FROM STATIC TO DYNAMIC VISUALIZATION OF THE SEA SURFACE HEIGHT ON A WEB GIS APPLICATION

A thesis submitted to the Delft University of Technology in partial fulfillment of the requirements for the degree of

Master of Science in Geomatics for the Built Environment

by

Georgios Dimopoulos

September 2019

ISBN 999–99–9999–999–9

The work in this thesis was made in the:

Faculty of Architecture & the Built Environment
Delft University of Technology

Supervisors: Prof.dr.ir. P.J.M van Oosterom
dr.ir. B.M. (Martijn) Meijers
dr. Fedor Baart (Deltares), Cindy Van de Vries (Deltares)
Co-reader:

ABSTRACT

During the last three decades, the Earth's climate is changing rapidly, with higher average temperatures every year that leads not only to the melting of the ice sheet in the arctic and on most of the glaciers all over the world but also to extreme weather phenomena. The rise of the temperature can affect the Sea Surface Height (SSH) in more than one way, and since 70% of the Earth's surface is covered by the oceans, if the oceans are being affected then the whole Earth also is. The monitoring of the SSH can help the scientist predict the changes that will take place in the future. The SSH is a dynamic phenomenon that constantly changes not only within different decades but also from year to year, month to month even within the same day. These changes are the result of various phenomena and are called anomalies. When the SSH is monitored different phenomena are represented in different time scales and it is important to be taken into consideration if there is the need for a proper understanding of the SSH phenomenon.

Many spatial data vendors are providing a large number of data-sets related to the monitoring of SSH and its anomalies and as a result, there is the need to find the most effective way to extract information from the data. Over the years has been established that one of the most effective ways to extract information from data is through the various visualization techniques and since the data of SSH is mainly spatial the main visualization technique is cartography. The advancements of the technology over the last couple of decades have led to a reality that the "online" application is the norm and consequently the web mapping and web geographical information system (GIS).

The goal of this thesis is to propose an architecture for a web GIS application that will be able to visualize dynamic data while adding elements of interactivity to improve the chances of the Sea Surface Height Anomaly (SSHA). IN order to achieve the goal of this thesis three main research questions need to be answered: What type of animation should be used (in order to visualize the passing of time), what interactivity elements should be added (e.g. zooming/panning in space and time)and what system's architecture is optimal for such application (server-side/client-side etc.).

This document is providing guidelines on how to create such an application and is resulting in the production of a prototype. The first part of this thesis is the review of the main ideas that are introduced in this project and how they were implemented by other researchers. Then, a comparison between the different implementation techniques (for every research question) is taking place to determine the main characteristics of the application. The final part is related to the implementation of the chosen techniques that lead to the development of the prototype application.

The resulted prototype even though it is not perfect, due to technical limitations that were a consequence of implementing some of the most recent concepts in web development, is functional and paves the way for the development of new improved dynamic/interactive web GIS applications *(https://giorgosdimo.github.io/MSc-Thesis/)*.

ACKNOWLEDGEMENTS

I would like to take the opportunity to express my gratitude to the people who played an important role in the development of this thesis.

The first person that I would like to thank is Dr. Fedor Baart from Deltares that he was the one that provided me with the project, helped me to define the research goal and even though he was super busy he had also time to help me with the technical aspects. The second person that was evolved with this thesis was my supervisor Prof. dr. ir. Peter van Oosterom that provided his expertise in order not only to define the topic but also due to his experience in the field of Geomatics gave me some new insights about my project. I would also like to thank my second supervisor dr.ir. B.M. (Martijn) Meijers was always present and ready to help me with anything related to this thesis project.

I spent exactly one year in Deltares and I was able to meet new and interesting people like Cindy van De Vries that was my second supervisor from Deltares and helped a lot with my thesis especially during the first few months that I was struggling with all the new information about HTML and JavaScript.

Finally, I would like to thank my family and friends that supported and tolerated me not only during my thesis project but also during the two years of MSc Geomatics.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

GIS geographical information system ... iii
GISs geographical information systems ... 3
NASA National Aeronautics and Space Administration ... 1
NOAA National Oceanic and Atmospheric Administration ... 2
SSH Sea Surface Height ... iii
SSHA Sea Surface Height Anomaly ... iii
OGC Open Geospatial Consortium ... 3
WMS Web Map Services ... ix
WFS Web Feature Services ... 18
WCS Web Coverage Services ... 18
WPS Web Processes Services ... 18
WMTS Web Map Tiled Services ... 67
WVMS Web Video Map Services ... 67
2D Two dimensions ... 7
3D Three dimensions ... 39
WWW World Wide Web ... 9
HTML Hypertext Markup Language ... 17
HTTP Hypertext Transfer Protocol ... 17
DHTML Dynamic Hypertext Markup Language ... 18
CGI Common Gateway Interface ... 18
XML Extensible Markup Language ... 19
AJAX Asynchronous JavaScript and XML ... 18
SVG Scalar Vector Graphics ... 19
KML Keyhole Markup Language ... 19
GIF Graphics Interchange Format ... 20
TMS Tiled Map Services ... 22
CSS Cascading Style Sheets ... 32
GPU Graphics processing unit ... 56
CPU Central Processing Unit ... 32
RGBa Red, Green, Blue, alpha ... 39
NetCDF Network Common Data Form) ... 43
DOM Document Object Model ... 52
RAM Random Access Memory ... 54
SSD Solid State Disk ... 54

1 INTRODUCTION

The spring of 2019 has been the warmest for Alaska on record [Kaplan, 2019]. At the same time, in mid-June, 45% of the Greenland's ice sheet was melting while, normally, this percentage should have been around 10% [Borunda, 2019].

Figure 1.1: Summer meltwater collects in a lake on the surface of the Greenland ice sheet. (https://www.nationalgeographic.com)

The climate is changing, and a lot of areas will be affected all around the world from Alaska and Greenland to Netherlands and Maldives. The melting of the ice in the Arctic and the sea level rise are just part of the climate change related problems that have major impact on the oceans and consequently on the built environment of many coastal countries. The sea surface height with the sea surface temperature are two phenomena that are being monitored by many environmental scientists, since these phenomena can provide a good indication of the changes of the climate. The use of the latest technology in remote sensing, satellite imagery etc., to gather as much data as possible is an important for the scientific community, in their quest to better understand these phenomena. Even though, extracting useful information from a huge amount of data is not an easy task, visualization of the data is one of the most effective and efficient ways to extract information from them.

