Università
della
Svizzera
italiana

Faculty
of Informatics

Bachelor Thesis

June 7, 2017

Smart–IVC

Enhanced Visualization of Cities, Through Smart Visual Queries

Andrea Vicari

Abstract

Web users have to aggregate different data from various sources whenever they are looking for some information on the Internet. Think about the student who is looking for a rented room near her university: she firstly uses a specific website to find the advertisement of a room for rent, she then looks for the address on another website that provides a map service to see if the house is located where she desires.

No such service exists that provides a unique environment in which the user can both visualize a city and interact with its elements. The only technologies available are either not exhaustive or too complex to use.

This thesis introduces Smart–IVC a web application that provides an intuitive interface and prevents the user from jumping from website to website. Through the form of a 3D-environment, this application provides an interactive visualisation of cities in which the user can directly âĂIJcommunicateâĂİ with the elements, executing queries on them. After having clicked on a building in the map, the user is able to get information (coordinates, address, floors etc.) about that construction and also find out what are the various relations between that specific building and the other entities in the city.

Smart–IVC is an application accessible by everyone and that aims to enhance the city visualisation, getting closer to the user needs.

Advisor
Prof. Michele Lanza
Assistant
Prof. Dr. Andrea Mocci

Advisor's approval (Prof. Michele Lanza): Date:

Contents

List of Figures

1 Introduction

Cities nowadays are in constant evolution. They change their structure overtime through the appearance of new elements and the disappearance of old ones, thus they are very unstable and prone to changes. Important changes, might be considered with regard to the deployment and management of all types of infrastructures within cities.

Moreover, cities have a very large impact on the economic and social development of nations: they represent the real foundation where people live, where companies have their business and in which numerous services are provided. Often, a city overview is not available, so decision may be taken without having a big picture of the surrounding environment. This could lead to choices that might have a negative impact on the current city, reducing the efficiency and the life quality of the citizens.

To control these changes, a visualization, which provides a city overview, is required. Such a visualization is considered to be the first step towards what it is called a Smart City. A Smart City can be defined as a city which uses information and communication technologies so that its critical infrastructure as well as its components and public services provided are more interactive, efficient and so that citizens can be made more aware of them.

1.1 Contributions

Smart–IVC aims to solve the problem generated by the static nature of data regarding cities (as mentioned above) through an Interactive 3D–Visualization model. Therefore, the main contributions that Smart–IVC provides are:

- The visualization of a 3D city model. As a use case for this Bachelor Project, the city of Lugano has been taken into account.
- The interactivity that the user has with the entities inside the city: buildings can be clicked in order to receive information about them and in order to receive information between the selected building itself and the rest of the entities in the city.
- The possibility to have a graphical city overview using a very intuitive system of visual queries which produce results that are immediately visible through the highlighting or colouring of the buildings.

1.2 Structure of the report

- **Chapter 2** is about the state of the art. It talks about already existing tools which allow city visualization in a virtual environment. In particular, it focuses on tools that use the Cesium framework comparing and contrasting the existing features with the ones proposed in Smart–IVC.
- **Chapter 3** is the main chapter of the entire report. It shows how Smart–IVC has been developed step by step. The decisions that have been taken and the technologies adopted will be explained and discussed in detail starting from the parsing of the unique .xml file provided by the Comune of Lugano and ending with the final result (i.e., the 3D–visualization of the entire city).
- **Chapter 4** is about showing some use cases of the application. Examples will be illustrated and explained in detail, in order to show the features proposed by Smart–IVC.
- **Chapter 5** is the conclusion of this report. Here, the current limitations of the application will be presented. Finally, the future work and the different paths that Smart–IVC could take will be discussed and analyzed.

2 State of the Art

As explained above, city visualization is a subject undergoing intense study, therefore there can found new projects about it everyday, on the Internet. Still, the uniqueness of Smart–IVC resides on the fact that it provides, not only a visualization of a city but, above all, it gives the user the possibility to interact with the city itself through a simple system of visual queries.

Before getting into the details with Smart–IVC, the current state of city visualization must be clarified and, since the projects about this topic are so various and so many, in this Chapter, the ones that relate most with the final outcome of this Bachelor Project will be presented.

