university of groningen / faculty of mathematics and natural sciences

Master Thesis Computing Science

Computational Science & Visualization

Interactive 3D GIS Focus-Plus-Context Visualisation Using WebGL

Author:
Lukas de Boer
l.de.boer.8@student.rug.nl
s1797727

First supervisor:
prof. dr. J.B.T.M Roerdink

Second supervisor:
prof. dr. M. Biehl

External supervisor:
Drs. A. de Jong, TNO

Nederlandse Organisatie voor Toegepast-Natuurwetenschappelijk Onderzoek
Business Information Services
TNO Groningen, Eemsgolaan

Scientific Visualization and Computer Graphics Research Group
Faculty of Mathematics and Natural Sciences
University of Groningen

August 22, 2016

ABSTRACT

Many visualization tools exist for displaying geographic information system (GIS) datasets on an interactive map on the web. Based on Leaflet, TNO has created the open-source CommonSense framework which allows users to flexibly enable filters to the dataset and apply styling in order to get better insight in the data. CommonSense has been used as a basis for many visualization applications because of its flexibility and interactivity. However, Leaflet, and thus the CommonSense framework, only supports 2D top-down views which limits the flexibility of the framework significantly.

This project introduces a solution to this based on Cesium which uses WebGL in order to enable a visualization with a fully interactive 3D globe. Using this new functionality, a new visualization of a 3D point cloud dataset generated by TNO-developed risk analysis software Effects is created, based on raycasting. A focus-plus-context approach is taken by rendering nearby building models based on LIDAR data in order to give better insight in the scale of the dataset, whilst maintaining the flexibility and interactivity that CommonSense provides.

CONTENTS

LIST OF FIGURES

ACRONYMS

GIS Geographic Information System

TNO Nederlandse Organisatie voor toegepast-natuurwetenschappelijk onderzoek

LiDAR Laser Imaging Detection And Ranging

BAG Basisregistraties Adressen en Gebouwen

AHN2 Algemene Hoogtebestand Nederland 2

1 INTRODUCTION

A Geographic Information System (GIS) allows users to visualize, question, analyze, and interpret data to understand relationships, patterns, and trends [14]. GIS is the go-to technology for making better decisions about location. Common examples include real estate site selection, route/corridor selection, evacuation planning, conservation, natural resource extraction, and many more. GIS-based maps and visualizations greatly assist in understanding situations and in storytelling. However, users don't want to install a large software package in order to be able to use these information systems. Recently, web browser based applications have become very popular in many domains [30].

Many visualization tools exist for displaying GIS data on an interactive map in a browser. The Nederlandse Organisatie voor toegepast-natuurwetenschappelijk onderzoek (TNO) has created the open-source web-based CommonSense framework which allows users to flexibly filter GIS data and apply styling in order to get an interactive insight in the data [56]. This framework has been used as a basis for many visualization applications that have been made by TNO. However, due to architectural decisions, the framework only allows the user to view the GIS in 2D from a top-down perspective. This means the camera is positioned above the map and is aimed downwards, similar to a bird's-eye view. This design decision limits the possibilities of the framework in the area of three-dimensional (3D) data. As GIS becomes more and more popular, the demand for the visualization and analysis of 3D data increases. Running such computationally intensive applications in the browser brings a lot of problems with it. This project will research the possibilities of adapting the CommonSense framework to allow a full 3D visualization in the browser. This adaptation will be guided by a model of TNO that is currently only visualized in 2D, but is able to output 3D data in some form.

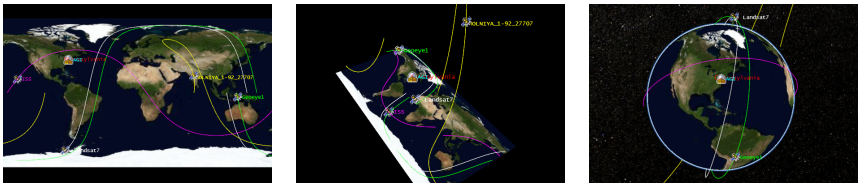
In order to get the most out of 3D GIS visualization, the user should get a sense of the context and scale of the data that is visualized. A method of achieving "focus-plus-context" is by visualizing the detail information (the data visualized) and overview information simultaneously. A possible source for overview information is the visualization of buildings that are nearby the detail data.

Interactivity is key in this project, which means the users should be able to adapt multiple aspects of the visualization, for example the

movement of the camera, color maps, filtering data and more. Users report that interactivity increases enjoyment of the usage of such geographic environments [11].

Tools

The CommonSense framework is currently based on Leaflet [64], which is an open-source Javascript library for mobile friendly interactive maps. However, Leaflet, and thus the CommonSense framework, only supports 2D top-down views, which means using Leaflet is not an option for this project. Other solutions exist, such as the open-source Cesium [1], which uses WebGL [35] for hardware-accelerated graphics, to visualize an interactive 3D globe in a web browser, which opens up a range of new possibilities for the CommonSense framework.



(a) 2D (b) 2.5D (c) 3D

Figure 1: Images that show the visualization features of Cesium: 2D top-down Leaflet-like view, a Columbus-style 2.5D view, and a 3D globe. Source: cesiumjs.org

Figure 1 shows the visualization features that Cesium has. It is able to display a Leaflet-like 2D top-down view, but also the 3D globe that is needed for this project. A final feature supported by Cesium is a Columbus-style 2.5D view, which is a tilted version of the 2D visualization.

Data

A model that TNO currently develops will be used as guidance for creating the visualization of 3D GIS datasets. It is called the Effects model [57], which is an advanced software suite used in performing risk analysis for the chemical industry. It models e.g. the movement of gas clouds as a result of a leak of a hazardous chemical under effects of wind. The model can be visualized in 3D in order to see the effects of height on gas leaks, which is an interesting aspect in a world where square meters get more and more expensive, which means buildings are built higher in order to save costs. This model is used as a guideline for the requirements for the adaptation to CommonSense,

but the final application should be as generic as possible when it comes to input data.

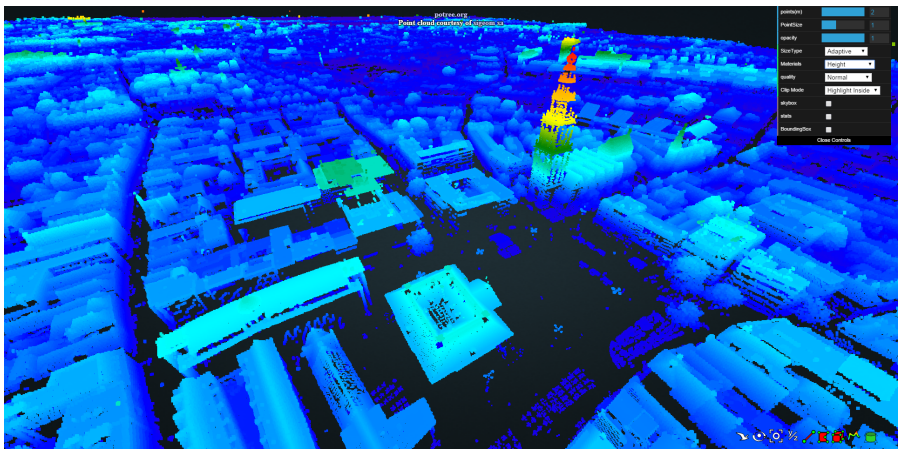


Figure 2: A screenshot of the visualization of the AHN2 point cloud of the "Grote Markt" in Groningen, rendered in Potree [47].

As of July 2015, there are almost 8.7 million buildings in the Netherlands [9]. Even when the focus is only on one city, it is too much work to create building models by hand. An automatic solution has to be found, that can reconstruct models from different datasets. A combination that can be made is between two datasets: the Basisregistraties Adressen en Gebouwen (BAG) [27], and the Algemene Hoogtebestand Nederland 2 [46]. The BAG contains outlines of buildings in the Netherlands, and information about these buildings such as when they were built. The AHN2 contains height information for the entire Netherlands, which was scanned by using Laser Imaging Detection And Ranging (LiDAR) from airplanes. The usage of LiDAR results in a high-resolution point cloud that contains height information, which can be seen in Figure 2.

This combination is the basis for the Top10NL building database containing $\approx$ 3 million building models in the Netherlands[28]. This database was made by using a point-in-polygon procedure in order to find heights corresponding to outlines of buildings, and they resulted in height information for the outlines of buildings, as can be seen in Figure 3.

In the rest of the chapter, the following will be explained. The project description ('what') is given in Section 1.1, its motivation ('why') in Section 1.2 and its method ('how') in Section 1.3. The project's scope is laid out in Section 1.4, and its objectives are presented in Section 1.5. In Section 1.6 the structure of the rest of this thesis is introduced. This chapter is concluded by a formal problem formulation in Section 1.7.

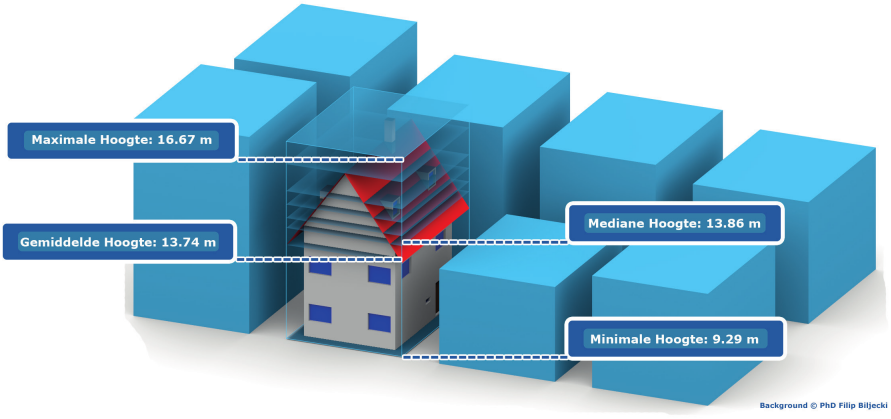


Figure 3: An image showing the properties that buildings have in the Top10NL Building database. Source: Kadaster.

1.1 PROJECT DESCRIPTION

In this project, an interactive WebGL based web application that visualizes 2D and 3D GIS datasets is created, combined with a 3D visualization of the nearby buildings in order to give the user a more context-rich experience. The application should allow the user to interactively change the parameters of the visualization and its surroundings in order to extract the most amount of information from the data. The goal of the project is to create a flexible, data-generic and feature-rich application. The development and research of potential features will be guided by a model that TNO develops, but the application should be able to visualize other datasets with similar results. The final application will be an extension to the CommonSense project.

1.2 MOTIVATION

The motivation for this project is twofold. One of the reasons comes from multiple projects within TNO. The CommonSense framework originated from a client project "Zorg op de Kaart"[1], which has been continually developed as an open source project since the release of Zorg op de Kaart, gaining functionality on the way. However, since the CommonSense framework is based on Leaflet, the possibilities for 3D GIS dataset visualization are limited. A screenshot of the Zorg op de Kaart can be seen in Figure 4.

TNO has multiple models that have a 3D coordinate system as a basis, such as the previously mentioned Effects model. The model can both take height into account, but can only do it for one height at a time.

1 See www.zorgopdekaart.nl

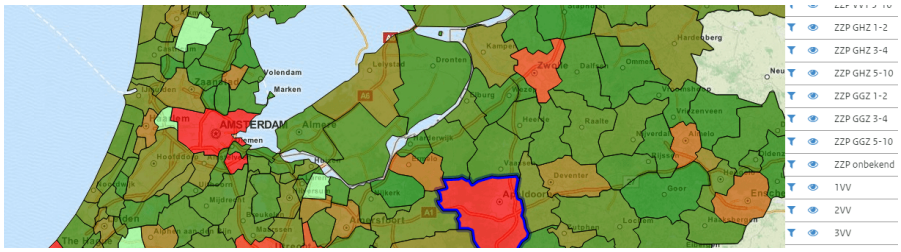


Figure 4: A screenshot of the Zorg op de Kaart application, based on the CommonSense project built by TNO. Source: zorgopdekaart.nl

The data exported from both these model is based on the height parameter of the input. However, the models are visualized in only two dimensions. This means information has to be combined or discarded in order to result in a dataset that can be visualized in 2D, where it is impossible to see if the gas comes from a leak on the ground or from a chimney high above the factory. These models are able to export a 2-dimensional output file containing points with an intensity value for a specific predefined height, or a 'slice'. These 'slices' can be stacked on each other into a full 3D dataset, which could result in extra insight in the data, and in the functioning of the model itself.

After an interview with model experts from TNO, it has become clear that the current 2D visualization techniques are not capable of visualizing the effect of height. A 3D visualization should give the model experts extra insight in how their model works, besides the extra insight gained from the added third dimension.

1.3 METHOD

During this project, an extension of the CommonSense framework will be made that allows the user to switch between different map renderers, allowing the users to visualize 3D GIS datasets if the selected renderer allows it. A map-renderer candidate is Cesium[1], which uses WebGL in order to visualize a 3D globe in the browser. The design and evaluation were done based on requirements that were set up in collaboration with model experts from TNO, and the thesis supervisor at TNO.

When there were presentable results or prototypes, these were shown to the thesis supervisor and to the model experts, by which further research could be guided. All development is done on the open source repository of the CommonSense framework [19].

1.4 SCOPE

The topic of visualization is vast, so the scope of this project needs to be clearly defined in the various areas that it interacts with. The main field of research is scientific and information visualization of 3D GIS datasets. Here, the interaction of the user and the parameters of this visualization in an interactive WebGL environment play an important role. The underlying theme of this research is interactivity. These guidelines will be further discussed and expanded upon in the requirements section: Section **??**.

The focus is not on creating a specific best visualization of the datasets mentioned, but the datasets are used as guidelines in order to design and create a flexible visualization application that can be used with other 3D GIS datasets.

1.5 OBJECTIVES

The aim of this project is threefold:

1. explore the advantages and drawbacks of *3D GIS visualization* in comparison to 2D visualization,
2. develop methods and techniques that achieve a *sense of context* and scale in the application,
3. explore the concept of *interactivity*

Each of these objectives will be expanded upon in the next sections.

1.5.1 *3D GIS visualization*

The current state of visualization of the models previously mentioned and requirements for a 3D GIS visualization of 2D and 3D GIS datasets are assessed. The possibilities of 3D visualization and 2D visualization are compared, and the advantages of visualizing 2D GIS datasets in 3D are explored. The types of datasets that are best suited for 3D GIS visualization are discussed, optimal data formats are shown and best practices are explained. Features to be implemented will be selected based on their feasibility and their usefulness for visualization.

1.5.2 *Sense of context*

When visualizing GIS datasets, it is useful for the user to get a sense of context and scale. If the user is able to see the size of the detailed data in relation to known objects, such as buildings and landmarks, the user should be able to get a better perception of the scale of the dataset. A top-down 2D scale is already achieved in CommonSense by using satellite images as an overlay on an interactive map, but there is no implementation for height yet. A solution to this 3D problem is using building models of the surrounding area. These building models can be constructed from open data sources, such as the AHN2 and the BAG dataset [69]. This is done by Kadaster, resulting in the TOP10NL building set, which provides models for buildings in the Netherlands, but only with a low resolution, which means the buildings appear 'blocky' [28]. One of the objectives here is to find out the importance of this resolution of the buildings and find a best source of building models.

1.5.3 *Interactivity*

A key aspect of this project is interactivity. The purpose of interactivity in this project is threefold:

- being able to alter the view of the dataset, e.g. rotating, panning and zooming the camera,
- being able to alter the visualization of the dataset, e.g. changing color maps, filters and data transformations of the dataset,
- the project should run entirely in the browser using WebGL, which result in a low usage threshold for the end-user.

Open-source projects such as Cesium are available that provide the last requirement of interactivity, but it is unknown whether it has sufficient performance when larger datasets are introduced. If there are performance issues, preprocessing steps may have to be taken in order to convert the dataset into a (smaller) format that is better suited for this task.

1.6 ORGANIZATION

The rest of this thesis is organized as follows. First, in Chapter 2, related literature on 3D object reconstruction, 3D GIS visualization, interactivity using WebGL and commercial GIS products are investigated. After that, the problem domain is explored in Chapter 3. In Chapter 4 the formal requirements of the application are analyzed

and formulated. Based on these requirements, the related work and the current design of CommonSense the design of the application is constructed in Chapter 5. The results of the project are presented in Chapter 6, and Chapter 7 presents the conclusions based on these results. This chapter also contains directions for future research.

Each of the following chapters starts with an introduction of its contents and structure. Then, the content of the chapter follows, and it is concluded by a short summary.

1.7 PROBLEM FORMULATION

What are the possibilities, (dis)advantages, problems and best practices of rendering 3D GIS in the browser using WebGL, using a focus-plus-context approach with a high level of interactivity?

2

RELATED WORK

This chapter positions the work of this project with respect to existing literature, giving references to related publications and indicating the relation and relevance of these publications to this project. This project is based on work done in the CommonSense framework, combined with previous work in several different fields within computer science, visualization in particular.

