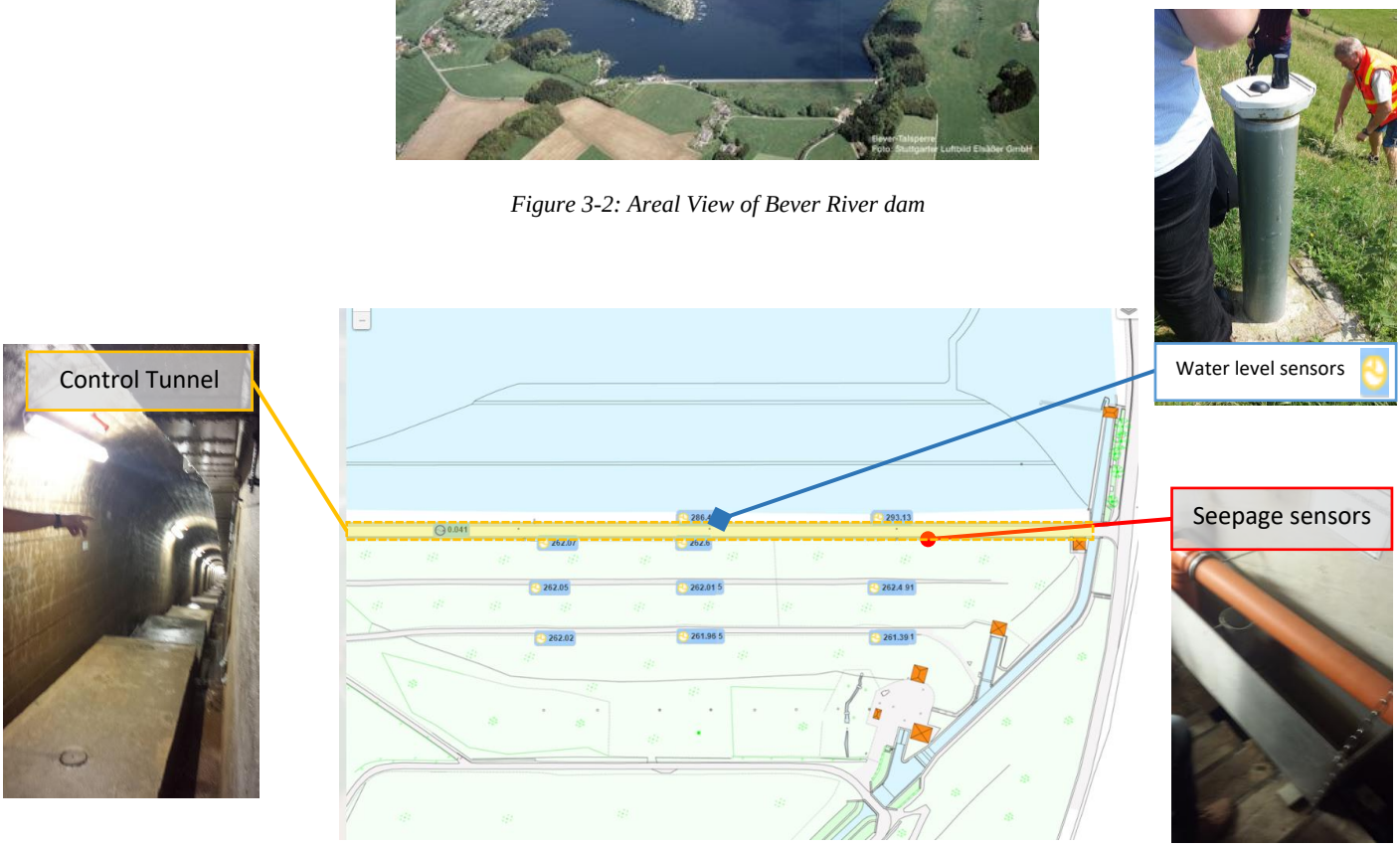

Figure 3-3: Map of the Bever River Dam with the water level measurement stations and the Control tunnel

The Existing Dam Monitoring system at TAMIS project already includes a 3D component developed by the author. This widget uses the powerful Three.js library, described in the WebGL technologies of the second chapter, to show the latest SOS values of the water levels on the terrain model. However, the model has no support for the WPS interpolation surface and dynamic data. The support for the mentioned WPS surfaces and the dynamic data values are implemented as part

of this thesis. In addition, the support for the integration of 3DCityDB[6] exported KML/COLLADA files are developed as part of this thesis to compare the functionalities of Cesium and Three.js. Figure 3-4 pinpoints the currently available 3D widget in the TAMIS control center. The application includes layers' list where the user can select each station and see the latest water levels values.

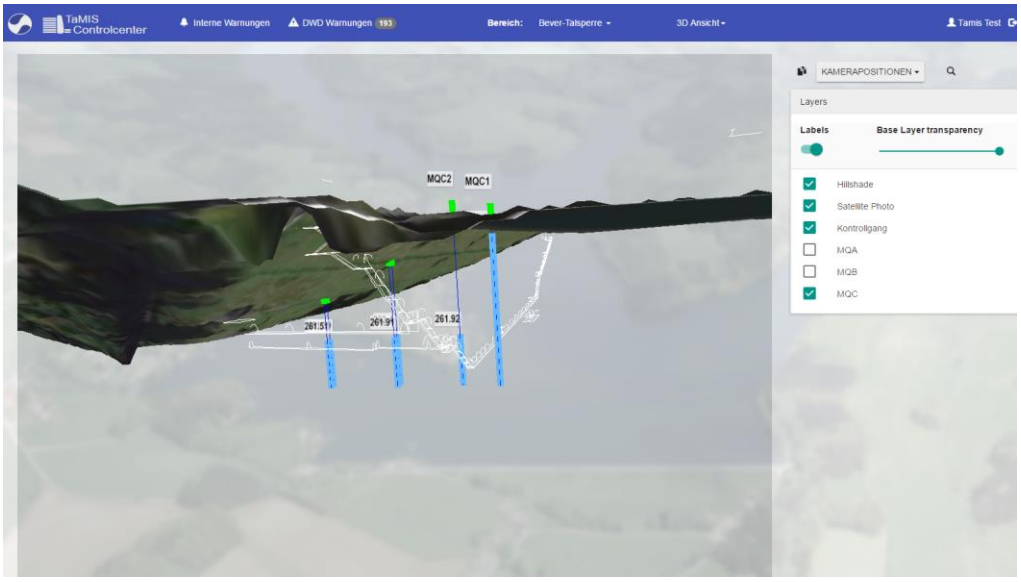

Figure 3-4: Existing TAMIS 3D widget

In the following section, the data used in the research and the necessary processing for their integration into the 3D model are presented.

The required data for caring out the implementation of this application mostly need some level of preprocessing. This is because an efficient visualization pipeline entails standardized data sources that can be recreated for other Dam facilities. Therefore, the available data for the dam are converted into CityGML as the most suitable standard for representing virtual 3D city models. This model will also facilitate the future updating of the models. And since the data is represented in different levels of detail (LOD) the infrastructure data can be further developed to include more detailed model definitions.

3.1.1 Terrain Model

The terrain model for this project includes numerous sources. The original Digital Elevation Model (DEM) for the immediate area around the Bever River Dam has been created using the equally spaced point data provided by Wupperverband. This dataset is converted into a Raster surface

[6] http://www.3dcitydb.org/3dcitydb/3dcitydbhomepage/

using QGIS (Version 2.12.0) providing the DEM model with the resolution of 8.6 meters in cell size.

Notwithstanding, the coverage of this terrain model when visualized on the Cesium globe is not sufficient and creates an island disconnected from the surrounding. To resolve this problem, since Cesium can handle bigger data terrain models, a larger data set with the lower resolution was used to create a more comprehensive terrain model that would comprise a vaster area. For this purpose, the openly available DEM from the DLR[7]'s SRTM X-SAR project with the spatial resolution of 25 meters was utilized. A combined layer from the overlap of the small higher resolution image and the lower resolution DEM creates the broad terrain basis for the visualization (Figure 3-5).

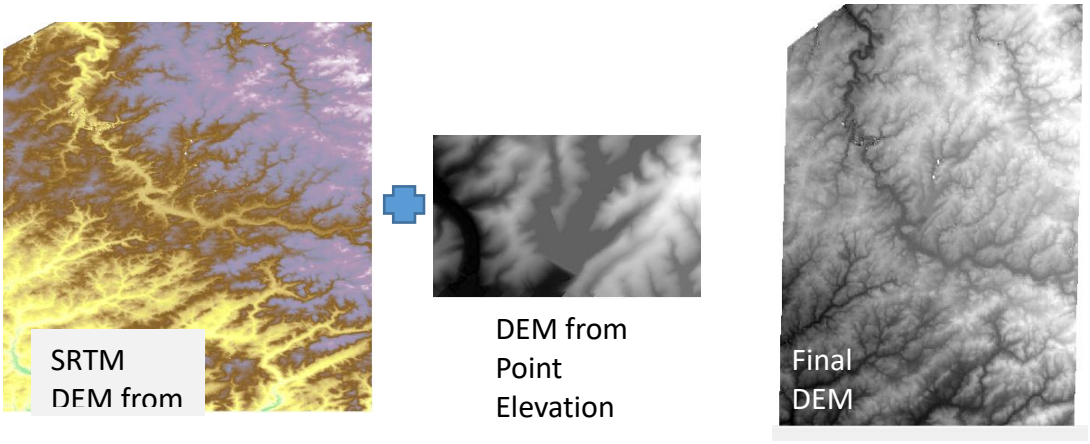

Figure 3-5: The integration of DEM layers

However, to use the generated Heightmap in Cesium, the output raster has to be converted into tile dataset format. Each tile in the Terrain Tile format contains 65 x 65 height values, with small overlap on the edges of the tiles to create a seamless terrain. Cesium translates the Heightmap tiles into a uniform triangle mesh. Cesium also supports quantized-mesh-1.0[8] format for the input terrain data. However, there is no open source software for creating these quantized-mesh surfaces at the time of this research. Therefore, the Heightmap tiles were generated using the Cesium Terrain Builder as a command-line utility developed by the GeoData Institute, University of Southampton [9]. The tool will return a set of tiled ". terrain" files in different zoom levels. This layer folder, when placed on a local server, can be used by Cesium for drawing the ground information.

[7] http://www.dlr.de/eoc/en/
[8] https://cesiumjs.org/data-and-assets/terrain/formats/quantized-mesh-1.0.html
[9] https://github.com/homme/cesium-terrain-builder

3.1.2 CityGML Waterbody object

The water body as the one main dynamic part of the final implementation must use the water surface interpolation data, resulting from the WPS to create the reference waterbody structure for the visualization. This object model structure should allow for future update of the height values.

Based on the need for a standard 3D object element, the waterbody geometry is constructed as a CityGML (Version 2.0) waterbody object. The LOD1 representation for the waterbody seems to show the necessary level of details. Therefore, the final output will have the LOD 1 and show a low level of illustration and high grade of generalization. The process in which the desired surface model is created requires the FME Benchmark and is fully shown in Figure 3-7. The steps for this process involves:

- The acquisition of the reference GeoTIFF from the Web Processing Service is the primary step. In this step, only to have a continuous visualization, the interpolation body of the starting day chosen for the final application is selected to be used as the reference surface waterbody.
- A generalization step to reduce the images pixel sizes. This phase is done to reduce the computation power required when updating the elements. The Nearest Neighbor algorithm resamples the surfaces cells and creates a lower resolution image from the WPS received GeoTIFF image
- The categorization of the height values into a number of groups and the addition of an id attribute. The categories and the id data simplify the update of cell values by providing a reference for each cell.
- The final surface object must follow the OGC standards of waterbody objects. Therefore, the necessary attributes and the structure of the waterbody element needs to be added. Based on The CityGML UML diagram in Figure 3-6 visibly shows that to have the standard CityGML object the surface must have MultiSurface Geometry attribute and the principal waterBounding surface in the output mo

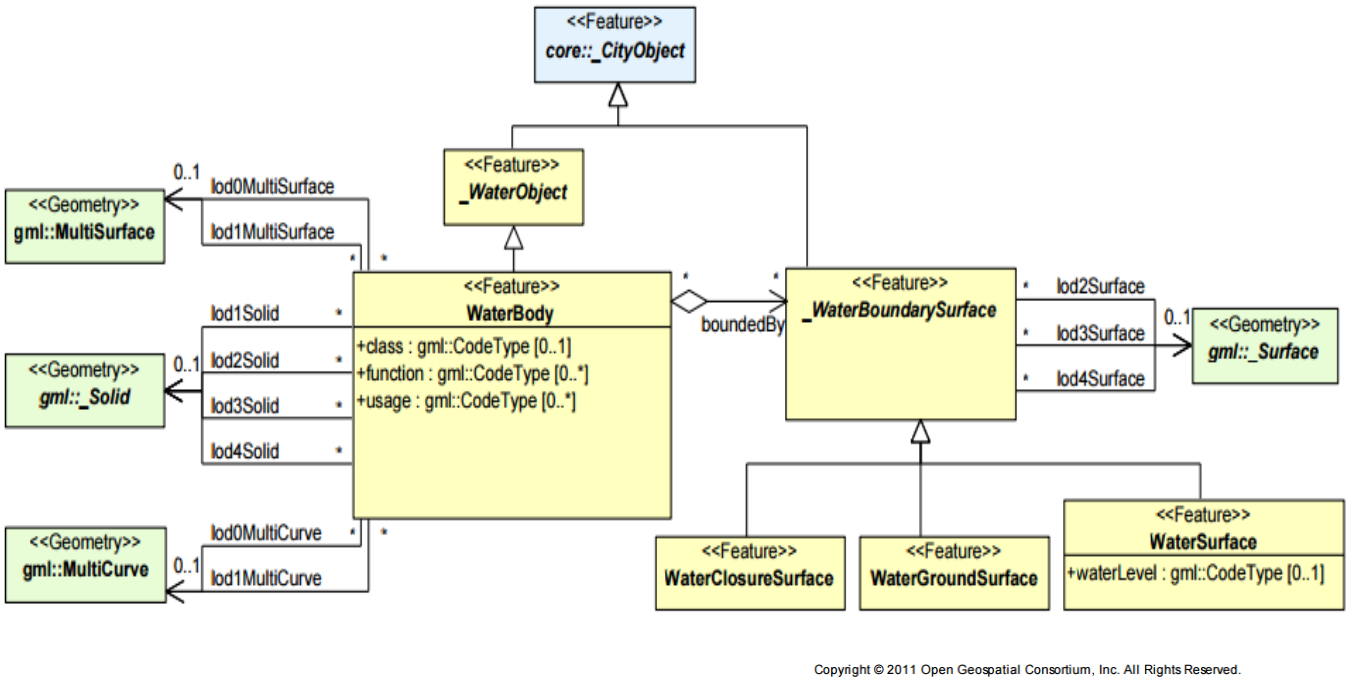

Figure 3-6: UML structure of the Waterbody CityGML standard

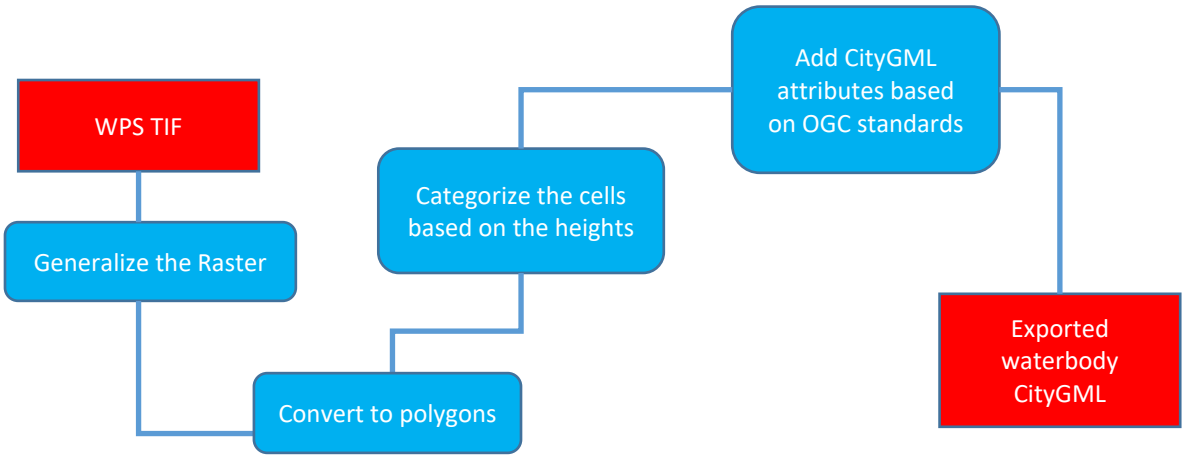

Figure 3-7:diagram of the Waterbody CityGML generation process

3.1.3 Dam infrastructures according to CityGML standards

As described in the introduction of this section, the available layers at the Bever River Dam must also be represented in CityGML (Version 2.0) format to enhance the scalability and consistency of the files for future projects. After creating the 3D objects with surface elements from the original CAD line representations, these features are converted to CityGML using the appropriate schema. The conversion process will define the 3D Shapefile objects as input and like the previous waterbody surface will include the addition of the necessary attributes based on the CityGML Schema.

The existing datasets include the dams control tunnel, the road entities, and the dam's facilities layer and the implemented CityGML schema of each data layer are shown in - Figure 3-10.

