TECHNISCHE UNIVERSITÄT DRESDEN

Faculty of Environmental Sciences Institute for Cartography

Master Thesis

Concept and Implementation of a Contextualized Navigable 3D Landscape Model: The Uch Enmek Example(Altai Republic,Siberia).

Mussab Mohamed Abuelhassan Abdalla
Born on: 7th December 1983 in Khartoum
Matriculation number: 4118733
Matriculation year: 2014

to achieve the academic degree

Master of Science (M.Sc.)

Supervisors
Dr.Nikolas Prechtel
Dr.Sander Münster

Submitted on: 18th September 2017

TECHNISCHE UNIVERSITÄT DRESDEN

Faculty of Environmental Sciences Institute for Cartography

Task for the preparation of a Master Thesis

Name:	**Mussab Mohamed Abuelhassan Abdalla**
Matriculation number:	4118733
Matriculation year:	2014
Title:	**Concept and Implementation of a Contextualized Navigable 3D Landscape Model: The Uch Enmek Example(Altai Republic,Siberia).**

Objectives of work

Scope/Previous Results:Virtual Globes can attract and inform websites visitors on natural and cultural objects and sceneries.Geo-centered information transfer is suitable for majority of sites and artifacts. Virtual Globes have been tested with an involvement of TUD institutes: e.g. the GEPAM project (Weller,2013), and an archaeological excavation site in the Altai Mountains ("Uch enmek", c.f. Schmid 2012, Schubert 2014).Virtual Globes technology should be flexible in terms of the desired geo-data configuration. Research data should be controlled by the authors. Modes of linking geo-objects to different types of meta-information seems evenly important for a successful deployment.

Motivation: For an archaeological conservation site ("Uch Enmek") effort has already been directed into data collection, model development and an initial web-based presentation.The present "Open Web Globe" technology is not developed any further, what calls for a migration into a different web environment. A revision of existing contents, data structures and visualization modes seems sensible.

Objective: Theoretical consideration shall give a comprehensive overview on uses of Virtual Globes in Cultural Heritage or -more specifically- an archaeological context. Web applications embedding Virtual Globes as the underlying technology, standers interoperability, implemented functionality, activities of the user community, performance, maintenance and expected sustainability, etc.

Practical Part: The present data on the "Uch Enmek" conservation site shall be adapted, and be toughened up to work within the favored new Virtual Globe environment. In dealing with a scientific landscape model, preservation of object structures and topological consistency (option of "clickability" for objects for further contextualization) shall be considered. Non photorealistic elements in the model can make sense and are welcome.Linked to the implementation, some presently critical issues shall be clarified in the best possible way:

-Spatial reference systems other than the one used in the Virtual Globe?

-Flexibility in respect to customized digital elevation models and image data?

-Object-by-object import versus import of compact object groups

-Transparent import work flows from 2D(GIS) and 3D and automation potential.

-Handling of surface and subsurface (Kurgan) objects.

-Parameters influencing the performance (total mesh number, depth of subdivision of the mesh into objects, texture detail(resolution),multiple instances of identical textures, etc.)

-Operating ability of the application in different browsers and on smart phones.

The achieved results shall be evaluated. Moreover, an outlook shall be given in respect to desirable amendments and extensions of the model.

Deliverables:

The results have to be submitted as a written document along with a digital version. This document has to be delivers in two copies. All data, relevant for further scientific treatment, has to be stores on a digital media,and to be added to the final submission. The contents of the digital media should be structured in such a way, that an easy continuation of project work will be facilitated. The results should furthermore be presented on an A0 poster; the suitable poster template is available through the institution's homepage.

Supervisors:	Dr.Nikolas Prechtel
	Dr.Sander Münster

Issued on:
Due date for submission:

Statement of authorship

I hereby certify that I have authored this Master Thesis entitled *Concept and Implementation of a Contextualized Navigable 3D Landscape Model: The Uch Enmek Example(Altai Republic,Siberia).* independently and without undue assistance from third parties. No other than the resources and references indicated in this thesis have been used. I have marked both literal and accordingly adopted quotations as such. There were no additional persons involved in the intellectual preparation of the present thesis. I am aware that violations of this declaration may lead to subsequent withdrawal of the degree.

Dresden, 18th September 2017

Mussab Mohamed Abuelhassan Abdalla

Acknowledgment

Abstract

The aim of the thesis can be divided into two main scopes. The first part is to examine and to validate available Virtual Globe environments and to make a decision based on a defined criterion on which environment shall be used to represent available Geo-data set. Most popular environments will be reviewed, the underlying technologies, functionality, performance, sustainability and use community activity will be compared for each environment. Additionally, a survey was conducted on real-world implementation of the use of Virtual Globes in area of cultural heritage and archaeology, from which the design concept of the practical part will be inspired. Secondly, after a decision is made, implementing rendering and visualization techniques provided by the selected web globe environment is carried out in order to convey the model data sources in a web application, the results shall be presented and discussed and further recommendations shall be made at the end of the thesis.

The motivation for the web application development which is built from vector data, photo collections and 3D models. is to carry on the efforts made by the Institute of Cartography, TUD in cooperation with Ghent University for assisting in research and management of the culturally-rich conservation area "Ethno-Nature Park Uch Enmek". The " Open Web Globe " technology, that has been used so far ([46], [44]), is not developed any further. This calls for a transfer of the project into a different technical environment. In this context, a revision of the existing data structures and organization and provision seems sensible.

List of Figures

List of Tables

Contents

1 Introduction

As Virtual Globe software has become extremely popular both inside and outside educational settings [47], lot of research has been carried out to explore potentials and roles of Virtual Globes. Known for their ability to render massive real-world terrain,imagery and vector datasets [17], Virtual Globes additionally have the power of three-dimensional visualizations to geographic information, and with recent development in web 2.0, namely webGL and HTML5 Virtual Globes became an improved delivery tool for the geographical information over the internet.

Virtual Globes has been tested and used as a tool of representation and has been compared in many previous studies in its performance and efficiency with two dimensional maps in all of it's forms(e.g digital,paper). Some of these studies concluded that analytical results depicted in 3D on a globe viewer make the reader perceive the analytical reality more actively and self-referenced [50], Virtual Globes have been found to enrich and stimulate spatial thinking for users [47], they are considered to be the forth generation of web mapping [40]

1.1 Motivation

The Russian Altai Mountains, which approximately coincide with the administrative borders of the Altai Republic within the Russian Federation, is the host of many valuable cultural and archaeological sites such as the burial mounds of "Uch Enmek" which reflects a lot of valuable information about the unique Scythian culture that flourished between 9th and the 1st century BC [9]. This has drawn the attention of many inside and outside the scientific community in the last decades. Archaeologists, anthropologists and geo-environmentalists among other academicians have been cooperatively making substantial effort maintaining and preserving the unique natural environment and cultural variety of the region, The TU Dresden Alti project is one of the initiatives that plays a main role in the geoscientific-methodological contribution to conservation activities [37].

A sequence of conducted researches and projects (Prechtel 2011, Schmid 2011 [44] & Burckhardt 2012 [14], Schubert 2014 [46] and Zimmermann 2015 [57]) resulted in a set of a Geo-database which consist of 2D data-collection, 3D models and an initial web-based presentation.The present "Open Web Globe" technology is not developed any further, this require a migration into a different web environment.

1.2 Objective

The aim of the thesis is to establish a comprehensive criteria which a decision will be made based upon, an investigation on the available web based Virtual Globes to find out which web Globe will be carrying the task. T 3D vitalization for the Archaeological sites of "Uch Enmek" will be made, data sources will be processed in order to be visualized in the new environment.

Assessment on the results and the methodologies will be discussed, and aspects like data integration, performance in different browsers and devices will be detailed, additionally representation potentials will be included in scope of validation as well. The thesis will include a review of the best available practices in the scope of utilizing Virtual Globe technology in Cultural heritage with the focus on the Archaeological application, the review will illustrate technology, data integration, design and interactivity of these practices, the design of the theses application will be inspired by some of the innovative concepts presented in the available projects.

1.3 Critical Issues

The critical issues that should be clarified:

- Spatial reference systems other than the one used in the Virtual Globe?
- Flexibility in respect to customized digital elevation models and image data?
- Object-by-object import versus import of compact object groups.
- Handling of surface and subsurface (Kurgan) objects.
- Parameters influencing the performance (total mesh number, depth of subdivision of the mesh into objects, texture detail(resolution),multiple instances of identical textures, etc.)
- Operating ability of the application in different browsers and on smart phones.

1.4 Structure

The thesis starts with the chapter of introduction which gives a brief outline on the thesis and the motivation and provides an overview the thesis structure.

The second chapter will contain some reviews of some aspects related to the scope of the thesis, it will introduce some fundamental concepts and will give a brief introduction of the technology related to the work flow to familiarize the reader with the technical implementation,it will also include the definition of the criteria of the search of the web globe environment, a list of web globes will be examined under the light of the defined criterion and a decision will be made on which of the proposed environments will be selected to visualize the data models.
Next, a survey on the implemented Virtual Globes/web based applications in cultural heritage will review some of the available projects, prototypes and researches on the topic.

The third chapter will illustrate the design of the web application that is made, it explains the data processing for the migration to the new environment and the steps were made for the integration of the data, the rendering and the implementation of the available functionality provided by the web- Globe platform, the problems that occurred and the solutions that were performed.regarding integration, interaction and performance.

In the last chapter the concluded results will be presented and evaluated and the thesis will be summarized, problem overview will be demonstrated and future work will be proposed.

2 Literature and Standards review

In this chapter the first section will explain some of the scientific and technical components related to the research, and it will include an overview on web 3D graphics related technologies and concepts in order to familiarize them for the reader. In the second section some of available Environments will be introduced with a review on their technologies and features, this section will include a technical classification of the presented Virtual Globes which will influence the decision on which environment will carry on visualization task for the thesis. The third section will demonstrate some of the available implementation of Virtual Globes usage in cultural heritage with the focus on archaeological context, finally this chapter will contain justification and more description of the selected environment.

2.1 Fundamental Concepts

2.1.1 3D web graphics

GLSL

The OpenGL Shading Language (GLSL) is the principle shading language for OpenGL. While, thanks to OpenGL Extensions, there are several shading languages available for use in OpenGL, GLSL is the only one that is a part of the OpenGL core.

GLSL is a C-style language. The language has undergone a number of version changes, and it shares the deprecation model of OpenGL. [54]

OpenGL

OpenGL is the premier environment for developing portable, interactive 2D and 3D graphics applications. Since its introduction in 1992, OpenGL has become the industry's most widely used and supported 2D and 3D graphics application programming interface (API), bringing thousands of applications to a wide variety of computer platforms. OpenGL fosters innovation and speeds application development by incorporating a broad set of rendering, texture mapping, special effects, and other powerful visualization functions. [55]

WebGL

Web Graphics Library (WebGL) is a cross-platform, royalty-free API used to create 3D graphics in a Web browser. WebGL uses the OpenGL shading language, GLSL, and offers the familiarity of the standard OpenGL API. Because it runs in the HTML5 Canvas element, WebGL has full integration with all Document Object Model (DOM) interfaces. WebGL is a DOM API, which means that it can be used from any DOM-compatible language e.g. JavaScript,

Java and Objective C. Because it is based on OpenGL and will be integrated across popular browsers, WebGL offers a number of advantages, among them [56]:

- An API that is based on a familiar and widely accepted 3D graphics standard Cross-browser and cross-platform compatibility
- Tight integration with HTML content, including layered composting, interaction with other HTML elements, and use of the standard HTML event handling mechanisms
- Hardware-accelerated 3D graphics for the browser environment
- A scripting environment that makes it easy to prototype 3D graphics,which means there is no need to compile and link before it is possible to view and debug the rendered graphics .

2.1.2 XML

Extensible Markup Language "XML" is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable, XML describes a class of data objects called XML documents and partially describes the behavior of computer programs which process them. XML is an application profile or restricted form of "SGML", the Standard Generalized Markup Language. [11], a handful of formats used in computer graphics 3D representation are build based on XML schema, such as **COLLADA**,**VRML** and it's successor **X3D**,

COLLADA

COLLADA is a COLLAborative Design Activity that defines an XML-based schema to enable 3D authoring applications to freely exchange digital assets without loss of information, enabling multiple software packages to be combined into extremely powerful tool chains. [6]

COLLADA does not accept binary data inside XML.The COLLADA schema defines the XML elements and attributes that enable COLLADA to represent many features using these XML building blocks. The COLLADA feature set includes [31]:

- Mesh geometry.
- Transform hierarchy (rotation, translation, shear, scale, matrix).
- Effects.
- Shaders (Cg, GLSL, GLES)
- Materials.
- Textures.
- Lights.
- Cameras.
- Skinning.
- Animation.
- Physics (rigid bodies, constraints, rag dolls, collision, volumes).
- Multi-representations.
- User data.

2.1.3 JSON

JavaScript Object Notation (JSON) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language. [18]

JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language. [18]

GeoJSON

GeoJSON is a geospatial data interchange format based on JavaScript Object Notation (JSON). It defines several types of JSON objects and the manner in which they are combined to represent data about geographic features, their properties, and their spatial extents.GeoJSON uses a geographic coordinate reference system, World Geodetic System 1984, and units of decimal degrees. [15]

```
{
  "type": "Feature",
  "crs": { "type": "name", "properties": { "name": "urn:ogc:def:crs:OGC:1.3:CRS84" } },
  "geometry": {
    "type": "Point",
    "coordinates": [125.6, 10.1]
  },
  "properties": {
    "name": "Dinagat Islands"
  }
}
```

Figure 2.1: Example of a simple geoJSON file (www.geojson.org)

GeoJSON supports the following geometry types: "Point", "LineString", "Polygon", "MultiPoint", "MultiLineString", and "MultiPolygon". Geometric objects with additional properties are "Feature" objects. Sets of features are contained by "FeatureCollection" objects.

GLTF

The GL Transmission Format "glTF" is a runtime asset delivery format for GL APIs: WebGL, OpenGL ES, and OpenGL. glTF bridges the gap between 3D content creation tools and modern GL applications by providing an efficient, extensible, interoperable format for the transmission and loading of 3D content. [32]

glTF assets are JSON files, and supporting external data. Specifically, a glTF asset is represented by [32]:

- A JSON-formatted file (.gltf) containing a full scene description i.e. node hierarchy, materials, cameras, as well as descriptor information for meshes, shaders, animations, and other constructs
- Binary files (.bin) containing geometry and animation data, and other buffer-based data (positions, normals, texture coordinates, vertex colors, etc)
- Image files (.jpg, .png, etc.) for textures

- GLSL text files (.glsl) for GLSL shader source code

Assets defined in other formats, such as images and GLSL shader source code, may be stored in external files referenced via Uniform Resource Identifier (URI) or embedded directly into the JSON using data URIs . This will result in a binary glTF.

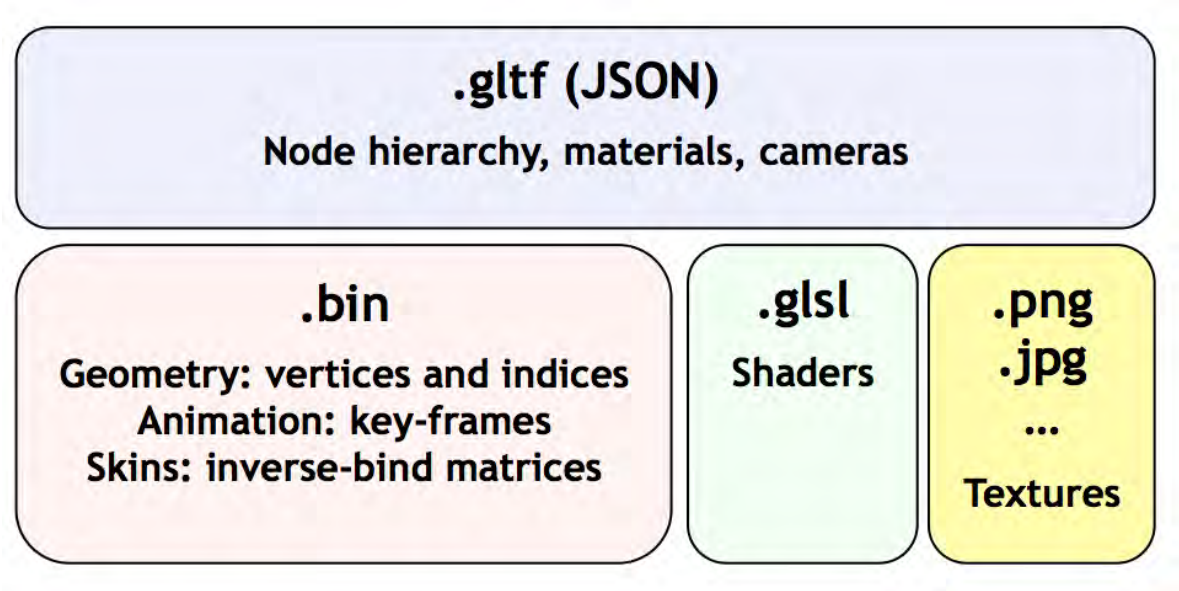

Figure 2.2: glTF design

2.2 Theoretical considerations

2.2.1 Virtual Globes

Virtual Globes are software providing a three dimensional representation of Earth, allowing users to explore Earth in a virtual environment [35]. The most popular Virtual Globe in the market is Google Earth, other known Virtual Globes are Bing Maps Platform from Microsoft, ArcGIs Explorer form ESRI, and NASA World Wind. Currently, the use of WebGL API triggered the emergence of new class of Virtual Globes. Factors like the context capability of Virtual Globes, their ease of use, the growth of provided reliable geospatial data such as satellite imagery, digital elevation models and vector data sets, and the increased internet accessibility have increased the popularity of Virtual Globes and offered potential improvements to the geo-visualization and scientific analysis [12]

Virtual Globes typically provides sets of functionalities that allow users to visualize, integrate, communicate and process geospatial data, provides volumes of freely available images and DEMs, and additionally researchers often use Virtual Globes as the geographical context in which to lay their own data and to effectively display and communicate their research findings [35]. Studies on the use of Virtual Globes in landscape proves that access to, and integration of information are improved by using Virtual Globes and their data [45] and users interest and awareness of the data are raised [3]. The use of connected 2D data displays and 3D views in Virtual Globes enhances the explorative analysis and evaluation of geospatial information, especially for data sets that have a direct relation to landscape [7].

2.2.2 Web Globes

From the various specifications and properties of the available Virtual Globes, the license type was the key criterion for selecting the environment to develop the web application, the free "open source" software are the solutions that will be included as candidates, some real-life application developed using "closed" license Virtual Globes and even some application that uses other HTML5 technologies are included in the survey section because of the adaptable features in these applications such as design principles, functionalities, innovative use of API to meet the specified needs. Some of the of the most popular open source - or free Virtual Globes - in the market are NASA World Wind, Marble, OssimPlanet, WebGL Earth and Cesium.

NASA World Wind

World Wind is a free, open source API for a Virtual Globe. World Wind allows developers to quickly and easily create interactive visualizations of 3D globe, map and geographical information. Organizations across the world use World Wind to monitor weather patterns, visualize cities and terrain, track the movement of planes, vehicles and ships, analyze geospatial data, and educate people about the Earth. World Wind is an SDK (software development kit) that developers can use to build their own applications. World Wind provides a geographic rendering engine for powering a wide range of projects, from satellite tracking systems to flight simulators. [1]

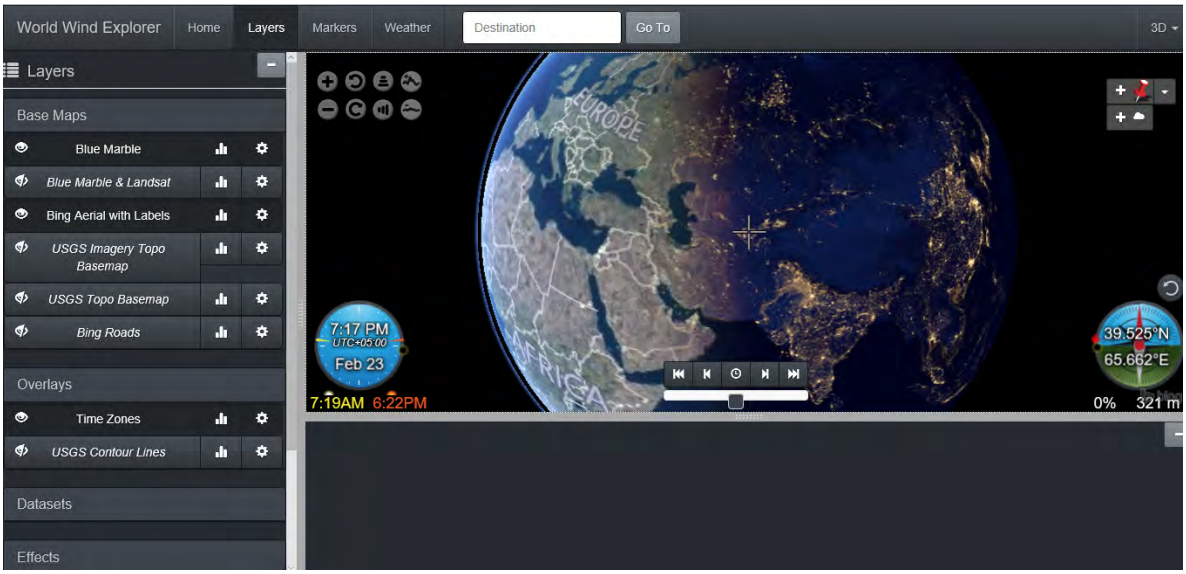

Figure 2.3: World Wind Explorer,an HTML5/JavaScript geo-browser built from the NASA WebWorldWind SDK

Features

Some of the general features of NASA World Wind include:

- The World wind is a SDK available in java for Android, java for desktop application and in java script for HTML5.

- Has many forges and clones (e.g. Geoforge which is also an open source software that has a plug-in mechanism).
- Available data sets are useful for scientific oriented applications, and for extraterrestrial visualization.

NASA World Wind supports Keyhole Markup Language **(KML)** and COLLAborative Design Activity **(Collada)** which is also an **XML** based format.

