

Service-Driven 3D Atlas Cartography

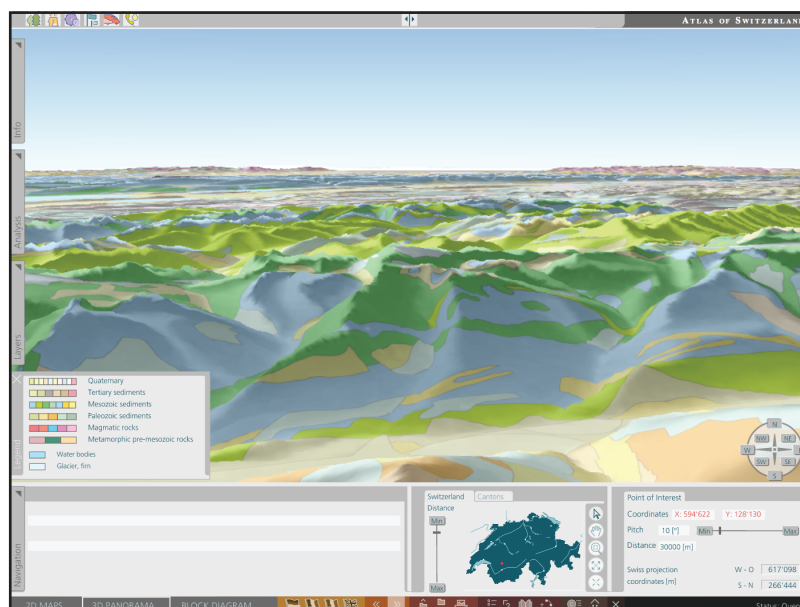
Master Thesis, Spring Term 2012

Geomatics and Planning Msc

Author

Nadia Panchaud

nadiap@student.ethz.ch



Supervisor

Prof. Dr. Lorenz Hurni

Supervising Tutors

Dr. Ionut Iosifescu-Enescu

Dr. René Sieber

July 2, 2012

|||||

Institute of Cartography and Geoinformation

Department of Civil, Environmental and Geomatic Engineering | ETH Zürich

Acknowledgement

This Master thesis has been realized at the Institute for Cartography and Geoinformation at ETH Zürich in the framework of my Master studies in Geomatics and Planning. It has given me the chance to deepen my knowledge about the stimulating and fast-evolving field of online cartography and its application to atlases. I have been able to immerse myself in the new challenges brought by the digital technologies to the atlas field. I have especially enjoyed exploring the world of 3D visualization and the incredible perspectives it offers, not only for atlases, but also for any types of geovisualization.

I would like to thank Dr. Ionut Iosifescu-Enescu for helping me defining the topic of my Master thesis, as well as for his unreserved support during this thesis. I could always count on his knowledge and competences to steer me in the right direction when I found myself in front of an obstacle. My thanks also go to Dr. René Sieber for his help regarding online digital atlases and to Remo Eichenberger for the captivating discussions about the Globe Capture Service and issues with 3D visualization as well as for his technical support.

I also thank my friends, Elisabeth and Olga, that had the patience and willingness to read and correct my report.

Finally, I would like to thank my family for their unconditional support during my studies in Zürich. In particular to my dad who is always there for me (and my computer) and to my mom who is always there to cheer me up.

Abstract

This Master thesis deals with the new and exciting field of 3D atlas cartography. Its goal is to assess the state of 3D geovisualization and to implement a prototype demonstrating that a service-oriented architecture for 3D atlas is possible. The literature review shows that 3D geovisualization has certain advantages over 2D geovisualization regarding shape understanding and orientation tasks. From the specifications and web services review, it is clear that the area of 3D visualization is growing rapidly and that standards for 3D web services are demanded to put some order in the profusion of formats and implementations, that are not compatible with each other.

The prototype uses a combination of a 3D web service, Globe Capture Service, which is based on osgEarth, and a Web Map Service, QGIS Server to display a panorama view of the landscape. The Web Map Service is used as a texture. The prototype also implements block diagrams using WebGL and a Web Map Service for the texture. The limitations of WMS for 3D geovisualization appears when dealing with point symbology, but using a Web Map Service is still a valid choice as a texture for the landscape with both the Globe Capture Service and WebGL. The Globe Capture Service still encounters performance and interoperability issues, especially due to the lack of standards, but otherwise shows promises for a 3D web atlases. WebGL, though, has two major weaknesses that could hinder its use for a full-fledge 3D web atlas: the size limitation and the absence of full support by most of the browsers.

The prototype implementation shows that service-driven 3D atlas are doable and that the use of a Web Map Service is suitable for the texture of topographic and choropleth maps. Further, the prototype and the literature come to the conclusion that standards for 3D web services are needed in order to allow for a full and interoperable service-oriented architecture. Regarding the 3D symbols and point symbology, that cannot be properly handle by a Web Map Service, it is therefore advice to use billboards which are requested through a 3D/2D symbols web service. There is also still need for research for the most effective display for 3D atlas, being in the form of panorama view, block diagram or virtual globe.

Table of Content

Preface	I
Abstract	III
Registers	IX
<i>Figures</i>	<i>IX</i>
<i>Tables</i>	<i>X</i>
<i>Code Extracts</i>	<i>X</i>
Abbreviations	XI
1. Introduction	1
1.1. Introduction and Problem Statement	1
1.2. Motivation	1
1.3. Goals	2
1.4. Structure of the Report	2
2. Methodology	3
2.1. Method	3
2.1.1. Analysis	3
2.1.2. Implementation	3
2.1.3. Discussion of Results	3
2.2. Data	3
2.3. Infrastructure	4
3. Literature Review	5
3.1. Atlases and 3D Visualization	5
3.1.1. Atlas Definition	5
3.1.2. Advantages and Challenges of 3D Visualization	6
3.2. Atlas Functions	7
3.2.1. Functions of Interactive Atlases	7
3.2.2. Functions Specific to 3D Atlas	8
4. Technology Review	9
4.1. Existing Online 3D Atlases and 3D Visualization Products	9
4.1.1. 3D Atlases and 3D Viewers of Geoportals	9
4.1.2. Virtual Globe	12
4.1.3. GIS-based Viewers	14
4.2. 3D Technologies Review	15
4.2.1. Graphic Formats	15
4.2.2. 3D Viewers	17

4.3. Web Services	17
4.3.1. Web Map Service (WMS)	18
4.3.2. Web Terrain Server (WTS)	18
4.3.3. Web Perspective View Service (WPVS)	18
4.3.4. Web Perspective View Service++ (WPVS++)	18
4.3.5. Web View Service (WVS)	18
4.3.6. Web 3D Service (W3DS)	19
4.4. Architectures	19
5. Requirements for Service-driven 3D Atlases	23
5.1. System Requirements	24
5.2. Visualization Requirements	24
5.3. Interactivity Requirements	25
6. Implementation	27
6.1. Architecture	27
6.1.1. Data Tier	27
6.1.2. Web Service Tier	29
6.1.3. User Interface Tier	30
6.2. High-level Workflow	31
6.2.1. The Panorama View	31
6.2.2. The Block Diagram	31
6.3. Presentation of the Prototype	32
6.3.1. General Description	32
6.3.2. Symbolization with WMS	35
6.3.3. Technical Aspects	35
7. Discussion of Results	37
7.1. Overview	37
7.2. Advantages and Weaknesses of the Used 3D Technologies	37
7.2.1. Combination of Globe Capture Service and QGIS Server	37
7.2.2. Combination of WebGL and QGIS Server	40
7.2.3. Summary	42
7.3. Recommendations for the Architecture	43
8. Conclusions and Outlook	45
8.1. Conclusions	45
8.2. Outlook	45
Bibliographic References	47
Appendices	51
I. <i>Illustrations of the Prototype</i>	51
II. <i>Panorama View Integration (getGlobeCapture.js)</i>	58

Table of Content	VII
III. <i>Block Diagram Integration (getBlockDiagram.js)</i>	63
IV. <i>Extract of the WebGL Integration (webGL.js)</i>	66
V. <i>Integration of WMS in the Globe Capture Request</i>	68
VI. <i>Examples of WMS Request for the Block Diagram</i>	70
VII. <i>Vertex and Fragment Shaders Integration</i>	71
VIII. <i>Functions of Existing 3D Digital Atlases</i>	72
IX. <i>Requirements Overview</i>	73
Declaration of Originality	74

Registers

Figures

Figure 1: The three core concepts	1
Figure 2: Block diagram and virtual globe from the Swiss World Atlas interactive	9
Figure 3: Viewer of the Géoportail. Relief view with the layers roads, orographic names, national maps (transparent) and aerial image	10
Figure 4: Corresponding views in 2D and 3D of the GeoPortal Bund	11
Figure 5: 2D Viewer of the IDEE	11
Figure 6: Panorama view and block image in the Atlas of Switzerland 3	12
Figure 7: Zoom in on Switzerland on the virtual globe of NASA World Wind (module WMS), with the isoline of the magnet field strengths from the WMS of the Swiss government	13
Figure 8: Close zoom in on the demo version of the OpenWebGlobe	14
Figure 9: Web service concept	17
Figure 10: 3D Visualization Pipeline according to the OGC	19
Figure 11: Balancing schemes between portrayal servers and clients according to the OGC	20
Figure 12: First family of architectures, using plugins to display the data in the browser	21
Figure 13: Architecture of a service-oriented subsystem	21
Figure 14: Architecture of the WTS, as realized by the open source system Deegree	22
Figure 15: Requirements for service-driven 3D atlases in their fields of influence	23
Figure 16: Architecture of the prototype	27
Figure 17: Parameters of the Globe Capture Service request	29
Figure 18: Flow of information for the panorama view mode	31
Figure 19: Flow of information for the block diagram mode	32
Figure 20: WebGL rendering pipeline	32
Figure 21: The three modes of the prototype. Top: 2D maps, left: panorama view, right: block diagram	33
Figure 22: Design of the GUI for the panorama view mode and its navigational functions	34
Figure 23: Design of the GUI for the block diagram mode and its navigational functions	34
Figure 24: Panorama view, left: bas map, right: geology map	38
Figure 25: Panorama view, top: population density (choropleth) and number of inhabitants (sphere), left: diagram for boat types, right: airport symbol and labels on the base map	39
Figure 26: Block diagrams, left: base map, right: population density and inhabitants	41
Figure 27: 3D symbolization of the buildings in Andermatt with WebGL	41
Figure 28: Billboards - possible solution for 2D and 3D symbols with the web service approach	43

Tables

Table 1: Characteristics of Virtual Globes and possible import formats	15
Table 2: List of layers used for the prototype and their name in the PostgreSQL database	28
Table 3: Extensions to the GUI, functions and description	36
Table 4: Assessment of the different requirements for 3D geovisualization systems	42

Code Extracts

Code Extract 1: iFrame and canvas elements	31
Code Extract 2: Example for the symbolization of alluvial deposits (Alluvionen) with SLD	35

Abbreviations

AdS	Atlas der Schweiz, Atlas of Switzerland
API	Application Programming Interface
CityGML	City Geography Markup Language
DEM	Digital Elevation Model, DHM
DHM	Digitales Höhenmodell, DEM
DOM	Document Object Model
GCS	Globe Capture Service
GML	Geography Markup Language
GPB	GeoPortal Bund, Germany
GPF	Géoportail, France
GPU	Graphics Processing Unit
GUI	Graphic User Interface
IDEE	Infraestructura de Datos Espaciales de España
KML	Keyhole Markup Language
OGC	OpenGIS Consortium, Inc.
OpenGL	Open Graphics Library
QGIS	Quantum GIS, open source software for geographic information system
SLD	Styled Layer Descriptor
SOA	Service-Oriented Architecture
SVG	Scalable Vector Graphic
SWAi	Schweizer Weltatlas interaktiv, Swiss World Atlas interactive
VRML	Virtual Reality Modeling Language
X3D	Extensible 3D Graphics
X3DOM	X3D + DOM
XML	Extensible Markup Language
W3C	World Wide Web Consortium
W3DS	Web 3D Service
WebGL	Web Graphics Library
WMS	Web Map Service
WPVS	Web Perspective View Service
WTS	Web Terrain Service
WVS	Web View Service

1. Introduction

1.1. Introduction and Problem Statement

We live in a world where visual transmission of information have a prominent role and where computers and information technologies are always more powerful, thus allowing 3D visualization and display to become more present in our daily life. Indeed, we now have 3D movie theaters, 3D TVs, and 3D game consoles. 3D technologies and portrayal techniques are now available not only to the experts, but also to the general public, and, for instance, for 3D atlases and virtual globes as well. And our knowledge is getting online, including such spatial knowledge and thus we need solutions for 3D geovisualization online.

Atlases are powerful tools to display, explore and share geospatial information. From the classical and static paper atlases, the technologies have allowed to move to more dynamic ways of displaying geospatial data on computers and to publish them through the World Wide Web. It allows the users to access and interact with geospatial information like never before, but from the paper copy to the 3D fully interactive web atlas, there is still a long way to go. For several geospatial tasks, 3D visualizations of landscape and other geospatial information might be more intuitive and easier to understand for the non-expert users as they are closer to how people see their world (Bleisch and Dykes, 2006; Meng, 2003; Rase, 2003; St. John *et al.*, 2001). Additionally, delivering them over the internet enables a greater number of users to access and explore geospatial information, including atlases. Web atlases can be seen either as a single product or as a complement to a paper copy (e.g. the Swiss World Atlas and its interactive version¹), but both options bring advantages regarding interactivity and richness of content.

The idea of this Master thesis is to look into the possibilities of service-driven (as opposed to client-only approach) 3D atlas cartography and to implement a prototype showing how such an atlas can be done and what are the challenges and advantages involved.

1.2. Motivation

Although 3D atlases do exist and although service-driven web cartography is already being used for 2D visualization, the combination of the 3D technologies and the service-driven concept has not been broadly explored and developed yet (Sieber *et al.*, 2011). Whether service-oriented architecture is appropriate for 2D and 3D web atlases is still to be investigated and these web atlases, 3D or not, have to be able to deliver comparable maps in terms of quality to desktop-based atlases (Iosifescu-Enescu, 2011). Despite that challenge, rendering 3D geodata in 3D display not only allows to keep all the information regarding the space, but moreover it creates the opportunity to use 3D symbology. 3D display and 3D symbology enable to display spatial and thematic information in new ways, taking full advantages of the possibilities that 3D technology offers. The concept of modern atlas could highly benefit from its combination with three dimensional visualization and service-oriented architecture.

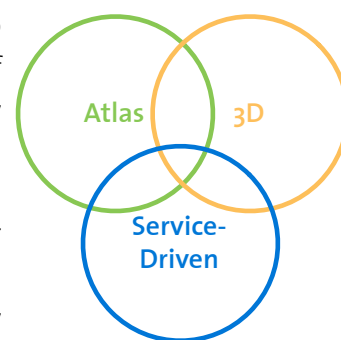


Figure 1: The three core concepts

¹ <http://schweizerweltatlas.ch>

3D data for atlases are available and solutions for their visualization already exist, however 3D technologies and data bring additional challenges regarding storage capacity, processing power, and accessibility. Thus, a service-driven approach could help solving these issues through remote storage and processing as well as facilitating the access. This approach additionally aims at interoperability or the ability and possibility to integrate data from diverse sources and to chain different web services while displaying the results at once. Furthermore, the updating process becomes simpler as only the database on the server have to be modified, while the service generates the new map on the fly. Thus, there is a gain in productiveness and a reduction of work related to map editing and updating. However, one problem remains as there is no standardized specification for 3D web service agreed upon yet.

The inspiring challenge of this work is to build a prototype with advantages that the 3D functionalities offer regarding geospatial data, as well as the advantages that the service-driven approach brings regarding the access and the handling of data.

1.3. Goals

The first part involves two main objectives. The first one is to make a general assessment of existing 3D atlases and of their functionalities, while exploring the literature for a complete view of possible functions in 3D atlases. It should lead to a list of requirements regarding 3D atlases. The second objective is to explore the different technologies and specifications available for service-driven 3D data portrayal and to assess their strengths and weaknesses.

In a second step, a prototype of service-driven 3D atlas will be implemented based on the findings in the first part and on the constraints of this work. This prototype should show that service-driven 3D atlases are possible as panorama view and as block diagram. The discussion of the results should bring into light advantages and future challenges as well as recommendations for service-driven 3D atlas.

1.4. Structure of the Report

After a short **introductory chapter (1)** that explains the current situation, the motivation and goals of this project, chapter two deals with the **methodology (2)**. It goes over the different steps followed, from the analysis part to the discussion of the results, as well as over the origin of the data and the infrastructure used for the project.

Then, chapters three and four concerns the analysis part itself. Chapter three reviews the **literature (3)** regarding atlases and their functionalities and analyze the issues regarding 3D visualization, whereas chapter four deals with the **technology (4)** review for digital visualization. Chapter five defines the **requirements (5)** for service-driven atlases.

Based on the previous chapters, chapter six explains the **implementation of the prototype (6)** and describes the product obtained.

Chapter seven discusses the **results (7)** and draws from the previous chapters to suggest recommendations for service-driven 3D atlases.

Concluding chapter eight focuses on the **conclusions and outlook (8)** of the project as a whole and discusses some future prospects.

2. Methodology

2.1. Method

This work is built on two different parts. First, an analysis is conducted based on existing literature and previous works in the subject to assess the present situation in the area of 3D atlases and 3D web technologies. Second, based on the first part, a prototype of a service-driven 3D atlas will be implemented to demonstrate the potential of such atlases.

2.1.1. Analysis

The analysis is conducted in two different areas, literature and technologies reviews, to define a set of requirements for service-driven 3D atlases. First, it looks into the literature and reviews the definition of atlases and 3D visualization. It draws from the literature to assess advantages and challenges of 3D visualization. Furthermore, it discusses the different functionalities found in digital atlases, especially the ones specific to 3D atlases. Second, it reviews the existing analyzes existing 3D atlases, other 3D data viewer products. Then, it evaluates the different technologies and specifications available for service-driven 3D portrayal. The analysis draws a broad picture of what has been done till now in the area of 3D web atlases and 3D online portrayal and conducts a comparison of available 3D technologies and specifications based not on advantages and weaknesses, but also appropriateness for atlases. Finally, it defines and explains the requirements for service-driven 3D atlas.

2.1.2. Implementation

In the implementation step, it will be decided what technologies will be used for the prototype. The decision will be based on the analysis part and the limitations of this work scope. A service-driven 3D atlas prototype will be implemented based on the four following elements: a Graphic User Interface, a Web 2D Service, a 3D Web Service, and WebGL. Concerning the latest, it will first only explore the possibilities of its implementation for bloc diagrams and then, if feasible, the technology will be implemented within the prototype.

2.1.3. Discussion of Results

The main results of the analysis parts and the experience resulted from the prototype will be discussed and used to describe an ideal architecture of a service-driven 3D atlas. It will also point out weaknesses and strengths of such architecture.

2.2. Data

The implementation prototype uses data from the Atlas of Switzerland 3 (AdS), which is developed at the Institute of Cartography and Geoinformation at the ETH Zurich, jointly with the Federal Office for Statistics. The actual atlas is currently available on DVD, but not through the internet. These data will be used exclusively for the purpose of this thesis and will be deleted after the thesis is submitted. The use of the swisstopo data is regulated by the terms on use available at http://geodata.ethz.ch/geovite/Infoblatt_Nutzungsbestimmungen_Swisstopo_EN.pdf. A selection of the dataset from the AdS, consisting of basic geometries and thematic data in 2D, has already been processed and set up to be used by a Web Map Service (WMS) in (Ortner, 2011) and constitute the available dataset for this work. The Data Elevation Models (DEM) that are required to build the block diagrams, belong to swisstopo as well and are recovered through GeoVite².

² <https://geodata.ethz.ch/geovite/>

2.3. Infrastructure

The research, the report and the prototype have been realized on my personal computer (MacBook Pro) as well as on the computers in the StudiLab and the computer with the Globe Capture Service server at the Institute of Cartography and Geoinformation, ETH Zürich.

For editing SVG and JavaScript documents, the software *Oxygen 13.2* was used. The symbolization of the data was achieved using *QuantumGIS 1.1.0* and *1.7.4* and the plugin *Publish To Web*³. The figures for the report used *OmniGraffle Professional 5.4*, *Illustrator CS 5* and *Photoshop CS 5* and the report was written with *Pages iWork 09*.

³ developed internally at the Institute of Cartography and Geoinformation at ETH Zürich (Iosifescu-Enescu, 2011)

3. Literature Review

3.1. Atlases and 3D Visualization

This section goes over the definitions of atlases and of 3D visualizations. It first provides with an evolution of atlases through the ages and then, deals with the advantages and drawbacks of 3D visualization.

3.1.1. Atlas Definition

The word «atlas», which is the name of a the Greek god that carries the world, was first used by the cartographer Mercator in the sixteenth century to describe his cosmography. In the eighteenth century, it refers to a collection of single maps systematically organized and picturing the whole earth. A general definition was later agreed upon in the twentieth century and atlases were seen as a collection of maps with a specific purpose and organized in the form of a book, which usually includes tables, graphs and text (Ramos and Cartwright, 2006). They usually cover a spatially (from the world to the city level) and thematically (either very specific, e.g. fish species atlas, or very diverse) defined area (Borchert, 1999). Classical atlases in paper form have a fixed linear structure and a fixed format and are not just any book with maps:

«To the layman, any book consisting mainly of maps is an atlas, but technically to the geographer, no cased collection of maps deserves the name unless it be comprehensive in its field, systematically arranged, authoritatively edited and presented in a unified format.» (Alonso, 1968)

With the emergence of digital atlases and computer science, defining atlases as mentioned above was not sufficient anymore. Thus, the cartography field looked for an expanded and more flexible definition of atlases. Further definitions included the idea of intention, of specific purpose, of narrative faculty and of systematic and coherent collection of maps. Now that the purposes of atlases are wider and more mixed and that users have the possibility of generating their own maps, the remaining core concepts are their narrative faculty and the intentional combination, not necessarily of maps, but of processed spatial data (Ramos and Cartwright, 2006) either on paper or on other media. However, atlases should not be just a simple collection of geoinformation data. They should aim at cartographically well designed maps, that not only highlight the characteristics of the information displayed, but that as well allow for exploration (Sieber *et al.*, 2011).

Digital atlases become then atlases that are specifically developed for digital media, either as standalone product or as a complement to a paper copy. Digital atlases allow for the use of multimedia, which leads to the fact that the terms «digital» and «multimedia» atlases are sometimes used as synonym in opposition to paper atlases. Digital atlases can be built on the view-only concept or offer significant interactivity. The main characteristics of interactive multimedia atlases are their ability to generate user-demanded or user-created maps thanks to the access to the styling options and to the data behind the map (Kraak *et al.*, 2007). It additionally allows for a wide panel of interaction and navigation tools, while combining different media: digital atlases can be highly dynamic and interactive.

3D atlases can be defined as digital atlases that offer, on one hand, a changing spatial viewpoint (3D navigation) and, on the other hand, three-dimensional topographic views with overlapping thematic data and or statistical visualization where the height represents the value of the variable (Persson *et al.*, 2006). A three-dimensional topographic view can be a panorama, a bloc diagram or a virtual

globe. Among digital atlases, a clear separation has to be done between online atlases and desktop atlases (usually available on DVDs). The latest ones are closed systems, as book atlases, and only up to date at the time of making. Whereas online atlases allow an easy access through the web with no distribution costs and they can be updated at any time. Furthermore, they can play the role of a geoportal when integrated in a national geodata infrastructure (Kraak *et al.*, 2007).

3.1.2. Advantages and Challenges of 3D Visualization

The advantages of 3D visualization over 2D visualization are still contentious and not as straight forward as one might think. However, several authors agree upon the fact that the advantageous characteristics of 3D visualization depend on the task at hand too. It is also important to mention that 3D visualization is here understood as 3D perspective views on a 2D surface (e.g. a computer screen), but that are perceived as 3D by the users.

The first characteristic of 3D visualization mentioned in the literature as an advantage concerns its naturalistic display. It is supposed to make 3D views easier to understand for the non-experts because it looks more similar to the real world (Beard *et al.*, 2005; Bleisch and Dykes, 2006; Rase, 2003). Thus, the non-experts, or non-mappers, intuitively comprehend the views because it is first more compelling and second the three dimensions are integrated (St. John *et al.*, 2001).

Regarding the cognitive, the «cognitive distance» between the real world and the representation is absent in 3D representations (Meng, 2003). It is thought that, because the third dimension does not have to be reconstructed from two dimension representation (e.g. isolines), it simplifies the cognitive process, especially for non-expert users (Rase, 2003).

The literature agrees on the fact that 3D visualization may be more helpful for shape understanding and orientation tasks, because it allows the users to gain a better comprehension of the landscape and its general form. It can also happen that 3D geodata lose some of their characteristics and relations to other features once displayed in 2D and thus it might distort the understanding and analysis of the real features (de Vries and Zlatanova, 2004). 3D visualization is specially useful for the qualitative understanding and surveying of space, as well as for approximative navigation (Bleisch and Dykes, 2006; Bleisch and Nebiker, 2008; St. John *et al.*, 2001; Tory *et al.*, 2006). The users more effectively grasp the layout of a scene with the help of depth cues, such as shadows, scaling and distortions. Additionally, texture and shading help for the pure 3D understanding of the view (St. John *et al.*, 2001).

Although 3D representations seem to have an upper hand regarding shape understanding tasks, it is not the case for tasks requiring relative positioning or location of several objects (Bleisch and Nebiker, 2008; St. John *et al.*, 2001). It seems that users do not accurately interpret variations in the landscape accordingly, for instance slope or exposition. Users also perform more slowly and less accurately for precise tasks, when using 3D representations than when using 2D representations (Bleisch and Dykes, 2006). This is partially explained by the fact that 3D space in 3D representations is non linearly distorted, which leads to foreshortening issues and some ambiguity regarding object location. It makes 2D representations more effective to judge of distance and angle between objects (St. John *et al.*, 2001). However, the question of the role of the interface for the usability of 2D and 3D representations was raised, and it seems that when the interface does not fit the purpose, both types of representation become quite useless (Beard *et al.*, 2005; St. John *et al.*, 2001). It emphasizes again the importance of an adapted and friendly user interface, as well as functions that support a good usability of the maps (Meng, 2003), even more when the target users are not trained for visual analysis and interpretation, as it is often the case for atlases.

One important remaining challenge for 3D visualization are the technologies and specifications for 3D. Although many softwares for 3D processing and visualization of geospatial data are available, the choice for online visualization is more restricted and might require to install a plugin into the browser (see more details in chapter 5).

3.2. Atlas Functions

This section discusses the different functions of interactive atlases, with an emphasis on functions that are specific to 3D Atlas. In many ways, digital 3D and 2D atlases are quite alike regarding their functionalities, apart from the third dimension which adds supplementary complexity for spatial navigation and analysis.

3.2.1. Functions of Interactive Atlases

The one aspect that mainly differentiates digital multimedia atlases from paper atlases is the interactivity and the functionality they offer. Several authors suggested different ways of classifying functions available in interactive atlases. Ormeling (1997) divided interactive functions in nine groups: general functions, navigational functions, map functions, database functions, atlas functions, educational functions, cartographic functions, map use and analysis function, and remaining functions. Whereas the functions for the AdS are classified into five categories: general functions, functions for thematic navigation, functions for spatial navigation and orientation, visualization functions and GIS functions (Bär and Sieber, 1997). Later Hurni (2005) went back to the classification from Ormeling, while completing and expanding it. Cron explored the classification of interactive functions for her work on the GUI of the AdS and adapted the division of Bär and Sieber (Cron, 2006). This classification proved its usefulness and adequateness for a later work on digital atlases and web service (Ortner, 2011). The Cron's classification is used in this work for two reasons: first, it was built on the previous classification and specially oriented towards the structure of a GUI (Cron, 2006), and second, it already offered a useful orientation tool for the implementation of service-driven atlas prototype (Ortner, 2011).

The classification has five main groups, which are divided in different subgroups. This short overview of the five groups and subgroups does not pretend to be exhaustive, but aims at laying down the different types of functions that one might encounter in an interactive atlas.

The *general functions*, which were present in all previous classifications, consists of non-atlas specific functions that are available at all time in the interface. Functions such as import, export, home, exit, preferences and help are part of the general functions. They are considered as granted and should be present in any digital atlas (Ormeling, 1997).

According to the classification from Cron (2006), the *navigational functions* encompasses three subgroups: spatial navigation, thematic navigation and temporal navigation. By *spatial navigation* are understood the functions allowing the users to orientate themselves with the help, for instance, of a reference map and to navigate the map by zooming in and out and panning. It can also refer to more complex orientation information such as coordinates, altitude, and direction of view and allow searching for geographic locations. Functions for the *thematic navigation* enable the users to chose and change the theme and might include a search tool for the themes and a theme index. Finally, the *temporal navigation* tools offer functions to manage the temporal dimension of the data, such as on a time axis or control buttons for animation.

The *educational functions*, also called didactic functions, are divided into *explaining functions*, such as additional text or graphics, and *self-check functions*, such as quizzes and games.

The *cartographic and visualization functions* are what makes the interactive atlases attractive and rich regarding data display and the exploration of the data. They are made of three subgroups: map manipulation, redlining, and explorative data analysis. *Map manipulation* tools allows to turn on and off layers and legend categories as basic functionalities. Further functions comprise map comparison, projection changes and modification of the symbolization. *Redlining* is the ability to modify the maps with annotations, drawings and other marks without modifying the original data. *Explorative data analysis* allows the users to explore and compare spatial distributions of phenomenon in a visual manner. Functions such as modification of the classification and selection of specific data for the analysis are part of this subgroup.

Finally, Cron (2006) follows the classification of Bär and Sieber and classify the *GIS functions* as an independent groups. It regroups functions that Hurni or Ormeling classified differently. The first subgroup concerns *spatial query functions*, such as coordinates and altitude of a point, measurement of distance and surface and drawing of profiles. There are also *thematic query functions* that allow the users to access information about the attributes of the spatial entities and to get statistical tables. The last subgroup concerns *analysis functions* oriented towards surface analysis, such as slope and orientation and aggregation or intersection of data and spatial entities.

How far these different types of functions are implemented in the atlas can impact the choice of technologies suitable for the project, because the choice of data type and technologies can limit or hinder the implementation of certain functions. For instance, raster image of a 3D view cannot provide the users with attributes related to what is on the view. The SWAi solved that by adding symbols for cities, mountains and volcanoes providing information through tooltips.

3.2.2. Functions Specific to 3D Atlas

Some functions only appear or take all their importance with 3D visualization. The main group of functions that plays a significant role for 3D visualization is the spatial navigation. The third dimension adds new navigation parameters to handle this additional dimension. A 3D panorama usually requires that a point of observation, often called the camera, be set. The pitch determines the angle between the viewer or camera and the point of interest. An 0° pitch means that the viewer and the point of interest have the same elevation and a 90° pitch means that the viewer is directly above the point of interest. The yaw is the azimuth, meaning in which direction the viewer is looking and a yaw of 0° is the North. The angle of view sets the width of the scene, commonly called field of view too (Singh, 2001). The distance to the point of interest and the altitude of the viewer are also relevant. The Pan function is very important because it gives the possibility to the users to navigate in a very intuitive way, as if they could play with a model.

It is important to mention that not all functions are implemented in all atlases, both 2D and 3D. The general functions and some spatial and thematic navigations are usually present, whereas GIS functions and cartographic functions are in comparison less often offered (Cron, 2006; Ortner, 2011).

Regarding available 3D web atlases and geoportal (see details in chapter 5), one can see that the 3D functionalities are still not well developed and that the implementation of complex functions only come with more mature and complete products, such as the AdS, whereas only basic functions are present in test and beta version (see table in appendix VIII).