At first, some 3D–city–model based on OpenStreetMap data will be presented, as **OSMBuildings**, **F4 Map** and **ViziCities**. Then, the **Cesium framework** will be introduced, since it has been used used as a basis for Smart–IVC. At the end, the work done by the **Swiss Geospatial Portal** (still using Cesium), will be analyzed and compared with the application described in this report.

2.1 OpenStreetMap–Based Frameworks

OpenStreetMap, is a project that creates and distributes free geographic data for the world. In the last period, the third dimension has become a growing topic at OSM, so it is now possible to add detailed buildings and a lot of minor objects. Here, some of the frameworks that use this data, will be analyzed.

2.1.1 OSMBuildings

OSMBuildings is an engine for displaying 3D buildings on a web map. It uses information on buildings provided by OpenStreetMap and it renders them on a map layer.
Altough in some cases, buildings are very detailed (i.e., in big cities like New York or Berlin), it does only provide a visualization without any kind of interaction.

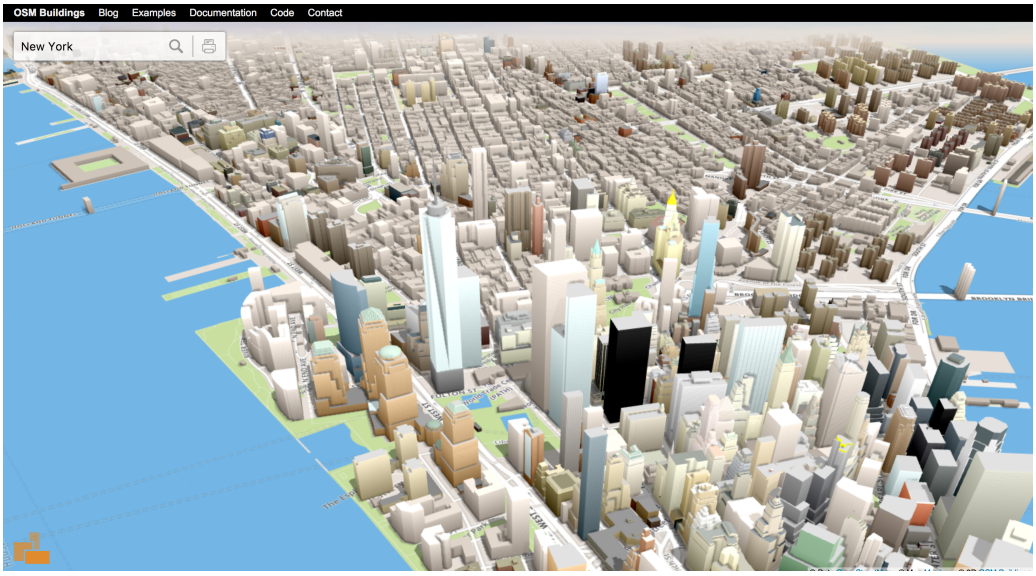


Figure 2.1. A visualization of New York city using OSMBuildings

2.1.2 F4 Map

The F4 Map is an OSM-based 3D map using the WebGL technology. Also this map, uses Open Street Map's buildings but also adds some non–OSM–provided features like trees, cranes and other data.
Some 3D models are used (not based on buildings data in OpenStreetMap) for some specific buildings (e.g. the Eiffel Tower in the example below).

Figure 2.2. A visualization of Paris using F4 Map

Unfortunately, the F4 group does not provide any kind of documentation and this project is not open source.

2.1.3 ViziCities

The third and last map that uses OSM information about buildings is ViziCities, a framework for 3D geospatial visualization in the browser. It's code is available on GitHub since it is an OpenSource project but they are still at version 0.3 of the application.
It let the developer add some useful information to the map like routes and GeoJSON (i.e., a format for encoding a variety of geographic data structures) but is still very limited and not very prone to interactivity.