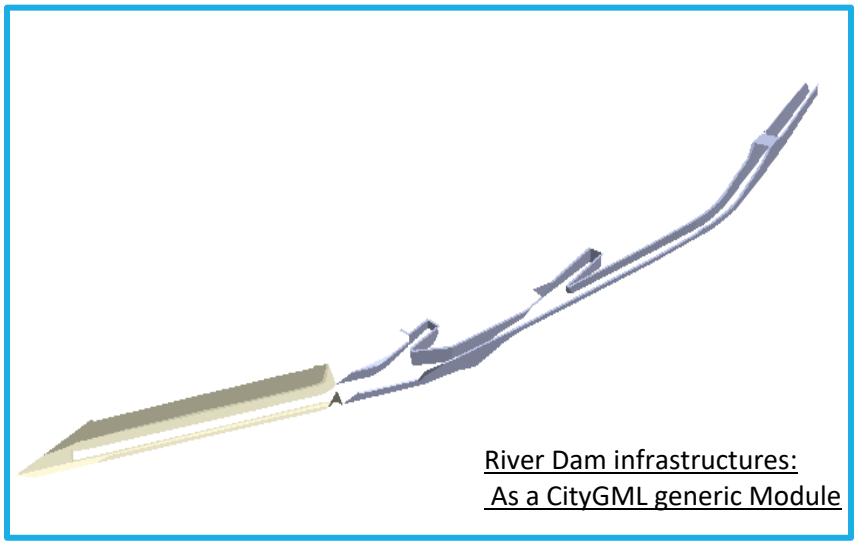

Figure 3-8: River Dam Infrastructures CityGML

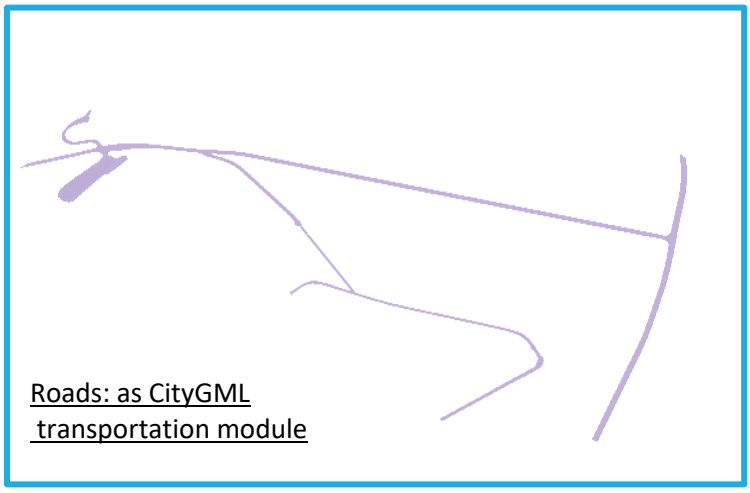

Figure 3-9: Roads CityGML model

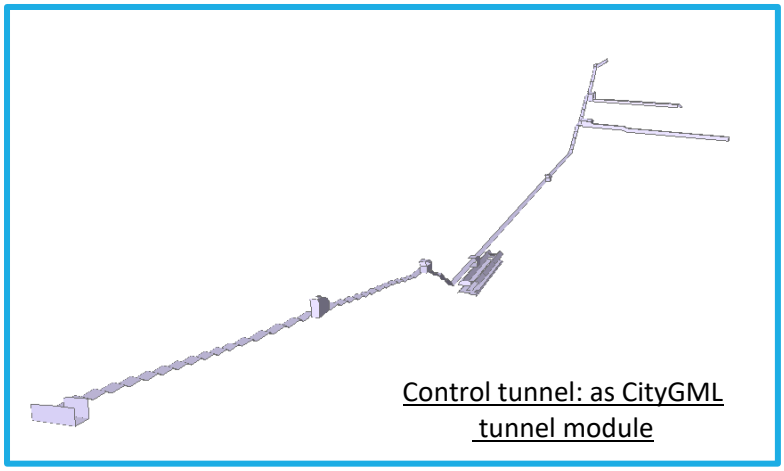

Figure 3-10: Control tunnel CityGML

3.1.4 Web Processing Services

The OGC standardized WPS 2.0 provided by 52°North[10] enables the dynamic value update of the waterbody elevations. The WPS is accessed using a REST API for the TAMIS project and provides a thin access layer to processing functionalities for timeseries data via RESTful Web binding. The Capabilities resource is encoded in the JSON format and provided at the root URL of the API. It lists common metadata like provider information and the available processes. This service provides numerous processing functionalities such as the regression predictions and interpolation capabilities. The Clients can resolve the URL to navigate to a single Process representation. In this research, for acquiring the data for the water body surface and the seepage surface the interpolation process is utilized.

The Image below (Figure 3-11) shows the job resource encoded in JSON with input, output and the time stamp value definition that needs to be posted to the API to create a new job.

[10] [10] http://52north.org/communities/geoprocessing/wps/

```
{
    "inputs": [
        {
            "id": "timespan",
            //      "value": "2016-02-01T10:00:00.00Z%2F2016-02-15T10:00:00.00Z",
            "type": "text/plain"
        },
        {
            "id": "target",
            "value"
                : "https://github.com/52North/tamis/raw/master/geotiff.tiff",
            "type": "application/geotiff"
        }
    ],
    "outputs": [
        {
            "id": "predictions",
            "type": "application/geotiff"
        }

    ]
}
```

Figure 3-11: The request JSON example

Due to the asynchronous execution of the API, a response is immediately returned after the post request. The response URL is regardless of the finishing status of the actual request. If the process is finished, links to the outputs are generated. Figure 3-12 is an example of the results acquired using the WPS service. It demonstrates the predicted water levels for a period from 04.01.2016 to 28.02.2016 using the sensor reading at the defined time stamps.

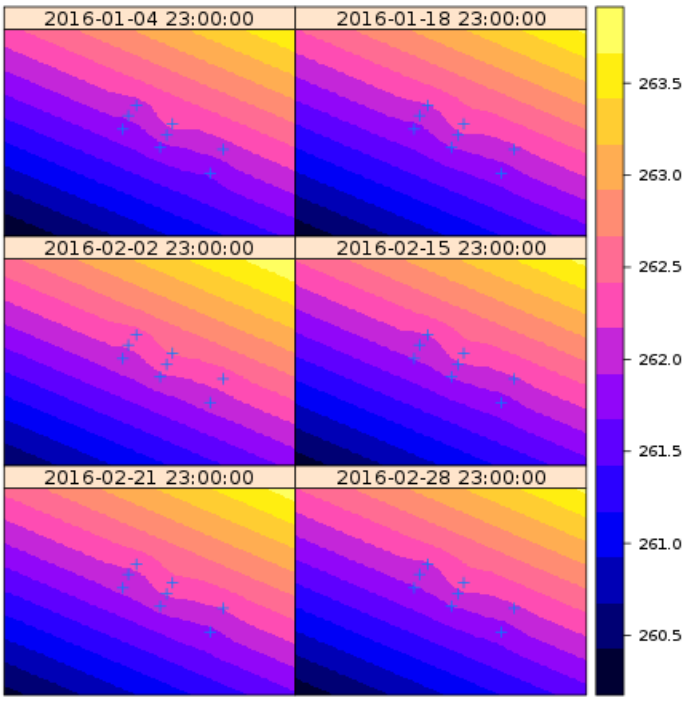

Figure 3-12: Example WPS Response Raster

3.1.5 Sensor Observation Services

The sensors monitoring the water level are accessible using an OGC standardized SOS service. This service is provided by 52°North's Sensor Web community. And it is available using a RESTful Web binding. The timeseries data is queried using the URL link. The example of a query and the structure of the query is described in Figure 3-13.

Query Example for getting the value for a time series in a given time period :

http://localhost:8080/tamis-proxy/proxy?requestUrl=http://fluggs.wupperverband.de/sos2-tamis/api/v1/timeseries/592/timespan=P14DT0h/2016-03-01T05:50:25.10Z

► Structure of the request is:

- http://localhost:8080/tamis-proxy/proxy?requestUrl=http://fluggs.wupperverband.de/sos2-tamis/api/v1 (SOS instance)
- timeseries (SOS Request parameter to get the lists of all timeseries available.)
- 592 (Timeseries of interest id)
- timespan= 2015-11-10T09:00:00Z/2015-11-10T12:00:00Z (iso8601 format of the interval)

Figure 3-13: The SOS Request structure

The response of the queried timeseries data will be a JSON data file. The representation in Figure 3-14 shows the structure of the timeseries data's reply without a timestamp query. This data shows the corresponding station property such as its position and the first and last value. Accordingly, the response shown in Figure 3-14 depicts an output in case of including a time span attribute in the request where only the value for all the timeseries reading in the requested period is returned.

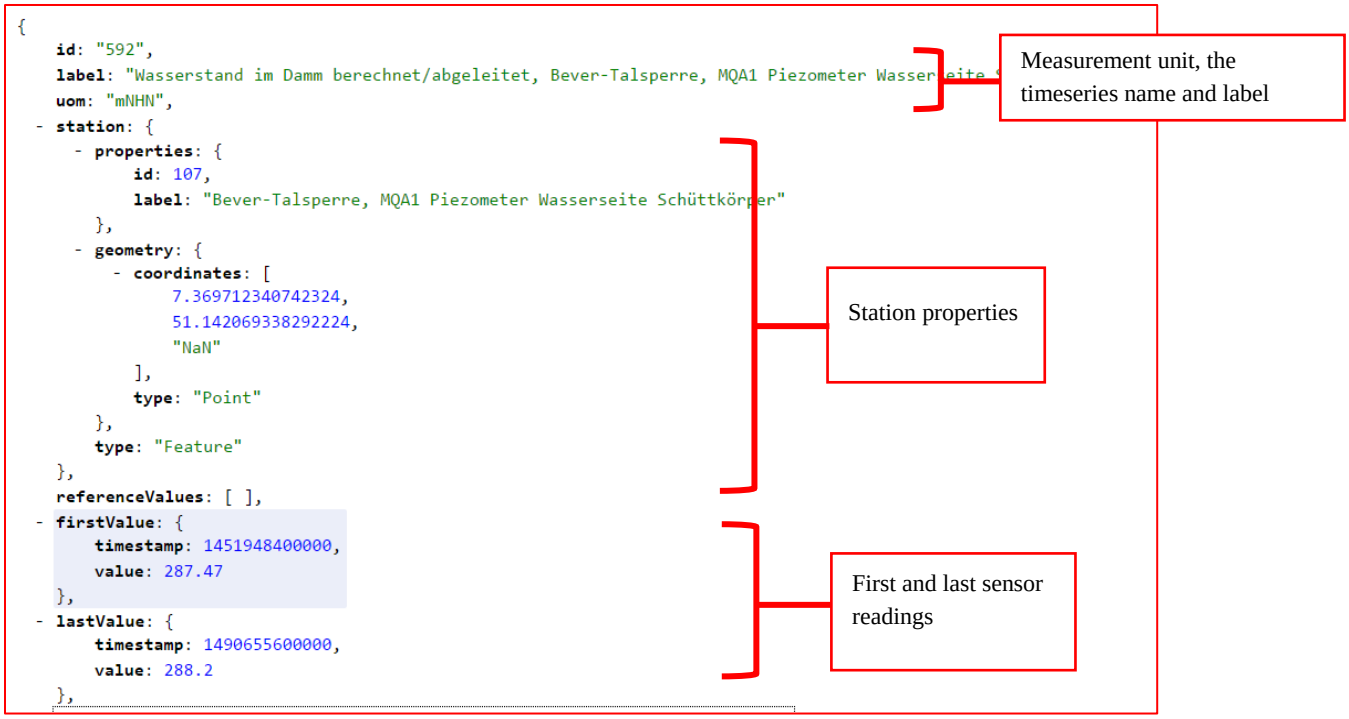

Figure 3-14:SOS response for a specific timeseries

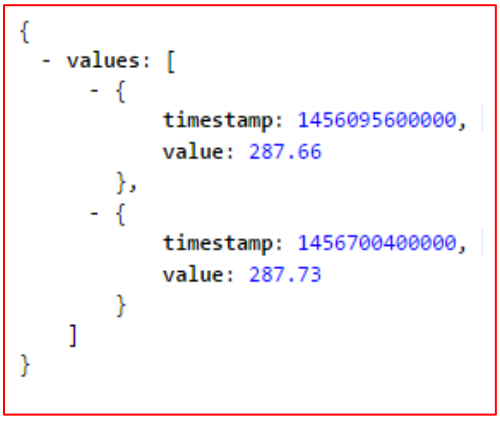

Figure 3-15: response for a specific timeseries

In the final implementation, the data provided by this SOS service is used in updating the station's water tube visualizations.

3.2 Software/Applications used

The necessary software packages and applications required for this thesis and their intent of use is enumerated in Table 3-1 below.

Table 3-1: Used Software

Software/Application	Purpose
QGIS 2.12.0	For converting the CAD dam infrastructure objects into 3D Shapefiles
FME 2016	Creation of a conversion process for generating the CityGML models.
Apache HTTP Server (Version 3.2.2)	Initiating the local web server to allow the communication between the client and server and hosting of the Terrain tiles.
Google Chrome 58.0.3029.96 (64-bit)	The basis for running the web application on web browser
Internet Explorer 11.0	To test the web application on web browser
NetBeans IDE 8.1	The environment for developing the application
TUM 3D-Web-Map-Client	To represent the KML/GlTF models in Cesium
TUM 3DCityDB (Version 3.3.1)	Import of CityGML files to the 3D city database and export as KML/glTF and KML/COLLADA
PostGIS version 2.0	Used by 3DCityDB as 3D city database to store 3D objects
Highchart.js Library (Version 5.0)	Utilized for the implementation of the interactive charts and the gauge indicator.
Cesium.js library (Version 1.33)	Visualizing the virtual globe as the basis for the application
Three.js Library (Version r85)	To develop a similar visualization environment with the Cesium globe to compare the results from two methods.
52°North clients	

3.3 Hardware used

Table 3-2 below indicates the Hardware used in conducting this research. These configurations match the minimum requirement for Cesium based applications.

Table 3-2: Used Hardware

Processor	Intel Core i5-3337M 1.80GHz
RAM	8 GB
Graphics card	NVIDIA GeForce GT 740M
Video RAM	2 GB

4 Design and Implementation

In this chapter, the different possible scenarios for the integration of OGC WPS and SOS data with waterbody objects within the Cesium platform are discussed, and the high-level architecture of the chosen method is represented, and additionally, the constituting components of the implemented solution are described.

4.1 Available approaches

Considering the constant update requirement of the real-time sensor data, choosing the most efficient at the same time most comprehensive method of integration can make a difference in the user experience. The possible options range from the methods that are more server oriented to the methods that rely more on the client side. The most suitable approach would be the one with a reasonable balance between the server and the client side.

Thereby, in the following section, all the possible visualization approaches for such goal in Cesium realm, and their efficacies are presented.

4.1.1 Representation using a KML Network Link file:

In this method, each time stamp would be represented using a KMLNetworkLink (KML Tutorial, 2013). In other words, a single KML file, containing the references of the water surface geometries is created, which will include the height value of each instance for the particular time stamp. Since the 3DCityDB Importer/Exporter provides the ability to export CityGML data to KML/glTF easily, this method seems to be the simplest approach. However, it would require a considerable amount of pre-processing of the data to create the network link for each time stamp. As a result, there will be a bulk of files stored on the server, that might not all even be called by the clients. Therefore, this method while being the most straight forward approach does not represent the best possible solution.

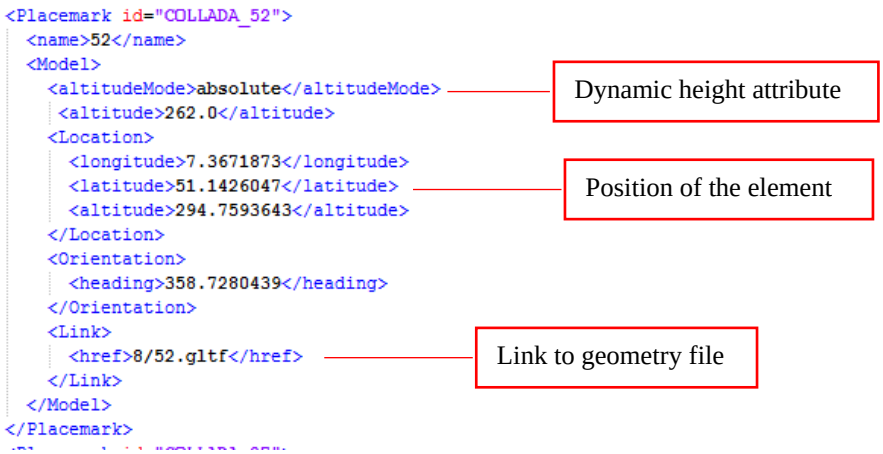

Figure 4-1:KML/glTF structure

Moreover, there can also be another iteration of this approach illustrated in Figure 4-1:KML/glTF structure that would include a single Master KMLNetworkLink file that will have the reference to geometries and the height attribute which will, in turn, be updated using the client's request to the server. This proposed method would eliminate the pre-processing issue, but it would require the interface to read the master KML file again each time the new date is requested, making it still an unpopular method.

4.1.2 Direct object generation using Cesium

Since having an external source as the updating element would require the constant reread of the sources from the server, an alternative would be to directly draw the dynamic elements within Cesium and change the different attributes on-the-fly. The direct rendering in Cesium means that each water body object will be created as a Cesium entity instances which will be fetched from the DOM and updated every time the client requests another time stamp. The result of such approach while acceptable has the following drawbacks;

- The initial loading time will be substantially higher than loading an external format since each object should be created using the script that converts the WPS resulting raster file to entity objects.
- Entity instances do not handle dynamic elements in an efficient way. Therefore, the already existing height values will be lost each time a new timespan is requested. This would cause redundancies in the way the data is loaded.

4.1.3 Representing variations using CityGML Dynamizers

Dynamizers, as introduced by (Chaturvedi & Kolbe, 2016), allows the representation of dynamic and time-varying attributes within semantic 3D city models. As the Figure 4-2 denotes, the concept allows defining a new dynamizer feature type for the CityGML Waterbody object. The dynamizer feature, on the one hand, allows representing dynamic variations of the height values, and on the other hand, allows referencing to the respective attribute of the CityGML object and override its values accordingly. Using dynamizers, the dynamic variations of the Waterbody objects can be represented with the CityGML source file itself. Each geometry object would have a dynamic attribute that would include the post request variables required for getting the water level data from the server. Therefore, once loaded Cesium could eventually load the corresponding WPS layer based on the dynamic attribute.

The Dynamizer concept has already been implemented as an Application Domain Extension (ADE) of CityGML 2.0 within the OGC Future City Pilot Phase 1[11] and is intended to become a part of the next version of CityGML (version 3.0). However, it is currently not supported by the 3DCityDB. Therefore, it cannot be applied for the purpose of this project.