The rest of this chapter is divided into sections according to the primary field of study of the publications. In Section 2.1 some general work related to 3D GIS is discussed. Representation of 3D data is discussed in Section 2.2, together with some short insight in how to store this information in Section 2.3. Reconstruction of real-world objects, such as buildings, is discussed in Section 2.4. In Section 2.5 general 2D and 3D visualization is discussed. In Section 2.6 some work relating to the concept of context is discussed. Furthermore, the concept of interactivity is explored in Section 2.7. As GIS visualization is a field with many commercial competitors, major players that are available on the market are discussed in Section 2.8. Finally, in Section 2.9 the findings of this section are concluded in a summary.

2.1 3D GIS

Multiple review papers have been written over the course of years on the topic of 3D GIS, e.g. by Zlatanova et al. [69] in 2002, and Stoter and Zlatanova [53] in 2003. Zlatanova et al. showed in 2002 that GIS is the most sophisticated system that operates with the largest scope of objects among all types of systems dealing with spatial information. At the time, the need for 3D information was rapidly increasing in various fields such as urban planning and geological and mining activities. Once developments in 3D GIS provide a compatible functionality and performance, the spatial information services will evolve into the third dimension. Traditional GIS vendors provide extended tools for 3D navigation and exploration. However, many of these systems were lacking full 3D geometry for 3D representation. At the time, an interesting shift was happening that can be seen as the base for this project, from monolithic individual desktop applications, to integration of strong database management and powerful editing and visualization environments. At the time, only the first step was made,

focusing mostly on geometry. The third dimension with respect to topological issues was still in the hands of the researchers, because there was not consensus on a 3D topological model at the time. A logical consequence of all the attempts was the agreement on the manner for representing, accessing and disseminating spatial information, i.e. the OpenGIS specification [39].

Feng et al. [15] and Zhou et al. [68] set up the foundation on which open-source software such as Cesium(Analytical Graphics Inc. [1]) are built. Feng et al. introduce a method of implementing a 3D WebGIS system based on WebGL technology. This system uses an Ellipsoidal Mercator projection, WGS84 coordinates, JSON file format for network transformation, and described popular methods for tile map services.

In 2012, Loesch et al. [32] introduced the OpenWebGlobe project, which is an open source virtual globe environment using WebGL. Loesch et al. claim that unlike other web-based 3D geo-visualization technologies, the OpenWebGlobe not only supports content authoring and web visualization aspects, but also the data processing functionality for generating multi-terabyte terrain, image, map and 3D point cloud datasets in high-performance and cloud-based parallel computing environments. However, later on in their paper they revealed that point cloud data has to be stored in a proprietary JSON format and significantly thinned out to decrease the data.

These techniques have been used as a basis for multiple 3D visualizations on the web, e.g. by Engel et al. [13], Krooks et al. [29], Prandi et al. [43], and many more.

In 2003 Stoter and Zlatanova [53] expanded on the research done by Zlatanova et al [69]. by addressing the three main bottlenecks presented by Zlatanova et al.: organization of 3D data, 3D object reconstruction, and representation and navigation through large 3D models.

2.2 REPRESENTATION

One of the important aspects of the organization of 3D data is their representation, according to Stoter and Zlanatova. For modeling 3D objects, several 3D abstractions are possible [34]. Stoter et al. address four different methods of 3D data representation:

1. Constructive Solid Geometry (CSG) [16, page 557],
2. Tessellation representation (Voxels),
3. Tetrahedrons (Carlson [8]; Verbree and van Oosterom [61]),

4. a boundary representation.

CSG is an approach for modeling 3D objects by a combination of primitives, such as spheres, cubes and cylinders, and set operations as union, intersect and difference [48]. The advantage here is a structured approach to the problem by using a semantic combination of operators and primitives. This means complex shapes such as a cube with a spherical hole is modeled semantically, as can be seen in Figure 5. A problem with this approach is that the modeling of real-world objects is that the objects and their relationships might become very complex.

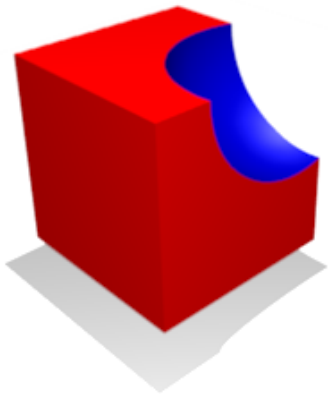


Figure 5: A screenshot of an object constructed by Constructive Solid Geometry, using the boolean difference operator on a cube and a sphere.

The second method of 3D representation is based on voxels. A voxel is a volume element (3D "pixel"). This method can represent 3D objects by a 3D cubical array, where each element holds one (or more) data values. A disadvantage of voxels is that high resolution data requires a large volume of computer space. Another problem is that a surface of natural objects is not regular by nature, there is always some roughness present. Point-based rendering techniques such as splatting can be applied to this data to overcome the last problem, but this might be intensive on the Graphical Processing Unit (GPU). This method is often used in LiDAR datasets, where each voxel corresponds to an intensity measure, where the intensity is the height at that voxel location.

Tetrahedra are a third method of representing 3D data. Carlson [8] proposed a model called the simplicial complex. The simplex is the simplest representation of a cell. A 0-simplex is a point, 1-simplex is the straight line between two 0-simplexes, 2-simplex is the triangle composed by three 1-simplexes and a 3-simplex is the tetrahedron

composed by four 2-simplexes that forms a closed object in 3D coordinate space. A disadvantage of tetrahedra is the fact that it might take many tetrahedra to construct one factual object.

A last method suggested by Stoter and Zlatanova is based on a boundary representation. This method is based on the idea to represent 3D objects by bounding low-dimensional elements, such as vertices (0D), lines (1D), polygons (2D) and polyhedra (3D), organized in various data structures. The main advantage of boundary representations is that it is optimal for representing real-world objects. The boundary of the objects can be obtained by measurements of properties that are visible, i.e. boundaries. Another large advantage is the fact that most rendering engines are based on boundary representations. A disadvantage is that boundary representations are not unique and constraints may get very complex. For example, a boundary element could be a face, triangle or polygon, with constraints such as 'holes' in the polygon, which describes parts of the polygon that should not be drawn. These 'holes' can each have their own holes, which should be drawn.

These methods were evaluated based on their advantages and disadvantages, but it was not until 2008 that the GeoJSON specification was conceptualized by Butler et al. [6]. The GeoJSON standard is an open standard format for encoding collections of simple geographical features using JavaScript Object Notation. The GeoJSON standard is based on the last method suggested by Stoter and Zlatanova, the boundary representation using points, lines and polygons.

2.3 DATABASE MANAGEMENT SYSTEMS

The second aspect of the organization of 3D data is the Database Management System (DBMS). It was first described in (Vijlbrief and van Oosterom [63]) that GISs are evolving to an integrated architecture, in which both spatial and non-spatial data is maintained in one DBMS. At the time, mainstream DBMS spatial types were implemented according to the OpenGIS Consortium specifications for SQL [38]. However, these implementations are only 2D and are based on the geometrical model defined with a boundary representation. This is due to the fact that DBMSs at the time did not support 3D objects, although z-coordinates can be used to store 3D objects. Only length and perimeter of polygons and polylines were 3D functionalities(PostGIS [44]; MapInfo [42]), and spatial indexing in 3D. Arens et al. [2] implemented a true 3D primitive (polyhedron) as an extension of the geometrical model in Oracle Spatial 9i, which is an extension that allows spatial data to be stored and analyzed in an Oracle database, which included operators to validate the 3D objects and

3D operators such as distance in 3D and point-in-polyhedron. This implementation was based on the proposal described in Stoter and Van Oosterom [52].

2.4 RECONSTRUCTION

3D GIS requires a 3D representation of objects, which has been explored in the previous section. However, 3D object reconstruction is a relatively new issue in GIS, since generating models used to be done with CAD software, such as MicroStation GeoGraphics [3] (now Bentley Map) and Google Sketchup [59]. A recent movement by Google was to crowd-source the creation of these objects for Google Earth [20], which shows the problem with labor intensity of 3D object reconstruction.

Traditionally, GIS makes use of data collection techniques such as measurements and surveying of the real world. A lot of 3D data is available in CAD designs, for example the Sketchup 3D Warehouse [60]. Most of the models created in CAD software, are industrial models designed for production purposes. Geo-applications these days require much more advanced functionality such as linking (part of) these models to (real-time) information. A relevant question is whether the CAD models can be used in 3D GIS.

A lot of research has been done toward automation of 3D object reconstruction. There are a variety of approaches based on different data sources and aiming different resolution and accuracy. Four general approaches are considered for the automated construction of 3D models:

1. bottom-up,
2. top-down,
3. detailed reconstruction of all details,
4. and a combination of all of the above.

The bottom-up approach uses footprints from existing 2D maps and extrudes the footprints with a given height using laser-scan data. The problem with this approach is that the detail of roofs of buildings can not be modeled, as only one value is used for every footprint. Due to this problem, buildings may appear as blocks. However, this might be a sufficient approach for applications that do not require a high accuracy or much detail on the roofs, since it is a very fast approach. This is the method that the Kadaster's Top10NL Building models dataset uses [28]. It uses the BAG (Kadaster [27]) dataset for footprints, and the AHN2 dataset (Rijkswaterstaat [46]) as laser-scan data to extrude these footprints.

The top-down approach is similar to the bottom-up approach, but it uses the roof obtained from aerial photographs, airborne laser-scan data and height information from the ground, such as points on the ground near the building for reference. This approach focuses on the modelling of the roof (Bignone et al. [4]; Gruen and Wang [22]), but the accuracy of the obtained 3D models is dependent on the resolution of the source data.

The third approach reconstructs a detailed object containing all details. The most common approach is based on raw 3D point clouds obtained from laser scan data, and then fitting predefined shapes [65], or 3D edges extracted from aerial photographs (Lowe [33]; Förstner [18]). This results in the best quality reconstruction of the real-world object, and it can be fully automated. Disadvantages are that high quality objects require more rendering performance, and the generation of these objects can be very time-consuming to generate since the algorithms used are very complex.

The final method combines all of the above methods, and is used by e.g. Hofmann et al. [25] for laser-scan data and scanned topographical data. Very good results were achieved with over 95% correct classifications in an urban area. Guo and Yasuoka [23] suggested a method of reconstructing the building footprints by using an active contour model, or snakes.

There is no universal automatic 3D reconstruction approach. Every method has advantages and disadvantages, and even if there is an optimal way of 3D reconstruction, it is often completed by manual methods. Creating detailed objects is a very labor-intensive task, which means it should be adapted to the requirements of the application. The approach of combining approaches is a risky one, because many data sources are used and combined, each with different scale and quality which might make the approach more complex. Using fewer data sources minimizes quality risks.

In 2010, Haala and Kada [24] published a review article in which the current state of automatic reconstruction was reviewed. Despite considerable effort, the difficulty of automatic interpretation enduringly limited 3D city modeling to systems with significant manual operations, and a small automated part. The development of fully automated algorithms is still a problem that is being tackled by large groups of researchers.

2.5 3D VISUALIZATION

The visualization of 3D geo-data has a lot of aspects that come into play when comparing to the visualization of 2D geo-data, such as

projections, readability of data, selection of 3D elements. Interacting in 3D environment asks for specific techniques. Usually, 3D models deal with large datasets, requiring efficient hardware and software. Techniques such as level of detail (LOD), where a high detail model is loaded when objects are close by, and low detail models are loaded when objects are further away, are methods that improve efficiency when navigating through a large number of objects [40]. This can be taken further by representing objects by a low-resolution simple imposter, which can be stored in the DBMS or created on the fly. The main problem with the LOD method is the fact that it requires a redundant storage of representations. If a realistic view is required, illumination, shade, etc. can be added to the geometry.

The foundation for scientific visualization regarding to this project has been laid out by Springmeyer et al. [51]. Insight is the goal of visualization, images is the medium through which this is achieved. Visualization tools often fail to reflect this fact both in functionality and in their user interfaces, which typically focus on graphics and programming concepts rather than on concepts more meaningful than end-user scientists. They note that 2D views are often used to establish precise relationships, whereas 3D views are used to gain a qualitative understanding and to present to others. Smallman et al. [50], John et al. [26] showed studies with various users and tasks have found that 2D views of 3D objects can enable analysis of details and precise navigation and distance measurements, because both dimensions of the visualization map directly to the two dimensions on the screen. On the other hand, 3D visualizations facilitate surveying a 3D space, understanding shape and approximate navigation [67].

A comparison between 2D and 3D visualization was made by Tory et al. [58] in the field of user interfaces. Tory et al. found that strict 3D visualization with additional cues such as shadows can be effective for approximate relative position estimation and orientation. However, precise orientation and positioning are difficult with strict 3D visualization, except for situations with specific circumstances, such as appropriate lighting and measurement tools. For precise tasks, a combined 2D/3D view is better than strict 2D or 3D views. Compared to 2D views, these combined views performed as well or better, inspired higher confidence and allowed more integrated navigation. This means that if precise orientation is required for an application, adding a 2D view to the 3D view may result in higher confidence levels and better results. Bleisch and Nebiker [5] also offers an insight in how to combine 2D and 3D views to achieve better insight in the data that is being visualized.

A technique that can be used to further improve visualization of 3D geo-data is Virtual Reality (VR) and Augmented Reality (AR) [62], e.g. by adding textures to objects and navigating through the 3D en-

vironment (Gruber et al. [21]). These days devices are available to support visualization in VR/AR environments, as well as track the movements of the user, such as the Oculus Rift [37], which is a head-mounted device full of sensors able to track the user in all six degrees of freedom, delivering new images based on this sensor information at high frequency.

One of the most important techniques for this project however, is WebGL [35]. It is an JavaScript API that allows the developer to render interactive 3D computer graphics and 2D graphics within compatible browsers without the use of plug-ins. Whereas CPU's reach the limit of performance increase per generation, GPU's are continually increasing in performance due to their parallel design. The WebGL technology allows for full use of this powerful device, from within the browser. Since the inception of this technique, it has found its application in many research projects, such as the visualization of large molecules in the browser [45], which was previously impossible without a plug-in.

2.6 SENSE OF CONTEXT

Card et al. [7] introduced the idea of focus-plus-context visualizations based on a review of [31]. The basic idea with focus-plus-context is to enable users to see the object of primary interest presented in full detail while at the same time getting an overview-impression of all the surrounding information, or context, available.

Focus-plus-context starts from three premises:

1. the user needs both overview (context) and detail information (focus) simultaneously,
2. information needed in the overview may be different from that needed in detail,
3. these two types of information can be combined within a single display, much as in human vision.

Focus-plus-context is a principle of information visualization: display the most important data at the focal point at full size and detail, and display the area around the focal point (the context) to help make sense of how the important information relates to the entire scene. Regions far from the focal point may be displayed smaller or selectively [17]. This concept is a useful technique to adding immersion to the visualization, and allowing the user to possibly extract more information from the visualization.

This technique has been applied by Piringer et al. [41] in the visualization of large 2D/3D scatter plots. When analyzing large datasets with

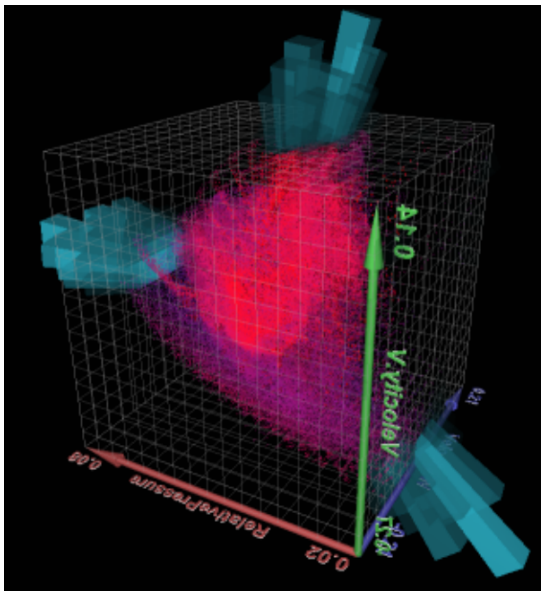


Figure 6: A screenshot of the focus-plus-context visualization of a 3D scatter plot by Piringer, Kosara, and Hauser [41]. The image shows a perspective projection of a subset of a dataset, where the blue objects represent the spatial context of the entire dataset.

many thousands of points, it it sensible to allow for zooming into the data by showing only a cubic cut-out of the whole scatter plot (spatial focus). The points outside this cubic cut-out (spatial context) are not rendered by default, but they are important to the focus visualization. The context data is projected on the border planes of the cubic cut-out, and the data is binned according to a desired resolution. Then a linear or logarithmic scaling is performed before the results are mapped to opacities and displayed using a geometry-based transparent representation. The result can be seen in Figure 6.

2.7 INTERACTIVITY

In 2002, Crampton [11] introduced and discussed various types of interactivity that can be used in digital geographical environments. Interactivity has been employed widely in geographic visualizations and has an intuitive appeal. Users report that interactivity increases enjoyment of the usage of such geographic environments. However, there is a surprising degree of variation in the literature in its usage and how it is employed. Crampton offers a very general definition of interactivity applicable to geographic visualization systems: "A system that changes its visual data display in response to user input.", where the system response time should be within a short time interval ($<$ 1s), in order to maintain the sense of interactivity. Four categories of interactivity are proposed:

1. the data (H),
2. the data representation (L),
3. the temporal dimension (M),
4. and contextualizing interaction (H).