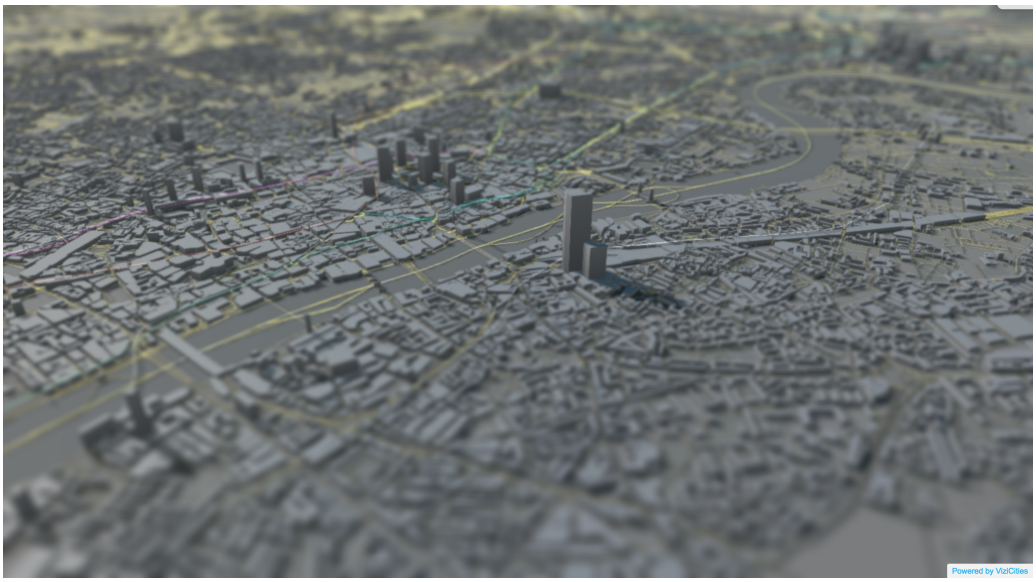


Figure 2.3. A visualization of New York city using ViziCities

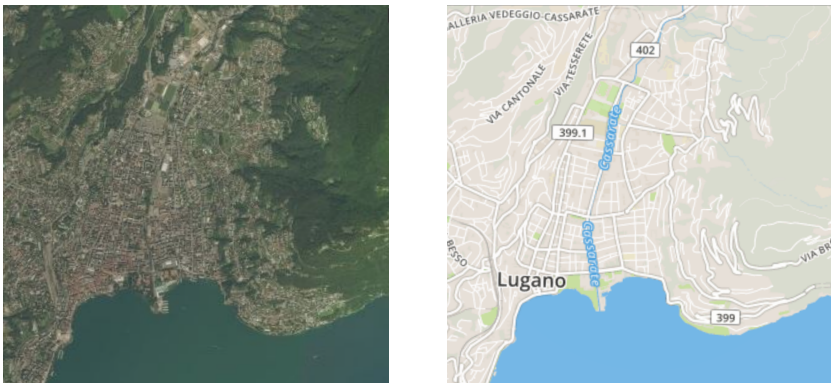
2.2 Cesium

Figure 2.4. Cesium logo

Cesium is an open-source JavaScript library for world-class 3D globes and maps that is used to create a web-based globe and map for visualizing dynamic data. This framework runs using WebGL, a JavaScript API for rendering 3D graphics within any compatible web browser that allows GPU–accelerated usage of physics and image processing and effects as part of the web page canvas.
The features that Cesium provides are the following:

- **A virtual globe:** a three-dimensional software model or representation of the Earth that provides the user with the ability to freely move around in the virtual environment by changing the viewing angle and position.

- **Different imagery providers:** the possibility to draw and layer high-resolution imagery (maps) from several standard services directly on the virtual globe.



(a) High-resolution, mesh-based terrain provided by Bing

(b) Streets basic imagery provided by Mapbox

Figure 2.5. Example of two imagery providers available on Cesium

- **Different terrain providers:** the possibility to visualize global high-resolution terrain and water effects for oceans, lakes, and rivers but mostly the possibility to represent mountain peaks, valleys, and other terrain features.

(a) The standard WGS84 Ellipsoid

(b) Terrain meshes provided by STK

Figure 2.6. Example of two terrain providers available on Cesium, this shows the benefits of a 3D globe compared to a 2D map.

- **Huge number of API provided:** since the uses of Cesium are really various, Cesium provides a very big number of API in order to control every aspect of the web application (i.e., draw every kind of geometry, handle animations, etc...).
  The official website of Cesium, also provides a complete documentation, various examples and playgrounds.