4. Technology Review

4.1. Existing Online 3D Atlases and 3D Visualization Products

This section explores the already available 3D web atlases and gives an overview of their main characteristics. 3D web atlases are not legion, far from that. Therefore, other online 3D visualization tools are included in the analysis. These includes virtual globes, 3D viewers and GIS-based viewers. Additionally, the analysis takes into account Atlas of Switzerland 3, which is a non-web 3D atlas, because its design and tools are considered as a best-practice example for a national atlas⁴.

4.1.1. 3D Atlases and 3D Viewers of Geoportals

3D atlases aim at fulfilling the traditional role of atlases while taking advantages of the interactivity and 3D possibilities offered by digital means, whereas the viewers of geoportals aim at displaying the information available, often at the national level within the framework of a national geodatabase or geodata infrastructure. The later ones usually offer raw thematic data and do not provide with classification of the data, nor additional information.

Swiss World Atlas interactive SWAi (Switzerland)⁵

The SWAi is a product of the Institute of Cartography and Geoinformation IKG at ETH Zurich. It offers 2D and 3D displays, including virtual globes and block diagrams, of a wide selection of thematic maps. It offers 2D maps ranging from a world view to local maps for selected locations. It completes the printed version of the Swiss World Atlas (SAW) and follows the same themes for the maps. The option globe is offered for twenty-five thematic maps and seven block diagrams are available for locations with topographic characteristics of interest (as of 16.03.2012). The virtual globe option is based on the NASA World Wind and the block diagrams have been developed in-house (Hurni *et al.*, 2011). The atlas graphic user interface is offered in four languages (German, French, Italian and English), but the legend is for now only in German. New maps are continuously added and the legends are being translated.

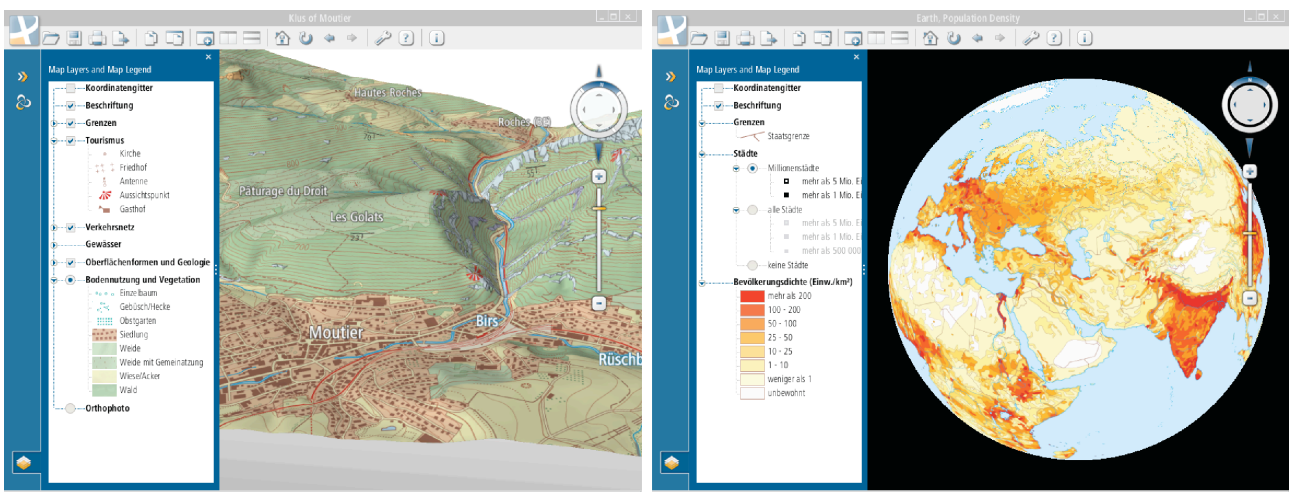


Figure 2: Block diagram and virtual globe from the Swiss World Atlas interactive

⁴ «The Atlas of Switzerland is highly regarded by the international cartographic and GIScience community and it is often referred to as "best-practice" for design, production and delivery of a national atlas.» Prof. Cartwright, President of the International Cartographic Association and Professor of Cartography and Geographical Visualization at the RMIT University in Melbourne (Hurni *et al.*, 2011)

⁵ <http://www.schweizerweltatlas.ch/de/schweizer-weltatlas/3.html>

No plugin is needed to access to the SWAi. It is accessible through a Java Web Start that allows it to run outside the browser as an independent application and thus overcome issues with browser compatibility.

Géoportail beta (France)⁶

The Géoportail is managed by the Institut Géographique National et the Bureau des Recherches Géologiques et Minières with the support of private suppliers for the technology part and of two public ministries. It offers 2D and 3D displays as well, but does not have block diagrams. It comprises of a virtual globe draped with aerial images that allows a bird-eye view of the relief. The 3D module is currently only in beta version and thus only provides ten themes or types of data that can be draped on the relief. Even though its name could suggest an access to data, it is not the case. The Géocatalogue allows the users to access the geospatial data. The Géoportail requires the installation of the plugin TerraExplorer to view the 3D module. It can be integrated in other webpages through its API and is compatible with Windows, Mac and Linux operating systems.

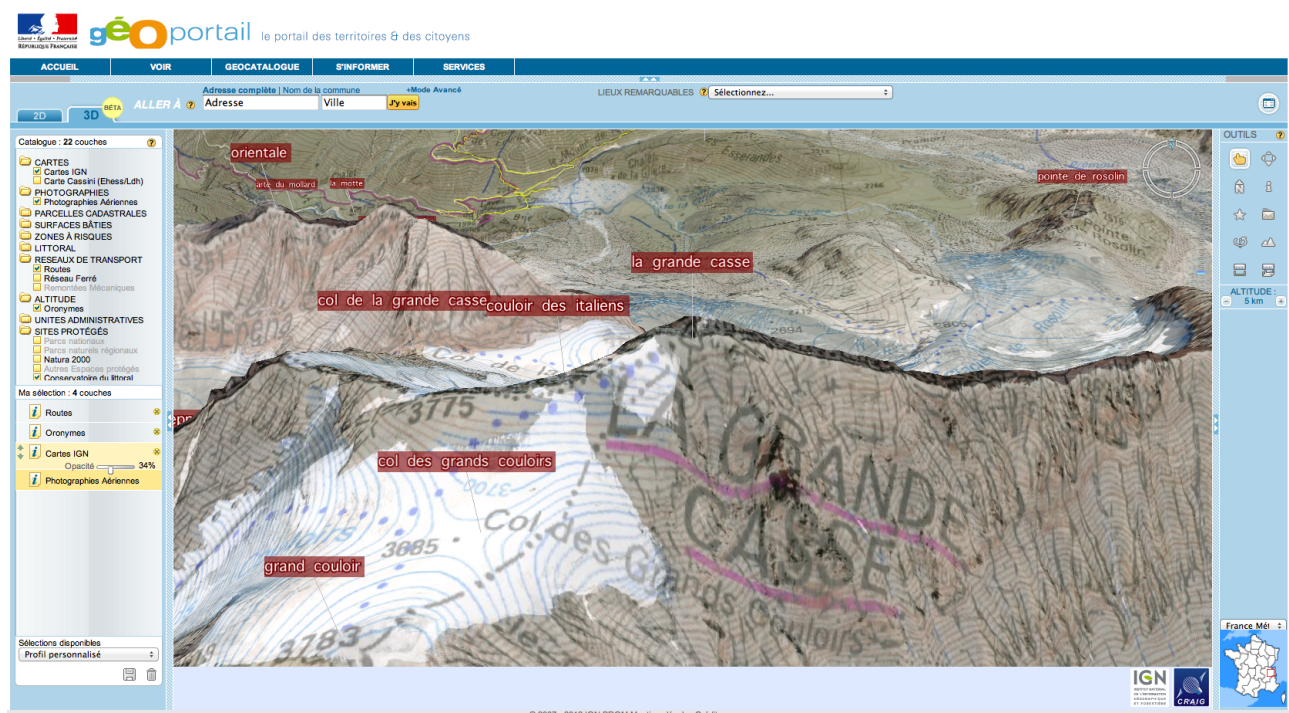


Figure 3: Viewer of the Géoportail. Relief view with the layers roads, orographic names, national maps (transparent) and aerial image

GeoPortal Bund (Germany)⁷

The GeoPortal is developed and managed by the Bundesamt für Kartographie und Geodäsie (BKG), mandated by the Interministeriellen Ausschusses für Geoinformationswesen (IMAGI) of the federal government.

The GeoViewer of the GeoPortal Bund offers three modes: BasisView, National Geo-DataBase and GDI-DE Model-Project for the protected areas theme. All of them provide with a 3D viewer for certain scale (1: 25 000 and smaller). The 3D viewer is still very basic with only the options viewpoint, exaggeration, distance and pitch, because the project is in development. The choice of layers has to be done in the 2D viewer and then the 3D view can be generated. No plugins are needed to view the 3D models, only a browser.

⁶ <http://www.geoportail.fr/5069712/visu3D/afficher-en-3d.htm>

⁷ http://geoportal.bkg.bund.de/nn_78634/DE/Geoviewer/Geoviewer_node.html

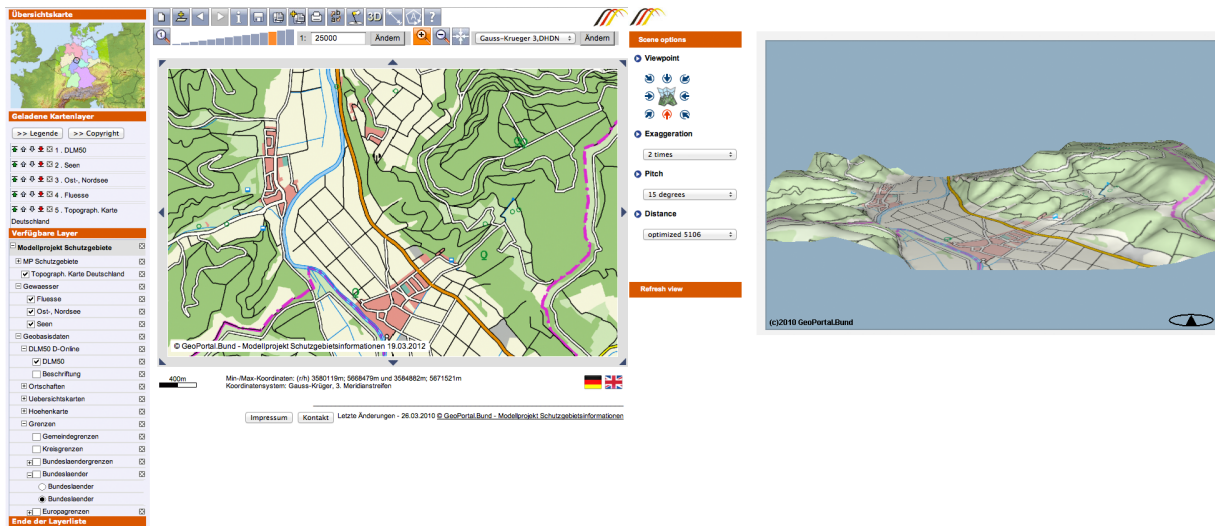


Figure 4: Corresponding views in 2D and 3D of the GeoPortal Bund

Infraestructura de Datos Espaciales de España IDEE (Spain)⁸

IDEE supports the creation of a Spanish Spatial Data Infrastructure, with data, metadata and services. It links several regional spatial information systems. The Spanish government, the Ministerio de Fomento (Ministry of Public Works) and the Consejo Superior Geográfico (Geographic High Board) are responsible for it. An API is available and the website offers a map viewer, WMS services and a 3D Viewer. The 3D Viewer only works in Explorer and Firefox and only on the operating system of Windows, but requires a plugin which is not compatible with Firefox 11.0, furthermore, on Windows Vista or Windows 7, it only works with Firefox. So it is impossible to make it work on an up-to-date computer. Additionally, the map viewer, while offering many options to link it with other data sources (servers and kml data), encounters problems when loading the base map in all browser, except Firefox. This is a perfect example that supports the necessity of interoperability and standards for 3D technologies online. The development team could not follow with the updates of the browsers and operating system, making the geoportal almost unusable.

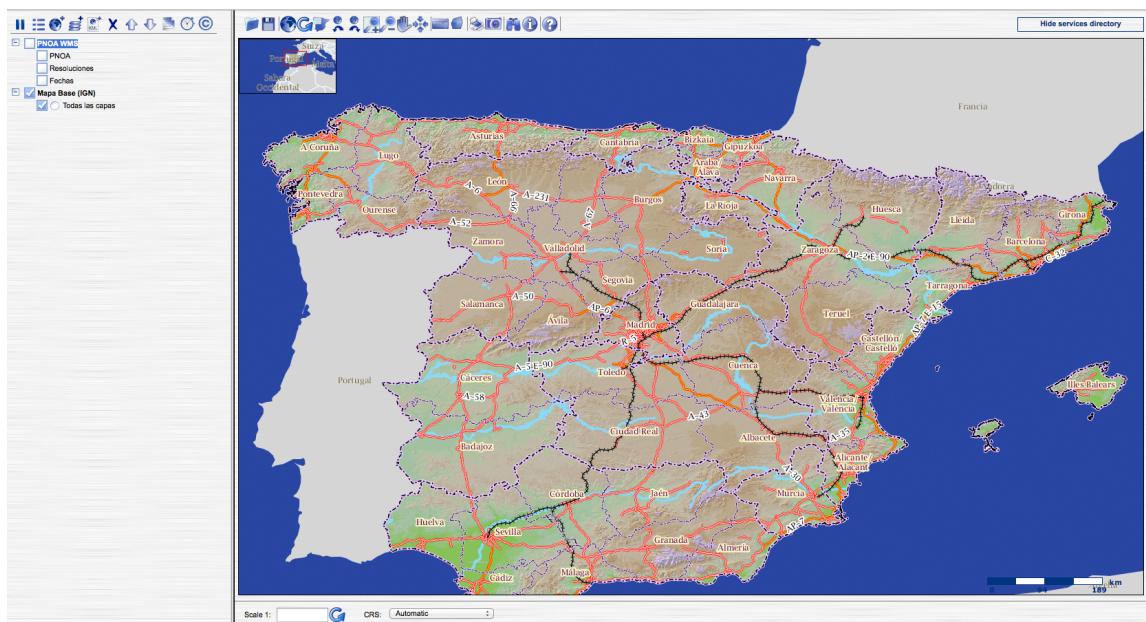


Figure 5: 2D Viewer of the IDEE

⁸ <http://www.idee.es>

Atlas of Switzerland 3 (AdS)⁹

The AdS is a project conducted by the Bundesamt für Landestopographie, the Bundesamt für Statistik, the Institut für Kartografie und Geoinformation (ETH Zürich) and the ETH-Rat. The AdS offers 2D thematic and physical maps as well as block diagrams, panorama views and prism maps in 3D. A multitude of statistical data is available under different themes (Nature and Environment, Society, Economy, State and Politics, Traffic and finally Energy and Communication). It provides the users with a wide range of tools, from setting the weather and light in the sky to the colors and classes of thematic maps.

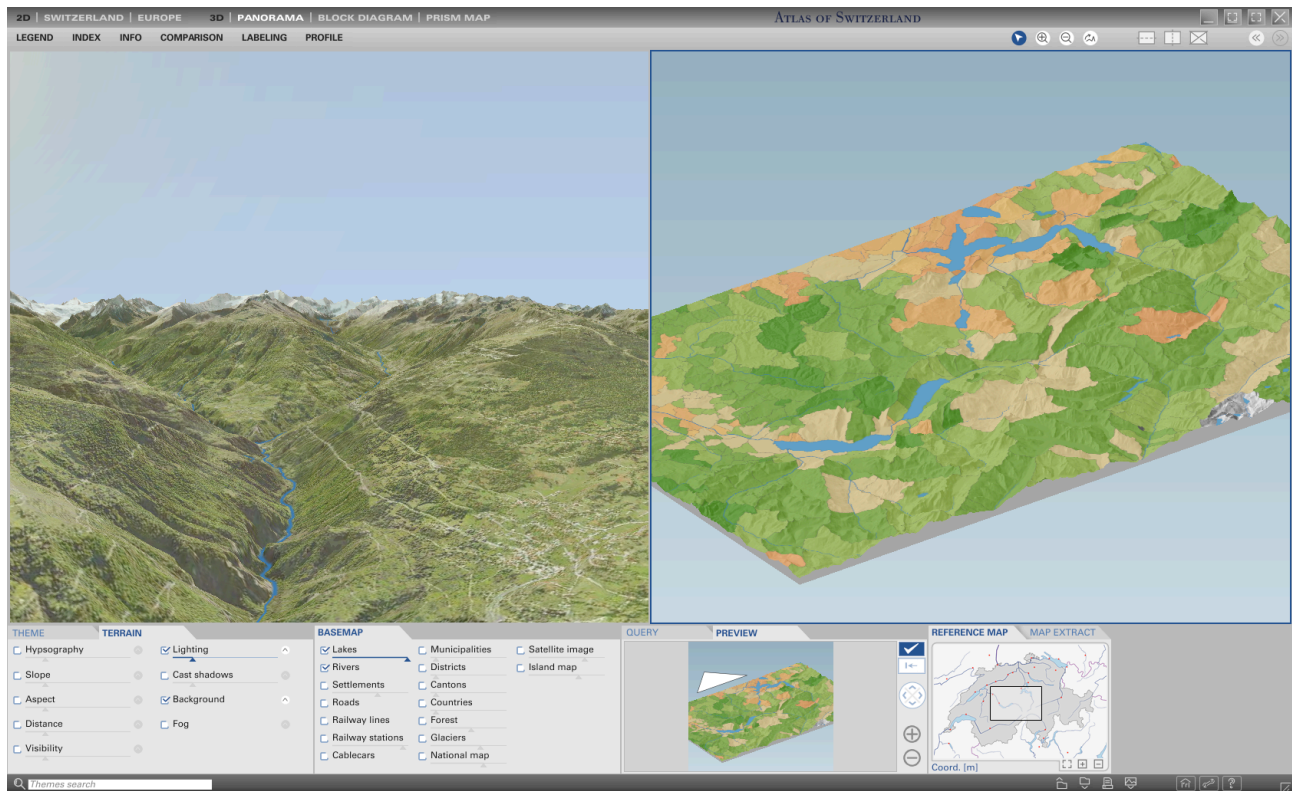


Figure 6: Panorama view and block image in the Atlas of Switzerland 3

It is available on DVD, but it does not require to install it on the computer and works with both Windows and Mac operating systems. Presently, there is a will of the different actors to bring the AdS online (Hurni *et al.*, 2011).

4.1.2. Virtual Globe

Google Earth¹⁰

Google Earth is probably the most well-known virtual globe application among the public. Although it is a closed source application, the basic version can be downloaded for free. It offers different view modes: Google Street View, Flight Simulator, Google Ocean, Google Sky and Google Sketch up.

It can handle vector, raster and kml formats, as well as dynamic objects, 3D building models, statistic surfaces, atmospheric and light effects. It is available as a desktop application, a web plugin and as a mobile version and it support WMS. Although Google Earth is compatible with a wide range of formats and offers a significant amount of default data, it has its limitations. First, it is closed source

⁹ <http://www.atlasderschweiz.ch/atlas>

¹⁰ <http://www.google.com>

and thus excludes any own development. Then, the data usage permissions are largely limited. Finally, it does not handle elevation data (Walker and Kalberer, 2010).

NASA World Wind

It is an open source and free virtual globe API, that was created by the NASA's Learning Technologies Project and which is now developed by NASA staff and open source community developers. The basis looks like Google Earth with DEM, satellite image and navigation tools. Additionally, there exist plenty modules with different capabilities. These modules can handle WMS, vector and raster formats, elevation, 3D objects and models and they are available through a Java Web Start. The downside is that each module can currently only be used as a single unit. Additional programming and development is possible with a Java Software Development Kit (SDK) (Hurni *et al.*, 2011; Walker and Kalberer, 2010).

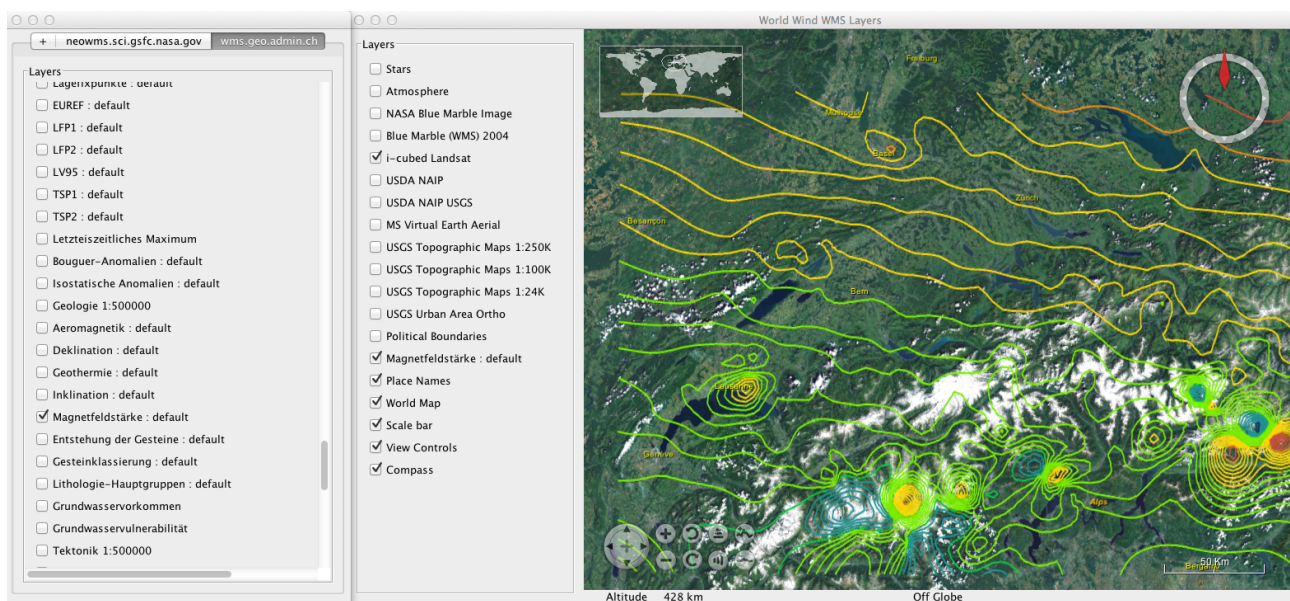


Figure 7: Zoom in on Switzerland on the virtual globe of NASA World Wind (module WMS), with the isoline of the magnet field strengths from the WMS of the Swiss government

osgEarth

It is an open source and free virtual globe, based on a geoid (Hurni *et al.*, 2011), developed and maintained from the firm Pelican Mapping (Pelican Ventures, 2012). It is a scalable terrain rendering toolkit for OpenSceneGraph (OSG) written in C++ using OpenGL and it is multi-platform and standard-compliant. OpenSceneGraph is an open source graphics toolkit for high-performance graphics application such as games, virtual landscape and scientific visualization. There exist many plugin to load a wide range of data formats as well (OSG Community, 2007).

OgsEarth supports raster and elevation formats in the form of WMS, WMS-T (time), WCS, TMS (Tile Map Standard) as well as vector formats, such as shapefile, and 3D models. It additionally offers a Virtual Planet Builder terrain database. The main features of this virtual globe are image rendering tasks (its ability to drape vector data on the terrain) and reprojection on the fly (Walker and Kalberer, 2010). Based on these features and functionalities, it could be a good candidate for the implementation of a 3D Viewer for a web atlas (Hurni *et al.*, 2011).

Open Web Globe

It is developed by the Fachhochschule Nord-West Schweiz Muttenz as an open source project and aims at true visualization of significant amount of geodata. For that, it uses exact reference frames. It is now in a beta version on WebGL, so it can be used directly in the browser. The i3D version supports

WMS, DEM, satellite images, dynamic objects, POIs, prism maps, 3D objects and real-time data. It can be further developed with the OpenWebGlobe SDK, which would allow to create cartographic tools, such as symbols and diagrams and thus is seen as a 3D-viewer candidate for a web atlas platform (Hurni *et al.*, 2011). However, vector and kml data are not supported yet and the beta WebGL version is not fully operational yet, for instance WMS are not implemented at the time of writing. The source code is written in C++ and the code for the viewer in JavaScript and thus it is cross-platform.

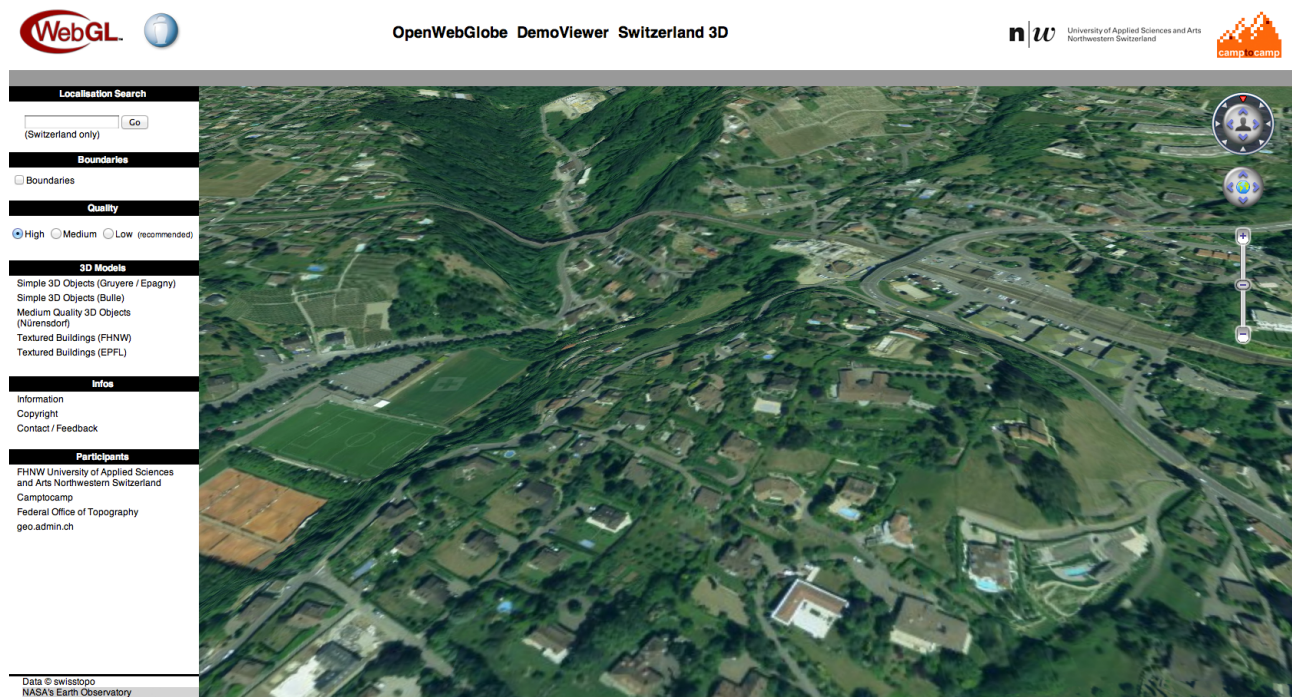


Figure 8: Close zoom in on the demo version of the OpenWebGlobe

4.1.3. GIS-based Viewers

gvSIG 3D

It is an open source and free GIS-based virtual globe. It acts as the 3D extension to gvSIG and needs to be installed on the computer. It supports many different formats: raster, vector, elevation, 3D models and OSG. It is fully integrated in the GIS desktop, which allows to edit and style the data very easily. Additionally to that, there is the osgVirtualPlanets, which is a standalone framework (Walker and Kalberer, 2010).

ossimPlanet

It is a free and open source 3D viewer for the GIS application ossim. It supports advanced geospatial image processing and OpenSceneGraph, but is not widely accepted in the open source community (Hurni *et al.*, 2011). It supports the following formats: raster, vector (as kml), WMS, elevation (ossim format) and a QGIS plugin project is currently running (Walker and Kalberer, 2010).

Summary

The following table gives an overview of the different capabilities of the virtual globes mentioned above. Many of them support a wide range of data formats and are open source, although under different licenses.

Table 1: Characteristics of Virtual Globes and possible import formats

Virtual Globe	Open Source	Type of License	Language	Raster	Vector	Elevation	KML	WMS	OSG	3D Models
Google Earth	no	Google Terms	C++ (?)	yes	yes	no	yes	yes	yes	yes
NASA World Wind	yes	NASA Open Source Agreement	Java	yes	yes	yes	yes	yes	yes	yes
osgEarth	yes	LGPL	C++	yes	yes	yes	yes	yes	yes	yes
Open Web Globe	yes	MIT	C++	yes	yes	yes	planned	planned	planned	yes
gvSIG 3D	yes	GPL	C++, Java	yes	yes	yes	yes	yes	yes	yes
ossimPlanet	yes	LGPL	C++	yes	yes	yes	yes	yes	yes	no

4.2. 3D Technologies Review

This section explores the available technologies for 3D visualization. It starts with data formats for online graphic data and, more specifically, 3D geodata. It makes an assessment of their strengths and weaknesses, as well as their possible implementation fields. Then, it looks at 3D viewers for such data online. After, it moves to web services that aims at handling 3D geodata. Finally, it concludes with some possible architectures that are found in the literature for combining these different technologies. This section does not aim at being exhaustive, but mentions the most used formats and technologies.

4.2.1. Graphic Formats

VRML¹¹ and X3D¹²

Virtual Reality Modeling Language is a standard file format for representing 3D interactive vector graphics. It was developed by the Web 3D Consortium and became an ISO standard in 1997. It has been superseded by X3D. Some of its limitations include the dependency on particular VRML viewer, the lack of ease for users to add data and the lack of some interactivity (Beard *et al.*, 2005).

X3D is the ISO standard based on XML for representing 3D graphics and was developed as the successor of VRML by the Web 3D Consortium. There are many open source platforms that can parse and interpret X3D, which makes it a good candidate for interoperable solution. It provides with easier interaction possibilities with other XML formats and it does not depend on specific viewers (Beard *et al.*, 2005), however a plugin (viewer) is still necessary.

WebGL¹³

WebGL is the equivalent of OpenGL ES for browsers developed by the Khronos Group. Through the extension of JavaScript capabilities, it provides with interactive 3D graphics that are based on OpenGL and that uses the HTML5 canvas elements. Thus, it does not require any plugin and can be implemented directly into the browsers. However, there is a rather significant limitation to the

¹¹ <http://www.web3d.org/x3d/vrml>

¹² <http://www.web3d.org/x3d>

¹³ <http://www.khronos.org/webgl>

number of vertices and thus, only small models at high resolution can be displayed. It could though be a good solution for block diagram because they have a limited extent. Otherwise, the model has to be broken down into different segments (Visual Size, 2011).

KML¹⁴

The Keyhole Markup Language is an OGC standard that provides with web-based geographic annotations and visualization for both 2D mobile maps and 3D virtual globes. It was, at first, developed for Google Earth and then was submitted by Google to be an OGC (Open Geospatial Consortium, 2012). KML data are widely accepted input for many open source virtual globes.

GML¹⁵ and CityGML¹⁶

The Geography Markup Language is a XML-based modeling language for geographic features. The CityGML is an application schema of the GML that allows for the storage and representation of virtual 3D city and landscape models. Both are OGC standards, but only GML is an ISO standard. Because these two standards are XML-based and thus can be transferred as text over the internet, they are good candidate for a web 3D data retrieval service.