1.1 SEA SURFACE HEIGHT

One of the biggest and most important categories of geographic data is the one related to the environment. Gathering environmental data can help to monitor the Earth and can help us predict possible changes in the Earth's physical environment. National Aeronautics and Space Administration (NASA) is one of the largest produc-

ers of environment-related data and on their dedicated web page, they explain why it is important to monitor the Ocean Surface Topography which they divide in Sea Surface Height and Sea Surface Temperature. 70% of the surface of the Earth is covered by the oceans and through the hydrology cycle, the oceans, are the main factor that affects the world climate. According to National Oceanic and Atmospheric Administration (NOAA) the Sea Surface Height is of interest for the scientists because it reveals information about how much heat is stored in the ocean. Furthermore, the rate of global sea-level rise has been accelerating in recent decades [Weeman and Patrick, 2018] and if the rate of ocean rise continues to change at this pace, sea level will rise 65 centimeters by 2100 — enough to cause significant problems for coastal cities.

The SSH is affected in a short timescale by the tidal forces of the Moon and the Sun acting on the Earth. Over longer timescales, SSH is influenced by ocean circulation. Typically, anomalies of the SSH resulting from these forces differ from the mean by less than 1 m on a global scale. Among other sources that can cause anomalies on the SSH, are the temperature, the salinity, the waves, and the loading of atmospheric pressure [Stewart, 2008]. The SSH has many parameters that can influence its measurements and all these parameters can vary not only from place to place (**spatial scale**) but also from time to time (**time scale**) something that can affect the results of an analysis and in the end, the information conveyed about this phenomenon. In the past few years the main focus for the researchers and the public is the overall sea-level rise due to the greenhouse effect (as mentioned above) and this type of change should be monitored yearly or even less frequent to have usable results (large time scale). The reason that should not be monitored in a period of less than a year is the presence of some extreme atmospheric phenomena (e.g. El Niño) that can cause significant changes in the SSH on a seasonal basis. These atmospheric phenomena are interesting for the scientists and they should be monitored on a yearly time scale (medium time scale) but again not on a smaller time scale. Reducing the time scale, even more, there is the tide that changes the SSH on a daily basis (small time scale). As it is evident from the above examples the time scale plays an important role in the SSH phenomenon and different time scales are representing different phenomena and depending on what phenomenon the scientist/analyst is interested in many different time scales can have its own importance.

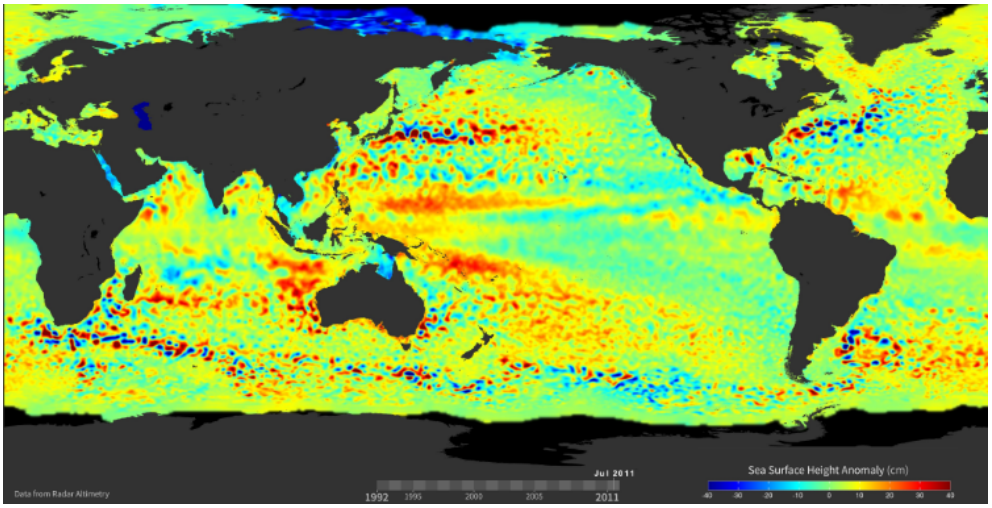


Figure 1.2: A Visualization of the Sea Surface Height. (https://svs.gsfc.nasa.gov)

1.2 VISUALIZATION

Every day the humans are producing a vast amount of data with a rate that is ever increasing. It is a fact that during the last 2 years alone were produced 90% of the total volume of the data of the history of mankind (Bernard, 2018) and this is a trend that shows no signs of slowing down in the future. Humans are producing so much data with the hope that these data will be useful to understand how our world and our society works. Extracting information from all these data is not an easy task. One of the best ways to extract information from data is by visualizing them. Visualization of data/information is not something new, the first map can be dated back to 6200 BC while the first world map was created on 550 BC [Friendly, 2006]. Tables, graphs and maps, static or dynamic, are some of the most common visualization techniques used in order to "see what lies within, determine the answer to a question, find relations, and perhaps apprehend things which could not be seen so readily in other forms" [Friendly, 2006]. All the above are possible through visualization since vision is the most dominant sense in the human sensory system [Hauser et al., 2018].

Cartography, the science/art of creating maps, is one of the oldest visualization techniques [Friendly, 2006] and one that shows great advancements even in the modern world. Actually, a large majority of the data produced daily is geographic data and the need for efficiently and effectively visualizing them is ever-growing. From the mid-20th century with the advancement of computers, the development of Geographic Information Systems (GIS) changed the traditional cartography. According to Clarke [1986] the "Geographic Information Systems (GISs) are computer-assisted systems for the capture, storage, retrieval, analysis and display of spatial data".

The advancement of the computers combined with the digital revolution and subsequent information age, not only helped in the advancement of GIS but also, have prompted changes that are as numerous as they are fundamental to the ways in which maps are produced and consumed, with interactivity being among the most significant of these new possibilities [Roth, 2013]. Numerous scholars have argued that even the paper maps are interactive, but a digital environment affords a wider array of interactions forms and can only be limited by the objectives of the map user and the skills of the developer [Roth, 2013].

When the World Wide Web was invented in the late 1980s, during these past 30years, the number of applications created over the web is in the rise and the geographical information systems (GISs) are not an exception. Since 1993 when the first Web GIS developed it has grown into a rapidly developing discipline [Al-Qurabat, 2015]. A Web GIS application provides the user with several advantages over a traditional GIS since it provides the opportunity to the user to access, analyze and visualize geographic information from anywhere at any time [Al-Qurabat, 2015].