Marble

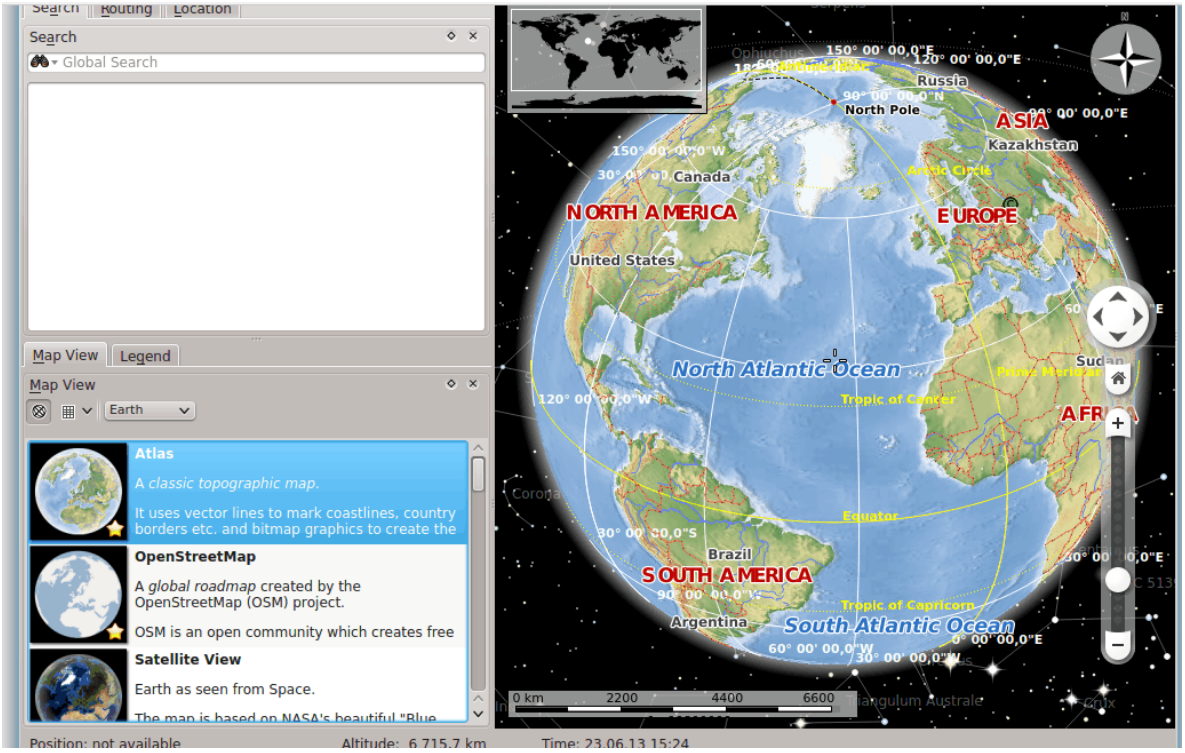

Figure 2.4: Marble Desktop

Marble is a free Virtual Globe and world atlas for desktop and mobile, it is open source with GNU Lesser General Public License (LGPL), it is written in C++ and provides bindings for Qt Quick (QML) and Python. It is part of the KDE (international free software community) education project and allows users to learn more about the Earth and other planets. Data is provided by OpenStreetMap, as well as NASA Blue Marble: Next Generation and others. [30]

features

Some of the features of Marble web environment [30]:

- Functionality includes address searching, positioning and tracking, routing and navigation for vehicles bikes and pedestrians, measurement and bookmarks.
- Links to Wikipedia articles.
- Great educational tool.

OssimPlanet

OssimPlanet is a cross-platform desktop Geospatial viewer, it is built on top of the 3D graphics application programming interface **OpenSceneGraph**, **Libwms** which is a library that is used to query WMS and **OSSIM**.**OSSIM** is a powerful set of geospatial libraries and applications for processing imagery, maps, terrain, and vector data, written in C++ and released under an LGPL license. OssimPlanet provides the ability to view native geospatial formats, Elevation data sets, and OGC Web Mapping Services (WMS) interfaces over the web, it supports navigation to street addresses through the geocoding menu item.

Figure 2.5: OssimPlanet Desktop

Webgl Earth

WebGL Earth [49] is an open-source Virtual Globe made with HTML5 and Canvas WebGL technology. It comes with a JavaScript API adapting the popular LeafletJS API. Similar to Leaflet mapping library and Google Maps API, WebGL Earth API gives the ability to visualize maps, satellite imagery and aerial photography on top of a virtual terrain.

WebGL is a free software available under standard Apache 2.0 license. It is calling internally CesiumJS engine.

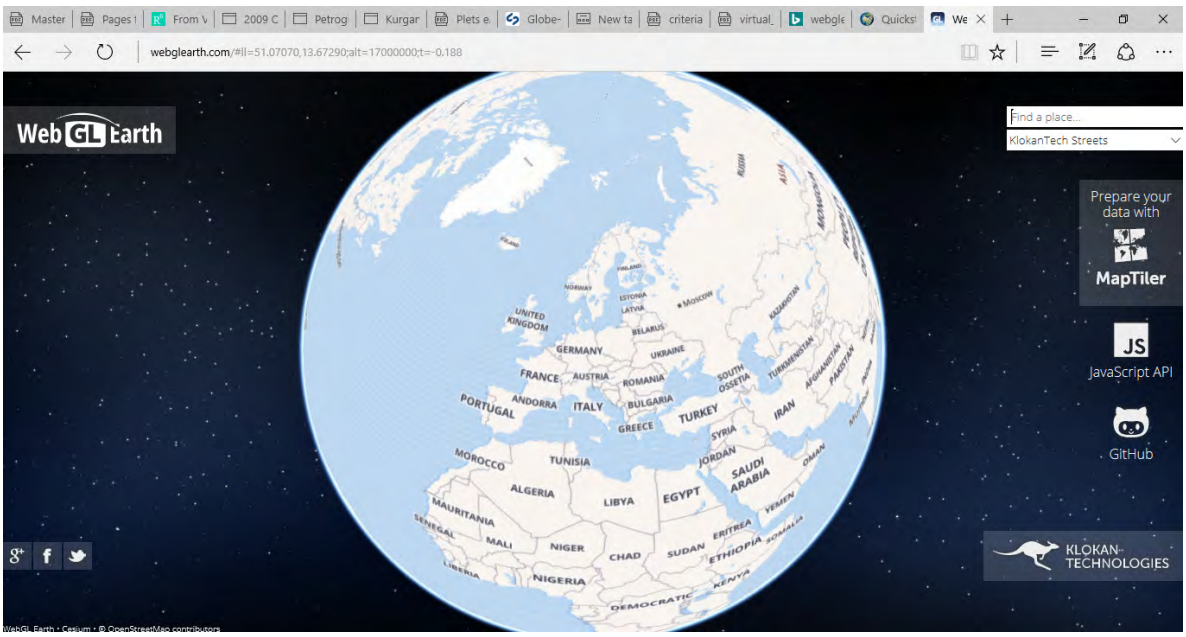

Figure 2.6: webgl Earth on browser

Features:

- Rotation and zoom of the globe,tilt of the camera, and free movement in space.
- Runs without a plugin in modern web browsers.
- Displays existing maps (OpenStreetMap, Bing, ...)
- Supports custom map tiles for the Earth or other planets, Custom geodata, such as GeoTIFF, ECW, MrSID, etc can be easily turned to compatible tiles using Maptiler.this means the ability to use the globe offline.

Cesium

Cesium is an open-source JavaScript library for world-class 3D globes and maps, it built on HTML5 technologies, most important is WebGl. Which gives a lot of potentials like the ability of visualizing 3d shapes in the globe, image layers, terrain models and time dynamic visualization.

Cesium is cross-platform, cross browser, and aimed at dynamic-data visualization.It is open-source under the Apache 2.0 license which means free for commercial and non-commercial use. Unlike platforms like Google Earth, it is not targting end users. It requires programming to use and has a lot of potential for customization and user added content. As it provides some base data to users,such as web map services from mapbox,Esri and OSM maps. it is also classified as a visualization application. [30]

Features:

- Support creating data-driven time-dynamic scenes.
- visualizes terrain data from several sources and allows adding custom terrain data.

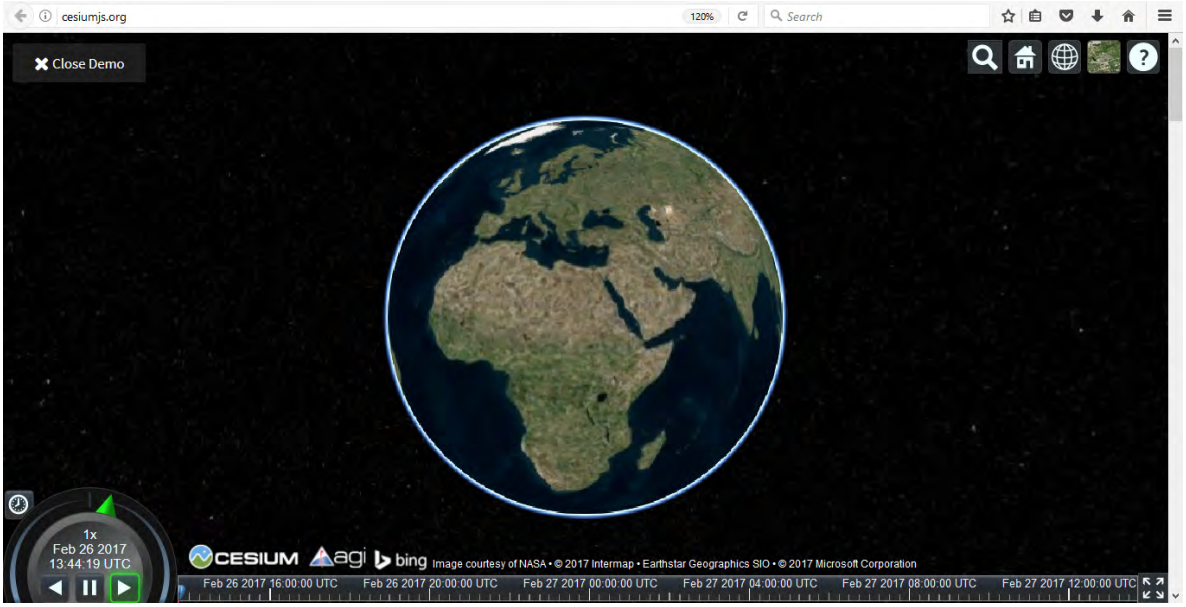

Figure 2.7: Cesium basic application on browser

- Render two dimensional GeoJSON and TopoJSON data,and render 3D models using GLTF with textures and animations.
- Draw spatial data such as points, markers, labels, lines, models, polygons, and volumes.
- Use mouse and touch handlers for rotate, zoom and pan.
- Switch from Virtual Globe (3D) to Columbus view (2.5D) TO map view (2D).

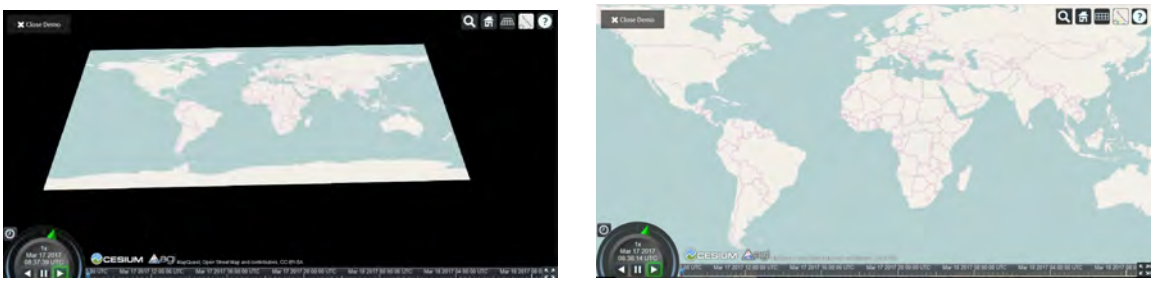

Figure 2.8: Colombus view (2.5D) in Cesium and 2D view

2.2.3 Classification

As mentioned in the previous section the main criterion for selecting the optimum environment is it's availability for users and developers; therefore for only open source Virtual Globes were included. The required platform for the Virtual Globe to run is another important feature to be considered, some of the Virtual Globe can be installed only in a specific operating system while others are cross-platform meaning that they can be installed in all operating systems, the ability to operate in mobile devices is yet another considerable feature some virtual Globes can run in tablets and mobiles but with lesser functionality than a desktop. Virtual Globes can be also classified based on their installing requirements , some are desktop applications, others

are web applications, some of them have some additional requirements and dependencies (e.g JAVA installed in the machine), yet some can be installed as desktop or can be requested as a web application [12].

The **visualization capabilities** of the new environment plays a critical role in shaping the resulted applications. Factors like what data-type can be represented and how flexible the representation is (e.g is the data type native in the Virtual Globe or it needs plug-ins to be visualized) emphasizes what kind of application can be developed and controls its level of innovation. The main criteria concerning visual capabilities are :

- Terrain rendering.
- Imagery layers (WMS, customized tiles).
- Vector data rendering support.
- 3D models rendering support.

Sustainability of the environment is an essential feature to be considered in selecting the environment, the Google Web API for example is officially deprecated as of March 20, 2015. Its last day of operation was March 22, 2017 [26], finding alternatives with sustainable status is one of the aims of this selection process, another factor that is strongly related to the subject is how often the environment is upgraded and updated, some Virtual Globes are updated in monthly bases while others do not have the same progressive development.

Usability: The provided functionalities by Virtual globes determines the options of customization that can be made into them for user's specific needs. Beside the basic tools provided by the platforms like basic visualization, navigation(e.g zoom-in, zoom out),or geocoding, the possibility to add new functionalities is an important feature for the platform that will be exploited to visualize the "Uch Enmek" data set. (Brovelli and Minghini, 2013 [12]) defined two approaches to customize virtual globes by the developers. (1) to utilize the available high-level application programming interface (API) which allows users to interact with the visualized features in the Virtual Globe. This approach does not require advance skills of programming and it is available for all the virtual globes in the market regardless of the license. (2) is to improve and customize basic functions through making changes and plugins to the source code of the platform for specific needs. This approach is applicable only in the open source platforms and it requires a higher level of programming skills. The great advantage of this approach is the ability to create complex functionalities which are not available using solely the external API.

The web application will be developed through the first approach .However, the ease of using APIs relays not only on their level of complexity, the learning curve of the API is shaped also by the provided learning materials, some platforms have proper documentations and tutorials that help developers from wide range of skills to easily grasp the fundamental – or even some advance – features of the environment, the activity of the developers community also is a considered factor to determine the learning curve of the platform.

Assessment

To determine the suitability of the proposed platforms each Virtual Globe was evaluated in the light of the classification factors, some criteria have more weight than others, this is

due to either being more associated to the application required task, or to uncertainty in the criterion measure, for example sustainability has no solid clear measure, it can be estimated based on - if exist - any past depreciation or any information on a future one, instead, the frequency of upgrading and updating releases of the platform can give a better indication of its sustainability, on the other hand ability to visualize 3D models considered to be a knock-out criterion since visualizing 3D objects is the most specific requirement for the final result.

Marble, OssimPlanet and NASA World Wind (desktop) are cross-platform software, they can be installed on any of the three major desktop operating systems (Windows, Mac OS and linux), while Virtual Globes that uses the Webgl technology(WebGl Earth, Cesium and web World Wind) are plug-in free and do not require any installation in the user's machine. They only require a browser that supports webGL regardless of the user's operating system or device, and this is very practical for mobile devices and tablets. Applications built using this technology can run on Android, iOS or Windows mobile. The accessibility of the web-based application is an element that expands the range of the targeted audience to a great level which is a major motivation for the "Uch Enmek" project.

Both the Marble library and OssimPlanet are written in C++ and provide bindings for Qt Quick (QML),QML (Qt Modeling Language) is a user interface markup language,while the rest of the Virtual Globes platforms uses JavaScript for building their web applications. JavaScript is a core technology in world web content production. Traditionally it is used for developing client-side application but recently it started to gain more popularity for server-side Web applications using run-time environments such as Node.js [51].

Learnability which defined as the ease and speed with which the users get familiar with the use of a new product [48], is referring in the case of this thesis to the developer as the product's user. Tutorials, Source code documentation and community blogs of each one of the proposed platforms are the variables to estimate the learnability.

In terms of tutorials web pages and example applications with their source code, Marble and WorldWind has a well build web pages that contains demos and user guide to build simple applications, They are directed to developers from wide range of skills, Marble's tutorials are introduced with 18 languages. Cesium has a similar developer's guide page to WorldWind but it has a sandcastle interactive application which is a documentation generator for the demo applications. WorldWind and Cesium has a very active community blogs where developing teams of the environment alongside with other developers exchanges experiences and troubleshoots problems and challenges faces web developers. WebGL Earth uses leaflet API, which is one of the leading libraries for building interactive web maps, which shallows its learning curve.

Globe	**Terrain data**	**GIS Data**	**3D Data**
NASA WorldWind	Yes	Yes	with plug in
Marble	No	Yes	No
webgl Earth	Yes	Yes	Yes
OssimPlanet	Yes	Yes	No
Cesium	Yes	Yes	Yes

Table 2.1: Visualization Capabilities

The visual capabilities of the five Virtual Globes which are shown in table 2.2.3 can be considered as the critical criteria in the selection process. The environment which are going to be utilized to develop "Uch Enmek" web application must support loading and rendering terrain data,GIS Data, imagery tiles and 3D Data.

The five environments support layering high-resolution imagery and maps from several standard services such as Bing maps and Mapbox, they also support layering any customized imagery. Marble globe does not support rendering terrain data while other web globes support terrain data visualization,Cesium and NASA WorldWind have their own data stored in servers (Cesium has 3 terrain providing servers), it is also possible in Cesium and WorldWind to store customized terrain data in a geo-server and then request it.

Vector data rendering is supported in the five environments, Cesium, WebGL Earth and world wind allows loading and rendering of geojson files. Visualizing meshed 3D models isn't supported in Marble Globe and OssimPlanet, while in WorldWind loading and rendering 3D models stored in COLLADA format is only possible using Add-ons [36], Cesium and Webgl Earth on the other hand supports 3D models rendering, Cesium accepts models stored in GLTF formats. WorldWind has the advantage of its capability to render the globe with any required ellipsoid which makes it consistent to work with any spatial reference, while Cesium only accepts WGS84. However, a version of "Uch Enmek" data set stored in the geo-database is available with WGS84 as it's spatial reference. Since WebGl Earth's Renderer is built on top of Cesium,WebGL Earth shares most of Cesium's visual capabilities, the difference is that WebGL Earth APi is compatible with leaflet, making it much easier and more simple, however, since leaflet is a library for developing 2D maps, the API limitation are far greater than Cesium's API. All of Cesium's time-varying properties are removed from WebGL Earth, visualizing moving objects is not possible the same way as in Cesium. Additionally, the examples on WebGL Earth web page, are using an old version of Cesium (1.14, released on October 2015) which makes it less sustainable.

Considering the described classification factors, visualization capabilities of the the five Virtual Globes and the nature of the thesis application, the decision on the environment that will host the "Uch Enmek" development ranges between Cesium and Webgl Earth since Webgl Earth is using under the hood Cesium for the rendering of the data. The main difference between the two environment is the javascript API that both environments are using, Webgl Earth's API adapts leafletjs API, leaflet mapping library is oriented towards mobile-friendly interactive maps while Cesium library is made for web-based globes and maps for visualizing dynamic data, 3D textured data can be easily loaded into Cesium in GLTF format which is native in Cesium. This is due Cesium's Graphics Stack. Therefore, Cesium was the favored environment for the thesis objective.

Cesium's Graphic Stack

The Graphics technology in Cesium [38] resembles a general graphics engine, but as we move up the layers of abstraction in Cesium, classes become more specific to handling Cesium's main domain which is Virtual Globes. The lowest level of the stack is the Renderer, which is a WebGL abstraction layer that handles WebGL resource management and Draw Command execution. The next layer is the Scene, which is responsible for rendering a frame by requesting commands from higher levels in Cesium, culling them, ordering them, and eventually dispatching them to the Renderer. The top layer in the graphics stack, built on the Renderer and the Scene, is the Primitives, which represent real-world objects that are rendered by creating commands and providing them to the Scene.

Globe

The Globe primitive renders the globe elements which are: terrain, imagery layers, and animated water. Cesium uses a quadtree, in geographic coordinates, for hierarchical level of

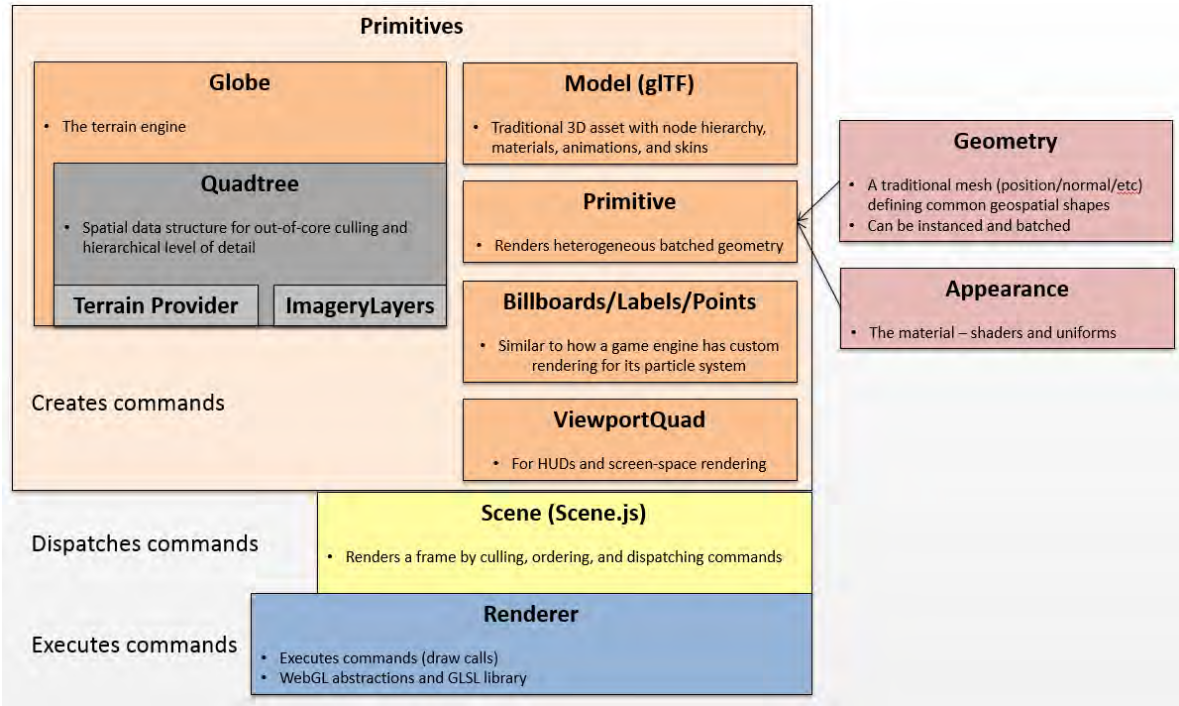

Figure 2.9: The Cesium Graphics Stack [38]

detail. At runtime, imagery tiles from one or more imagery layers, are mapped to each terrain tile. This enables runtime flexibility and support for open standards, which makes it easy to re-project non-native (user's customized) imagery projections. When the Scene calls the Globe's update function, the quadtree is traversed and commands are returned for terrain tiles. Each command references textures for each overlapping imagery tile and uses a procedurally-generated shader based on the number of overlapping imagery tiles and other conditions such as if lighting, post-processing filters, or animated water is enabled.