Flash¹⁷

Flash is a multimedia cross-platform browser-based application that can be used to add video, animation and interactivity in the browser as 2D content. It is a proprietary format from Adobe and it might require to install a plugin (Google Chrome and Internet Explorer already have Flash integrated). It is a widely used technology, but not open source.

SVG¹⁸

Scalable Vector Graphic is a XML-based file format, mainly for two-dimensional vector graphics and pseudo 3D, but it also supports raster file and foreign object, e.g. HTML. It was developed by the World Wide Web Consortium (W3C) (Dahlström *et al.*, 2011). Most browsers support some versions of SVG and it is a very flexible open standard. It is accessible by any DOM parser and it allows for interactivity with JavaScript and its Event Listeners. It is WMS-compliant and can be modified in a text editor, while needing very little local data for an interface (George, 2006). All of this and its styling capabilities make SVG a good candidate for a graphic user interface (GUI). It allows to use solely SVG for webpages, thus making them scalable independently from the screen resolution (Schnabel, 2002).

X3DOM

X3DOM is framework and runtime to support discussion within the W3C and Web3D communities on how to integrate HTML5 and X3D. It is currently at the experimental stage and it aims at being an open and human-readable 3D scene-graph embedded in the HTML DOM in a very similar way to how SVG is embedded in HTML (Behr *et al.*, 2011). It has the ability to render 3D models within a HTML5 browser through WebGL, without any plugin (x3dom, 2012).

¹⁴ <http://www.opengeospatial.org/standards/kml>

¹⁵ <http://www.opengeospatial.org/standards/gml>

¹⁶ <http://www.opengeospatial.org/standards/citygml>

¹⁷ <http://www.adobe.com/products/flashplayer.html>

¹⁸ <http://www.w3.org/Graphics/SVG>

Collada

Collada is an open standard XML schema used to exchange 3D asset between different applications that have incompatible authoring formats. Its filename extension .dae stands for digital asset exchange (Collada Community, 2011). Collada can be seen as a set of document and an API that help solving issues of transferring data between software without losing information (Pulli, 2006). It allows to combine software packages into powerful tool chains to create 3D content. The Khronos Group manages this 3D computer graphics format and many game engines and 3D software applications, such as CAD and GIS softwares, can render Collada file and export their data into the Collada format.

4.2.2. 3D Viewers

FreeWRL/FreeX3D

FreeWRL/FreeX3D is an open source and cross-platform compliant VRML/X3D browser. It is now developed under the LGPL and is believed to outclass other X3D plugins (FreeWRL, 2012).

BS ContactGeo

BS ContactGeo allows to visualize geographic information by enhancing third-party hardware and software products via the integration of interactive and Internet ready real-time 3D technology (Web 3D Consortium, 2011). It is 3D viewer that supports a wide range of formats such as X3D, CityGML, kmz.

4.3. Web Services

Web services allow end users to access functionality delivered by a service provider. It uses the concept of request-response between a client and a server. The client (service consumer or end user) sends a request to the service provider and in return the service provider sends a response, containing the results asked for, to the client (Iosifescu-Enescu, 2011). The response depends on the type of web services. Some services provide directly geodata, such as the Web Feature Service (vector data) or the Web Coverage Service (raster data), other deliver their visualization, such as the Web Map Service (WMS), in the form of an image, whereas other offer processing tools, such as Web Processing Service.

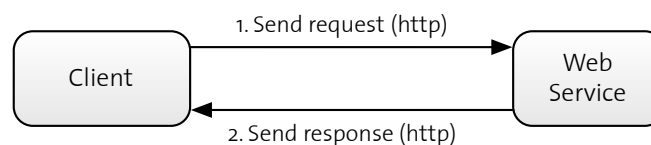


Figure 9: Web service concept

Each web service offers a service description, that allows to identify its interface and that defines its request format. The request itself contains parameters telling the server what is asked for by the user. Regarding geodata, important parameters concern the geographic extent, the coordinated reference system and the content, for instance. The request can be made through http GET, encoded in KVP, or http POST (optional) that includes a message body.

In cartography, and especially when talking about 3D visualization, the amount of data needed for the display of the information can be tremendous. Having all the data on the client side is not always possible nor desirable for performance or compatibility reasons. Web services enable to deliver only the visualization of the data and not the data themselves and thus allowing thin-client to access the information. Additionally, a service-oriented architecture guarantees that the application is independent from the ultimate implementation because the hardware and processing capabilities of the end user do not have to be taken into account. Another major advantage of web service concerns

their modularity: they can be chained (one web service serves as input for another one) or combined (several geodata sources can be display together, without having any data changing hands).

4.3.1. Web Map Service (WMS)

The WMS is widely use in cartography because it provides the user with spatially referenced 2D maps in the form of an image (jpg or png) dynamically from geographic information (Open Geospatial Consortium, 2006). The request GetMap enables the client to specify the geographic extent, the reference system, the layers and styles as well the format and size of the output image. The symbolization is either already defined by the administrator of the service or, when enable, by the user, which send the styling information with the request. The styling is done with the help of the Styled Layer Descriptor (SLD) and Symbology Encoding (SE) standards as well as Filter, which all are standards from the OGC.

4.3.2. Web Terrain Server (WTS)

WTS was developed by the OGC already in 2001 in an attempt to create a standard for a 3D scenes when it became clear that it would require many extensions and modifications to the WMS parameters. A 3D scene or a view can be defined as a 2D projection of three-dimensional features into a viewing plane (de Vries and Zlatanova, 2004). It produces views or 3D scene of geo-referenced data in different image formats, but does not contain nor deliver any actual data. Parameters such as Point Of Interest, Pitch, Yaw, Distance and Angle Of View were introduced to allow the generation of the views (Singh, 2001). Otherwise, on the conceptual level, it resembles a WMS with GetCapabilities and GetView requests. The drawback of WTS is the navigation through the views, because it requires to regenerate the view from the server every time. It never became a standard because the OGC replaced it with the Web Perspective View Service (see below).

4.3.3. Web Perspective View Service (WPVS)

The WPVS is an internal OGC draft developed in 2005 as the successor of the WTS. It does not allow for thematic data and requires navigation step by step like for the WTS. However, it does offer high-quality visualization and that regardless of the client hardware capabilities for rendering (Hagedorn *et al.*, 2010). Thanks to that and to the fact that the WPVS only transfers standard image formats, it is appropriate for simple client applications and requires little data transfer. The WPVS also never became a standard, because of its drawbacks regarding the use of the portrayed geodata, the navigation and the retrieval of feature information. Thus, the OGC decided to further develop a more complete version of a 3D web service under the name of the WVS (see below).

4.3.4. Web Perspective View Service++ (WPVS++)

To overcome the drawbacks of the WPVS, the Hasso Plattner Institute suggested an improved WPVS. WPVS++ can be used for thematic information transfer through multiple layers of images. The information would be encoded as images. They also developed complex request parameters to allow an easier handling of more advanced projection systems and interaction with the image. They proved that such a service could be done on a thin client based on JavaScript (Hagedorn *et al.*, 2010).

4.3.5. Web View Service (WVS)

The WVS is the latest standard candidate for an OGC 3D web service (along with the W3DS, see below). The WVS extends and overcomes the restrictions of the Web Terrain Service (WTS) and the OGC-internal Web Perspective View Service (WPVS) (Hildebrandt and Döllner, 2010); in a way, the WVS can be considered as the 3D counterpart for the well-established WMS. It became an OGC Discussion Paper in 2010. It provides with a portrayal service for three-dimensional geodata, mainly by delivering 2D images displaying a 3D scene constructed from 3D geodata (Hagedorn, 2010b). It supports thin

clients, like before, but now also supports analysis, navigation and information retrieval for portrayed 3D geovirtual environments. It is still built on an image-based approach, allowing a thick server-thin client concept.

In a WVS, a 3D view is defined by its projection type and parameters (how to go from a 3D world to a 2D representation), its data layers, the styling information and the basic portrayal information (dimension, format) (Hagedorn, 2010b). All these parameters can be encoded in a simple HTTP GET interface allowing for encoding views on complex 3D worlds by URL, transferring them easily, e.g., by email, and embedding them, e.g., into webpages (Hagedorn, 2010a).

4.3.6. Web 3D Service (W3DS)

W3DS is a standard candidate for a portrayal service for three-dimensional geodata delivered as scenes. These scenes are made of display elements, optimized for efficient real time rendering and can be explored in internet browsers with 3D plugins or loaded on virtual globe. The output is not an image, but a scene graphs consisting of a tree like structure of nodes, groups, transforms, shapes, materials and geometries (Schilling and Kolbe, 2010). It has been developed since 2005 and became a discussion paper in 2010 alongside with the WVS. It aims at supporting X3D and KML at first and offers different levels of details (LOD). Contrary to the WVS that allow a thin client concept, the W3DS requires a medium client, because the client is responsible for the rendering of the elements (Schilling and Kolbe, 2010). Thus, it requires more bandwidth and likely a 3D plugin for the visualization.

4.4. Architectures

Service-driven cartography is built on the concept of Service Oriented Architecture (SOA). This architecture couples loosely interacting software components that provides services (Iosifescu-Enescu, 2011). Services are central to this architectural concept and can be seen as modular units that allow to access, manage, process, combine and visualize heterogenous complex and vast geoinformation sources (Hildebrandt, 2008). These modular units can thus assemble into a network of services dealing with different tasks and data source. The SOA further supports distributed systems and thus allows to distribute the resources required for the visual representation in terms of network, storage and computing capacity over the whole network of computer (Hildebrandt *et al.*, 2011).

The OGC defines different ways to balance the geovisualization pipeline between client and server. The four-stage pipeline defines a process that goes from the non-graphical raw data on a repository to the visualization on a display device (Hagedorn, 2010b).

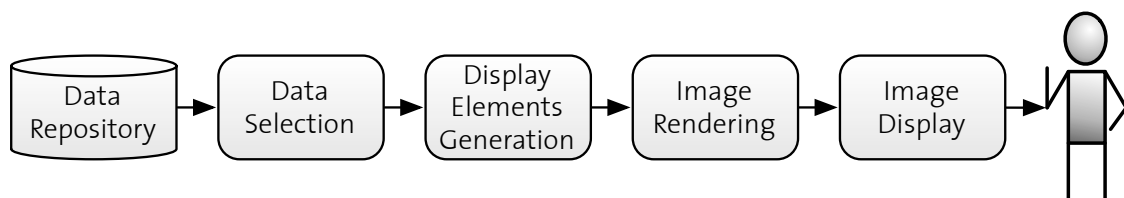


Figure 10: 3D Visualization Pipeline according to the OGC, adapted from (Hagedorn, 2010b; Schilling and Kolbe, 2010)

When the pipeline is run on a single desktop computer within a single software framework and no information sharing is required, it usually does not represent any problem to deal with 3D geodata. However the components of the portrayal pipeline do not have to be on the same system, they can be

distributed over the internet and that is where the SOA concept comes in. In server-client application, the components are split between one or more servers and clients. Usually, the lower level component is on the server and the remaining tasks are dealt with by the client. That is was is called a thick client, because of its high complexity. The advantage is that the client can run any analysis or rendering tasks on the data. However, regarding 3D geodata in web application, it reaches some limits due to computing and memory resources and bandwidth requirement to deal with 3D data on the client side (Schilling and Kolbe, 2010).

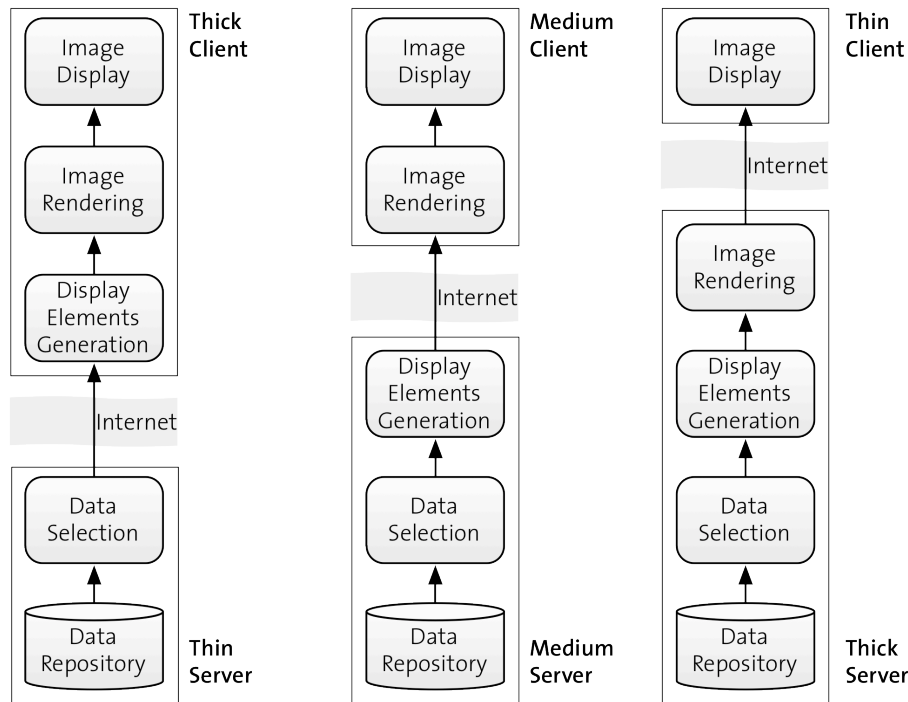


Figure 11: Balancing schemes between portrayal servers and clients according to the OGC, adapted from (Hagedorn, 2010b; Schilling and Kolbe, 2010)

To go around that issues, the different tasks can be split differently and the client has then only the display stage to deal with, which allows to access 3D geodata without issues about interoperability, computing resources for rendering or memory for storage. Because the processing, the generalization, the symbolization and the rendering of the data are be done on the server side and optimized for visualization and transfer, it enables any thin client to have access to complex 3D geodata sources in the form of images.

Two main types of architecture for 3D service driven geovisualization. The first family requires plugins to display the views or data in the browser. It is generally be either VRML plugin displaying data from VRML modules (Beard *et al.*, 2005) or, in later projects, X3D plugins that are linked to X3D models or GML features from a WFS or a W3DS (Beard *et al.*, 2005; de Vries and Zlatanova, 2004; Hetherington *et al.*, 2007; Hildebrandt *et al.*, 2011). But there are also other less known plugins that are available, such as the one from the Géoportail France, TerraExplorer. The OGC further suggested a simple workflows with a W3DS, including a browser with a 3D plugin for the requested and supported 3D formats and possibly connection to a WFS for input data. It returns a complete static 3D scene for exploration and analysis within the browser (Hagedorn *et al.*, 2008). The advantage is a really simple architecture, however the necessity of a plugin is a drawback. A similar architecture was used by Geoscience Australia (Beard *et al.*, 2005) to display sea bottom on the internet, but using VMRL models and a VRML plugin for the browser.

Generally, the use of plugin allows to display more complex models and supports more interaction with the data, because it bypasses the limitations of the browser. However, it can causes interoperability and cross-platform issues. These architectures are often used for massive 3D city models or large landscape model from complex sets of geodata.

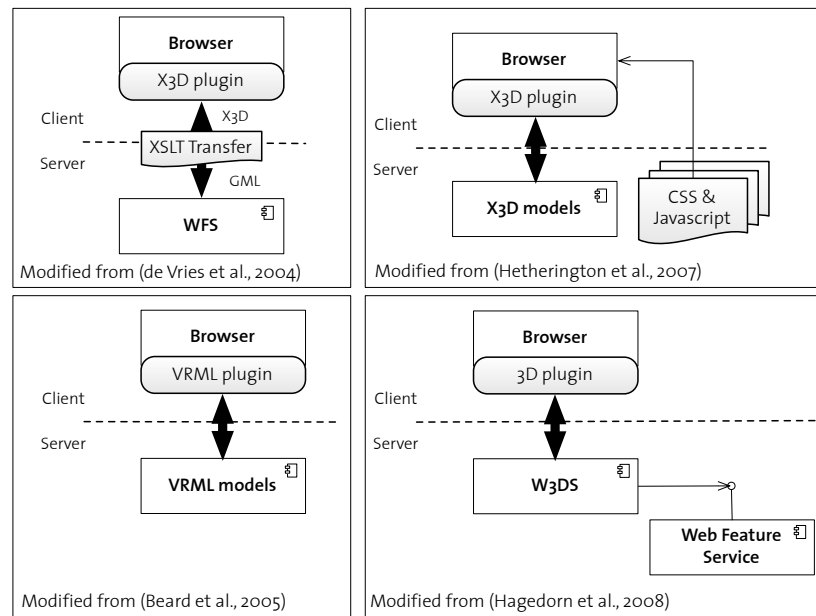


Figure 12: First family of architectures, using plugins to display the data in the browser

The second family uses WVS and similar services in order to generate perspective views of landscape. They do not require any plugin and offer a wide variety of combination and chaining of different web services. They usually take advantages of API, such as Google Earth or Google Maps to display the 3D views and models or they uses CityGML models. However, the simplest use consists of displayin the view in an image format directly within the browser.

For instance, Hildebrandt (2008) suggested an exemplary architecture of a service-oriented subsystem, using existing and newly proposed services that implements a process for the on-the-fly integration, generalization and portrayal of map-like 2D and 3D visual representations. Its goal is to offer a standardized and configurable service-driven visualization pipeline for heterogeneous geodata. The new services are uses input data from existing web services from the OGC.

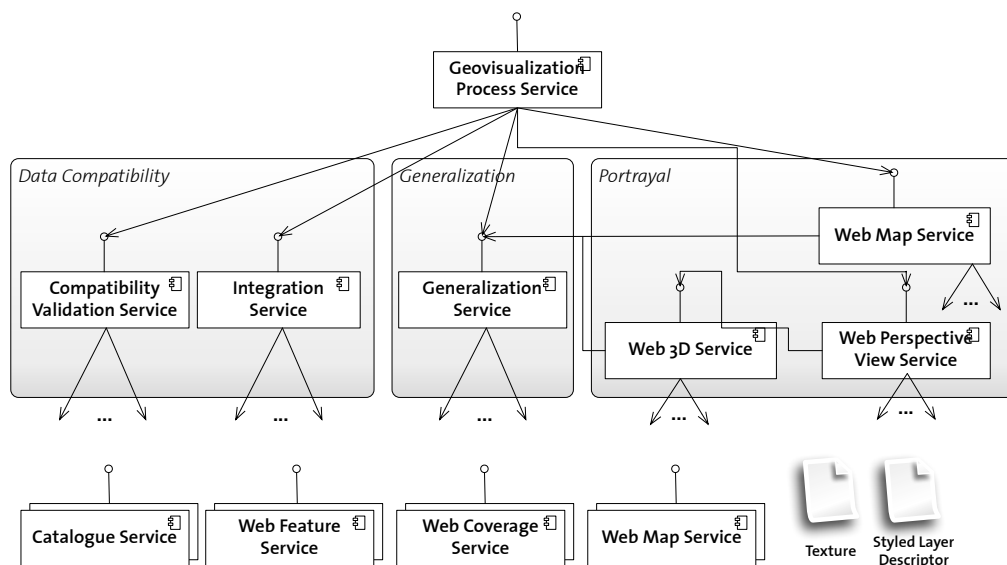


Figure 13: Architecture of a service-oriented subsystem (Hildebrandt, 2008)

Another architecture, simpler and from an earlier time was suggested by Altmaier and Kolbe (2003). It used the WMS, WFS and WCS as inputs for a WTS. It aimed at providing perspective views of maps, consisting of digital terrain models draped with 2D maps and orthophotos. Because the WTS could not integrate 3D data directly (such as WVS and W3DS) and can only be considered as 2.5D. However, it is a good example of a simple architecture using other services as inputs.

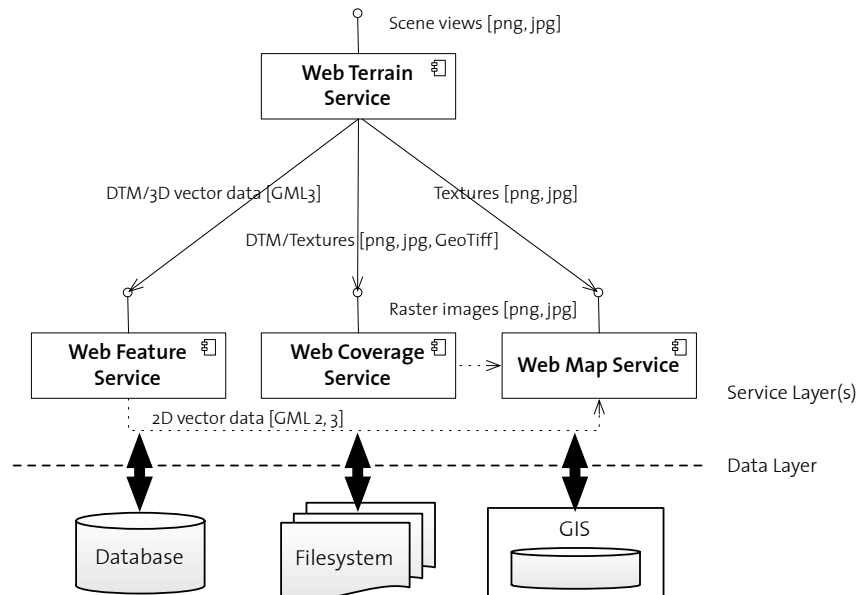


Figure 14: Architecture of the WTS, as realized by the open source system Deegree (Altmaier and Kolbe, 2003)

5. Requirements for Service-driven 3D Atlases

This chapter explores the different requirements for service-driven 3D geovisualization and 3D atlas systems. It looks into previous works and literature to list the core criteria for such systems.

With the increasing role that internet plays in the sharing of geodata, two trends of requirements for the visualization of 3D geodata have emerged. On one hand, users want to be able to access large amount of spatial data from many different sources as well as processing power and computing functionality. On the other hand, these resources have to be accessible from different places and from different users (Hildebrandt and Döllner, 2010). These two trends concern mostly system requirements, to which one must not forget to add visualization and interactivity requirements that are more specific to the atlas and 3D concepts.

This works attempts to combine three important concepts, that are atlases, service-driven system and 3D visualization. The following figure illustrates in which concept field of influence the different requirements are found.

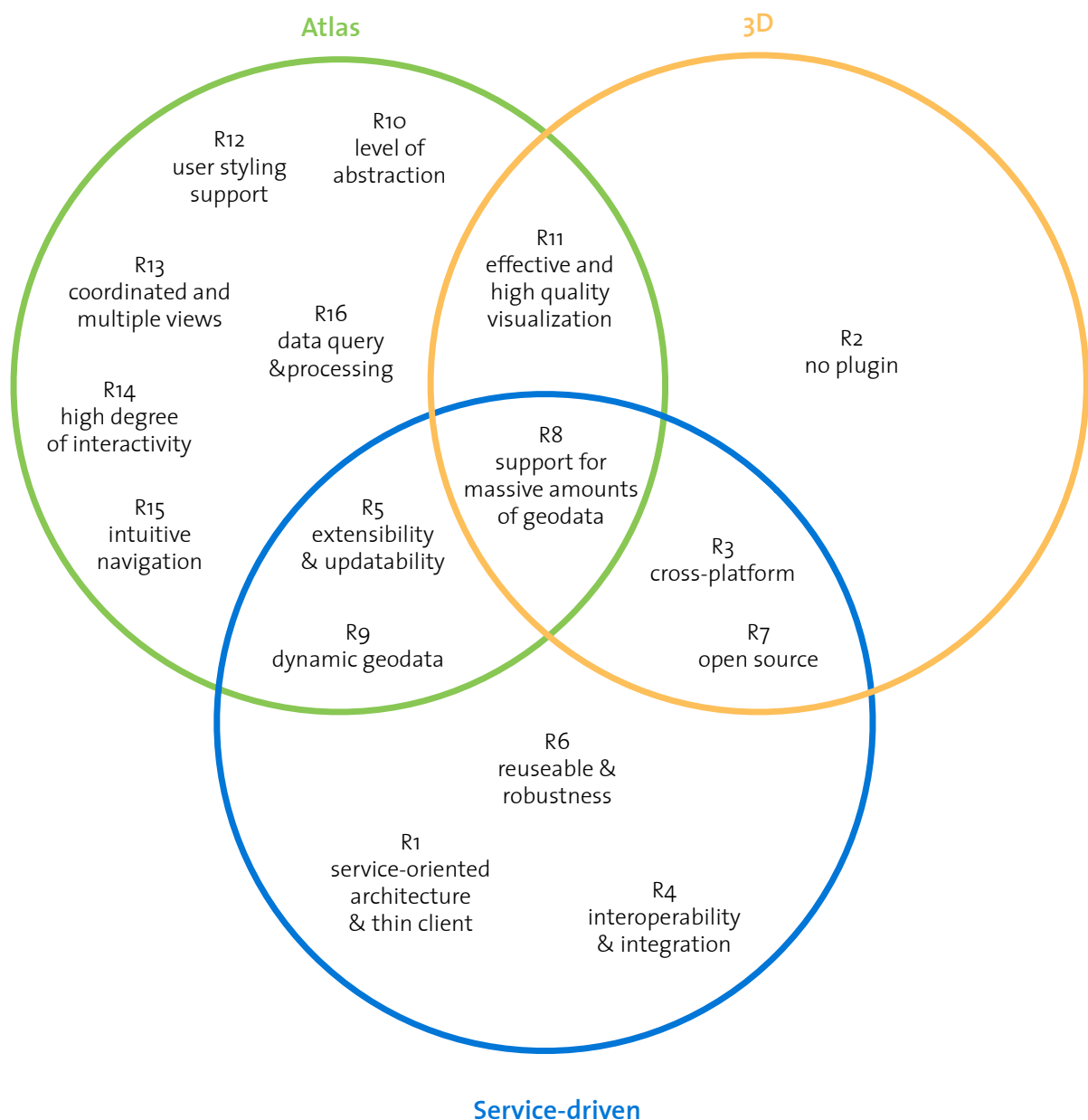


Figure 15: Requirements for service-driven 3D atlases in their fields of influence

5.1. System Requirements

The system requirements are based on general requirements for 3D geovisualization systems and on specific technical requirement for this project. These are relevant for 3D geovisualization, but not specific to it. First and foremost, the system is based on a service-oriented architecture (R1) to allow access to the visualization of geodata within the atlas, no matter what are the processing softwares or capacities of the client. It has to be accessible from a thin client. The visualization happens directly in the browser and no plugin is needed (R2). The absence of plugin aims at simplifying the use of the atlas and to avoid any compatibility issues between plugin and browser or platform. For instance, the IDEE 3D viewer requires a plugin that only works with some versions of Firefox, and thus highly limiting the access to the spatial data infrastructure platform. The system has to be able to work fully and independently from the software and hardware platform (R3), which increases the dissemination and prevent any compatibility issues (Hildebrandt and Döllner, 2010; Hildebrandt *et al.*, 2011).

Integration and interoperability (R4), which increases the integration of external data sources, especially through applying standards, are required to connect computer systems effectively and efficiently on different levels of abstraction (Brodie *et al.*, 2007; Hildebrandt and Döllner, 2010; Hildebrandt *et al.*, 2011). Interoperability additionally guarantees the access to 3D data sources in a homogeneous way for all users as well as the application of the same tools for processing and analysis to data from different sources (Altmaier and Kolbe, 2003). This allows to build flexible and adapting systems that can be readjust for different tasks and target users. Additionally, the different components of the system can serve several systems in varying combinations and with diverse goals (Andrienko *et al.*, 2005).

The support for easy updating, scale-up, extensibility (R5), reuse, and robustness (R6) are part of non-functional features. However, they help make the product long-lasting and optimizes its use and development, as well as its performance (Hildebrandt and Döllner, 2010; Hildebrandt *et al.*, 2011). For instance, a specific web service can be used in different atlas products provides significant gain in time or an existing web service for a national atlas can be extended to build a world atlas. Open source solutions (R7) should be preferred because they allow a direct access to the code, thus enabling a high flexibility and extensibility of the product, while benefiting from the developments within the community.

The last two system requirements are more specifically oriented towards 3D geovisualization. To build a comprehensive atlas requires support for massive amount of geodata (R8), as well as dynamic geodata (R9). This is especially important regarding 3D geospatial data that can be voluminous. The speed of data access and of the display generation are significant factors in the ease of use of 3D geovisualization (Andrienko *et al.*, 2005). Without the availability and support for large set of data and its dynamic access, the usefulness of such atlas decreases.

5.2. Visualization Requirements

Visualization requirements are specific to 3D geovisualization and atlases. 3D web atlases are a specific type of 3D geovisualization and because they aim at a general public, which is less familiar with geodata and online representations, some of the requirements are more prominent than others.

The ability to build geovisualization systems at an increasing level of abstraction (R10) is becoming a relevant requirement (Döllner, 2005; Hildebrandt and Döllner, 2010), especially in order to raise productivity. The higher the level, the less details have to be handled and thus the amount of code is reduced. For instance, the visualization has to move from a single-object representation to a higher level of abstraction such as textures that replicate a group of single objects.

A crucial requirement concerns the quality and the effectiveness of the visual representation (R11). The web services should be able to deliver visual representations that are as good and effective as standard cartographic products (Hildebrandt *et al.*, 2011; Iosifescu-Enescu, 2011). It has already been demonstrated that 2D maps of an equivalent quality could be generated from a cartographic web service (Ortner, 2011) and the challenge is now for 3D view and 3D objects. For instance, the quality and effectiveness of service-driven atlas application regarding the visualization can use desktop application as benchmark. Support for user styling (R12) is highly relevant for web atlases because it allows the users to manipulate not only the representation of the landscape, but also the representation of the thematic data (Hildebrandt *et al.*, 2011). It makes it possible to generate different views from the same data, for instance by offering color schemes or landscape and atmosphere setting to match the time and the weather. Multiple and synchronized (or coordinated) views (R13) of maps is an important feature for simple exploratory visualization because it allows to visually compare geodata at the same time from different point of views (Hildebrandt and Döllner, 2010). 3D geovisualization offers more complex information to the users and thus coordinated views help understanding the representation by breaking down the complexity.

5.3. Interactivity Requirements

Interaction and dynamic display (R14) have to be at the center of digital atlas conception in order to get the best out of the geodata visualization. It allows the users to build their own knowledge and representations from the geodata by exploring and comparing them. Interaction can be rather basic in the earlier phases of the conception of a web atlas, but to fully take advantage of the multimedia capabilities of the web, the user has to be able to interact with the visual representation of the geodata. Interactive tools must be design with user-center design and using the less-is-more concept, otherwise they might become an obstacle to a user-friendly GUI. There is a few interactive functions that can be seen as minimum requirements for an atlas to be useable and that are found in most atlas, even the simpler ones. The general functions should always be present (Ormeling, 1997), although there are found in different manners and quantities. They are often set up as output elements such as icons or status bar (Cron, 2006). They are important because they give the users information about the state of the atlas (zoom mode or information mode for instance) and allow to launch general functions such as quit, print and forward/backward. Another group of interactive functions that are vital for any basic web atlas are navigation functions, especially the spatial and thematic ones. Atlases organize spatial knowledge spatially and thematically and without proper navigation functions to interact with the atlas content, the atlas has little use.

3D geodata are more complex than 2D geodata and thus an appropriate and intuitive spatial navigation (R15) is indispensable. Because the 3D display appears more realistic, the user expects to be able to manipulate it like one manipulates a 3D object in the real world: one wants to be able to see the different faces of the object or to go closer and further from it for example. This navigation should be intuitive enough so that no training is required for the user to use the atlas spatial navigation panel beyond tooltips and meaningful symbols. For instance, using the scroll and zoom functions already implemented for the navigation in web pages goes in this direction, as well as using the arrow on the keyboard. In a more advanced phase, querying and simple processing (R16) of data are high level requirements, but they allow to get the most of the geospatial data. For example, access to the histogram of the data or searching tools are part of this requirement.