The development of many maps generated by map servers and a range of mapping options for users to view maps online led to the need to combine data from multiple sources on one hand and on the other hand to achieve platform independence came with the need to package and standardize the interface to these web map servers. The purpose is to provide mapping information directly, not just to users, but also to other software programs that can consume and produce information [Veenendaal et al., 2017]. The Open Geospatial Consortium (OGC) is an international industry consortium of over 529 companies, government agencies, and universities participating in a consensus process to develop publicly available interface standards. OGC Standards support interoperable solutions that "geo-enable" the Web, wireless and location-based services and mainstream IT. The standards empower technology developers to make complex spatial information and services accessible and useful with all kinds of applications (http://www.opengeospatial.org).

1.3 MOTIVATION

Even though many researchers have tried in the past to visualize spatial data such as the SSH (Chapter 2, Chapter 3), there is more than enough room for improvement, while adding some extra value to such visualizations. One part that can add value to such a visualization is related with the fact that the SSH is a dynamic phenomenon, which means that this phenomenon changes over time, so, it is preferable to create an animated visualization to represent the time dimension of the phenomenon as accurate as possible. The time dimension can also be visualized by introducing the different time scales in which the SSH can be represented to give to the user the opportunity to observe the different phenomena that can affect it. Another important aspect that should be added in web visualizations is the introduction of interactivity elements that can help the user to better understand the information that the mapmaker wants to deliver. Nowadays more and more GIS applications are available online (over the web) and, as it is described in Section 1.2, the geographic related topics have an important added value when provided over the web. In order to create a visualization with the above characteristics it is important also to consider the architecture of the Web GIS platform that will host it.

The above-mentioned characteristics of the visualization (animated, different time scale, interactive, over the web) of Sea Surface Height even though are not unique as standalone, the combination of them is the challenging part of this thesis project.

1.4 RESEARCH QUESTIONS

The main research question that derives from the motivation part can be described as:*" What is an optimal WebGIS-architecture for making an interactive - dynamic visualization of the sea-surface height phenomenon?"*. In order to answer this question a series of sub - questions should be also answered.

1. *Sea surface height is a dynamic phenomenon (2.5D + time), what technique of animation should be used and why?*

   Historically, animation techniques that have been used are simple slide – shows and GIFs (Graphics Interchange Format) but the past years the videos are the most widespread. The answer to this question is to describe what animation format is the most relevant, why and how to use it for the context of this thesis project.

2. *What elements of interactivity are relevant to a web mapping application and which ones should be implemented?*

   This question should answer what interactivity elements can be considered important for such visualization and why. The most commonly used interactivity elements are zooming and panning but examples of interactivity elements can also be the click on a button that triggers a certain process, turn a layer on or off, change portrayal/color of a map, options that manipulate the animation such as stop, slow, back, forward, etc.

3. *What type of architecture are more appropriate for an implementation with these characteristics?*

   In order to implement the interactivity and the animation of the Web GIS application it is important to have a decision on architecture related topics such as server-side/client-side: communication protocols, formats, buffers/blocks, streaming.

1.5 RESEARCH SCOPE

The focus of this thesis project is to create a Web GIS application for dynamic visualization of the Sea Surface Height phenomenon. This application will use pre-processed data-sets, which are described in Chapter 5, and as a result no extended data analysis will be performed. In the interactivity part only a few interactivity elements will be chosen and will be implemented. These elements will be an outcome of a literature review and no usability test will be performed. Even though the Sea Surface Height is a 2.5D + time phenomenon, in the framework of this thesis project it will be portrayed as a 2D + time with the height value represented with color. Finally, about the architecture of the application, it will be based on existing implementations with additions/changes/transformations in the parts that are needed to achieve the goals of the research questions.

1.6 THESIS OUTLINE

After the conclusion of the introduction (Chapter 1) that is the first chapter of this thesis, follows the Chapter 2 that references the theoretical background on the related topics of this thesis will take place. Chapter 3 describes the related work while Chapter 4 describes the methodology that is a description of the approach used in order to answer the research questions. Chapter 5 is about the implementation of the methodology described in Chapter 4 and the final results of this thesis. The last (Chapter 6) is the one that describes the conclusions of this thesis as well as a reference to some future work is proposed as a result of these conclusions.

2 THEORETICAL BACKGROUND

The introduction chapter (Chapter 1) has two main purposes that help the reader to have a first insight into what this thesis project is about. The first purpose is about the introduction of the concepts and technologies that will be mentioned in the course of this thesis and the second one is to describe the motivation for this thesis and the research questions on which this thesis project will be based. The goal of the present chapter is to focus on the above-mentioned concepts in order to help the reader better understand the parts that will follow in this thesis.

2.1 CARTOGRAPHY IN THE 20TH CENTURY

The first section of the Theoretical Background Chapter is focused on the description of the technologies and novelties that were introduced in cartography during the 20th century. The focus on the 20th century is based on the fact that on one hand, it was the period that some really important ideas were introduced on the field of cartography and on the other hand the concepts that will be used in this thesis project are based on technologies introduced during this period. The section starts with a short description of Cartography as a discipline and a brief reference on the history of cartography in order to demonstrate that even though this section focuses on the 20th centuries this discipline is one of the oldest visualization techniques. After the introductory subsection there will be a subsection for every important (and relevant with this thesis) ideas that were introduced in the 20th century starting from the "colors in maps", the "animated maps" and concluding with the "GIS" subsections.

2.1.1 Overview of Cartography

According to encyclopedia Britannica the definition of cartography is:"Cartography, the art and science of graphically representing a geographical area, usually on a flat surface such as a map or chart. It may involve the superimposition of political, cultural, or other non-geographical divisions onto the representation of a geographical area." The goal of cartography is to model reality in ways that the spatial information should be represented effectively. Every cartographer while designing a map should answer some basic questions in order to produce an effective map. The first question is about the feature (physical like land masses or abstract like political boundaries)of the reality that need to be visualized. The second question is related with the "how" to represent the the above mentioned features on a Two dimensions (2D) surface. The process that represents the surface of the earth on a 2d plain is called map projection. The third question is about the reduction of map's complexity and about the elimination of irrelevant characteristics. These two procedures are called map generalization. The last question is related with the map design that is about the best possible selection of the visualization techniques and elements in order to better convey its message to its audience.