[11] http://www.opengeospatial.org/projects/initiatives/fcp1

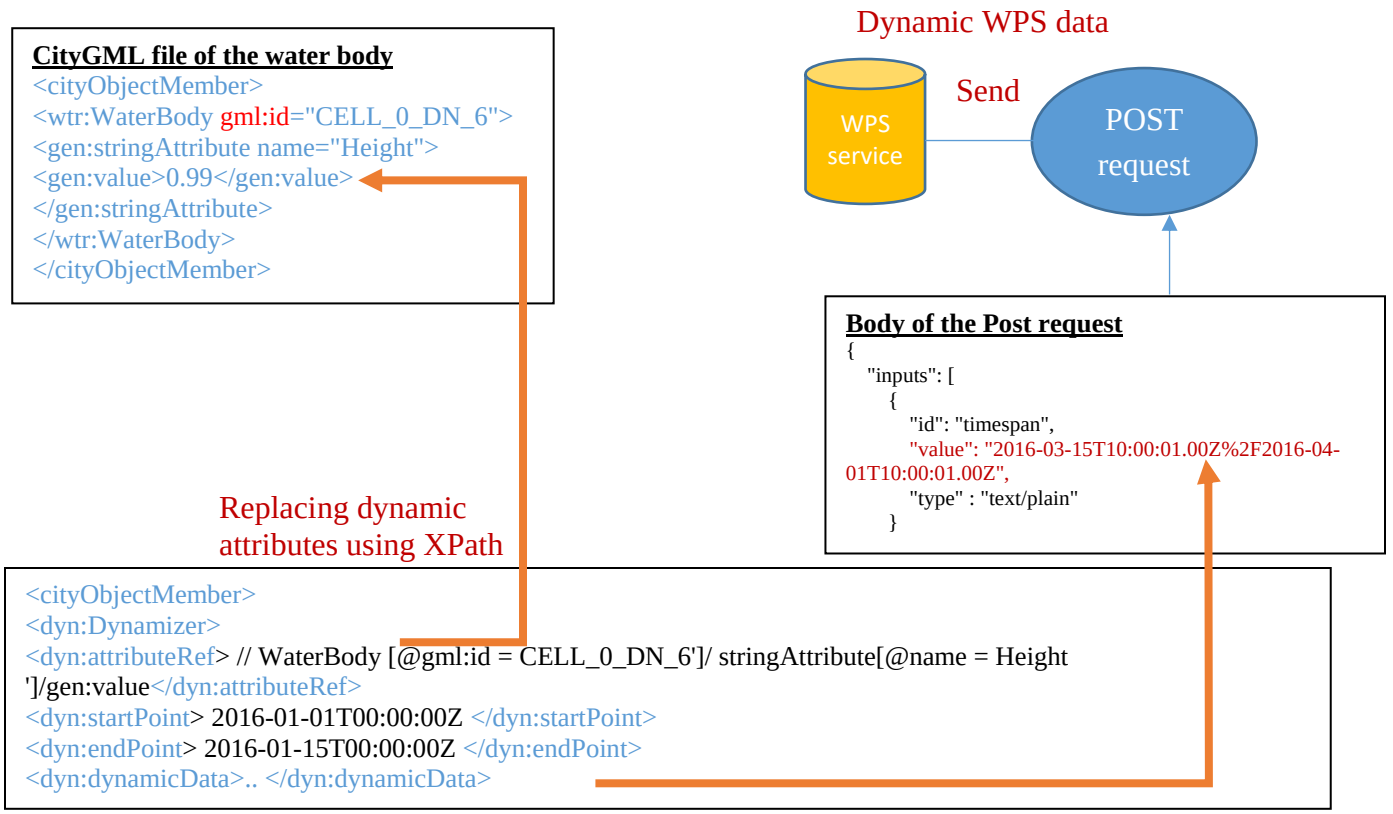

Figure 4-2: Diagram of Dynamizers integration with the WPS

4.1.4 Visualizing the KML files in the Google Earth environment

An additional possible visualization method would be an outside browser solution using Google Earth Pro. Despite the notable improvements in the browser-based visualization methods, there are still good reasons to use the conventional desktop app to visualize 3D data. Furthermore, since Google Earth primarily recognizes KML as its XML-based language, the current KML NetworkLink Link structure of the available data seems to be presentable on this platform. It is important to note that, Google Earth doesn't support the glTF format and the models must be in COLLADA. In addition, while previous browsers allowed the end-user to modify their browser by downloading and installing the Google Earth plugin, this feature is not available anymore, making the desktop based application of this solution the only possible approach. And the newly introduced browser-based Google Earth is still in its production and has many shortcomings in visualizing external data sources. Hence, the possible workflow for such visualization, in the desktop version of the application, is visualized in Figure 4-3. This method would be similar to visualization of the KML NetworkLink file in Cesium globe where the updates are made using a JavaScript code running on nodeJS. However, this would not have the issues confronted in the first proposed method while there is not browser communication in between.

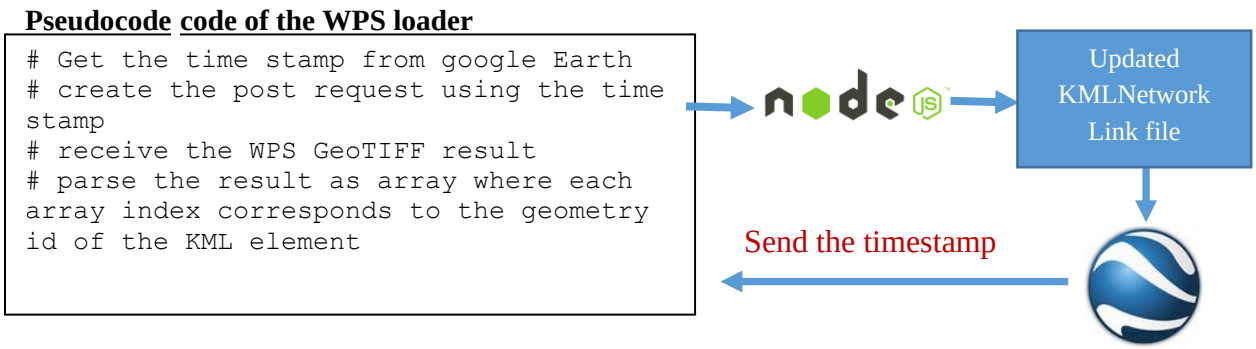

Figure 4-3: Workflow of visualization of KML in Google Earth Pro

All in all, this method would limit our ability to have easily accessible visualization platform. Moreover, Google Earth support can be uneven for machines not running Microsoft Windows. And the constant reloading of the data source can be problematic and would create blinking effects.

4.1.5 Using CZML

As already noted in the literature review section, CZML or Cesium Language is a JSON based language that was designed especially for the Cesium program. CZML can accurately describe properties that change value over time(Cesium,2016). Another additional feature of CZML is that it is structured for efficient, incremental streaming to a client, making it less cumbersome for the client side. And as already discussed in the introduction of this section the ability to provide the balance between the server and the client side is the decisive factor in choosing the optimum visualization method. Accordingly, using CZML will provide for the representation of static elements of the visualization pipeline within the server while the dynamic variation is stored on the client side. Such an ability would provide the opportunity to recall the values for the already existing timestamps without the need for a repetitive request to the server. Each geometry element is depicted in a CZML packet with the glTF model as their model attribute.

The structure of the JSON file will allow for the addition of various timestamps. In this study, the different attribute is the height element of the position attribute, but since the future addition of timestamps to the model is done through Cesium, using the corresponding entity instance, the position values are converted into Cartesian values, which makes the addition of height measurements in meters for each time stamp a big problem. To avoid the unnecessary conversion and the possible inaccuracies based on the file structure of the model shown in Figure 4-4 the node positioning of the glTF models representing each CZML packet is considered for the intervals

Figure 4-4: Structure of the CZML file with the Time interval Collections

In short, it can be said that CZML would provide the best visualization method while it can be flexible enough to incorporate the time dynamic elements and have the speed and lightness to quickly and widely render the items onto the scene. Moreover, this format has the possibility to be extended to communicate additional static or time-dynamic data, such as charts in this scenario, to a more sophisticated client. However, the only limitation with this approach is that it will only be usable with Cesium virtual globe.

4.2 High-level architecture

The following would describe the high-level overview of the architecture established in order to visualize the dynamic data of waterbodies.

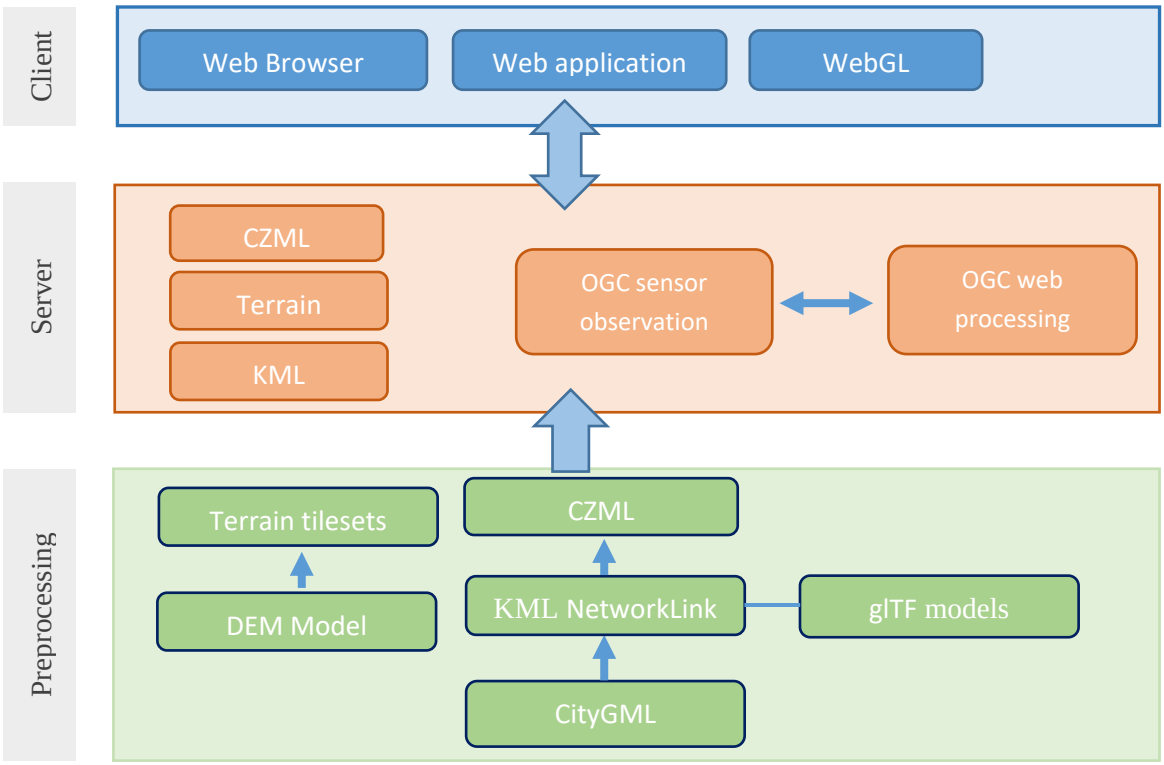

Figure 4-5: Diagram of the High-level Architecture

As visible in Figure 4-5 The whole system architecture consists of three levels;

- pre-processing: This section will include the conversion of the CityGML files into the KML NetworkLink files that will later be parsed into CZML JSON files that will constitute the semantic data of the visualization.

- Server side: The server is where the static CZML data and the dynamic OGC services reside. The first one is the direct result of the pre-processing section while the latter is provided by coupling of the services in TAMIS Geoprocessing architecture. This part of the server communicates the WPS water body results and the SOS readings to the client.

-Client: The web application is the main party in the client side. It will include the interface for the user to visualize and interact with 3D objects. The interface also incorporates the functionality to call new timespans to update the waterbody and the seepage indicators on the map.

Overall this architecture can be better visualized in the detailed integration diagram shown in Figure 4-6 where the cesium platform will use the OGC SOS and WPS REST APIs to add time intervals to the JSON based CZML files within the DOM.

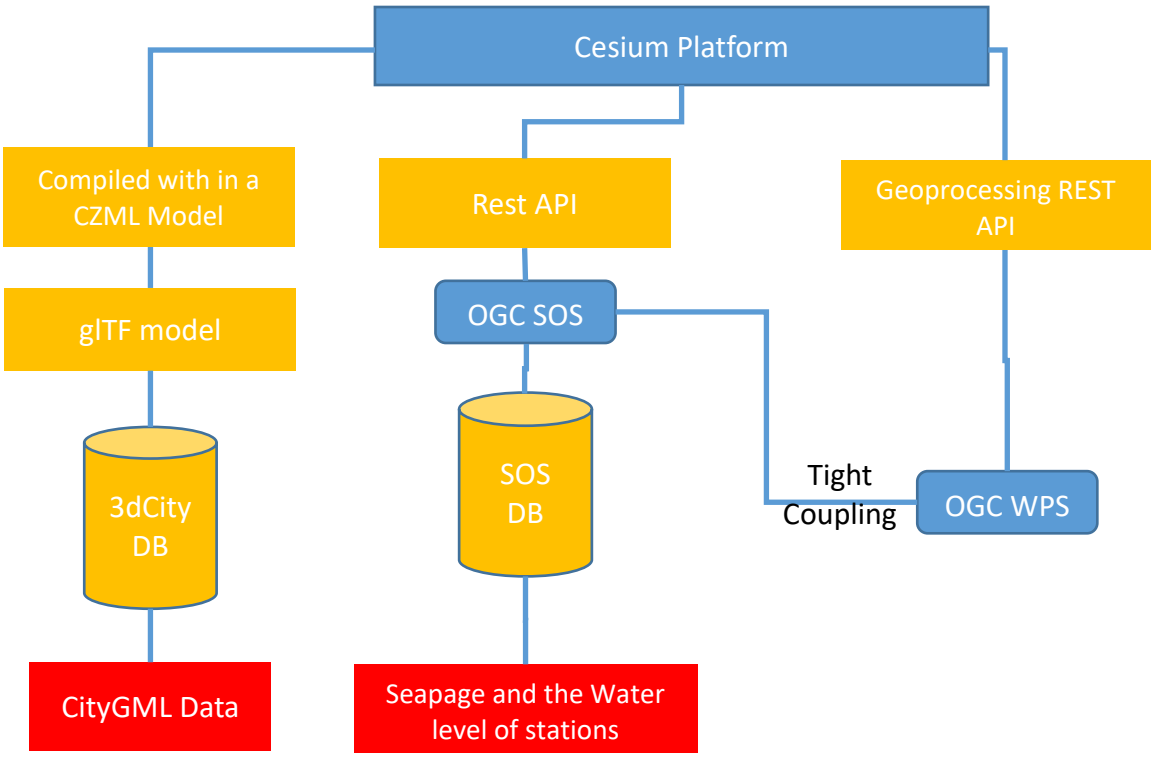

Figure 4-6: Workflow for integration of Models and the OGC services with Cesium

The design of each section is as follows;

4.3 Pre-processing

As described in the architecture diagram the bottom layer will encompass the conversion process. In this level, the CityGML model is converted to a KML NetworkLink file. The NetworkLink file will include the reference to the glTF surface geometries. This is essential since glTF models do not have projected positions and the CityGML model only include the local spatial reference system. Thus, the resulting KML file will have the positions in a global reference system, in this case, WGS84, for each surface member. This KML reference file is used to create the CZML based file for the visualization.

In order to achieve this goal, the 3DCityDB Importer/Exporter is used. 3D City Database as a free geodatabase allows the storage, representation, and managing of virtual 3D city models on top of a standard spatial relational database. As an open source tool, this application allows the import of CityGML file to a 3D city database and export of the contents of the database in the form of KML NetworkLink file. The functionality can be described with the help of Figure 4-7:

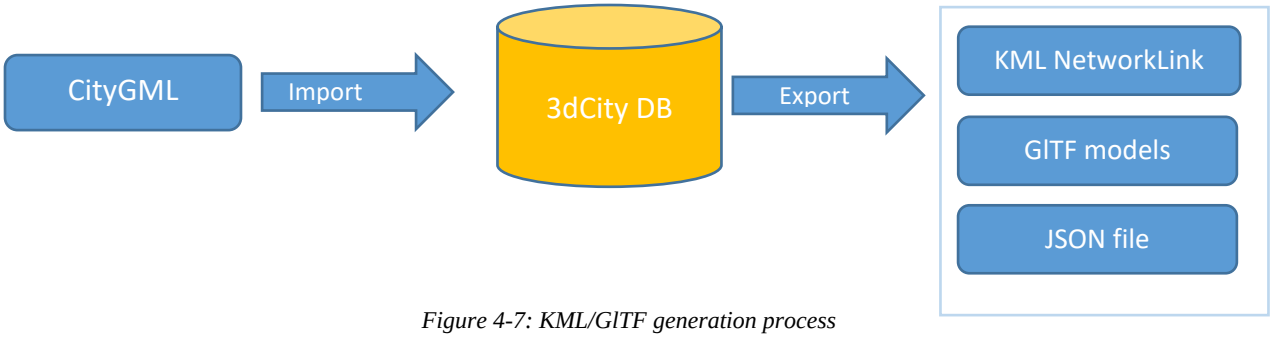

Figure 4-7: KML/GlTF generation process

Based on the figure above, the surface waterbody CityGML model is imported into the database. The toolkit provides the option to import various object types from the source file. However, since the used CityGML only includes the waterbodies, therefore, is it fully imported within the database.

Given the need for the standard relational database for the 3D city database, PostGIS v2.0 is used as the spatial database to store the model data. Having 3D city database installed on top of PostGIS v2.0 can provide the tables to store water body class object of the model and the other object classes of the CityGML standard.

Once the data has been successfully imported into the database, the exporting process can begin.

In order to export the models from the source CityGML waterbody object to a Cesium visualizable model, the 3DCityDB Importer/Exporter is used. This tool can export city models from the 3D city database in KML/COLLADA, and KML/glTF format.

According to the specification of GlTF and COLLADA, it can be said that while COLLADA is designed to include all the details of the scene and the entirety of the available information for import in any 3D content authoring program, the glTF format is mostly optimized for delivery to WebGL-enabled web browsers. Each glTF file component is designed to be as lightweight as possible for minimizing the processing and rendering demands placed upon the Web Browser(Khronos, 2016). This speed and performance difference is also visible with the dataset used in this research. The Table 4-1 below shows the difference in loading speed and the memory space used in the browser when loading a single water surface cell in different browsers.

Table 4-1: comparison of COLLADA and GlTF's performance

	glTF		COLLADA	
Browser	Memory (KB)	Loading Time (s)	Memory (KB)	Loading Time (s)
Chrome	0.476	2.01 s	4.3	17.43 s
Internet Explorer	1.5	2.13 s	4.23	19.24 s

The benchmark was run on a regular laptop, having the specifications mentioned in section 3.2 .

Based on the described merits of the glTF model, the glTF (Khronos Group, Version1.0,2016) format is chosen as the desired format in the exporting feature of the exporter toolset. This toolset uses a tiling strategy and creates a reference of the glTF files with their corresponding tiles in each master KML file using KMLNetworkLink (KML Tutorial, 2013). In other words, a single KML file is created, containing the references of glTF files in each tile and their details such as the geographical location and the gml id. If there are no tiles and glTF models, there will be only one KML file containing geometry and semantic information of the CityGML file.

The Master KML NetworkLink file now needs to be processed to CZML (Version 1, 2016 Cesium) format in order to be imported in Cesium Virtual Globe. The code example in Figure 4-8 best presents the changes made to achieve the CZML JSON-based file format from the KML resource file. In this section, a JavaScript code is run using NodeJS to export the JSON file.

Each of the CZML elements or packages include the references glTF file as their model. In addition, the position and the id element are filled in accordance with the KML file. Accordingly, to improve the speed of time interval addition and eliminate the need for additional projection change, as discussed previously, the node transformation is added to include the current elevation position and the future height time intervals. This file is later visualized in Cesium and will have the ability to have added time stamps on the DOM level.

Sample KML Networkfile element

```
<Placemark id="COLLADA_52">
  <name>52</name>
  <Model>
   <altitudeMode>absolute</altitudeMode>
   <Location>
    <longitude>7.3671873</longitude>
    <latitude>51.1426047</latitude>
    <altitude>262.7593643</altitude>
   </Location>
   <Orientation>
    <heading>358.7280439</heading>
   </Orientation>
   <Link>
    <href>8/52.gltf</href>
   </Link>
  </Model>
 </Placemark>
```

Sample CZML element

```
{
  "id": "52",
   "name": "CELL_52_DN_6",
    "position": {
        "cartographicDegrees":
          [7.3671873, 51.1426047,0]
            },
     "model": {
          "gltf": "8/52.gltf ",
          "nodeTransformations": {
                  "Y_UP_Transform": {
                      "translation": {
                          "cartesian":[0,0, 262.7593643]
                       }
                  }
          }
      }
```

Figure 4-8: KML to CZML conversion process

The process of creating CZML models must be repeated for all the geometry data that would have dynamic properties. Therefore, the Stations and water level tubes must also have the CZML representation. However, Since the SOS service contains the station's details including the locations and other necessary information, nodeJS is used to directly acquire the static geometry information and create the corresponding CZML element.