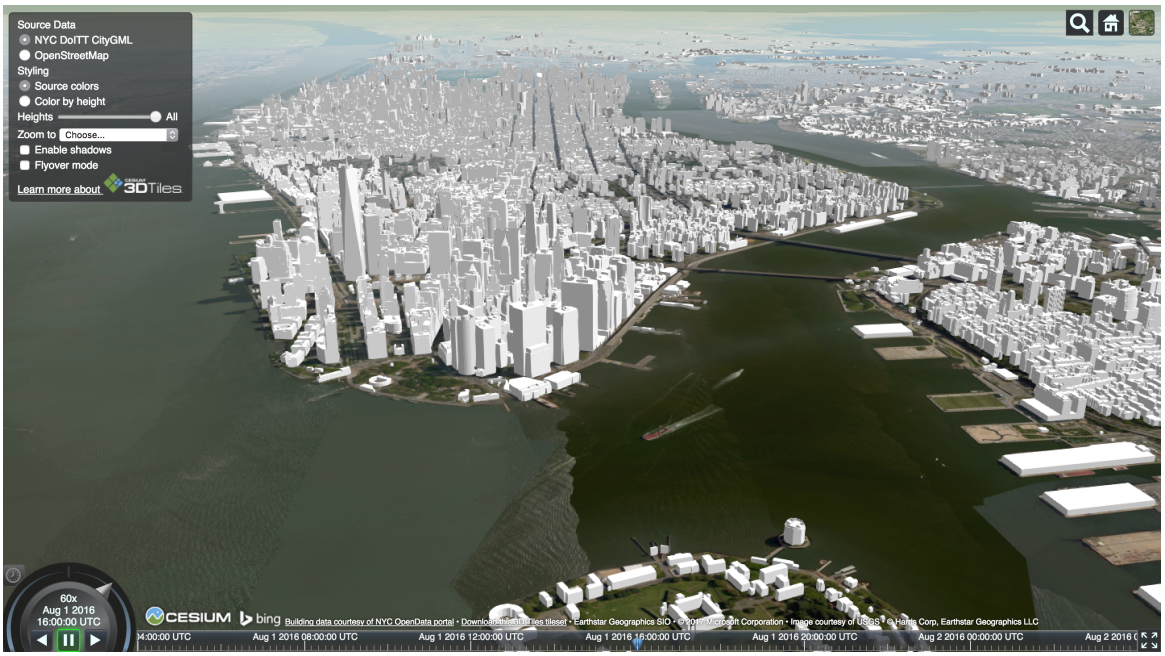


Figure 2.7. An example showing a 3D visualization of the city of New York. Using Cesium 3D Tiles Tecnhology

2.3 Swiss Geospatial Portal Using 3D Tiles

Figure 2.8. A 3D visualization of the city of Bern in the Swiss Geospatial Portal. Using Cesium 3D Tiles Tecnhology

3 Smart–IVC

3.1 Environment and Frameworks

The environments selected to develop Smart–IVC are the following:

- **The Server–side** was written using the **Java** Programming Language. The **Spring Framework** was used, in particular using its most famous convention–over–configuration, called **Spring Boot**. The database used to store the information about the city is **MySQL**.
- **The Client–side** was written using the **Javascript** scripting language. The **jQuery** library was used and, of course, **HTML5** and **CSS3**. As mentioned above, the 3D visualization of the city was achieved using the **Cesium Framework**

3.2 The Overall Structure

Figure 3.2 shows how the final prototype of Smart–IVC looks like.