Each of these categories is ranked ordinally (Low, Medium, High) that indicates the powerfulness of the interactivity.

Allowing direct user interaction with the data gives the user a high sense of interactivity. As is well established, in large datasets it is critical to be able to identify, discover and select pertinent patterns in the data [66]. Four types of data interactivity are identified here:

1. database querying & data mining,
2. geographic, statistical and temporal brushing,
3. filtering,
4. highlighting

Brushing is an interesting technique for exploring correlations between statistical and geographical patterns, where an active brush can be moved across a map, and all enumeration units within the area of the brush will be highlighted on an associated statistical plot, typically a scatter plot. This technique would reveal any statistical regularities in geographic regions.

When interacting with the data representation, the user obtains different views (perspectives) of the data by manipulating the way they look. In general, this visualization technique rates as less interactive than other interactivity types such as data interactivity and context interactivity. Types of interaction that correspond to this category of interactivity are lighting, changing viewpoints, changing the orientation of data, zooming, rescaling, and remapping symbols.

Of the four major categories of interaction presented by Crampton, dynamic mapping is the most prototypical. By explicitly incorporating movement into the map dynamic maps are direct opposites of static traditional cartography. Dynamic maps refer to "displays that change continuously, either with or without user control" [49]. Types of interaction that manipulate the temporal dimension are navigation, fly-bys (automatic movement of the camera), toggling (toggling between time steps to see detail in changes between them), and the sorting of the data that is visualized.

The context in which information appears is critical to analysis. The conclusion of analysis to be drawn from the data is likely to be affected by context. This does not mean everything is relative, but it emphasizes the importance of how decision-making can be framed

by a particular situation. Therefore, it is extremely important in interactive systems to freely manipulate context. Techniques that come into play here are multiple views, combining data layers, window juxtaposition, and linking.

An analysis of the popular map-based request service MapQuest.com is drawn by Crampton [11], where it is found to be a geographical environment with only a limited set of interactivity types, even though it is very popular. This means the public may therefore be gaining an unnecessarily constrained idea of the range of interactivity possible.

2.8 COMMERCIAL PRODUCTS

There are a number of commercial products on the market that are worth mentioning here:

1. ArcGIS (ESRI)
2. F4 map
3. OpenStreetMap

ESRI is the market leader in the GIS visualization sector, and ArcGIS is their most promoted software. It is used for creating and using maps, compiling geographical data, analyzing mapped information, and much more. ESRI has also released a web-based version, ArcGIS Online, including a 3D viewer to view 3D GIS datasets.

The F4 Map uses buildings from OpenStreetMap, and is powered by WebGL in order to achieve an aesthetically pleasing 3D visualization of an urban environment. However, this mapping environment has its focus on the quality of the visualization, such as lighting and reflection, and not on the quality of the information visualization.

2.9 SUMMARY

In this section relevant literature and research was explored for many related techniques and methods. A general definition was found for interactivity, together with 4 categories to classify types of interaction, and the foundations have been found for many concepts such as the representation of 3D data and how this led to the GeoJSON standard. The concept of reconstructing 3D models has been explored, but unfortunately the field is still to have fully automated reconstruction of real-world objects. A short foray was taken into the back-end of storing 3D models, but it is a bit too much outside the scope of this project to fully dive into. Finally, the most important commercial 3D visualization tools were lined up to compare to the theory and techniques.

This section raised multiple solutions to the problems described. Choices have to be made based on the requirements of the application, such as the quality of 3D reconstruction and the level of interactivity desired. To set up the requirements, scenarios in which 3D GIS is used are drawn in the next section.

3

PROBLEM DOMAIN

In this chapter, the problem domain will be explored by laying out a foundation on which this research is built. This includes not only a motivation for the usage of data visualization, but also an introduction in the capabilities of CommonSense and the current visualization state of the Effects model, along with their limitations.

In order to comprehend what the end-user is using visualization for, it is best to find out what data visualization is, and what its use cases are. This is done in Section 3.1.

After a general introduction into data visualization, the features and limitations of the currently available visualization framework "CommonSense" are shown in detail in Section 3.2. Also, terminology specific to CommonSense is discussed.

In order to propose improvements to a visualization, one has to know the basis on which the improvements are to be made. To do this, the current visualization methods that are used are explored in Section 3.3.

3.1 DATA VISUALIZATION

Visualization is a technique for creating images, diagrams or animations to communicate a message. Visualization through visual imagery has been an effective way to communicate both abstract and concrete images for a long time. Visualization is used to transform raw data into insightful answers to questions. Questions may include examples such as

- Where is the best location for a new building or a community service offering?
- What are the most efficient alternate routes a bridge is closed for repair?
- Where will a fire most likely spread?

Visualization can be seen as a pipeline of multiple steps [54], which can be seen in Figure 7:

1. Data acquisition (conversion, formatting, cleaning)
2. Data enrichment (transformation, resampling, filtering)

3. Data mapping (produce visible shapes from data)
4. Rendering (draw and interact with the shapes)

This pipeline contains a feedback loop where the data is imported, filtered, mapped and rendered, and insight in the original phenomenon is applied to the original measuring device or simulation.

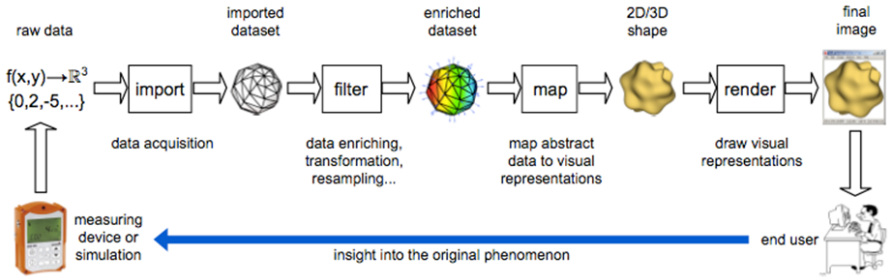
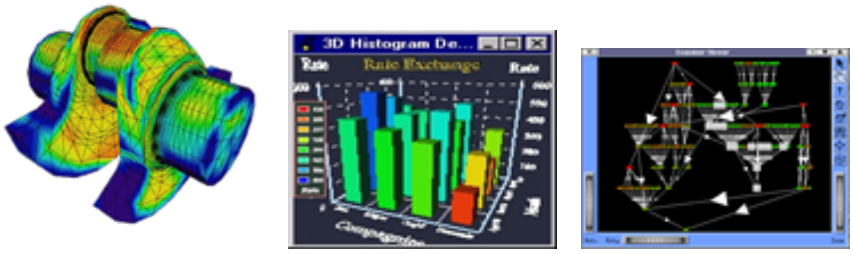


Figure 7: An image showing the visualization pipeline. Source: Telea [54]

Data visualization is typically divided into three categories, which can be seen in Figure 8:

1. *Scientific* Visualization
2. *Information* Visualization
3. *Software* Visualization



(a) Scientific (b) Information (c) Software

Figure 8: Images that show different categories of data visualization. Source: Telea [54]

Scientific visualization is the use of computers or techniques for comprehending data or extracting knowledge from the results of simulations, computations or measurements [36].

Information Visualization is applied to abstract quantities and relations in order to get insight in the data [10].

Software Visualization is concerned with the static or animated 2D or 3D visual representation of information about software systems based on their structure, history or behavior in order to help software engineering tasks [12].

All of these categories concern Visual Analytics, which is the science of analytical reasoning facilitated by interactive visual interfaces [55].

Why and when is visualization useful?

Visualization is used for confirming the known and discovering the unknown. For example, it can be used to validate the fit of a known model to a given dataset, or for finding support for a new model in the data.

Visualization is most useful in the following cases: when there is too much data and no time to analyze all of it. Visualization can show an overview in which relevant questions can be answered. Also the search domain can be refined by using visualization.

A second case where visualization is useful is when questions can not be captured directly in query. Visualization can be a very useful tool when an overview is desired, and questions can be answered by seeing relevant patterns. A third and last case is communication, for example towards different stakeholders which might not be technically adept.

3.2 COMMONSENSE

CommonSense is the basis of this project. It is existing open source visualization software built by TNO. CommonSense is "an intuitive open source web-based GIS application, providing casual users as well as business analysts and information managers with a powerful tool to perform spatial analysis", according to the open-source repository on GitHub[56]. A GIS (Geographic Information System) is a system that integrates, stores, edits, analyzes, shares, and displays geographic information. CommonSense is aimed to be easy accessible by running completely in the browser of the user. In this section, the techniques used to achieve this will be discussed.

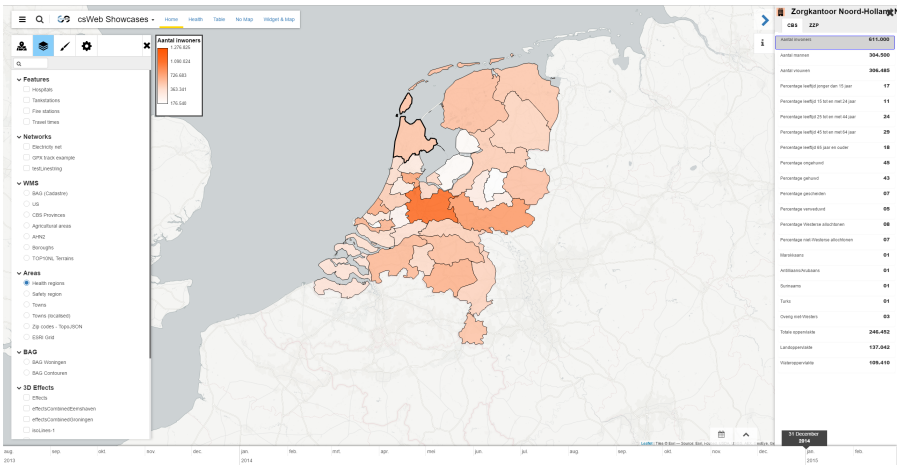


Figure 9: The main CommonSense user interface with a single layer loaded and the property "Aantal Inwoners" is used for styling the polygons.

A screenshot of the main CommonSense user interface can be seen in Figure 9. Here, it can be seen that the visualization is central to the user experience, as it takes up the largest space of the interface. On the left, modifications can be made to the current visualization by the user, where layers can be toggled on or off depending on the interests of the end user. On the right, a detail panel is shown that shows values of the properties of the currently selected feature. In the following sections, the functionality of CommonSense will be expanded upon. First, the terminology within the project will be explored.

Terminology

Within the CommonSense project, there are several terms that occur. In this section the meaning of these terms is explained within the context of this project.

CommonSense works with **layers**, which are sets of **features** that have the same properties. For example, a layer can contain all hospitals in a certain area. Layers are combined into a **project**, which is a file that combines relevant layers into a group. For example, the hospital layer could be a part of a project that has different layers for police and fire stations in a certain area. These layers can, depending on settings, be shown simultaneously or function like a radio-button where only one layer can be visualized at a time. This structure is illustrated in Figure 10.

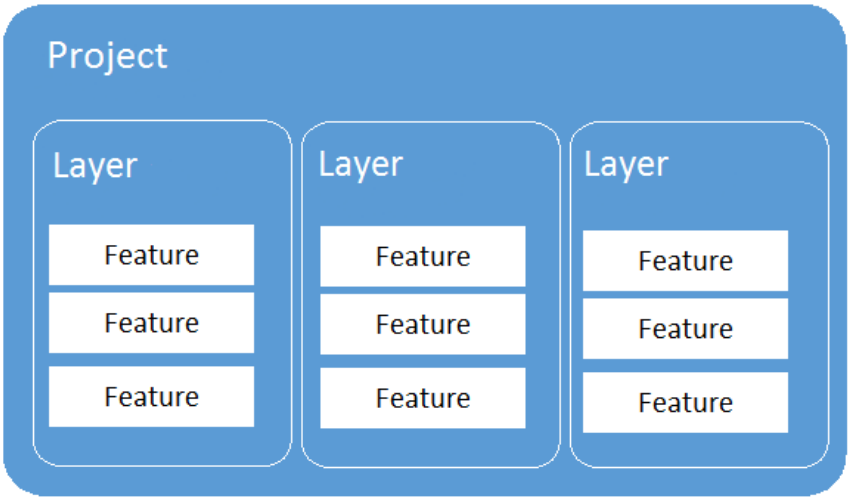


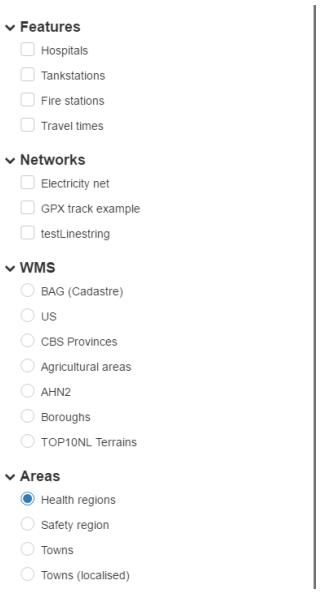
Figure 10: The main CommonSense structure, where a project file can contain multiple layers, which can have multiple features.

Features are single entities within a layer. A feature can correspond to a specific hospital. Features have **FeatureTypes**, which define the properties that features have. This means that a FeatureType "Hospital" dictates that all hospitals features should have a property that contains the number of beds that that hospital has. The styling and filtering functions are more effective due to the knowledge of the properties.

A FeatureType also has a **RenderType**, which defines how features should be drawn on a map. This render type can be either a point, line, or a polygon. Because render types are bound to FeatureTypes-Rendertypes can not differ between features and thus are the same for all features in a layer.

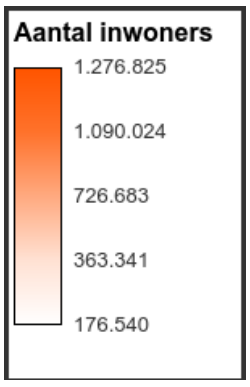
The next pages, in Figures 11,

(a) Here, the user can select a visualization feature to change in the current visualization. For example, the user can search, edit layers, change the base layer of the visualization, filter, style and much more.

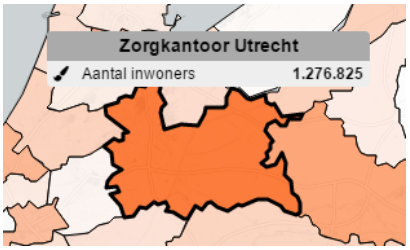


(b) In this interface, the user can select layers that are to be visualized. Layer groups define whether multiple layers can be visualized at the same time. If only one layer should be visualized, the group uses a radio button to achieve this. Here, the "Health regions" layer is currently enabled.

(c) In the right side the properties of a feature can be shown when a feature is selected. Here, the properties are shown of a Zorgkantoor in the Netherlands, which has different percentages and numbers as properties.

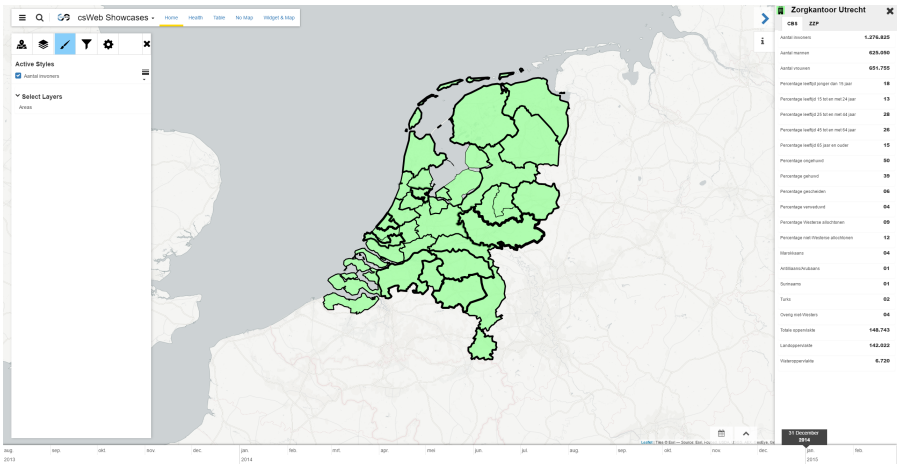


(d) Overlayed on the main visualization, a legend can be seen that shows the user what the range of colors mean.

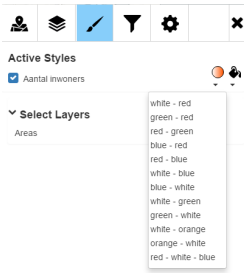


(e) When a specific property of a feature is styled, the mouse-over event displays the value of that property for the feature that is being hovered. Here, the property "Aantal inwoners" is used for styling, and the mouse is hovering Utrecht.

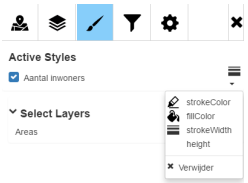
Figure 11: Cutouts of screenshots of the CommonSense user interface.