6. Implementation

To prove the feasibility of a 3D web atlas based on a service-oriented architecture, a prototype displaying perspective views of the landscape and block diagrams is implemented using web services and WebGL. The prototype implementation can be divided in four parts: the Web Map Service (WMS), the 3D Web Service (Globe Capture Service GCS), the Block Diagram implementation (WebGL) and the Graphic User Interface (GUI).

This chapter details the implementation of the prototype. It starts with the architecture of the prototype and then explains the different components and the technologies they use. It ends with the presentation of the prototype and its features.

6.1. Architecture

The following figure demonstrates the system architecture of the prototype. It consists of three tiers, the data tier, the web service tier and the user interface tier. The architecture uses the SOA concept and makes use of the advantages of using web services that communicate with the data tier and the user interface, allowing to have a thin client – thick server system. The web services receive requests from the user interface tier through JavaScript and then request the corresponding data from the data repository on the servers. Finally, they get the response as an image (in this case) and send it to the user interface for display.

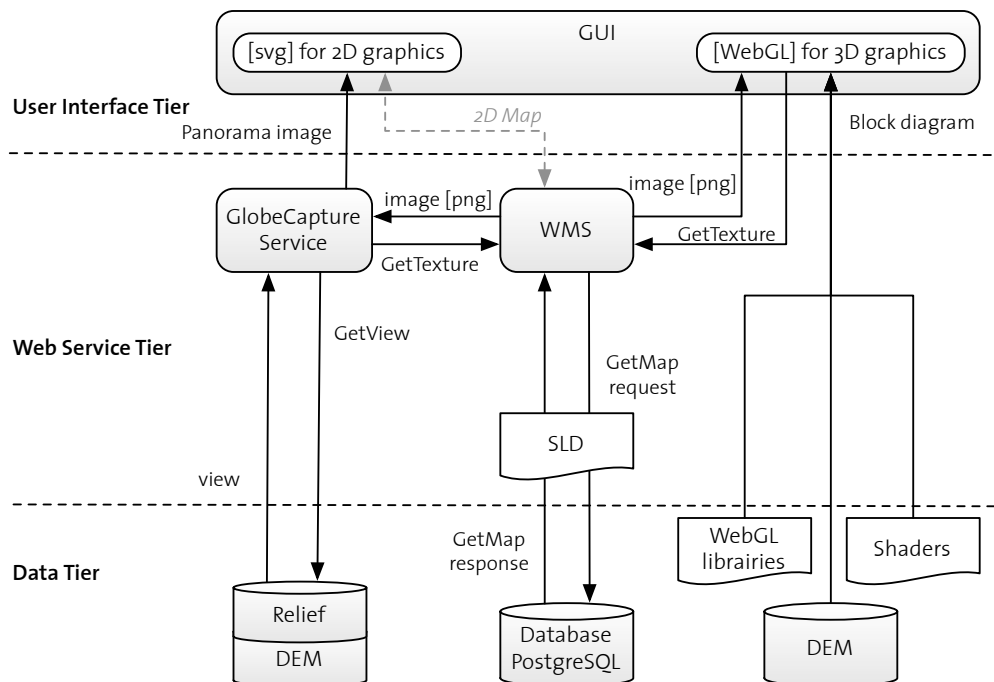


Figure 16: Architecture of the prototype

6.1.1. Data Tier

The data tier contains the data themselves, organized in database and files. There is one PostgreSQL database that holds the thematic data and the geometries for the WMS. These data come from a previous work regarding the service-driven generation of map for web atlases (Ortner, 2011). They originally come from the cartographic database of the Ads3 in the scale of 1:500'000 in Illustrator

format and have been processed and converted into shapefile and then into a PostGIS database. The following table shows which data have been selected out of that previous work for this prototype.

Table 2: List of layers used for the prototype and their name in the PostgreSQL database from (Ortner, 2011)

Thematic data from (Ortner, 2011)	Layer Name for the WMS and SLD	Type of data
admin_landesgrenzband_li_500k	Border_line	line
admin_landesgrenzband_500k	Boder_band	surface
admin_laenderoo_500k	Background	surface
basis_fluesse_500k	Rivers	line
basis_seen_500k	Lakes	surface
basis_staedte_500k	Agglomeration	surface
basis_strassen_500k	Roads	line
basis_wald_500k	Forest	surface
bevoelkerung_bz0901_500k	Population_density	surface
bevoelkerung_bz0901_pu	Population	point
geologie_500k	Geology	surface
verkehr_flug_verkehrsart_pu	Airports	point
verkehr_flug_verkehrsart_pu	Airports_labels	labels
verkehr_schiff_bestand_kt_pu	Boat_type	point
verkehr_schiff_bestand_kto8_500k	Boat_density	surface

Another data repository contains the relief as a raster file and the Digital Elevation Model (DEM) data for the GCS, the 3D web service. The global DEM for the service is made up of several data sets. The DEM for the Swiss territory is a combination, depending on zoom level, from the DHM25 (25 m grid) and the DTM_AV (2 m grid, but only for regions below 2000 m a.s.l) (Geodata © swisstopo). The Ferranti data set cover the area outside the national border (de Ferranti, 2012).

Furthermore, a third server holds the DEM and WebGL libraries and shaders for the generation of the block diagram. The DEMs for the block diagram come from from the DHM25 and the DHM RIMINI (Geodata © swisstopo). The DHM25 offers a matrix model with a 25m grid in the scale of 1:25'000 in ASCII format. It shows the elevation (without vegetation and buildings) of Switzerland. The RIMINI data set is DEM in the scale of 1:250'000. Both are raster files with a color value in each pixel encoding the elevation.

6.1.2. Web Service Tier

The web service tier is the core of service-driven cartography. It allows a flexible architecture, which is accessible from thin clients.

QGIS Server (Web Map Service)

The QGIS WMS Server allows to prepare 2D thematic maps that are used as a texture in the GCS and for the block diagrams. The QGIS Server is an open source and enhanced WMS. It supports WMS 1.3.0 and 1.1.1 implementation and styling with SLD 1.0.0, as well as cartographic extensions to SLD, allowing to draw diagrams, patterns and custom symbols (Iosifescu-Enescu, 2011). With the help of the plugin «Publish to Web», it is possible to directly generate the styling parameters (SLD document) for each layer from the QGIS desktop. The QGIS map server is implemented as a FastCGI/CGI application written in C++. It works with a web server (here Apache) that invokes the FastCGI/CGI application. It uses QGIS as backend for the GIS and map rendering logic (Iosifescu-Enescu, 2011; Quantum GIS, 2012). It is licensed as an open source program under the GPL license and it is supported by the following platforms: Linux, Windows XP and MacOSX (Quantum GIS, 2012).

Through a GetMap request, the QGIS WMS accesses the data on the PostgreSQL database and return an map image based on the size, extent and content parameters that are defined in the request. For the symbolization, it uses the SLD document, written in XML-format, that describes the style and then says which styles apply to which layers.

Globe Capture Service (3D Web Map Service)

As a 3D Web Service, the prototype uses the GCS developed at the Institute of Cartography and Geoinformation at ETH Zürich. It is based on the prototype version of the web version of AdS (still in development).

It allows to request a perspective view of any point in Switzerland. It uses a DEM and a grayscale relief raster image to generate a simple view and additionally allows to integrate a WMS to provide other textures. It offers the following parameters regarding the spatial navigation: *Point of Interest*, *Yaw*, *Roll*, *Pitch*, and *Distance*. The *Point of Interest* is defined by a pair of coordinates, the *Distance* represents how far the observer is from the *Point of Interest*. The other parameters define in which direction the observer is looking (*Yaw* or heading), at which angle (*Pitch* or elevation) and whether it is tilted (*Roll* or bank) (see figure 17). Yaw is what one does when one turns one's head to the left or the right, pitch is when one nods and roll would be when one tilts one's head to the side.

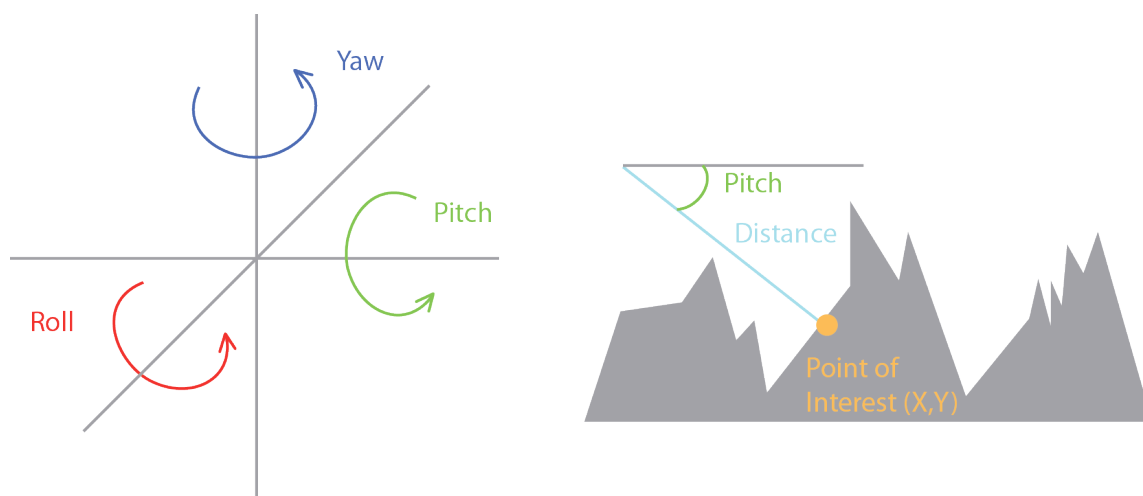


Figure 17: Parameters of the Globe Capture Service request

A few additional features were implemented parallel to this project. It is now possible to chose the time of the day for the illumination of the relief, thus allowing to see the stars at night or to have longer shadows. Moreover, the WMS layers now have an opacity parameter. The GCS offers a simple user interface to send requests, but it is integrated within the GUI through a HTTP Get request, using JSON encoding.

The GCS is based on osgEarth for the rendering task (see section 4.1.2) and it uses the mongoose web server, which is licensed under the terms of the MIT license. It can work as an embedded web server library to provide web interface to applications. No external library or configuration is needed, because launching Mongoose executable in any directory starts to serve that directory on port 8080 (Lyubka, 2012), making it practical for demos and prototype testing.

The GCS was chosen for the implementation because it was already available, albeit simple and coarse and thus allowing to manage the time constraint efficiently. As there is no standard for 3D web service yet, this service fits perfectly the needs for the proof of concept and could be modified to support a standard once it becomes available.

WebGL (Block Diagram)

The block diagrams consist of blocks of certain extent of the landscape that can be turned around as if the diagram were in the hand of the users. The prototype tests the capabilities of WebGL, which is an API based in OpenGL Embedded Systems API (Application Programming Interface). OpenGL ES is a royalty-free, cross-platform API for full-function 2D and 3D graphics on embedded systems, including consoles, phones, appliances and vehicles and it enables full-programmable 3D graphics (Khronos WebGL Working Group, 2012). WebGL allows to render 3D graphics within the browser without any plugin by using directly the graphic card on the users' computers. WebGL uses the HTML5 *canvas* element to render a scene. It is (partially) supported by all major browser (Chrome, Firefox, Safari). The base document used as a starting point in this work has been developed at the Institute of Cartography and Geoinformation with the help of the libraries from Google (webgl.utils.js)¹⁹ and Brandon Jonson (glMatrix.js)²⁰ (Bär, 2011). WebGL requests the texture from the WMS with the help of JavaScript. Because the scale is quite larger than for the panorama view, several styles for a same layer in WMS are defined to match the different scales.

6.1.3. User Interface Tier

The Graphic User Interface (GUI) is an essential part of the prototype because it allows the users to interact with the maps and representations. It contains all control tools for the interactivity and navigation. It was decided to use a GUI from a previous diploma thesis based on SVG. This GUI was developed as a web version of the Atlas of Switzerland and thus was expressively designed for web atlases, which is ideal for this project (Cron, 2006). This GUI uses JavaScript for interactivity and navigation and is based on the framework offered on carto:net (Neumann and Winter, 2011). It was chosen because of its flexibility and easy adaptability, thus allowing to expand it in order to offer 3D views and block diagram. Additionally, SVG does not require any plugin and is supported by all major browsers (Chrome; Firefox, Safari; and Internet Explorer only from version 8 on and with a plugin).

¹⁹ Available under <http://code.google.com/p/webglsamples/source/browse/book/#book%2Fextension> (retrieved on 25.05.2012)

²⁰ Available under a BSD-like license on <https://github.com/toji/gl-matrix> (retrieved on 25.05.2012)

6.2. High-level Workflow

This section explains the flow of information between the data repositories, the web services and the GUI for the panorama views and for the block diagrams.

6.2.1. The Panorama View

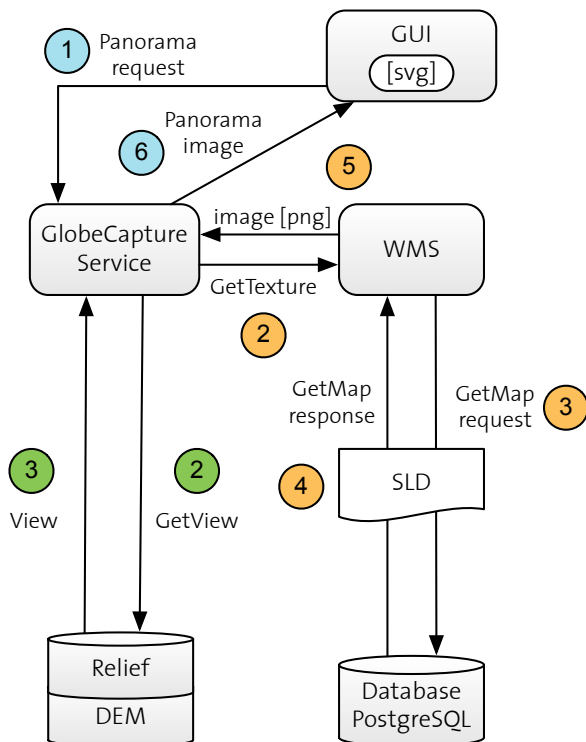


Figure 18: Flow of information for the panorama view mode

From the GUI, the user activates the panorama function (getGlobeCapture) by interacting with an element on the screen, such as a tab or a button. The panorama function sends a ① request with the parameters regarding the camera settings and the layer information to the GCS. It processes the information regarding the coordinates of the point of interest, as well as for the yaw, the pitch and the distance to the point and sends a ② request to the database to get the corresponding data. It receives an ③ image (png) of the panorama view made of the combination of the DEM and a relief. Then, the GCS sends a ② request to the QGIS Server to obtain the texture to overlay on the panorama. The WMS sends a ③ getMap request to the database to obtain the data. The data are then symbolized according to the SLD document and sent back to the WMS ④, which sends it further to the GCS ⑤. The GCS sees the WMS layer as an additional layer, on the same level as the shaded relief image (see appendix V). This step requires to define the following parameters of the WMS request: LAYERS, STYLES, SRS, TRANSPARENT, FORMAT (image). A few other are needed by the GCS to render the panorama: FORMAT (layer), OPACITY and URL. From the relief as well from the DEM and the map texture, the GCS renders the panorama view and sends it to the GUI for display ⑥.

Every modification of the request parameters from the GUI, such as the distance to the point of interest or the angle of view, sends a new request to the GCS and thus triggers again the entire flow of information.

6.2.2. The Block Diagram

The block diagram function works differently because the WebGL technology is quite different from SVG and web services. WebGL models can be integrated within the GUI either using the <canvas> or the <iframe> element from HTML5, both requiring the <foreignObject> tag in the SVG document. The <canvas> is used to draw directly the WebGL Context within the allotted space in the GUI, whereas the <iframe> defines an inline frame allowing to include external element, such as a HTML page in which the same canvas element is used. The option using directly the canvas element is chosen.

```

<foreignObject id="glforeignObject_canvas" x="25" y="25" width="700" height="700">
  <canvas id="glCanvas" width="700" height="700" xmlns="http://www.w3.org/1999/xhtml"></canvas>
</foreignObject>
<foreignObject id="glforeignObject_iframe" x="25" y="25" width="700" height="700">
  <iframe width="700" height="700" src="http://example.com" xmlns="http://www.w3.org/1999/xhtml"/>
</foreignObject>
  
```

Code Extract 1: iFrame and canvas elements

The block diagram function sends a ① request with the parameters regarding the texture and the location of the block diagram (see figure 19). This function calls upon the a DrawScene function to draw the WebGL context and the block diagram ②, because WebGL needs a Context in which the

objects are drawn. Once the Context is called, the rendering pipeline works as follow (see figure 20):

First, vertex arrays containing the vertices attributes, such as their position, texture and how they react to illumination are created. They are then sent to the GPU (Graphics Processing Unites) by putting them into vertex buffers while calling a function to draw an object. Then, WebGL processes the data that the user previously defined as attribute (now within the buffers) and uniform variables (that are used for the projection and the model view matrices). All of these are passed to the vertex shader. It calls the vertex shader for each vertex with the appropriate attributes, projects them and puts them into varying variables, one of them giving the position of the vertex. The WebGL then calls the fragment shader for every pixel without a vertex and fills them through a linear interpolation. Finally, everything is assembled within the frame buffer, which is eventually displayed on the screen (Caballero, 2011; Thomas, 2012). For this prototype, the texture comes from a WMS as an image (steps ② ③ ④ ⑤, same as in section 6.2.1)

Figure 19: Flow of information for the block diagram mode

which is used as a parameter in the loadTexture function of the block diagram (see appendix IV).

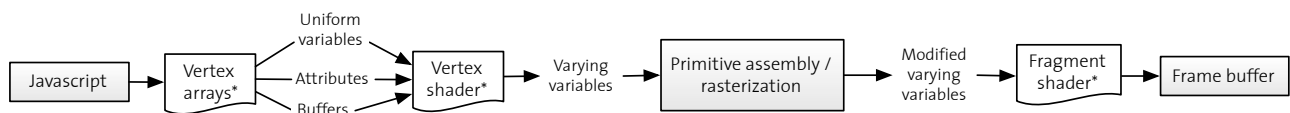


Figure 20: WebGL rendering pipeline, modified from (Caballero,2011; Thomas, 2012)

* user input

Then the texture is applied on the block diagram and sent to the GUI for display in the browser. Once the texture is applied, there is no need to send a new request to the WMS for every spatial navigation step. Although, if the user wants to change the texture, a new request has to be sent.

6.3. Presentation of the Prototype

6.3.1. General Description

The prototype is built on an existing GUI, of which the basic functionalities and their implementations are fully described in (Cron, 2006). The prototype, at the moment this thesis is written, has three different modes: 2D maps (which is not treated here), panorama view and block diagram. The following figure shows the consistency between the GUI of the three modes. The different tabs on the left side allow for the access to the legend, layers, analysis and information, when they are available. The bottom part is reserved for the spatial navigation and possible information about the features (only for 2D maps mode at the moment), as well as general functionalities, such as language choice, backward and forward.

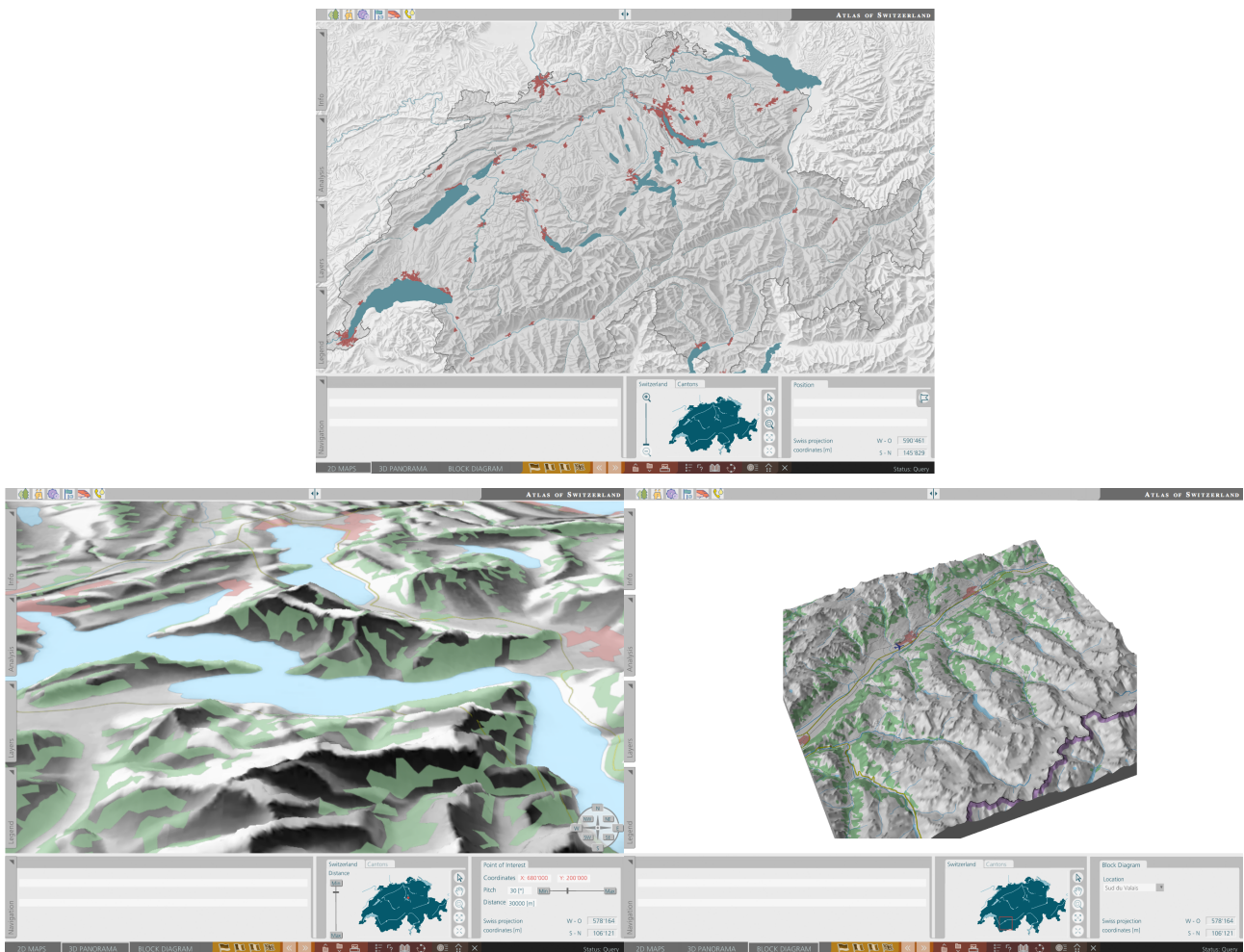


Figure 21: The three modes of the prototype; top: 2D maps, left: panorama view, right: block diagram

The 2D Maps mode uses a main map and a reference map for the navigation. Due to the 3D component, the main map is not useful in this form for the other modes, however, the reference map is essential for location purpose in the other modes. The panorama image and the block diagram each use an additional nested `<svg>` element in the DOM, where the image of the panorama and the WebGL scene are appended and drawn, respectively.

The Panorama View interface offers several tools to navigate spatially, thematically and temporally. It allows to chose the theme that is overlaid on the relief, the period of the day and the location of the point of interest. Because this is a prototype, only four themes are available with checkboxes, although one can imagine using a drop list menu for a more comprehensive web atlas, as it is suggested for the 2D Maps mode (Cron, 2006). The legend panel is right below the thematic navigation box. Both boxes can be move around, as well as close and open at will.

The temporal navigation offers the day and night option for the illumination and could be extended by allowing to chose different seasons and different times of the day for instance. The spatial navigation tools cover more options. First, one can choose the point of interest by clicking on the reference map as well as define the different parameters of the view, such as the angle of view (pitch), the direction of view (rose wind) and the distance to the point of interest, which can be seen as a zoom function). When only going over the reference map with a mouse pointer, the coordinates of the mouse are also displayed. Additionally, a red dot shows the location of the point of interest on the reference map.

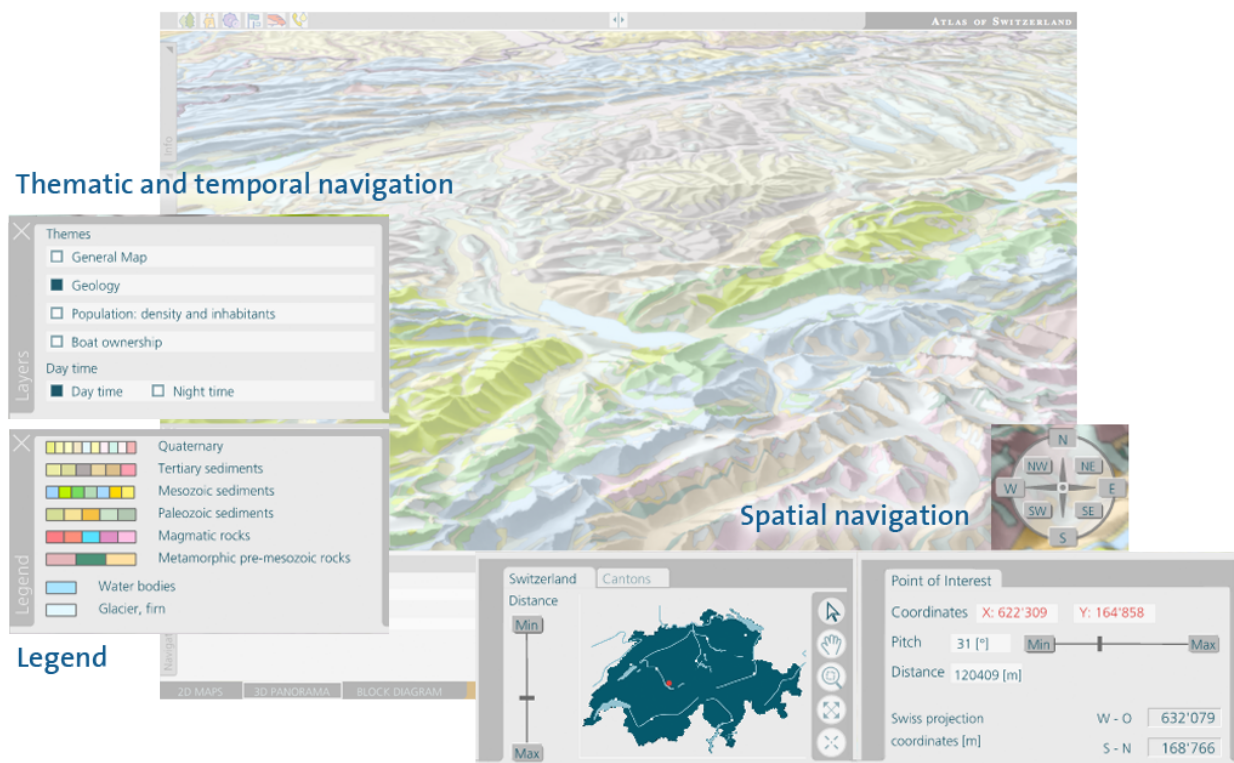


Figure 22: Design of the GUI for the panorama view mode and its navigational functions

The Block Diagram interface offers a selected choice of locations and themes to be displayed. A drop-down list allows the users to pick the location, which is then displayed on the reference map with a red frame. By clicking on the block, the user can make it turn around in all three directions, as well as zoom in and out. The thematic navigation and the legend are located at the same place as for the panorama and offer the same functionality.

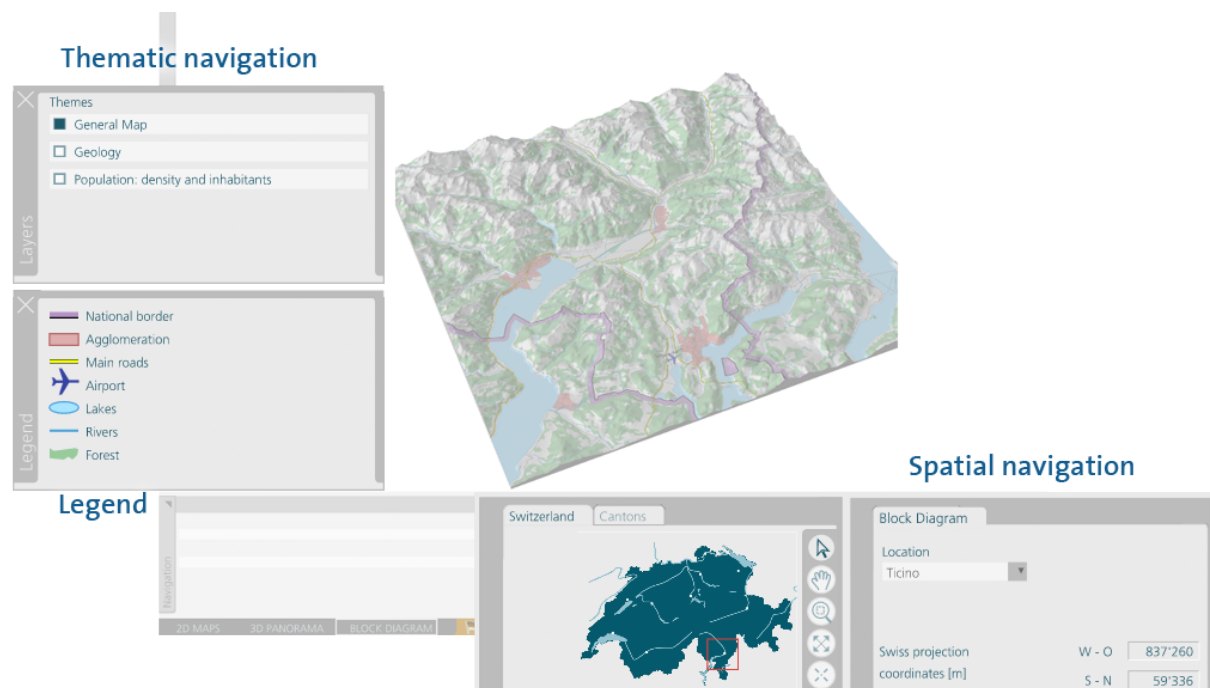


Figure 23: Design of the GUI for the block diagram mode and its navigational functions

6.3.2. Symbolization with WMS

WMS uses a SLD document containing rules on how to symbolize the geographic features located on the database. The SLD document is generated by the *Publish to Web* plugin in QGIS and is added to the server. Some options available in the SLD specifications are not offered in the symbolization toolbox from QGIS and thus have to be added manually when needed. For instance, the scale dependent symbolization or the minimal size for graduated symbols. It is also necessary to specify the EPSG (coordinates system, here 21781 for CH1903/LV03, the Swiss coordinates system) for each layer within the SLD. This work uses the *UserStyle* element to define the style of the layers. Filters are used to assign symbolization parameters, such as fill and stroke, to the different features based on their attributes (or properties).

```
<UserLayer xmlns="http://www.opengis.net/sld">
  <Name xmlns="http://www.opengis.net/sld">Geology</Name>
  ...
  <UserStyle xmlns="http://www.opengis.net/sld">
    <Name xmlns="http://www.opengis.net/sld">Geology</Name>
    <FeatureTypeStyle xmlns="http://www.opengis.net/sld">
      <Rule xmlns="http://www.opengis.net/sld">
        <Filter xmlns="http://www.opengis.net/ogc">
          <PropertyIsEqualTo xmlns="http://www.opengis.net/ogc">
            <PropertyName xmlns="http://www.opengis.net/ogc">ads_name</PropertyName>
            <Literal xmlns="http://www.opengis.net/ogc">Alluvionen</Literal>
          </PropertyIsEqualTo>
        </Filter>
        <PolygonSymbolizer xmlns="http://www.opengis.net/sld">
          <Stroke xmlns="http://www.opengis.net/sld">
            <CssParameter xmlns="http://www.opengis.net/sld" name="stroke" >#646464</CssParameter>
            <CssParameter xmlns="http://www.opengis.net/sld" name="stroke-width" >0.1</CssParameter>
          </Stroke>
          <Fill xmlns="http://www.opengis.net/sld">
            <CssParameter xmlns="http://www.opengis.net/sld" name="fill" >#f8f8bd</CssParameter>
          </Fill>
        </PolygonSymbolizer>
      </Rule>
      ...
    </FeatureTypeStyle>
  </UserStyle>
</UserLayer>
```

Code Extract 3: Example for the symbolization of alluvial deposits (*Alluvionen*) with SLD

6.3.3. Technical Aspects

The index.svg file, containing some JavaScript and comprising the main elements of the GUI, is extended in two ways. First, the instances for the interaction in the panorama and block diagram modes, such as sliders, checkboxes and buttons, are declared with the other interactive tools already present in the GUI at the beginning of the file. Then the shaders files for the block diagram are added as standalone script at the beginning as well (for details, see appendix VII).