After the definition and the purpose of cartography it is interesting to refer to its history that spans over several millennia. According to the research contacted by [Friendly and Denis, 2001] the first ever map that have been found dates back to 6200 BCE in Konya of present day Turkey. This first map represents the Konya town

with an erupting volcano (Figure 2.1). Whitehouse [2000] dates the first map even further back during the ice age that a cave was found at Lascaux in central France, with a painting of what is believed to be the night sky with stars and planets.

Figure 2.1: Map of Konya from 6200 BCE.

Harrell and Brown [1992] mentions that the Turin Papyrus Map, That dates back to 1160 BCE, is the oldest geological map. It was found in Egypt and it was representing geological resources and providing also information about quarrying of those resources. Between the 6th century BCE and 3rd century BCE some more unique maps were created such as the first world map (Figure 2.2) and the first route map (Figure 2.3) [Friendly, 2006]. The idea of coordinates was used by ancient Egyptian surveyors in laying out towns, earthly and heavenly positions were located by something akin to latitude and longitude at least by 200 BCE, and the map projection of a spherical earth into latitude and longitude by Claudius Ptolemy [c. 85–c. 165] in Alexandria would serve as reference standards until the 14th century [Friendly, 2006].

Figure 2.2: Anaximander world map.(http://ccnmtl.columbia.edu)

According to Friendly [2006] for the next almost two millennia (until the 17th century) cartography was evolving but in a really slow pace nevertheless, some major advancements took place during that period like the introduction of paper, the introduction of triangulation for the determination of mapping locations and the invention of cylindrical projection. From the 17th century, there was a bigger inter-

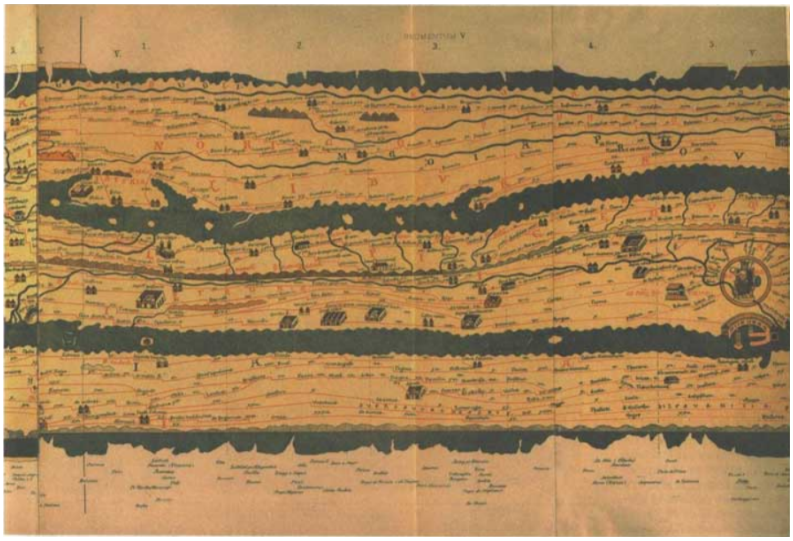


Figure 2.3: The first route map, showing the whole of the Roman world. (http://datavis.ca/milestones/)

est in cartography and data visualization in general in such a degree that almost every new century the advancements are comparable in number and importance with those of the previous 1700 years. Referring to all these new ideas even though can be interesting are not relevant to the concept of this thesis.

The only century that it appears to have the relevance of its innovations with the present project is the 20th. The first half of the century does present important new ideas in the field of cartography but it is considered important in the history of cartography because it was a period that it was mainly focused on applications [Friendly, 2006]. During this period there was the advancement in photography and the use of airplanes that led to the increase of the map production (making the map production process easier and faster) and as a result, many theories that were formulated in previous centuries found the ground to be implemented and analyzed in more depth. Such theories are the use of colors in maps, in order to improve the understanding of the maps by the audience (until then the majority of the maps were black and white), and the map generalization. Lastly during the first half of the 20th century was the time that modern animation techniques and computer systems were invented two technologies that will be used later in the same century.

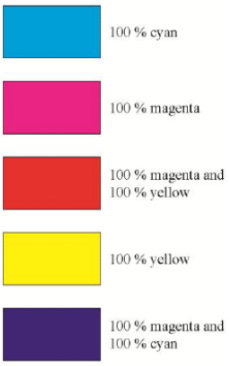
In the second half of the 20th century, it was the time that new ideas are formulated again at an accelerating rate especially after the pause that it was experienced during the first half of the century. The most important innovation of the period is the use of computers in order to analyze, produce, store and visualize spatial data ultimately creating what is known today as GIS. The use of computing systems helped in the introduction of the interactive maps as well as the animated maps. The last major innovation of the second half of the 20th century is the introduction of the World Wide Web (WWW) that actually defined the next century (21st) and will be analyzed in the next section (Section 2.2)

2.1.2 Colors in Maps

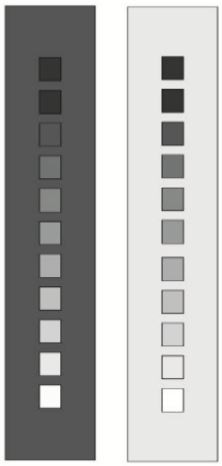
Adding color in maps is not an idea that was first introduced during the 20th century but rather an idea that it was popularized during that period. After all, as it has already mentioned in Section 2.1.1 the early 20th century it was a period of applications. The main reason that it became more popular it has to do with the extended use of photography since until then all the maps and especially those with color where hand made a really difficult and time-consuming method.

According to Christophe et al. [2011] the "Color is physiological sensation resulting from all radiations received by the eye when looking at an object lit in solar light. We can define color as a personal impression but it is difficult to measure it, because variations in human perception are big, and it's quite impossible to make standard, objective observations or develop standardized or quantitative rules for using color". In the context of cartographic visualization, the color can be considered as the most important graphic variable or means of cartographic expression. In the cases that the color is well-chosen the content of the map is easier to be distinguished. Also, the color can evoke certain feelings to the user that can affect his understanding and decision-making process to a larger degree than with any other cartographic mean of expression [**?**]blaha2014).