(a) In this figure, the property "Aantal Inwoners" has been styled using an adaptive border width. This means that a thicker border indicates a larger value for that property. This can be useful when multiple properties are visualized at the same time. The supported rendering features are shown in Figure 12c.

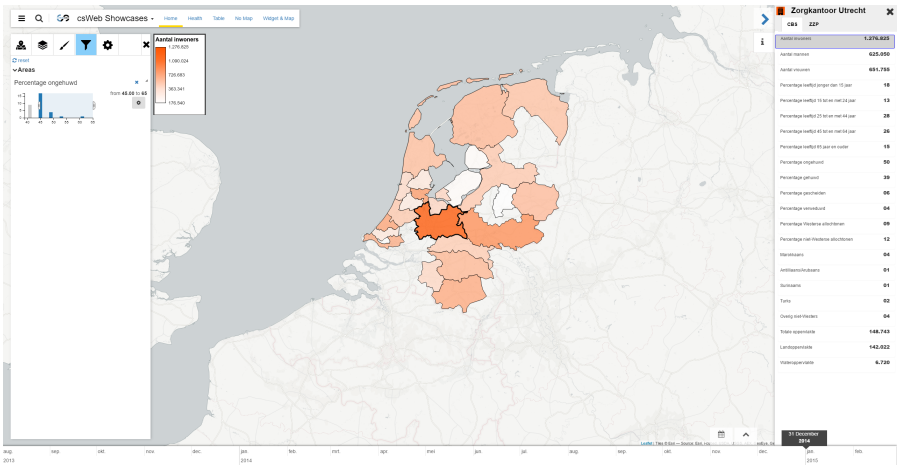


(b) In this drop-down box, supported color maps are shown. When this value is changed, the visualization is updated together with the legend.

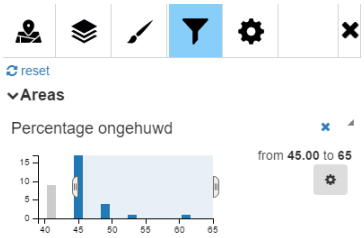


(c) In this figure, the supported render types are shown. Stroke color means the color of the border, the fill color indicates the

Figure 12: Styling functions in CommonSense.



(a) In this figure, the property "Percentage Ongehuwd" has used as a filter by the user. Only features where this property lies between 45 and 65 are visualized on the map. Note that styling is still effective.



(b) Here, the user can filter the value "Percentage Ongehuwd" by entering limits in the text boxes, or visually using a histogram and sliders.

Figure 13: Styling functions in CommonSense.

TECHNICAL OVERVIEW

CommonSense is a web application written in TypeScript. TypeScript is an open source programming language developed by Microsoft. TypeScript is a strict superset of JavaScript, which is the most used language used for web based programming. TypeScript adds static typing and class-based object oriented programming to JavaScript, which allows the developers to create more maintainable and clean code. The application runs in an Express.js webserver on top of Node.js.

AngularJS is used as the model-view-controller framework for CommonSense. Angular is a very flexible tool that allows declarative programming to create user interfaces and connect components. This results in a clean separation of controller logic and views, which is non-trivial thing to do in JavaScript.

Bootstrap is used to style the web application, and Leaflet is the current 2D map renderer. Leaflet does not currently support 3D rendering, which is a major disadvantage of this map renderer. Furthermore, node plugins such as d3, dc and crossfilter are used for styling and filtering.

3.3 CURRENT MODEL VISUALIZATION

In this project a modification will be made to the CommonSense framework where 3D functionality is added. In order to guide and test this modification, TNO models with interesting properties are used. After drawing up an inventory of possible models, there is one model that is flexible enough to be used, and contains a real life use case for the addition of 3D functionality. The model is called Effects and is developed by TNO.

Effects

Effects is the model that will be used for testing the resulting application. Effects is advanced software to assist in performing safety analysis for the (petro)chemical industry throughout the whole chain, from exploration to use [57]. It calculates the effects of the accidental release of hazardous chemicals, allowing the user to take steps to reduce the risks involved. Effects calculates and clearly presents in tables, graphs and on geographical maps, the physical effects of any accident scenario with toxic and/or flammable chemicals. Contours of effects like overpressure and heat radiation and consequences like lethality and structural damage, provide safety professionals with valuable information for hazard identification, safety analysis, quantitative risk analysis (QRA) and emergency planning [57].

The interesting part here is the visualization on geographical maps. This technique is used to assess the risks of gas leaks in an urban scenario. In Figure 14 it is directly visible what the dangerous zones are, where color denotes the mortality rate.

However, this is the only visualization that Effects is currently capable of. An interview was conducted with a model expert of the Effects software. The expert explained that currently the effect of height is not visible in the visualization, which inhibits insight in the exported dataset. This means that a factory with a high chimney leaking hazardous materials might have a different mortality rate for a person on the ground than a skyscraper with an open window on the 100th floor. It could prove useful to be able to explore the effect of height on the visualization and analysis of the dataset. The model experts also could get more insight in how the model works by visualizing the model in 3D.

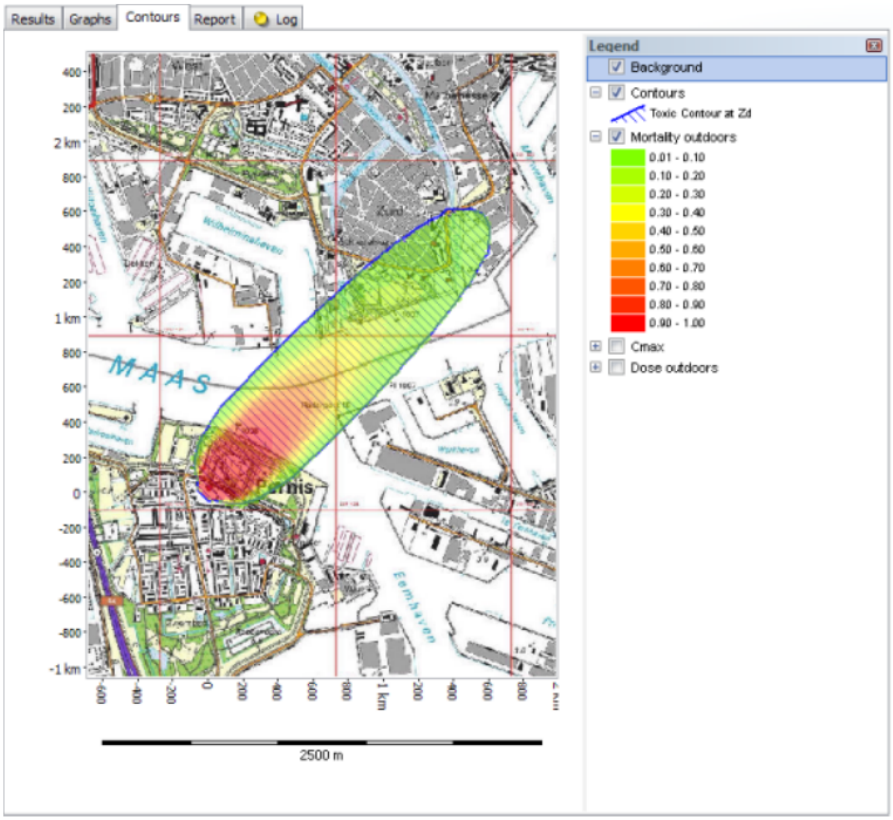


Figure 14: A screenshot of the 2D visualization of an Effects dataset in Effects. The color denotes the mortality rate of a certain gas, where blue is high mortality rate and green is low mortality rate. Source: Effects version 9 user manual.

For this project, a server running Effects software was provided that is able to calculate the outcome of the model in several situations. The user is able to run the model where a chosen amount of a certain substance is released in the atmosphere at an arbitrary location. An export was created by the model and imported in CommonSense to see what the results are by directly visualizing it. The results can be seen in Figure 15.

Effects runs a model that is not dependent on the location of gas release, so the placement of the release can be chosen by the user. In the future, it could prove useful to use location in order to take height into account, for example in places where there are mountains or hills. However, this is outside the scope of this project.

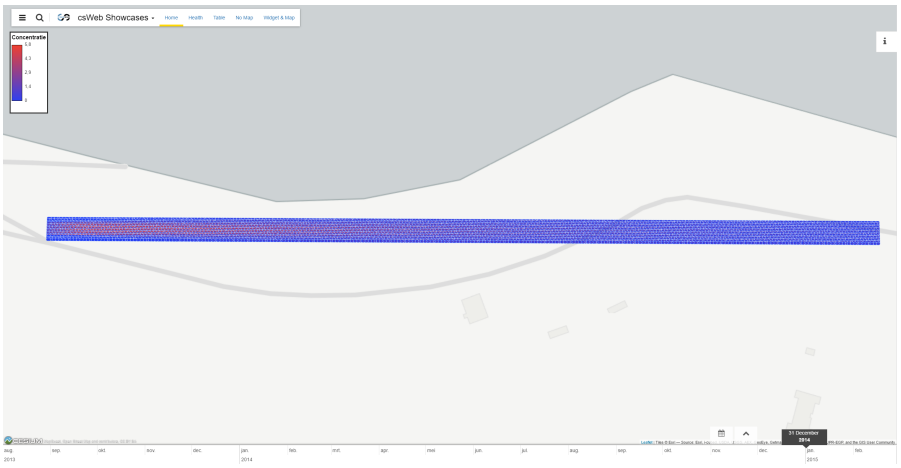


Figure 15: A screenshot of the 2D visualization of an Effects dataset in CommonSense. The color denotes the concentration of a certain gas, where red is high intensity and blue is lower intensity. The dataset is positioned near a factory in the Eemshaven, Netherlands.

4

REQUIREMENTS ANALYSIS

Now that the problem domain has been expanded, the requirements for this project have to be drawn up in order to guide the development of the modification to the CommonSense application. In order to create relevant requirements, the objectives that have been found in the introduction will be used to guide this process:

1. explore the advantages and drawbacks of *3D GIS visualization* in comparison to 2D visualization,
2. develop methods and techniques that achieve a *sense of context* and scale in the application,
3. explore the concept of *interactivity*

Each of these objectives will be elaborated upon in their own sections in this chapter. In each section, the relevance of the corresponding objective will be discussed, on which requirements will be based. The first of these objectives is 3D GIS visualization.

4.1 3D GIS VISUALIZATION

The visualization of data in 3D is one of the objectives of this project, and it is the most fundamental of the three. The addition of 3D rendering functionality will allow the user to get insight in the dataset that currently is not possible, or non-trivial, with 2D visualization. Since a goal of visualization is giving the user an intuitive look into datasets, simplifying this process is an important milestone.

In the related work chapter, the advantages and disadvantages of 2D versus 3D visualization were explored. It was found that 3D visualization is better for approximating relative positions and orientations, and 2D visualization is better for precise orientation and positioning. Based on this information, the following requirements follow.

Regarding 3D GIS visualization, the application should be able to:

4.1.1. visualize a full 3D environment in a modern browser (such as Google Chrome),

4.1.2. visualize 2D GIS datasets in 3D, using the extruded height of a polygon to visualize properties,

4.1.3. visualize 3D GIS datasets in the 3D environment,

4.1.4. visualize the 3D GIS dataset using decluttering techniques if necessary,

4.1.5. visualize this 3D environment at an acceptable performance.

4.2 SENSE OF CONTEXT

In this section, the importance of context is explored. In the related work section, context found its roots in the concept of focus-plus-context, but it was mentioned as a category of interactivity as well. This objective is relevant for the user to be able to get a sense of context on how the important information relates to the other objects in the vicinity of the focus object. In this project, the context comes from the addition of building models to the visualization, allowing the user to gauge distances in the dataset (focus) by using the building models as reference (context).

The building models have to be reconstructed from real-world data. In the related work chapter, multiple methods of reconstructing the models have been discussed. The bottom-up method is the method that is the best suitable for this project. The method uses building outlines from existing 2D data sources (BAG/TOP10NL), and extrudes the outlines with a given height using laser-scan data (LiDAR). Even though this results in a model where the details of the roofs of the buildings are very low, and buildings look like blocks because the roof is flat, it is expected that this method offers a good balance between performance and fulfilling the requirements as a context element. This is acceptable because the context dataset does not need to be a high resolution in order to give the user a sense of context. The models will be visualized in a way that they do not draw the attention from the main focus object.

The TOP10NL 3D database has used this method to create a database of about 99,7% of all buildings in the Netherlands. However, this is a huge file and it has to be segmented in smaller files in order to be able to realistically use this database as a source of context.

Regarding the sense of context, the application should be able to:

4.2.1. visualize the building models in the 3D environment,

4.2.2. visualize the building outlines in the 2D environment,

4.2.3. visualize the buildings in a manner that they do not draw the attention from the main dataset,

4.2.4. combine the visualization of a 2D/3D GIS dataset with building models,

4.2.5. visualize the buildings with acceptable performance.

4.3 INTERACTIVITY

Interactivity is key in this project. In the related work, interactivity has been split into four categories, based on the segment of the application they are mostly relevant. In this section interactivity will be explored by these four categories, combined with their perceived powerfulness as a tool to enhance interactivity. Low powerfulness means it does not make the user feel in control of the visualization, whereas a high powerfulness gives the user the feeling that it is in control.

Low: Data Representation interactivity

Data representation interactivity is one of the main categories that the focus is on in this project. Data representation interactivity mainly concerns the visualization of the data. For example, changing the viewpoint of the camera and zooming is in this category. Currently, it is possible to move the camera around in the visualization and zoom in- and out, but this is only possible in 2D. The application should be able to change the camera viewpoint in 3D too.

Medium: Temporal interactivity

Temporal interactivity concerns the alteration of the time dimension in the visualization. CommonSense currently supports a timeline to visualize data at different moments in time. However, this is not the focus of this project.

High: Data interactivity

Data interactivity relates to altering the data based on certain queries, which results in a high level of perceived interactivity of the user. Currently CommonSense supports filtering and highlighting as the main data interactivity tools. It is possible to apply filters to the data based on the user selecting a certain range of data that the focus is on. There are no plans to extend functionality in the data interactivity category, such as statistical brushing. However, this functionality should be extended to also work in the 3D visualization.

High: Contextualizing interactivity

Contextualizing interactivity is the last category that the focus will be on in this project. The combination of data layers is an important one, where the combination of building models and the 3D visualization is the focus of this project. However, this concerns the same requirements as the previous section, so there will be no requirements in this section regarding contextualizing interactivity.

Regarding interactivity, the application should be able to:

4.3.1. (Data Representation) allow the user to rotate, pan, and zoom the camera in the 3D visualization,

4.3.2. (Data Representation) allow the user to change the styling of the 3D visualization, by changing e.g. color maps,

4.3.3. (Data) allow the user to filter data based on the properties that the features have,

4.3.4. (Data) allow the user to highlight data when it is selected,

4.3.5. (Data) allow the user to alter the parameters of the Effects simulation.

5

IMPLEMENTATION

After the requirements have been drawn out for this project, the implementation was done based on these requirements. A modification has been made to the CommonSense framework to allow it to render 3D GIS data. In this chapter, the implementation of this modification is described. The aim is to give the reader an impression of how the implementation works. This chapter also shows problems that have occurred during the implementation and the solution to these problems.

5.1 COMMONSENSE

CommonSense is the base application that has been modified, so it is a logical place to start the implementation description. CommonSense runs server-side as an application, and it can be accessed in a web browser, e.g. on a different computer. CommonSense is written in Typescript, which is a language developed by Microsoft and it is a strict superset of Javascript. The main benefit of Typescript over Javascript is optional static typing and it allows object-oriented programming which benefits the cleanliness of (large) software projects. Typescript compiles directly to Javascript and it is run server-side in the node.js environment, which is based on the V8 Javascript engine that has been developed by Google for their internet browser Chrome.

Angular is used as the web application framework, which is also developed by Google and some others, which provides a model-view-controller (MVC) framework which also benefits the cleanliness of the codebase. CommonSense has a lot more functionality than just map rendering, so it also contains many other libraries such as d3, dc, and crossfilter. However, this project is most interested in the map rendering functionality, which is provided by Leaflet. Leaflet.js is an interactive map renderer that was, before this modification, tightly coupled with CommonSense. Unfortunately, Leaflet does not support 3D map rendering so a new solution has to be found.

5.2 CESIUM

A solution to this is Cesium, which is an open source 3D rendering library specifically aimed towards the rendering of 3D globes. Alternatives exist, such as Three.js, but these alternatives are more aimed towards general webGL rendering. This means a lot of work has to be done in order to equal base functionality of Cesium, such as setting up a scene and coordinate system, as well as base tiles and much more. Cesium is written in Javascript and WebGL and is aimed towards being run in the browser. A screenshot of some basic geometry rendering functionality of Cesium can be seen in Figure 16.