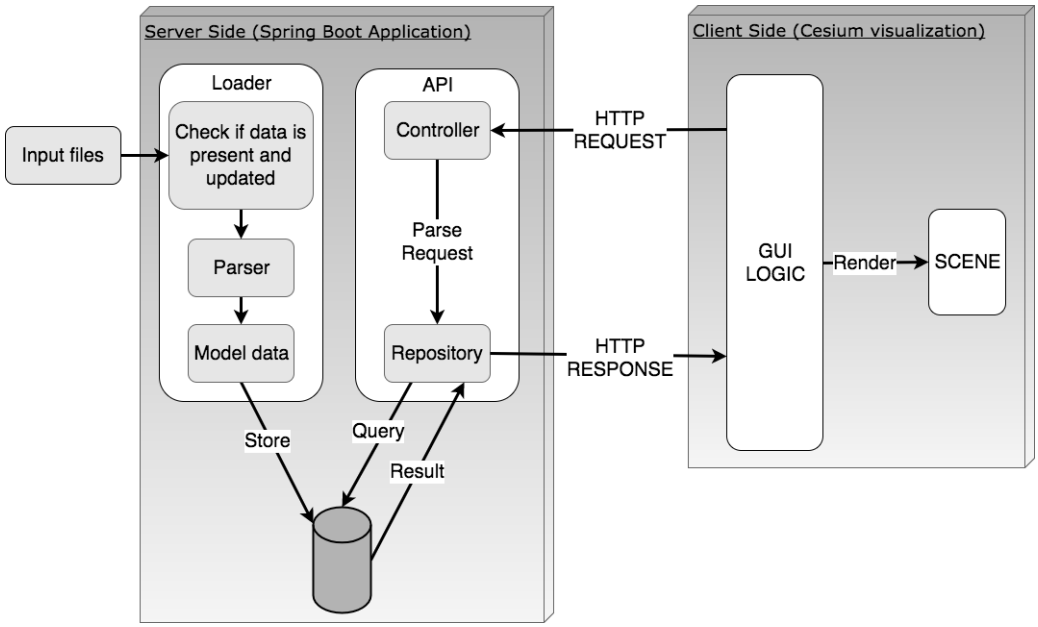


Figure 3.1. Smart–IVC architecture scheme

The upcoming sections, illustrate Smart–IVC and how it was developed. The first part proposes the Smart–IVC modelling of data. Afterwards a section about the parser and the cron–jobs created is present. Followed by a description of the API provided to the user. Finally it concludes with some discussions about how the structure and the logic behind the Web–Application was developed.

3.3 Server Side

3.3.1 Available Data and Modelling

The entire work starts from the “xml” file provided by the Comune of Lugano. The first choice to take was how to model the available data in the best way such that the way of retrieving information about any building was as fast and as consistent as possible.
The models created were the following:

- City
- Suburb
- Building
- Address
- Type

3.3.2 Parser

The data type taken as input is of xml type. It is used in the process to load buildings data.
At the beginning of the file an header which describes the content is defined. It regulates fields and types as well as values range, tolerances in the coordinates system and other set ups parameters. After the header, a long record of elements which represent buildings follows (exactly 18904). Structure and constraints, of record elements are defined in the header. The record presents several data, out of this data Smart–IVC finds use on almost all of them. Here there will be described the tags that can be found in the xml file and that were found useful for Smart–IVC

- **SHAPE:** it stores both the perimeter of the building and the max bounds that it occupies
- **Descrizione:** it represents the type of the building. Between them only the ones of type “Edificio” (i.e., building in Italian), are stored
- **Sezione:** it is a numeric value that represent the suburb in which the building is located
- **NUM_CIVICO_ID:** it is a 8–character–long numeric value that contains: the number id representing the street name in the first four characters and the civic number in the remaining four.
- **EGID_UCA:** it uniquely identifies buildings on the entire Swiss ground on the basis of the “Registro Federale degli Edifici e delle abitazioni” (REA) from the “Ufficio federale di statistica”.
- **PIANI:** the number of floors per building
- **SHAPE_AREA:** the area of the plane described by the building
- **SHAPE_LENGTH:** the length of the perimeter of the building

The data is assimilated in Smart–IVC through the loader. A parser has been created in order to read and parse a file structured in the way described above. The parser creates a building model and an address model for every record in the xml file and sets their fields using the information stored in the xml tags. Once the data is created, the models gets stored into a database.

Coordinates of buildings are stored in a data type called CH1903. It represents the Swiss projection coordinates system. It uses an Oblique Mercator on a 1841 Bessel ellipsoid. An Oblique Mercator is an oblique conformal cylinder projection. Together with the 1841 Bessel ellipsoid, a reference ellipsoid of geodesy with base at the old observatory in Bern, gives meaning to the projected coordinates. The transformation computations of the data from CH1903 to WGS84 is provided from the Swiss confederation. The precision with this transformation is of respectively 1 meter and 0.1". In order to derive the transformationa specific formulae has to be applied. The way these coordinates are converted can be found on the website of the Federal Office of Topography Swisstopo under the section NAVREF.