There are three main dimensions of the color and these are the hue, the value, and the saturation. The hue is defined as the different colors that humans can perceive (e.g. red, green, blue)(Figure 2.4a). The value is defined as the lightness or the darkness of a hue. The value of a hue can be affected by the background and for example, the value looks lighter when surrounded by darker shades or vise versa (Figure 2.4b). The last dimension is the saturation that is defined as the intercity of color and is related to the percentage of gray that is contained in a color (Figure 2.5)[Christophe et al., 2011].



(a) Example of different hues.([Christophe et al., 2011])



(b) Example of different values of gray with a different background color.([Christophe et al., 2011])

Figure 2.4: Examples of different hues and values

The first that worked on color theory was Newton that his primary focus was the analysis of the visible spectrum of the electromagnetic spectrum (380 to 780 nm) and how the "white" light can decompose into seven basic colors and then how these seven colors can be recomposed into the white light. The second that worked on color theory was Goethe that he was devoted to answering the physiological, physical and chemical questions related to color, as well as to psychology and aesthetics. Itten was one of the pioneers of the color theory and he was the one that divided the color spectrum into three primary colors (yellow, red, blue), the secondary colors (orange, green and violet) by mixing the primary ones and finally the tertiary colors (yellow-orange, red-orange, red-violet, blue-violet, blue-green and yellow-green) that is a result of mixing the primary colors with the secondary ones [**?**]blaha2014)(Figure 2.6).

The use of color in cartography is extremely important mainly for surface areas since it affects the map's harmony the most. According to Christophe et al. [2011] "harmony is the art of how to bind a variety of colors in a "good" equilibrium of threshold contrasts (intrinsic and spatial), giving a sense of understanding particu-

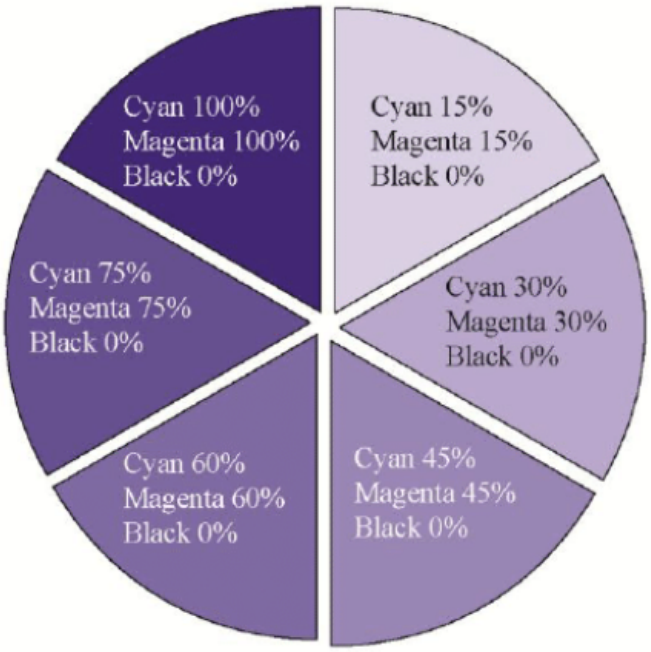


Figure 2.5: Example of different saturation for the same hue and value.([Christophe et al., 2011])

larly subjective". Equally important as the selection of colors for the surface areas is the selection of the color of the points and lines that are typically the overlaying layer of a map [Bláha and Sterba, 2014].

The three dimensions of color (hue, value, saturation) can be used in order to produce a more harmonious map. The difference in hue and especially the use of the three primary colors can be used in the depiction of qualitative phenomena like the demarcating of regions on political maps. In the same concept of different hues, it is important to mention the theory of cold and warm colors. Warm colors are those that are associated with the yellow and red hues (top part of Figure 2.6) and the cold colors are associated with the blue hues (lower part of Figure 2.6). The contrast between warm and cold colors is typically associated with positive and negative values of a phenomenon (Figure 2.7).

The next dimension is about the value of a color (how bright or dark a color is) that in cartography is mainly associated with quantitative phenomena, in which light tones are used for lower values and dark tones for higher values (Figure 2.8a). The last dimension is the saturation of the color that is used for the improvement of the formation of gure and background, for differentiating special terrain and for the visual balancing of large geographic phenomena, such as forests or lakes[?]blaha2014))(Figure 2.8b) .

2.1.3 Animated Maps

Today one of the most significant research challenges is the monitoring of the environment and pend upon capturing, analysing and representing dynamic geographic processesHarrower et al. [2008]. For thousands of years the cartographers where trying to perfect the visualization of dynamic phenomena with static 2D maps. Since the early decades of the 20th century some cartographers where experimenting with animated map displays. According to [Harrower et al., 2008] animations are "defined as sequences of static graphic depictions (frames), the graphic content of which, when shown in rapid succession (typically 24-30 frames per second), begins moving in a fluid motion". The animated maps have become really popular nowadays not only among the experts for data exploration and knowledge discov-

Figure 2.6: Itten's 12-hue color circle.([Bláha and Sterba, 2014])

ery but also to the simple user that can, for example, see the weather maps loops on television. The main advantage of the animated maps and the reason that they have become really popular the recent years is the fact that the passage of time can be represented congruently.

The first attempts for the production of animated maps date back to 1930s where the animation process was manual drown by hand each frame at a time using cartoon animation techniques. Interestingly one of the first popular animated map was created by the Walt Disney Company in 1940 and it was depicting the invasion of the Warsaw by the Nazis in 1939 (Figure 2.9)(cartography2.org [2018]).

In 1963 it was the time that the first computer assisted animated map was produced in bell laboratories. This visualization was showing the orbital path and position of a satellite around a planet. Even though the animated map was produced and displayed through a computer, all the frames were transferred to conventional film in order to store the animation. During the 1980s the desktop computers arrived and as a result the animated maps had found a new production platform, a new storage method and a new display device. But the most important characteristic of this era was the massively decreased cost for the production of the animated maps that was a result of the advancements in both the hardware and software of the computing systems. After the 1990s the introduction of the World Wide Web gave the opportunity to display animated maps through the internet (Section 2.2.4).

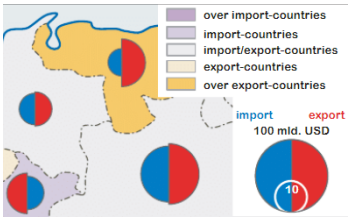
2.1.4 Geographic Information Systems

One of the most important developments of the 20th century in cartography is the development of Geographic Information Systems (GIS). According to Clarke [1986] the GISs are computer-assisted systems for the capture, storage, retrieval, analysis and display of spatial data. The spatial data are data related to the "where" things are, were or will be. More specifically, spatial data are related to the geographic space which is having positional data relative to the Earth's surface.