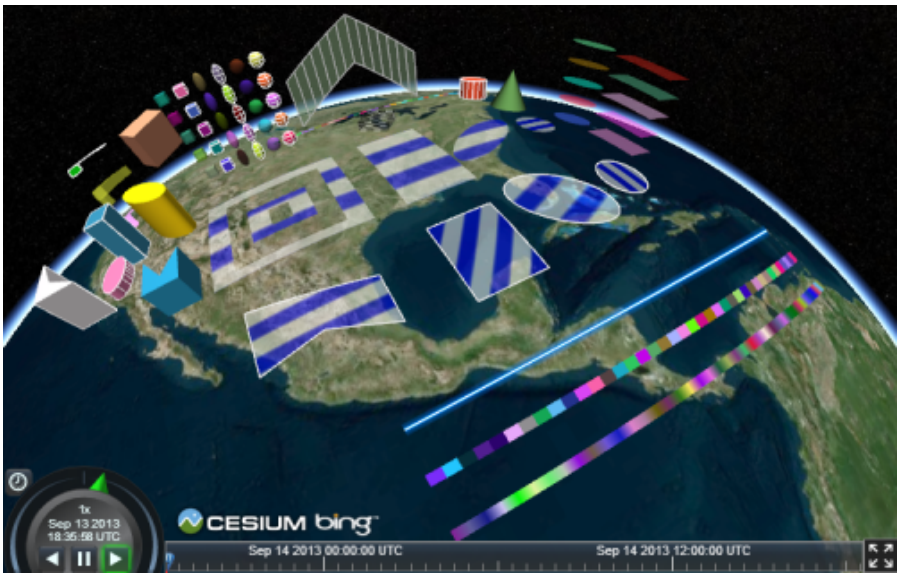


Figure 16: Basic geometry rendering functionality of Cesium. Source: Cesiumjs.org

The problem of the tight coupling between CommonSense and Leaflet has been overcome by writing a MapRenderer class that receives rendering calls from CommonSense, and sends them to either Leaflet or Cesium, based on the current render setting. A button was added to the CommonSense menu that allows the user to switch between 2D and 3D rendering mode, which can be seen in Figure 17.

When this button is clicked, the currently shown features are automatically loaded into the new rendering engine and the map bounds are updated, so it is a seamless experience for the user to switch between render modes.

Cesium supports the rendering of custom geometry and appearances, which is essential to a flexible implementation in CommonSense. However, for this project, a lot of features of Cesium are disabled, such as shading effects and terrain height, which would only make the implementation more complex and could influence results.

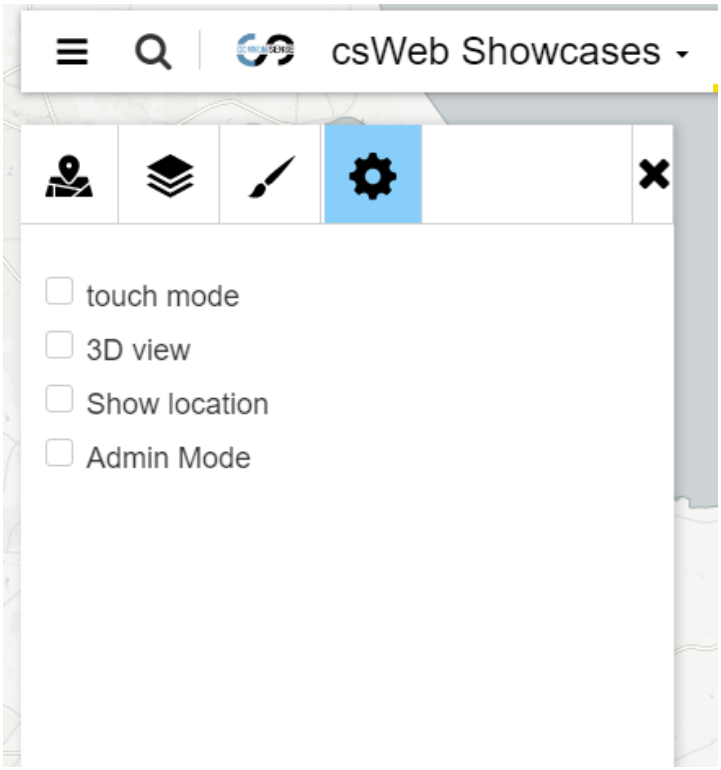


Figure 17: The button that allows the user to switch between 2D and 3D rendering.

Implementation

In CommonSense, the class that handles the rendering of layers in Cesium was written and it had to support all functionality that CommonSense provides, such as filtering and styling. Because this was done correctly, it supports most of the functionality that CommonSense has in 2D, with some added functionality that has become available because the map is now rendered in 3D. The most noteworthy here is that now the third dimension can be used to visualize some property that features have. For example, it is possible to give a polygon a height denoting the value of that property. An example of a 2D rendering can be seen in Figure 18, whereas an extra property has been be visualized in the 3D rendered version in Figure 19.

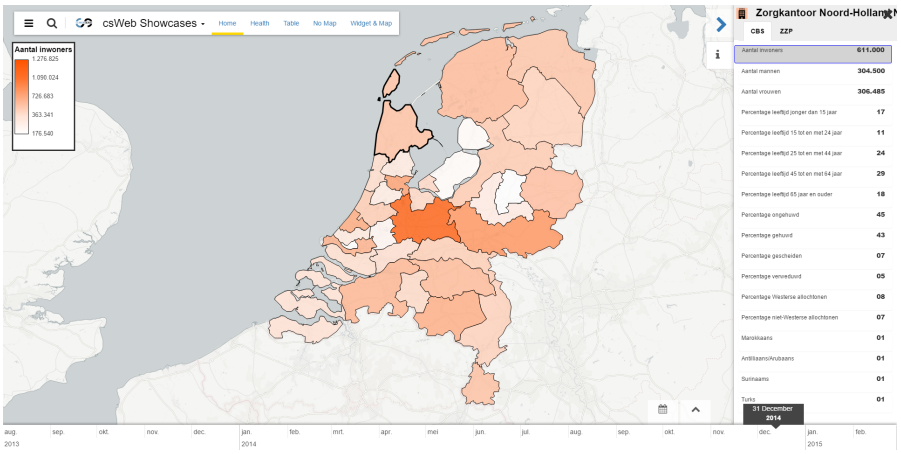


Figure 18: 2D visualization of a single property "Aantal Inwoners" in Zorgkantoren in the Netherlands.

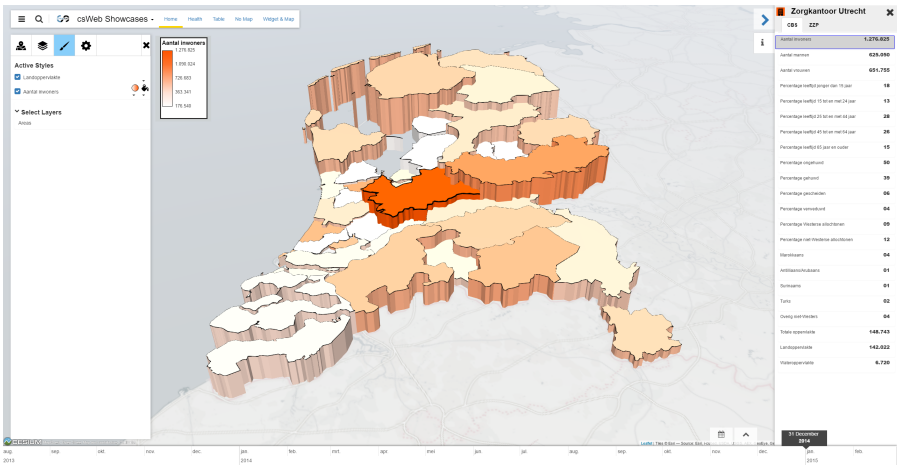


Figure 19: 3D visualization of the property "Aantal Inwoners" in color, and "Landoppervlakte" in polygon height of the Zorgkantoren in the Netherlands.

WebGL

Cesium uses WebGL as a basis for its rendering capabilities. WebGL is a Javascript API that allows very low-level Graphical Processing Unit (GPU) access to developers from within the browser, as was previously only possible with native OpenGL applications. A problem that comes with this low-level access is the fact that everything has to be set up by the user as well in a very low-level fashion. This means that all buffers and rendering matrices have to be set up, which takes a lot of time in order to get it done correctly. Some 3D Javascript libraries, such as Three.js, can take a lot of work out of the hands of the user in this category. Cesium takes almost everything out of the hands of the user here, which turns out problematic if there is low-level access needed such as writing custom shaders. This problem is explained later on in this chapter, and the solution too. Alternatives to WebGL exist, such as Java OpenGL and Stage3D (in Flash), but these need extra software to run and are not as widely supported as WebGL.

GeoJSON

Cesium supports GeoJSON as an input data type, which is also used by CommonSense. In the related work section, various methods of data representation were introduced, that can be used for the storing of the building models. For this project, using a boundary representation specification is the best option for data storage. This is due to the flexibility of this representation, and because it is the basis of GeoJSON, which is the internet standard for encoding collections of simple geographical features. GeoJSON is the standard that will be used in this project for the representation of data. It is a boundary representation of shapes, and it can encode various basic geometries, such as points, lines and polygons. GeoJSON will be used for the representation of the building models and for the 3D GIS datasets, since the datasets do not use complex geometry as a basis, but they usually use points.

5.3 SENSE OF CONTEXT

As shown in the previous chapter, context is important. To recap: the idea is that the main visualization is a 3D visualization of a gas cloud from Effects in CommonmSense, where the models of the surrounding buildings are drawn simultaneously. This is done to help make

sense of how the important information relates to the entire data structure.

LiDAR

It is very labor intensive to create and shape models of all buildings in the Netherlands by hand. This is why this has to be done semi-automatically. A method that is used in different countries is called Light Imaging, Detection and Ranging, or LiDAR in short. In LiDAR, a complete height map is made of a location with lasers in order to create a point cloud at a desired resolution of this location. This technology has many uses in various sectors such as agriculture, archeology and geology. However, a LiDAR scan could also be made of an urban environment. A result of this can be seen in Figure 20.

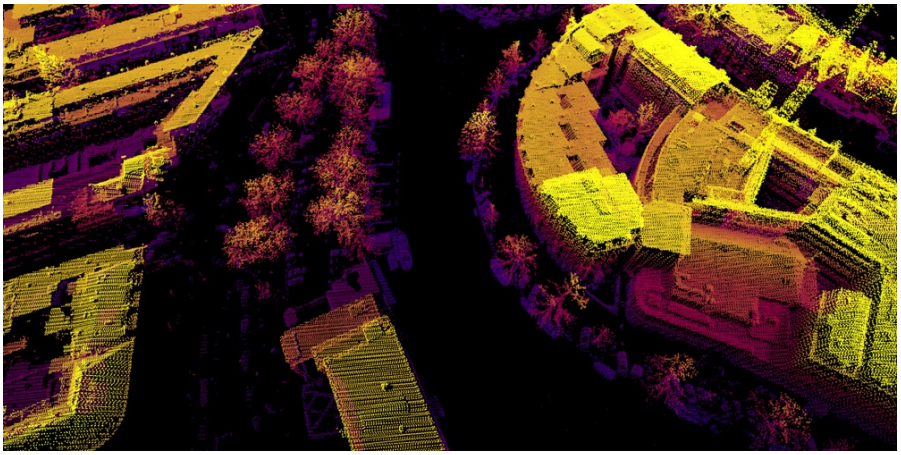


Figure 20: Raw point cloud visualization of a LiDAR scan. Source: www.oscity.eu

In the Netherlands, the government has published a LiDAR scan of the entire Netherlands as an open dataset that is called Algemene Hoogtebestand Nederland 2 (AHN2). AHN2 is available as a point cloud that has a resolution of 50 cm between points[1]. The AHN2 dataset has been captured from the air with airplanes flying over the Netherlands between 2007 and 2012, and it consists of filtered and unfiltered data. The filtered data contains just the solid objects such as buildings, whereas unfiltered data contains measurements from trees, cars and other objects.

The main problem here is the sheer number of points. Figure 21 shows the partition of the AHN2 dataset in the form of a grid. Every grid cell here is available as a LAZ file, which is a compressed binary

1 AHN3 has been released at time of writing, which has a 5 cm resolution between points. However, this increased resolution does not benefit this project because not a high resolution dataset is needed and only adds to complexity.

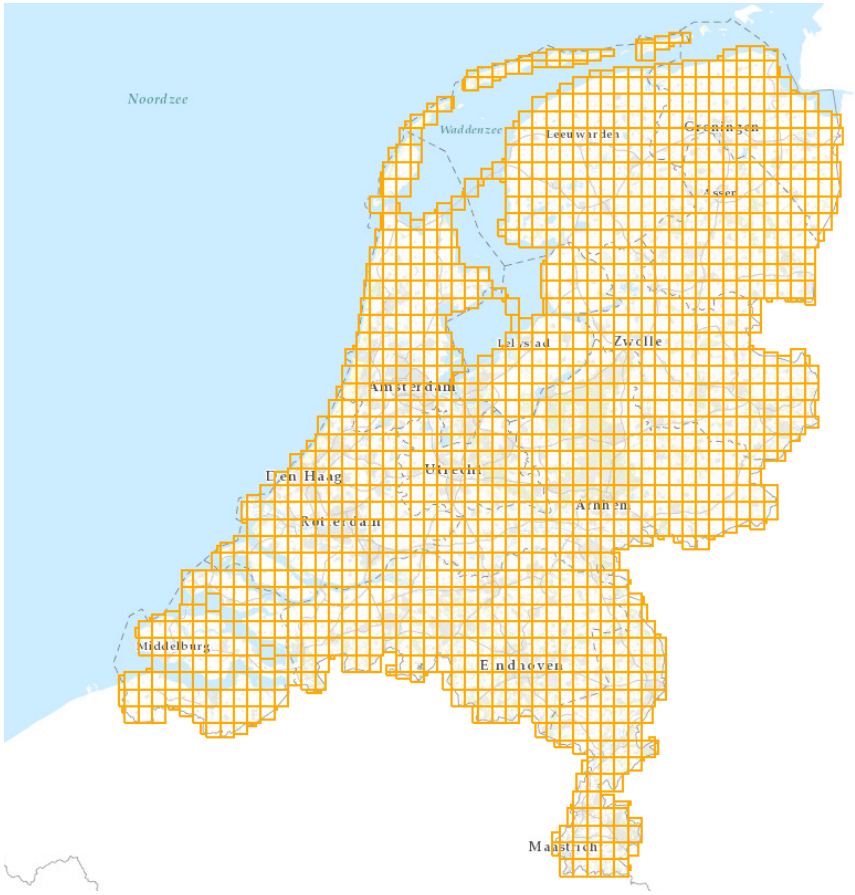


Figure 21: Grid cells that the AHN2 dataset is divided into. Every cell contains roughly 300MB of filtered data, and millions of points. Source: PDOK (Publieke dienstverlening op de kaart)

format often used for LiDAR. Every cell contains 300MB of filtered data and 500MB of unfiltered data, which sums up to millions of points per grid cell. To use this dataset directly would require setting up a special data structure and would be a research on its own.

Top10NL

Fortunately, there is a solution that uses the data from the AHN2 dataset but is greatly reduced in size. It is a dataset from the TOP10NL database which contains low-resolution polygons for over a million buildings in the Netherlands. This data has been created using the outlines of all buildings registered with the Basisadministratie Adressen en Gegevens (BAG), which was then combined with the AHN2 point-cloud using a point-in-polygon approach. An example of this approach can be seen in Figure 22.

When this approach is used, each building outline results in a number of points that lie within the outline of the building, and each point is

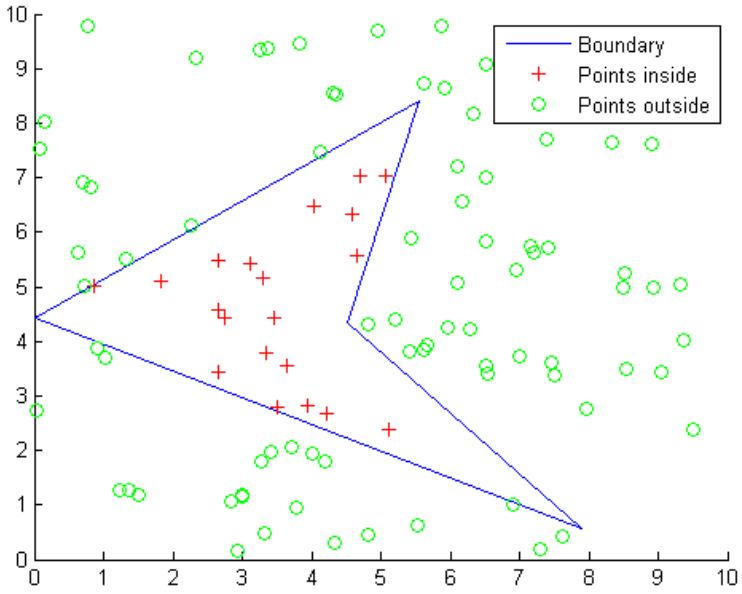


Figure 22: A basic example of the point in polygon method, which is a function that shows for a point whether it lies within or outside a given polygon.

a measurement from AHN2 which has their own height. This means some statistics have to be applied to these numbers, so the minimum, maximum, medium and average of these heights is given for each building. An image that shows the result of this can be seen in Figure 23. For this project, the median of these values is used in order to filter out outliers but still use real data.