Table 3 describes the JavaScript functions that have been implemented for the two 3D modes. One principal function *get_maps()* plays the role of the switch button between the different modes. Then, each mode has a main function responsible to display and hide the interactive tools necessary and to render the panorama view or the block diagram. These two main functions, *getGlobeCapute()* for the panorama view and *getBlockDiagram()* for the block diagram, call upon other functions that deal with the thematic navigation, and some of the spatial navigation tools.

Table 3 also explains the role of each function as well as its location in the three different JavaScript files that are added to the original GUI. Each function has a name and might have parameters, that are found in the brackets.

Few other modifications to the original GUI are necessary. In order to get the coordinates values from the reference map into the request for the panorama view, the event-listener «onclick» is added to the navigation.js file from the original GUI. It allows to capture the coordinates that are displayed on the mouse click and to send the request with the corresponding coordinates of the point of interest. The design is updated to the latest stand of the Atlas of Switzerland, on which it is based: the color scheme is change to gray and the symbols for the theme categories are updated, as well as the division of the different categories available.

Table 3: Extensions to the GUI, functions and description

Element and file name	JavaScript functions	Description
Navigation between views (get_map.js)	get_maps()	Change from one type of view to another; calling of the other functions needed to render the view itself
Panorama (getGlobeCapture.js)	getGlobeCapure()	Removal of unnecessary tools; display of necessary tools for the panorama view; calling of the getImageGC() function
	getImageGC()	Request of the panorama view to the Globe Capture Service; updating of the parameters display in the GUI
	setDistanceTo(changetype,id,value)	Change of the distance to the point of interest; calling of the getImageGC() function
	setPitchTo(changetype,id,value)	Change of the pitch from the point of interest; calling of the getImageGC() function
	setMaxMin(id)	Setting of the distance and pitch to their maximal or minimal values; calling of the getImageGC() function
	setYawTo(groupId,evt,buttonState,buttonText)	Change of the yaw from the point of interest; calling of the getImageGC() function
	setDaytime(id,selectedId,labelText)	Change of the time parameter; calling of the getImageGC() function
	setPointOfInterest(evt)	Change of the location of the point of interest; calling of the getImageGC() function
	setTheme(id,selectedId,labelText)	Change of the theme; calling of the getImageGC() function
Block diagram (getBlockDiagram.js)	getBlockDiagram()	Removal of unnecessary tools; display of necessary tools for the block diagram view; calling the getDiagram() function
	setThemeBD(id,selectedId,labelText)	Setting of the theme for the block diagram; calling the getDiagram() function
	getDHM(groupId,indexNr,value)	Setting of the DEM for the block diagrams; calling the getDiagram() function
Block diagram (webGL.js)	getDiagram()	Draw the scene

7. Discussion of Results

This section discusses what has been achieved during this Master thesis, starting with the analysis and the general implementation of the prototype. Then, it goes over the different weaknesses and advantages of the prototype regarding 3D visualization for an atlas, while putting into light possible solutions to solve some of the issues with the different technologies that are implemented.

7.1. Overview

The literature review shows that atlases have evolved with the technologies available at the time, from the paper version, to digital desktop application and now online atlases. Presently, the technologies and specifications enable to build service-driven 3D atlases, although they are rather at a beginning stage. 3D geovisualization offers advantages for shape understanding and orientation tasks, but these advantages cannot be fully used in service-driven application yet, because of specifications and standards issues. On the subject of atlas functionalities, the review shows that a set of functions are commonly found in all interactive atlases, 3D atlases included: general functions, spatial and thematic navigation. The technologies review demonstrates the large number of 3D graphics formats and specification and the lack of standards when it comes to 3D web services.

The analysis of the literature about 3D atlases and geovisualization as well as available specifications and technologies lays the foundations for the definition of the requirements. Sixteen requirements for service-driven 3D atlases are defined and explained. The system requirements are the foundations on which the other requirements should be built because the choice of the system logic can hinder or even prevent the fulfillment of the later ones. The requirements regarding the visualization and the interactivity can be fulfilled at different level depending of the target group and the stage of development of the atlas. These requirements are important guidelines and goals for the development of solutions that are sustainable, flexible and extensible and most importantly easy to use. Keeping these requirements in mind, a prototype is developed and shows that service-driven 3D atlases are possible. Not all requirements are fulfilled to their maximum within this work, however it shows the main issues and future challenges in the field of service-driven 3D atlases. With future developments in the field, additional requirements could join the ones defined in this work.

7.2. Advantages and Weaknesses of the Used 3D Technologies

7.2.1. *Combination of Globe Capture Service and QGIS Server*

One of the main advantage of using a 3D web service for perspective view is its very simple output: an image in png-format. It is a standard format for image that allows for transparency and relatively small file size, when considering that most screen displays have a resolution of 72 dpi. It is supported by any browser and does not require any plugin to display. It can be accessed by a thin client without any problem thanks to the absence of any additional processing engine needed to render the view. It can thus be envisaged to have a version for mobile devices as well.

Once the web service is set up, it can be used for different goals and provides not only the atlas with perspective view, but also a geoportal, for instance. However, the lack of standards for 3D web service, being for images or features, means that interoperability and integration in other systems cannot be guaranteed. It supports WMS standards, but vice versa with other standards is not true. Nevertheless, once a standard is agreed upon, the Globe Capture Service can be modified to match it. Depending on

the characteristics of the future standard for W3D or WVS, the standardization phase might however be long.

The combination of a 3D web service/Globe Capture Service, providing perspective views, and a WMS, providing the texture, shows substantial advantages regarding the symbolization of surfaces and lines, such as land use and road network or geological formations (see figure 24). All types of thematic data related to or in relation with the landscape and the slope can benefit from a 3D geovisualization.

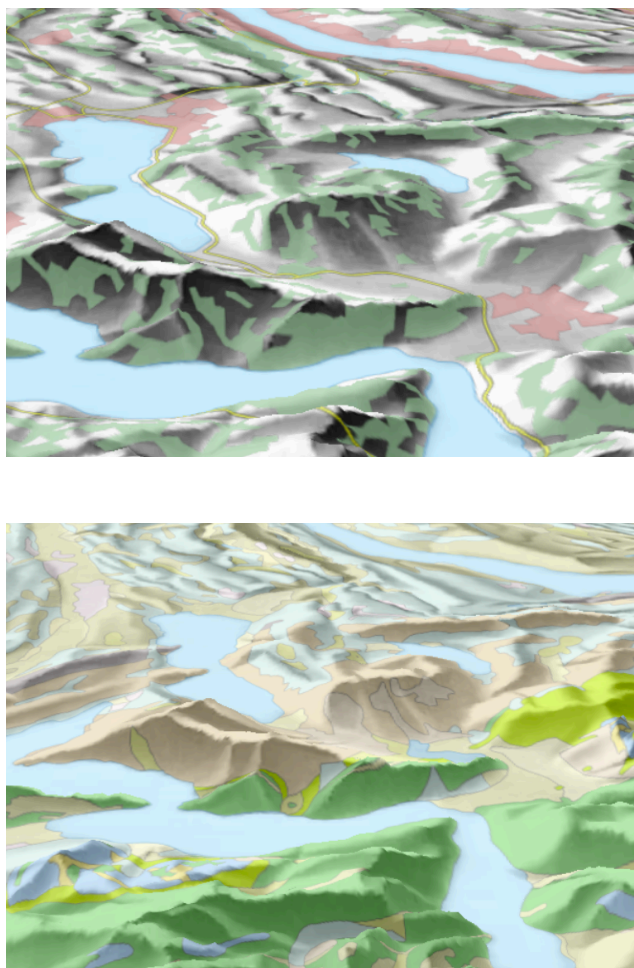


Figure 24: Panorama view, left: base map, right: geology map

Questions regarding the usefulness of displaying thematic data such as rate and percentage on a 2.5D or 3D landscape can be raised, though. No studies have been found on whether classical 2D maps would be here more effective for the understanding of the spatial phenomenon and relations than 3D geovisualization (see figure 24 and 25).

A clear downside of the GCS – WMS combination is the fact that is not well adapted for point symbolization and labeling. Indeed, icons, symbols, and diagrams are flattened onto the landscape relief and often distorted (see figures 25). They might also be hidden by the relief.

Another weakness of the GCS lies in the absence of 3D symbols, which is one of the attractive quality of 3D visualization. A solution could be to use billboards for symbols. Billboards are 2D or 3D symbols for point features that «float» above the point they symbolize and that are always oriented toward the camera. These are found in Google Earth, for instance.

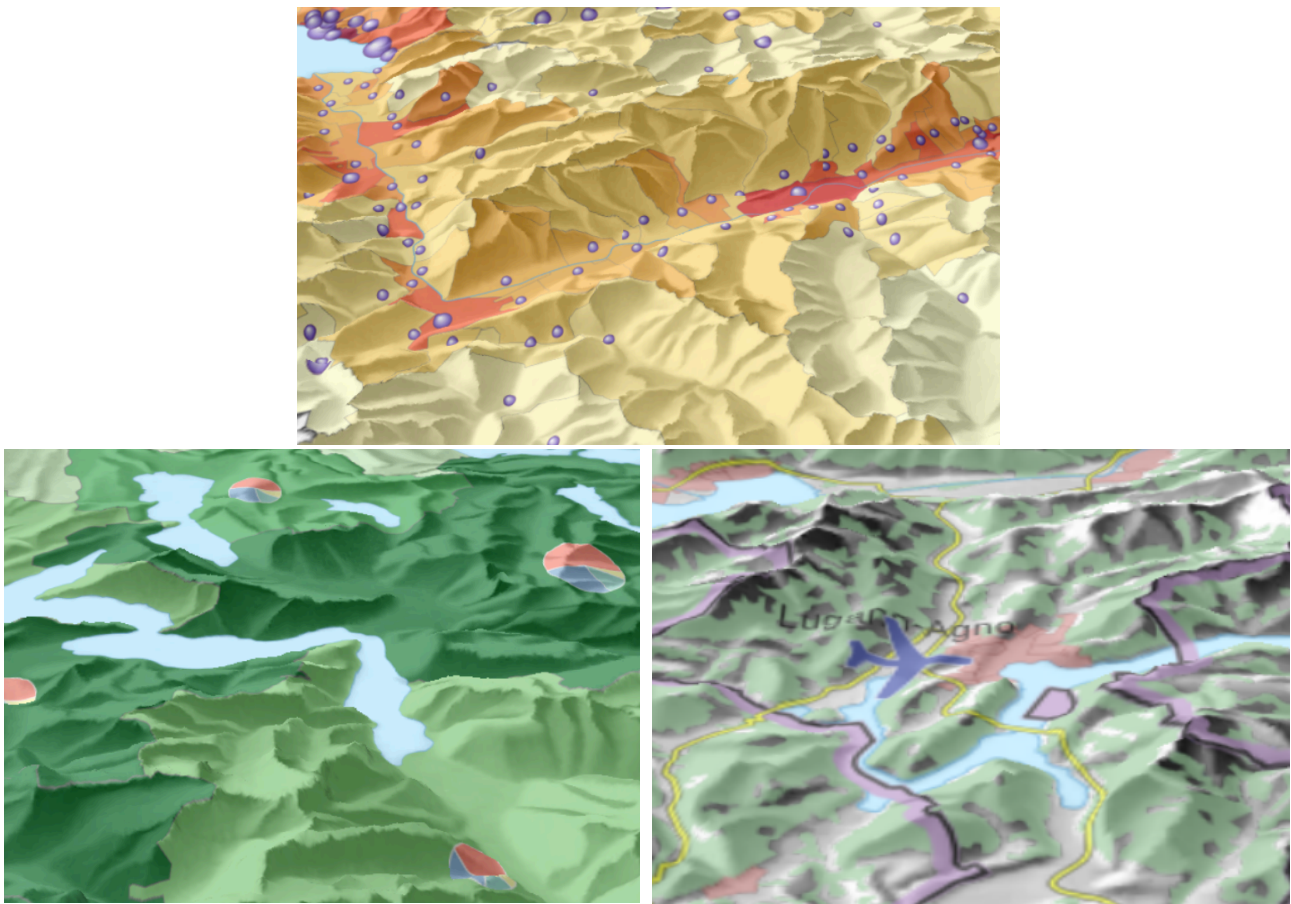


Figure 25: Panorama view, top: population density (choropleth) and number of inhabitants (sphere), left: diagram for boat types, right: airport symbol and labels on the base map

Due to the fact that it is a prototype, it still has many performance issues. The GCS takes some time and several requests to load the image at full resolution. Once it is loaded, it is not a problem any more for changing the camera and coordinates settings. However, every time the view requires a new combination of layers, the same problem arises due the service architecture. A workaround (but not an optimal solution) to fix it consists of running one server per map or combination of layers. This only works for this prototype, because just a few example layers are offered. Ideally, each layer should be queryable independently from the others. A long-term solution would be to redefine the request parameters and to optimize the loading process of the GCS. It is to expect that once the standards for WVS and W3DS are set, the communication and interoperability issues between these 3D web services and WMS be resolved. This would also help regarding the user styling requirement. There is no user styling option in this prototype. However, styling options could be implemented through a user interface allowing the user to define new styling parameters that would be then sent through the SLD document. It is also common to use different symbolization to adapt to the scale and although the Web Map Service can deal with it, the actual combination with the GCS and its tiling mechanisms does not offer a satisfying solution yet. The resolution of the tile and where they cut depends on several factors. By increasing the size of the tile, there is more chance (but no certainty) that the user only sees one tile, but the loading of the image is slower. Depending on the scale threshold in which a tile is found, the resolution of the image can vary and thus messing up the scale-based symbology.

It is to expect that, when the standards for 3D web services are available, they be fully compatible with WMS and SLD. The multiple and coordinated views were not implemented in this prototype, but

it would be feasible by sending two requests with the same extent, but different layers and style content.

JavaScript allows for a wide range of interactive tools, from spatial to thematic navigation. However, the function that display additional information when going over a feature with the mouse (tooltips or highlight effects, for example) are not directly possible in this case, because only the representation of the features are displayed, and not the features themselves. As the client only receives the representation, as a picture, from the server, it is also not possible to process or query the data directly. The spatial navigation could be more intuitive. Although the panorama mode offer many navigation tools, it lacks for instance the panning tools which is really practical and highly intuitive. It could be implemented by taking into account the perspective view and calculating the approximate position of the center point after a panning action.

7.2.2. Combination of WebGL and QGIS Server

WebGL itself is part of the GUI, however the block diagrams are web-based because they request the texture from the WMS. It does not need any plugin, while being supported by all major browser (except IE). However, WebGL is not fully cross-platform as the rendered output is not always exactly the same in the different browsers. Another issue that might compromise the future of WebGL lies in the fact that it is not supported by Microsoft (for security reasons) and is not available for the public on iOS 5 devices and likely not on iOS 6 devices too²¹.

Regarding the interoperability and integration, there is still a few issues, especially with SVG. Whereas SVG is by definition scalable, the canvas element does not rescale with the browser window and its placement within the GUI is not optimal. The canvas element positions itself above all other elements, no matter where the element in the DOM is and does not stay within a SVG frame or window when scaling the browser window. The loadProject function could be adapted to recalculate the size and position of the canvas element after each resize of the window, but it s only a fix and does not solve the problem entirely. One could also imagine using an iframe element, but the same problem arise in the prototype as with the canvas element.

The number of vertices in the browser is limited, which means that for a high resolution DEM, the extent has to be smaller: higher resolution means more point per surface. For instance, a DEM with a 25m grid allows to draw a block diagram up to 6'300 x 6'300 m², whereas the RIMINI DEM with a grid of 250m allows to draw a block diagram up to 63'000 x 63'000 m². For a higher resolution and bigger extent, it is possible to draw several blocks next to each other. It then would require to adjust the navigation function to make them move, scale and rotate as one.

WebGL is a low-level API and thus does not provide for high level of abstraction, but the data could be created in a high-level application and then transferred into WebGL.

Due to the limitations on the vertices number, the block diagrams display at a rather large scales. Because, the WMS is identically used for the texture, the same down sides appear: symbols and labels are distorted and flatten onto the surface, which is not an ideal representation (see figure 26). Regarding 3D symbolization, WebGL has an advantage over a combination of a WMS with a 3D web service, because the 3D symbols can be directly realized with WebGL (see figure 27).

²¹ <http://caniuse.com/webgl>

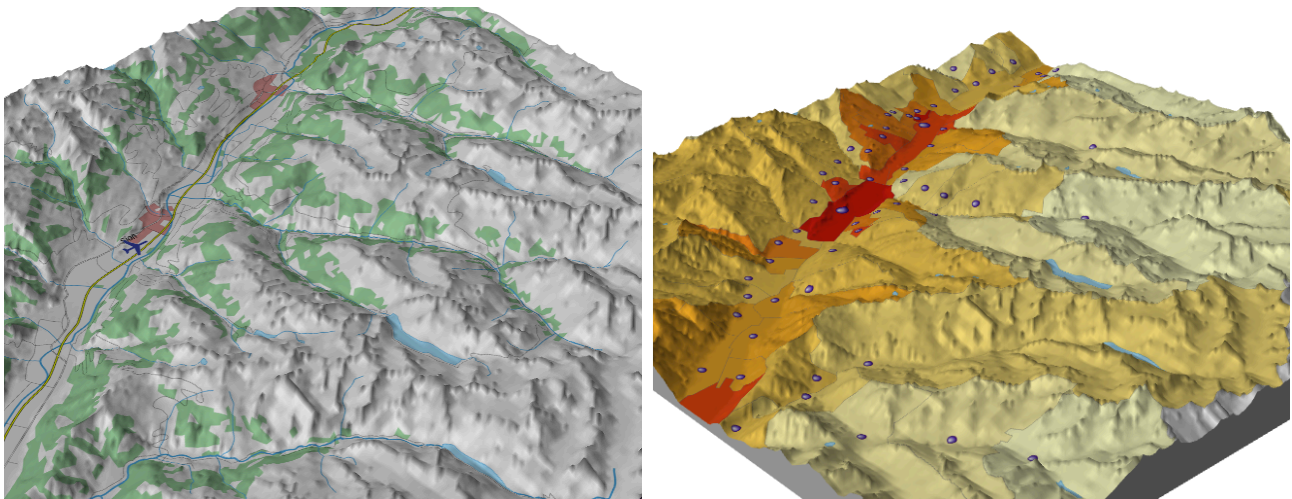


Figure 26: Block diagrams, left: base map, right: population density and inhabitants

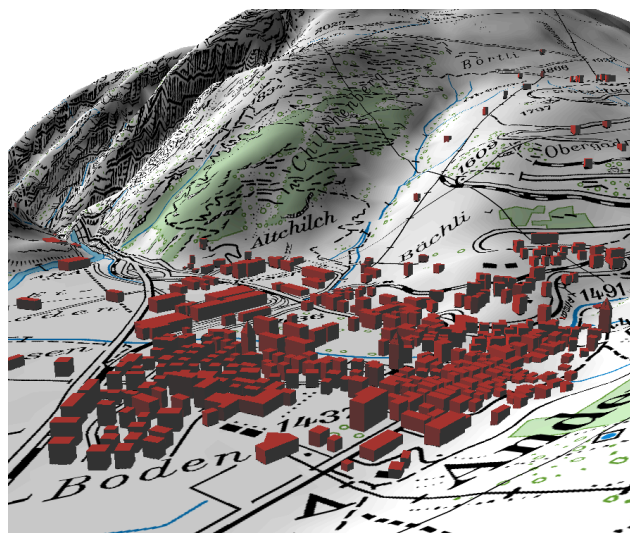


Figure 27: 3D symbolization of the buildings in Andermatt with WebGL (Friedli, 2012)

Remarks regarding user styling, and multiple and coordinated views from the panorama view mode also apply for the block diagrams. On one hand because the symbolization comes from the same WMS and on the other hand because the coordination of different block diagrams could be done by linking the navigation function of both diagrams.

Interactivity with WebGL has a significant potential because WebGL also uses JavaScript. For instance, tooltips have been implemented in an anatomy project²², but still in beta version and they should be applicable to cartographical needs. Highlights and retrieval of thematic information with mouseovers and click could also be conceivable. Furthermore, the spatial navigation around the block diagram is very intuitive. The user can rotate the block with the mouse and zoom in and out easily.

Regarding data processing and query, WebGL might have an advantage against the use of strict image, because it already works with attributes. However, it has not been tested yet.

²² <http://radeberger.cs.uni-magdeburg.de/~sbirr/web3D/showcase/index.php?site=start>

7.2.3. Summary

The following table describes the advantages and weaknesses of the two modes, as well as the expected results with fully standardized WVS and WMS. It takes each requirement defined in chapter 5 and states whether it can be considered as an advantage (dark green = tested in the prototype, light green = theoretically possible), an area where improvements are needed (orange) or a weakness (red).

Table 4: Assessment of the different requirements for 3D geovisualization systems

	GCS + WMS	WebGL + WMS	WVS + WMS
R1: system oriented architecture and thin client			
R2: no plugin			
R3: cross-platform			
R4: interoperability, integration			
R5: extensibility and update			
R6: reusable and robust			
R7: open source			
R8: support for massive geodata			?
R9: dynamic geodata			
R10: higher level of abstraction			
R11: high quality and effective visualization			?
R12: user styling options			
R13: coordinated and multiple views			
R14: interactivity			?
R15: intuitive navigation			?
R16: data query and processing			

The main weakness of the GCS is found by the interoperability and integration requirements. Indeed, there is no standard regarding 3D web service or Web View Service yet and it hinders the full integration of the WMS. For instance, the GCS does not allow for an easy and flexible change of layer combination, preventing to fully apply the cartographic principle that states that each layer should be independent. Here, only predefined combinations of layers can be used. Additionally, the definition of different symbolizations based on different scales in WMS is not taken into account by GCS due to how the perceptive view is generated. There are also some issues regarding the performance, especially in the area of speed of display and resolution of the image. However, these issues can be solved by optimizing the request process.

Regarding WebGL, the main weakness beyond the size limitations is the lack of wide support among popular browsers. Only Chrome offers full support, whereas the latest versions of Safari, Firefox and Opera only offer partial support. The support for mobile devices is even lower²³. WebGL also encounter some issues with interoperability and integration with SVG.

Requirements 12, 13 and 16 have not been tested in the prototype, but they should theoretically works. Styling options and retrieval of information about the features are already possible with WMS (see section 7.3) and coordinated views could be done by requesting to perspective view with the same spatial parameters while modifying the others.

²³ <http://caniuse.com/webgl>

The last column shows the expected capabilities of a combination between a WMS and a WVS, when the standard is available. All issues of interoperability and reusability should be solved, but it stays the question regarding performance for cartography visualization because they depend on the actual implementation of the standard. It can be seen that a standard is highly needed because it will resolve the major issues of integration with the 2D WMS for the texture, with any external sources of data and any other web services that could be needed, such as a WPS or hypothetical cartographical web services for symbolization.

7.3. Recommendations for the Architecture

The prototype in this work tested two different approaches to service-driven 3D atlas cartography. This section provides possible solutions to resolve their different weakness points and highlights the important differences between the two approaches.

The first approach solely based on web services puts less strain on the the GUI as the WebGL approach, which means that it is better suited for projects that require very light clients. The approach of combined web services has two major drawbacks: the 3D web service, Globe Capture Service, is not standardized and the display of symbols is not adequate for cartographic use. There are great expectations on the coming of a standard for 3D web services and it has been promised to be agreed upon soon. This would solve the problems regarding the interoperability, especially with other web services. It raises the question of how much resources should be put into improving the GCS against waiting for a standard that can be available tomorrow as well as in one year. As it is expected to see a significant evolution in the world of 3D web services, it might be worth to wait until the standard is available in order to implement a long-term and extensive project.

The lack of ability to display symbols properly on the landscape can be seen as the main obstacle to the use of web service for 3D cartography. This could be solved by using a web service delivering georeferenced billboards (see figure 28). Such billboard are not implemented yet, but should theoretically work with osgEarth, on which GCS is based and which could be the input data for a standardized 3D web service. These symbols would be always turning toward the camera and would not be subject to the relief, thus avoiding foreshortening effect. 2D symbols such a bar charts and billboards in 3D geovisualization environment have show their usefulness (Bleisch, 2011).

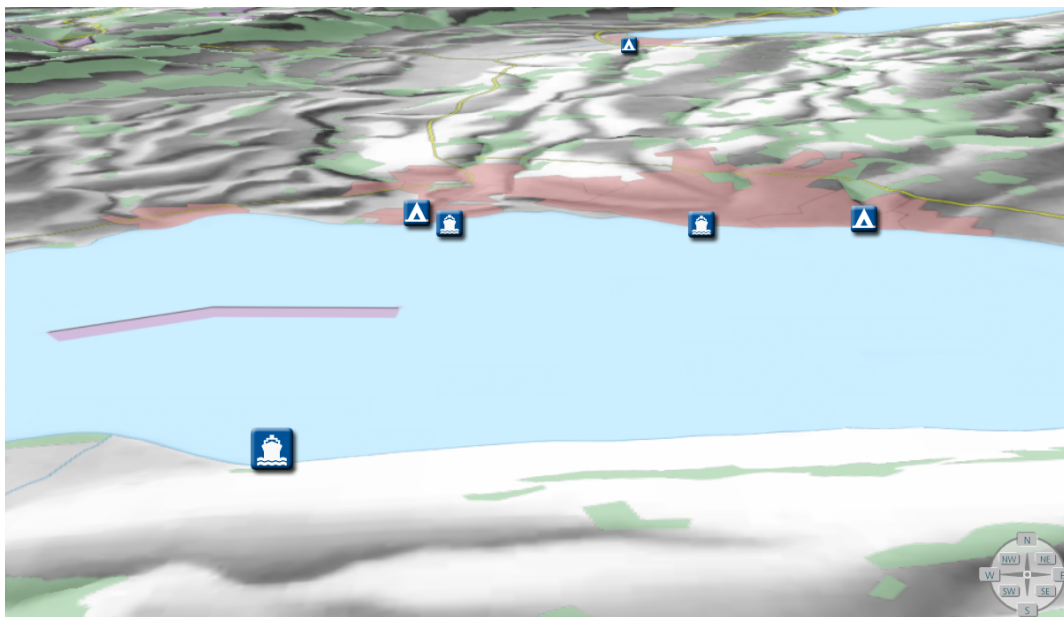


Figure 28: Billboards - a possible solution for 2D and 3D symbols with the web service approach

Regarding the interactivity with the image displaying the relief, it can be imaged to use the request `GetFeatureInfo` from the WMS standard that delivers feature information by identifying a point on the map via its pixel location. Due to the third dimension, it would require ray-tracing to define the actual location of the screen pixel on the relief. Another possibility is to use the principle in WPVS++ using image to encode information, but it seems that the OGC is going away from that idea.

The approach with WebGL demands more from the client and one has to be aware that it gets complicated to render large parts of landscape at high resolution due to the limitation of 64k vertices. Nonetheless for large scale representation, it allows to render the landscape and any symbolization with the same technology. It would be perfectly adapted for small and well defined areas. In atlases, this could be specific points of interest that require their own map for instance.

WebGL has a rather complicated pipeline and programming interface, thus one might consider creating the content in another format before exporting it to WebGL. It is possible to prepare the geodata with the help of another software and then to display them with WebGL in the browser. It was demonstrated that it is possible to use Sketch Up to create the 3D content, then to export it into the COLLADA exchange format, which can be loaded into WebGL (Friedli, 2012).

In a SOA, it is always possible to add more web services with different functions and because symbols and labels are not displayed in a satisfactory way with WMS, creating a symbols web service or even a 3D symbols web service goes in the direction of a fully web-based atlas.

8. Conclusions and Outlook

In this last chapter, the conclusions of this Master thesis are drawn and outlook in the future developments in the area of service-driven 3D atlas cartography is given.

8.1. Conclusions

This Master thesis shows that service-driven 3D atlas are doable and that a service-oriented architecture bring benefits for 3D atlases.

The first part reviews the literature and existing digital 3D atlases about atlas functionalities and 3D geovisualization and it gives an overview of the available specifications and graphics formats for 3D visualization. It is found that 3D visualization can represent certain advantages over 2D geovisualization, especially in the area of shape understanding and task orientation. Furthermore, it can be said that the multitude of 3D graphics formats and technologies renders any attempt at standardization or interoperability quite tedious.

This review process enables the definition of sixteen requirements for service-driven 3D atlases based on two categories: system requirements that are non-specific to 3D geovisualization and requirements about visualization and interactivity that are specific to atlases and 3D geovisualization. Then, this thesis explores some possibilities for service-driven 3D atlas and shows that existing specifications can be used to implement a prototype based on the above mentioned requirements. The concept of using a 3D web service, in combination with a Web Map Service for the texture, to display a panorama view is valid, although it shows its limitation for 3D symbols and point symbology in general. The Web Map Service and its extension for cartographic symbolization allow to display complex texture in the landscape. The use of WebGL for block diagram with a Web Map Service deals with the same issues regarding point symbology. In addition, it encounters a trade-off between the extent of the block diagram and its DEM resolution.

Nevertheless, as long as there is no standard for a 3D web service, issues of interoperability and integration will necessary arise. Web atlases could highly benefit from a service-driven 3D atlas that would allow not only access to the spatial data even from light client, but also access to external data sources. The latest offers interesting possibilities regarding the access to the data representation versus the acquisition of the actual data, which is often expensive.

Specifications for 3D geovisualization, especially for 3D web services, are not fixed yet and interoperable solutions are not easy to define. The development of service-driven atlas could use the definition of standards for 3D web services as soon as possible.

The performance, in term of speed and visual quality, of the prototype does not reach the one of desktop application yet. However, the performance was not a criteria for this work and it is reasonable to expect that the prototype could be highly optimized, especially when standards are available.

8.2. Outlook

The use of web services delivering the different parts of the visualization definitely goes in the direction of the actual tendency of devices with little processing capacity combined with all-streaming or online applications. It is to be expected that more and more atlases go online. Furthermore, the attractiveness of 3D atlas is not just a trend effect; 3D visualization does bring real advantages for geodata. In addition, it allows for 3D symbology and a whole new world of possibilities

for point symbolization. Thus, atlases should not overlook what 3D geovisualization can bring for their future developments.