Figure 2.10: A House mesh from the Tuekta model (Burckhardt 2012) converted into GLtf.

Model

The Model primitive which represents a traditional 3D model like an artist would create in Blender, Maya, etc. is loaded in Cesium as Gltf (see figure 2.10). When the Scene calls a Model's update function, the model traverses its node hierarchy - if needed - and applies any new transforms, animations, or skins. A model may return one command or several hundred

depending on how it was authored, how many materials it has, and how many nodes are targeted for animation.

Primitive

Cesium's generic primitive, plainly named Primitive consists of a large library of geometries for common objects drawn on a globe such as polylines, polygons, and extruded volumes. Most Cesium applications draw the majority of their content using geometries.Cesium geometry tessellation algorithms are designed to conform to the surface of an ellipsoid such as WGS84, as opposed to a flat Cartesian world. For example, polygon triangulation includes an addition subdivision surfaces stage to approximate the curvature of the ellipsoid.

Billboards/Labels/Points

Cesium's billboards provide a lot of flexibility such as the ability to show/hide, set pixel and eye offsets, set screen-space rotation, and scale their size based on the distance. This can lead to "fat vertices" so Cesium uses packing and compression to minimize the number of vertex attributes. Billboards may be completely static, completely dynamic, or somewhere in-between where, for example, the position property for a subset of billboards changes every frame, but the other properties do not. To handle this efficiently, billboards dynamically groups vertex attributes into separate vertex buffers based on their usage so, for example, if only the position changes, the rest of the vertex attributes are not written. Text labels are implemented using billboards, where each character is one billboard. This has some per-character memory overhead, but does not increase the number of commands generated since the same texture-atlas-enabled batching is used. Billboards are also used to draw points, however, starting recent Cesium releases (1.10), the new point primitive is more efficient. It's not as general-purpose as billboards, so it has lower overhead like no index data and no texture atlas.

2.2.4 Virtual Globes in Cultural Heritage

The capabilities and potentials offered by virtual Globes triggered the creation of several Geographic Information System applications dedicated to cultural heritage, this section presents some of the projects, researches and applications which are devoted to utilizing navigable 3D landscapes in Cultural heritage and archaeology, with the focus on applications that were created with Virtual Globes or related technologies.
(Prechtel & Münster) [42] have defined a set of concepts that should be achieved for comprehensive 3D representations in cultural heritage. (1)For the 3D objects a user-friendly reconstruction that provides accurate content is a requirement,(2) standardized visualization technology: the utilization of web 2.0 Technologies in particular HTML5 and WebGL makes it possible for everyone to explore complex 3D models, (3) volunteered geoinformation: the use of user-generated data is worth to be considered despite the difficulty to assess their quality (4) data integration should achieve the most possible interoperable geo-data. The reviewed projects meets most of these concepts -some projects didn't involve any user-generated data-.

This survey serves two purposes, first, it helps to evaluate the performance of Virtual Globes and other web- based technologies in terms of -if applicable- the ability of fulfilling the key concepts mentioned above, second, it helps shaping the design paradigms of the intended "Uch Enmek" application. Adopting some design concepts is the reason for including some studies conducted in technologies other than virtual globes (e.g MAYARCH3D)

The searches were made in the following databases: Google Scholar, Research Gate, Academia and Cesium Demos. In the first three databases the terms "Virtual globe Archaeology" and "Virtual globe heritage" were used to search for scientific papers with a similar topic, as for Cesium Demos the name tag "History" was used to filter the Demos list. Scientific papers and publications were previewed and the studies that included development of applications were selected for further analysis.

Virtual 3D tour Huelva (Spain)

The authors of this project [25] have produced a virtual journey supported by multiple web platforms. A “KMZ” file was created to give the user a guided navigation ability, the data sources are multi-source sets of vector (thematic layers such as Topographic maps and Geological maps) and raster data (satellite imagery and DEM), integrated data layers are available to view and edit as WMS-WFS.
The geospatial data can be viewed in any free or commercial browser (Google Earth, Terra Explorer), compatibility with mobile phones and tablets is considered. Utilizing user's location in phones and tablets, the user can orient information into his or her location. Field photographs, interpretive information, and value assessments were added for each placemark -point of interest (POI)-.

The application objective is to serve as a learning resource designed to improve the teaching and learning process. The authors have utilized Google Earth capabilities to integrate multiple multi-source geospatial data, web-based capabilities were used to combine the visualization of geo-spatial data the descriptive data in forms of diagrams, photographs and information sheets to quantitatively assess the cultural tourism, scientific and educational value of the site.The virtual tour gives a good example of interaction with the temporal dimension. The virtual flight itineraries are provided in compatible video formats to be available in smartphones, tablets and iPads, this alongside with providing the tour in "KMZ" format increases the domain of accessibility.

The project was developed in Google Earth, and there is no 3D objects represented in this application, yet it successfully integrates a variety of data types and the use of the tours demonstrates Google Earth's navigation capability. The accessibility of the application is one of the issues, "KMZ" file format requires an installed version of a compatible software like Google Earth, although recently Google Earth launched a mobile application that allows users to run KMZ files, still the use of WebGl technology appears to be the most efficient method to augment the accessibility factor.

Policrowd

Policrowd 2.0 [20] is an application that allows all smartphones or multiple-sensor mobile devices' users, including cameras and GPS detectors, to become providers of geographical data as well as help change and increase 3D virtual world mapping with personal geospatial data. It enables users to create and personalize projects related to specific topics such as environment, society, scientific research, etc. Personalization is done through a Virtual Globe based interface provided as a desktop application.
Georeferenced historical maps are rendered on the virtual globe and overlapped with the Italian orthophoto with pixel resolution of 0.5 meters and the local digital elevation model with ground resolution of 20x20 meters, users can also exploit a temporal slide-bar to visualize the evolution of the region over time, another slide-bar allows manipulating transparency of the maps-image-tiles.

Buildings are provided as vector layers which are rendered in 2D, but using available information about the buildings like height, year of construction and year of demolition, users can visualize a 3D model from the vector data through extruding heights, and filter the buildings according to different criteria.

NASA World Wind SDK is the platform used as visualization panel for this project, this project shows World Wind's flexibility with geospatial reference systems and different data types, and it emphasizes the advance capabilities of NASA World Wind's API, we can see it's ability to render customized terrain data and to develop spatio-temporal analysis tools. However, the fact that it requires JAVA installation in the user's machine limits the accessibility, 3D objects consist of meshes and textures are absent which also shows restrictions in the potential to represent visually powerful 3D models.

The Paths of Via Regina

One of the basic ideas is that the knowledge of the territory is fundamental for its promotion and protection and that the modern techniques of Web mapping can help in this regard (specifically the multidimensional visualization abilities) [13]. Hence, Web World Wind API is selected as the tool for setting up the Via Regina geoportal [21]. It is a cross-platform application that only requires a web browser that supports Web Graphics Library (WebGL). Web World Wind API provides mostly the same visualization tasks done by the WORLD WIND SDK -mentioned in Policrowd project above-, in addition it offers some base maps, such as Landsat imagery, Bing aerial imagery and OSM. It also has view controls and controls for compass, coordinates and geocoding.

The application aims to integrate virtual globes and VGI systems, using different applications for crowd sourcing - this task is done through SQL and NoSQL with the focus on cross-platform data bases-, and NASA WORLD WIND for the visualization task. Although WORLD WIND SDK provides some advanced visualization features that are not implemented in Web version, yet the Web version is more accessible and usable for the public, it provides a more standardized technology, and it merges the use of SQL and noSQL databases, the project is good example of the usability of virtual globes for crowdsourced data visualization.

The Old Town of Girona

The Old Town of Girona [28] is a prototype created by The "Institut Cartogràfic i Geològic de Catalunya" (ICGC).The application is a 3D Model of the Old Town of Girona, a historical section of Catalonia, Spain. It's a good example of integrating data collected from numerous sources and converting them to a unified format.

The base image layer is an orthophoto image of Catalunya provided from ICGC's WMTS service, tiles are requested through Cesium's API. The application's "terrain provider" renders a local terrain tiles which were generated locally. The 3D model of the town buildings is a textured mesh that was created using oblique photogrammetry camera, the software Acute3D, was used to create 42 COLLADA models with LOD 1. It uses calculation-intensive processes which allows to convert images into a mesh of small triangles that still defines the elevation model in 3D. Each triangle associates with it a corresponding piece of image, the models were converted into GLTF format using an open source tool -COLLADA2GLTF-, additionally vector data that represents the buildings loaded as GeoJSON, then extruded to 3D polygons with a height value that corresponds to the stored value in the "features" properties in the GeoJSON.

This model allows users to view buildings classified by urban use and functionality. The

photographic model is visually very powerful, but it is not ready to integrate data for each building, since it is a continuous model. Instead, the user can view the vector buildings which are individually defined by the volume, they can contain information which is stored as height. In this case, clicking on one of them will get the corresponding planning and cadastral information.
The integrated data are mostly customized, the terrain data, 3D data, vector data and satellite imagery are provided by the developers, and they were integrated into the scene directly by Cesium API after being converted to the corresponding format. The 3D model integration with it's relevant data is not functioning in an optimized way, mainly because the procedure of creating the model resulted in a non-segmented mesh, but generally, this project gives a good idea about Cesium's potentials and about data integration.

Usability Assessment of a Virtual Globe-Based 4D Archaeological GIS

In this paper [19] a prototypical 4D archaeological GIS application was developed based on the virtual globe Cesium. It then demonstrates by means of a usability test with a group of archaeologists, the aim is to uncover usability problems and the general attitude towards the created system. The application is a high-fidelity prototype, but to reduce its cost in this early phase of the development, programming workload was limited by narrowing down the number of features. While the functionalities are limited in number, they were chosen to represent still the intended set of functions. The prototype is aimed to work as a GIS application targeting an audience with variety of expertise levels in interacting with GIS, this means that the developed functionalities are mainly data management and analysis tools. Buttons in the user interface allow for: display of attribute information on click, consulting and updating attribute info in table form, filter on attribute, temporal filtering, distance calculation and 3D Buffer around point.

The authors did not describe the detailed system architecture or the technical aspects of the prototype in this paper, but it appears that these functionalities were implemented through the use of Cesium's API. In a similar way to "The Old Town of Girona" prototype, vector 2D buildings were extruded using the value of an attribute that describes the height. The application uses a plug-in to clip the ground to a specified depth in order to visualize the excavation site, however, the "Ground-Push" plug-in works only with the older releases of Cesium, and it has some issues with the camera movement on the edges of the "pushed" hole. Although the functionalities of the prototype are limited the application gives a good insight of the platform's potential to develop valuable analytical functionalities.

MAYARCH3D The site of Copan(Honduras)

The MayaArch3D project aims to develop an open source research tool that combines 3D-Models with the functions of Geographical Information Systems (GIS) for the documentation and analysis of complex archaeological sites on one internet platform [43] [52].

The open source system (http://www.mayaarch3d.org) has four prototype tools that can manage, store, query, visualize, and analyze 3D archaeological data:

- (1) The **2D Geobrowser** serves as a portal for georeferenced 2D and 3D geometry stored in a PostgreSQL database, it allows users to navigate in an interactive 2D map where they can view geometries and their associated attribute.
- (2) The **3D single object viewer** allows users to interact with a collection of individual 3D models, this viewer is developed in an the open source GIScene.js library, which is

build on top of three.js the WebGL 3D library, the tool is good to view 3D models in a close range scale.

- (3) The **3D scene viewer** visualizes georeferenced 3D landscapes consist of 3D models created from 2D footprints(shapefiles) and a terrain model derived from airborne LIDAR data, the 3D models are clamped to the terrain model and linked to the corresponding data allowing users to click on models to access the archaeological data.
- (4) The **Virtual panoramic tour** tool is virtual tour of the Copan Archaeological Park, it consists of twenty-four georeferenced panoramic images, that includes text, images, sound clips and interactive 3D models.

The project developed with consideration of the challenges associated with 3D archaeological data representation, the design of the project attempts to overcome these challenges through the use of web technologies to assure accessibility and 3D data sustainability, linkage between 3D models and the archaeological databases and preparing data for data reuse.

The approach of visualizing 3D data with different resolution, and scale (landscape and single objects like rooms, relics...etc) is implemented by providing different tools for each category(large scale 2D, 3D landscape, and indoor).

The design of "Uch Enmek" application should adapt "some" the mentioned aspects, the semantically segmented 3D models which are linked to a database [5] seems as a reasonable feature to be included in the application, since the capabilities of Virtual Globes provides native solutions for some of the challenges the separated viewers(tools) approach for example, can be melted into one unified viewer (canvas) where multiple representations can be visualized in one scene at once.

3 Methodology and Technical Implementation

The practical part of the thesis aims at an implementation of the theoretical aspects discussed in the second chapter by exploited Cesium framework to create a virtual 3D landscape. It was found out that the best way to reflect and evaluate all the theoretical issues is to create a web-based application. This chapter will introduce the study area and the available data. It will then emphasize the data processing steps were made to prepare the data for the visualization pipeline. It will continue to demonstrate the design of the web application and explain the workflow of the application system. It will also explain the implementation process in creating the application, for the task of "Uch Enmek" visualization.

3.1 Data Sources and Study Area

The Geo-data set provided for the thesis covers the study area of the Ethno-Natural Park "Uch Enmek" (see figure 3.1), which contains a number of the most famous Scythian burial sites in the Altai Republic. Within the national park, two sites "KARAKOL" and "TUEKTA" contain the most important monuments amongst the thousands of smaller monuments spread over the total area of the park [10]. The data set covers -in detail- those two sites.

The data set which is provided separately for each site can be divided, based on type, into vector data, satellite imagery, digital elevation models, and 3D models stored in the modeling software Cinema 4D library. All contained geodata is based on geographic coordinates using World Geodetic System 1984 (**WGS84**). In this section a detailed overview of the data will be presented.

Figure 3.1: Overview map of the "UCH Enmek" natural Park (Prechtel 2010)

Vector Data

A set of vector data exists in the Altai project's geo-database. the available thematic layers are mostly polygons covering man-made, natural and archaeological features within the site premises. With the exception of the archaeological features which were measured by a team of archaeologists from Ghent University (J. Beaugois 2007 [10]) using GPS devices, the shapefiles were digitized from an IKONOS imagery(Scmid 2012 [44],Burckhardt, 2011 [14]).

The vector layeres are stored in ESRI shapefile format and were classified using remote sensing classification techniques [14]. The shapefiles share similar names to their classes with a different last four attributes (see table 3.1), where (TWGS) refers to Tuekta site and (KWGS) to Karakol site.

File name	Geometry	Description
archeoTWGS-KWGS	polygon	archaeological sites
forestTWGS-KWGS	polygon	forest cover
riverTWGS-KWGS	polygon	river
allroadsTWGS-KWGS	polygon	roads,tracks, streets
stonyTWGS-KWGS	polygon	stony ground
buildingTWGS-KWGS	polygon	building footprints, base elevation, and centre point
frameTWGS-KWGS	ployline	model outline
greenTWGS-KWGS	polygons	different vegetation classes apart from forest
muicipTWGS-KWGS	polygones	the contour of the municipality
fenceTWGS-KWGS	polyline	The village garden outlines

Table 3.1: GIS raw Data

Satellite Imagery

Two generated RGB images (Prechtel 2017) of the scene were provided for the thesis. The satellite data from the IKONOS satellite has been granted by GeoEye Foundation to the Institute of Cartography in 2011. The two pan-sharpened IKONOS color composites are saved as georeferenced "Tiff" raster format (GeoTiff) and they represent the -locally provided - base imagery for the thesis.

3D Model

3D models for each site exist in the modeling software Cinema 4D format. Each model consist of buildings, fences, trees, hydrology features, roads and many other features. There are seven different classes of buildings, see (table 3.1). Each class was assigned with a certain texture or color. Texturing and coloring of the building objects was done based on the building class. The classification was made based on the processing of remote sensing data (Burckhardt, 2011 [14]),and the texturing along with the creation of the "TUEKTA" model is the work of (Schmid 2012) [44], while "KARAKOL" landscape model is the work of (Zimmerman 2015) [57]. The level of detail (LOD) of the "Tuekta" 3D landscape is higher than "Karakol", therefore, 3D prototypes of "Karakol" are extruded polygons defined by their color-tag, while buildings in "Tuekta" model are "Textured" and uses a different prototype. However, The "Karakol" model includes two burial mounds 3D reconstructions (Dießner 2013), the underground chambers are rendered with a hypothetically reconstructed indoor objects such as caskets, tables, and artifacts.

Feature-Class	Designation	ClassCode
Buildings	DWELLING (gable roof)	44110
	DWELLING (hip roof)	44120
	STABLE (gable roof)	44200
	BARN (gable roof)	44220
	CABIN (gable roof)	44330
	SHANTY	44350
	YURT (pitched roof)	44370
	ANNEX (pult roof)	44400
	RUIN (without roof)	44500
bridges		
roads	country road (tarred)	62120
	gravel road (6 m)	62230
	gravel road (3,5 m)	62240
	driveway	62300
	driveway grass	62340
landuse	garden	77710
	built-up area	77720
fence		
trees	deciduous large	71500
	deciduous Small	71520
	conifer large	71600
	conifer large	71620

Table 3.2: 3D data classes

The parts of the 3D models that were involved in the project are the triangle meshed objects. In the "Tuekta" model the buildings objects consist of textured meshes, while in "Karakol" the meshed objects include buildings, roads, burial mounds, bridges, terrain surface, and river.The application will be rendering bridges, burial mounds and houses. Buildings in the virtual models are classified into 7 classes based on the usage of the building, each class includes buildings of a certain functionality (table3.1). The buildings are geo-referenced based on the horizontal coordinates retrieved from the Digital Elevation Model, The Altai-DEM was interpolated from digitized topographic maps. The buildings heights are uniformed for each prototypes.

Terrain Data

Another raster data source is the digital elevation model data for each site. The DEM was generated to inherit the consistency of the 2D data [41], therefore, the roads and river bear a realistic elevation in the resulted relief. Two Geotiff raster DEMs were provided to generate the terrain for "Uch Enmek" application.

3.2 Data Processing

This section will emphasize the preparation steps taken to obtain the necessary data for the visualization.

The vector data sets can be rendered in Cesium after being exported from ESRI ARCGIS software in Geojson format. The Geojson files can be visualized in Cesium as layers attached to the terrain surface or to the ellipsoid, however, the Ground clamping setting can only work

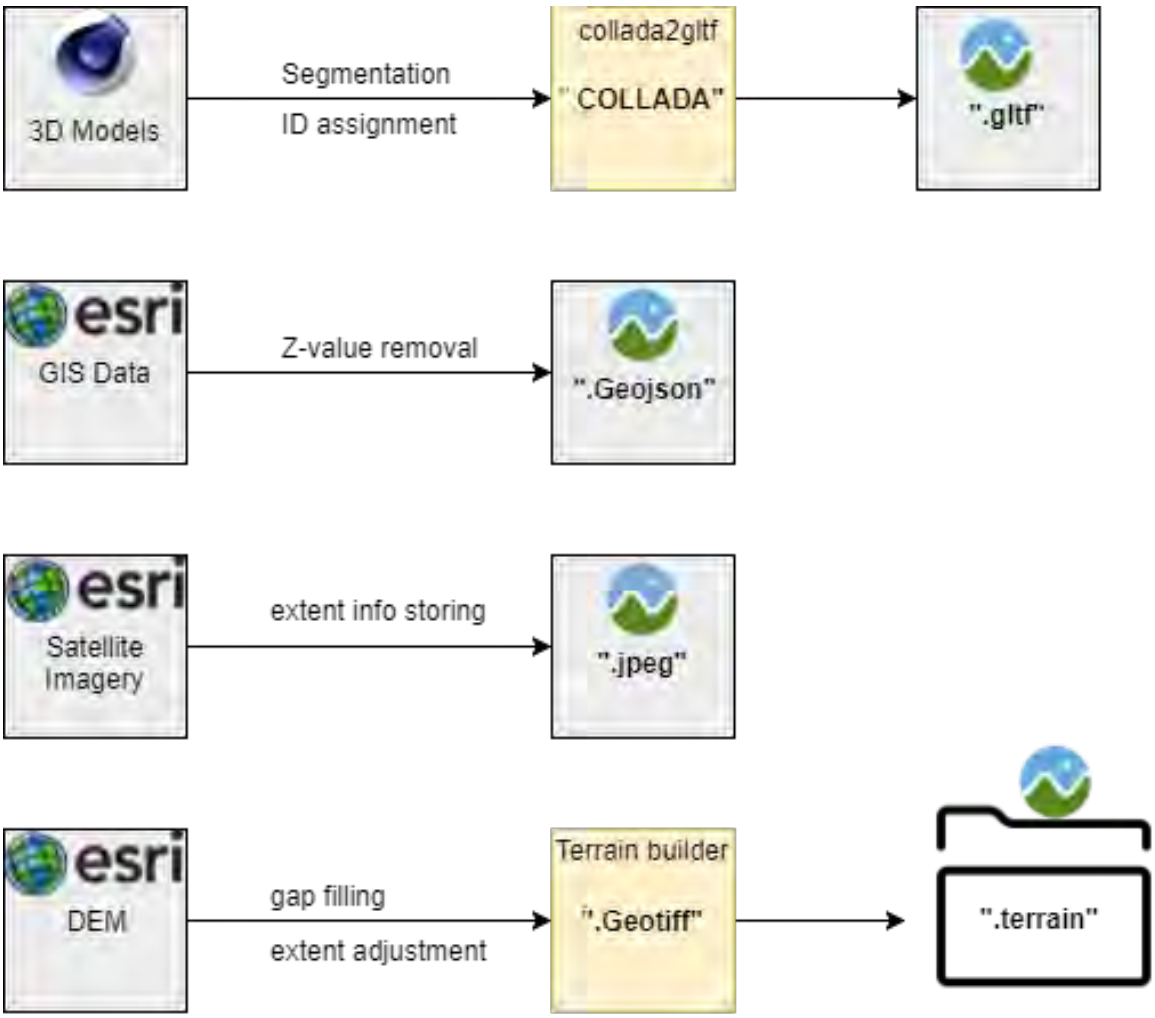

Figure 3.2: The Data integration work-flow into Cesium compatible formats

when the height values of the Geojson data are set to undefined. Therefore, all the Z values were removed from the vector data before exporting is to the compatible Geojson format.

The IKONOS imagery can be served in the application as a single tile. The size of each image is relatively small (less than 7 MB) which makes the cost of loading the entire image at once in the client machine bearable. Therefore, the approach of dividing the images into tiles and serving it as WMS can be avoided in this case, instead, the IKONOS images were exported to JPEG format which means that their storage size were reduced and they are no longer associated with any geographical data, Cesium API has a tiling scheme that allows projection of an image tile within the provided geographical coordinates of the image corners (extent).