In order to create the Stations CZML a JavaScript code send a request to the SOS and gets the location of the stations and creates a cylinder CZML instance for each station and as for the water

tubes, the SOS requests gets the value for the initial timestamp, also used for creating the waterbody geometry, timespan and creates a CZML polyline instance.

In addition to these models the terrain tile set needs to be provided to Cesium hence the seamless underlying topography can be created. Cesium offers a number of different sources for the integration of terrain data, one of which is height map data. In order to create the height map Cesium Terrain Builder [12]as a C++ open source library is used. The tile set created with this tool can sit behind a terrain server used by Cesium. This tool was executed using a virtual machine running on Linux.

With the help of this library, from a GDAL raster representing a Digital Elevation Model (DEM), the gzipped terrain tiles are created, saving the resulting tiles to a directory. The terrain tileset will be a heightmap-1.0 file format and will be represented in a simple quadtree pyramid that follows the Tile Map Service (TMS) layout and global-geodetic profile.

The tool calculates the maximum zoom level concomitant with the native raster resolution and creates terrain tiles for all zoom levels between that maximum and zoom level 0 where the tile extents overlap the raster extents and resampling.

4.4 Server side:

The Apache HTTP server has been used in this research, which acts as a local server and is responsible for providing the data to the client side. In addition, in order to avoid the CORS (Cross-origin resource sharing) errors while requesting the SOS links from an external source, a Tomcat server is implemented.

The server side will have two constituting parts, one being a local server and another being responsible for the processing services.

The geoprocessing side of this application mainly manages the request for new time spans for the visualization. On each user request for the SOS and WPS, the server receives a time-dependent link for the first service and a header including the time stamp for the second one. The services in return retrieves the required GeoTIFF in the case of the WPS, and the JSON result in the SOS requests and updates the visualization.

TAMIS Geoprocessing services, used in this research, are separate components. Therefore, each request for the SOS and WPS results are made using a different call. However, while the WPS uses the SOS data, it can retrieve the observations from the SOS without the client's involvement.

[12] https://github.com/geo-data/cesium-terrain-builder

4.5 Client side

Throughout this section, the client side of the implementation is described. This part will include the web-based application, the visualization of 3D elements and on the fly updating of the dynamic data. It will also specify the web browser and the underlying rendering WebGL technology used by Cesium.

4.5.1 Web Browser and WebGL

The initial stage in the implementation of this project will be consideration of different web browsers. Since this technology is part of a Dam Monitoring system, it should have the ability to be integrated into the already existing dam monitoring dashboards. In addition, the implementation of this research requires a HTML5 enabled browser that has the capability to support WebGL specifications. These mentioned criteria limit the implementation of this application to two main browsers. While the web browser comparison is out of the scope of this research, in the following, a short description of the browsers functionalities in running such 3D visualization application is presented.

- Google Chrome (Version 57.0.2987.133): This browser was the chosen browser for the development process of the application. This browser provides a fast and seamless interaction of the user with the models, and the updates are implemented without any lag or interference with the current visualization scene.
- Internet Explorer (Version 11.1066.14393): Since the existing dam monitoring system at TAMIS has the requirement to run on Internet Explorer, due to compatibility with older computers, the application performance was tested in the IE environment. It appears that while IE handles the visualization and on the fly updates in a reasonably acceptable manner, it still has some shortcomings when compared to Google Chrome. The latter presents a faster terrain tile update and a more lag free panning and zooming experience when compared with this browser.

And as for the WebGL API, it should also be mentioned that this API is the factor enabling the visualization of such 3D models in a browser without a need for any plugins. In addition, its provided hardware acceleration enhances the browser's handling of 3D contents. And as discussed in previous sections, the Scene and the Renderer components of Cesium API use WebGL. Hence the Cesium API can provide a development capability in a low-level language for web applications.

4.5.2 Web application

The web based application as the primary component of the project has been developed to run on top of an HTML5 browser utilizing WebGL as an interface for the end-user to perform the functions as part of the Dam monitoring system toolkit. The processing of the application can be described with the help of following process flow diagram in Figure 4-9:

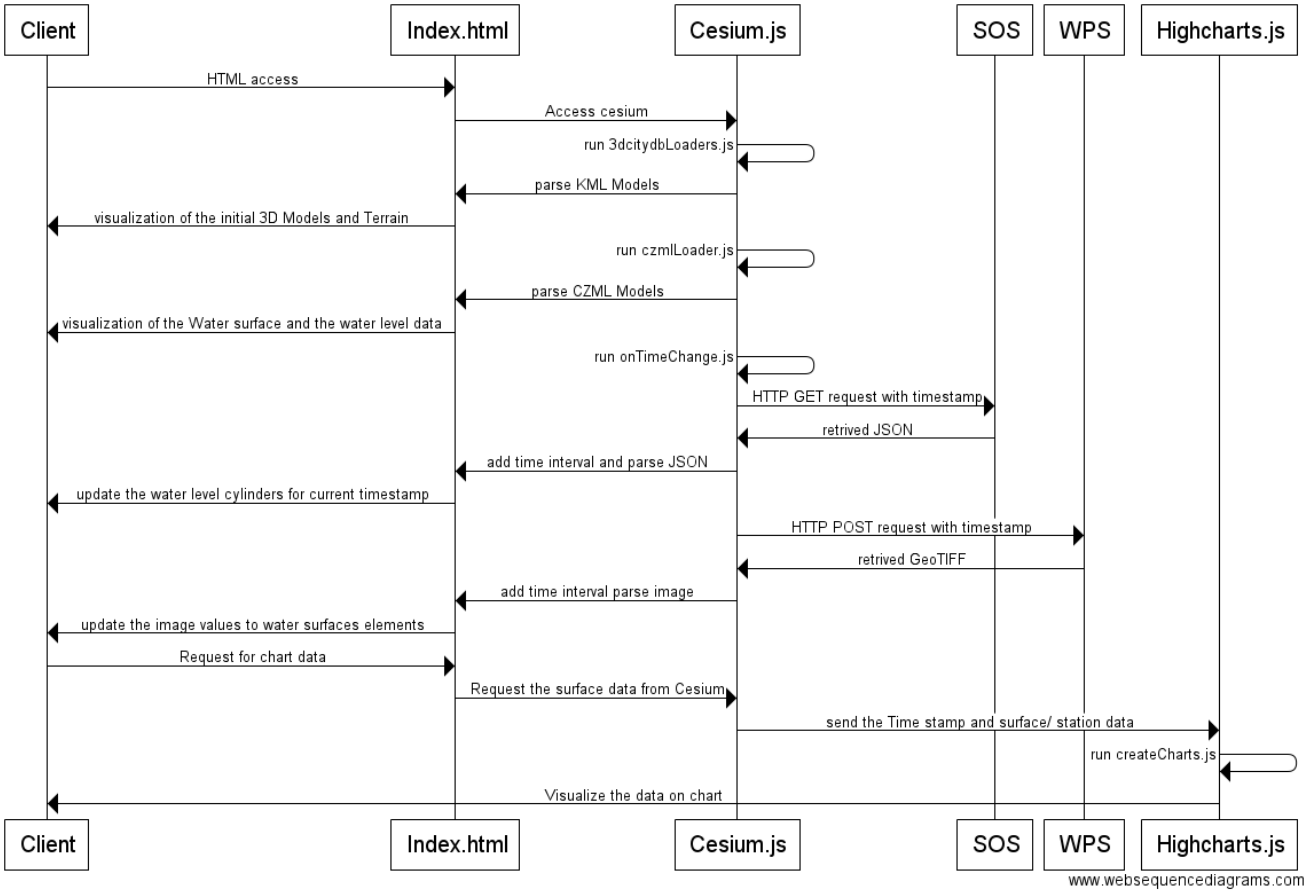

Figure 4-9: Application's flow diagram

The process includes the following sections:

4.5.2.1 Initial Web interface

In loading the initial models;

- The client sends a request to the server for index.html
- index.html calls Cesium.js, which in turn accesses related JavaScript and CSS files. These files are responsible for loading first virtual globe on the web browser, within the HTML5 canvas element.
- Consequently, the 3DityDbLoader.js JavaScript file is loaded into the application. This code will load the initial static elements of the Dam surrounding such as the roads and other infrastructures. These models have exported CityGML models, and they are rendered using

their Master JSON file using 3DCityDB-Web-Map (Version 1.1.0, Chair of Geoinformatics, Technical University of Munich).

- In addition, the Terrain Model is loaded using CesiumTerrainProvider API.

4.5.2.2 Initial loading of the dynamic geometries

As the next step of the application to load the CZML models, the index.html file accesses the CZMLLoader.js code with the help of Cesium.js.

This code will initiate the load of the dynamic elements including the waterbody surface, seepage level, and the station's water level tubes.

The parser will create a Cesium entity element based on each CZML packet using the id of the CZML element, and moreover, a function dedicates a color property for the surface bodies based on their height value to better represent the changes in values.

This in effect will create the initial water surface visualization, the seepage surface and the water level tubes. Each surface geometry vertex can be picked as an individual object. Thus, the individual surface elements height value can be retrieved on mouse click.

4.5.2.3 Retrieving the WPS and SOS data

Integration of the dynamic data to our model starts from obtaining the data from the server. In this section, the amount of necessary communication between the client and server is defined. Below the different practice in the client- server relation when it comes to requesting data from a Rest API are described.

- **Server-side caching**: In this request, a single call to the WPS service fetches all the GeoTIFF raster layers representing the water levels, and these files are stored on the server and Cesium will access the data from the server and update the water surface values.
  This method while requiring one request and being light on the client side, might take very long time to complete and in the case of failing at any stage, the visualization process might interfere. Therefore, this method will have too much of a server side involvement.
- **Client side caching**: In this approach, like the previous method, the client makes only one request to the server to get all the available interpolation rasters, and from here Cesium will process the data until all the surface geometries time spans are updated with the data for all the time frames.
  This method will have little to no burden on the server side and will rely mostly on the client to process the data, However, just like the first method, in this process, the first request’s response time will be extremely lengthy. Also, more complication will exist in processing the data and the implementation on the client side.

- **Gradual client side caching**: Based on this methodology, the client sends many individual requests to the server based on the time stamp it requires. The results are then parsed and used by Cesium to update the height values.
  When using this approach, only the necessary requests are made, and there will not be many unused results on the client side. Nevertheless, this means that the user might have to wait before seeing the data when a new request for an unexacting time stamp is made.

Based on the nature of the WPS and SOS data available in this research, the third methodology was adopted. Since the high initial waiting time for the user is not desired. Moreover, the apparent drawback of using the methods, such as the waiting of the user when accessing new time span, was addressed by enabling a buffer method. The proposed method will constantly monitor the available data as the Cesium clock instance changes and check the availability of the value of future time intervals. This is possible since we know that the sensor data is updated every 14 days for the water level data and every two days for the seepage levels.

4.5.2.4 Updating the dynamic values

The updating of the dynamic data, as the section requiring the most processing, is tailored to produce the fastest update capability possible for the geometries by optimizing the amount of proccing needed on the server and the client side. The process for achieving this aim is as follows;

- The onTimeChange.js code is implemented as the loading of the CZML models is resolved.
- The application uses a JavaScript event handler and listens for a change in the cesium clock property that indicates the change in the time slider value and will initialize the update process.
- Before updating the time interval collection of the geometries, the application must make sure that the geometry group lacks the new timestamp within its time interval collection. This makes sure that an already existing time interval is not loaded twice. Consequently, helping the application to run faster.
- Once the application has specified that the geometry group lacks the current time stamp value, the corresponding OGC service is run to obtain the values. Each of the request types and their on the fly integration into the already visualized models is described in detail in the following.

4.5.2.4.1 Updating values for Water surface and Seepage Surface

With the completion of previous steps, the application is now ready to make the POST request to the WPS service to obtain the interpolated surface raster for the timespan. As already discussed in the available data sources section Since the execute operation is done via HTTP POST, a proper Execute Request Payload should be specified as arrays in the form of inputs and outputs. The timespan received from Cesium will comprise the input timespan attribute of the payload and the desired geoprocessing result layer, which will be the prediction interpolation map in this research, constitutes the output array.

The resulting WPS response from the server will be a link to the prediction map in case of a successful response. This link can now be retrieved using an HTTP GET request. Eventually, this will result in a GeoTIFF file which can be parsed using the GeoTIFFJS library.

The parsed GEOTIFF arising from the request will be an array containing the cell values of the raster image. It is important to note, however, that since the initial surface geometries including the seepage and the water level surfaces are all a generalized version of the starting raster image, the newly received raster maps should also be generalized with the same algorithm.

The algorithm used in preparing the starting surfaces was the Nearest Neighbor resampling method. Hence, the same procedure is followed to create a resampled subset from the GeoTIFF array values, where the dataset is resampled by the factor of 10 to reduce the computation time. Figure 4-10 reveals the procedure in which the algorithm selects the value of the pixel center closest to the x, y location of the center of the generalized raster dataset and assigns that value to the generalized raster's pixel.

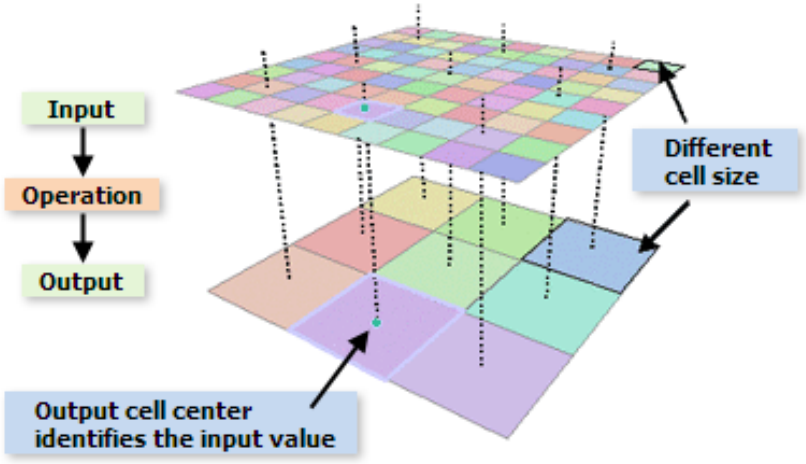

Figure 4-10: The Nearest Neighbor Algorithm used for resampling

Finally, the resulting data subset can be mapped to its corresponding geometry element on the virtual globe. Given the fact that our CZML elements had been rendered with their respective CityGML Ids, representing their cell positions, index of the resulting subset GeoTIFF array's

values can help select the matching Cesium entity object. At this point, a new time interval is added to the Z value of the entity objects node transformation which will include the raster value of the related pixel as illustrated in Figure 4-11 and the time interval instance. As stated in the source data representation section the temporal resolution of the observations are 14 days since the sensors are updated with 14-day intervals. This means that the added time instance to the Cesium objects will have the current time, acquired from Cesium, and 14 days later from that as the start and the end of the interval. However, for the seepage data, this interval is only 2 days. This makes the seepage level data more current when compared to the water level readings.

The seepage level surface and the water level surface while both represent the changes of a variable underground, each characterizes a different element. The first body depicts the height of the water from the lowest point of the piezometer whereas the second surface only depicts the changes in the flowing water levels underground and has values that would hardly go over few meters. Based on this characteristic of these surfaces, the seepage level data is only visualized as color values and not the actual height on the draped geometry over the dam body structure.

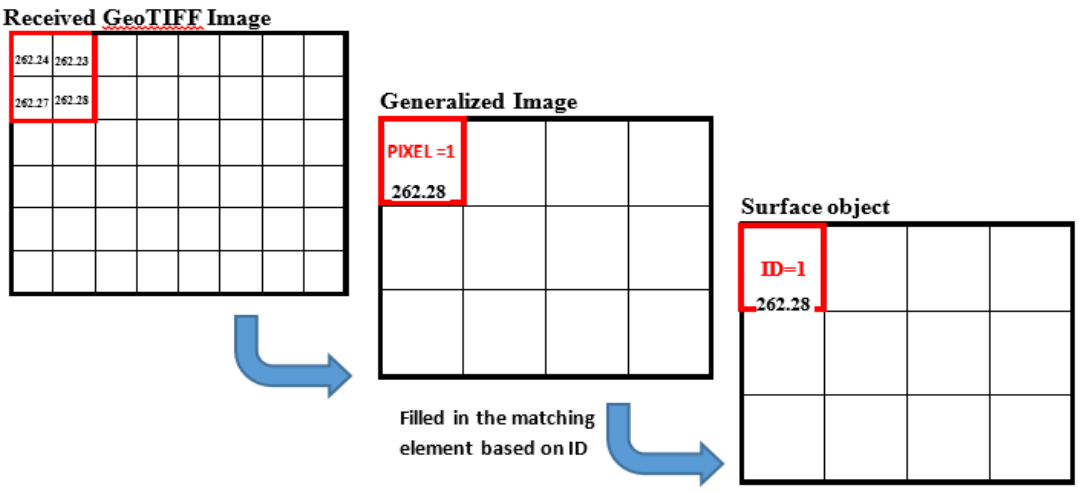

Figure 4-11: Surface members value update mechanism

4.5.2.4.2 Visualization of the Underground Elements in Cesium

As the previous step is finalized, the layer representing the seepage level, since being already placed over the surface of the dam, can easily be updated. However, Since in Cesium the camera cannot yet be placed below the ellipsoid, there are some depth perception issues in representing object below the surface. Therefore, different measures, currently applicable to Cesium (Version 1.33) in overcoming this problem are presented in this research.

Currently, the only provided solution in Cesium for visualizing underground data is the depthTestAgainstTerrain functionality of the globe, which if disabled yields a see-through surface

for objects under the ground to be seen. Moreover, the Cesium Ground-Push[13] plugin developed by Chris Cooper tries to give the user the ability to eliminate the terrain data for the region of interest. Despite the efforts, this repository is not maintained anymore and doesn't support the latest version of Cesium. Notwithstanding, this method could not have helped this research while the terrain data constitutes an important part of this visualization and its existence is essential. Thus, the implemented methods for better visualization of the underground data are as follows:

- Using variation in opacity: in visualizing the water level layer, an opacity attribute is proposed where each element will have transparency level based on their distance to the surface. In other words, every element that is closer to the surface of the globe will be more visible, while geometry elements that are further away from the surface seem more transparent. This transparency value is calculated upon the loading of the CZML surface elements to the virtual globe by subtracting the height value from the surface elevation obtained from the Terrain data.

- Adding depth layer: The issue of depth perception seems to be the most critical issue when it comes to visualization of subsurface data in virtual globes. In Theory, this issue can be resolved by the introduction of a ground surface under the current terrain model. While one solution for achieving, such illustration could be a creation of a secondary globe with a different ellipsoid inside the current earth representation, this method is currently not implementable in Cesium since Cesium's terrainProvider function that creates the Terrain tiles can only mosaic the tiles on the globe surface of the scene. And currently, there can only be one globe per viewer element in Cesium. However, Cesium provides an Ellipsoid geometry object which, placed inside the initial earth globe, could create the reference ground under the surface. Meanwhile, it was observed that this feature object can't be viewed at higher zoom levels and will disappear because only small part of the geometry is in the scene. This leaves us with the only option, known as depth cues previously presented by (Ziolkowska & Reyes, 2016) in which a rectangle geometry layer is created beneath the surface, and an image is used as the material. This method when combined with an outline KML layer at the same underground height, would create the height definition.