Although it is possible to develop a service-driven 3D atlas without standards (for instance with proprietary software or in-house development), the availability of standards and their use highly simplify the integration and interoperability processes not only within the architecture of the atlas, but also for external applications or user data. A full-fledged service-driven 3D atlas using standards might also prove itself more robust and interoperable in the long run. The combination and chaining of web services allows for complex workflows and they can be hindered by the lack of interoperable solutions if no standards are available. For instance, this prototype cannot be said to be a full-fledge service-based 3D atlas because it does not offer a satisfactory solution for point and label symbology and because it does not have access to DEM on the fly for block diagrams. Web services delivering 3D symbols, billboards or DEM on the fly are a logical step toward this direction.

Regarding WebGL, it is important to follow its development and its support in the future, because its capability for large-scale block diagrams is attractive and could be used once cross-platform issues are solved. The use of WebGL for 3D objects is also interesting and should be further explored. At the moment, WebGL is not an ideal solution for a complete atlas, but it could play specific roles, such as 3D objects and symbols or local geovisualization of specific spatial phenomenon.

This work did not explore the virtual globe in its implementation section, but the profusion of virtual globes that are being developed shows the interest for this solution as well. They might be relevant for world atlas, but maybe less for national atlas.

The use of 3D geovisualization for thematic data is rather new and a next challenge is to assess the strengths and weaknesses of panorama view,s block diagrams and virtual globes for thematic representation. Furthermore, the perception of quantitative and qualitative data visualized on a 3D landscape should be explored, as well as its meaning for the understanding and knowledge building processes of the users.

Bibliographic References

- Alonso, P. G. (1968): The First Atlases. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 5(2), 108-121.
- Altmaier, A., and T. H. Kolbe (2003): Applications and Solutions for Interoperable 3d Geo-Visualization. *Proceedings Photogrammetric Week*. Wichmann Verlag. Stuttgart. 253-267.
- Andrienko, G., N. Andrienko, J. Dykes, D. Mountain, P. Noy, M. Gahegan, J. C. Roberts, P. Rodgers, and M. Theus (2005): Creating Instruments for Ideation: Software Approaches to Geovisualization. In: J. Dykes, A. M. MacEachren and M.-J. Kraak (Ed.) *Exploring Geovisualization*. Elsevier Ltd. 103-125.
- Autodesk Inc. (2012): Autodesk FBX - Data Interchange Technology. <http://usa.autodesk.com/fbx/> (retrieved on 12.06. 2012).
- Bär, H. R. (2011): Übungsmaterial zur Vorlesung "Geovisualisierung". Place: Institut für Kartografie und Geoinformation, ETH Zürich.
- Bär, H. R., and R. Sieber (1997): Atlas der Schweiz – Multimedia Version : Adaptierte GIS-Techniken und qualitative Bildschirmgraphik. In: Grünreich and Dietmar (Ed.) *GIS und Kartographie im multimedialen Umfeld*. Bonn. Kirschbaum Verlag GmbH.
- Beard, D. J., H. R. J., M. G. Nicoll, and D. O. Edge (2005): 3D Wep Mapping – 3D Geoscience Information Online. *Proceedings SSC 2005 Spatial Intelligence, Innovation and Praxis: The National Biennial Conference of the Spatial Sciences Institute*. Spatial Sciences Institute. Melbourne.
- Behr, J., Y. Jung, T. Drevensek, and A. Aderhold (2011): Dynamic and interactive aspects of X3DOM. *Proceedings 16th International Conference on 3D Web Technology*. ACM. Paris, France. 81-87.
- Bleisch, S. (2011): Evaluating the appropriateness of visually combining quantitative data representations with 3D desktop virtual environments using mixed methods. *Doctoral Dissertation*, City University London.
- Bleisch, S., and J. Dykes (2006): Planning Hikes Virtually – How Useful are Web-based 3D Visualizations? *Proceedings GIS Research UK 14th Annual Conference*. Nottingham, UK. 313-318.
- Bleisch, S., and S. Nebiker (2008): Connected 2D and 3D Visualizations for the Interactive Exploration of Spatial Information. *The International Archive of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXVIII(B2), 1037-1042.
- Borchert, A. (1999): Multimedia Atlas Concepts. In: W. Cartwright, M. P. Peterson and G. Gartner (Ed.) *Multimedia Cartography*. Berlin Heidelberg. Springer Verlag. 75-86.
- Brodie, K. W., et al. (2007): Adaptive Infrastructure for Visual Computing. *Computer Graphics Forum*, 23(2), 223-251.
- Caballero, L. (2011): An Introduction to WebGL. <http://dev.opera.com/articles/view/an-introduction-to-webgl/> (retrieved on 23.05 2012).

- Collada Community (2011): COLLADA - Digital Asset and FX Exchange Schema. https://collada.org/mediawiki/index.php/COLLADA_-_Digital_Asset_and_FX_Exchange_Schema (retrieved on 12.06.2012).
- Cron, J. (2006): Graphische Benutzeroberflächen interaktiver Atlanten: Konzept zur Strukturierung und praktischen Umsetzung der Funktionalität. Diplomarbeit, Hochschule für Technik und Wirtschaft Dresden, Dresden, Germany.
- Dahlström, E., P. Dengler, A. Grasso, C. Lilley, C. McCormack, D. Shepers, J. Watt, J. Ferraiolo, J. Fujisawa, and D. Jackson (2011): Scalable Vector Graphics (SVG) 1.1 (Second Edition). <http://www.w3.org/TR/SVG/> (retrieved on 10.04.2012).
- de Ferranti, J. (2012): Digital Elevation Data. <http://www.viewfinderpanoramas.org/dem3.html> (retrieved on 06.06.2012).
- de Vries, M., and S. Zlatanova (2004): Interoperability on the Web: The Case of 3D Geo-Data. Proceedings IADIS International Conference e-Society. Avila (Spain). 667-674.
- Döllner, J. (2005): Geovisualization and real-time 3D computer graphics. In: J. Dykes, A. M. MacEachren and M.-J. Kraak (Ed.) Exploring Geovisualization. Amsterdam. Elsevier Ltd.
- FreeWRL (2012): FreeWRL/FreeX3D Home Page. <http://freewrl.sourceforge.net> (retrieved on 10.04.2012).
- Friedli, R. (2012): Topographische 3D-Signaturen: Gestaltung und Implementierung. Projektarbeit Masterstufe, ETH Zürich, Zürich.
- George, R. (2006): Scalable Vector Graphics Interfaces for Geographic Applications. In: E. Stefanakis, M. P. Peterson, C. Armenakis and V. Delis (Ed.) Geographic Hypermedia – Concepts and Systems. Heidelberg. Springer Verlag.
- Hagedorn, B. (2010a): Web View Service Discussion Paper. OpenGIS Discussion Paper, Open Geospatial Consortium. http://portal.opengeospatial.org/files/?artifact_id=37257.
- Hagedorn, B. (2010b): Web View Service, an Interactive Image-Based 3D Portrayal Service: Potentials. <http://www.webviewservice.org/potentials> (retrieved on 20.03.2012).
- Hagedorn, B., D. Hildebrandt, and J. Döllner (2010): Towards Advanced and Interactive Web Perspective View Services. In: T. Neutens and P. D. Maeyer (Ed.) Developments in 3D Geo-Information Sciences, Lecture Notes in Geoinformation and Cartography. Berlin Heidelberg. Springer Verlag
- Hagedorn, B., A. Zipf, A. Schilling, and S. Neubauer (2008): 3D Portrayal Services – Use Cases. OpenGIS Discussion Paper, Open Geospatial Consortium.
- Hetherington, R., B. Farrimond, and P. Clynych (2007): Interactive Web Visualisation of Proposals for Site Developments. Proceedings 11th International Conference Information Visualization. 613-622.
- Hildebrandt, D. (2008): Towards the Automatic Generation of Effective, Map-Like Visual Representations from Heterogeneous Geodata in a Service-Oriented Infrastructure. In: C. Meinel, H. Plattner, J. Döllner, M. Weske, A. Polze, R. Hirschfeld, F. Naumann and H. Giese (Ed.)
-

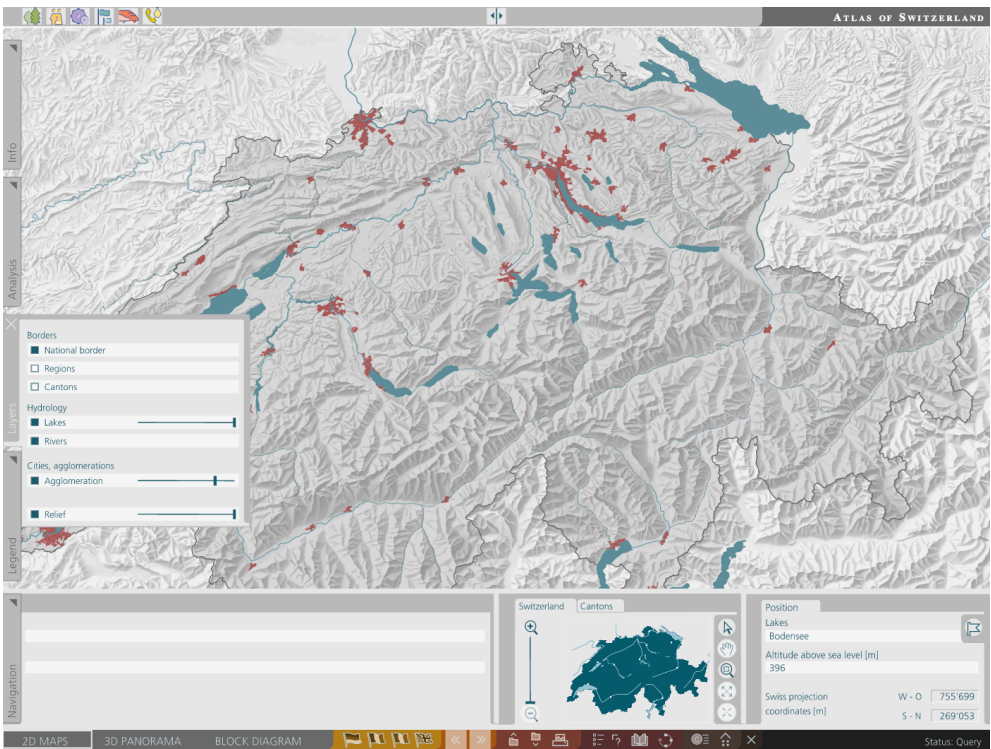
- Proceedings of the 3rd Ph.D. Retreat of the HPI Research School on Service-oriented Systems Engineering. Universitätsverlag Potsdam. 15.11-15.12.
- Hildebrandt, D., and J. Döllner (2010): Service-oriented, standards-based 3D geovisualization: Potential and challenges. *Computers, Environment and Urban Systems*, 34(6), 484-495.
- Hildebrandt, D., B. Hagedorn, and J. Dollner (2011): Image-based strategies for interactive visualisation of complex 3D geovirtual environments on lightweight devices. *Journal Location Based Service*, 5(2), 100-120.
- Hurni, L., R. Sieber, R. Eichenberg, L. Hollenstein, and B. Odden (2011): Atlas der Schweiz. Konzept 2012-2016 und 2017-2020. Institut für Kartografie und Geoinformation IKG, ETH Zürich.
- Iosifescu-Enescu, I. (2011): Cartographic Web Services. Dissertation, ETH Zurich, Zurich, Switzerland.
- Khronos WebGL Working Group (2012): OpenGL ES - The Standard for Embedded Accelerated 3D Graphics. <http://www.khronos.org/opengles/> (retrieved on 04.05. 2012).
- Kraak, M.-J., F. Ormeling, W. Broeder, E. MacGillavry, and W. v. d. Goorberg (2007): The Dutch National Atlas in a GII environment: the application of design templates. Proceedings 23rd International Cartographic Conference of the Cartographic Association. Moscow (Russia).
- Lyubka, S. (2012): mongoose. <http://code.google.com/p/mongoose/> (retrieved on 25.05. 2012).
- Meng, L. (2003): Missing Theories and Methods in Digital Cartography. Proceedings 21st International Cartographic Conference. ICA. Durban.
- Neumann, A., and A. M. Winter (2011): Carto:net – SVG, scalable vector graphics: tutorials, examples, widgets and librairies. <http://www.carto.net/> (retrieved on 01.06. 2012).
- Open Geospatial Consortium (2006): OpenGIS® Web Map Service, Implementation Specification. Version 1.3.0. <http://www.opengeospatial.org/standards/wms> (retrieved on 15.04. 2012).
- Open Geospatial Consortium (2012): OGC KML. <http://www.opengeospatial.org/standards/kml> (retrieved on 20.03. 2012).
- Ormeling, F. (1997): Functionality of Electronic School Atlases. Proceedings Seminar on Electronic Atlases II. ICA Commission on National and Regional Atlases. Prague and The Hague. 33-41.
- Ortner, F. (2011): Dienstbasierte Kartengenerierung für Web-Atlanten unter Anwendungen erweiterter OGC-Standards Universität Zürich, Universität Zürich.
- OSG Community (2007): OpenSceneGraph - Introduction. <http://www.openscenegraph.org/projects/osg/wiki/About/Introduction> (retrieved on 15.04. 2012).
- Pelican Ventures, I. (2012): osgEarth – Terrain On Demand. <http://osgearth.org/> (retrieved on 19.03. 2012).
- Persson, D., G. Gartner, and M. Buchroithner (2006): Towards a Typology of Interactivity Functions for Visual Map Exploration. In: E. Stefanakis, M. P. Peterson, C. Armenakis and V. Delis (Ed.) *Geographic Hypermedia – Concepts and Systems*. Heidelberg. Springer Verlag. 275-292.

- Pulli, K. (2006): New APIs for Mobile Graphics. Proceedings SPIE Electronic Imaging: Multimedia on Mobile Devices II. 1-13.
- Quantum GIS (2012): QGIS Features. <http://www.qgis.org/en/about-qgis/features.html> (retrieved on 16.04. 2012).
- Ramos, C. d. S., and W. Cartwright (2006): Atlases from Paper to Digital Medium. In: E. Stefanakis, M. P. Peterson, C. Armenakis and V. Delis (Ed.) *Geographic Hypermedia – Concepts and Systems*. Heidelberg. Springer. 97-119.
- Rase, W.-D. (2003): Von 2D nach 3D – perspektivische Zeichnungen, Stereogramme, reale Modelle. *Kartographische Schriften, Band 7: Visualisierung und Erschliessung von Geodaten*, 13-24.
- Schilling, A., and T. H. Kolbe (2010): Draft for Candidate OpenGIS Web 3D Interface Standard. OpenGIS Discussion Paper, Open Geospatial Consortium. http://portal.opengeospatial.org/files/?artifact_id=36390.
- Schnabel, O. (2002): Conception for an online-National Atlas of Germany. Diploma Work, Dresden University of Technology.
- Sieber, R., L. Hollenstein, B. Odden, and L. Hurni (2011): From Classic Atlas Design to Collaborative Platforms – The SwissAtlasPlatform Project. Proceedings Proceedings of the 25th International Cartographic Conference.
- Singh, R. R. (2001): OGC Web Terrain Server (WTS). OpenGIS OGC Interoperability Program Report, Open GIS Consortium. http://portal.opengeospatial.org/files/?artifact_id=1072.
- St. John, M., M. B. Cowen, S. H. S., and O. H. M. (2001): The Use of 2D and 3D Displays for Shape-Understanding versus Relative-Position Tasks. *Human Factors*, 43(1), 79-98.
- Thomas, G. (2012): Learning WebGL. <http://learningwebgl.com/blog/?p=134> (retrieved on 23.05. 2012).
- Tory, M., A. E. Kirkpatrick, M. S. Atkins, and T. Moller (2006): Visualization Task Performance with 2D, 3D, and Combination Displays. *Visualization and Computer Graphics, IEEE Transactions*, 12(1), 2-13.
- Visual Size (2011): 3D Models Using Multiple Photos - Interactive Demos. <http://www.visualsize.com/3ddemo/index.idemo.php> (retrieved on 26.04. 2012).
- Walker, M., and P. Kalberer (2010): Comparison of Open Source Virtual Globe. Proceedings FOSS4G. Barcelona (Spain).
- Web 3D Consortium (2011): Web3D Product listing details. <http://www.web3d.org/products/detail/bs-contact-geo1/> (retrieved on 20.03. 2012).
- x3dom (2012): x3dom Instant 3D the HTML way! http://www.x3dom.org/?page_id=2 (retrieved on 20.03. 2012).
-