3.2.1 Digital Terrain Model

The adopted approach used by Cesium to visualize the terrain is called the Tile-Model which is a recursive subdivision of a portion of an area based on a quad-tree structure [17]. Hence, the level of detail (LOD) management is done based on the zoom level or the camera position [22]. Cesium API supports two formats of elevation datasets to be requested: Height Map and Quantized Mesh 1.0. However, there is no open source software to generate quantized-mesh

format at the moment. Therefore, Height map data was generated to be requested by Cesium API using "The Cesium Terrain Builder", an open source tool developed by University of Southampton, GeoData institute. [58]

Cesium Terrain Builder is a C++ library and an associated command tools that creates terrain tiles, the generated tile sets can be provided by a server to be requested in Cesium. The output of this tool is a set of tiled files in heightmap format ".terrain" stored in different zoom levels.

The raster-based DEM files were edited and processed to meet the input requirements of the tool, first, the empty raster cells or the noDATA values -also called holes or gaps- can not be handled by the Cesium Terrain Builder, hence, noDATA gaps of the raster image were filled with ESRI ArcMap software, using Map Algebra analysis tool each noDATA cell were given the mean value of neighboring cells using a 10x10 mask. In addition, the edges of the raster were clipped to get rid of the no data values in the image edges, finally the images were ready to be used by The Cesium Terrain Builder.

The tool was executed using a virtual machine running on Linux, the tool can generate from any Geospatial Data Abstraction Library (GDAL)-based raster Digital Elevation Model(DEM),the result was a terrain tile-set saved in a directory and ready to be served via a server, the tool calculated the maximum zoom level associated with the native rater resolution and then generated tiles for all zoom levels between that maximum and zoom level 0 where the tile extents overlap the raster extents.

3.2.2 3D Models

The 3D models were prepared to be suitable for Cesium and to fulfill the design requirements of the application using the 3D modeling software CINEMA 4D (see section3.3). The application design requires that every 3D object in the scene(e.g. house, bridge..etc) must be defined as single node, the "Tuekta" 3D model has a hierarchical schema where the model is divided into seven nodes (tags), each one represents a class, and each one of the seven "parent" nodes has children "nodes", the children nodes represents the houses, using CINEMA 4D the children nodes were given names that are concomitant with the data base records(see section3.4). The "karakol" Model on the other hand has all the objects of one type wrapped in a single node, using CINEMA 4D objects were manually selected and separated into an independent tag(node) for each object and named in similar matter to "Tuekta".
Finally, the interior objects of the burial chamber were textured to give them more realistic appearance. New materials were created and added the materials library, and were textured using images, then they were wrapped around the chamber's objects.

Among the many 3D formats that CINEMA 4D can export, COLLADA is the XML based format that is compatible for web representation, however, Cesium natively supports 3D models using GLTF format, the open source **collada2gltf** tool was used to convert the COLLADA files to GLTF, the GLTF files and their associated textures -image files- were saved in a directory on the server to be served as a static content.

3.3 Design

The two main consideration for the system design of the "Uch Enmek" application are the available data sources and the objectives of the application itself. The proposed design and implementation aim to fulfill the basic required aspects mentioned in chapter two(see 2.2.4)

and to provide solutions for the known issues. These issues are mainly related to the 3D virtual models. One of the proposed usages of 3D models in archaeology is **data sustainability**, this can be achieved by making the relevant data accessible and reusable by users [43], at the same time integrating VGI with the 3D model in a joint visualization requires adapting an efficient standard. One approach is the use of CityGML , which is a promising solution [34], however, since CityGML is originally designed for urban related applications, adapting it in archaeology cannot follow a predetermined path [42].

The application provides a similar approach, which is to store all the information of the 3D model objects in a database and to use the 3D model visualized in Cesium as an interface to interact with the relevant data. The same functionality can allow users to generate and edit the data. The architecture of the database is not one of the thesis objective, so the use of a flexible database seems reasonable, a simple schema-less noSQL database will be used for the application, a schema can be added to the noSQL database in the future if a database design is determined.

Another major consideration is the underground objects which may considered the most valuable feature in the entire model. The Kurgans burial mounds models consist of furnished chambers which means to view them the user should be able to navigate inside the chambers. Virtual globes -including "Cesium"- are designed generally to navigate in the geographic space, in addition "Cesium" does not support subsurface navigation naturally.However, Cesium API allows the control of the navigation in the scene.Therefore, an approach to visualize the subsurface features through Cesium API is going to be used.

The system architecture of the application (figure 3.3) can be divided into two components.(1) A server side which consists of a set of applications that handle most of the dynamic content (terrain, 3D objects information) as well as the static content. (2) A client side script which makes up the front end application. the client side will provide The main HTML web pages, where the main visualization canvas and user interface will be presented, and the interaction functionalities will be applied.

Server-side Application

The server side of the application is responsible for serving the dynamic and the static content of the project -except the WMS which can be requested from the cloud through Cesium API -. The application performs the database queries based on the Asynchronous JavaScript And XML(AJAX) requests from the client. The terrain data are served using an open source tool, while a tool written in javaScript was developed to serve the database content.

Ubuntu which is an open source operating system software and one of the distribution systems of Linux is installed on the server machine. The Apache HTTP Server software is installed in the server to serve the static content of the application "HTML files, javaScript files,css files, and the associated data such as 3D models (GLTF), GIS (Geojson),Satellite imagery...etc"

Client-side Application

The client side application is developed using Cesium alongside with some additional javascript libraries(see table 3.3.1) , the application consist of three HTML pages, one contains the canvas where the whole 3D landscape is visualized and the user-interface where the user is able to

Figure 3.3: The System architecture of the web application

select and filter between different data types,and interact (navigate, zoom in, etc) with the 3D landscape, the second HTML page provides database queries, specifically the update "PUT" requests. The third HTML page is created as the an introductory page, it includes an abstract description about the "Uch Enmek" project, the user can navigate from this page to the "3D Model" to run the visualization application.

3.3.1 Used Software and Applications

The set of software, applications and programming languages/ frameworks are listed in table 3.3.1.

Used programming Languages:

Javascript is a high-level, dynamic, untyped, and interpreted programming language, the overwhelming majority of modern websites use JavaScript, and all modern web browsers, desktops, game consoles, tablets, and smart phones—include JavaScript interpreters [24].

Used frameworks and libraries:

Node.js is a runtime system for creating (mostly) server-side applications. It's best known as a popular means for JavaScript coders to build real-time Web APIs. [53]Nodes basic modules

Software / Framework	Functionality
ESRI ArcGIS	The software used for 2D data processing
Cinema 4D	The software used for 3D data processing
Visual Studio Code	The code editor for building and debugging the application
Apache HTTP Server	Web server software used to host the application
Cesiumjs	The main visualization environment
MongoDB	Database that stores the 3D objects Archaeological information
Nodejs	The framework used to develop the server-side application
jQuery	A library used to build the user interface and the AJAX requests
Bootstrap	A Framework used to build for the responsiveness of the HTML page

Table 3.3: Used Software packages and Frameworks with their purpose

are mostly written in JavaScript, and developers can write new modules in JavaScript. The runtime environment interprets JavaScript using Google's V8 JavaScript engine.

jQuery is a cross-platform JavaScript library designed to simplify the client-side scripting of HTML. [29] It is free, open-source software.jQuery's is designed to make it easier to navigate a document, select DOM elements, create animations, handle events, and develop Ajax applications. The modular approach to the jQuery library allows the creation of powerful dynamic web pages and Web applications.

MongoDB is a non SQL "NoSQL" database management system released in 2009. It stores data as JSON-like documents with dynamic schemas (the format is called BSON). MongoDB has its focus oriented to flexibility, power, speed and ease of use. The reason for choosing MongoDB as the noSQL datatbase for the project that it supports replicated servers, indexing, geospatial indexing and it offers drivers for multiple programming languages [8].

3.4 Implementation

Having the data prepared and with consideration of the system design, the process of building the web application started. As explained in the previous section, the application consists of two components, the client side and the server side.

3.4.1 Server-side application

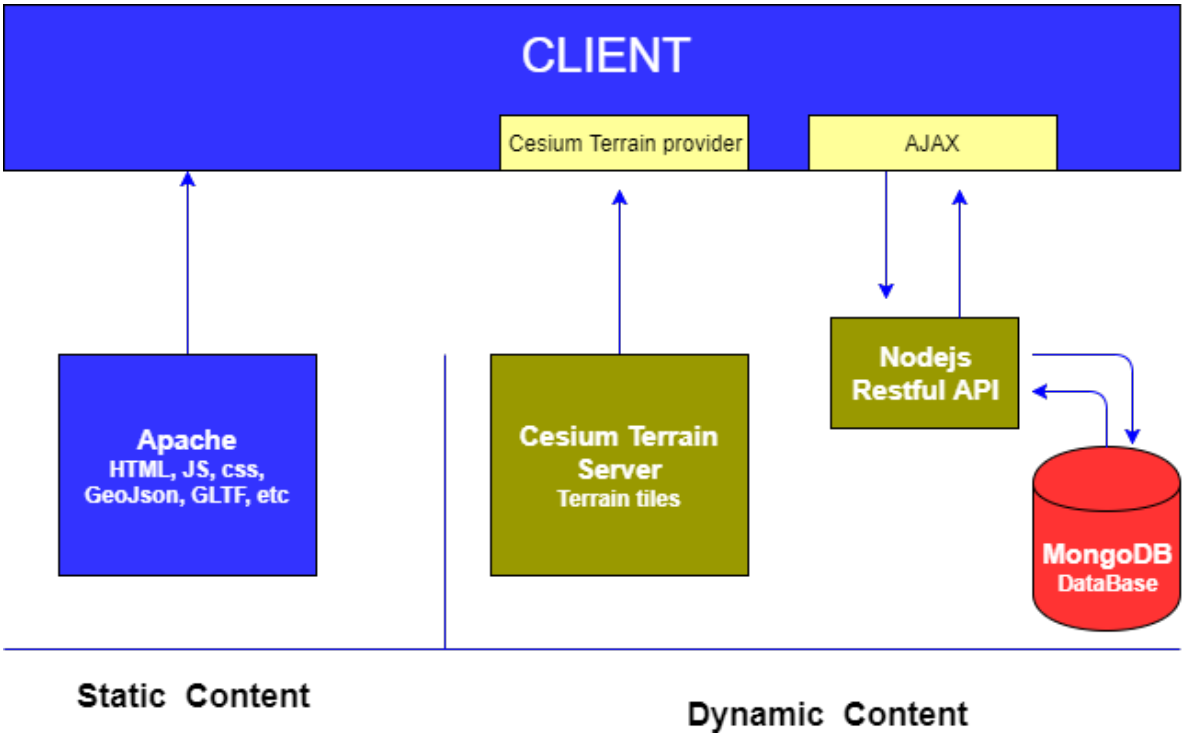

Figure 3.4: Workflow of the back-end services

The server side (back-end) application runs on a Virtual Cloud Server. The operating system of the server is the Linux distribution Ubuntu. Four applications are running on the sever as services making up the server side application. **APache HTTP server**, **MongoDB**, **Cesium Terrain Server** and **Uch Enmek Server**. The services provide static and dynamic content of the application to the client side (see figure 3.4).

The services are ruining using **Systemd**. Systemd is a Linux initialization(init) system and system manager. The main functionality of an init system is to initialize the components that must be started after the Linux kernel is booted [23]. The init system is also used for service management.

Systemd can perform a set of tasks targeting a "unit" (service). The tasks are stored in a '.service' file. After configuring the unit (.service) file, running systemd commands will **start** the service, **stop** the service, and **enable** the service, enabling the service will automatically run the service when the machine is rebooted .
Below is an example of "tuekta.service" file, it contains the configurations of "Cesium Terrain Server" that is serving the "Tuekta" terrain tile set:

```
[Unit]
Description=tuekta terrain srever - serving terrain tiles for tuekta village
Documentation=
After=network.target

[Service]
Type=simple
User=baloola
WorkingDirectory=/home/baloola/work/bin
ExecStart=/home/baloola/work/bin/Cesium-terrain-server -dir /home/baloola
    /terrain/tuekta -port 8888
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

The [Unit] section contains the variables "Description" and "Documentation", which describes the service functionality and purpose. The 'network.target' value of the "After" variable tells systemd to start the service when the server boots up, but it should run only after the server is connected to the internet.
The [Service] section contains the main configurations of the service. The most interesting variables in this section are "ExecStart" which contains the command that should run to launch the application, and "Restart" which specify the condition of restarting the application.

The script bellow shows the commands used to run the tuekta service after configuring the tuekta.service file. Other services - except Apache HTTP sever - where initialized in a similar manner:

```
$ sudo systemctl daemon-reload
$
$ sudo systemctl start tuekta
$
$ sudo systemctl enable tuekta
```

The four applications running as services in the server are:

Apache HTTP Server

The Apache HTTP Server software was installed on the machine to host the client side application files (HTML,JavaScript, css, images...etc). The "Uch Enmek" web page content is located in the APACHE public directory where it is served when the IP address is typed in the browser. The Apache service runs in the server by default when the software is installed and configured.

Cesium Terrain Builder

The Cesium Terrain Builder [59] is an open source basic server for serving file based tilesets representing terrain models. The tool is created by the GeoData institute of University of Southampton specifically for testing the terrain tiles created using "Cesium Terrain Builder".

The tool is written in **GO** programming language and requires Go to be present in the system to be installed. Therefore, version 1.6 of GO was installed on the server, and using

the command "**GO get**" the terrain server was installed on the machine. The terrain server provides a set of command line options, the commands were used to specify the path to the tilesets, and the port number. The "Tuekta.service" example above describes the steps taken to run this application as a service.

MongoDB

MongoDB database was installed on the server and is also running as a service, it is responsible of the creation,management and hosting the physical database of the application project. MongoDB can be communicated from the "Server" application and perform the requested queries.

Uch Enmek Server

The "uch Enmek" server application is a Representational state transfer (RESTful) API, it was developed in nodejs framework and then deployed in the server. Running as a service called "server", the node application allows the client to communicate with the project's database through HTTP requests. The development process of the "server" application is described in the section below.

The RESTful API Server

"Uch Enmek" server is a basic RESTful API application developed to implement -on the server side-the joint visualization (3D model and associated information) approach of the project. The application is written in nodejs and is designed to function as a server and as a database management application, figure 3.5 demonstrates the functionality of the application.

Applications developed in nodejs can include a third party packages ("modules"), the packages that were used in this application are:

- Express: Express is a nodejs framework that provides a set of features to develop a web application, it's functions were used to set up the server.
- mongodb: The MongoDB API allows the application to interact with the MongoDB service.
- cors: This framework is used to enable Cross-origin resource sharing (CORS).
- body-parser: A framework that reads the HTTP "PUT" data and parse it into a json object.

The Application can be divided -based on functionality- into two parts: the server, and the database manager.

The server:

The server (Appendix.2.1) is the main application , it listens to all HTTP requests on a specified port (3000), and performs the reactions whether it is "GET" requests or "PUT" requests. The HTTP "GET" method is used to retrieve (read) data from the database. The HTTP "PUT" is used to update existing records in the database, the new data is included in the request's body. The server uses the "body-parser" package to convert the "PUT" data

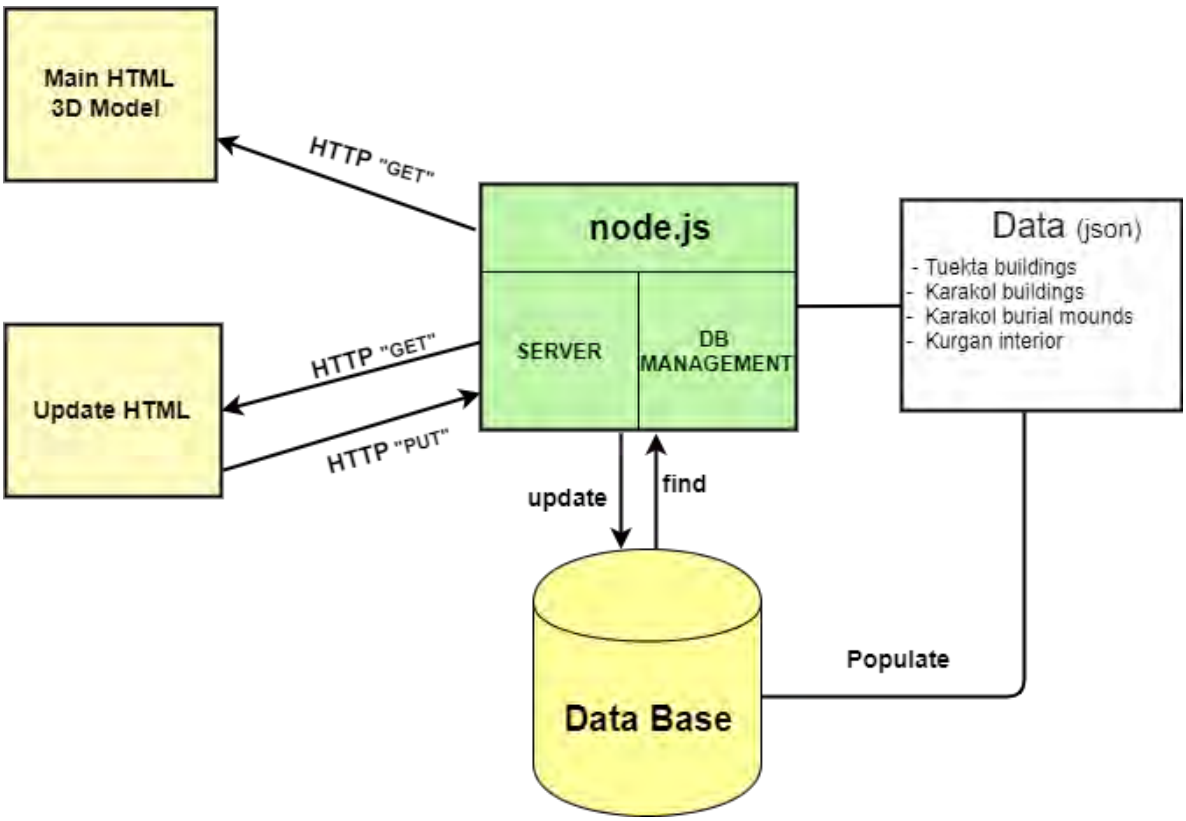

Figure 3.5: "Uch Enmek" web server functionality

in the request body into a json object. The server also calls internally the database manger which is written as module and performs its functions to retrieve or update the data.

The Database Manager:

The Database Manager is developed as a Module (Appendix.2.2), it consists of a set of functions and it is called by the server. The Database manager performs the following tasks: Database Creation, Database Connection, Fetching Data and Updating data

-Database Creation:

The created database has no pre-specified framework, it is aim to work as a prototype to implement the application's VGI / joint visualization approach. Other reasons for creating a structure-less database are:

- The Project objective is not related directly to the use of Database in archaeology.
- There is no archived archaeological data in the "Uch Enmek" geodatabase.

The "Uch Enmek" database (see figure 3.6) consists of one collection "buildings", which contains four documents, each one archives the information of one of the 3D model's objects group -parent node-. The documents are: **Tuekta** for Tuekta buildings, **Karakol** for Karakol buildings, **Archaeo** for the burial mounds and **Indoor** which contains the furniture and the relics inside the burial chamber.
Each document consist of three fields: "object id", "description" and "image url". Since the

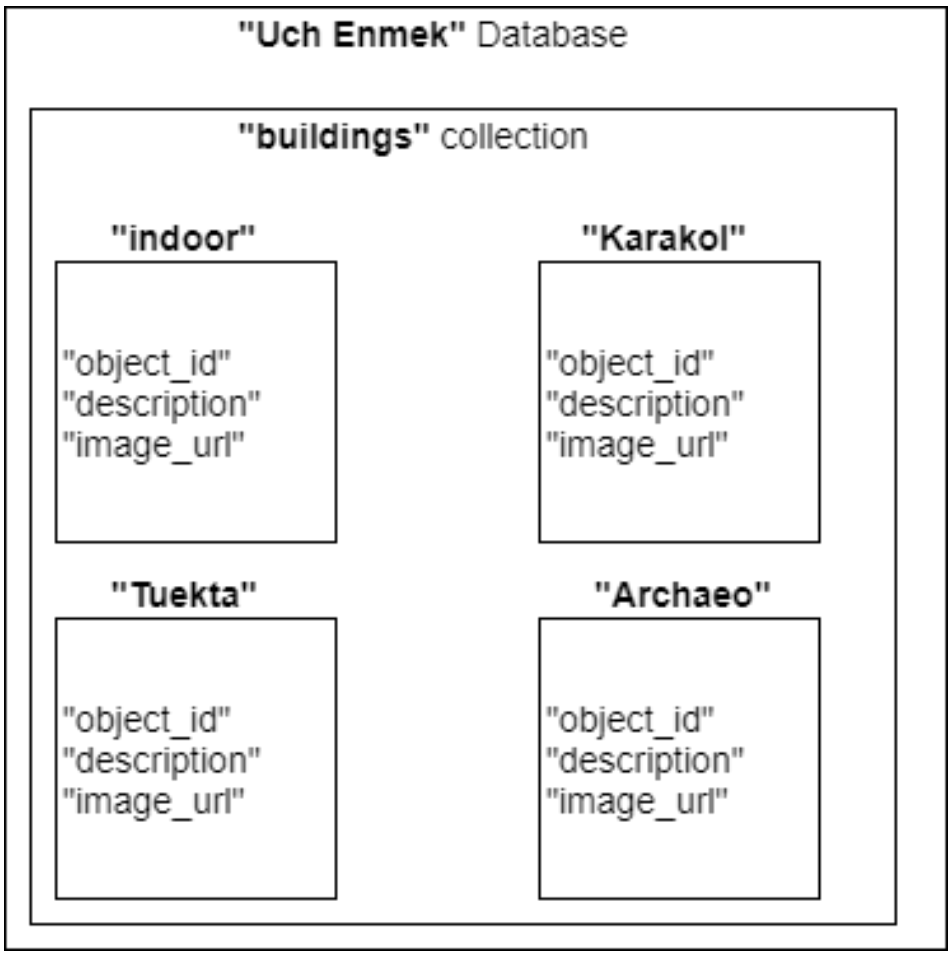

Figure 3.6: "Uch Enmek" Database

3D models were derived from the 2D GIS data -except the kurgan interior-, the database can also be generated from the 2D GIS data. Therefore, the "database creation" function loops in the associated Geojson files (see figure 3.5) and parse the GIS attributes into the Database records, the "Class Code" attribute value is assigned to "Description". The "object id" values were added to the shapefiles in ArcGIS before exporting them in Geojson. "Object id" plays the same role of the key-value in relational database. The interior of the burial chamber (Dießner 2013) 3D reconstruction was not modeled based on the 2D geodata, so the "indoor" fields were created manually and saved in a json file to be generated by the "database creation" the rest of documents. Additionally , all the "image url" fields were given a unified image path.

-Database Connection:

The connection function connects the application to the "buildings" collection. first, it checks if the database exists if not it runs the "database creation" function, then it connects the application to the created database. This procedure insure that "create" function runs only when the database does not exist, so it is called the first time the application is running, and/or when the database is deleted. This functionality was implemented to ease the application's deployment in new machines. After the connection is initialized, the application can finally act as a middleware.

-Fetching Data:

This task is performed by "find by id " function, this functions is exported to be called in the main (server) application. using MongoDB API It fetches the filed information ("description" and "image url"). First,it perform a search where the "object id" value is equal to the request "id" value, the response function is to send back the resulted information.

-Updating Data:

The "update building description" performs data updating task, it first apply the same search mechanism as "find by id ", but it response differently. Instead , it parses the request body into an object, the object consist of "description" and "image url", then it replaces the filed records with this object if the field exist, if not, it adds this object which means that a new field was created. This function is also exported as module and is called in the main application.

3.4.2 Client-side application

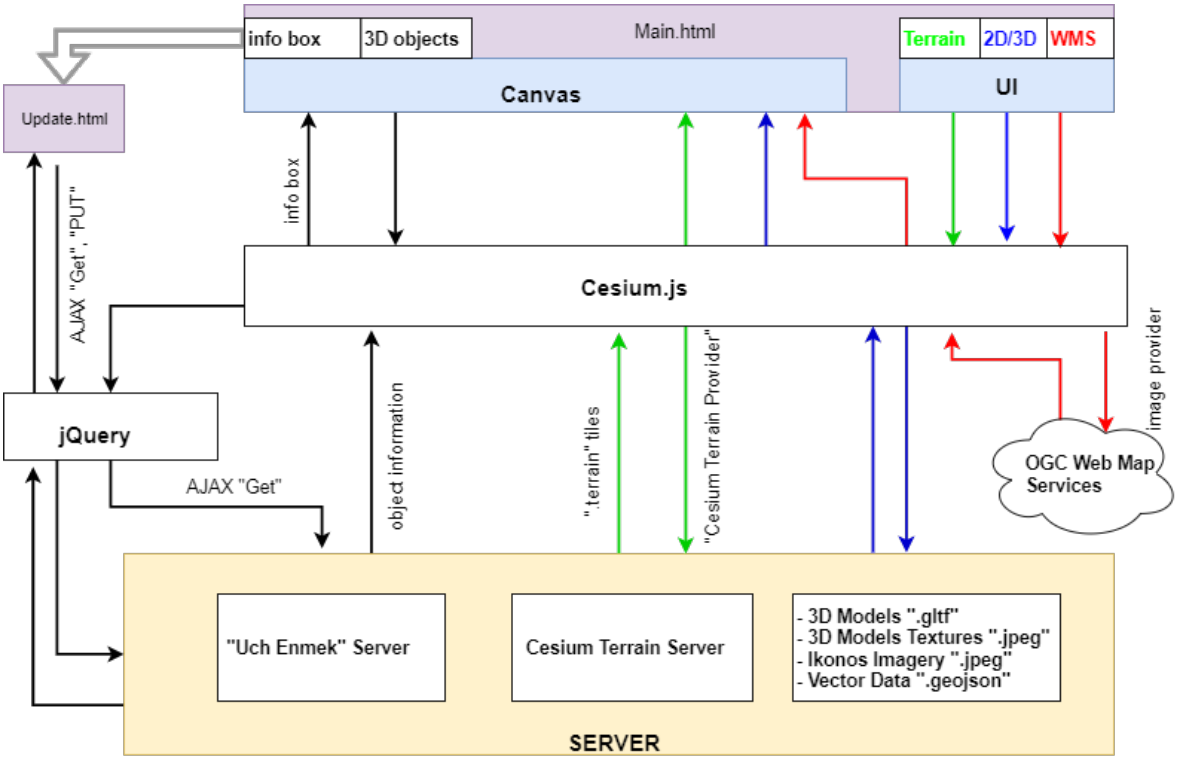

Figure 3.7: Workflow of the front end application

The client-side (front-end) web application represents the primary component of the project, it carries out the two main functionalities of the application: **visualization** and **interactivity**. the front-end application runs on top of the browser as a user interface allowing users to perform desired functionalities.
Figure (3.7) illustrates the general processing of the application, which is explained in the following sections:

Initial interface

the application is initialized when the user requests the main.html file from the server Then, the javascript program (Appendix .3.1)start running and main.html body elements are rendered in the browser window. The main body elements are: a **canvas** to view and interact with the visualization, and a **side menu** which acts as a user interface(UI). The side menu (UI) provides three site visits: "UCH Enmek" which is an overview of the natural park, and two sites "Karakol" and "Tuekta", additionally the UI allows the user to interact with different data types of each selected site.
The initialization processes performed by the program running in main.html are:

- Using Cesium.js the program renders the main scene, which is the virtual globe. The visualization canvas style is customized to fit the design of the HTML page, the program uses a plug-in called "**Cesium-navigation**" [2] to add a compass and a distance scale to the canvas.

- The program runs a set of background process, thy load the data sets (imagery, 3D models and geojson) and store them in the client machine(cache). After the processes are complete, the data can be visualized when requested.

- From the three site visits, the program runs the "Uch Enmek" visit. **"UI interaction"** section will include an explanation on how the "visit" works.

Background Processes:

The program runs a set of functions that uses Cesium.js provided functions to request and prepare the data-sets to be viewed by the user. These functions run only on initialization, the user then shows or hide the data-sets through the UI interface, this mechanism reduces the load of requesting / removing data-sets. The implemented functions add the following :

- Base imagery : using "Single Tile Imagery Provider" function, the satellite IKONS images and the topographic map are imported and georeferenced, but not added to the visualization (imagery layers).

- Labels and billboards : they are added as entities, and their visibility is set to hidden.

- Vector data : using "GeoJson Data Source" function the geojson files are loaded and configured as a "Data Source" and it awaits to be added to an entity to be visualized. The geojson color is assigned based on the layer class.

- 3D model: the GLtf files are added to the application as an entity, specifying the position of the model's center in the globe will place the model in the required place. after adding the models, their visibility is set "hidden".

UI interaction

The user interface consists of a javascript program (Appendix.3.2) and an HTML side menu with clickable items and buttons, they perform the tasks of sites visits, show/hide (Satellite imagery),(3D features),(Vector data), base map selection, and indoor Mode initialization. The "switchable" data sets are:

Imagery:All images are initially loaded to the scene, georeferenced(since they are no longer GeoTIFF) and are hidden. upon "switch on" request, the image is set to be visible and on

top of other imagery layers, when "switch off" the layer is set to be hidden.

GIS Layers: the geojson layers are loaded and stored in as an object called "datasource" which is a collection of entities. "switching on" a layer will add it to the scene entities, and will be then visualized, switching off the layer will remove it from the scene entities but not from the "datasource" collection.

3D features : there are two types of 3D features: The 3D model and the terrain. When the application is initialized the 3D models are loaded and positioned and set to be "hidden", and it will be visualize by resetting their visibility. The terrain is requested with the "Cesium Terrain Provider" which requests the terrain tiles from the "Cesium Terrain Server" service in the server.

The base map selection process is a requested using "imagery providers", the two provided OGC web map services are: Bing maps, and open street map OSM. The third base map option is the topographic map which is a single tile image, it is loaded in similar way to the satellite imagery.

The data sets are provided separately for each site. When the user requests a "switchable" data set, the program visualizes the data-set of the current visited site. And when the user requests a new site, the program visualized the "switched" data- sets of the newly visited site, and remove the data of the previous site. Figure 3.8 demonstrates the workflow of the program when the user requests a site visit.

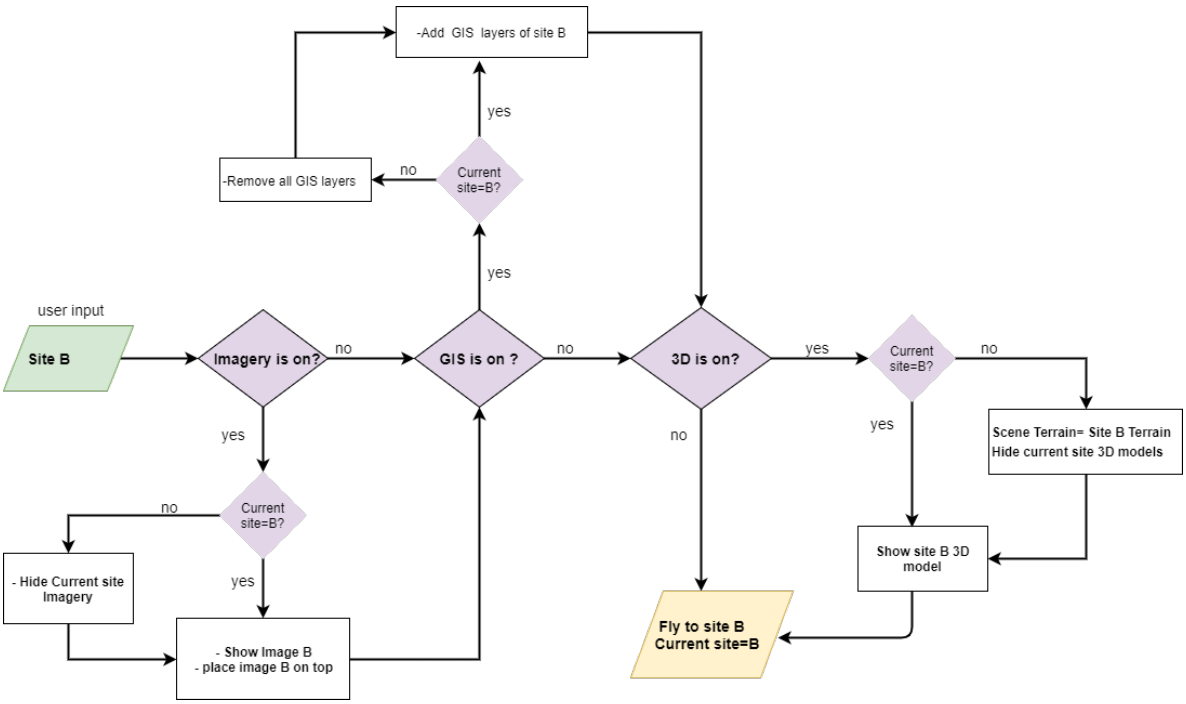

Figure 3.8: The UI execution flow

After visualizing the selected data-sets of a site, the camera moves "Fly" to a predefined location and orientation to have an overview of the site. When the user requests a site visit while in the site, the camera "Fly To" is the only function that is executed. This design allows the addition of a new sites to the application in the future without the need to add new functions.

Canvas interaction

The canvas performs the visual interactivity tasks and, the entire visualization tasks of the application. After the application is initialized and the scene is rendered, the user can interact with the virtual globe and the visualized features using Cesium API default interactive functions. Using the mouse or the touch the user can navigate the virtual globe by panning, zooming - in/out-, and rotating the view.
Cesium API allows the user to interact with the geojson files by default, clicking on a vector feature will create a box that includes attributive data of the selected feature.

The program, however runs a set of functions to give the user the ability to interact with the 3D model, this procedure is explained in diagram (3.9).
When the user clicks on an a 3D object, the program responses by viewing two type of information panels: a small **billboard** contains the geographic coordinates of the selected point, and a small HTML document embedded inside the canvas (**iframe**) that contains the 3D object's information. This procedure is implemented as the following:

- The "mouse left click" is registered as an event, which means every time a click happens, the following tasks are executed.

- The program stores the position of the point that the mouse have picked in the scene (Cartesian x,y,z).

- The program checks if this point is defined " belongs to an identified 3D object in the scene -e.g. ground, 3D model-". It then proceed if this 3D object is a node that belongs to a 3D model (gltf). The program execute these two tasks: "billboard creation" and "iframe creation" :

- **Billboard** :
The program convert the Cartesian coordinates into cartographic coordinates (longitude, latitude, elevation). Then it creates and configures a billboard by (1) locating the billboard at the clicked "picked" position, (2) add with the values of the cartographic coordinates as labels to the billboard .

- **Iframe**:
Using Cesium API, one can create an iframe and assign it as a "description" of an entity. Every time the entity is selected, the iframe will be rendered on the screen. Hence, the program creates a HTML template (iframe), the template has three elements making up the content of the iframe:(1) an **image** element with a variable value of the image's URL,(2) a description **paragraph** that is also a variable, and a **link** "update" that launches a new HTML page. The content (variables) of the iframe is filled using the following steps.
iframe content: First, the program stores the id of the selected node in a variable called "id". Then, the program sends an AJAX "GET" request to the server to fetch the associated information of the record "id". The server response by sending two objects : "image url" and "description". After retrieving the 3D object's information. the program assign these values to the corresponding variables in the iframe.The program then creates a new entity, and then assign the iframe as the entity's description, and "select" the entity.

AJAX Requests

The client-side application uses AJAX framework provided by jQuery to communicate with the server. It uses two requests :(1)"GET" where "id" value is added to the request's url, after the data is received the program executes the previously mentioned tasks. (2) "PUT", where the the request uses the same url method as "GET", additionally, the request "body" contains the user input, the server response by updating the database with data embedded in the "body".

Update.html

The Update.HTML acts as an interface for the database. the user can only update the database content through update.html. The main elements of the page are tow input elements(text boxes) for the image url and the 3D object information respectively, and a submit button that will launch the "PUT" request. When the page is initialized the application executes a "GET" request, and fill the input boxes with the retrieved data.

The page is launched (appendix.3.3) when the user clicks on the "link" element in the iframe. The page needs the 3D object id to perform the AJAX requests, but due to the security policy that is enforced by the different web browsers (**Same-origin policy**), the browser prevents the application from passing variables to a new domain (or even to the parent window in some browsers). The way the program overcome this issue is by passing the node's id to a query string, then attaches the query string to the "link" element . Below is the url of the "link" element that navigates to update.HTML,