Our world is dynamic and many aspects of our lives are changing over time. These changes can be caused either due to natural phenomena (e.g. volcano, earth-

Figure 2.7: An example of warm-cold contrast that indicates the positive and negative values of a phenomenon in this case the population growth of a region.([?]blaha2014))

(a) Example of a map that uses different value of a color (employment in agricultural sector).([Christophe et al., 2011])



(b) Example of a map that uses saturation in order to distinguish the background from the foreground (foreign trade) .([Christophe et al., 2011])

Figure 2.8: Examples of maps that utilize the differences in value (brightness) and saturation for improved harmony.

quakes) or by us humans (e.g. reclaiming land from the sea). There are also and some changes that we do not yet fully understand why they happen. So, to better understand our world we study the phenomena that bring about geographic change having as a goal not only to deepen our understanding but also help us in our decision making to follow the best course of action. One of the biggest challenges that are related to many GIS applications is the solution of spatiotemporal problems. In these cases, the problem's characteristic does not only change between different locations but also through time [Huisman and De By, 2009]. Overall, the desire of humans to understand how our world works and how we can utilize this knowledge to improve our decision making led to the development of spatial analysis and ultimately of Geographic Information Systems.

The first attempt to try to explain a phenomenon that is related to the geographic space took place in Paris, France in 1832 that geographer Charles Picket tried to visualize the number of deaths by cholera per district of the city. Following the footsteps of Picket, John Snow determined the source of the cholera outbreak in London in 1854. These two cases are considered for many as the first attempts of spatial analysis.

In the early years of the 20th century, it was the development of the photozincography that made it possible to divide a map into layers simplifying the process that until then was time-consuming and most of the time confusing for the draughtsman. The layers were at first drawn on glass plates that later where replaced by plastic films. The use of layers in map-making is now considered as the main feature of any modern GIS.

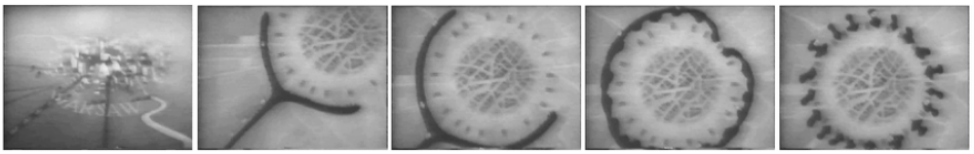


Figure 2.9: Five frames from the Disney Studios production of the invasion of Poland in 1939.(Peterson [1999])

In the late 1950s and 1960s advances in computing technology were making possible forms of automated cartography that in due course would lead to the development of GIS. The 1960s was also the decade that the development of scanners and plotters took place that along with rapid advances in software, began to open exciting possibilities, even at that very early stage in the development of computing Goodchild and Haining [2004].

The first true GISystem was created due to Roger Tomlinson's pioneering work to initiate, plan, and develop the Canada Geographic Information System resulted in the first computerized GIS in the world in 1963. The Canadian government had commissioned Tomlinson to create a manageable inventory of its natural resources. He envisioned using computers to merge natural resource data from all provinces. Tomlinson created the design for automated computing to store and process large amounts of data, which enabled Canada to begin its national land-use management program. He also gave GIS its name [ESRI, 2019].

The development of computer systems and GIS produced a new type of data structure called raster or bitmap. A bitmap is a rectangular grid of pixels, with each pixel's color being specified by a number of bits. The main characteristic of a bitmap the width and height of the image in pixels and the number of bits per pixel (which determines the number of colors it can represent) Figure 2.10. This new data structure is the widest used data structure for storing geographic information surpassing the more traditional one called vector data structure. The vector data structure is defined by points that are connected by lines and curves to form polygons and other shapes Figure 2.11.

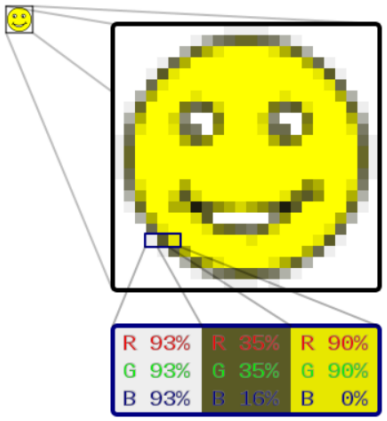


Figure 2.10: An example of a raster image.(https://en.wikipedia.org)

During the late 1990s and early 2000s, the internet and the World Wide Web started to become extremely popular among the users and as a result, the concept of web mapping started to flourish within the GIS community. The development of free-to-use and easily accessible mapping applications like Google Maps, Bing Maps, and OpenStreetMaps, gave public access to huge amounts of geographic data.

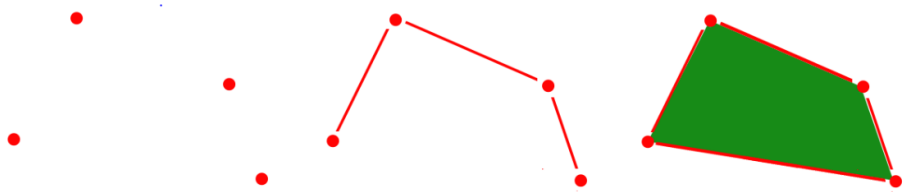


Figure 2.11: The points, lines and polygons/shapes that represent the vector data stucture.(https://gisgeography.com)

2.2 CARTOGRAPHY IN THE 21ST CENTURY

This section discusses the new ideas that were introduced in cartography during the 21st century. Even though it has been only two decades into the 21st century, some major concepts have been introduced in cartography that has changed the field quite heavily. These new concepts are Web GIS and web mapping. The World Wide Web was invented during the last decade of the 20th and the first web GIS and mapping applications were introduced, but from the beginning of the new century (21st) it started to become mainstream with a plethora of important applications to be developed.

2.2.1 Background of web mapping

The term web mapping has a short history starting from the early 90s with the development of the World Wide Web. This discipline has three main elements: the geodata/geoinformation and their visualization, a geospatial application (Web GIS application/platform) and the web [Veenendaal et al., 2017]. Overall, the terms web mapping and Web GIS are getting confused with each other and in many cases, they are considered synonymous, but the term web GIS is considered as more general [Veenendaal et al., 2017].