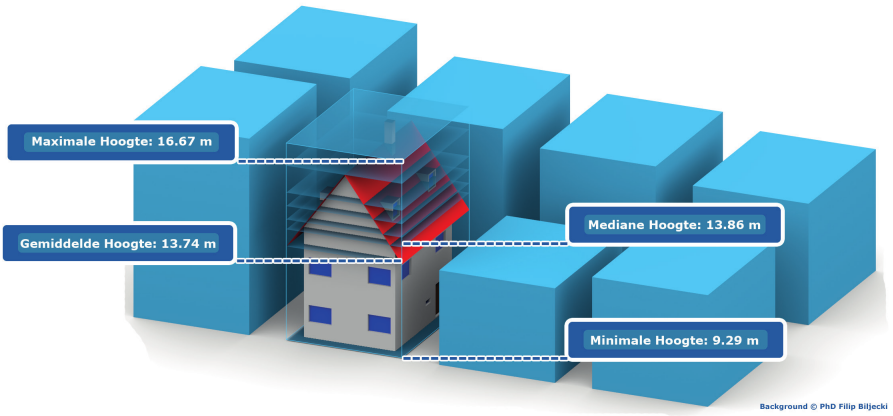


Figure 23: This image shows the properties that each building has from the Top10NL building dataset. Source: Top10NL

Now, these buildings look very square and low-resolution, especially compared to the LiDAR datasets. However, in the related work it was discovered that the context dataset does not need to be high-

resolution in order to fulfill its task as context information. If it were higher resolution, it might even be distracting from the focus dataset. This means that these building models are faster to render than the LiDAR dataset and use less data but, in this case, fulfill the task as a context dataset.

Coordinate systems

The Top10NL and a lot of government data sets in the Netherlands including AHN2, are distributed in the Rijksdriehoek coordinate format. The Rijksdriehoek coordinate format is a Cartesian coordinate system where the center $(x = 155000, y = 463000)$ is located at the "Onze Lieve Vrouwetoren" in Amersfoort, the Netherlands. This is done such that every coordinate that lies in the Netherlands has a positive x and y value.

However, this is a system that is only works in the Netherlands, which can be seen in Figure 24. The Rijksdriehoek coordinate system is only usable in the pink, blue and green zones.

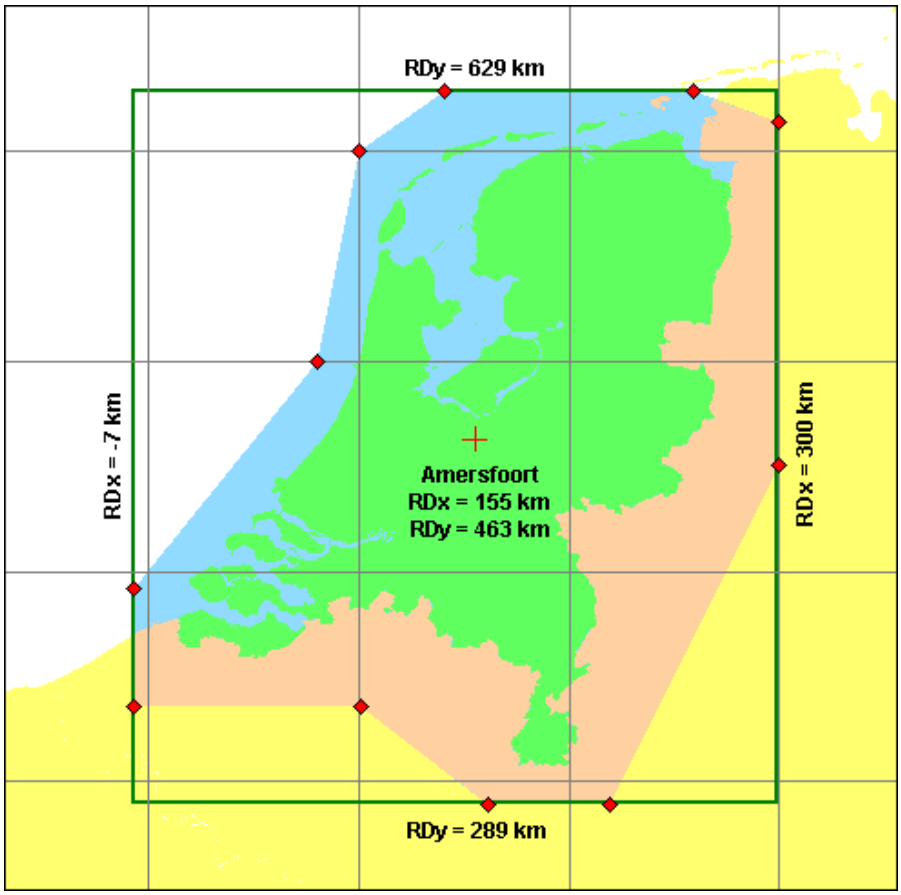


Figure 24: An image that shows the boundaries of the Rijksdriehoek coordinate system, used in the Netherlands. Source: wikipedia.org.

Because it is only used in the Netherlands, Cesium does not support the Rijksdriehoek Coordinate system, and this means the Top10NL database has to be converted to the default GeoJSON coordinate system, the World Geodetic System 1984, or WGS84 for short. WGS84 is a Earth-centered, terrestrial reference system based on a set of constants that describe the Earth's size and shape.

This conversion was done using the Geospatial Data Abstraction Library, or GDAL in short. This is an open source library that is often used in the GIS sector. There is a conversion tool that can convert between the GML format that the Top10NL database is in originally and GeoJSON, and simultaneously convert from Rijksdriehoek to WGS84.

This conversion results in a very large GeoJSON file ($\approx 2,3$GB), which is still too large to directly load into Cesium. This takes a lot of time to download, even on a very fast connection and then the rendering time is not even taken into account. In order to solve this, this file was combined with a GeoJSON file that knows the shape of all the municipalities (NL: gemeentes) in the Netherlands. For every municipality, all the buildings were looped through and a point-in-polygon function from the turf library was used in order to find out if the building lies in that municipality. If it is, it is added to a temporary list and when all buildings were looped through written to a separate GeoJSON file. This results in 408 separate files that each are a couple of megabytes in size, which is much more doable. It is now a much more flexible dataset, in which the buildings in municipalities can be loaded separately without having to load all buildings in the Netherlands.

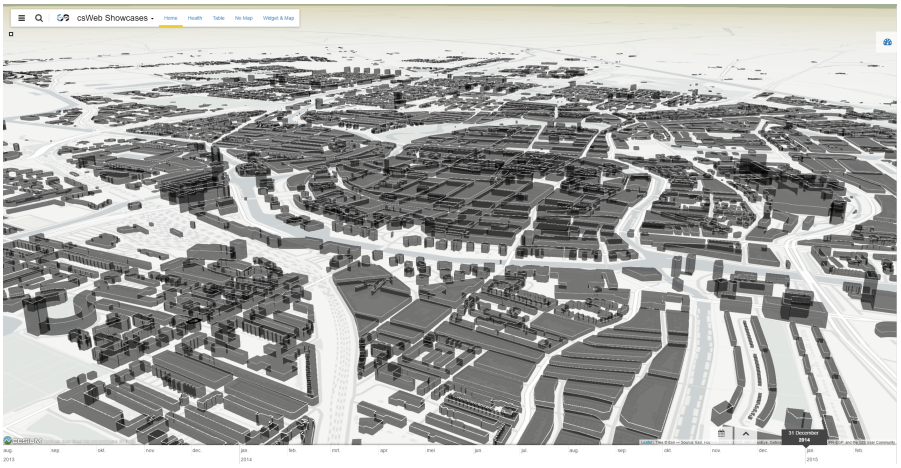


Figure 25: An image that shows the buildings in Groningen from the Top10NL dataset in the Netherlands, rendered in CommonSense using Cesium.

The result of this can be seen in Figure 25, which shows the buildings in Groningen rendered in CommonSense using Cesium. A trans-

parency and a gray color was chosen in order to not pull the attention to the buildings, because the buildings are the context dataset and should not steal the show from the main visualizatio, that is to be rendered, in which the user is most interested.

5.4 3D GIS VISUALIZATION

Now that the general implementation of Cesium into CommonSense has been made, and a method of achieving context has been found, the focus now lies on the main visualization of the Effects gas cloud. This section will first explain where the data comes from and in which format, then it will explain how the data is used in Cesium and finally the actual rendering will be discussed, including the choices that have been made during the development.

Effects

As explained before, Effects is commercial software that is very extensive in the different methods and models that are supported. It is also an application with a Graphical User Interface which needs a license to calculate new models. Luckily, a version of EffectsClient was provided that connects a server version of Effects. EffectsClient is a Windows application that sends calculation requests with specific properties such as the release mass of the gas substance, type of model to be calculated to the server, then waits for it to complete the calculation and then retrieves the calculated results. A screenshot of EffectsClient can be seen in Figure 26.

In order to find out how this application connects to the server, the network packets that are sent and received are logged with an application called Wireshark. This application monitors all network traffic on a computer, which made it very easy to see that there is an API on the Effects server that accepts requests in a specific JSON format. This means it is very easy to integrate this API into CommonSense, in order to let the user customize calculation requests that are to be sent to the server. Since Commonsense uses Angular, this is done very easily, and the results can be seen in Figure 27.

For this project, it was recommended by an Effects model expert to use the "Neutral gas dispersion: toxic dose" model simulation, which simulates the release of 1000 kg of carbon monoxide (CO), because of its simple resulting shape. Later, other models could be explored but this project focuses on this model. It is also important to note that this model is location and surroundings unaware. So later on, the result will be placed on a map and the surrounding building models will be

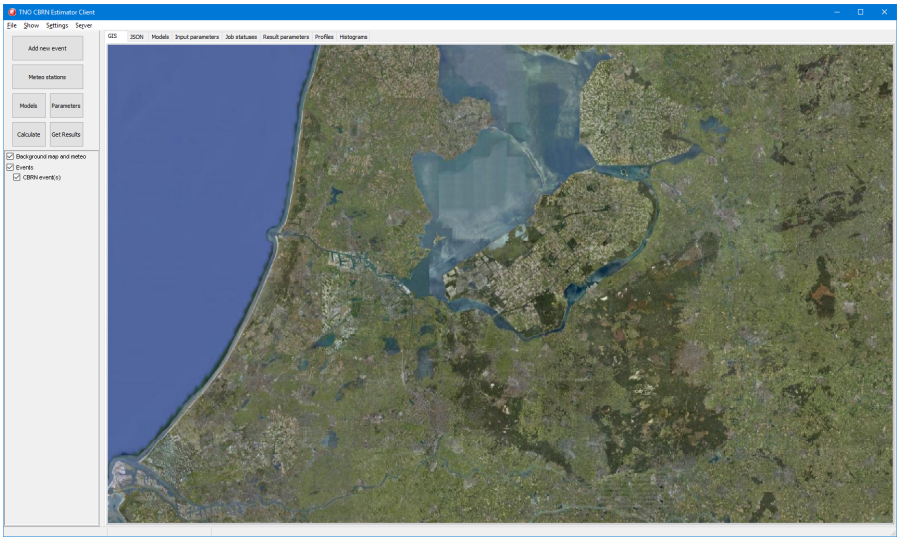


Figure 26: An image that shows user interface of the EffectsClient, an application that connects to an Effects server.

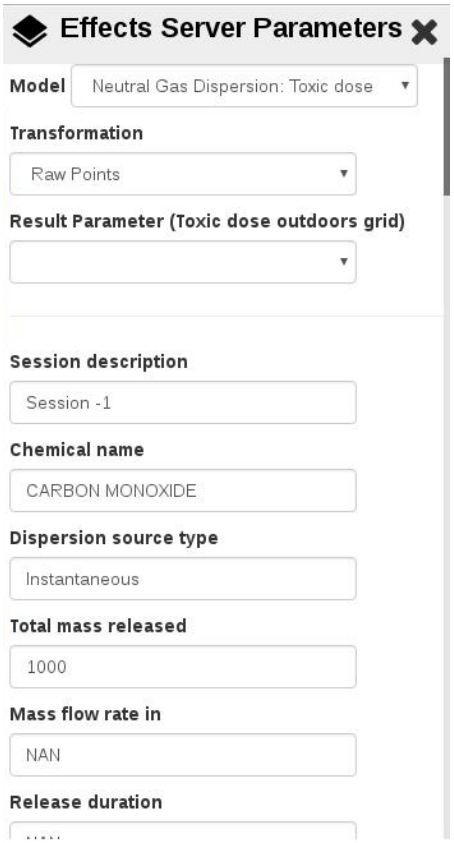


Figure 27: An image that shows the interface in CommonSense, in which the user can change simulation parameters of the Effects model calculation.

rendered, but the location and buildings are not taken into account when the model is calculated. So every time a calculation request is

sent to the server for a different location, the output will be exactly the same.

After the request has been sent and the calculation is complete, the response is retrieved and the correct output parameter is chosen (Toxic dose). This resulting parameter is in an ESRI grid format, which is not supported by Cesium, so it needs to be converted. This grid format only has relative positions in fixed intervals, which can be seen in Figure 28. In the header, it describes exactly how many points are in this file, after which the data section starts that contains a value for every grid cell in the grid.

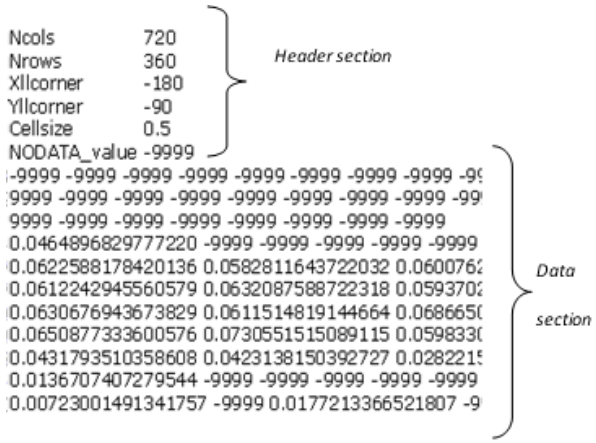


Figure 28: An image that shows the raw data that an ESRI grid file is composed of.

This grid data needs to be converted to a GeoJSON point collection, which is done using existing code in the CommonSense code base. This code can convert an ESRI grid to GeoJSON based on a specific point that the grid should be placed. This point can be chosen arbitrarily by the user. However, this code converts the grid to a set of square polygons. This is solved by using the centroid function from the earlier mentioned turf library to find the centroid of the polygon. If this is then written to a different GeoJSON file, it can be read in by CommonSense and visualized in 3D in Cesium, as can be seen in Figure 29.

However, as can be seen, there is no height in this visualization which is because the data only exists for a single height. The Effects server can only return the value for a specific height, which can be sent to the server as one of the properties ("Study z-coordinate"). In order to get a true 3D volume point cloud, requests are sent for every height between 1 and 10 meter, and every layer is individually calculated[2]. After this calculation, the point layers are combined into a true point cloud which is saved into a separate GeoJSON file which then can be

2 There is a lot of overhead here, but after evaluation with the Effects team this is one of the points they will pick up as a result of this project.

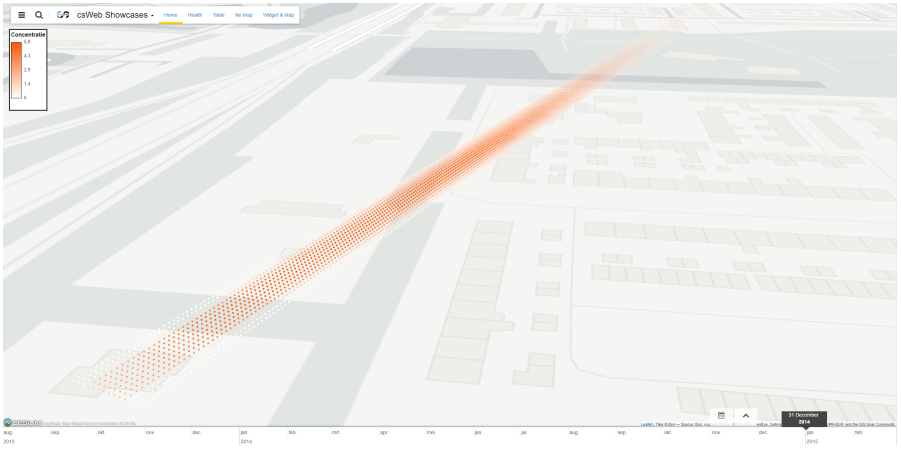


Figure 29: An image that shows the visualization of a single layer of the Effects model. The color denotes the intensity at that point.

easily loaded into CommonSense. The result of this can be seen in Figure 30.