```
www.update.html?id=${id}
```

In the url text, the query string is the part that comes after the question mark. When update.html is launched the program retrieves the query string value using JavaScript's Browser Object Model (BOM), and then proceeds with the previously described tasks.

Indoor Mode

One of the most important features in the "Uch Enmek" 3D landscape are the underground reconstructions of the burial chambers, which known as "Kurgans". These tombs are considered to be the main focus of the archaeological activities in the park [10]. However, it is not an easy task to provide a navigable visit to these chambers. There are two factors making the visualization of Kurgan interiors such a challenge in Cesium:(1) They are under the surface,(2) The burial tombs are an indoor features. Since Cesium is a virtual globe, the default navigation events like: Left click and drag (camera rotation around the globe) and middle wheel scrolling (zoom in/out) are developed to appropriate outdoor navigation. Additionally, the camera cannot penetrate surfaces.

One of the proposed approaches was to use the Cesium "Ground-Push" plug-in [16], which "excavates" the terrain of a defined rectangle. However, The tool is not maintained anymore and is not supported in the recent releases of Cesium.
The implemented approach was to develop a navigation "mode", where the user control inputs(events) are different than the default ones. Inspired by the Cesium "Camera" tutorials [4], a module "Indoor" (appendix.3.4) was developed to initialize a visit to the inside of the Kurgans and to provide new control events to navigate inside the tomb.

Once the mode is initialized, five HTML buttons are set to be visible: UP, Down, Balance, Exit and Help. The program disable all of the navigation events, and define the new indoor navigation events. The new events are (1) **Mouse/Touch drag**"mouse left down (press), mouse move, and mouse up (release)": when the user drags the mouse, the view direction

changes based on the dragged distance and direction. (2) **UP arrow/ UP button**: moves the camera forward (towards the camera looking direction) using keyboard's UP arrow or the UP screen button. (3) **Down arrow/ Down button**: moves the camera backwards. The camera move speed "move rate" is proportional to the camera height. Since the Kurgans height values are more than 800 m , the single move event would move the camera outside the tomb. There fore the move rate is multiplied by 0.1 to appropriate the camera movement.

After defining the new navigation events, the program renders the 3D model, then "fly to" the chamber and sets the orientation angles (Heading,Pitch,Roll) so that the camera is looking at the horizon. The **Balance** button executes a "fly to" method to the camera's current position and orientates the camera with the current heading while sets the Roll and Pitch values to zero. This procedure will balance the camera around the z and x axis making it looking at the horizon while maintaining the same azimuth. The **EXIT** button re-sets the application's navigation events and "fly to" the previously visited site.

HTML Design :

The introduction HTML page was created using Mobirise, a free website builder software that uses bootstrap framework (see table **??**). This introductory HTML page was used as an HTML template for the main application page, and the update page, which were designed using own developed css script and bootstrap framework. Additionally, Cesium css script was hacked to alter Cesium's canvas style(e.g. infobox, help button) to harmonize HTML design of the application.

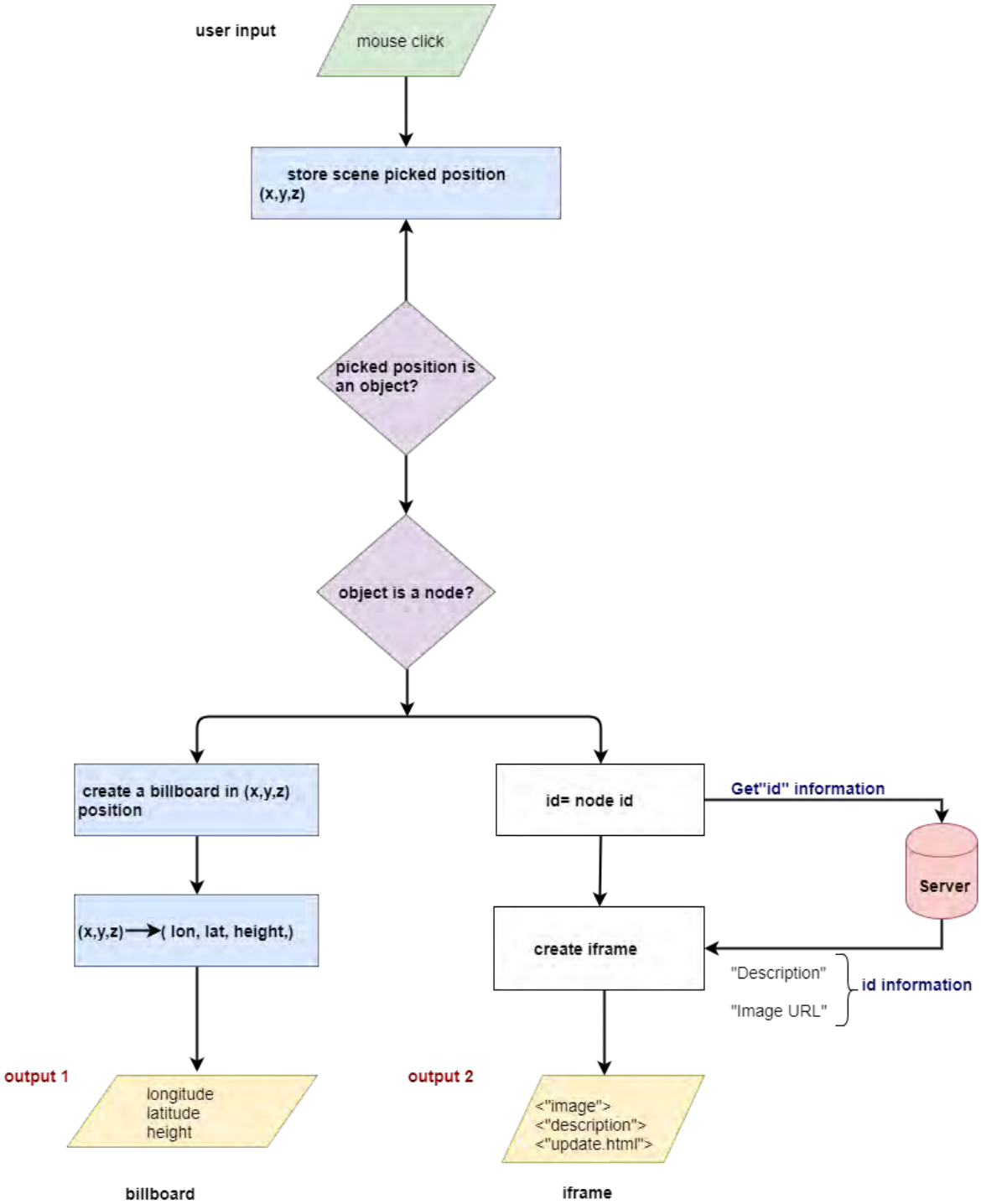

Figure 3.9: Canvas interactivity algorithm

4 Results and Discussion

This chapter presents the final results of the thesis, it demonstrates the functionality and performance of the final product and discusses the related research aspects in a critical manner.

4.1 Results

A domain name was pointed to the virtual cloud server hosting the application, users can access the introductory page by visiting `www.baloola.eu`, the web page (see figure 4.1) is a prototype for the project's Home web page, it is designed to have an introduction and a description about the project. The navigation menu on the top of the page consists of a **logo** that links to the website of the Institute of Cartography, a title **Uch Enmek** that links to the main page and three buttons: "**Home**" which is a links to the home page, "**3D Model**" that links to the web application, and "**Options**" which is disabled in the main page. Clicking on "3D Model" will initialize the main web application.

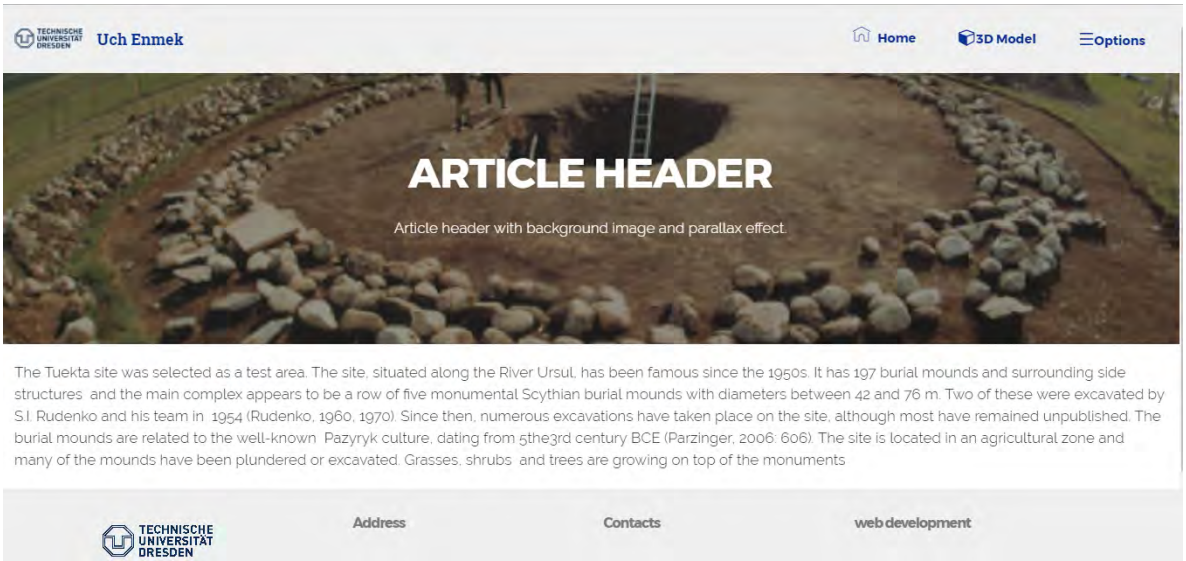

Figure 4.1: `www.baloola.eu` home page prototype

4.1.1 Web Application

The application interface (4.2) consists of Three main elements: The navigation menu, the canvas and a side menu. The navigation menu can be used to navigate from/to the application

and to "toggle" the side menu by clicking the "**Option**" button.

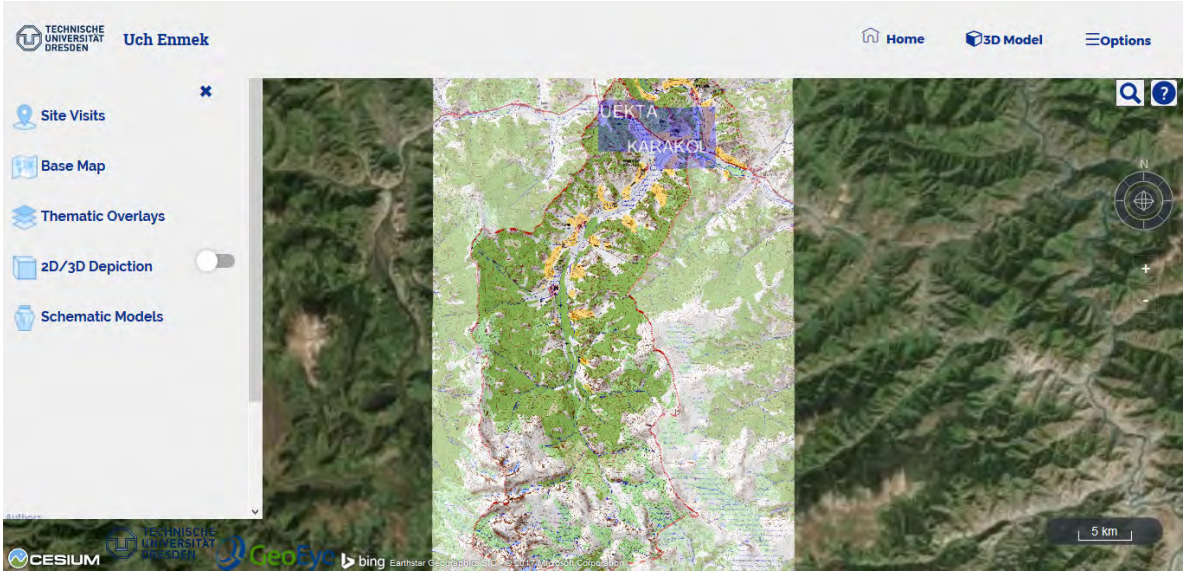

Figure 4.2: initialized view of the application, the "Uch Enmek" Etno-Nature Park

The canvas interface has two Cesium's default buttons located at the top right corner of the canvas :"Help", which opens a navigation instruction box, and Cesium "Geocoder" for finding addresses and landmarks, and flying the camera to them.Beneath the default buttons, Cesium-navigation tools are located on the right side of the canvas. They are a Compass, Zoom in/out buttons and a distance scale on the bottom right corner of the canvas. The bottom left corner of the canvas contains a set of organizations logos (Cesium, TUD, GeoEye and bing).
The (UI) panel is a side menu that slides from/to the left side of the canvas, it contains four sub-menus and a button, the user can use this menu to perform the visits, filter data, and switch modes. Additionally, the project's data authorship information is mentioned in the bottom of the side menu.

User Interface

The side menu has four "dropdown" panels: "**Site Visits**","**Base Map**","**Thematic Overlays**" and "**2D/2D Depiction**", and a button "**Schematic Models**".

- **Site Visits:**

The application provides three site visits, "**Karakol**" and "**Tuekta**" for the two archaeological sites, and"**Uch Enmek**" which is an overview of the whole "Uch Enmek" Ethno-Natural Park. The application preforms "Uch Enmek" visit when it gets initialized (see figure 4.2), it layers the topographic map of the Karakol valley with an outline of "Uch Enmek" park and two rectangular frames for Tuekta and Karakol sites respectively. When in this state (visit), the user cannot view the different data-types (e.g. Vector layers, 3D models) since no site is specified (Tuekta and Karakol).
When the user switches between the two sites (Tuekta and Karakol), the "switched-on" data-types of the newly visited site will replace the data of the previously visited site.

- **Base Map:**

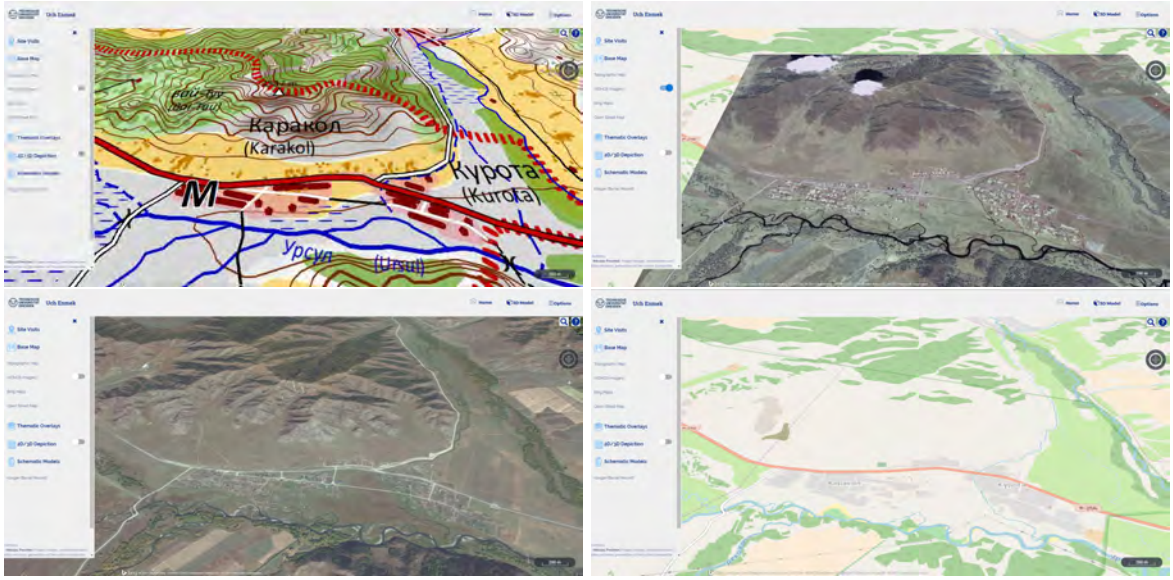

Figure 4.3: The Four base layers covering (Karakol)

The base map dropdwon panel contains the available base maps/imagery for the application (see 4.3). The application provides two web map services "Bing Maps" and "OpenStreetMap", and two single tile base layers: The topographic map which covers the area of "Uch Enmek" and the IKONOS imagery . The Two "IKONOS" images are provided separately for each site (Tuekta and Karakol). When requested, the IKONOS image is layered on top of the globe's imagery layers.

- **Thematic Overlay:**

The "Thematic Overlays" panel consists of three buttons: "**Natural Features**", "**Man-Made Features**" and "**Archaeological Sites**". Each button represents a feature type. When The user switch on any of the layers, a map legend slides from the right side of the canvas (see figure 4.4).

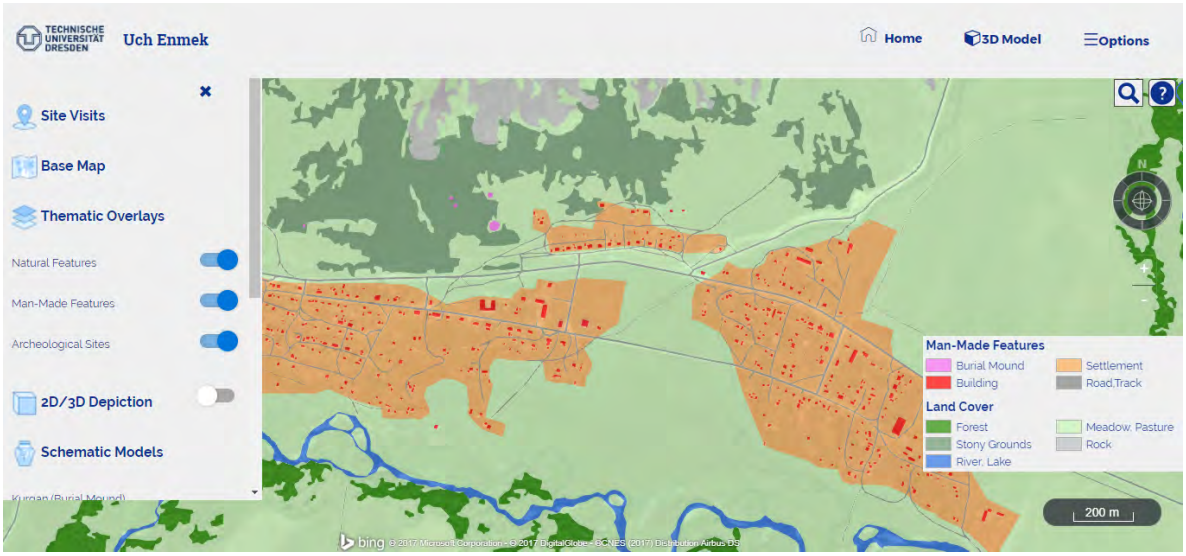

Figure 4.4: Thematic layers and the legend (Karakol)

- **2D/3D Depiction:**

The "**2D/3D Depiction**" button provides the 3D landscape visualization. When switched on, it visualizes the 3D prototypes of the "current" site, additionally it requests the terrain tiles of the site from the server and renders them based on the zoom level (see figure 4.5).

Figure 4.5: 3D landscape of Karakol

- **Schematic Models:**

The "**schematic models**" dropdown provides a list of the 3D reconstructions. In this project, the burial mounds is the available schematic model. Clicking the "**Kurgan (Burial Mound)**" link will initialize the indoor mode, and the camera will fly to the interior of the burial mound underground. The camera orientation will be altered so that it looks at the horizon, the user will be able to interact properly with the Kurgan's 3D model features.

When the indoor mode is initialized (see figure 4.10), the side menu (UI) is disabled, and new five buttons appear in the canvas: "**exit mode**","**help**","**down**","**up**" and "**balance**". Additionally, the navigation inputs are disabled(e.g. mouse wheel/zoom, left click+drag/ pan, etc).

The indoor mode provides a different navigation functionality. The user is able to change the camera direction (look around) by clicking the left button, then drag the mouse around. Using the keyboard up/down or the screen up/down buttons the user can move forward and backward. The constant movement and changing of the direction might tilt the camera view, which may confuse the user .Therefore, the balance key resets the camera orientation to look at the horizon again (see figure 4.7).

Canvas Interface

In addition to the visualization task, the canvas provides a portion of the application's interactivity functions. The user can interact with the visualization by navigating in the canvas scene using Cesium default tools(mouse controls), "Cesium-Navigation" plug-in tools (e.g. compass, zoom in/ out button), or the indoor mode customized navigation tools.

Figure 4.6: The interior of the Kurgan schematic model (Karakol)

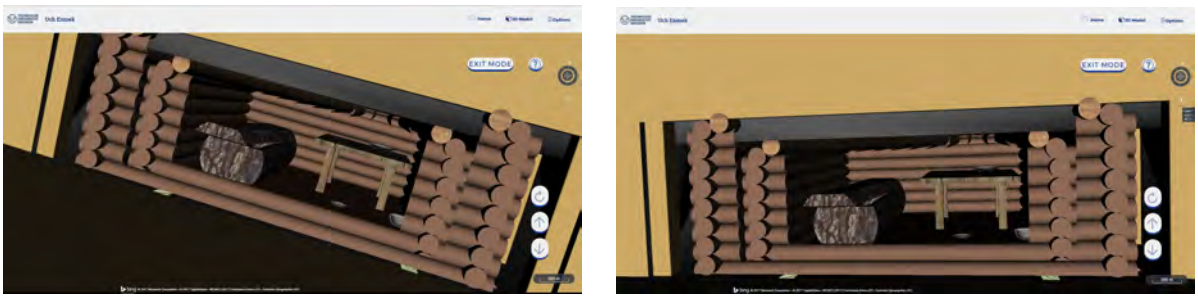

Figure 4.7: The Balance button resets the tilted camera axis (left) and balances the camera view(right)

Cesium provides joint visualization for the vector data. When the user clicks on a geojson feature, the application creates an infobox that contains the attributive data of the feature (see figure 4.8)

The user can also interact with the 3D model objects. The application provides the ability to view the associated information of the selected object and the spatial information of the selected point on the object's surface (clicked point). When the user clicks on a 3D object (see figure .13), an infobox contains a small image and a description is created, and a small billboard is visualized at the picked position, it contains the cartographic coordinates(longitude, latitude and elevation) of the picked point on the 3D object's surface.

- **Update Page:**

The user can also interact with the infobox content. The link "**Update Description**" will navigate to a new HTML page (see figure 4.10).The user can add/replace the description article, and can add a URL for the description image. When the user click the button "**Update**", the application sends a "PUT" request to the database, once the content is updated a "**Success**" button appears on the screen, clicking on it will close the update page.

Figure 4.8: Selected Building feature (Karakol)

Figure 4.9: Infobox and coordinates panel (Tuekta)

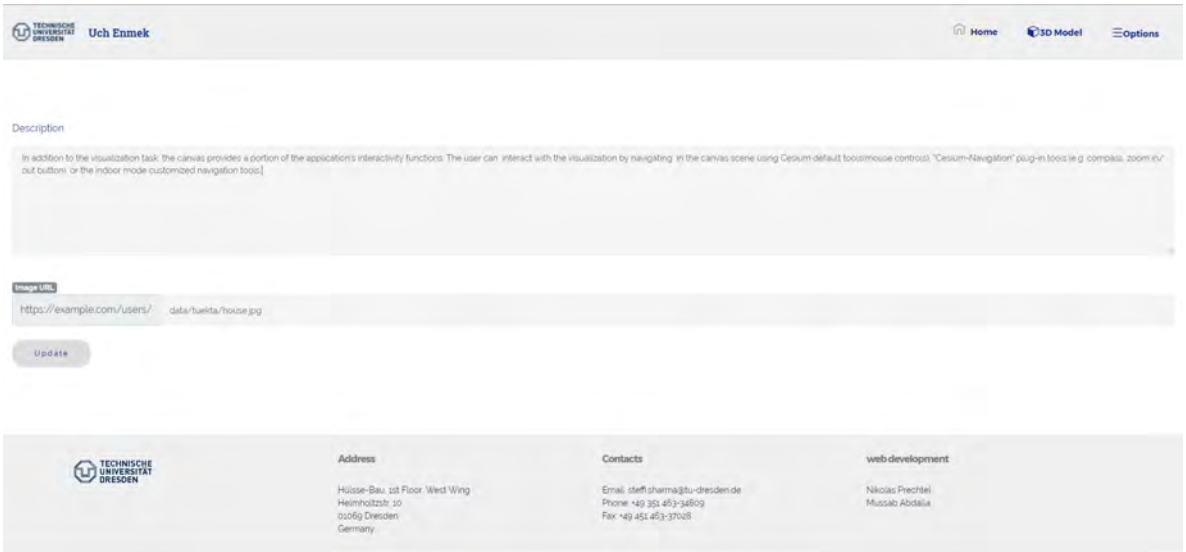

Figure 4.10: Update.html

Figure 4.11: Updated description (Tuekta)

4.2 Discussion

This section discusses the final output of the thesis, it includes critical assessment of the project in terms of meeting the thesis objective, and provides the answers to the research questions.

4.2.1 Critical issues Clarification

Spatial reference systems other than the one used in the Virtual Globe?

Cesium default spatial reference system is "**WGS 84**". According to Cesium documentation, customized ellipsoids can be defined and then override the default WGS84 in the scene initialization process. However, Cesium developers do not have enough feed back from developers about the customized ellipsoids performance [39]. According to the OpenLayers - Cesium integration library "**Ol-Cesium**" developers, Cesium only works properly with WGS84 [27].

Finally, the terrain generation tool used in the project cannot properly process data sources in spatial reference systems other than WGS 84.

Flexibility in respect to customized digital elevation models and image data?

Cesium API provides the ability to request terrain tiles from multiple datasets: (1)**Cesium STK World Terrain**,(2) **Cesium Small Terrain**,(3) **Esri ArcGis Image Server** and (4) **VT MAK VR-TheWorld Server**. Additionally, Cesium API requests and visualizes terrain tiles from third party servers, as long as the formats of the elevation dataset are supported.

The main challenge in the project was to produce terrain tiles in a a format supported by Cesium. The open source tool that was used to generate the terrain dataset accepts as input any (GDAL)-based raster Digital Elevation Models(DEM). Additionally, an open Source server application was used to serve the terrain tiles in the server side.
Alternatively, a commercial solution provided by **"Analytical Graphics, Inc (AGI)"** is also available to generate and serve customized DEM's.

Cesium also provides multiple web map services (WMS). Customized imagery can be tiled, and then requested by Cesium. However, the image data provided for the project, has relatively Small coverage, so it was imported as a single tile to the scene.Since Cesium does support Geotiff imagery, single tile images should be converted into another acceptable format, which means that the geographic information will be lost and the resolution will be reduced, the geographic information of the image extent must be provided to position the image correctly. To conclude, Cesium requirements for customized imagery and DEM's can be provided by any GIS software and open-source tools.

Object-by-object import versus import of compact object groups

The core of gltf model is a JSON file with the following top-level elements : "**scenes**" (basic structure of the model). "**nodes**" (which contain meshes). "**meshes**" (3D geometries). ("**buffers**", "**bufferViews**", "**accessors**"). "**materials**" (defines how objects will be rendered), ("**textures**", "**images**", "**samplers**") which defines the surface appearance of the objects.

The "image" object refers to image files that contain the texture data (as URI or it can be included directly in the JSON), every time a gltf file is rendered in the screen, the browser requests the associated images and projects them (wrapping and scaling) on the objects surfaces.

This means if we imported several gltf files -even if they share the same image files- the browser will repeat the process of requesting the image files with every gltf import. Therefore, the loading time will increase with number of imported (textured) gltf files.
Thus, the compact object group seems to be the better choice.

Handling of surface and subsurface (Kurgan) objects

The 3D objects height values and the terrain tilesets were derived from the same elevation data source. Therefore, there were no need for special processing for the 3D model features to assure that they will be clamped to the ground accurately.On the other hand, the ability to navigate in the scene was the key factor in the visual interaction with the surface and subsurface features.

The performance of Cesium default navigation tools cannot be assessed without a user study. However, (De Roo 2017 [19]) have noted that some of the usability problems in Cesium was the standard zoom functionalities of the Cesium Viewer. additionally the lack of a compass might make it difficult to re-orient. Therefore, the the application used "Cesium-Navigation" plug-in to add additional tools in order to optimize the outdoor navigation.

As for the subsurface objects, the level of camera control provided by Cesium high-level API, has enabled the development of the subsurface/indoor mode. Which allows the user to navigate inside the kurgans using customized navigation functionalities. However, the implemented approach is a experimental and needs to be further optimized. Perhaps an adaptation of the **First-person perspective** concept which is used in 3D video game developments [33], would provide the best possible navigation mode for the indoor features.

Parameters influencing the performance

As already discussed, the compact importing of a model will reduce the runtime of Cesium applications. However, the use of non-pictorial materials (e.g. Karakol prototypes) increases the size of the gltf file, as well as the cost of its visualization.
The 3D models that were visualized in the project were relatively small,. the size of Karakol gltf file and the associated images (for texturing) are less than 8 MB, while Tuekta 3D model files are around 20 Megabytes. Therefore there were no noticeable effect on the general performance, and t was difficult to measure the influence of factors like the total mesh number, or node hierarchy in "Karakol" and "Tuekta" 3D models. The integration of other type of 3D models (point cloud, photo-generated) in the application would provide a better understanding for the influence of the mentioned factors.
A test for the web application performance was conduced (`https://www.webpagetest.org`. it shows that imagery data, have the most notable effect on the application performance (Appendix.1).

Operating ability of the application in different browsers and on smart phones

The application operability in different browsers was evaluated by running the application in four browsers (Chrome, Firefox, Microsoft edge, and Safari). The web page was requested in each browser and main functions were executed. The application shows similar behavior in the different browsers. The exception was a failure running the update functionality in Microsoft Edge. The reason is that Microsoft Edge prevents the program which uses "location.search" method from accessing the URL box and then retrieve the building id value to execute the AJAX requests.

As for the smart phones, the application was tested in an iphone 7, Safari browser and a Galaxy S6, Google chrome. The two tests shows two similar issues:

- The single tile imagery were not rendered (IKONOS imagery and the Topographic map) see figure 4.12.

- In the indoor mode the (UP) and (Down) keys does not response which means that the camera cannot move, while the direction changing (touch and drag) and the balance button.

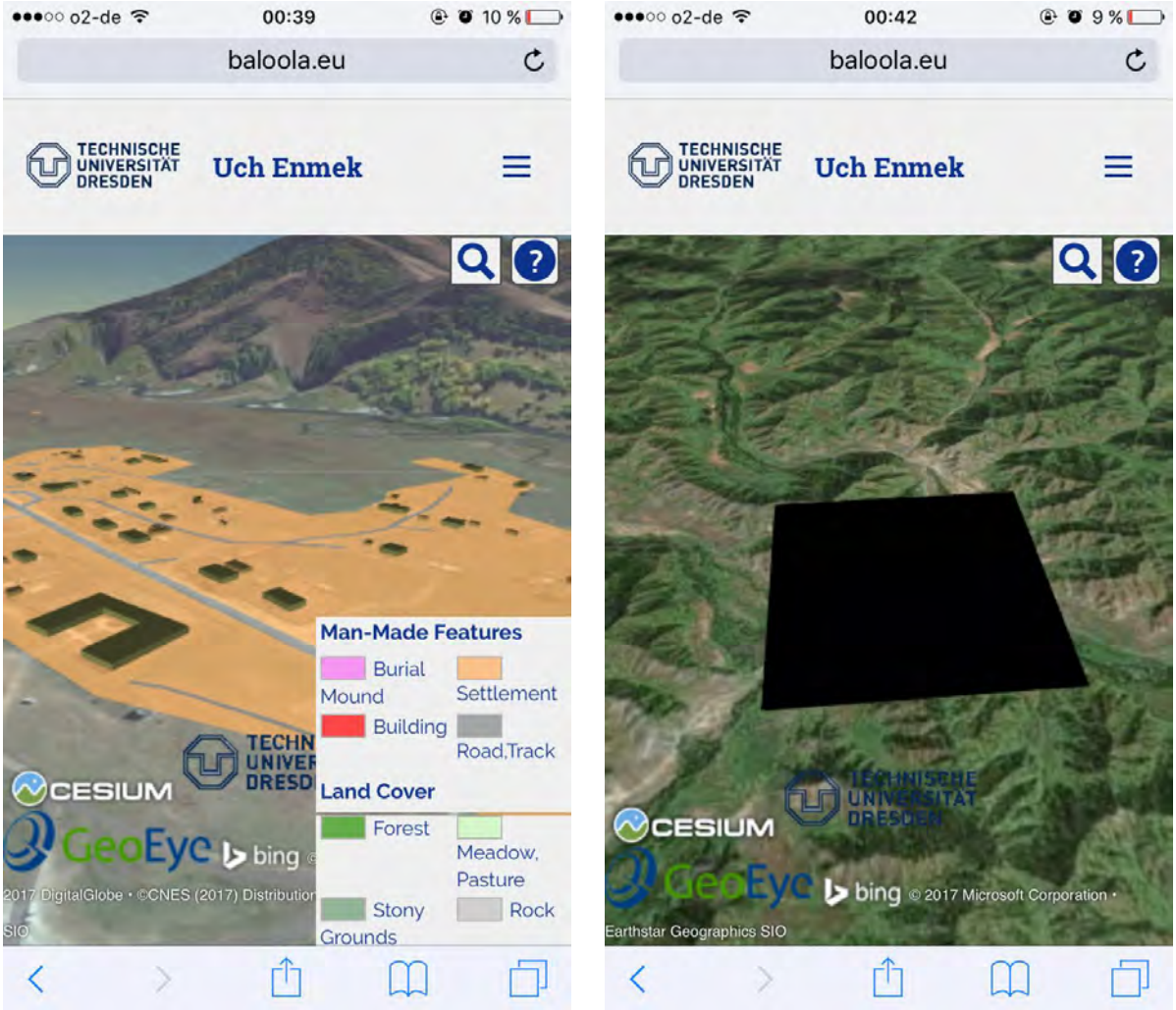

Figure 4.12: Karakol 3D landscape on iphone 7 (left). IKONOS imagery of Karakol appears as a black rectangle (right)

4.3 Conclusion and future work

The theoretical part of the thesis provides an overview of the current implementation of virtual globes in the context of cultural heritage, and an assessment of the available open source virtual globes.

An application was developed in Cesium virtual globe environment to provide accessible, user-friendly 3D reconstruction. The application provides a prototype of VGI implementation, and a joint visualization solution at the same time. The thesis provides a mechanism for processing geodata to prepare it for Cesium environment visualization pipeline. A prototype was developed to interact with subsurface features within the same visualization interface, which can work as an approach to provide a full range of spatial dimension.

As an Environment, Cesium is a rapidly growing platform, which makes considerable promises about the potentials of Cesium's performance in archaeological representation. A proposal for designing database would help further explore the linked visualization, user-generated data and VGI approaches. Additionally, the use of the newly introduced Cesium's 3D tiles specification could help rendering massive amount of data. Hence, high resolution photo-base 3D models, point clouds could be integrated to the Geodata. Cesium's API functionalities can be further utilized to overcome some of the presents challenges. Tiling the base imagery data and serve it as WMS could enhance the performance of the application in desktops smart phones. Finally, the 3D reconstructions in the future should be developed with consideration of their compatibility to work with Cesium.

Bibliography

[1] NASA Ames , Ames Research Center (ARC). nasa world wind.

[2] Alberto Acevedo. Cesium navigation , 2016.

[3] Kheir Al-Kodmany. Visualization tools and methods in community planning: from freehand sketches to virtual reality. *CPL bibliography*, 17(2):189–211, 2002.

[4] Inc Analytical Graphics. Cesium Camera Tutorials , 2015.

[5] Michael Auer, Giorgio Agugiaro, Nicolas Billen, Lukas Loos, and Alexander Zipf. Web-based visualization and query of semantically segmented multiresolution 3d models in the field of cultural heritage. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2(5):33, 2014.

[6] Mark Barnes and Ellen Levy Fi nch. Collada – digital asset schema. pages 330–335, 2008.

[7] S Bleisch and S Nebiker. Connected 2d and 3d visualizations for the interactive exploration of spatial information. In *Proc. of 21th ISPRS Congress, Beijing, China*, number 1999, pages 1037–1042, 2008.

[8] Alexandru Boicea, Florin Radulescu, and Laura Ioana Agapin. Mongodb vs oracle-database comparison. In *EIDWT*, pages 330–335, 2012.

[9] Larissa Bonfante. *The Barbarians of Ancient Europe: Realities and Interactions*. Cambridge University Press, 2011.

[10] I BOURGEOIS, Wouter Gheyle, Rudi Goossens, Alain De Wulf, Eduard Dvornikov, AV Ebel, Leon Van Hoof, Stéphanie Loute, Kaatje De Langhe, Anne Malmendier, et al. Survey and inventory of the archaeological sites in the valley of the karakol (uch-enmek park). report on the belgian-russian expedition in the russian altay mountains 2007-2008. *University of Gent/GASU: Gent*, 2007.

[11] Tim Bray, Jean Paoli, C Michael Sperberg-McQueen, Eve Maler, and François Yergeau. Extensible markup language (xml). *World Wide Web Consortium Recommendation REC-xml-19980210. http://www. w3. org/TR/1998/REC-xml-19980210*, 16:16, 1998.

[12] MA Brovelli, P Hogan, M Minghini, and G Zamboni. The power of virtual globes for valorising cultural heritage and enabling sustainable tourism: Nasa world wind applications. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 4:W2, 2013.

[13] MA Brovelli, CE Kilsedar, and G Zamboni. Visualization of vgi data through the new nasa web world wind virtual globe. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 41, 2016.

[14] Manuel Burckhardt. Modellierung ländlicher Siedlungen aus hochauflösenden Satellitenbilddaten als Baustein einer dreidimensionalen Landschaftsmodellierung. Master's thesis, TECHNISCHE UNIVERSITÄT DRESDEN, Germany, 2011.

[15] Howard Butler, M Daly, A Doyle, S Gillies, T Schaub, and C Schmidt. Geojson specification. *Geojson. org*, 2008.

[16] Chris Cooper. Cesium Ground-Push Plugin , 2015.

[17] Patrick Cozzi and Kevin Ring. *3D Engine Design for Virtual Globes*. CRC Press, 1st edition, June 2011.

[18] Douglas Crockford. The application/json media type for javascript object notation (json). 2006.

[19] Berdien De Roo, Jean Bourgeois, and Philippe De Maeyer. Usability assessment of a virtual globe-based 4d archaeological gis. In *Advances in 3D Geoinformation*, pages 323–335. Springer, 2017.

[20] Politecnico di Milano. Policrowd 2.0 The Social World Wind Platform , 2013.

[21] Politecnico di Milano. The Paths of Via Regina , 2013.

[22] Umberto Di Staso, Marco Soave, Alessio Giori, Federico Prandi, and Raffaele De Amicis. Heterogeneous-resolution and multi-source terrain builder for cesiumjs webgl virtual globe. *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering*, 10(1):129–135.

[23] Justin Ellingwood. How To Use Systemctl to Manage Systemd Services and Units, 2015.

[24] David Flanagan. *JavaScript: the definitive guide*. " O'Reilly Media, Inc.", 2006.

[25] JA González-Delgado, AM Martínez-Graña, J Civis, FJ Sierro, JL Goy, CJ Dabrio, F Ruiz, ML González-Regalado, and M Abad. Virtual 3d tour of the neogene palaeontological heritage of huelva (guadalquivir basin, spain). *Environmental earth sciences*, 73(8):4609–4618, 2015.

[26] Help Google EarthEnterprise. .

[27] Beraudo Guillaume. Ol3-cesium: 3d for openlayers.

[28] Institut Cartogràfic i Geològic de Catalunya (ICGC). The Old Town of Girona as the crow flies , 2015.

[29] jQury. What is jQuery , 2017.

[30] Jessica Keysers. Review of digital globes 2015. *CRCSI: Victoria, Australia*, 2015.

[31] Khronos. collada,wiki , 2008.

[32] The Khronos Group Inc. *glTF Specification*, 11 2016. version 1.0.

[33] Eric Laurier and Stuart Reeves. Cameras in video games: comparing play in counter-strike and the doctor who adventures. *Video at work. New York: Routledge, p. NYP*, 2014.

[34] M Milanese. A novel approach to 3d documentation and description of archaeological features. In *Proceedings of the 38th Conference on Computer Applications and Quantitative Methods in Archaeology*, volume 38, pages 1–7. FJ Melero, 2010.

[35] Marco Minghini. *Multi-dimensional GeoWeb platforms for citizen science and civic engagement applications*. PhD thesis, Italy, 2014.

[36] blog Nasa worldwind. , 2014.

[37] TU Dresden Department of Cartography. The Alti Project tu dresden.

[38] AGI Pattrick Cozzi. Graphics Tech in Cesium - The Graphics Stack , 2015.

[39] Hanna Pinkos. cesium-dev, 1999.

[40] Brandon Plewe. Web cartography in the united states. *Cartography and Geographic Information Science*, 34(2):133–136, 2007.

[41] N Prechtel, S Münster, C Kröber, C Schubert, and S Sebastian. Presenting cultural heritage landscapes-from gis via 3d models to interactive presentation frameworks. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences XL-5 W*, 2:253–258, 2013.

[42] Nikolas Prechtel and Sander Münster. Cultural heritage in a spatial context–towards an integrative, interoperable, and participatory data and information management. In *3D Research Challenges in Cultural Heritage II*, pages 272–288. Springer, 2016.

[43] Heather Richards-Rissetto and Jennifer von Schwerin. A catch 22 of 3d data sustainability: Lessons in 3d archaeological data management & accessibility. *Digital Applications in Archaeology and Cultural Heritage*, 2017.

[44] Marcel Schmid. The Paradox of Overfitting. Master's thesis, TECHNISCHE UNIVERSITÄT DRESDEN, Germany, 2011.

[45] Olaf Schroth, Ellen Pond, Cam Campbell, Petr Cizek, Stephen Bohus, and Stephen RJ Sheppard. Tool or toy? virtual globes in landscape planning. *Future Internet*, 3(4):204–227, 2011.

[46] Christian Schubert. Vom 3D-Landschaftsmodell zu einer integrativen Web-basierten Informationsapplikation für ein archäologisches Schutzgebiet (Uch Enmek, Republik Altai). Master's thesis, TECHNISCHE UNIVERSITÄT DRESDEN, Germany, 2013.

[47] Richard B Schultz, Joseph J Kerski, and Todd C Patterson. The use of virtual globes as a spatial teaching tool with suggestions for metadata standards. *Journal of Geography*, 107(1):27–34, 2008.

[48] Shirish C Srivastava, Shalini Chandra, and Hwee Ming Lam. Usability evaluation of e-learning systems. In *Encyclopedia of Information Science and Technology, Second Edition*, pages 3897–3903. IGI Global, 2009.

[49] Klokan Technologies. WebGL Earth JavaScript API, 2015.

[50] D Tiede and S Lang. Analytical 3d views and virtual globes-putting analytical results into spatial context. In *ISPRS, ICA, DGfK-Joint Workshop: Visualization and Exploration of Geospatial Data, Stuttgart*, 2007.

[51] Stefan Tilkov and Steve Vinoski. Node. js: Using javascript to build high-performance network programs. *IEEE Internet Computing*, 14(6):80–83, 2010.

[52] Jennifer von Schwerin, Heather Richards-Rissetto, Fabio Remondino, Giorgio Agugiaro, and Gabrio Girardi. The mayaarch3d project: A 3d webgis for analyzing ancient architecture and landscapes. *Literary and Linguistic Computing*, 28(4):736–753, 2013.

[53] Ben Wen. 6 things you should know about Node.js , 2013.

[54] OpenGL Wiki. Opengl shading language — opengl wiki,, 2015. [Online; accessed 16-March-2017].

[55] OpenGL Wiki. Main page — opengl wiki,, 2017. [Online; accessed 16-March-2017].

[56] WebGL Public Wiki. Getting started — webgl public wiki,, 2011. [Online; accessed 16-March-2017].

[57] Sebastian Zimmermann. Vom GIS-Modell zur 3D-Landschaft – Ergänzungen und Workflowreview im „Uch-Enmek-Modell“. Master's thesis, TECHNISCHE UNIVERSITÄT DRESDEN, Germany, 2015.

[58] Homme Zwaagstra. cesium-terrain-builder , 2015.

[59] Homme Zwaagstra. cesium-terrain-builder , 2015.

.1 Web Application Performance Test

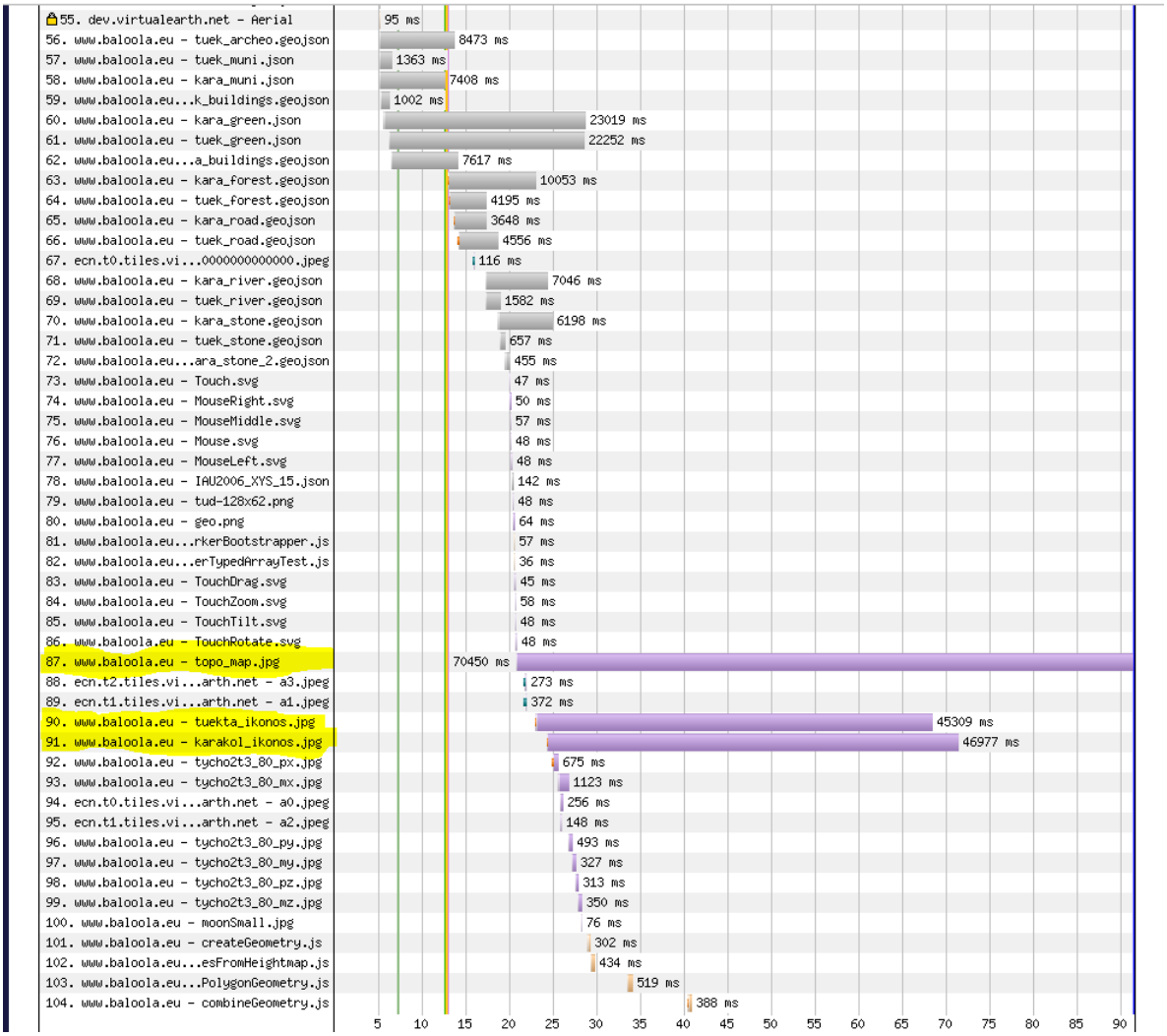

Figure .13: A screen shot from the performance test shows the load time of the imagery data "purple" and geojson data "grey" (http://www.webpagetest.org)

.2 Server-Side Application

.2.1 "server.js": Uch Enmek Server