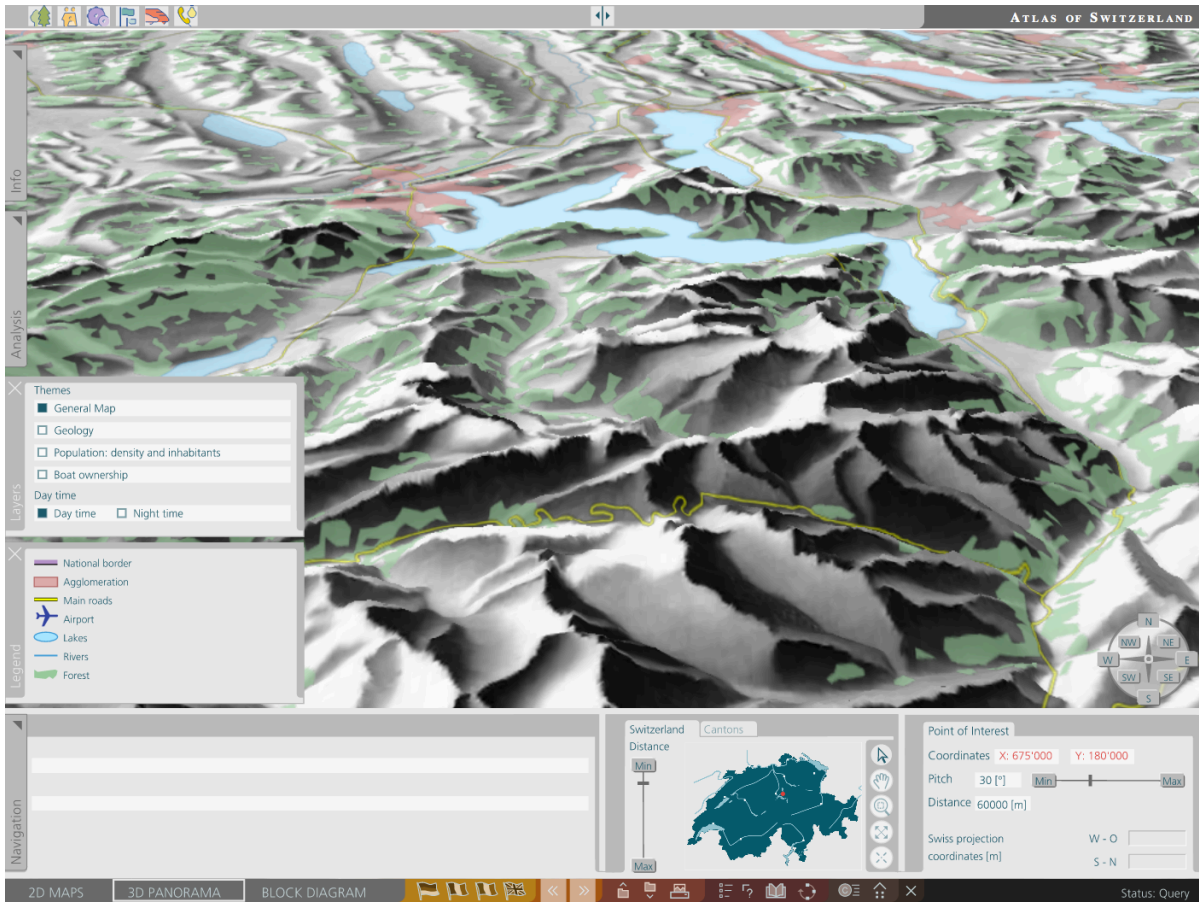
Appendices

I. Illustrations of the Prototype

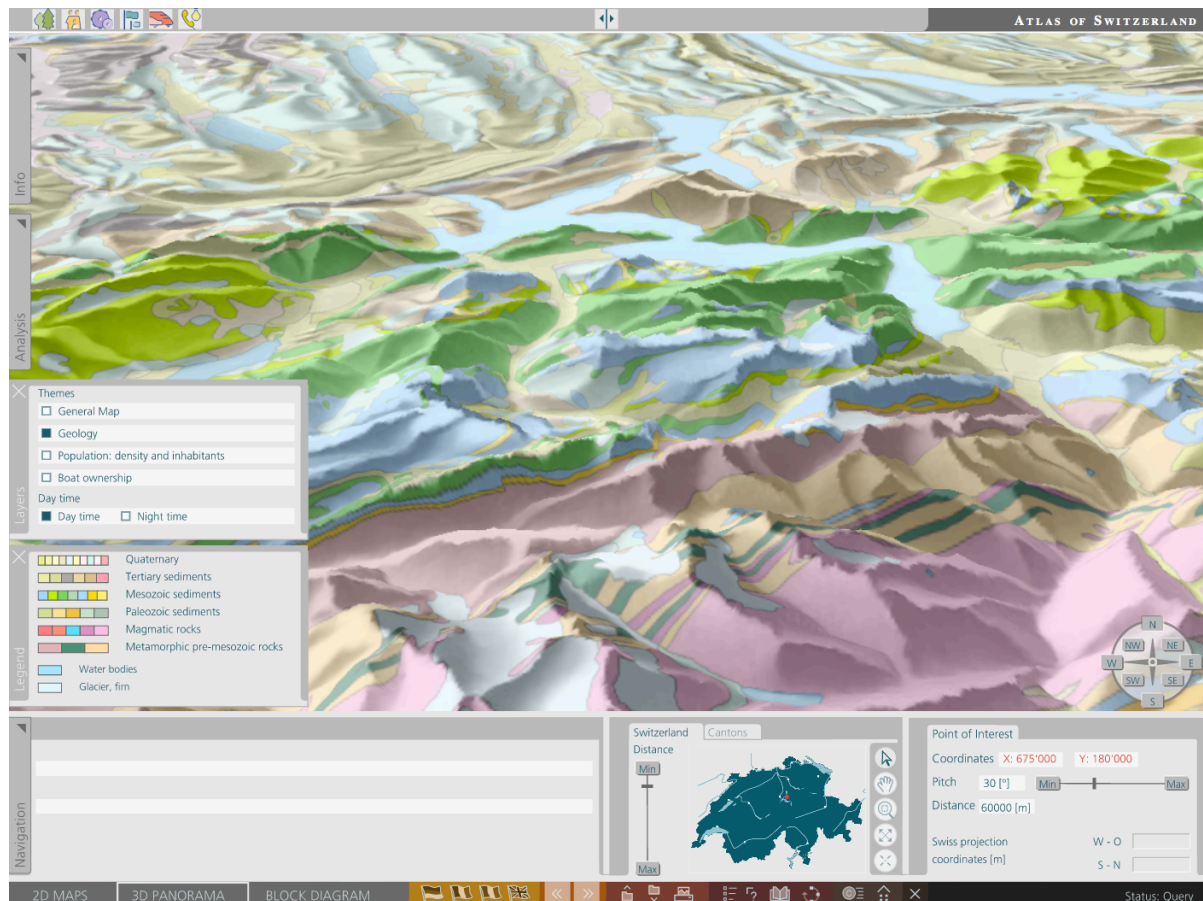
a. 2D Maps Interface



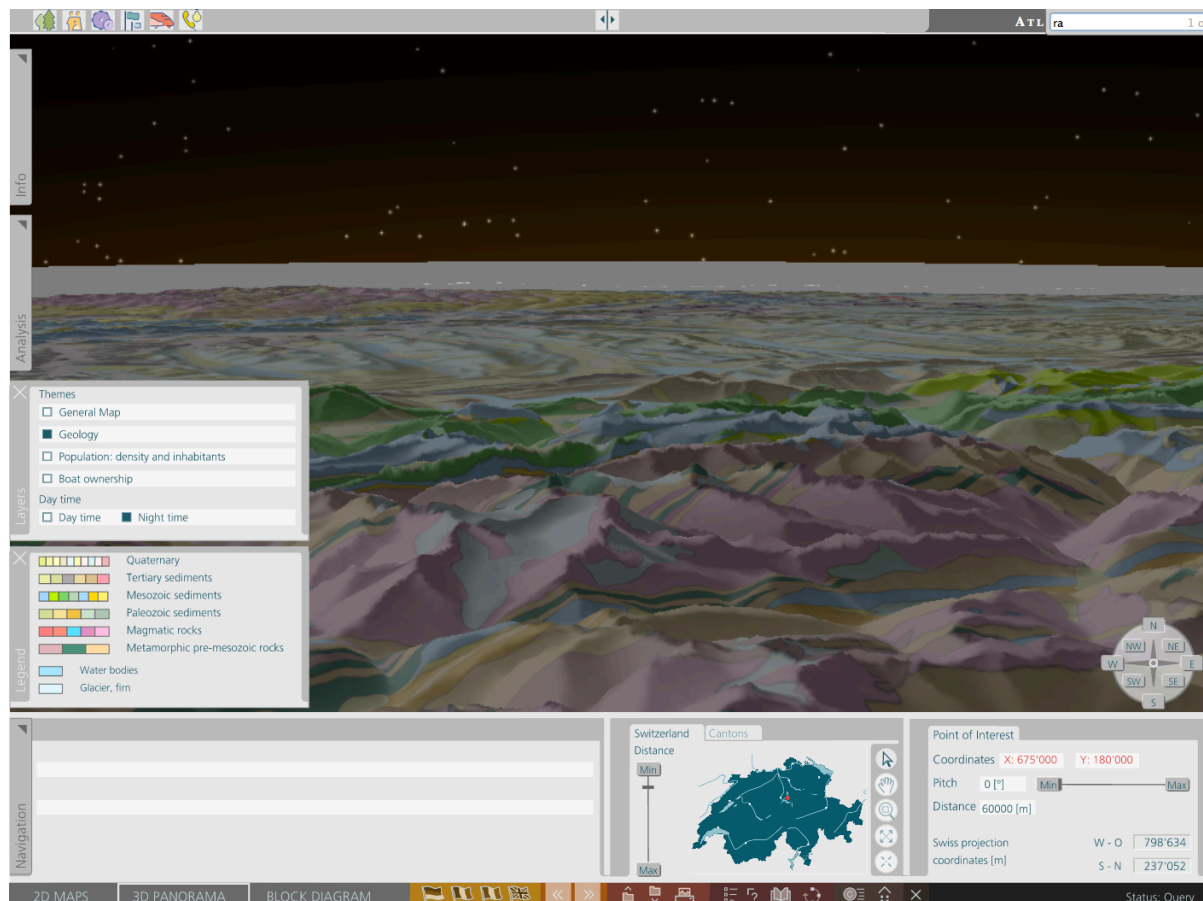
b. Panorama View: General Map



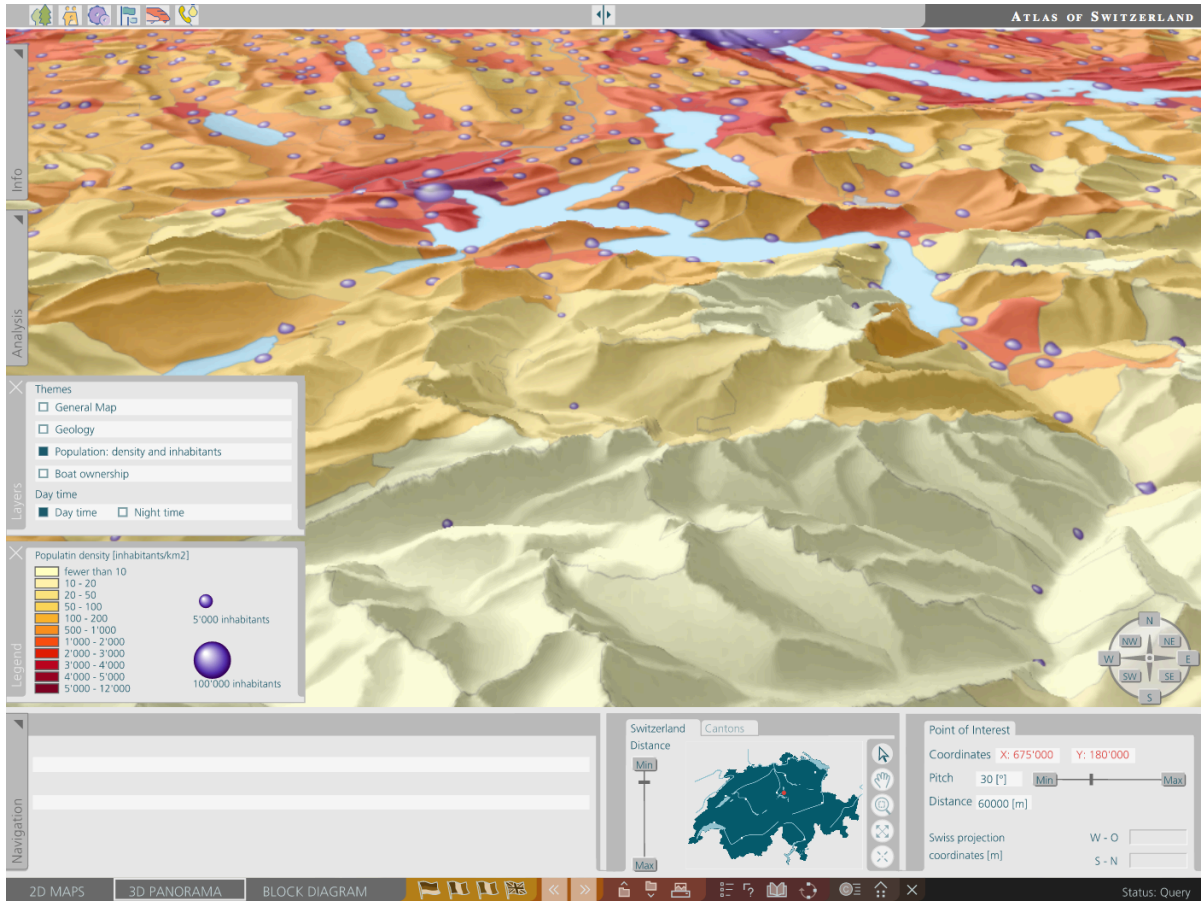
c. *Panorama View: Geology*



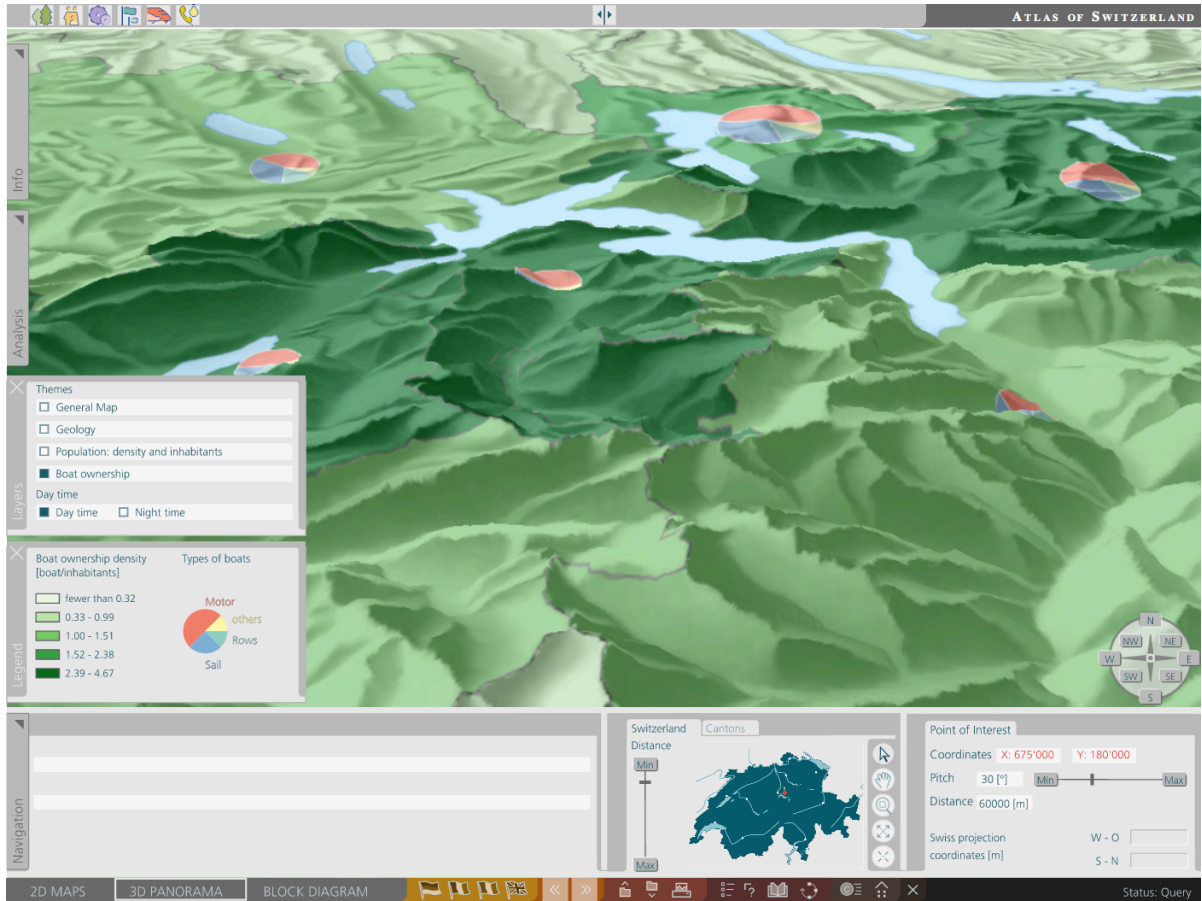
d. *Panorama View: Geology, by night*



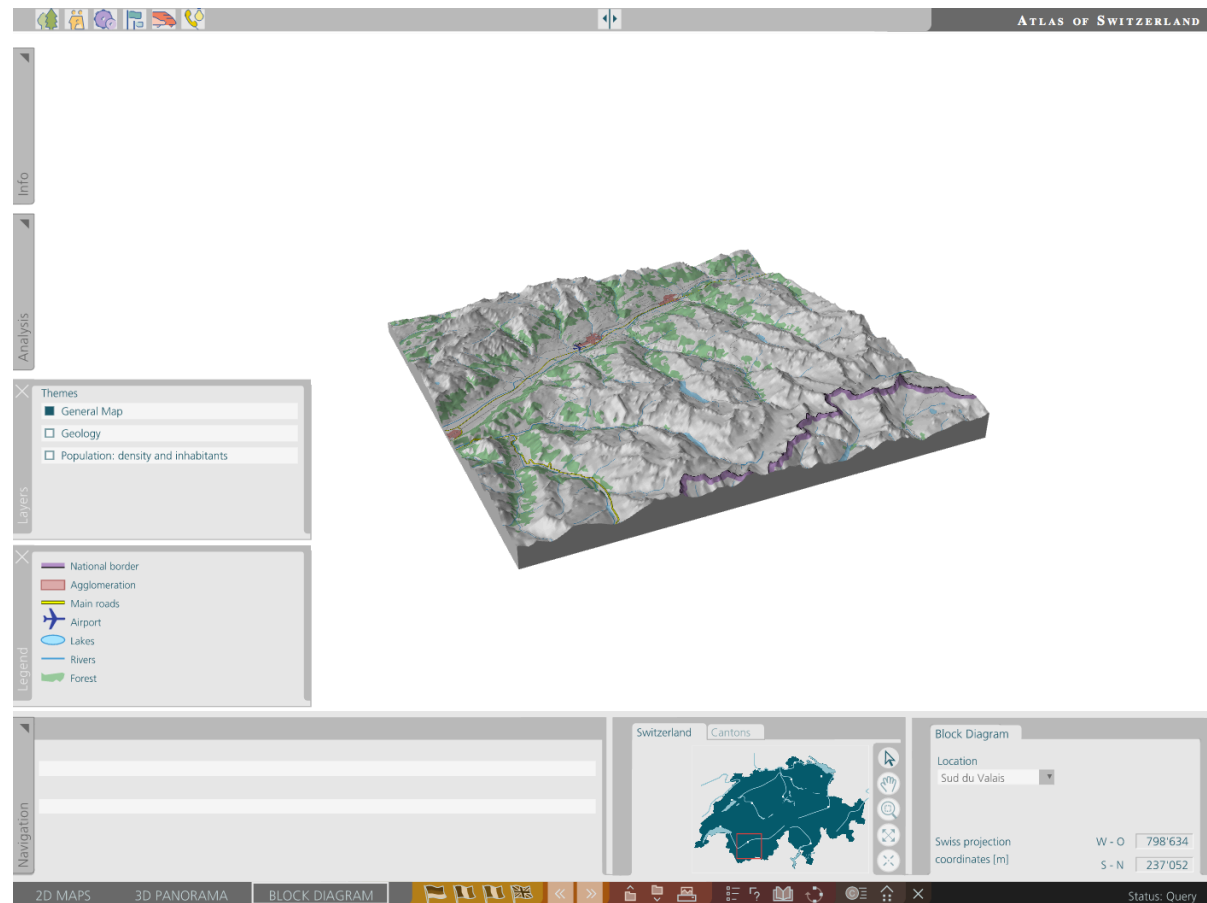
e. Panorama View: Population Density and Inhabitants



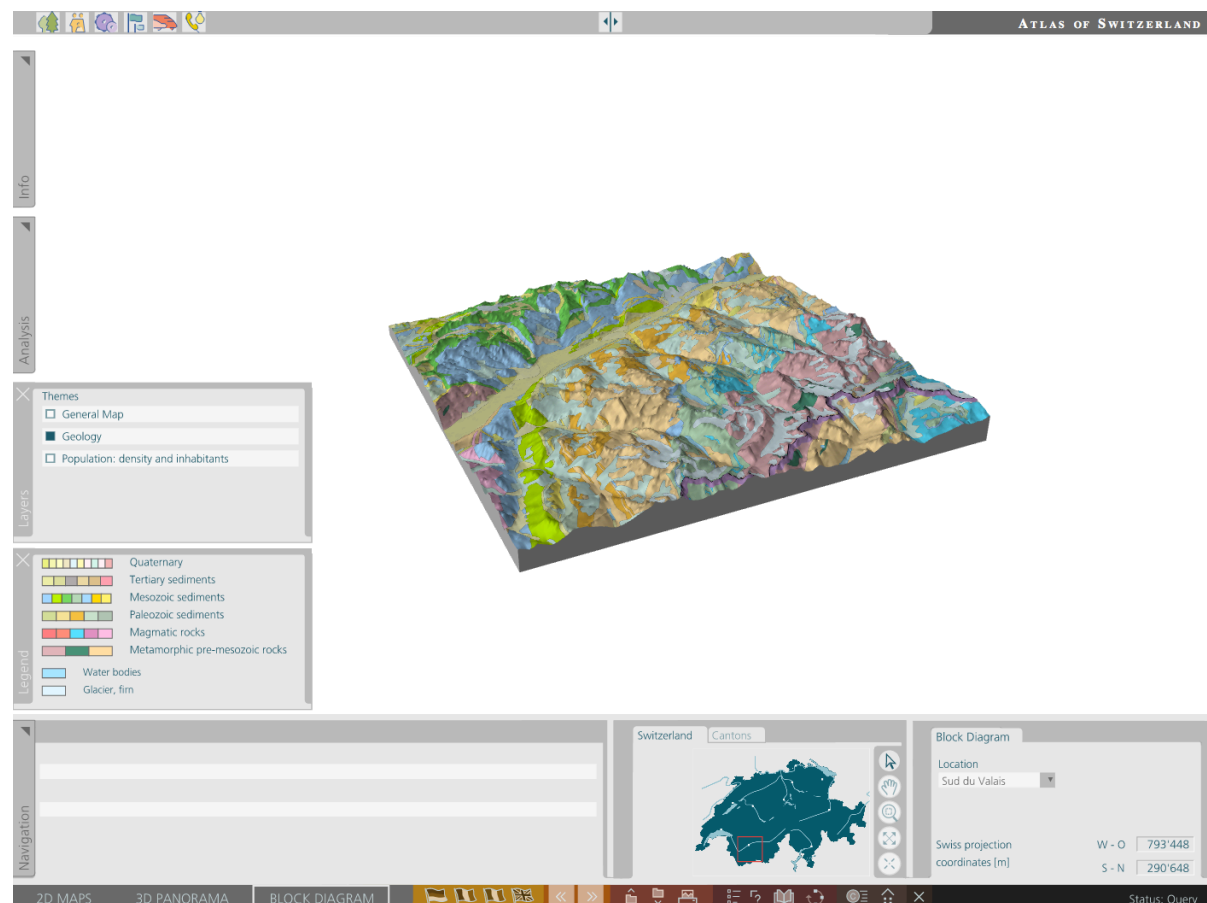
f. Panorama View: Boat Ownership Density and Types of Boats



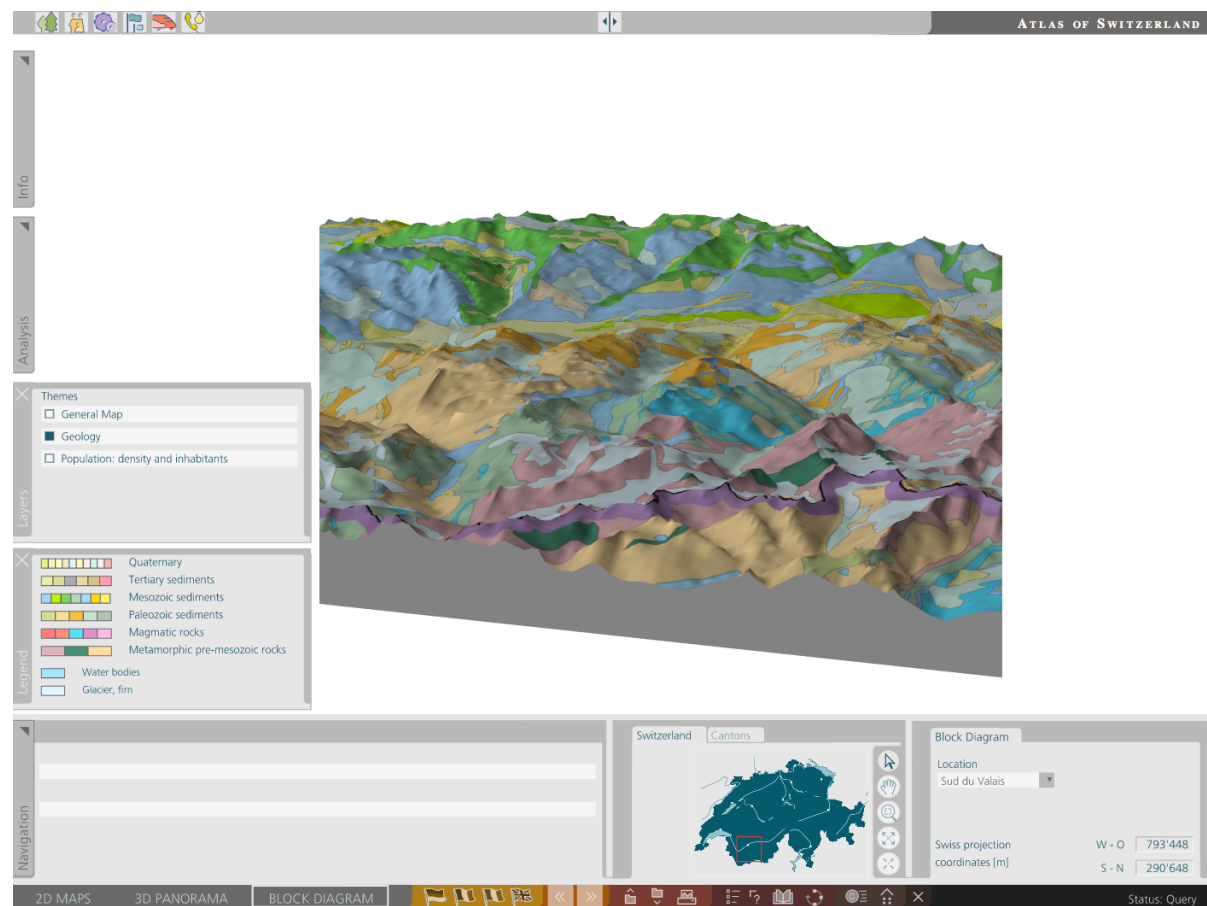
g. Block Diagram: General Map, Valais



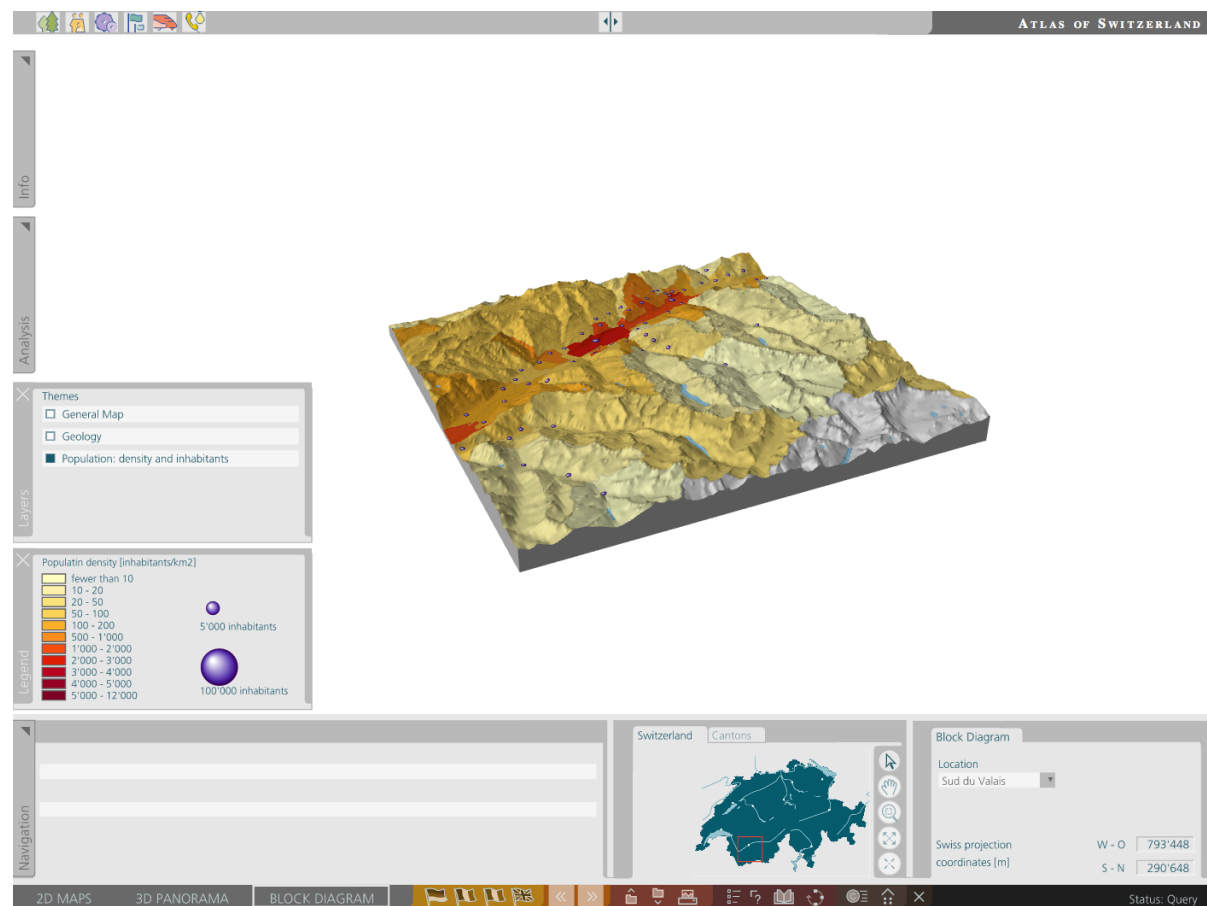
h. Block Diagram: Geology, Valais



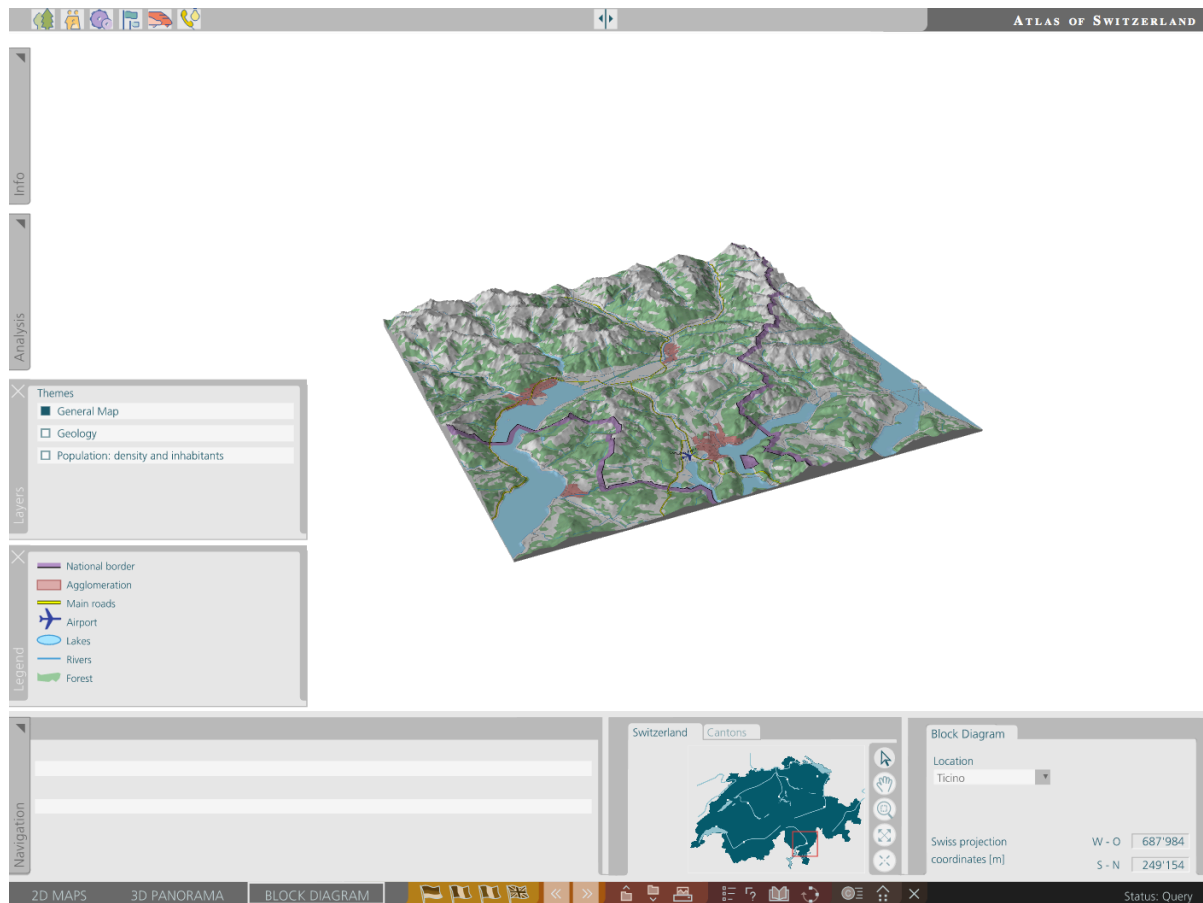
i. Block Diagram: Geology, zoomed in, Valais



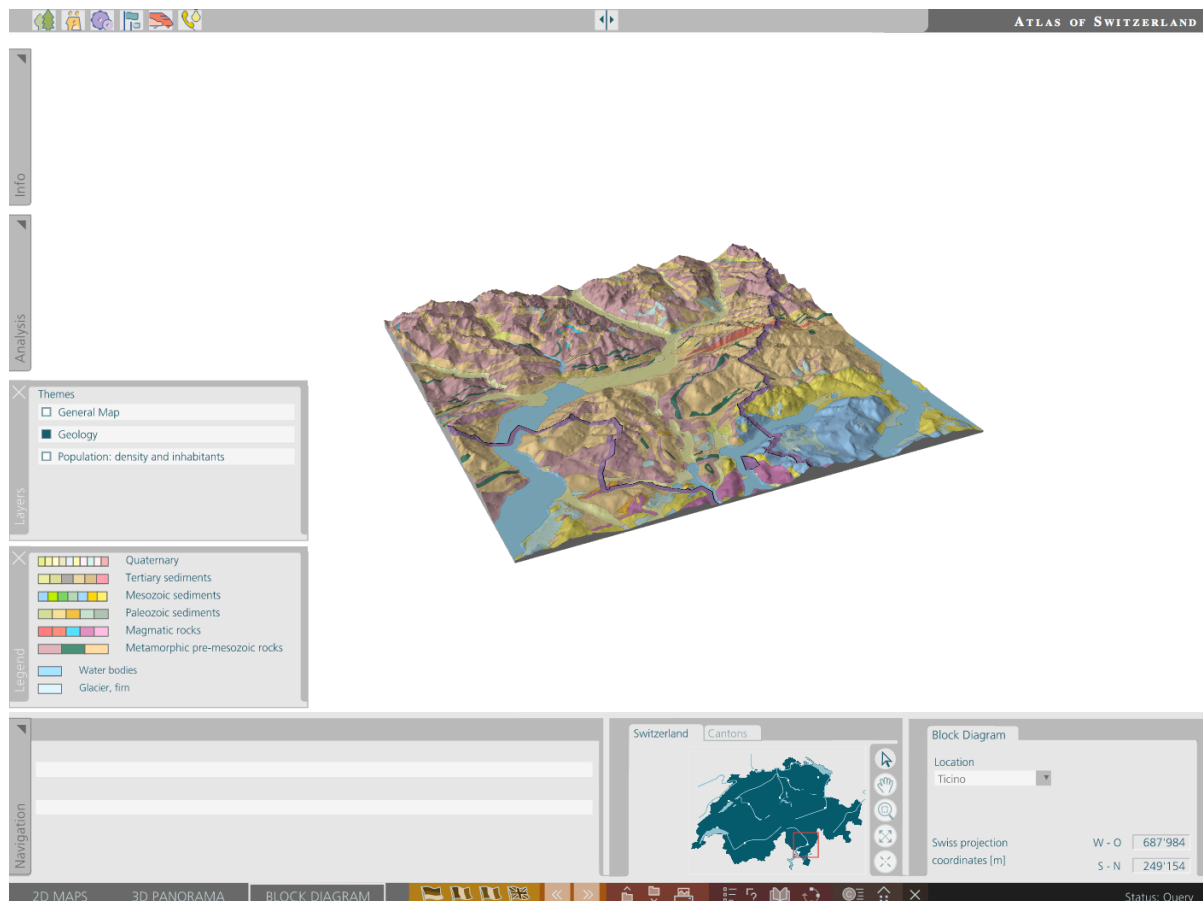
j. Block Diagram: Population Density and Inhabitants, Valais



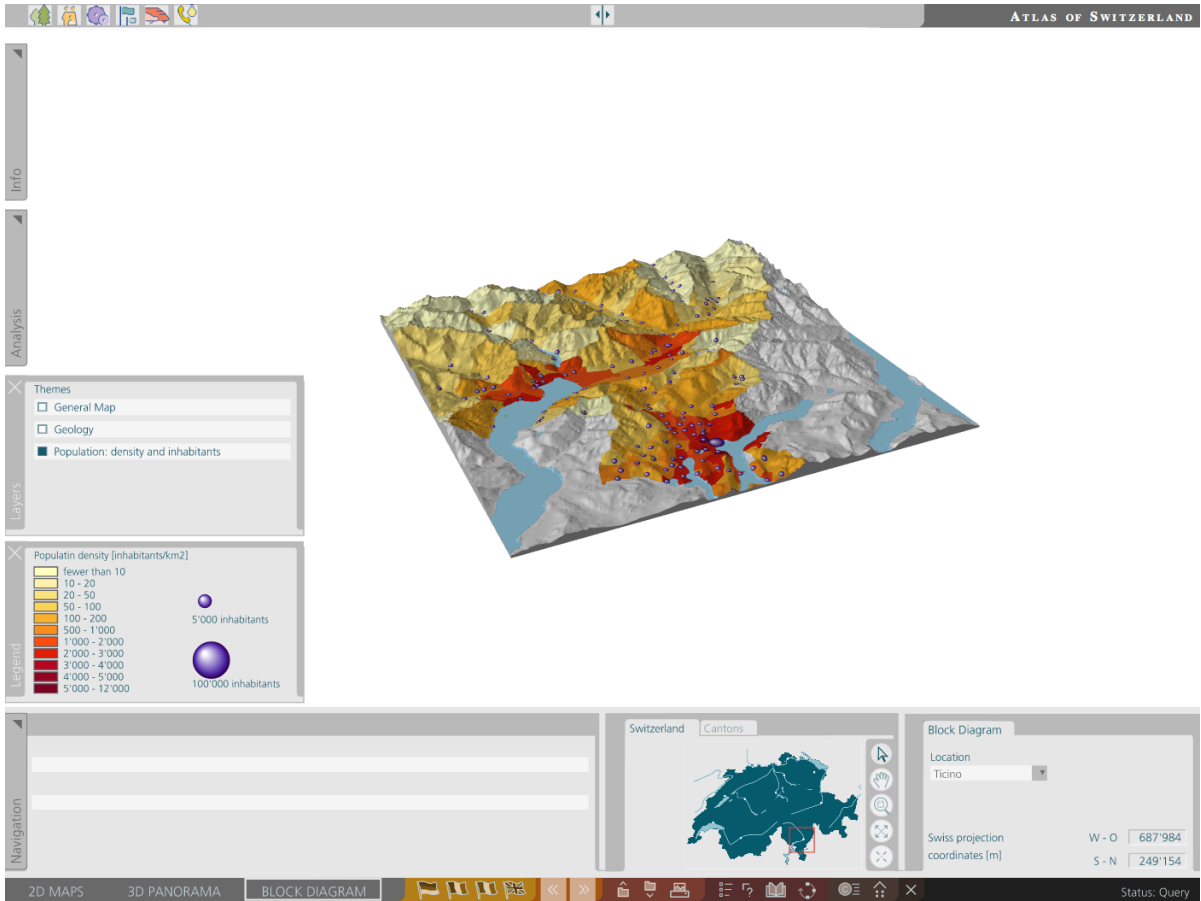
k. Block Diagram: General Map, Ticino



l. Block Diagram: Geology, Ticino



m. Block Diagram: Population Density and Inhabitants, Ticino



II. Panorama View Integration (getGlobeCapture.js)

```
//Default parameters for the request
var sl_distance = 30000;
var yaw = 0;
var pitch = 30;
var roll = 0;
var poiX = 680000;
var poiY = 200000;
var slider_distance;
var slider_pitch;
var daytime = "\"12:00\"";
var port = 8082;
var poiX_text = "680'000";
var poiY_text = "200'000";

// Function called from the GUI through the tab
function getGlobeCapture (){
    //Remove other views and other useless things
    document.getElementById("map0").setAttributeNS(null,"display","none");
    document.getElementById("map1").setAttributeNS(null,"display","none");
    document.getElementById("position").setAttributeNS(null,"display","none");
    document.getElementById("attributeDisplay").setAttributeNS(null,"display","none");
    document.getElementById("kartentitel").setAttributeNS(null,"display","none");
    document.getElementById("tabgroupThema").setAttributeNS(null,"display","none");
    document.getElementById("mapZoomSlider").setAttributeNS(null,"display","none");
    document.getElementById("zoomIn").setAttributeNS(null,"display","none");
    document.getElementById("zoomOut").setAttributeNS(null,"display","none");
    document.getElementById("buttonpin").setAttributeNS(null,"display","none");
    document.getElementById("position3").setAttributeNS(null,"display","none");
    document.getElementById("kartentitel3").setAttributeNS(null,"display","none");
    myMapApp.tabgroups["navigation"].disableSingleTab(1);
    myMapApp.buttons["zoomManual"].deactivate();
    myMapApp.buttons["panManual"].deactivate();
    myMapApp.buttons["recenterMap"].deactivate();
    myMapApp.buttons["zoomFull"].deactivate();
    document.getElementById("frameGL").setAttributeNS(null,"display","none");
    document.getElementById("locFrame").setAttributeNS(null,"display","none");
    document.getElementById("checkBoxes").setAttributeNS(null,"display","none");
    document.getElementById("textMap2D").setAttributeNS(null,"display","none");
    document.getElementById("WEE_Map2D").setAttributeNS(null,"display","none");
    document.getElementById("WEE_BlockDiagram").setAttributeNS(null,"display","none");
    document.getElementById("WEL_BlockDiagram").setAttributeNS(null,"display","none");
    myMapApp.Windows["Ebenen"].close(false);
    myMapApp.Windows["Legende"].close(false);
    deleteTheme();

    // Set the panorama view and others to visible
    document.getElementById("position2").setAttributeNS(null,"display","inherit");
    document.getElementById("kartentitel2").setAttributeNS(null,"display","inherit");
    document.getElementById("slider_distance").setAttributeNS(null,"display","inherit");
    document.getElementById("buttons_yaw").setAttributeNS(null,"display","inherit");
    document.getElementById("slider_pitch").setAttributeNS(null,"display","inherit");
    document.getElementById("frame3D").setAttributeNS(null,"display","inherit");
    document.getElementById("loc_Poi").setAttributeNS(null,"display","inherit");
    document.getElementById("WEE_Panorama").setAttributeNS(null,"display","inherit");
    document.getElementById("textPanorama").setAttributeNS(null,"display","inherit");
    document.getElementById("checkBoxesPanorama").setAttributeNS(null,"display","inherit");
    myMapApp.Windows["Ebenen"].appendContent("WEE_Panorama",true);
    myMapApp.Windows["Legende"].appendContent("WEL_Panorama",true);

    // Get the default image
    getImageGC();
}
```

```

// Set the distance value
function setDistanceTo(changetype,id,value){
    if(id == "slider_distance"){
        sl_distance = value.toFixed(0);
        getImageGC();
    }
}

// Set the pitch value
function setPitchTo(changetype,id,value){
    if (id == "slider_pitch"){
        pitch = value.toFixed(0);
        getImageGC();
    }
}

// Set the max and min distance/pitch to the point of interest
function setMaxMin (id){
    if (id == "distanceMax"){
        slider_distance.setValue(200000,true);
    }
    else if (id == "distanceMin"){
        slider_distance.setValue(5000,true);
    }
    else if (id == "pitchMax"){
        slider_pitch.setValue(90,true);
    }
    else if (id == "pitchMin"){
        slider_pitch.setValue(0,true);
    }
}

// Set the yaw
function setYawTo (groupId,evt,buttonState,buttonText){
    if (groupId == "N" & buttonState == true){
        yaw = 0;
        myMapApp.buttons["S"].setSwitchValue(false,false);
        myMapApp.buttons["E"].setSwitchValue(false,false);
        myMapApp.buttons["W"].setSwitchValue(false,false);
        myMapApp.buttons["NE"].setSwitchValue(false,false);
        myMapApp.buttons["NW"].setSwitchValue(false,false);
        myMapApp.buttons["SE"].setSwitchValue(false,false);
        myMapApp.buttons["SW"].setSwitchValue(false,false);
    }
    else if (groupId == "S" & buttonState == true){
        yaw = 180;
        myMapApp.buttons["N"].setSwitchValue(false,false);
        myMapApp.buttons["E"].setSwitchValue(false,false);
        myMapApp.buttons["W"].setSwitchValue(false,false);
        myMapApp.buttons["NE"].setSwitchValue(false,false);
        myMapApp.buttons["NW"].setSwitchValue(false,false);
        myMapApp.buttons["SE"].setSwitchValue(false,false);
        myMapApp.buttons["SW"].setSwitchValue(false,false);
    }
    else if (groupId == "E" & buttonState == true){
        yaw = 270;
        myMapApp.buttons["N"].setSwitchValue(false,false);
        myMapApp.buttons["S"].setSwitchValue(false,false);
        myMapApp.buttons["W"].setSwitchValue(false,false);
        myMapApp.buttons["NE"].setSwitchValue(false,false);
        myMapApp.buttons["NW"].setSwitchValue(false,false);
        myMapApp.buttons["SE"].setSwitchValue(false,false);
        myMapApp.buttons["SW"].setSwitchValue(false,false);
    }
    else if (groupId == "W" & buttonState == true){