- Enabling underground Camera movement: Another lacking feature in Cesium is the unnatural camera movements near underground elements. In an attempt to reduce this abnormal behavior for such features, the tolerance variables such as the 'minimumTrackBallHeight' and the 'minimumCollisionTerrainHeight' of the 'screenSpaceCameraController' of the scene element was lowered to the minimum height of the available features. Hence, the models can now be viewed from under the surface.

[13] https://github.com/NICTA/cesium-groundpush-plugin

This provides a more responsive panning and zooming experience when the camera and an object are at a close distance to each other.

With the completion of these steps and addition of the new time interval, the time interval is visualized on Cesium, and the process is repeated for the next timestamp.

4.5.2.4.3 Updating values of the station's water level tubes

In the effort to update the values of the water level tubes heights, the OGC ‘s Sensor Observation Services provided by 52°North Series REST API is used. The method of retrieving the SOS data and updating the geometry on the globe is essentially the same as the WPS loading mechanism, with variations in request type and the parsing of the data. To fetch a particular timeseries data a GET request against a SOS instance is required. The link for the request is represented by a long URL where it includes the timeseries of interest and a time span. Each CZML element of the water cylinders is stored with the timeseries id value as their id, hence making it easy to obtain and update the elements. The request link is created for each CZML element of the water level polylines, consisting of their id as the timeseries of interest and the beginning and end time stamp. The starting time is just as in WPS received from the Cesium clock element, and for the end timespan, the already mentioned temporal resolution of 14-days is considered. The ISO8601time string of this period is what constitutes the SOS request links.

Subsequent to sending the get request including the time stamp and the timeseries ids, a JSON array including the time stamp and the timeseries value at the specified period is retrieved. At this point, a new time interval instance will be added to the time interval collections of the Cesium entity polylines position property. The value attribute of the timeseries in the retrieved JSON is added as the Z value in the new time interval instance with the addition of the time span representing the start and the end of the instance. This period will match the time frame sent in the initial HTTP request.

4.5.2.5 Visualization of data on chart

Illustrating the dynamic data on charts requires the completion of data loading in Cesium. As Cesium acquires the water level data from the WPS service, this data is available for the line chart section to be called. In this step, the chartsCreation.js JavaScript code is run to allow for the further processing of the Cesium data within the Highcharts.js (Version 5.0, Highsoft).

As for the gauge widget inside the virtual globe, this element is dynamically updates based on the loaded data. And for the chart data, provided that the user selects a surface member of the water level data or a timeseries this data is sent to the Highcharts.js (Version 5.0, Highsoft) library, used in this research, to update the values on the chart.

The chart values are updated every time Cesium’s time property changes. Therefore, it will immediately include any newly loaded data.

5 Results and Discussion

In this Chapter, the results from the implementation process and the final web application is outlined. Moreover, the discussions concerning each step is also represented.

5.1 Web-based interface

The starting point of this application is the web-interface, the client accesses this page through running the HTML page on the local server. Figure 5-1 and Figure 5-2 illustrates the initially loaded page at different zoom levels.

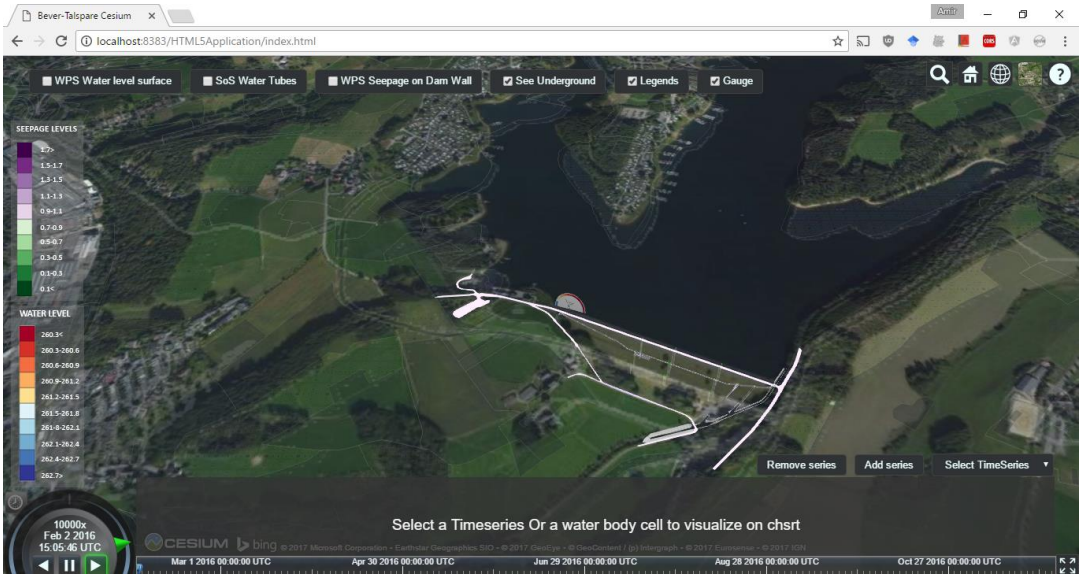

Figure 5-1: Initial Application interface

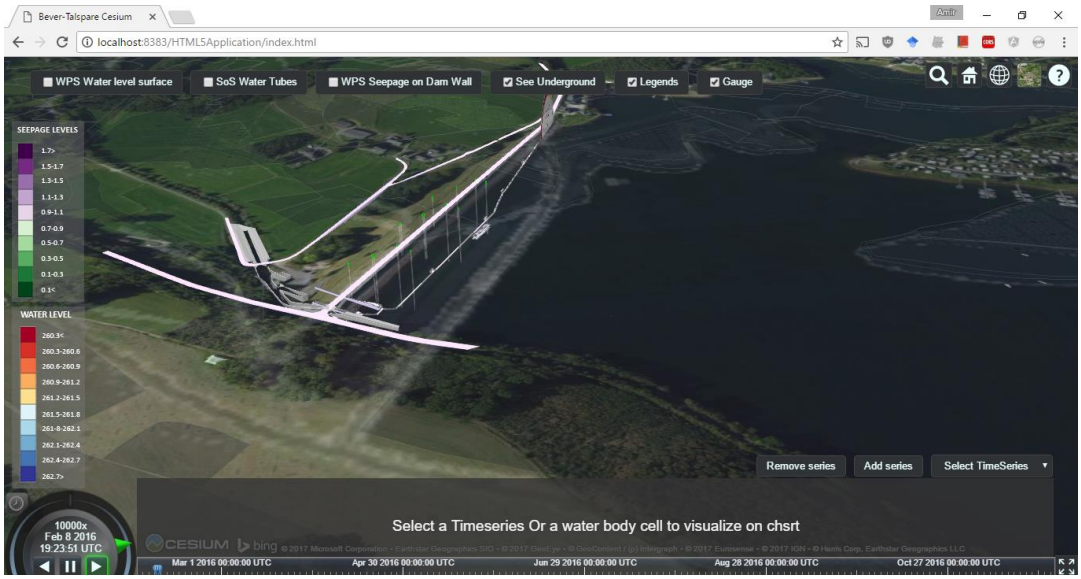

Figure 5-2: Initial Application interface

What is shown in the screenshots, is the web interface representing the Cesium Virtual Globe that utilizes the WebGL API. The models described in this level are the infrastructure layers and the terrain data. The terrain data is loaded based on the Cesium's tiling scheme according to the zoom level from the Apache server. Additionally, the infrastructure layers represent the roads, the sensor stations, the control tunnel and the water exit way of the dam.

The interface has three part of interactive buttons. The group of buttons on the right, as the built-in Cesium elements, include the location search, the ellipsoid to 2D view navigator and the base layer imagery selector. The group of functionalities on the left, however, are applied to better interact with the visualized surface models, the SOS and WPS data. In addition to the mentioned features, there exists a legend map representing the color ramps for the water levels and the seepage data. Finally, the chart data representation panel is lodged on the bottom of the screen. This section will provide the functionalities to interact with the data on charts. Since having all the layers active at the same time might produce some occlusion, the buttons introduced here help the user switch between the layers' visibility. These buttons provide the following functions;

- WPS Water Level surface: switches the water level surface layer on and off. Considering, the fact that this layer is beneath the surface level, having this layer disabled might become handy in some cases.
- SOS Water Tubes: This controls the visibility of the water tubes representing the water level in each station. Given the dependence of the surface levels to these water level readings and the overlap of information the user can select between having this layer or the surface layer based on their application.
- WPS seepage level surface: This segment represents the seepage level on the surface of the dam. As already discussed in the first chapter, the dam's seepage level is a result of various elements one of which is the water levels near the dam. Therefore, this reading can be helpful in its self when shown on the surface of the dam as the space where it has the most impact.
- Show Underground: The feature is included to give the user the option to view the Cesium Globe in its best-intended form. While the initially visualized globe has a disabled depth transparency, this feature can enable the user to have the underground elements hidden, and only see the possible overflow of the water surface if it exists.
- Legend: The legend switch only assists in decluttering the screen in case the information on the left side of the screen obstructs the visualization to an unacceptable level.
- Gauge: Allows the user to remove the gauge element indicating the current mean water level on the dam site.
- Chart data interaction features:
    - Add Series: In combination with the drop-down selection element this button will call the timeseries data requested by the user and depict the data on the chart. The already existing Cesium time bar can be used to navigate the charts data range.
    - Remove Series: Consequently, eliminates the series data from the chart.
    - Moreover, there exists a hidden button that is only activated on the selection of a section of the water level data. In this functionality, when the user chooses a cell from the water body the surface element's height variation through time is visualized on the chart.

5.2 Initial loading of the dynamic geometries

This level is what immediately follows the starting step. Thus, assuming an existence of a reliable internet connection (in the case of running on a remote server), this step and the previous step are visualized synchronously.

The CZML data models are what is loaded in this step. The water level surface (Figure 5-3), the stations' water level shapes (Figure 5-4) and the seepage level surface (Figure 5-5) constitute the loaded elements. These geometries are added to the globe at the same time. The seepage data, However, is not shown on the map to avoid confusion on the first interaction with the data.

The user can now zoom, pan or rotate to visualize the dam from different angles.

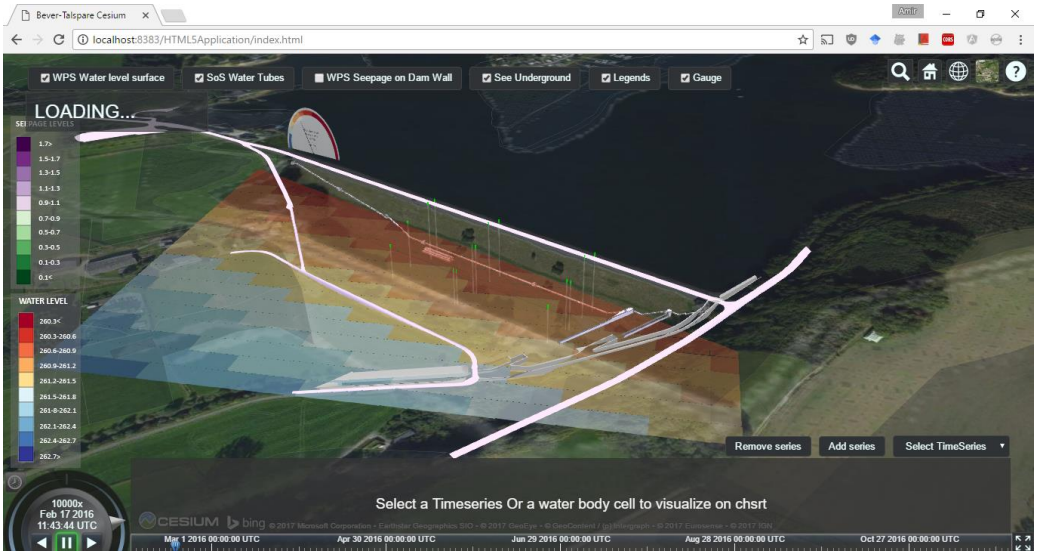

Figure 5-3: Loaded Models with the water surface

The opacity level of the pixels on the water surface shown in Figure 5-3 gives the user a better understanding of its position in relation to the terrain. This is done to overcome the blending mode effect that results from the see-through terrain data.

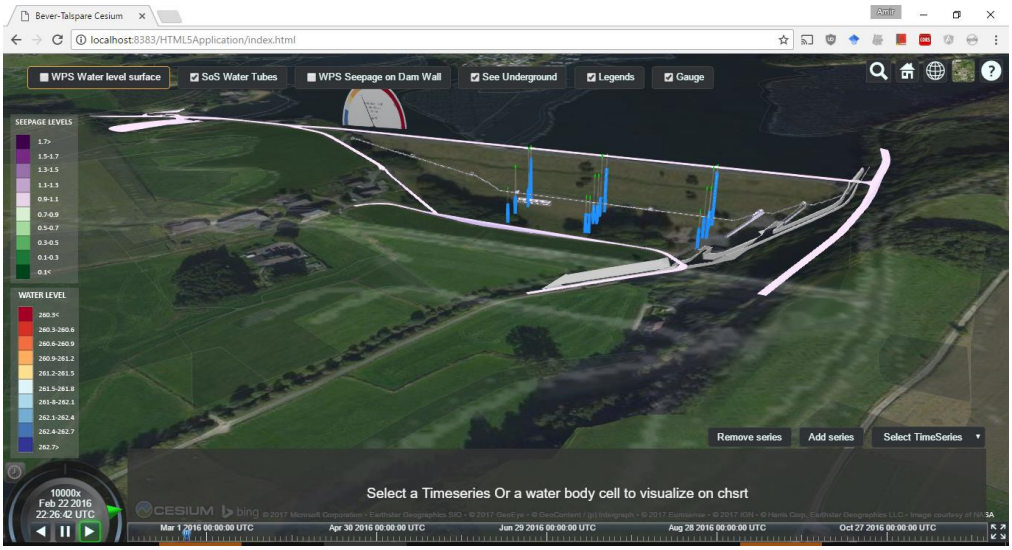

Figure 5-4: The Water level data for stations

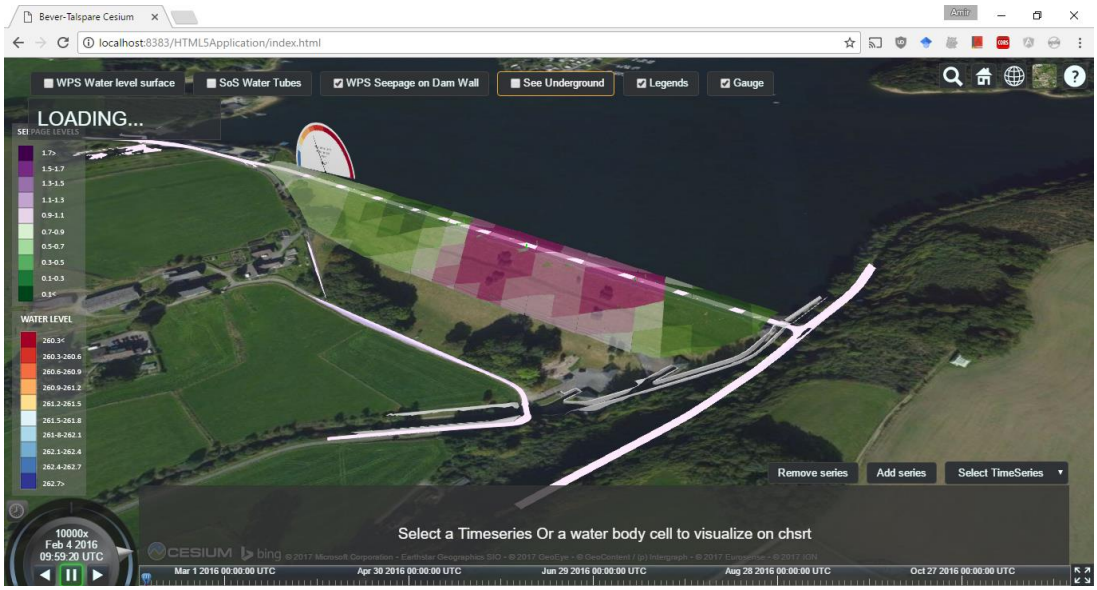

Figure 5-5: Seepage Level Layer

The user can also select a geometry element in these features and view its value. This value denotes the height of the geometry at the current time. This value will later be updated base on the time stamp. This is depicted in Figure 5-6.

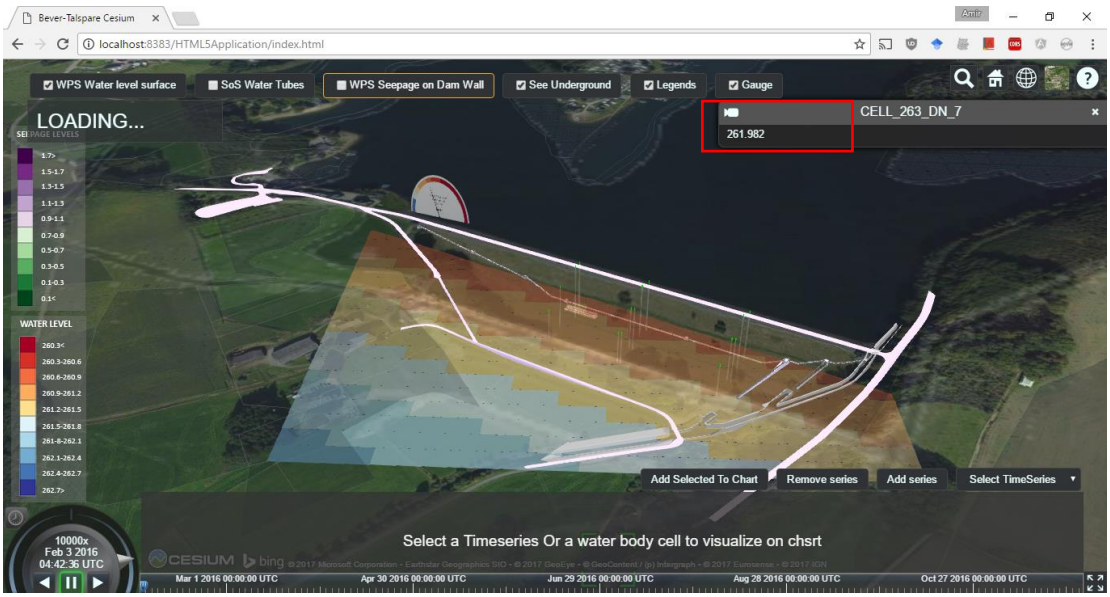

Figure 5-6: Model's behavior on selection

5.3 Supporting temporal variations of height/seepage levels

As soon as the models are loaded in the browser, the listener for a change in the Cesium's Clock instance takes effect. Here the Cesium time slider indicates the change in the Clock variable. And the play and pause button on the slider can start or stop the change and the animation. As the slider is programmed to run after the models' loading is complete, the updating of the timestamp value triggers the JavaScript code to check for the last available timestamp for the object groups availability and load the next timestamp with requests to the WPS before the time slider has reached the time. Therefore, the loading is always done beforehand, and it is stored for access when called by the slider. The image compilation in Figure 5-7 demonstrates the different water surfaces and sensor stations' water level data acquired in different time frames as the slider, shown in the red box, continues to play the animation.