Web mapping in its development over the years has been heavily influenced by the parallel development of the general web. The general web is divided into four different generations described with the terms Web 1.0 to Web 4.0 [Choudhury, 2014]. Each generation of the web is described as the hypertext web (Web 1.0), social web (Web 2.0), semantic web (Web 3.0), and ultra-intelligentnt web (Web 4.0) [Choudhury, 2014]. Similarly, web mapping has been also divided into different eras from various researchers (Table 2.1)

mapping eras.PNG

Peng & Tsou, 2003	Tsou, 2005	Plewe, 2007; Tsou, 2011	Hall & Tiropanis, 2012	Multiple Authors over Various Years
• Static map publishing • Static web mapping • Interactive web mapping • Distributed GIServices	• GIS awareness – free satellite imagery, disaster response • User response times – AJAX, image tiling • Virtual globes & online mapping services	• First generation – static • Second generation – dynamic • Third generation – interactive • Fourth generation – virtual earth globes • Fifth generation – Cloud computing, RIAs, crowdsourcing	• Web of documents • Web of people • Web of data & social networks	• Web 1.0 • Web 2.0 • Web 3.0 • Web 4.0 • Web 5.0?

Table 2.1: Previously characterized eras of web mapping ([Veenendaal et al., 2017].

The above-mentioned eras have been defined by a major development that has played a significant influence on web mapping developments (technology, data) and communities [Veenendaal et al., 2017]. Even though all the above-mentioned developments are related to this thesis project, it is important to focus on one main part of the developments, the visualization. The most recent classification of web mapping has been proposed by Veenendaal et al. [2017] and it is demonstrated in fig:frameweb.

of web mapping eras.PNG

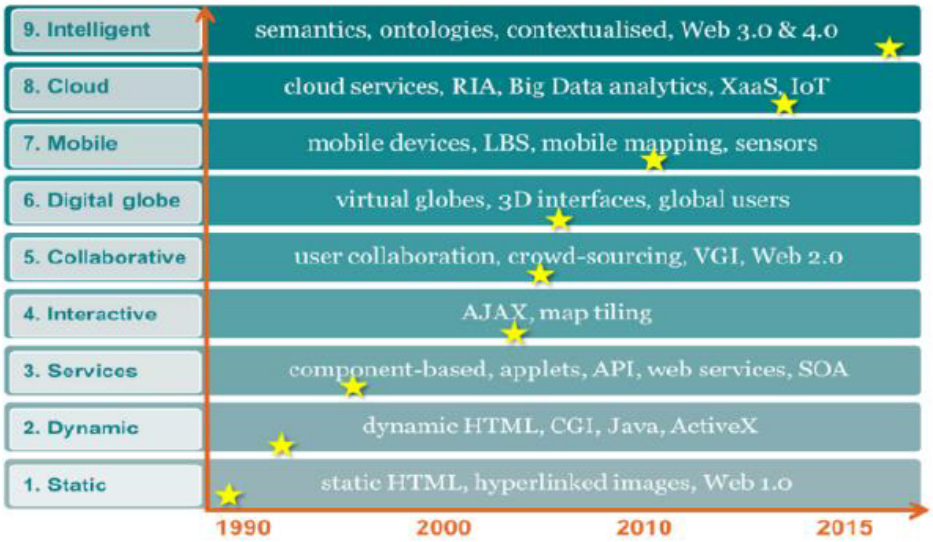


Figure 2.12: Framework of web mapping eras [Veenendaal et al., 2017].

In Figure 2.12 there are defined nine eras (classes) and almost all of them are related to the visualization, starting from the static and the dynamic in the 1990s, continues to the services and reaches to the interactive, the digital globes and the mobile in the 2000s.

The visualization part of the web mapping can be divided into two main categories the static maps and the dynamic ones. The term static map is rather clear and in the case of maps it represents simple images that represent one moment in time and does not allow any interaction with user [Adnan et al., 2010]. On the other hand, the dynamic visualizations as a term are rather general and, in related literature, have different meanings. Some papers when referring in the dynamic visualizations are oriented to maps that the user has the ability to interact (e.g. digital globe) [Veenendaal et al., 2017] [Adnan et al., 2010] or in other papers it is oriented towards representations that the time is also a parameter of the visualization [Harrower et al., 2008] [MacEachren, 1998] and this second category of dynamic visualization is related with the term animation. In the context of this thesis project, the second approach on the dynamic maps will be used and with the term dynamic, the presence of the time dimension will be implied. Even though the development of the digital globe had major importance on the development of web mapping introducing the third dimension on the users and a more accurate representation of the earth, it can actually be considered as a subcategory of the two main categories of maps, static and dynamic, depending on the inclusion or not of the time as dimension. As a result, this chapter will refer to three main categories of representations and these are the static, the interactive (static) and the dynamic (time series).

2.2.2 Static visualization

Throughout the course of history and until very recently the only way that geographic information can be visualized is by drawing a static map on a paper. According to [Roth, 2013] many researchers believe that these paper maps are interactive but through the advancement of technology and the digital revolution has made possible different levels of interactivity for the user and as a result, the old paper maps are still considered as static. Even though the digital epoch that we live in is providing a lot of interactivity options the first maps that were found on the web were static. The static web mapping era refers to the beginnings of Web 1.0 in the early 1990s, where the phenomena of a web map were realized through an Hypertext Markup Language (HTML) image, especially clickable images, and hyperlinks. Web 1.0 defines the read-only web focusing on the retrieval of information from what was then essentially a data repository of web pages. This era was founded on the basic Hypertext Transfer Protocol (HTTP) and HTML technologies implemented to disseminate online linked information according to the vision of Berners – Lee [Veenendaal et al., 2017].