3.3.3 Commands

On the application start, the application checks if the data is already present in the database, if not it executes a command that reads the xml file and parses it using the parser described above and stores the models in the database. At this point, the data is not ready yet, since it does not contain all the important information required from Smart–IVC to work.
That is why some commands where create for different purposes. Again, on application start, it is checked if the data is updated properly, if not the following commands are executed:

- The **CityLoaderCommand** contains two commands to be executed: the first one, as explained above, parses the xml and stores the models in the database. The second one, adds some additional information about the buildings (i.e., number of apartments–per–building that are primary and secondary houses). This additional information was received by the Comune of Lugano in a second time in the format of a txt file. Therefore, the matching between buildings listed in the two files was done using the EGID value.
- The **ConverterCommand** is used to convert the coordinate system used for the perimeters of the buildings from CH1903 to WGS84. The conversion is done using the service provided by the Web APIs of the Federal Office of Topography Swisstopo where, given a pair of coordinates in the CH1903 system, their respective conversion in WGS84 is returned.

- The **CityInformationCommand** is used to add additional information to each building though three different services. The first one adds information about the membership suburb using an APIs service provided by Swisstopo that uses the EGID in order to get such an information. Another service provided by Swisstopo API's is used to get information about the address name and the civic number of the building, still using the EGID value. The last command uses the API service provided by OpenStreetMaps in order to get information about addresses, civic numbers (for the building that have not an EGID value stored) and type of the building (e.g., Hospital, School, University etc . . .).

Once these commands are executed, the data is ready to be used and retrieved in order to be shown on the web application.

3.3.4 Controllers

3.4 Client Side

3.4.1 The first Attempt: Babylon.js

Immediately after having gathered the essential data useful to draw the buildings of the city of Lugano, a first attempt of visualizing them was made using BabylonJS. It is a JavaScript framework for building 3D environments with HTML5 and WebGL. It allows the creation of a scene with customized lights, cameras, materials, meshes, animations, audio and actions. It also supports scene picking (i.e., an element on the scene is clickable and it is possible to interact with it).

In order to test the capabilities of BabylonJS, a small portion of Lugano (i.e., the part around the lake) was selected and rendered. The result can be seen in the following image:

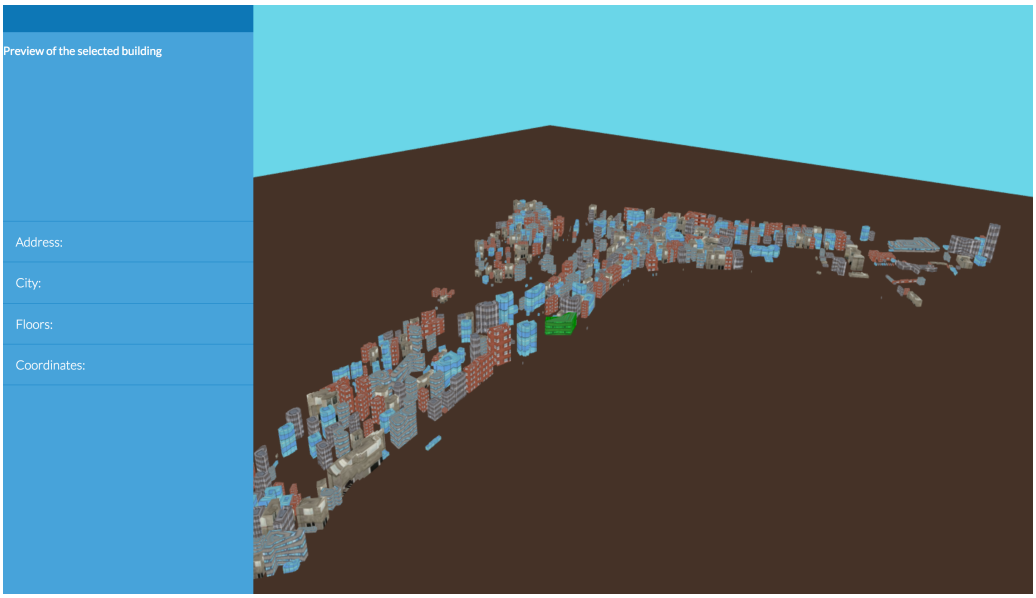


Figure 3.2. The first attempt of city visualization using BabilonJS

As long as the buildings showed in the scene where under the two–thousand, the browser was very fast in rendering during the loading of the page and the movement of the camera was smooth.
The main problems that make the idea to use BabylonJS discarded, were has been basically two:

- The buildings to be rendered were far above the mentioned threshold, the rendering would take more than 30 seconds.
- BabylonJS provides no basis to start from: the initial scene is completely empty and, as it is possible to see in the image above, buildings lay on a plane. That represented a serious problem since the visualization of the city was planned to be shown in a way that would be as realistic as possible.