```

var express = require('express'),
// requesting wines.js the database managment module
    wine = require('./routes/wines');
var path = require('path')
var bodyParser = require('body-parser');
var cors = require('cors');
var app = express();


app.configure(function () {
   // app.use(allowManyDomains);
    app.use(express.logger('dev'));      /* 'default', 'short', 'tiny', 'dev'
        */
```

```
    app.use(express.bodyParser());
    app.use(cors());
    app.use(bodyParser.json());
});

app.get('/', function(req, res) {

   res.sendfile(path.join(__dirname + '/index.html'));

});


app.get('/buildings/:id', wine.findById);
app.put('/buildings/:id', wine.updateBuildingDescription);

app.listen(3000, '0.0.0.0');
console.log('Listening on port 3000...');
```

.2.2 "wines.js": Database Management Module

```
var mongo = require('mongodb');

var Server = mongo.Server,
  Db = mongo.Db;

var server = new Server('localhost', 27017, {
  auto_reconnect: true
});
db = new Db('tuektadb', server);
// establish a connection with the database
db.open(function(err, db) {
  if (!err) {
    console.log("Connected to 'buildings' database");
    db.collection('buildings', {
      strict: true
    }, function(err, collection) {
      if (err) {
        console.log("The 'buildings' collection doesn't exist. Creating it
           with sample data...");
        populateDB();
      }
    });
  } else {
    console.log("Cannot connect to tuektadb database");
  }
});


// fetching records
exports.findById = function(req, res) {
  var id = req.params.id;
  console.log('Retrieving building description: ' + id);
  db.collection('buildings', function(err, collection) {
    collection.findOne({
      'object_id': id
    }, function(err, item) {
      res.send(item);
    });
  });
};


// updating records
exports.updateBuildingDescription = function(req, res) {
```

```
  var id = req.params.id;
  var building = req.body;
  console.log(req.params);
  console.log('Updating building: ' + id);
  console.log(JSON.stringify(building));
  db.collection('buildings', function(err, collection) {
    if (err) {
      console.log('Error while opening collection buildings: ' + err)
    } else {
      collection.findOne({
        'object_id': id
      }, function(err, item) {
        item.description = building.description;
        item.image_url = building.image_url;
        collection.update({
          'object_id': id
        }, item, {
          safe: true
        }, function(err, result) {
          if (err) {
            console.log('Error updating building: ' + err);
            res.send({
              'error': 'An error has occurred'
            });
          } else {
            console.log('' + result + ' document(s) updated');
            res.send(building);
          }
        });
      });

    }
  });
}

/*
    ----------------------------------------------------------------------------------------------------
    */
// Populate database with sample data -- Only used once: the first time the application is started.
// this is usable only if the application is deployed in a new machine and there is a need for a new database.
var populateDB = function() {
  var fs = require('fs');
  var obj,ds;
  var files = ['data/kara_arch.geojson','data/kara_build.geojson','data/tuek_build.geojson','data/indoor.json'];
  for (var i = 0; i < files.length; i++) {
  ds=files[i];

  fs.readFile(ds, 'utf8', function(err, data) {
    if (err) throw err;
    obj = JSON.parse(data);
    var buildings = []
    for (var index = 0; index < obj.features.length; index++) {
      item = obj.features[index];
      object = {
        object_id: item.properties.id,
        description: item.properties.ClassCode,
        image_url: 'data/tuekta/house.jpg'
      }
      buildings[index] = object
    }
```

```
    db.collection('buildings', function(err, collection) {
      collection.insert(buildings, {
        safe: true
      }, function(err, result) {
        if (err) {
          console.log("something is wrong man!!");
        }
      });
    });

  });
}
};
```

.3 Client side Application

.3.1 "main.js": The core program of the client side application

```
Cesium.BingMapsApi.defaultKey = 'AovQCWpnauHAxG1dQkX69IJyrXUHqCWGdG1xafecdti7Trv9aAn49GABW4umeIqJ';
var viewer = new Cesium.Viewer('CesiumContainer', {
  selectionIndicator: false,
  baseLayerPicker: false,
  fullscreenButton: false,
  homeButton: false,
  sceneModePicker: false,
  timeline: false,
  animation: false

});
//  viewer.extend(Cesium.viewerCesiumInspectorMixin);

viewer.extend(Cesium.viewerCesiumNavigationMixin, {});
viewer.infoBox.frame.sandbox = 'allow-same-origin allow-top-navigation allow-pointer-lock allow-popups allow-forms allow-scripts';
viewer.infoBox.frame.removeAttribute('sandbox');
viewer.scene.globe.depthTestAgainstTerrain = true;
viewer.scene.frameState.creditDisplay.addDefaultCredit(new Cesium.Credit('CesiumContainer', 'assets/images/tud-128x62.png ', 'https://tu-dresden.de/bu/umwelt/geo/ifk'));
viewer.scene.frameState.creditDisplay.addDefaultCredit(new Cesium.Credit('CesiumContainer', 'assets/images/geo.png', 'http://www.geoeyefoundation.org/'));

var scene = viewer.scene;
var camera = scene.camera;
var handler = new Cesium.ScreenSpaceEventHandler(scene.canvas);
var logoUrl = 'data/billboards/goat.jpg';
var model_im, other_im, current, labelEntity, current_mode, IkonosIsOn, BuildingsIsON, uchIsOn;
IkonosIsOn = BuildingsIsON = uchIsOn = false;
var layers = viewer.scene.imageryLayers;
var topo_image = new Cesium.ImageryLayer(new Cesium.SingleTileImageryProvider({
  url: 'data/imagery/topo_map.jpg',
  rectangle: Cesium.Rectangle.fromDegrees(85.6209426986409312, 50.3921250959869140, 86.1110566586886108, 50.9892180403075112)
}));
var t = viewer.entities.add(new Cesium.Entity());
var k = viewer.entities.add(new Cesium.Entity());
var tuekta_terrainProvider = new Cesium.CesiumTerrainProvider({
  url: ' http://207.154.237.111:8888/tilesets/tiles/'
});
```