```

```

yaw = 90;
myMapApp.buttons["N"].setSwitchValue(false,false);
myMapApp.buttons["S"].setSwitchValue(false,false);
myMapApp.buttons["E"].setSwitchValue(false,false);
myMapApp.buttons["NE"].setSwitchValue(false,false);
myMapApp.buttons["NW"].setSwitchValue(false,false);
myMapApp.buttons["SE"].setSwitchValue(false,false);
myMapApp.buttons["SW"].setSwitchValue(false,false);
}
else if (groupId == "NE" & buttonState == true){
  yaw = 315;
  myMapApp.buttons["N"].setSwitchValue(false,false);
  myMapApp.buttons["S"].setSwitchValue(false,false);
  myMapApp.buttons["E"].setSwitchValue(false,false);
  myMapApp.buttons["W"].setSwitchValue(false,false);
  myMapApp.buttons["NW"].setSwitchValue(false,false);
  myMapApp.buttons["SE"].setSwitchValue(false,false);
  myMapApp.buttons["SW"].setSwitchValue(false,false);
}
else if (groupId == "NW" & buttonState == true){
  yaw = 45;
  myMapApp.buttons["N"].setSwitchValue(false,false);
  myMapApp.buttons["S"].setSwitchValue(false,false);
  myMapApp.buttons["E"].setSwitchValue(false,false);
  myMapApp.buttons["W"].setSwitchValue(false,false);
  myMapApp.buttons["NE"].setSwitchValue(false,false);
  myMapApp.buttons["SE"].setSwitchValue(false,false);
  myMapApp.buttons["SW"].setSwitchValue(false,false);
}
else if (groupId == "SE" & buttonState == true){
  yaw = 225;
  myMapApp.buttons["N"].setSwitchValue(false,false);
  myMapApp.buttons["S"].setSwitchValue(false,false);
  myMapApp.buttons["E"].setSwitchValue(false,false);
  myMapApp.buttons["W"].setSwitchValue(false,false);
  myMapApp.buttons["NW"].setSwitchValue(false,false);
  myMapApp.buttons["NE"].setSwitchValue(false,false);
  myMapApp.buttons["SW"].setSwitchValue(false,false);
}
else if (groupId == "SW" & buttonState == true){
  yaw = 135;
  myMapApp.buttons["N"].setSwitchValue(false,false);
  myMapApp.buttons["S"].setSwitchValue(false,false);
  myMapApp.buttons["E"].setSwitchValue(false,false);
  myMapApp.buttons["W"].setSwitchValue(false,false);
  myMapApp.buttons["NW"].setSwitchValue(false,false);
  myMapApp.buttons["NE"].setSwitchValue(false,false);
  myMapApp.buttons["SE"].setSwitchValue(false,false);
}
getImageGC();
}

// Set the day time
function setDaytime(id,selectedId,labelText){
  if (selectedId == "day"){
    daytime = "\"12:00\"";
  }
  if (selectedId == "night"){
    daytime = "\"24:00\"";
  }
  getImageGC();
}

// Get the coordinates of the Point of Interest
function setPointOfInterest(evt){

```



```

var Index = myMapApp.tabgroups["view"].activeTabIndex;
if (Index == 1 || Index == 2){
    myMapApp.buttons["zoomManual"].deactivate();
}
if(evt.type == "click" && Index == 1 ){
    poiX_text = document.getElementById("coordX").firstChild.nodeValue;
    poiY_text = document.getElementById("coordY").firstChild.nodeValue;
    var x_array = poiX_text.split("");
    var y_array = poiY_text.split("");
    poiX = parseFloat(x_array[0] + x_array[1]) ;
    poiY = parseFloat(y_array[0] + y_array[1]) ;
    document.getElementById("circlePol").setAttributeNS(null,"cx",poiX);
    document.getElementById("circlePol").setAttributeNS(null,"cy",-poiY);
    getImageGC();
}
}

// Set the theme of the texture
function setTheme(id,selectedId,labelText){
    if(selectedId == "base"){
        port = 8082;
        document.getElementById("pan_legend_base").setAttributeNS(null,"display","inherit");
        document.getElementById("pan_legend_geology").setAttributeNS(null,"display","none");
        document.getElementById("pan_legend_pop_density").setAttributeNS(null,"display","none");
        document.getElementById("pan_legend_boat").setAttributeNS(null,"display","none");
    }
    else if(selectedId == "geology"){
        port = 8083;
        document.getElementById("pan_legend_geology").setAttributeNS(null,"display","inherit");
        document.getElementById("pan_legend_base").setAttributeNS(null,"display","none");
        document.getElementById("pan_legend_pop_density").setAttributeNS(null,"display","none");
        document.getElementById("pan_legend_boat").setAttributeNS(null,"display","none");
    }
    else if(selectedId == "pop_density"){
        port = 8084;
        document.getElementById("pan_legend_pop_density").setAttributeNS(null,"display","inherit");
        document.getElementById("pan_legend_base").setAttributeNS(null,"display","none");
        document.getElementById("pan_legend_geology").setAttributeNS(null,"display","none");
        document.getElementById("pan_legend_boat").setAttributeNS(null,"display","none");
    }
    else if(selectedId == "boat"){
        port = 8086;
        document.getElementById("pan_legend_boat").setAttributeNS(null,"display","inherit");
        document.getElementById("pan_legend_base").setAttributeNS(null,"display","none");
        document.getElementById("pan_legend_pop_density").setAttributeNS(null,"display","none");
        document.getElementById("pan_legend_geology").setAttributeNS(null,"display","none");
    }
    getImageGC();
}

// Send the request to the Globe Capture server
function getImageGC(){

    //Set the display values
    document.getElementById("value_distance").firstChild.nodeValue = sl_distance + " [m]";
    document.getElementById("value_pitch").firstChild.nodeValue = pitch + " [°]";
    document.getElementById("PolX").firstChild.nodeValue = "X: " + poiX_text;
    document.getElementById("PolY").firstChild.nodeValue = "Y: " + poiY_text;

    // Remove Previous GlobeCapture Image
    var myRasterLayer = document.getElementById("frame3DGroup");
    if (myRasterLayer.childNodes.length > 0) {
        var oldImage = document.getElementById("frame3DGroupImage");
        myRasterLayer.removeChild(oldImage);
    }
}

```

```

// Create Element to Hold the Image and set Attributes
var myRasterImage = document.createElementNS(svgNS, "image");
myRasterImage.setAttributeNS(null, "x", 0);
myRasterImage.setAttributeNS(null, "y", 0);
myRasterImage.setAttributeNS(null, "width", 1024);
myRasterImage.setAttributeNS(null, "height", 728);
myRasterImage.setAttributeNS(null, "id", "frame3DGroupImage");
var x = myRasterImage.getAttributeNS(null, "x");
var y = myRasterImage.getAttributeNS(null, "y");
var width = myRasterImage.getAttributeNS(null, "width");
var height = myRasterImage.getAttributeNS(null, "height");

// Create the URL of the ClobeCapture Image
var myImageUrl = "http://129.132.127.194:" + port + "/generate_image?json=[{"cmd\":\"set_image_spec\", \"image\":
{ \"width\":",
myImageUrl += width + ", \"height\":", height + ", \"type\":\"png\"}},",
myImageUrl += "{"cmd\":\"set_camera\", \"camera\":{\"lookat\":[" + poiX + ", " + poiY + "], \"ypr\":[" + yaw + ", " + pitch + ", " +
roll + "], \"distance\":", sl_distance + ", \"max_time_to_wait_in_s\":1}, \"date_time\":\" + daytime + "]}]";
myImageUrl += "&format=\"png\"";

// Append the Image
myRasterImage.setAttributeNS(xlinkNS, "xlink:href", myImageUrl);
myRasterLayer.appendChild(myRasterImage);
}

```

III. Block Diagram Integration (getBlockDiagram.js)

//Function called from the GUI through the tab

```
function getBlockDiagram () {

    // Remove other views and other useless things
    document.getElementById("map0").setAttributeNS(null,"display","none");
    document.getElementById("map1").setAttributeNS(null,"display","none");
    document.getElementById("frame3D").setAttributeNS(null,"display","none");
    document.getElementById("position2").setAttributeNS(null,"display","none");
    document.getElementById("kartentitel2").setAttributeNS(null,"display","none");
    document.getElementById("slider_distance").setAttributeNS(null,"display","none");
    document.getElementById("buttons_yaw").setAttributeNS(null,"display","none");
    document.getElementById("slider_pitch").setAttributeNS(null,"display","none");
    document.getElementById("position").setAttributeNS(null,"display","none");
    document.getElementById("attributeDisplay").setAttributeNS(null,"display","none");
    document.getElementById("kartentitel").setAttributeNS(null,"display","none");
    document.getElementById("tabgroupThema").setAttributeNS(null,"display","none");
    document.getElementById("mapZoomSlider").setAttributeNS(null,"display","none");
    document.getElementById("zoomIn").setAttributeNS(null,"display","none");
    document.getElementById("zoomOut").setAttributeNS(null,"display","none");
    document.getElementById("buttonpin").setAttributeNS(null,"display","none");
    document.getElementById("loc_Pol").setAttributeNS(null,"display","none");
    document.getElementById("WEE_Panorama").setAttributeNS(null,"display","none");
    myMapApp.tabgroups["navigation"].disableSingleTab(1);
    myMapApp.buttons["zoomManual"].deactivate();
    myMapApp.buttons["panManual"].deactivate();
    myMapApp.buttons["recenterMap"].deactivate();
    myMapApp.buttons["zoomFull"].deactivate();
    document.getElementById("checkboxes").setAttributeNS(null,"display","none");
    document.getElementById("textMap2D").setAttributeNS(null,"display","none");
    document.getElementById("WEE_Map2D").setAttributeNS(null,"display","none");
    document.getElementById("WEE_Panorama").setAttributeNS(null,"display","none");
    document.getElementById("WEL_Panorama").setAttributeNS(null,"display","none");
    myMapApp.Windows["Ebenen"].close(false);
    myMapApp.Windows["Legende"].close(false);
    deleteTheme();

    // Set the map and others to visible
    document.getElementById("position3").setAttributeNS(null,"display","inherit");
    document.getElementById("kartentitel3").setAttributeNS(null,"display","inherit");
    document.getElementById("frameGL").setAttributeNS(null,"display","inherit");
    document.getElementById("locFrame").setAttributeNS(null,"display","inherit");
    myMapApp.Windows["Ebenen"].appendContent("WEE_BlockDiagram",true);
    myMapApp.Windows["Legende"].appendContent("WEL_BlockDiagram",true);
    document.getElementById("WEE_BlockDiagram").setAttributeNS(null,"display","inherit");
    document.getElementById("WEL_BlockDiagram").setAttributeNS(null,"display","inherit");

    // Windows to keep the Block Diagram in the middle **** Does not work!*****
    var placeholdersStylesWF = {"fill":"none","stroke":"none"};
    var windowStylesWF = {"fill":"none","stroke":"red"};
    var titlebarStylesWF = {"fill":"none","stroke":"none"};
    var statusBarStylesWF = {"fill":"none"};
    var titletextStylesWF = {"font-family":"Frutiger LT","font-size":12,"fill":"#1D5A69"};
    var statustextStylesWF = {"font-family":"Frutiger LT","font-size":9,"fill":"#1D5A69"};
    var buttonStylesWF = {"fill":"none","stroke":"none"};
    var titlebarHeightWF = 0;
    var statusBarHeightWF = 0;

    // Get the diagram in file webGL.js
    getDiagram();
}

// Set the theme of the block diagram
function setThemeBD(id,selectedId,labelText){
```

```

if(selectedId == "base"){
    texture = "basemap.php";
    document.getElementById("bd_legend_base").setAttributeNS(null,"display","inherit");
    document.getElementById("bd_legend_geology").setAttributeNS(null,"display","none");
    document.getElementById("bd_legend_pop_density").setAttributeNS(null,"display","none");
}
if(selectedId == "geology"){
    texture = "geology.php";
    document.getElementById("bd_legend_geology").setAttributeNS(null,"display","inherit");
    document.getElementById("bd_legend_base").setAttributeNS(null,"display","none");
    document.getElementById("bd_legend_pop_density").setAttributeNS(null,"display","none");
}
if(selectedId == "pop_density"){
    texture = "pop_density.php";
    document.getElementById("bd_legend_pop_density").setAttributeNS(null,"display","inherit");
    document.getElementById("bd_legend_geology").setAttributeNS(null,"display","none");
    document.getElementById("bd_legend_base").setAttributeNS(null,"display","none");
}
}
getDiagram();
}

// Set the right DEM for the block diagram
function getDHM(groupId,indexNr,value){
    if (value == "Matterhorn"){
        loc = "matt_";
        dhm = "matt_dhm.txt";
        document.getElementById("loc_matt").setAttributeNS(null,"display","inherit");
        document.getElementById("loc_lav").setAttributeNS(null,"display","none");
        document.getElementById("loc_val").setAttributeNS(null,"display","none");
        document.getElementById("loc_tic").setAttributeNS(null,"display","none");
        document.getElementById("loc_stg").setAttributeNS(null,"display","none");
    }
    else if (value == "En Lavaux"){
        loc = "lav_";
        dhm = "lav_dhm.txt";
        document.getElementById("loc_lav").setAttributeNS(null,"display","inherit");
        document.getElementById("loc_matt").setAttributeNS(null,"display","none");
        document.getElementById("loc_val").setAttributeNS(null,"display","none");
        document.getElementById("loc_tic").setAttributeNS(null,"display","none");
        document.getElementById("loc_stg").setAttributeNS(null,"display","none");
    }
    else if (value == "Sud du Valais"){
        loc = "val_";
        dhm = "val_dhm.txt";
        document.getElementById("loc_val").setAttributeNS(null,"display","inherit");
        document.getElementById("loc_matt").setAttributeNS(null,"display","none");
        document.getElementById("loc_lav").setAttributeNS(null,"display","none");
        document.getElementById("loc_tic").setAttributeNS(null,"display","none");
        document.getElementById("loc_stg").setAttributeNS(null,"display","none");
    }
    else if (value == "Ticino"){
        loc = "tic_";
        dhm = "tic_dhm.txt";
        document.getElementById("loc_tic").setAttributeNS(null,"display","inherit");
        document.getElementById("loc_matt").setAttributeNS(null,"display","none");
        document.getElementById("loc_val").setAttributeNS(null,"display","none");
        document.getElementById("loc_lav").setAttributeNS(null,"display","none");
        document.getElementById("loc_stg").setAttributeNS(null,"display","none");
    }
    else if (value == "St Gallen"){
        loc = "stg_";
        dhm = "stg_dhm.txt";
        document.getElementById("loc_stg").setAttributeNS(null,"display","inherit");
        document.getElementById("loc_matt").setAttributeNS(null,"display","none");
    }
}

```

```
document.getElementById("loc_val").setAttributeNS(null,"display","none");
document.getElementById("loc_tic").setAttributeNS(null,"display","none");
document.getElementById("loc_lav").setAttributeNS(null,"display","none");
}
getDiagram();
}
```

IV. Extract of the WebGL Integration (webGL.js)

The following functions are the main functions for the block diagram generation.

setUpCanvas() generate a basic canvas to draw the block diagram in later:

```
function setUpCanvas(){
    gl.clearColor(1.0, 1.0, 1.0, 1.0);
    gl.clear(gl.COLOR_BUFFER_BIT);
    gl.enable(gl.CULL_FACE);
    gl.cullFace(gl.BACK);
}
```

addShader(id) allows to compile and attach the shaders, both for vertices and fragments, to the block diagram:

```
function addShader(id){
    var shaderScript = document.getElementById(id);
    var shaderType = getShaderType(shaderScript);
    var source = getShaderSource(shaderScript);
    var shader = gl.createShader(shaderType);
    gl.shaderSource(shader, source);
    gl.compileShader(shader);
    if (!gl.getShaderParameter(shader, gl.COMPILE_STATUS)){
        throw "Could not compile shader " + id + "\n" + gl.getShaderInfoLog(shader);
    }
    gl.attachShader(prog, shader);
}
```

loadData() assigns the texture from the WMS, the DEM and the texture for the sides of the block diagram into variables used to draw the scene, and calls the loadTexture(srs) function:

```
function loadData(){
    mapTexture = loadTexture("php/" + loc + texture);
    baseTexture = loadTexture("data/socket.png");
    grid = new Grid();
    grid.runWhenLoaded(start);
    grid.loadData("data" + dhm);
}
```

loadTexture(src) creates the texture from the WMS, the DEM and the texture for the sides of the block diagram from the data repository and then calls the drawScene() function:

```
function loadTexture(src){
    var texture = gl.createTexture();
    var image = new Image();
    image.onload = function(){
        gl.bindTexture(gl.TEXTURE_2D, texture);
        gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER, gl.LINEAR);
        gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MAG_FILTER, gl.LINEAR);
        gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_WRAP_S, gl.CLAMP_TO_EDGE);
        gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_WRAP_T, gl.CLAMP_TO_EDGE);
        gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, gl.RGBA, gl.UNSIGNED_BYTE, image);
        WebGLTexture.prototype.loaded = true;
        drawScene();
    };
    image.onerror = function(){
        alert("Error while loading image " + src + ".");
    }
    image.src = src;
    return texture;
}
```

drawScene() actually draws the block diagram within the canvas:

```

function drawScene(){
    gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
    gl.uniform4f(gl.getUniformLocation(prog, "color"), 0.5, 0.5, 0.5, 1.0);
    gl.enable(gl.TEXTURE_2D);
    var modelMatrix = mat4.lookAt([0,0,2.4], [0,0,0], [0,1,0]);
    var ratio = canvas.height / canvas.width;
    mat4.scale(modelMatrix, [objectScale * Math.max(ratio, 1.0), objectScale * Math.max(1.0 / ratio, 1.0),
1.0]);

    mat4.rotate(modelMatrix, deg2Rad(xRot), [1, 0, 0]);
    mat4.rotate(modelMatrix, deg2Rad(zRot), [0, 0, 1]);
    mat4.translate(modelMatrix, [-0.5, -0.5, -zOffset]);
    var mMatLoc = gl.getUniformLocation(prog, "modelMatrix");
    gl.uniformMatrix4fv(mMatLoc, false, modelMatrix);
    var nr = grid.getNumRows();
    var nc = grid.getNumCols();
    var tul = gl.getUniformLocation(prog, "textureSampler");

    if (mapTexture.loaded){
        gl.uniform1i(tul, 0);
        gl.activeTexture(gl.TEXTURE0);
        gl.bindTexture(gl.TEXTURE_2D, mapTexture);
        for (var i = 0; i < nr - 1; ++i){
            gl.drawElements(gl.TRIANGLE_STRIP, 2 * nc, gl.UNSIGNED_SHORT, 4 * nc * i);
        }
    }

    if (baseTexture.loaded){
        gl.uniform1i(tul, 1);
        gl.activeTexture(gl.TEXTURE1);
        gl.bindTexture(gl.TEXTURE_2D, baseTexture);
        var offset = 4 * nc * (nr - 1);
        gl.drawElements(gl.TRIANGLE_STRIP, nc * 2, gl.UNSIGNED_SHORT, offset);
        gl.drawElements(gl.TRIANGLE_STRIP, nr * 2, gl.UNSIGNED_SHORT, offset += 4 * nc);
        gl.drawElements(gl.TRIANGLE_STRIP, nc * 2, gl.UNSIGNED_SHORT, offset += 4 * nr);
        gl.drawElements(gl.TRIANGLE_STRIP, nr * 2, gl.UNSIGNED_SHORT, offset += 4 * nc);
    }

    gl.flush();
}

```

V. Integration of WMS in the Globe Capture Request

```
function get_default_cmd(cmd){
  switch(cmd) {
    case "load_map":
      return {
        "cmd": 'load_map',
        "options": {
          "lighting": "enabled"
        },
        "layers": [
          {
            "type": "image",
            "name": "relief",
            "profile": {
              "srs": "EPSG:4326",
              "extents": []
            },
            "url": "q:/relief_500_raster/tms.xml",
            "format": "tms"
          },
          {
            "type": "elevation",
            "name": "DHM Ferranti",
            "profile": {
              "srs": "EPSG:4326",
              "extents": []
            },
            "url": "q:/dhm_ferranti_wgs84/tms.xml",
            "format": "tms"
          },
          {
            "type": "image",
            "name": "wms1",
            "profile": {
              "srs": "EPSG:4326",
              "extents": []
            },
            "url": "http://karlinapp.ethz.ch/cgi-bin/qgis_map_server/adsweb2/
qgis_mapserv.fcgi",
            "layer": "Geology,Border_band,Border_line",
            "image_format": "png",
            "style": "Geology,Border_band,Border_line",
            "srs": "EPSG:4326",
            "transparent": "true",
            "format": "wms",
            "opacity": "0.5"
          }
        ]
      };
      break;
    case "set_camera":
      return {
        "cmd": 'set_camera',
        "camera": {
          "lookat": [667970.37,203403.09,0],
          "ypr": [0,30,0],
          "distance": 50000,
          "max_time_to_wait_in_s": 3
        }
      };
      break;
    case "set_image_spec":
      return {
        "cmd": "set_image_spec",
        "image": {
```



```
        "width": 640,  
        "height": 480,  
        "type": "png"  
    }  
};  
break;  
};  
}
```

VI. Examples of WMS Request for the Block Diagram

This example is the request for the block diagram of Ticino with the general map.

```
<?php
header('Content-Type: image/png');
readfile('http://karlinapp.ethz.ch/cgi-bin/qgis_map_server/adsweb/qgis_mapserv.fcgi?
SERVICE=WMS&version=1.3.0&REQUEST=GetMap&LAYERS=Background,Forest,Agglomeration,LakesBD,RiversBD,
Border_band,Border_line,RoadsBD,RoadsBD,Airports,Airports_labels&STYLES=Background,Forest,Agglomeration,Lak
esBD,RiversBD,Border_band,Border_line,RoadsBD_stroke,RoadsBD_fill,Airports,Airports_labels&FORMAT=image/
png&CRS=EPSG:
21781&WIDTH=1200&HEIGHT=1200&BBOX=693400,87650,743400,137650&TRANSPARENT=true');
?>
```

This example is the request for the block diagram of Valais with the geology map.

```
<?php
header('Content-Type: image/png');
readfile('http://karlinapp.ethz.ch/cgi-bin/qgis_map_server/adsweb/qgis_mapserv.fcgi?
SERVICE=WMS&version=1.3.0&REQUEST=GetMap&LAYERS=Geology,Border_band,Border_line&STYLES=Geology,
Border_band,Border_line&FORMAT=image/png&CRS=EPSG:
21781&WIDTH=1200&HEIGHT=1200&BBOX=572100,83200,622100,133200&TRANSPARENT=true');
?>
```

VII. Vertex and Fragment Shaders Integration

```

<script id="vertex" type="x-shader/x-vertex">
    uniform mat4 perspectiveMatrix;
    uniform mat4 modelMatrix;
    attribute vec3 pos;
    attribute vec3 norm;
    varying vec2 textureCoordinates;
    varying float intensity;
    const vec3 lightDirection = vec3(1.0, 1.0, -1.0);
    const float ambient = 0.3;
    const float diffuse = 0.5;
    const float specular = 0.2;
    const float roughness = 20.0;

    void main()
    {
        textureCoordinates = pos.xy;
        gl_Position = perspectiveMatrix * modelMatrix * vec4(pos, 1.0);
        vec3 normalVector = normalize(modelMatrix * vec4(norm, 0.0)).xyz;
        vec3 lightVector = normalize(lightDirection);
        vec3 specularVector = normalize(vec3(0.0, 0.0, -1.0) + lightVector);
        float diffuseIntensity = diffuse * max(0.0, dot(normalVector, lightVector));
        float specularIntensity = specular * pow(max(0.0, dot(normalVector, specularVector)),
roughness);

        intensity = ambient + diffuseIntensity + specularIntensity;
    }
</script>
<script id="fragment" type="x-shader/x-fragment">
#ifdef GL_ES
    precision highp float;
#endif

    uniform sampler2D textureSampler;
    varying vec2 textureCoordinates;
    uniform vec4 color;
    varying float intensity;
    const float gamma = 1.0;

    void main(void)
    {
        vec4 color = texture2D(textureSampler, textureCoordinates);

        // This command draws the map without the shadows
        // gl_FragColor = color;

        // This command draws only the shadows
        // gl_FragColor = vec4(vec3(intensity), 1.0);

        // This command draws the map with the shadows
        gl_FragColor = vec4(intensity * pow(color.rgb, vec3(gamma)), 1.0);
    }
</script>

```

VIII. Functions of Existing 3D Digital Atlases

		AdS	SWAi	GPF 3D Viewer	GPB 3D Viewer
General Functions		<ul style="list-style-type: none"> – language choice – next/previous – tooltips – help – global options – starting page – save as image – print maps – load/save maps – exit – fullscreen view – minimize/maximize window 	<ul style="list-style-type: none"> – language choice – print – preferences – tooltips – help – impressum – start page – exit – system information – import/export/save maps – double windows – next/previous – maps history 	<ul style="list-style-type: none"> – hotspots – organization of layout – tooltips – bookmarks – sent to a friend – help – impressum – full screen view – save maps 	n.a.
Navigation Functions	<i>Spatial</i>	<ul style="list-style-type: none"> – zoom in/out – pan – reference map – settings of location – viewing direction – aperture angle – viewing distance – 	<ul style="list-style-type: none"> – zoom in/out – pan – regional organization – orientation – coordinates display – search 	<ul style="list-style-type: none"> – zoom in/out – pan – orientation – relief view – reference map – altitude – search location – fly over – navigation with the mouse 	<ul style="list-style-type: none"> – viewpoint – distance – pitch
	<i>Thematic</i>	<ul style="list-style-type: none"> – search – choice of displayed layers – choice of theme 	<ul style="list-style-type: none"> – choice of displayed layers 	<ul style="list-style-type: none"> – choice of displayed layers – organized within themes 	n.a.
	<i>Time</i>	<ul style="list-style-type: none"> – choice of period 	<ul style="list-style-type: none"> – year selection for some maps 	n.a.	n.a.
Didactic Functions	<i>Explanation</i>	<ul style="list-style-type: none"> – information regarding thematic data only for block diagram in 3D 	n.a.	n.a. There is a web platform specifically aimed at schoolers and teachers: Edugéo. It includes 2D and 3D maps and works as an exchange platform for geographic teacher. Access restricted.	n.a.
	<i>Self Check</i>	n.a.	n.a.		n.a.
Cartographic and Visualization Functions	<i>Map Manipulation</i>	<ul style="list-style-type: none"> – layers on/off – legend on/off – transparency – modification of the legend color schemes – changes of labels settings 	<ul style="list-style-type: none"> – layers on/off – legend on/off 	<ul style="list-style-type: none"> – transparency of layers – layers on/off – legend on/off 	<ul style="list-style-type: none"> – exaggeration of the relief
	<i>Redlining</i>	n.a.	n.a.	n.a.	n.a.
	<i>Explorative Data Analysis</i>	<ul style="list-style-type: none"> – modification of classes – sky settings – map comparison 	<ul style="list-style-type: none"> – map comparison (location synchronization) 	n.a.	n.a.
GIS Functions	<i>Spatial Query</i>	<ul style="list-style-type: none"> – profile – coordinates display – 	n.a.	<ul style="list-style-type: none"> – altitude – distance and surface measurement 	n.a.
	<i>Thematic Query</i>	<ul style="list-style-type: none"> – coordinates – 	n.a.	n.a.	n.a.
	<i>Analysis</i>	<ul style="list-style-type: none"> – aspect – slope 	n.a.	n.a.	n.a.

IX. Requirements Overview

System requirements		Keywords
Service-oriented architecture	A service oriented system is required to allow thin-client access to the visualization of massive geodata within the atlas, no matter what are the processing softwares or capacities of the client.	Service-driven Thin client
No plugin needed	The absence of plugin aims at simplifying the use of the atlas and to avoid any compatibility issues between plugin and browser or platform.	No plugin Browser visualization Ease of use
Cross-platform	The system has to be able to works fully and independently from the software and hardware platforms.	Scope of distribution Platform independency
Interoperability and integration	Integration and interoperability (increases the integration of external data sources, especially through applying standards) are required to connect computer systems effectively and efficiently on different level of abstraction.	Interoperability Use of standards Integration
Non-functional features	Support for easy updating, scaling up, extensibility, reuse, and robustness for instance. It helps make the product long-lasting and optimize its use and development.	Update & Extensibility Reuse & Robustness
Open source	It allows a direct access to the code and prevent any issue with copyright, while providing the support of the developers' community	Open source
Support for massive amount of geodata and dynamic data	To build a comprehensive atlas requires support for massive amount of geodata, as well as dynamic geodata. Without this availability and support, the usefulness of such atlas is quite small.	Amount of geodata Dynamic geodata
Visualization requirements		Keywords
High level of abstraction	Ability to build geovisualization system at an increasing level of abstraction.	Abstraction
Effective, high quality visual representations	The complexity of the system should not be at the expense of the visualization quality and of its efficiency.	Visualization quality
Support for user styling	This is highly relevant for web atlases because it allows the users to manipulate not only the representation of the landscape, but also of the representation of the thematic data.	Color scheme settings Atmosphere and landscape settings
Multiple views	Multiple and synchronized (or coordinated) views of map is an important feature for simple exploratory visualization because it allows to visually compare geodata.	Synchronized views Collaborative views Exploratory visualization
Interaction requirements		Keywords
High degree of interactivity	Interaction and dynamic display have to be at earth of such atlas conception in order to get the best out of the geodata visualization. It allows the users to build their own knowledge and representations from the geodata.	Interactivity
Intuitive navigation (thematic and spatial)	3D geodata are more complex than 2D geodata and thus an appropriate and intuitive navigation within the data is indispensable.	Navigation Handling of complexity
Query and processing	Requirements for well-developed atlas, allows to get the most out of the spatial data.	Query Data processing



Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of Originality

This sheet must be signed and enclosed with every piece of written work submitted at ETH.

I hereby declare that the written work I have submitted entitled

Service-driven 3D Atlas Cartography

is original work which I alone have authored and which is written in my own words.

Author

Last name

Panchaud

First name

Nadia

Supervisor

Last name

Dr. Prof. Hurni

First name

Lorenz

Supervising tutors

Last name

Dr. Iosifescu-Enescu

Dr. Sieber

First name

Ionut

René

With this signature I declare that I have been informed regarding normal academic citation rules and that I have read and understood the information on "Citation etiquette" (http://www.ethz.ch/students/exams/plagiarism_s_en.pdf). The citation conventions usual to the discipline in question have been respected.

The above written work may be tested electronically for plagiarism.

Place and date

Signature