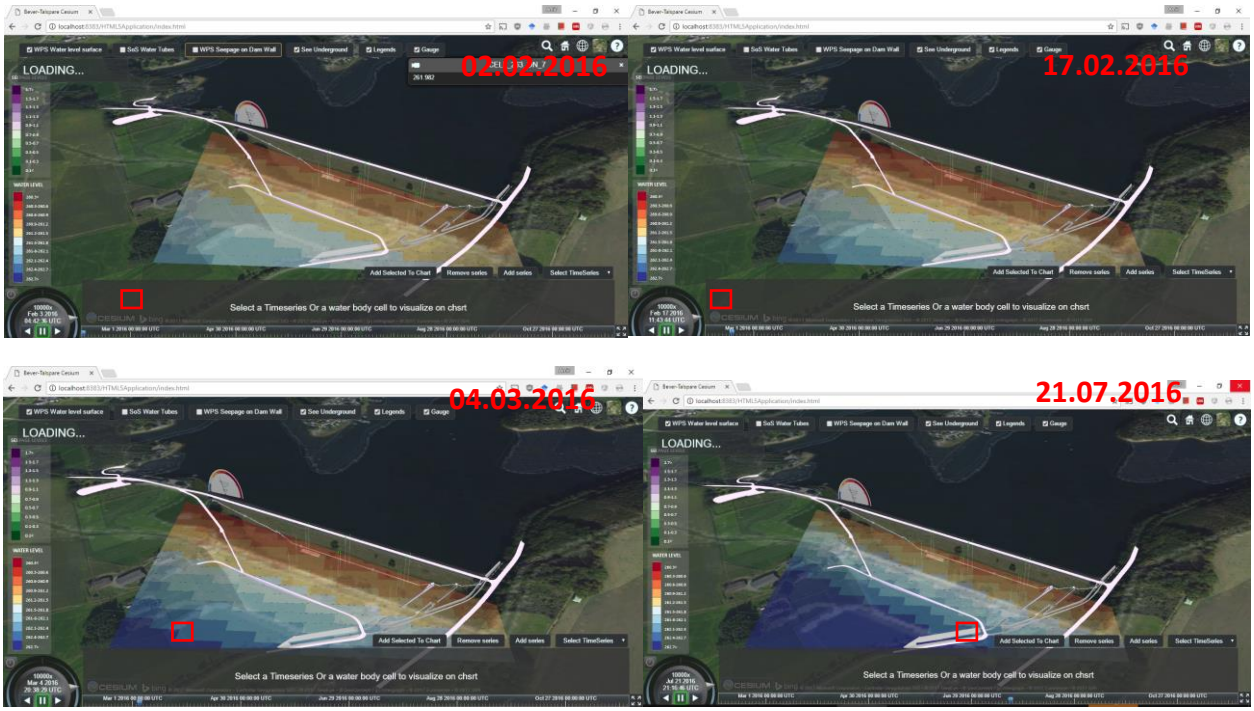

Figure 5-7: Visualization of the dynamic water level data in 4 time stamps

In case the user changes the slider point, as the Figure 5-8 indicates if the geometries lack the timestamp in their height intervals the animation is stopped showing a loading indicator to the user. Consequent to the successful loading of the data, the animation will resume and the forward-looking method described will continue so there can be a seamless animation of height value changes without any unnecessary pauses.

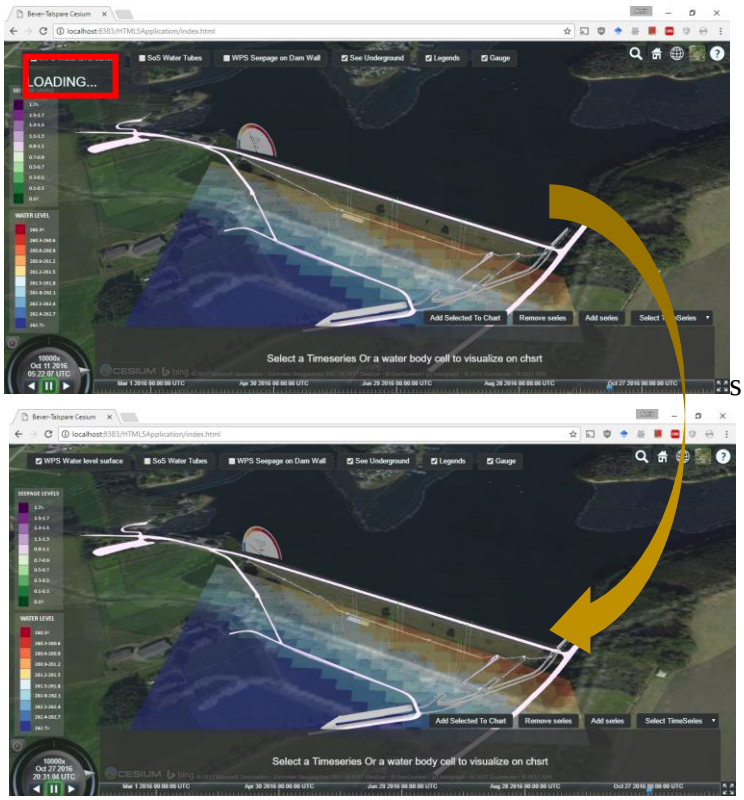

Figure 5-8:The Buffer indicator

Similarly, the seepage surface is constantly updating. The series of images in Figure 5-9 demonstrate the variation in the seepage surface water levels in the course of a week. The update of the water level surface and the seepage level surfaces occur at the same time and do not have any interference despite their different update frequencies.

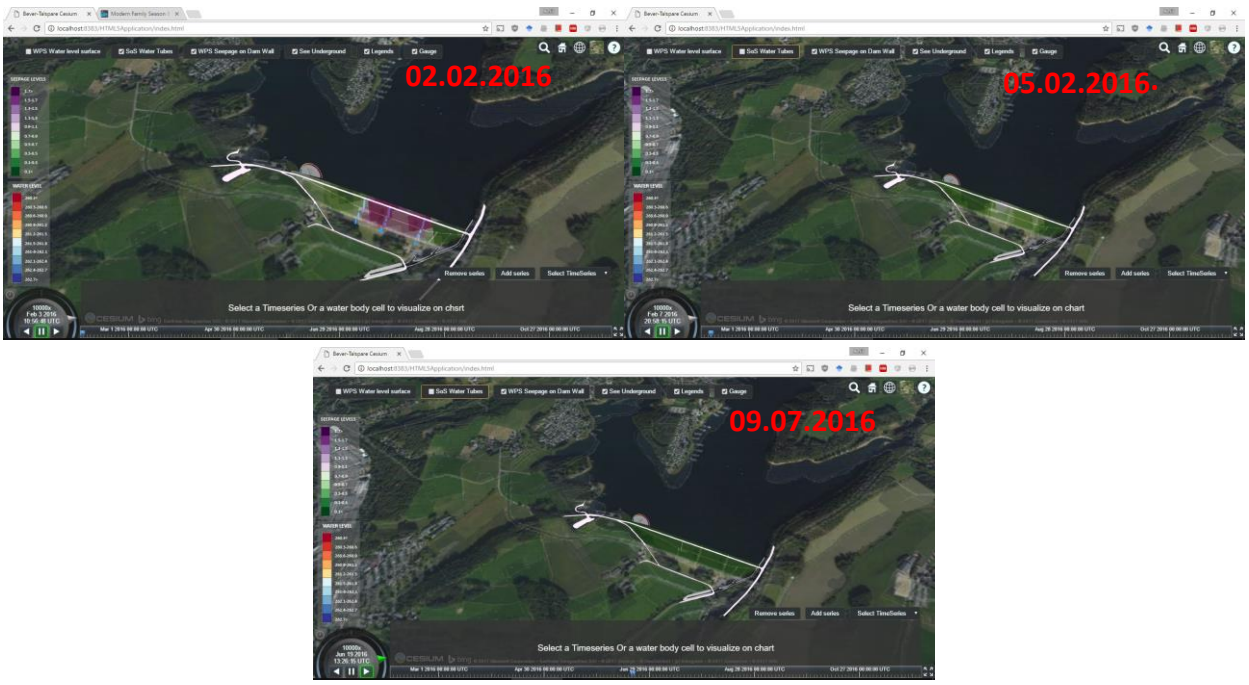

Figure 5-9: Visualization of the dynamic Seepage level data in three different time stamps

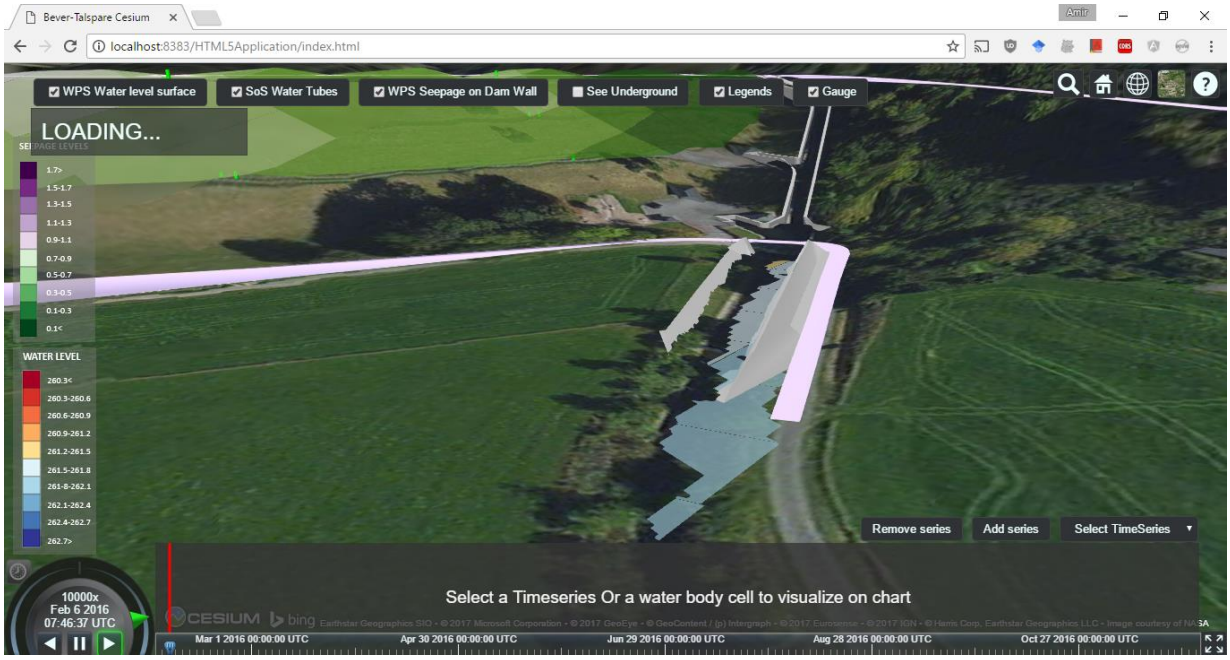

Figure 5-10: Visualization without the underground data

An additional feature in this application is the ability to hide the underground elements. This can be helpful to declutter the environment and only visualize the water level on exposed surfaces or overflow cases. With this feature, the user can still monitor the seepage level changes and have access to the displayed gauge regarding the water levels. Also, as visible in Figure 5-10, the water levels in critically important areas like the section in the exit channel of the dam are still visible in the no-underground mode.

5.4 Supporting temporal variations in charts

Among the primary goals of this research is to implement the 3D visualization as part of a dam monitoring system. Therefore, the existence of analytic tools that help the understanding of this vast amount of information is necessary.

In respect to this goal, numerous data analyzes, and compression features are added to the chart functionality of the application. The data provided by the 3D representation combined with the aggregated data on chart enables the client to have a more detailed perception of the site.

The chart element of the web-app is created using the Highcharts.js Library (Version 5.0.0, Highsoft). This library offers an easy way to integrate interactive charts into web applications. Highchart's provides a framework to create SVG based interactive maps which are perfectly capable of integration with virtual globes such as Cesium. While the library provides various chart types, in this research, the line chart, and the angular gauge have been used to visualize the water

level and the sensor data. The principle behind this part of the functionalities is to provide the user with a mechanism to compare and realize the sensor reading data and the Web Processing services output under one roof. This way the graphs give the sensor reading's numbers shape and form.

The main chart feature of this implementation, shown in Figure 5-11, is located above the Cesium time bar this bar while managing the 3D model, affords the ability to navigate the charts and change the time resolution by zooming in and out. In addition, it can be seen from the figure that Cesium's current time representation pin is extended using the Highchart's Plotline capability to comprise the whole chart. This improves the correlation between the data on the virtual globe and the sensor readings.

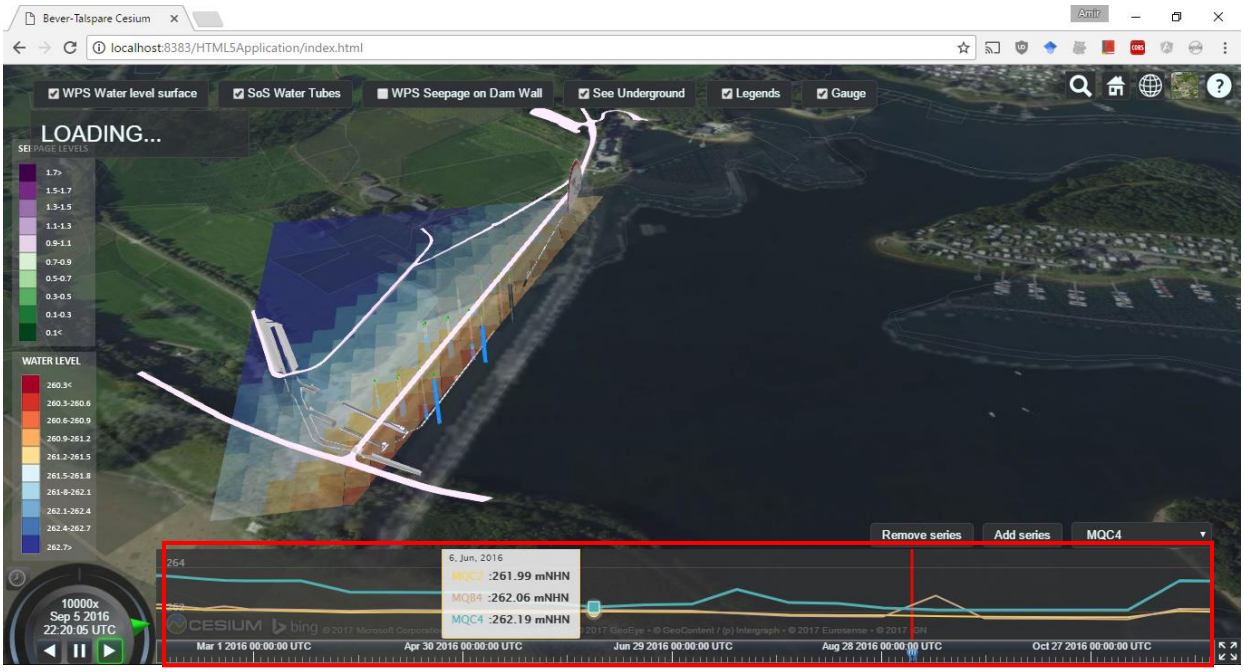

Figure 5-11: Interactive chart section of the application

Various sources can feed the data into the chart for visualization:

- As the most obvious interaction feature the drop-down list, as highlighted in Figure 5-12, provide the list of available Piezometer stations. The data for these stations are obtained from the SOS server upon a request of the user, and the parsed results are added to the graph. To reduce the client's waiting time this data is initially requested in four-month intervals and later updates gradually to fill the complete time span of the Cesium time widget. The sensor readings can be stacked on each other in the chart by adding and removing different stations. Using the Add and Remove buttons on the left side of the list.
- Figure 5-13 in the below section presents another data input source possible in this application. Upon selection of the stations' tube cylinders, the corresponding Cesium Infobox will include the button to add and remove the data from the charts. This method

will follow the overview zoom and filter details-on-demand principle promoted in data visualizations.

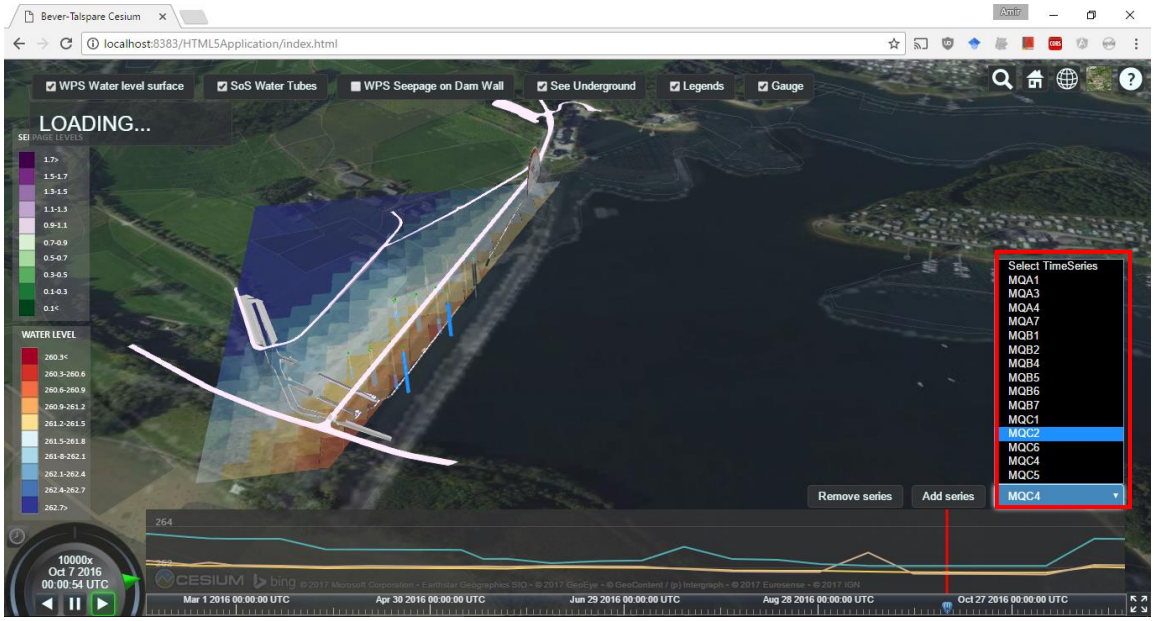

Figure 5-12: Drop down menu for selecting timeseries to be added to the chart

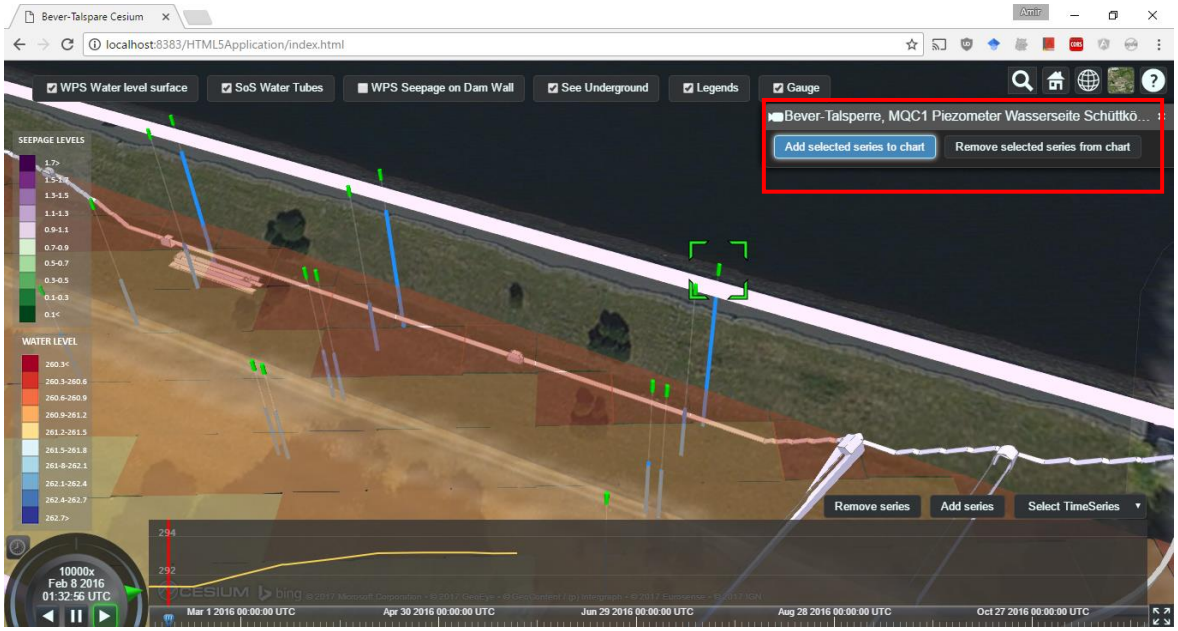

Figure 5-13: Station geometry selection for adding timeseries data to the chart

- An additional data source is the water level interpolation surface, acquired from the WPS services. In this approach, the user can select the individual geometry elements representing the high variations. This selection, when added to the chart, represents the surface element's water level changes during the available time span. To better distinguish

the interpolated surface measurements from the sensor data, this information is presented as a red dashed line. (Figure 5-14)