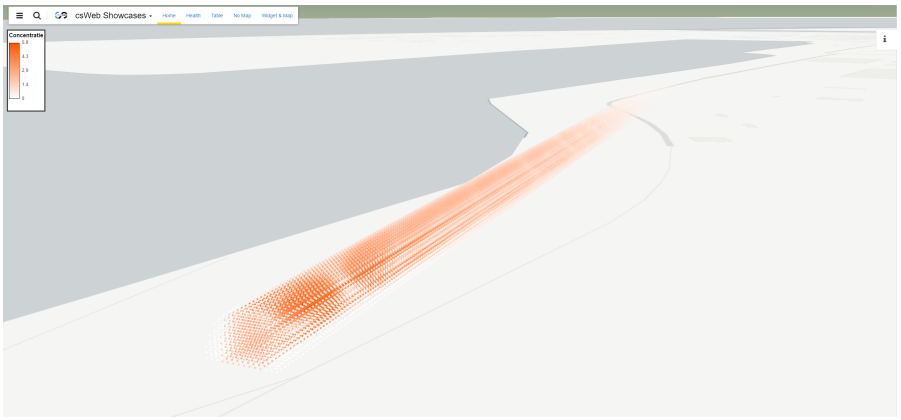


Figure 30: An image that shows the entire raw point cloud that has been created by combining multiple layers of the Effects model.

A small adaptation has been made here in order to simplify later visualization processes by overriding the distance between points from 1.2 meter to 1.0 meter, and by choosing the distance between layers as 1.0 meter in order to create an uniform grid. This means that the point cloud is actually 20% larger than is visualized in the x and y dimensions. For exact measurements, this is incorrect but for this project the extra work to do this precisely was not possible due to time constraints.

Volume Rendering

Now that the data is in the correct format, a visualization technique for this must be chosen for the point cloud data. Many options ex-

ist, such as splatting an shear warping. First, some basic techniques were tried, such as isolines which can be seen in Figure 31. However, this does not show the relation between layers very well. The 3D analogue of isolines, isosurfaces was also considered but there are some problems with it, such as occlusion and obstruction of deeper layers. Isosurfaces is a method that does a surface reconstruction, which is quite resource intensive.

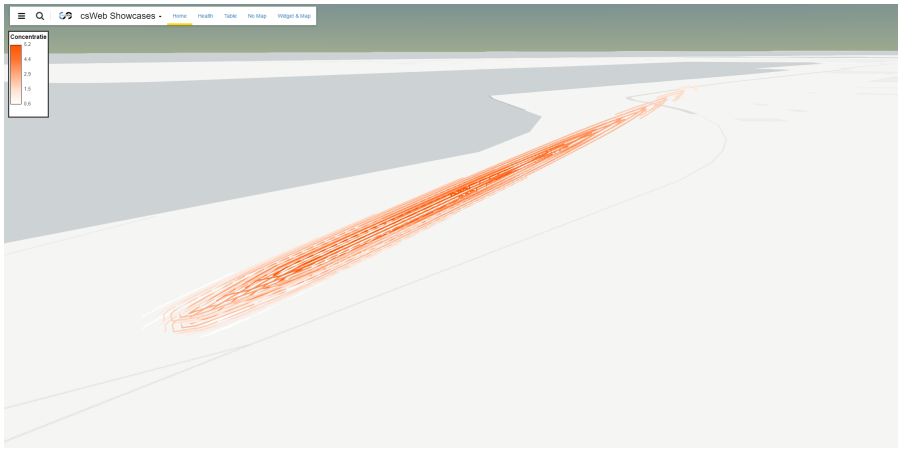


Figure 31: An image that shows the isolines visualization of a single layer of the Effects model. The color denotes the intensity at that point.

A simpler and more direct method for doing this is volume raycasting. Volume raycasting is a direct rendering approach to the visualization of 3D volumetric data sets. It is an image-based method, which means that it iterates over the pixels of the image that has to be produced, rather than iterating over objects in the scene. It is described by 4 steps that are shown in Figure 32:

1. For each pixel, shoot a ray through the volume and find out the beginning and the end of the intersected volume.
2. Between the beginning and the end of the intersected volume, select equidistant sampling points. Because these points will probably not coincide with the data points, interpolation is necessary.
3. For each sampling point, calculate shading based on the surface orientation and location of the light.
4. Finally, compose the final color for the pixel based on the calculated shading for every sampling point along the original ray.

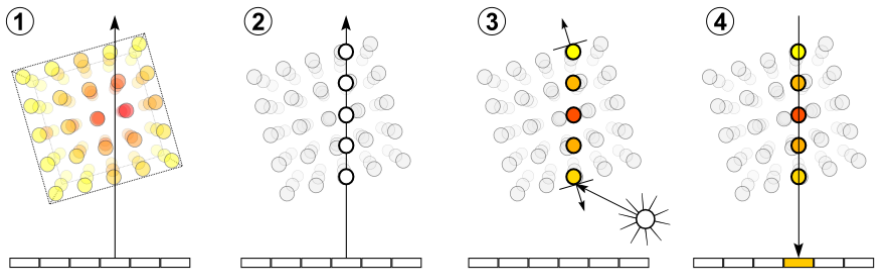


Figure 32: An image that shows the four steps of volume ray casting.

Cesium

Now, the visualization algorithm has been chosen and the data is in the correct format from the Effects server. After this, it has to be implemented in Cesium, which is not without problems.

The algorithm that has been chosen is an image-based approach which means the easiest way to implement it is by writing a custom shader. This method is the easiest, because otherwise every time the camera is adjusted, the complete object has to be recalculated in the CPU which brings additional complexity. In Cesium, it is possible to write custom shaders via Fabrics. A Fabric is a high level method of describing a material in Cesium based on properties such as specular reflection intensity and diffuse color that are passed via uniforms, which is then converted to a custom shader. However, it is also possible to override the shader code via this approach, which is the method that is used for this project.

Now that access to writing custom shader code is achieved, it is still necessary to get the volume data into the shader. The usual approach here in OpenGL is transferring the volume data into a Texture3D object, which can be directly accessed by the shader and intermediate values are interpolated automatically by the GPU. Unfortunately, Texture3D is not supported until WebGL 2.0. A workaround, suggested by Google, is to stack 2D textures onto each other, in order to create a larger 2D texture that is supported by WebGL 1. Some custom code has to be written that accepts 3D texture coordinates and converts this to a coordinate in the 2D texture. The result of this stacking can be seen in Figure 33.

In order to correctly support bilinear interpolation on the GPU, it is important to convert all values in the volume data from floats to bytes. Another note here is that in Cesium, it is not allowed to directly pass volume data via the Fabric construct to the shader. However, an adaptation was made to Cesium that allows developers to pass Textures to the shader via uniforms. A pull request containing this modification was sent to Cesium and it was accepted.

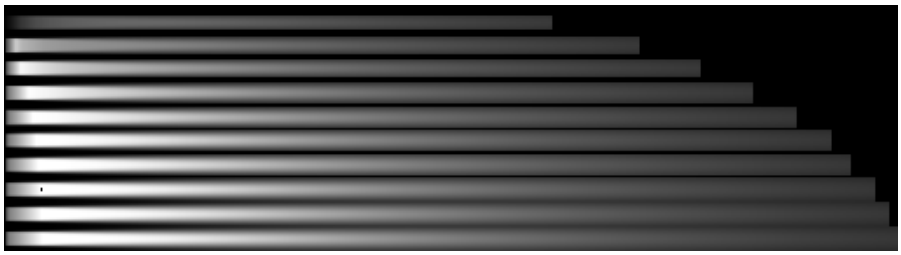


Figure 33: The stacking of 2D textures in order to work around the missing functionality of 3D textures in WebGL 1.

Now that shader access has been gained, and the data is accessible in the shader, only the Volume raycasting is what remains. Unfortunately, this is not without problems too. For step one of the raycasting algorithm, the front and the back of the volume have to be found. This is typically done in a two-pass manner, where in the first pass a cube is drawn and the depth buffer is read for use in the second pass. However, in Cesium this is not an easy task and a single-pass workaround has to be done. Using the single-pass method, the front and back of the volume are found using a ray - axis aligned bounding box (AABB) intersection. There are some problems in Cesium here where the world coordinate system is slightly tilted compared to the surface. This is logical however, because there can not be a single coordinate system where the z-coordinate is in the direction of the surface normal for every location on the planet. An image of the artifacts that derive from this can be seen in Figure 34.

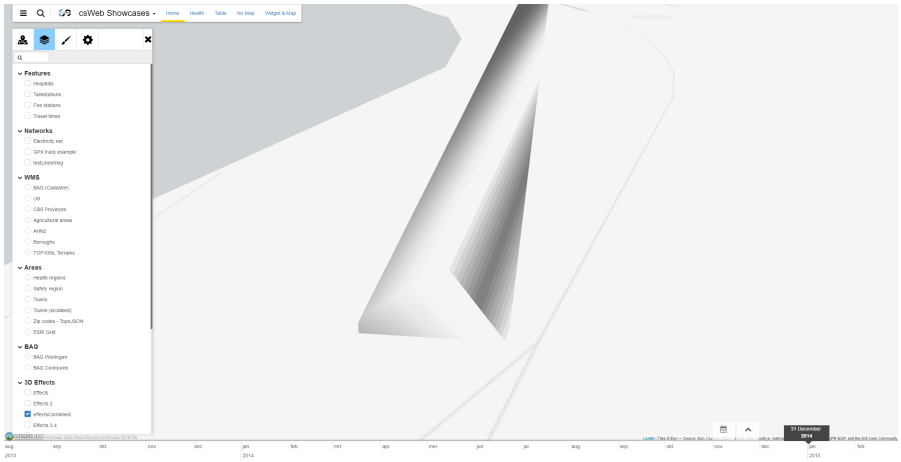


Figure 34: Artifact in ray-AABB intersection algorithm because the bounding box is not axis aligned. This can be seen in the fact that the bounding box is rotated, where one of the corners of the bounding box is in the middle of the rendered rectangle whereas it should be aligned with the corners of the rectangle.

The solution to this problem is creating a local coordinate system, by creating a transformation matrix in Cesium that transforms coordinates and orientations from an earth center fixed reference frame

to a local coordinate system where the x dimension points north, y points eastward and z points in the direction of the surface normal. After this matrix is passed to the shader, all coordinates have to be multiplied by this transformation matrix in order to be in the same reference frame and the problems are solved.

6

RESULTS

The results of this project are shown in the form of a storyboard. A use case was devised that combines all aspects of this project into a logical sequence of screenshots. Since Effects is risk analysis software, this storyboard is in the same sector.

The main idea in this storyboard is as follows: there has been a gas leak at a gas station, and the end-user wants to know what the influence of this gas leak is.

As the first step, a layer that contains all gas stations in the Netherlands has been created. An example of this layer, focused on Groningen, can be seen in Figure 35.

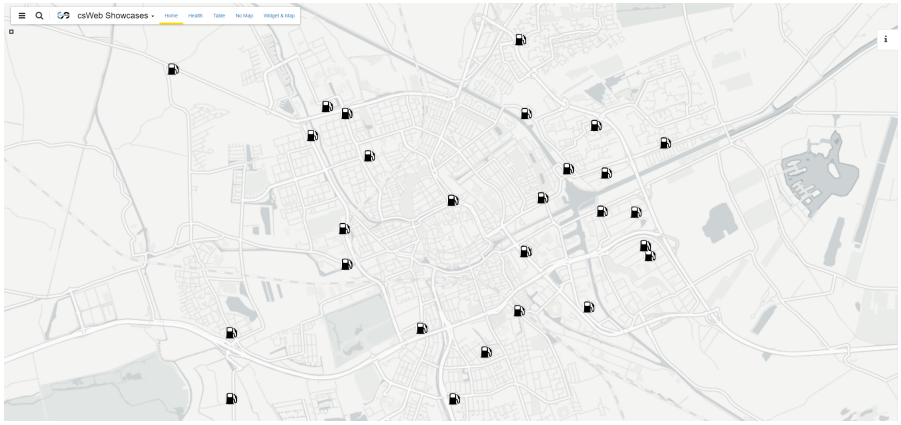


Figure 35: Visualization of the gas station layer, showing all gas stations in Groningen, the Netherlands.

Now, for every gas station in this layer, the user can right click it and simulate an Effects model at that specific location, which can be seen in Figure 36.

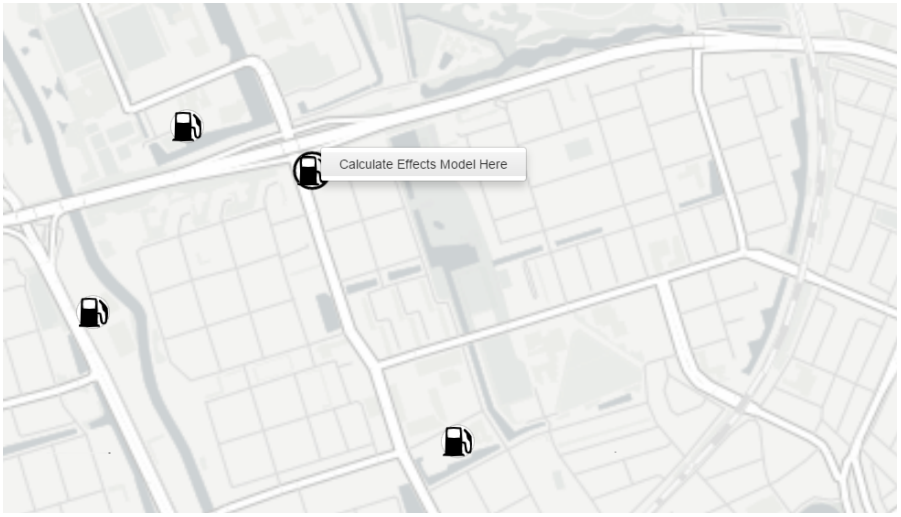


Figure 36: Every gas station can be right clicked, allowing an Effects model to be calculated there.

When the user clicks this button, two things happen simultaneously: all nearby building models that are in the same borough are loaded, and a menu is shown on the right side of the screen where the user can select parameters for the simulation. This can be seen in Figure 37.

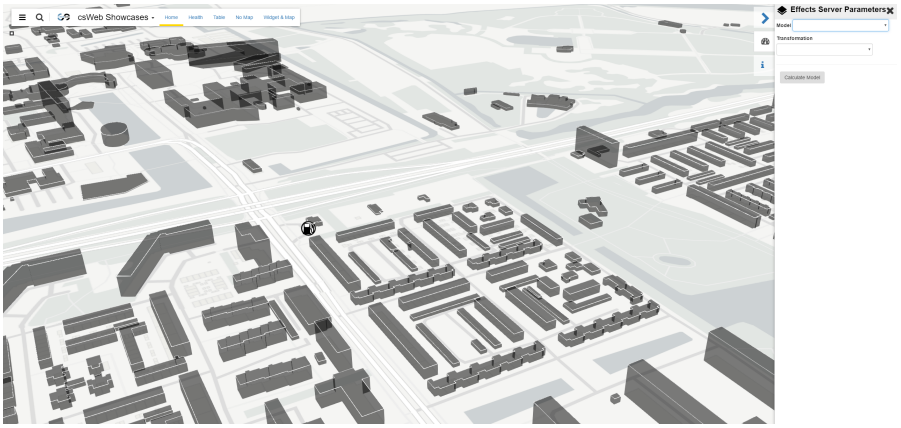


Figure 37: Nearby building models and a menu on the right side are loaded if the simulate button is clicked.

In the menu on the right side of the screen, the user is able to select which model to simulate at the location of the gas station, which return parameter to use (only "Toxic dose outdoors grid" is used at the moment), and which transformation to use. Currently supported transformations are a isolines transformation, and a raw point transfer which just passes the returning point cloud to the client to visualize. A short example of this can be seen in Figure 38. Not all options are visualized because there are a lot of options.

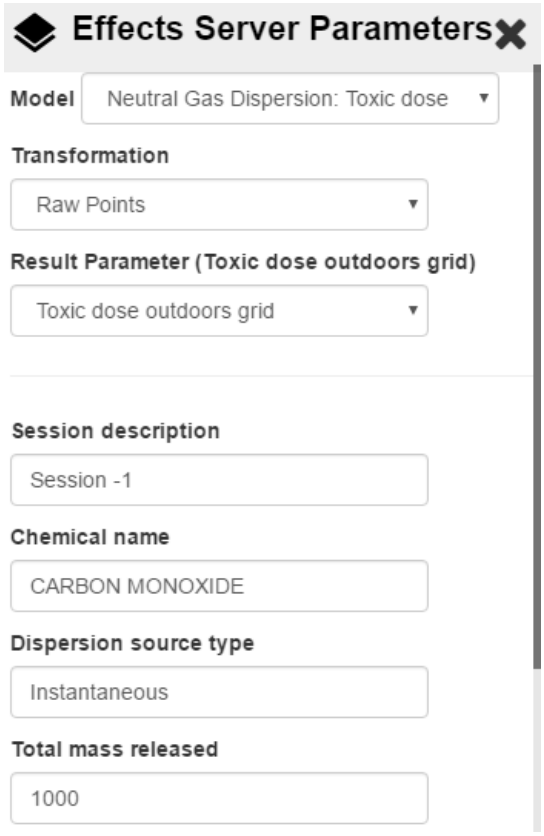


Figure 38: The first few options that the user can select in the Effects menu. Not all options are shown.

At the bottom of all those options is the button that simulates the model, which then sends a calculation request to the server. At this point, the server is calculating all different layers of the volume data that is about to be sent to the user. When the server is done, the results are sent to the client which immediately loads it, and visualizes it. The final combined result can be seen in Figure 39.