```
var tuekta_cent = Cesium.Cartesian3.fromDegrees(85.88383177000060, 50.83765943000035);
var tuekta_hpr = new Cesium.HeadingPitchRoll(Cesium.Math.toRadians(-0.90), 0.0, Cesium.Math.toRadians(0));
var tuekta_orientation = Cesium.Transforms.headingPitchRollQuaternion(tuekta_cent, tuekta_hpr);
var tuekta_image = new Cesium.ImageryLayer(new Cesium.SingleTileImageryProvider({
  url: 'data/imagery/tuekta_ikonos.jpg',
  rectangle: Cesium.Rectangle.fromDegrees(85.8460710132402056, 50.8080086113231388, 85.9226210132402031, 50.8511286113231407)
}));
var karakol_terrainProvider = new Cesium.CesiumTerrainProvider({
  url: ' http://207.154.237.111:8000/tilesets/tiles/'
});

var karakol_image = new Cesium.ImageryLayer(new Cesium.SingleTileImageryProvider({
  url: 'data/imagery/karakol_ikonos.jpg',
  rectangle: Cesium.Rectangle.fromDegrees(85.9238437666018200, 50.7927500947573805, 86.0044637666018303, 50.8442500947573777)
}));
var karakol_cent = Cesium.Cartesian3.fromDegrees(85.95298, 50.81486, 884.4);
var karakol_hpr = new Cesium.HeadingPitchRoll(Cesium.Math.toRadians(-0.76), 0.0, Cesium.Math.toRadians(0));
var karakol_orientation = Cesium.Transforms.headingPitchRollQuaternion(karakol_cent, karakol_hpr);
var bing_source = new Cesium.ImageryLayer(new Cesium.BingMapsImageryProvider({
  url: 'https://dev.virtualearth.net'
}));

var osm_source = new Cesium.ImageryLayer(Cesium.createOpenStreetMapImageryProvider({
  url: 'https://a.tile.openstreetmap.org/'
}));

// "Uch Enmek" frames and outline
var outline = new Cesium.GeoJsonDataSource();
var park = new Cesium.GeoJsonDataSource();

outline.load('data/GIS/main_frame.json', {
  fill: Cesium.Color.BLUE.withAlpha(0.3),
  clampToGround: true
});
park.load('data/GIS/uch_enmek.json', {
  fill: Cesium.Color.WHITE.withAlpha(0.4),
  clampToGround: true
});

var gisLayers = ['manMade', 'natural', 'arc'];

var laylay = [{
    name: 'kara_archeo',
    path: 'data/GIS/kara_archeo.geojson',
    alpha: 0.7,
    fill: '#f970f5'
  }, {
    name: 'tue_archeo',
    path: 'data/GIS/tuek_archeo.geojson',
    alpha: 0.7,
    fill: '#f970f5'
  }, {
    name: 'tuek_muni',
```

```
    path: 'data/GIS/tuek_muni.json',
    alpha: 0.7,
    fill: '#fcb05f'
  }, {
    name: 'kara_muni',
    path: 'data/GIS/kara_muni.json',
    alpha: 0.7,
    fill: '#fcb05f'
  }, {
    name: 'tue_build',
    path: 'data/GIS/tuek_buildings.geojson',
    alpha: 0.7,
    fill: '#ff0000'
  }, {
    name: 'kara_green',
    path: 'data/GIS/kara_green.json',
    alpha: 0.7,
    fill: '#cbf7bb'
  }, {
    name: 'tue_green',
    path: 'data/GIS/tuek_green.json',
    alpha: 0.7,
    fill: '#cbf7bb'
  },
  {
    name: 'kara_build',
    path: 'data/GIS/kara_buildings.geojson',
    alpha: 0.7,
    fill: '#ff0000'
  }, {
    name: 'kara_forest',
    path: 'data/GIS/kara_forest.geojson',
    alpha: 0.7,
    fill: '#2c9107'
  }, {
    name: 'tue_forest',
    path: 'data/GIS/tuek_forest.geojson',
    alpha: 0.7,
    fill: '#2c9107'
  }, {
    name: 'kara_road',
    path: 'data/GIS/kara_road.geojson',
    alpha: 0.7,
    fill: '#858687'
  }, {
    name: 'tue_road',
    path: 'data/GIS/tuek_road.geojson',
    alpha: 0.7,
    fill: '#858687'
  }, {
    name: 'kara_river',
    path: 'data/GIS/kara_river.geojson',
    alpha: 0.7,
    fill: '#3077e8'
  }, {
    name: 'tue_river',
    path: 'data/GIS/tuek_river.geojson',
    alpha: 0.7,
    fill: '#3077e8'
  },
  {
    name: 'kara_stone',
    path: 'data/GIS/kara_stone.geojson',
```

```
    alpha: 0.7,
    fill: '#c2c3c4'
  }, {
    name: 'tue_stone',
    path: 'data/GIS/tuek_stone.geojson',
    alpha: 0.7,
    fill: '#c2c3c4'
  }, {
    name: 'kara_stone_2',
    path: 'data/GIS/kara_stone_2.geojson',
    alpha: 0.7,
    fill: '#6e9b74'
  }
];

// adding the GIS layers to Cesuim GeoJsonDataSource collection
laylay.map(function (laylay) {
  this[laylay.name] = new Cesium.GeoJsonDataSource();
  this[laylay.name].load(laylay.path, {
    fill: Cesium.Color.fromCssColorString(laylay.fill).withAlpha(laylay.alpha),
    clampToGround: true
  });
  return laylay.name;
});


var karakol = {
  camera: {
    destination: Cesium.Cartesian3.fromDegrees(85.95115667356953, 50.78042115209972, 3079.575032013868),
    orientation: {
      heading: 0.0,
      pitch: Cesium.Math.toRadians(-35.0),
      roll: 0.0
    }
  },
  other_image: tuekta_image,
  other_model: tuekta,
  other: t,
  parent: k,
  cent: karakol_cent,
  orientation: karakol_orientation,
  terrainProvider: karakol_terrainProvider,
  image: karakol_image,
  scale: 1,
  uri: 'data/3d/kara.gltf',
  manMade: [kara_road, kara_build, kara_muni],
  natural: [kara_stone, kara_river, kara_green, kara_forest, kara_stone_2],
  arc: [kara_archeo]

};


var tuekta = {
  camera: {
    destination: Cesium.Cartesian3.fromDegrees(85.88276761171318, 50.80612402131404, 2787.550547062463),
    orientation: {
      heading: 0.0,
      pitch: Cesium.Math.toRadians(-35.0),
      roll: 0.0
    }
```

```
  },
  other_image: karakol_image,
  other: k,
  other_model: karakol,
  parent: t,
  cent: tuekta_cent,
  orientation: tuekta_orientation,
  terrainProvider: tuekta_terrainProvider,
  image: tuekta_image,
  scale: 100,
  uri: 'data/3d/tuekta.gltf',
  manMade: [tue_build, tue_road, tuek_muni],
  natural: [tue_stone, tue_river, tue_green, tue_forest],
  arc: [tue_archeo]

};


var bing = {
  source: bing_source,
  other: osm_source,


};
var osm = {
  source: osm_source,
  other: bing_source
};


//karakol Kurgans billboards
viewer.entities.add({

  parent: k,
  position: Cesium.Cartesian3.fromDegrees(85.935161405039807,
      50.819749536016246, 950),
  billboard: {
    image: logoUrl,
    scaleByDistance: new Cesium.NearFarScalar(1.5e2, 2.0, 1.5e7, 0.5),
    scale: 0.03
  }
});
viewer.entities.add({
  parent: k,
  position: Cesium.Cartesian3.fromDegrees(85.952872864512827,
      50.817814523725225, 920),
  billboard: {
    image: logoUrl,
    scaleByDistance: new Cesium.NearFarScalar(1.5e2, 2.0, 1.5e7, 0.5),
    scale: 0.03
  }
});
// "Karakol" and "Tuekta" lables
viewer.entities.add({
  position: Cesium.Cartesian3.fromDegrees(85.94, 50.81, 1000),
  label: {
    text: 'KARAKOL',
    translucencyByDistance: new Cesium.NearFarScalar(15e3, 0, 18e3, 1.0),
    fillColor: Cesium.Color.fromCssColorString('#fff7f7'),
    font: '20px sans-serif'
  }
});
viewer.entities.add({
```

```
  position: Cesium.Cartesian3.fromDegrees(85.88, 50.84, 1000),
  label: {
    text: 'TUEKTA',
    translucencyByDistance: new Cesium.NearFarScalar(15e3, 0, 18e3, 1.0),
    fillColor: Cesium.Color.fromCssColorString('#fff7f7'),
    font: '20px sans-serif'
  }
});

// uch Enmek visit
function uch() {

  if (!uchIsOn) {
    viewer.dataSources.removeAll();
    viewer.dataSources.add(outline);
    viewer.dataSources.add(park);
  }
  var topo_camera = {
    destination: {
      x: 298421.54756878485,
      y: 4102622.049784298,
      z: 4985542.956093062
    }
  };
  moding(topo_image);
  viewer.camera.flyTo(topo_camera);
  uchIsOn = true;
  current = uchIsOn;
  return uchIsOn;
}



function flyto(model) {
  viewer.camera.flyTo(model.camera);
}


// adding the site datasets without shiwing them
function render(threeD) {

  viewer.entities.add({
    name: 'Loading......',
    parent: threeD.parent,
    orientation: threeD.orientation,
    position: threeD.cent,
    model: {
      scale: threeD.scale,
      uri: threeD.uri
    }
  });
  layers.add(threeD.image);
  threeD.image.show = false;
  threeD.parent.show = false;
  current = threeD;
}

//initializing the site visit
function launch(model) {

  uchIsOn = false;
  if (IkonosIsOn) {
    ikonos(model);
```

```
  }
  GIScheck(model);
  if (BuildingsIsON) {
    create(model);
  }

  flyto(model);
}


// adding imagery of the current site  'if swithced' and remove the prevouis
    one
function moding(mode) {

  if (current_mode != mode) {
    if (current_mode = topo_image) {
      layers.remove(topo_image, false);
    }
    if (!mode.source || !mode.other) {
      layers.add(topo_image);
    } else {
      layers.remove(mode.other, false);
      layers.add(mode.source);
    }
    if (IkonosIsOn) {
      ikonos(current);
    }
    current_mode = mode;
    return current_mode;
  }
}


function TerrainBuild(model) {

  if (model != current) {
    viewer.dataSources.removeAll();
  }
  viewer.terrainProvider = model.terrainProvider;
}



function GIScheck(model) {

  if (current != model) {
    viewer.dataSources.removeAll();
    current = model;
    gisLayers.map(function (layers) {
      draw(layers);
    })


  }
  return current;
}


function ikonos(model) {

  if (!uchIsOn) {
    if (model != current) {
      model.other_image.show = false;
```

```
    }
    model.image.show = true;
    layers.raiseToTop(model.image);
    IkonosIsOn = true;
    return IkonosIsOn;
  }
  return IkonosIsOn;
}

//3D model visualization
function create(model) {

  model.parent.show = true;
  model.other.show = false;
  TerrainBuild(model);
  BuildingsIsON = true;
  return BuildingsIsON;
}

// sending the request using the picked object id
function get_description(id, onsuccess) {

  var url = 'http://207.154.237.111:3000/buildings/' + id;
  var query = {};
  var type = 'get';
  var content_type = 'application/json; charset=utf-8';
  var data = JSON.stringify(query);

  jQuery.ajax({
    url: url,
    type: type,
    contentType: content_type,
    data: data,
    success: function (response) {
      // response
      onsuccess(response.description, response.image_url);
    },
    error: function (data) {
      // error

    }
  });
}

//  label shows coordinates of the picked object
labelEntity = viewer.entities.add({
  label: {
    show: false,
    showBackground: true,
    font: '14px monospace',
    horizontalOrigin: Cesium.HorizontalOrigin.LEFT,
    verticalOrigin: Cesium.VerticalOrigin.TOP,
    pixelOffset: new Cesium.Cartesian2(15, 0)
  }
});

// picking
handler.setInputAction(function (movement) {
  // var foundPosition = false;
  var cartesian = scene.pickPosition(movement.position);
  var pickedObject = scene.pick(movement.position);

  if (Cesium.defined(pickedObject)) {
```

```
    closeNav();
    if (Cesium.defined(pickedObject.node) && Cesium.defined(pickedObject.mesh)
        ) {

      var id = pickedObject.node.name;
      get_description(id, function (description, image_url) {

        var entity_1 = new Cesium.Entity({
          name: 'THIS IS Object NO ' + '"' + id + '"'
        });
        var des = description;

        //setting the infobox content
        entity_1.description = `
     <div>
     <img
     width="50%"
     style="margin: 0 1em 1em 0;"
     src="${image_url}"/>
     <div>
     Description
     </div>
     <div>
     <p>
     ${des}
     </p>
     </div>
     <div>
     <a href="update.html?id=${id}" target=_blank>Update Description</a>
     </div>
     </div>
     `;
        viewer.selectedEntity = entity_1;

      });

      if (Cesium.defined(cartesian)) {
        var cartographic = Cesium.Cartographic.fromCartesian(cartesian);
        var longitudeString = Cesium.Math.toDegrees(cartographic.longitude).
            toFixed(4);
        var latitudeString = Cesium.Math.toDegrees(cartographic.latitude).
            toFixed(4);
        var heightString = cartographic.height.toFixed(3);

        labelEntity.position = cartesian;
        labelEntity.label.show = true;
        labelEntity.label.text =
          'Lon: ' + ('   ' + longitudeString).slice(-7) + '\u00B0' +
          '\nLat: ' + ('   ' + latitudeString).slice(-7) + '\u00B0' +
          '\nAlt: ' + ('   ' + heightString).slice(-7) + 'm';

        labelEntity.label.eyeOffset = new Cesium.Cartesian3(0.0, 0.0, camera.
            frustum.near * 1.5 - Cesium.Cartesian3.distance(cartesian, camera.
            position));

        // foundPosition = true;
      }

      // TODO: highlight material on event
      // var primitive = pickedObject.primitive;
      // var meshh = pickedObject.mesh._materials;
      // var r =[];
      // var newOne =  primitive.getMaterial('Material.18');
```

```
      // r.push(newOne);
      // r=meshh;
      // console.log(meshh);
      // console.log(r);

    }

  } else {
    // foundPosition = false;
    labelEntity.label.show = false;
  }
}, Cesium.ScreenSpaceEventType.LEFT_CLICK);
```

.3.2 "ui.js": The user interface code

```
var jj = false;
var mq = window.matchMedia("(min-width: 765px)");

$(document).ready(function () {
  jj = true;
  tog();
  render(tuekta);
  render(karakol);
  uch();

});


function toggle() {
  if (jj) {
    labelEntity.label.show = false;
    viewer.selectedEntity = undefined;
    tog();
  }
}


function tog() {
  if (open) {
    closeNav();
  } else {
    openNav();
  }
}

function openNav() {
  document.getElementById('mySidenav').style.width = '300px';
  open = true;
}

function closeNav() {
  document.getElementById('mySidenav').style.width = '0';
  open = false;
}



function draw(T_layer) {
```

```
  if (!uchIsOn) {
    var lay = current[T_layer];
    var leg = document.getElementById(T_layer).checked;
    show()
    for (var i = 0; i < lay.length; i++) {
      if (leg) {

        viewer.dataSources.add(lay[i]);
      } else {
        viewer.dataSources.remove(lay[i]);
      }
    }
  }
}



function IkonosCheck() {

  if (document.getElementById('ikonos').checked) {

    ikonos(current);
    IkonosIsOn = true;
  } else {
    if (!uchIsOn) {
      current.other_image.show = false;
      current.image.show = false;
    }
    IkonosIsOn = false;
  }
  return IkonosIsOn;
}

function buildingcheck() {
  if (document.getElementById('Buildings').checked) {
    if (!uchIsOn) {
      create(current);
    }
    BuildingsIsON = true;
  } else {
    t.show = false;
    k.show = false;
    if (BuildingsIsON) {
      viewer.terrainProvider = new Cesium.EllipsoidTerrainProvider();
    }

    BuildingsIsON = false;
  }
  return BuildingsIsON;
}

// initialize indoor mode
function grave() {
  document.getElementById('kurgans').style.visibility = 'visible';
  toggle();
  jj = false;
  viewer.dataSources.remove(outline);
  viewer.dataSources.remove(park);
  uchIsOn = false;
  launch(karakol);
  create(karakol);
  hideDefaultBar();
  kurgan();
```

```
  viewer.terrainProvider = new Cesium.EllipsoidTerrainProvider();

}
// exit indoor mode
function back() {
  labelEntity.label.show = false;
  viewer.navigationHelpButton.container.hidden = false;
  viewer.selectedEntity = undefined;
  document.getElementById('kurgans').style.visibility = 'hidden';
  reset();
  buildingcheck();
  jj = true;
}



// legend show/ hide

function show() {

  if (document.getElementById('manMade').checked
  || document.getElementById('arc').checked
  || document.getElementById('natural').checked) {
    if (mq.matches) {
      // window width is at least 500px
      document.getElementById('my-legend').style.left = '78%';
    } else {
      // window width is less than 500px
      document.getElementById('my-legend').style.left = '55%';
    }


  } else {
    document.getElementById('my-legend').style.left = '100%';

  }


}
document.getElementById('osm').addEventListener('click', function () {
  moding(osm);
}, false);

document.getElementById('bing').addEventListener('click', function () {
  moding(bing);
}, false);

document.getElementById('topo').addEventListener('click', function () {
  moding(topo_image);
}, false);

document.getElementById('uch').addEventListener('click', function () {
  document.getElementById('Buildings').checked = false;

  buildingcheck();
  uch();
}, false);

document.getElementById('tuekta').addEventListener('click', function () {
  viewer.dataSources.remove(outline);
  viewer.dataSources.remove(park);
  uchIsOn = false;
  launch(tuekta);
```

```
170 }, false);
171
172 document.getElementById('karakol').addEventListener('click', function () {
173   viewer.dataSources.remove(outline);
174   viewer.dataSources.remove(park);
175   uchIsOn = false;
176   launch(karakol);
177 }, false);
178
179 document.getElementById('grave').addEventListener('click', function () {
180   grave();
181
182 }, false);
```

.3.3 "update.js": The Update Page Interface

```
1  function update_article(id) {
2
3    var description = $('#desc').val();
4    var image = $('#basic-url').val();
5    var url = 'http://207.154.237.111:3000/buildings/' + id;
6    var query = {
7      'description': description,
8      'image_url': image
9    };
10   var type = 'put';
11   var content_type = 'application/json; charset=utf-8';
12   var data = JSON.stringify(query);
13
14   jQuery.ajax({
15     url: url,
16
17     type: type,
18     contentType: content_type,
19     data: data,
20     success: function(response) {
21       // reponse
22
23       $('#myModal').modal('toggle');
24
25
26     },
27
28     error: function(data) {
29       // error
30
31     }
32   });
33 }
34
35 // sending the request using the picked object id, this will give back the
     description from MongoDB
36 function get_description(id, onsuccess) {
37
38   var url = 'http://207.154.237.111:3000/buildings/' + id;
39   var query = {};
40   var type = 'get';
41   var content_type = 'application/json; charset=utf-8';
```

```
  var data = JSON.stringify(query);

  jQuery.ajax({
    url: url,
    type: type,
    contentType: content_type,
    data: data,
    success: function(response) {
      // response

      onsuccess(response.description, response.image_url);

    },
    error: function(data) {
      // error

    }
  });
}
// A $( document ).ready() block.
$(document).ready(function() {
  var id = location.search.split('id=')[1];

  get_description(id, function(description, image_url) {
    $('#desc').val(description);
    $('#basic-url').val(image_url);

  });
  $('#update').click(function() {
    update_article(id);

  });


});
```

.3.4 "indoor.js": The indoor navigation mode

```
var canvas = viewer.canvas;
var ellipsoid = scene.globe.ellipsoid;
var controller = scene.screenSpaceCameraController;
var startMousePosition;
var mousePosition;
var looking = false;
var moveForward = false;
var moveBackward = false;

var handler = new Cesium.ScreenSpaceEventHandler(canvas);

function update(value) {
  value = 0;
  return value;
}
handler.setInputAction(function(movement) {
  looking = true;
  mousePosition = startMousePosition = Cesium.Cartesian3.clone(movement.
      position);
}, Cesium.ScreenSpaceEventType.LEFT_DOWN);
```

```
20
21 handler.setInputAction(function(movement) {
22   mousePosition = movement.endPosition;
23 }, Cesium.ScreenSpaceEventType.MOUSE_MOVE);
24
25 handler.setInputAction(function(position) {
26   looking = false;
27 }, Cesium.ScreenSpaceEventType.LEFT_UP);
28
29 // TODO: wheel zoom indoor
30 // var gg=false;
31 // var ff =false;
32 //     handler.setInputAction(function(delta,g) {
33 //
34 //       if (delta < 0){
35 //          gg=g;
36 //
37 //       }else if (delta > 0){
38 //          ff=g;
39 //
40 //       }
41 //
42 //
43 //
44 //      }, Cesium.ScreenSpaceEventType.WHEEL);
45
46
47 function setKey(event, isOn) {
48
49   if (event.keyCode == 38) { // Up.
50     moveForward = isOn;
51
52   } else if (event.keyCode == 40) { // Down.
53     moveBackward = isOn;
54
55   } else {
56     //for other movement setters
57     // return true;
58   }
59
60
61 }
62
63 function keyDown(event) {
64
65   return setKey(event, true);
66 }
67
68 function keyUp(event) {
69   return setKey(event, false);
70
71 }
72 var d = function indoor(colck) {
73
74
75   if (looking) {
76     var width = canvas.clientWidth;
77     var height = canvas.clientHeight;
78
79     // Coordinate (0.0, 0.0) will be where the mouse was clicked.
80     var x = (mousePosition.x - startMousePosition.x) / width;
81     var y = -(mousePosition.y - startMousePosition.y) / height;
82
```

```
    var lookFactor = 0.05;
    camera.lookRight(x * lookFactor);
    camera.lookUp(y * lookFactor);
  }

  // Change movement speed based on the distance of the camera to the surface of the ellipsoid.
  var cameraHeight = ellipsoid.cartesianToCartographic(camera.position).height;
  var moveRate = cameraHeight / 10000.0;

  if (moveForward) {
    camera.moveForward(moveRate);
  }
  if (moveBackward) {
    camera.moveBackward(moveRate);

  }
};

function hideDefaultBar() {
  viewer.navigationHelpButton.container.hidden = true;

}


function kurgan() {

  camera.flyTo({
    destination: Cesium.Cartesian3.fromDegrees(85.95286679336539, 50.81761313448716, 887),
    orientation: {
      heading: 0.0,
      pitch: 0.0,
      roll: 0.0
    }
  });
    // disable the default event handlers
  navigate(controller, false);
}


function balance() {
  camera.flyTo({
    destination: viewer.camera.position,
    orientation: {
      heading: viewer.camera.heading,
      pitch: 0.0,
      roll: 0.0,
    }
  });

}



function reset() {
  navigate(controller, true);
  launch(current);
}
```

```
143 function navigate(options, set) {
144   options.enableRotate = set;
145   options.enableTranslate = set;
146   options.enableZoom = set;
147   options.enableTilt = set;
148   options.enableTilt = set;
149   if (set) {
150     viewer.clock.onTick.removeEventListener(d);
151   } else {
152     viewer.clock.onTick.addEventListener(d);
153   }
154
155 }
```