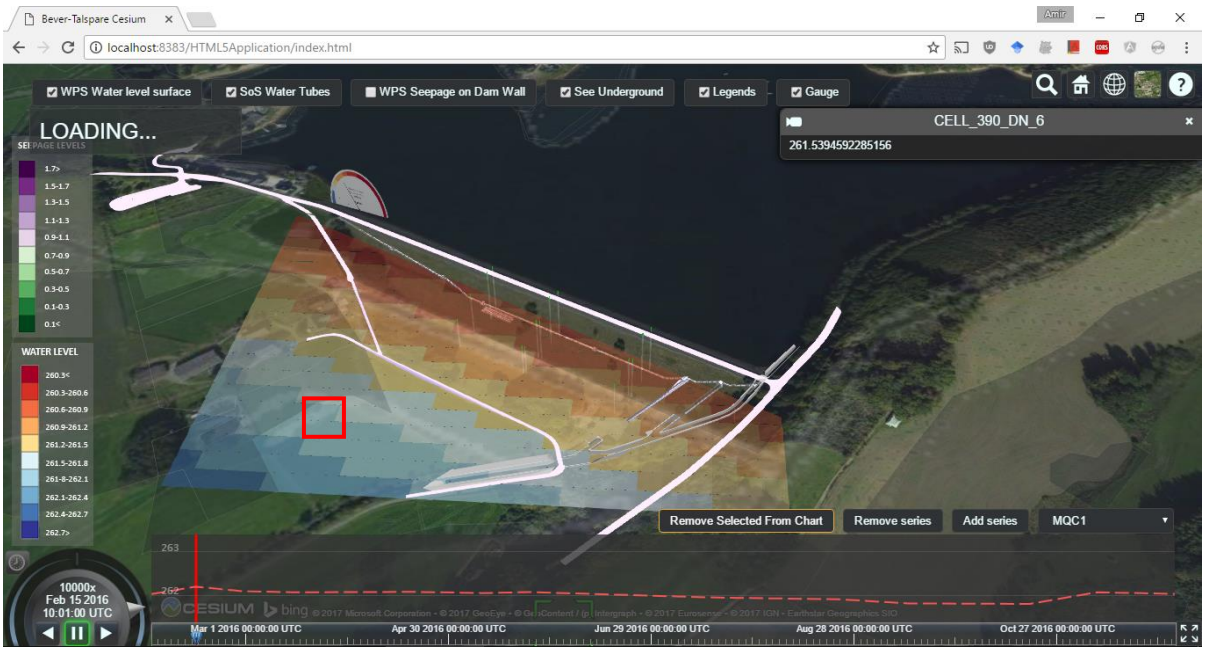

Figure 5-14: Selection of waterbody cell data to be added to the chart

Given the existence of each cell as an independent Cesium entity with time interval attributes, the data interpolation amounts for the cell is represented on the chart based on the corresponding data in the time interval collection of the Cesium entity, which is constantly updating for the selected time span.

Subsequently, the interface includes a mean water level indicator on the body of the dam. This data will show the average water level on the site in a gauge element and is always updating based on the visualized surface data.

While this chart is also produced using the Highchart.js library, it has been configured to be embedded in the Cesium globe. This is done while having the gauge inside the model gives a more realistic feel to the model and better shows the dynamic nature of the visualization. Whereas, the creation of the indicator as a dashboard element would create an overflow of information for the user at first glance.

For this purpose, the SVG element of the produced Angular Gauge chart is used as the material for a Cesium Wall entity placed on the corner of the dam structure. This material is then in turn updated based on the timestamp. Visible in Figure 5-15 , beside the number indicating the water level, the gauge also contains the legends color categories which help the understanding of the overall water trends and variations.

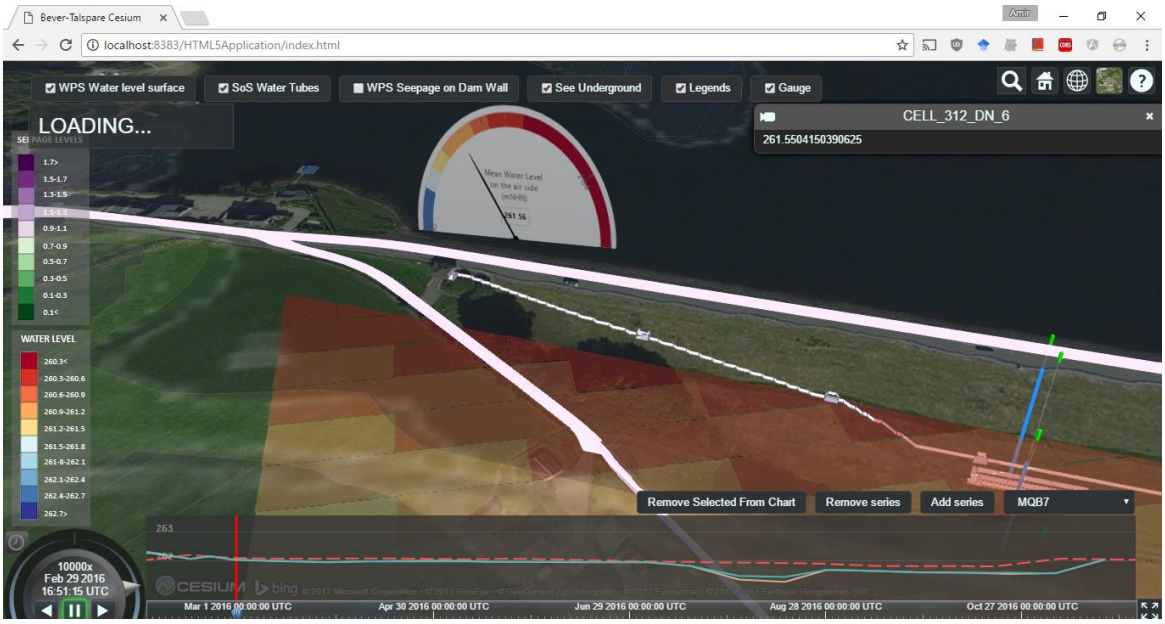

Figure 5-15: The water level indicator gauge

5.5 Comparison with other WebGL frameworks

As described in the available data chapter, the existing 3D visualization element of the TAMIS project developed by the author already supports the visualization of the static station's water level visualization. However, for the purpose of this research, the support for the dynamic water level surfaces in addition to the support for CityGML models is added on top of the current model in order to have a common base for the comparison between the two platforms.

The Three.js product, previously used GeoJSON to draw the infrastructures of the river dam but with the creation of the CityGML objects from the dam facilities, the shifts toward integration of CityGML models seems reasonable. For creating a pipeline that can use the already existing KML NetworkLink files provided by the City3dDB exporter, used by Cesium, the COLLADA loader plugin of Three.js is used. Because, unlike Cesium, Three.js doesn't currently support the integration of glTF models.

Also, since being in the initial stages of development, the dynamic loading of the data is not done through a time slider, but it is done through a next and previous button. Clicking the buttons will send the timestamp with the requests to the server and retrieves the corresponding GeoTIFF. This image is then parsed and visualized as a mesh geometry object with heights on the existing model.

Unlike Cesium where the glTF tiles of the water surface would load in less than 2 seconds, in Three.js each COLLADA feature takes more than 20 seconds to load. Therefore, some part of the checkboard tiles might fail due to the vast waiting time. Additionally, finding the COLLADA models based on their ids and updating their value is an extremely time-consuming process. This is while, unlike Cesium that enables searching for objects within a specific data sources Entity Collection, Three.js only stores all the elements in one single object which evidently increases the

time required for the search loop required for getting an element with their id. These deficiencies have led us to use a mesh surface for creating the water levels. A mesh surface geometry is created from the image pixel. The mesh pixels' height and color are what is updated on each call.

The Figure 5-16 below indicates the interface and the initially visualized model on the TAMIS control center 3D widget. Here the opacity of the base layer is reduced to make the underlying water surface visible. The left side controls provide the options to switch between layers and request a new period.

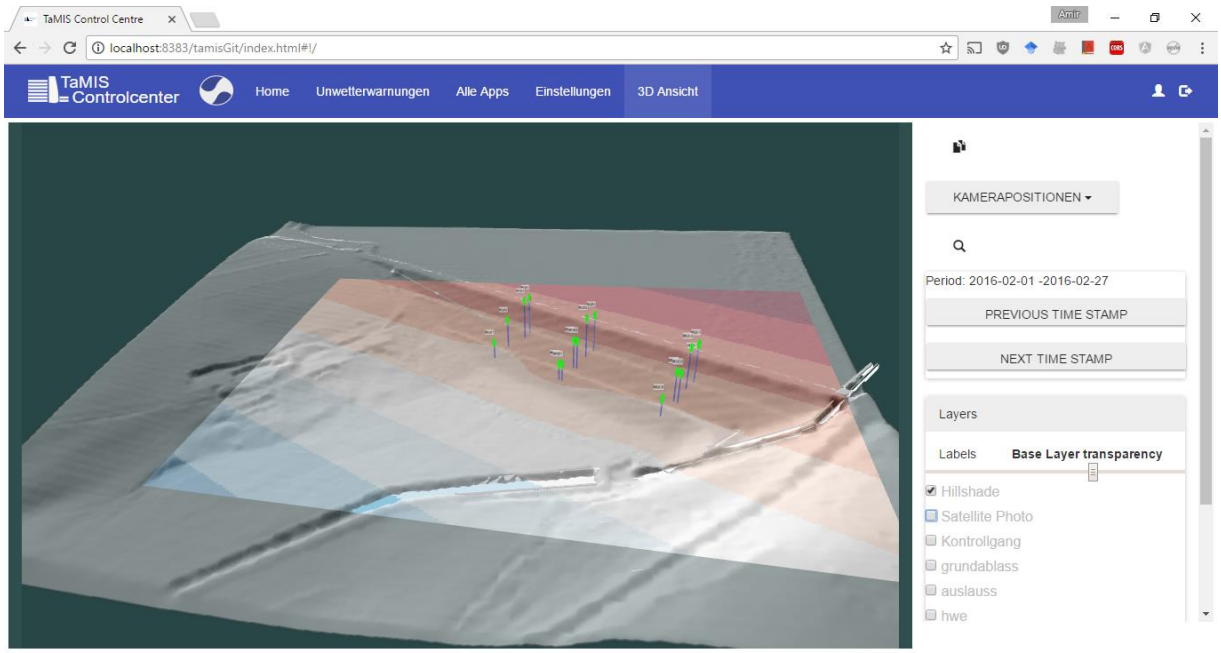

Figure 5-16: Initial Web interface of the extended Three.js based 3D widget

While the time slider in Cesium provides the chance to load the upcoming time events' data in advance, the implemented button navigation here forces the user to wait for the layer to load. Also, the absence of a flexible time attribute for storing dynamic data in Three.js forces us to reload the data for each time instance. This by itself can also contribute to the speed and flexibility of the result.

It should also be mentioned that the mesh nature of the water data makes it hard to include any additional information, like what cesium can provide upon selection of a pixel cell.

On the other hand, however, Three.js seems to provide a better representation concerning the intersection points of water surface and the object. As visible in Figure 5-17 the water level variations inside the exit way of the dam are better visible in the Three.js model.

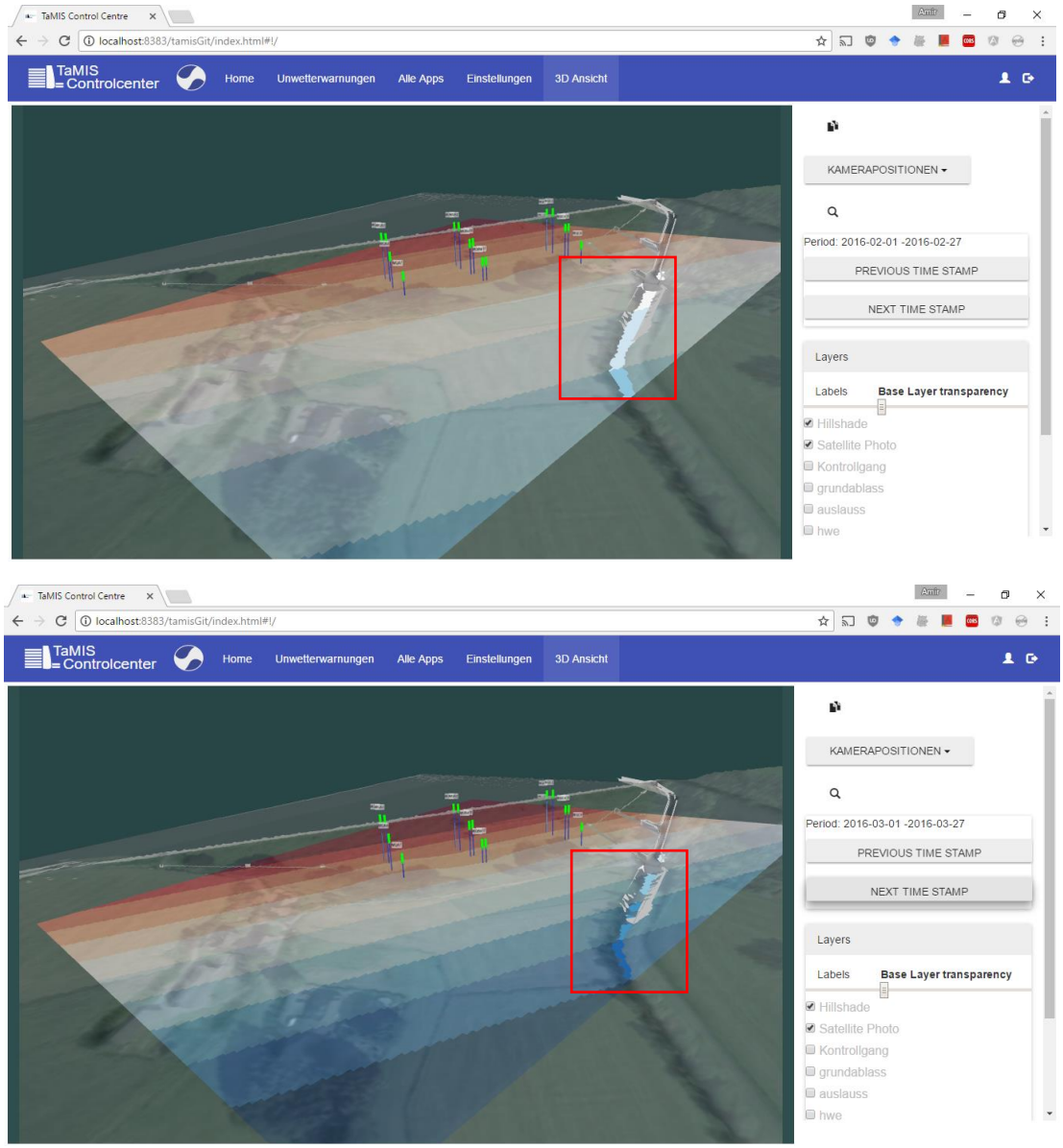

Figure 5-17: Water level on the exit gates during two different time stamps

A further merit of the non-globe 3D visualization of data in Three.js is the ability to represent underground features and elements better. The simple navigation below the terrain model and the double-sidedness of the mesh structures create a more emerging experience when interacting with subsurface data (Figure 5-18).

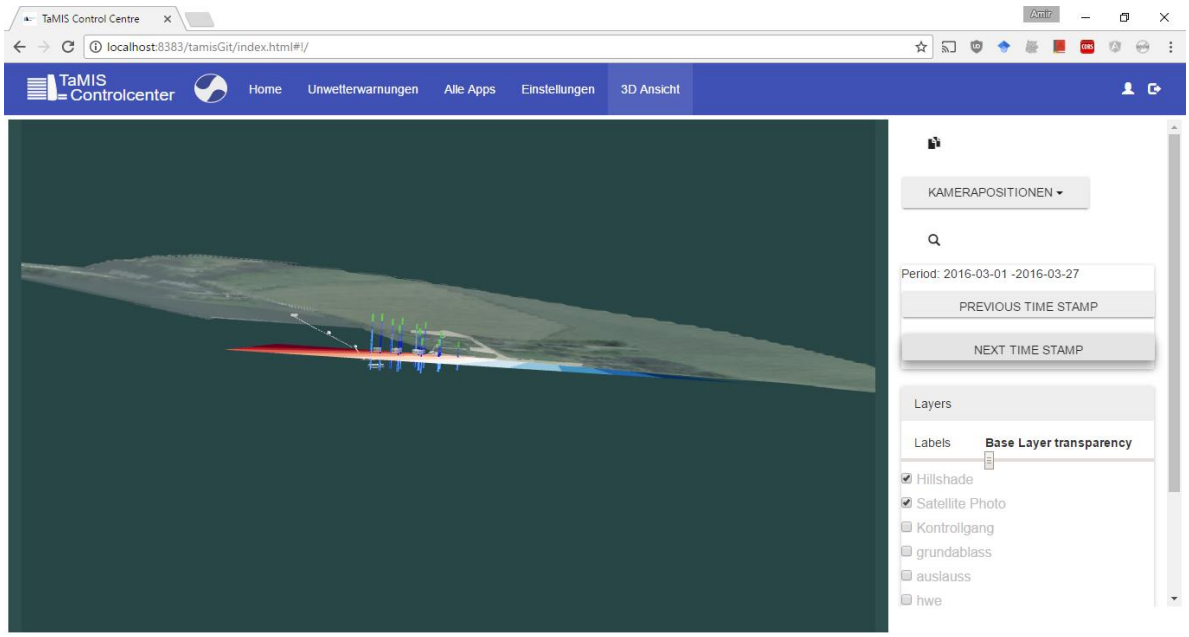

Figure 5-18: Visualization of subsurface models

Table 5-1in the below section gives a comprehensive comparison between these libraries in visualizing SOS and WPS data in the dam monitoring context.