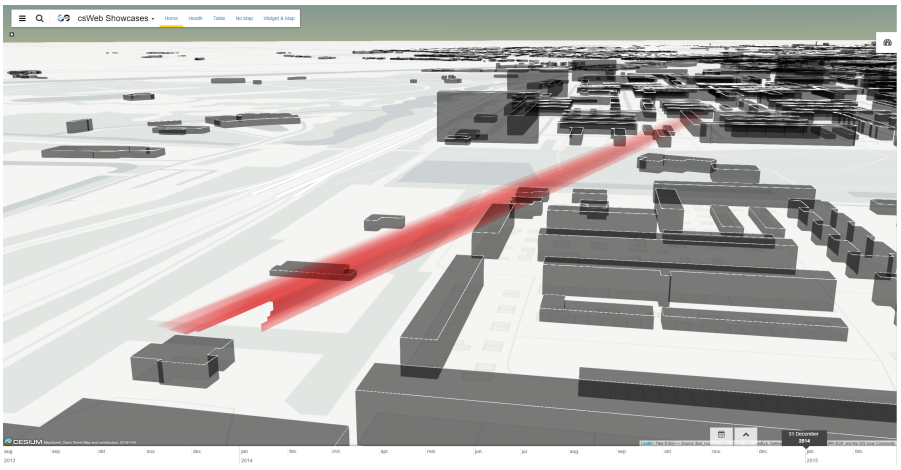


Figure 39: The resulting image of the volume ray cast point cloud, combined with the surrounding building models.

A second image that shows the result in a different, less urban, scenario can be seen in Figure 40.

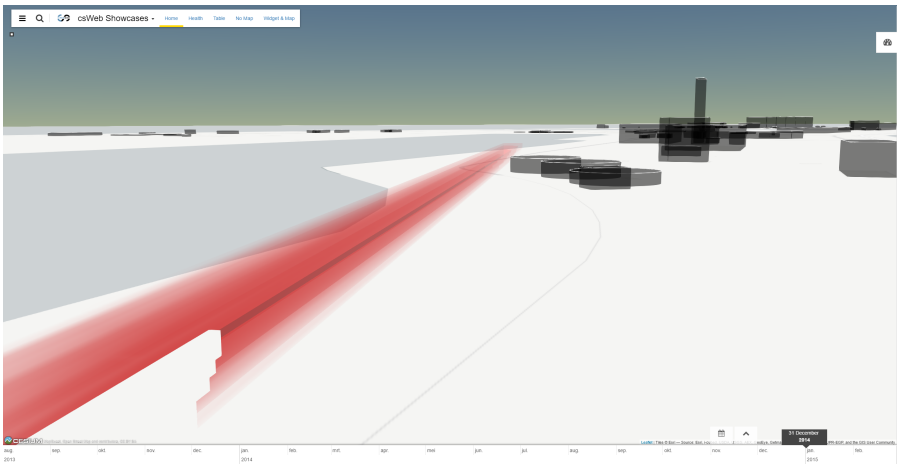


Figure 40: The resulting image of the volume ray cast point cloud, combined with the surrounding building models in a second scenario.

Note that there is an artifact at the beginning of the ray cast volume, it looks like there is some missing data.

7

SUMMARY AND CONCLUSION

In this project, 3D visualization was explored and the CommonSense framework was expanded with new functionality added by the implementation of Cesium. This resulted in new possibilities for the visualization capabilities of the entire framework. Building models from Top10NL were used, and even though they were low resolution they still fulfill the requirements for the sense of context.

The new functionality was tested using volume raycasting of a model simulated by Effects, visualized in combination with the surrounding building models. The combination with Effects was done in an interactive manner, where the user is able to alter simulation parameters before calculation. However, not all interactivity requirements were satisfied. For example, it was not possible to alter the color map of the ray cast volume.

The new 3D visualization of Effects data brings new possibilities to Effects. After a presentation for the Effects team, they were very enthusiastic about the results of this project. Also, CommonSense now supports 3D visualization which opens up an entire new domain of possibilities which is yet to be explored.

Concluding, there is a lot of new functionality for Effects and CommonSense, and this project is a good basis for this functionality, but there is a lot more to explore.

FUTURE WORK

In the future, some more exploration can be done in the field of the interactivity of the visualization. For example, realtime editing of the parameters of Effects would be interesting to see. Also, altering color maps of the visualization would have benefits.

It would be interesting to add a time element to the visualization, where the user is able to scroll through the simulation as it evolves over time. More models could also be supported.

A final point of interest is the data transfer between the Effects server and the client, where every layer has to be calculated individually, which can be done much faster. The team that maintains this server is already working on this.

BIBLIOGRAPHY

[1] Analytical Graphics Inc. A JavaScript library for creating 3D globes and 2D maps in a web browser without a plugin. http://cesiumjs.org/, 2015.

[2] Călin Arens, Jantien Stoter, and Peter Van Oosterom. Modelling 3D spatial objects in a geo-DBMS using a 3D primitive. *Computers & Geosciences*, 31(2):165–177, 2005.

[3] Bentley Systems, Incorporated. Microstation Geographics. http://www.bentley.com/en-US/Products/Bentley+Map/Migrating-to.htm, 2003.

[4] Frank Bignone, Olof Henricsson, Pascal Fua, and Markus Stricker. Automatic extraction of generic house roofs from high resolution aerial imagery. In *Computer Vision-ECCV 96*, pages 83–96. Springer, 1996.

[5] 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.

[6] Howard Butler, Martin Daly, Allan Doyle, Sean Gillies, Tim Schaub, and Christopher Schmidt. The GeoJSON format specification. *Rapport technique*, page 67, 2008.

[7] Stuart K Card, Jock D Mackinlay, and Ben Shneiderman. *Readings in information visualization: using vision to think*. Morgan Kaufmann, 1999.

[8] Eric Carlson. Three-dimensional conceptual modeling of subsurface structures. In *Auto-Carto*, volume 8, pages 336–345, 1987.

[9] Centraal Bureau voor Statistiek. Voorraad woningen en niet-woningen. http://www.cbs.nl/nl-NL/menu/themas/bouwen-wonen/cijfers/default.htm, 2015.

[10] Ed H Chi. A taxonomy of visualization techniques using the data state reference model. In *Information Visualization, 2000. InfoVis 2000. IEEE Symposium on*, pages 69–75. IEEE, 2000.

[11] Jeremy W Crampton. Interactivity types in geographic visualization. *Cartography and geographic information science*, 29(2):85–98, 2002.

[12] Stephan Diehl. *Software visualization: visualizing the structure, behaviour, and evolution of software*. Springer Science & Business Media, 2007.

[13] J. Engel, S. Pasewaldt, M. Trapp, and J. Dollner. An immersive visualization system for virtual 3D city models. In *Geoinformatics (GEOINFORMATICS), 2012 20th International Conference on*, pages 1–7, June 2012. doi: 10.1109/Geoinformatics.2012.6270289.

[14] ESRI. What is GIS? http://www.esri.com/what-is-gis, 2015.

[15] Lei Feng, Chaoliang Wang, Chuanrong Li, and Ziyang Li. A research for 3D WebGIS based on WebGL. In *Computer Science and Network Technology (ICCSNT), 2011 International Conference on*, volume 1, pages 348–351. IEEE, 2011.

[16] J.D. Foley. *Computer Graphics: Principles and Practice*. Addison-Wesley systems programming series. Addison-Wesley, 1996. ISBN 9780201848403. URL https://books.google.nl/books?id=-4ngT05gmAQC.

[17] Foraker Labs. Usability First Glossary:focus+context. http://www.usabilityfirst.com/glossary/focuscontext/, 2003.

[18] Wolfgang Förstner. A framework for low level feature extraction. In *Computer Vision-ECCV'94*, pages 383–394. Springer, 1994.

[19] Github. CommonSense Source Code Repository. https://github.com/TNOCS/csWeb, 2014.

[20] Google Incorporated. Google Earth. http://www.google.com/earth/, 2001.

[21] Michael Gruber, Marko Pasko, and Franz Leberl. Geometric versus texture detail in 3-D models of real world buildings. In *Automatic extraction of Man-made objects from aerial and space images*, pages 189–198. Springer, 1995.

[22] Armin Gruen and Xinhua Wang. CC-Modeler: a topology generator for 3-D city models. *ISPRS Journal of Photogrammetry and Remote Sensing*, 53(5):286–295, 1998.

[23] Tao Guo and Yoshifumi Yasuoka. Snake-based approach for building extraction from high-resolution satellite images and height data in urban areas. In *Proceedings of the 23rd Asian Conference on Remote Sensing. Kathmandu, Nepal, unpaginated CD-ROM*. Citeseer, 2002.

[24] Norbert Haala and Martin Kada. An update on automatic 3D building reconstruction. *ISPRS Journal of Photogrammetry and Remote Sensing*, 65(6):570–580, 2010.

[25] Alexandra D Hofmann, Hans-Gerd Maas, and André Streilein. Knowledge-based building detection based on laser scanner data and topographic map information. *INTERNATIONAL ARCHIVES OF PHOTOGRAMMETRY REMOTE SENSING AND SPATIAL INFORMATION SCIENCES*, 34(3/A):169–174, 2002.

[26] Mark St John, Michael B Cowen, Harvey S Smallman, and Heather M Oonk. The use of 2D and 3D displays for shape-understanding versus relative-position tasks. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 43(1):79–98, 2001.

[27] Kadaster. Basisregistraties Adressen en Gebouwen. https://www.kadaster.nl/bag, 2008.

[28] Kadaster. Top10NL Building Models. http://www.kadaster.nl/web/artikel/producten/TOP10NL.htm, 2014.

[29] A Krooks, J Kahkonen, L Lehto, P Latvala, M Karjalainen, and E Honkavaara. WebGL Visualisation of 3D Environmental Models Based on Finnish Open Geospatial Data Sets. *ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 1:163–169, 2014.

[30] Tim Lammarsch, Wolfgang Aigner, Alessio Bertone, Silvia Miksch, Thomas Turic, and J Gartner. A comparison of programming platforms for interactive visualization in web browser based applications. In *Information Visualisation, 2008. IV'08. 12th International Conference*, pages 194–199. IEEE, 2008.

[31] Ying K Leung and Mark D Apperley. A review and taxonomy of distortion-oriented presentation techniques. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 1(2):126–160, 1994.

[32] B Loesch, M Christen, and S Nebiker. OpenWebGlobe-an open source SDK for creating large-scale virtual globes on a WebGL basis. In *International Archives of the Photogrammetry Remote Sensing and Spatial Information Sciences XXII ISPRS Congress*, 2012.

[33] David G. Lowe. Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (5):441–450, 1991.

[34] Martti Mäntylä. An introduction to solid modeling. 1988.

[35] Chris Marrin. WebGL Specification. *Khronos WebGL Working Group*, 2011.

[36] BH McCormick. and engineering opportunities. *Computer Graphics*, 21:6, 1987.

[37] Oculus VR. Oculus Rift. https://www.oculus.com, 2015.

[38] OpenGIS Consortium. OpenGIS Simple Features Specification for SQL, revision 1.1. http://www.opengis.org/, 1999.

[39] OpenGIS Consortium. OpenGIS Specifications. http://www.opengis.org/, 2002.

[40] Wouter Pasman and Frederik W Jansen. Scheduling level of detail with guaranteed quality and cost. In *Proceedings of the seventh international conference on 3D Web technology*, pages 43–51. ACM, 2002.

[41] Harald Piringer, Robert Kosara, and Helwig Hauser. Interactive focus+ context visualization with linked 2d/3d scatterplots. In *Coordinated and Multiple Views in Exploratory Visualization, 2004. Proceedings. Second International Conference on*, pages 49–60. IEEE, 2004.

[42] Pitnet Bowes Software. MapInfo. http://www.mapinfo.com, 2003.

[43] Federico Prandi, Giulio Panizzoni, Daniele Magliocchetti, Federico Devigili, and Raffaele De Amicis. WebGL virtual globe for efficient forest production planning in mountainous area. In *Proceedings of the 20th International Conference on 3D Web Technology*, pages 143–151. ACM, 2015.

[44] Refractions Research. PostGIS, an PostgreSQL extension adding extra geometrical types to the DBMS. http://postgis.refractions.net, 2003.

[45] Nicholas Rego and David Koes. 3Dmol.js: molecular visualization with WebGL. *Bioinformatics*, 31(8):1322–1324, 2015.

[46] Rijkswaterstaat. Algemene Hoogtebestand Nederland, version 2. http://www.ahn.nl, 2014.

[47] Markus Schütz. Potree, a WebGL based point cloud viewer for very large datasets, based on Scanopy. http://potree.org/, 2015.

[48] Peter Shirley, Michael Ashikhmin, and Steve Marschner. *Fundamentals of computer graphics*. CRC Press, 2009.

[49] Terry A Slocum. *Thematic cartography and geovisualization*. Prentice hall, 2009.

[50] Harvey S Smallman, Mark St John, Heather M Oonk, and Michael B Cowen. Information availability in 2D and 3D displays. *IEEE Computer Graphics and Applications*, (5):51–57, 2001.

[51] Rebecca R Springmeyer, Meera M Blattner, and Nelson L Max. A characterization of the scientific data analysis process. In *Proceed-*

ings of the 3rd conference on Visualization'92, pages 235–242. IEEE Computer Society Press, 1992.

[52] JE Stoter and PJM Van Oosterom. Incorporating 3D geo-objects into a 2D Geo-DBMS. In *Proceedings FIG, ACSM/ASPRS, Washington DC, April 19-26, 2002*, 2002.

[53] JE Stoter and Siyka Zlatanova. 3D GIS, where are we standing? In *ISPRS Joint Workshop on'Spatial, Temporal and multi-dimensional data modelling and analysis', Québec, October, 2003*, 2003.

[54] A. Telea. Practical data visualization. Slides for the course Practical Data Visualization, University of Groningen, 2015.

[55] James J Thomas, Kristin Cook, et al. A visual analytics agenda. *Computer Graphics and Applications, IEEE*, 26(1):10–13, 2006.

[56] TNO: Nederlandse Organisatie voor toegepast-natuurwetenschappelijk onderzoek. CommonSense: A web-based open source GIS application, written in Typescript, with a focus on usability and connectivity. https://github.com/TNOCS/csWeb, 2015.

[57] TNO: Nederlandse Organisatie voor toegepast-natuurwetenschappelijk onderzoek. EFFECTS - Software for safety and hazard analysis. https://www.tno.nl/en/focus-area/urbanisation/environment-sustainability/public-safety/effects-software-for-safety-and-hazard-analysis/, 2015.

[58] Melanie Tory, Arthur E Kirkpatrick, M Stella Atkins, and Torsten Möller. Visualization task performance with 2D, 3D, and combination displays. *Visualization and Computer Graphics, IEEE Transactions on*, 12(1):2–13, 2006.

[59] Trimble Navigation. SketchUp. http://www.sketchup.com/, 2000.

[60] Trimble Navigation. Sketchup 3D Warehouse. https://3dwarehouse.sketchup.com/, 2005.

[61] Edward Verbree and PJM van Oosterom. The STIN method: 3D surface reconstruction by observation lines and Delaunay TENs. In *Proceedings of ISPRS Workshop on 3D-reconstruction from airborne laserscanner and InSAR data, Dresden, Germany*, 2003.

[62] Edward Verbree, Gert Van Maren, Rick Germs, Frederik Jansen, and Menno-Jan Kraak. Interaction in virtual world views-linking 3D GIS with VR. *International Journal of Geographical Information Science*, 13(4):385–396, 1999.

[63] Tom Vijlbrief and Peter van Oosterom. The GEO++ system: An extensible GIS. In *Proc. 5th Intl. Symposium on Spatial Data Handling, Charleston, South Carolina*, pages 40–50, 1992.

[64] Vladimir Agafonkin. Leaflet: An Open Source JavaScript Library for Mobile Friendly Interactive Maps. http://leafletjs.com/, 2015.

[65] George Vosselman. Building reconstruction using planar faces in very high density height data. *International Archives of Photogrammetry and Remote Sensing*, 32(3; SECT 2W5):87–94, 1999.

[66] Christopher Westphal and Teresa Blaxton. Data mining solutions: methods and tools for solving real-world problems. 1998.

[67] Christopher D Wickens, David H Merwin, and Emilie L Lin. Implications of graphics enhancements for the visualization of scientific data: Dimensional integrality, stereopsis, motion, and mesh. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 36(1):44–61, 1994.

[68] Guoqing Zhou, Zenyu Tan, Minyi Cen, and Chaokui Li. Customizing visualization in three-dimensional urban GIS via web-based interaction. *Journal of urban planning and development*, 132 (2):97–103, 2006.

[69] Siyka Zlatanova, Alias Rahman, and Morakot Pilouk. 3D GIS: current status and perspectives. *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences*, 34(4): 66–71, 2002.

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

http://code.google.com/p/classicthesis/

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

http://postcards.miede.de/

Final Version as of August 22, 2016 (`classicthesis` version 4.1).