A solution to the last problem would have been using the Google Elevation API. This service, given a pair of coordinates (i.e., latitude and longitude), returns the exact altitude of that point.

Unfortunately, the API system of Google provides just 2,500 free requests per day. In the case of Lugano, that extends its territory for more that $25km^2$, considering one request for meter it would have been taken more than 10 days to get the entire terrain structure (for the entire city of Rome, that spans almost $46km^2$, the days taken would have been more than 20).

Nonetheless, this would have made the rendering slower since, in addition to the visualization of the buildings, also a rendering of the terrain (i.e., lakes, mountains and rivers) would have taken place.
Therefore, this lack of both Google–API–requests and good performances, lead the idea to use BabylonJS to be discarded.

3.4.2 The final version: Cesium.js

As stated above, the Cesium Framework was finally used to build the client–side of the application.
Cesium, on the contrary, provides a ready–to–use virtual globe

3.4.3 The Side Menu and The Query Builder

3.4.4 Cesium Framework

4 Use Cases

4.1 Access information

4.2 Visualize the City

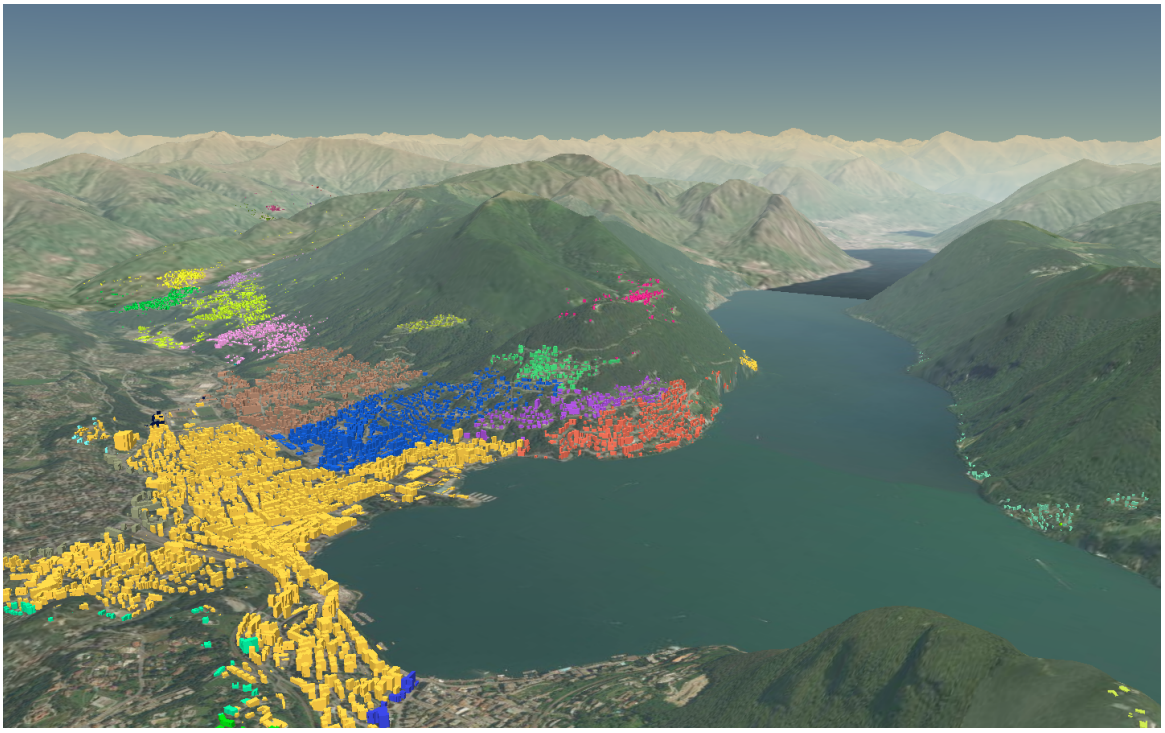


Figure 4.1. A Visualization of the city of Lugano where every suburb is coloured differentrly

4.3 The Query System

4.3.1 Building Selection

4.3.2 City Gradient Map

5 Conclusions and future work or possible developments

5.1 Summary

5.2 Future Work

References