Table 5-1: Comparison of the Three.js and Cesium.js Libraries in creating a dynamic dam monitoring 3D application

	Three.js	**Cesium.js**
Terrain Models	- Mesh structures represent the ground surface - Multiple Mesh objects can be created and overlaid. - Easily generated using Mesh object and elevation map without the need for pre-preparation of data. - not suitable for visualization of big areas	- Multi-Resolution quadtree pyramid of Heightmaps (heightmap-1.0 format). making it suitable for globe visualization - Provides existing terrain data sources - Quadtree pyramid of additional terrain data needs to be created using the Cesium Terrain Builder
Coordination systems	- Supports local coordination only. - Geographic data need to be projected to local coordinate using libraries such as D3JS	- Supports geographic coordinates.
Imagery Layers	- Single images can be loaded as overlay material on mesh objects	- Supports several standard Ready-to-use high-resolution imagery layers - Images are represented as tiles
Representation of Geometries	- Object meshes, lines and polygons are what constitutes all the objects that are stored in the scene based on their name	- Based on the provided data source a collection of entities is made. Under the hood, the entities are made from

	- There can be many objects with the same name.	various primitives representing the geometry. - Entities must have unique ids
Data sources variety	- Supports reading of COLLADA models when used with a plugin - Still, lacks the support for rendering KML data and glTF format	- Various data sources can be added including KML, glTF and Cesium's native CZML files.
Handling OGC WPS and SOS data	- Both frameworks support the integration of OGC sensor observation services and Web processing services.	
Handling temporal variation	- Doesn't have native support for time-dependent data. - Data must be reloaded on each time stamp	- Cesium offers the easy loading of time-dynamic data .it also enables the customization of graphics visualization around it. - Time intervals can be used to represent all the objects property during each time stamp
Speed and response time	- The mesh structures should fully load to be rendered in the scene. This makes the initial loading time to increase. - Implementation of COLLADA models will further delay the loading time while each COLLADA element may take up to 10 times more than there glTF counterparts to load	- The adapted Tilling scheme enables a gradual loading of elements in the globe. - Supported glTF models are easily loaded into model
Support for Subsurface elements	- Since the mesh structures are drawn on a plane scene, navigating around the surfaces is easily done. - Geometry elements can be drawn as double sided features.	- Globe structure hinders out of the box support for underground data and representation of depth. - Camera navigation near the ground and under the surface is a major hurdle.
Feature selection	- Ray casting needs to be done to find selected object	- Selecting a feature will prompt an HTML Iframe object that can include any desired data.

6 Conclusions and Recommendations

6.1 Conclusion

In this research, the visualization of spatiotemporal Web Sensor data using HTML5 and WebGL API for use in a dam monitoring context has been implemented.

In achieving this goal, the 3D models of the dam infrastructures have been combined with the dynamic OGC Sensor Observation Services and Web Processing Services in order to depict dynamic data variation through time. The methodology developed enables the visualization of time varying water body elements in the Cesium virtual globe. It enables the integration of mentioned OGC services with the Cesium model and provides access to interactive data analysis tools such as charts and indicator gauges that, in correlation with the Cesium environment, allow for better observation of height changes over time.

6.2 Answers to research questions

What is the most suitable format to visualize water bodies and the surrounding building structure for the mentioned dam monitoring system in the web browser using Cesium virtual globe?

Since the data elements include static and dynamic objects, different data formats are deployed for each group of data sources. The static nature of infrastructure layers such as the control tunnel, roads, and water facilities, makes them a great candidate to be visualized using their already existing CityGML Models as they follow the CityGML standard issued by the Open Geospatial Consortium. After these models have been exported from the 3DcityDB, they can be easily visualized on Cesium using the 3dcitydb-web-map plugin. The exported format from the database is KML/glTF file while the combination of glTF's ideal loading speed and the reference system provided by KML creates the best results in visualizing the non-dynamic elements of the visualization.

On the other hand, for representing the time dependent waterbody surfaces, the seepage level, and the piezometer water levels, a different file format had to be chosen. This data type needed to create a balance between a server side and client side when updating new timespans. Among the available options, CZML was the only format that adequately fulfilled the requirements for a data type that can have temporal variation and can be easily updated using an index based search for the elements. The CityGML waterbody objects created from a sample WPS response were therefore converted to CZML JSON based features that will have interval elements in their elevation attribute.

What is the most efficient way to retrieve dynamic data from Web Processing Service and Sensor Observation Service?

In this research from the various available methods in retrieving the Rest API results, after analyzing all the existing models, the client caching model is chosen. In this method, the request from the users is only sent when Cesium needs it for visualization. Therefore, solely the necessary time frames are used and unnecessary loading of the data is avoided. Other solutions, however, would require lengthy initial loadings that would affect the user experience. Nevertheless, the downside of this loading method will be the waiting time when requesting the data for a new time windows. To minimize this issue, the proposed solution in the animation mode of Cesium uses a buffering mechanism to load the data for the upcoming time intervals before the intervals have been reached. This way the user can experience a seamless animation with minimum waiting time. This combination allows the use of the Cesium time component for managing the requested data.

What is the most efficient way to visualize such time-dynamic variations using Cesium.js?

As already discussed, the CZML file format chosen for visualization in this research can handle time interval collections that include the time stamp and the values in its properties. Hence, eliminating the need for reloading from the server when accessing the already existing time intervals. To obtain the time intervals value property, a methodology was proposed to parse the SOS and WPS services' responses into time interval collections for each element. These time intervals are called by the Cesium API when the corresponding time value contained in the interval is reached on the time bar. While the parsing from the SOS services JSON response is to some level straight forward, the process for parsing the WPS's GeoTIFF response requires further measures. The applicable methods included the id based update of elements or the category based updating. The latter method is requiring the categorization of height values into a number of groups. This method while being to some level faster does not produce the desired results in the case of this research. As the groups constantly change in each time interval and don't follow a constant pattern. Thus, the best possible method for updating the waterbody objects was to use the cell id of the initial CityGML file to couple the GeoTIFF pixel values with the water surface elements. And since each time stamp constitutes a new interval element in the Cesium entity object, when the user requests the already existing time stamp there will be no need for the reload of the data from the WPS service.

How can the dynamic 3D visualization model of water level within the dam monitoring system help real-time prediction of potential natural hazards and detection of irregularities in the dam structure?

This research in line with the objectives of the TAMIS project makes use of various sensor and processing technologies to monitor river dams. The 3aD visualization of the various data sources provides the following merits in a dam monitoring system:

- The currently collected sensor data each follow their own temporal, spatial and contextual requirements and have different formats. Thus, the combination of these various sensor and processing elements in a 3D environment creates a uniform context for the acquired data as it integrates the geological terrain data with sensor measurements. This provided environment in combination with the charts integration, allows the user to display and compare the time series data values as a graph and a 3D representation. This helps in understanding the relation between the timeseries values and the surface values. It can also provide a good evaluation of the accuracy of the interpolation surface.
- The induvial sensor measurements hinder the comprehensive conclusions or the identification and detection of cascading influences. Therefore, the visualized 3D models enable the representation of the water levels positions in relation to the critical infrastructure and surrounding environment. This model depicts the water level variations' effect on the facilitates overt time.
- The application can also facilitate the simultaneous visualization of various measurement sensors. This feature will provide a bigger picture in case of potential irregularities by letting the monitoring body know if the inconsistency is part of a bigger problem or it is only a minor issue. The incorporation of various sensor data is extremely crucial while in cases where irregularities are induced by natural hazards it is probable that cascading effects such as heavy rain, earthquakes, rainfall will follow. Thus, the multiple visualized layers allow for the appropriate actions in mitigating the contributing factors to be taken.
- Conventionally the WPS water level prediction surfaces could only be retrieved and viewed locally, but this application not only allows the user to visualize the surface levels for a specific time stamp it also gives the user the ability to navigate through the data and visually compare the changes in values through time and determine the water retention curve. This characteristic is crucial in determining unexplained seepage.
- All in all, the application simplifies the understanding of the complex sensor data which will improve reaction time in cases of potential failure.

How can the gauge readings be visualized within a line graph as a navigable feature along the timelines?

To be able to better understand the changes in water level values a line chart is integrated with the web application. The proposed diagram uses the flexible Highchart JavaScript Library (Version 5.0, Highsoft) with a combination of Cesium Clock instance to draw the axis on the fly. In this model, the charts range is fully controlled with the Cesium time bar panel which provides the navigation and change of resolution in the charts. The graph needs to update the axis values on each clock change instance and retrieve the data from the cesium entities in the virtual globe. The time interval collection, as the enabler for such dynamic visualization in Cesium, includes the time and the changing value for each geometry. And this is what the chart retrieves to draw the horizontal and vertical axis at each point in time. The ability to illustrate multiple timeseries data on the chart drastically improves the user's ability to compare the data for a given time span. Moreover, various methods for visualizing a gauge element on the dam visualization was analyzed. The required method had to be representable on the Cesium globe as a Cesium object and have updatability. Therefore, the flexible and updatable SVG charts were selected to be used with Cesium Wall entity element to have fixed positioning of the chart on the globe. The SVG dynamic objects placed as the material for the Cesium entity object was the most effective method to visualize such dynamic objects as a fixed element on the globe.

6.3 Recommendation and future works

Considering the results of this research, it can be said that future studies should be done on the following topics to further enhance the dynamic data visualization on the Cesium virtual globe for a dam monitoring application:

Use of Dynamizers: As mentioned in the design and implementation section, Dynamizers allow the storage of dynamic values alongside static models. This feature is yet to be integrated within 3DcityDB toolkit. When complete it can be used to integrate the WPS links to the timestamps with the static water surface geometry. This way the request for the values is made based on the Dynamizers' dynamic data property. This will reduce the workflow in integrating the WPS data with the static surface elements.

Subsurface data visualization: In contexts similar to dam monitoring systems, underground data constitutes a major part of the available features. However, Cesium currently has very limited functionality in the representation of under surface data. Hence, the development of new capabilities in visualizing the subsurface features can further improve the visualization.

Increased Update frequency: Currently the available data sources provide the data with a temporal resolution of 14 days, therefore, having the data with more temporal accuracy can improve the visualization and enhance the monitoring of the dam structures.

Usage of additional sensors: Based on the information provided overtopping, seepage water flows, and deformation is among the main causing factors of dam failure. In this research, only the Seepage flows and their contributing factors are represented. incorporation of other sensor technologies such as the GPS sensors can establish a better framework for monitoring the dam's facilities with respect to other causes of irregularities. 3D representation of the structure deformation of the dam infrastructure could also help the detection of potential hazards.

7 References

AnalyticalGraphics Inc. (2016). CZML Guide. Retrieved from https://github.com/AnalyticalGraphicsInc/czml-writer/wiki/CZML-Guide

Analytics Graphics Inc. (2016). Cesium - WebGL Virtual Globe and Map Engine. Retrieved from http://cesiumjs.org/

Arsenault, R., Ware, C., Plumlee, M., Martin, S., Whitcomb, L. L., Wiley, D., . . . Bilgili, A. (2004). *A system for visualizing time varying oceanographic 3D data.* Paper presented at the Oceans'04. Mtts/ieee techno-ocean'04 - nov 9-12 2004 Kobe,Japan.

Arthur Na, & Priest, M. (2007). OGC Sensor Observation Service, V 1.0. *Open Geospatial Consortium (OGC)*(OGC Doc No. 06-009r6).

Botts, M. (2007). *OGC implementation specification 07-000: OpenGIS sensor model language (SensorML)*. Retrieved from https://portal.opengeospatial.org/files/?artifact_id=21273

Botts, M., Reichardt, M., & Outreach, O. (2006). *Sensor web Enablement: Overview And High Level Architecture* Retrieved from

Botts, M., Robin, A., Greenwood, J., & Wesloh, D. (2014). OGC® SensorML: Model and XML Encoding Standard. *Technical Standard, 2*(12-000).

Bröring, A., Echterhoff, J., Jirka, S., Simonis, I., & Lemmens, R. (2011). New generation sensor web enablement. *Sensors, 11*(3), 2652-2699.

Bröring, A., Vial, D., & Reitz, T. (2014). *Processing real-time sensor data streams for 3D web visualization.* Paper presented at the Proceedings of the 5th ACM SIGSPATIAL International Workshop on GeoStreaming Nov 4-7 2014 TX, USA.

Chaturvedi, K., & Kolbe, T. H. (2016). *Integating Dynamic data and Sensors with Semantic 3D City Models in the context of Smart Cities.* Paper presented at the ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences, Athens, Greece.

Chaturvedi, K., Yao, Z., & Kolbe, T. H. (2015). *Web-based Exploration of and Interaction with Large and Deeply Structured Semantic 3D City Models using HTML5 and WebGL.* Paper presented at the Wissenschaftlich-Technische Jahrestagung der DGPF und Workshop on Laser Scanning Applications, Cologne.Germany.

Chen, Z., Stuetzle, C. S., Cutler, B. M., Gross, J. A., Franklin, W. R., & Zimmie, T. F. (2011). Analyses, Simulations, and Physical Modeling Validation of Levee and Embankment Erosion *Geo-Frontiers 2011: Advances in Geotechnical Engineering* (pp. 1503-1513). Dallas, Texas, United States: American Society of Civil Engineers.

Christen, M., & Nebiker, S. (2011). *Openwebglobe sdk an open source high-performance virtual globe sdk for open maps.* Paper presented at the 1st European State of the Map Conference, Vienna.

Cox, S. (2011). OGC Observations and Measurements - XML Implementation- V 2.0. *OGC Doc*(OGC Doc No. 10-025r1).

Craglia, M., Kees de Bie, Jackson, D., Pesaresi, M., Remetey-Flpp, G., Wang, C., & Annoni, A. (2012). Digital Earth 2020: towards the vision for the next decade. *International Journal of Digital Earth, 5*(1), 4-21. doi:10.1080/17538947.2011.638500

Dominguez-Acosta, M., Granados-Olivas, A., Hibbs, B., Eastoe, C., & Hawley, J. (2004, November 16–19, 2004). *Computer Based Three-Dimensional Modeling of Hydrogeologic units in the Transboundary Ciudad Juárez-Paso del Norte region.* Paper presented at the Second International Symposium on Transboundary Water Management Proceedings. , Tucson, AZ.

Elvidge, C., & Tuttle, B. (2008). How virtual globes are revolutionizing earth observation data access and integration. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 37*, 137-139.

Fan, G., Zhong, D., Ren, B., Cui, B., Li, X., & Yue, P. (2016). Real-time grouting monitoring and visualization analysis system for dam foundation curtain grouting. *Transactions of Tianjin University, 22*(6), 493-501.

Gröger, G., Kolbe, T., Nagel, C., & Häfele, K. (2012). OGC city geography markup language (CityGML) encoding standard V2. 0. *Open Geospatial Consortium (OGC)*(OGC doc no. 12-019).

Hofer, B. (2015). Uses of online geoprocessing technology in analyses and case studies: a systematic analysis of literature. *International Journal of Digital Earth, 8*(11), 901-917.

Khronos. (2016). glTF—the Runtime Asset Format for WebGL, OpenGL ES, and OpenGL (Patrick Cozzi,Tony Parisi ed.).

Klokan Technologies. (2011). WebGL Earth - open source 3D digital globe written in JavaScript. Retrieved from http://www.webglearth.org/

Mrdoob. (2013). Wiki for Three.js as JavaScript 3D library. Retrieved from https://github.com/mrdoob/three.js/wiki

Mueller, M., & Pross, B. (2015). OGC® WPS 2.0 Interface Standard Corrigendum 1. *Technical Standard Open Geospatial Consortium (OGC), Version: 2.0.1*(OGC 14-065).

Pantea, M. P., Hudson, M. R., Grauch, V., & Minor, S. A. (2011). *Three-Dimensional Geologic Model of the Southeastern Española Basin, Santa Fe County, New Mexico* (2328-0328). Retrieved from New Mexico: U.S:

Parisi, T. (2014). *Programming 3D Applications with HTML5 and WebGL: 3D Animation and Visualization for Web Pages*: " O'Reilly Media, Inc.".

Prieto, I., Izkara, J., & Delgado del Hoyo, F. (2012). Efficient visualization of the geometric information of CityGML: application for the documentation of built heritage. *Computational Science and Its Applications-ICCSA 2012 ,Salvador de Bahia, Brazil*, 529-544.

Research Project TAMIS. (2014, 02 12). http://www.wupperverband.de/internet/web.nsf/id/.

Schilling, A., Bolling, J., & Nagel, C. (2016). *Using glTF for streaming CityGML 3D city models.* Paper presented at the Proceedings of the 21st International Conference on Web3D Technology, Anaheim, CA ,USA.

Schut, P. (2007). OpenGIS Web Processing Service. *OGC Doc, 1*(05-007r7).

Sjödahl, P., Dahlin, T., & Zhou, B. (2006). 2.5 D resistivity modeling of embankment dams to assess influence from geometry and material properties. *Geophysics, 71*(3), G107-G114.

Spies, K.-H., & Heier, C. (2008). *„OGC Sensor Web in der Praxis-Bereitstellung von Sensordaten in Geodateninfrastrukturen und personalisierter Hochwasserwarndienst “.* Paper presented at the Proceedings 20. AGIT-Symposium Salzburg, July 2-4 2008, Salzburg, Austria.

Trevett, N. (2013, June 2013). *3D Transmission Format.* Paper presented at the 2013 Web3d Conference Presentation, San Sebastian, Spain.

Wu, B. P., Cui, B., & Zhong, D. H. (2012). *Web-Based 3D Visualization of Dam Safety Monitoring.* Paper presented at the Advanced Materials Research.

Yang, J., Bao, T.-d., Liang, D.-s., Mi, Y.-f., & Yang, L. (2009, 26 Dec - 28 Dec 2009). *Management information system for dam safety monitoring based on B/S structure.* Paper presented at the 1st International Conference on Information Science and Engineering (ICISE), Nanjing, China.

Zimmerman, P., Jordan, D., & Newell, V. (2016, June 10 , 2016). *Dam Safety Instrumentation Data Acquisition, Monitoring, and Data Visualization Improvements at the Rocky Mountain Hydroelectric Plant.* Paper presented at the American Association of State Dam Safety Officials Annual conference, New London - USA.

Ziolkowska, J. R., & Reyes, R. (2016). Geological and hydrological visualization models for Digital Earth representation. *Computers & Geosciences, 94*, 31-39.