Even though this first category of static maps does not allow any user interaction with the maps because they are just static images, so they do not allow users to pan or zoom in, out or around the map, the resulting map is very simple and easy to interpret, because there is nothing required on the client-side. Because, a user cannot interact with the maps they are not suitable for rich Internet applications, or the new Web 2.0 applications [Adnan et al., 2010]. An example of such visualization is presented in Figure 2.13

sea surface.PNG

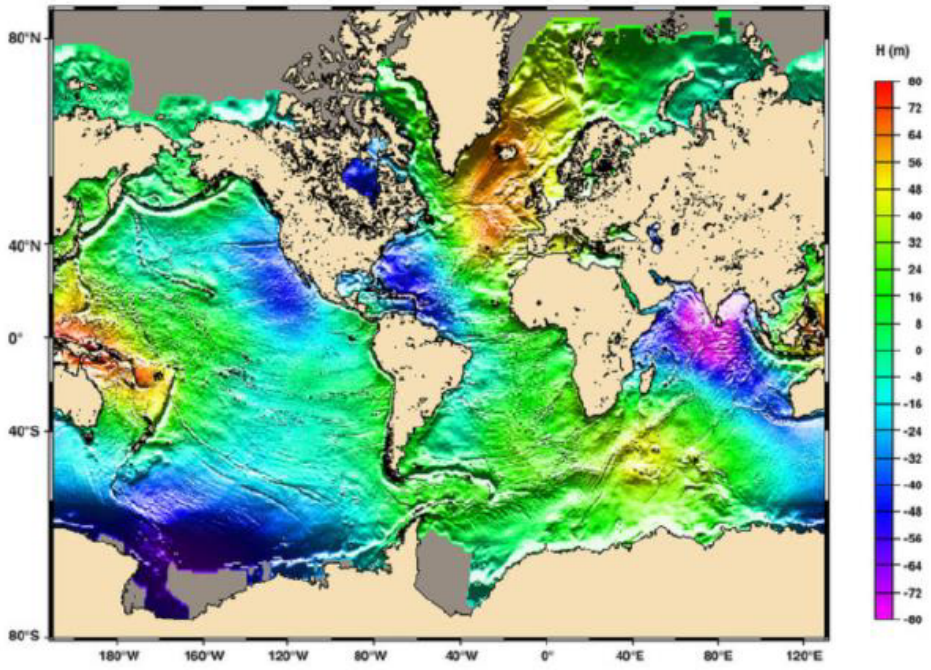


Figure 2.13: Global mean sea surface height from altimetry (https://www.esa.int)

2.2.3 Interactive Visualization (Static)

The interactive maps were possible due to the digital revolution that allowed the map-maker to add elements to the visualization according to the user's needs. The web mapping discipline was the natural second step after the introduction of the

static maps in the early 90s. The first attempts were using Dynamic Hypertext Markup Language (DHTML), Common Gateway Interface (CGI), Java applets and servlets, plugins and ActiveX technologies that enabled users to retrieve maps that were dynamically created and constructed on the server according to user preferences and choices [Veenendaal et al., 2017]. This kind of first interactive representation was used to produce the first interactive online atlases and the beginnings of GIS functionality on the web.

The limitation of the early dynamic maps was that client requests and server responses were synchronized so that users, after submitting a map request, had to wait patiently for the server to assemble and return the response. The technological solution to enhance user interaction with the map was to provide user-client interaction simultaneously with client-server interaction. The emerging Asynchronous JavaScript and XML (AJAX) technology combined with image tiling technology significantly enhanced user interaction with the map by allowing online maps to be delivered to a user in a continuous and responsive manner while the user was simultaneously interacting with the map interface [Veenendaal et al., 2017]. Google Maps, Microsoft Virtual Earth, Yahoo Maps, Mapnik, Open Layers, and ArcGIS, as well as derivative products such as MapTube, are based on this new technology that revolutionized the web mapping discipline [Adnan et al., 2010].

maps.PNG

Figure 2.14: A snapshot of the Google Maps that is a typical example of an interactive map (https://www.google.nl/maps/)

The introduction of the interactive maps on the web happened in parallel with the rapid increase of the web maps in general. All these web maps should not be meant to be used only by the users but also by other applications. The interoperability of the spatial data was so important that many companies, institutions, and universities founded the OGC. Some of the first standards issued by the OGC include WMS with the first version released in 2000 for web maps, Web Feature Services (WFS) in 2002 for vector-based features, Web Coverage Services (WCS) in 2005 for multidimensional coverage data, and Web Processes Services (WPS) in 2007 for processing operations. [1] The number of geospatial standards has since increased to over 50 specifications currently released by the OGC to support web mapping and the GeoWeb [Veenendaal et al., 2017].

In order to explain how the applications that are implementing these standards work, there will be the use of WMS as an example. In general, most modern web applications have two main parts: the server (a system for storage and computing that is more powerful and efficient than the machine that the users normally have) and the client (the web browser that runs on the user's machine). The client

1 http://www.opengeospatial.org

through its interface sends requests through a web server (a software that handles the requests and responses of/to the client) to provide data from a local file system (server) (Figure 2.15).

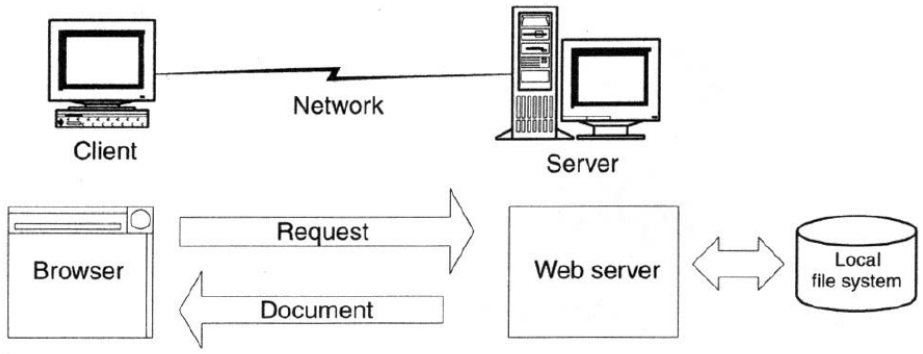


Figure 2.15: A diagram that describes the request response process.([van Oosterom and de Vries, 2018])

The OGC standards and in this example the WMS is providing a specific format for this request-response process in order to be able to be used from all the applications that are designed following these standards without the requirement of any kind of extra computation/transformation/change. For the case of WMS the standard dictates that the first request from the client is the GetCapabilities that is actually the get metadata operation. The response is an Extensible Markup Language (XML) file. The second step is the GetMap that the response should be in one of three data formats: raster, Scalar Vector Graphics (SVG) and Keyhole Markup Language (KML). The last step is related with any extra information that the user may request through the GetFeatureInfo request (Figure 2.16).

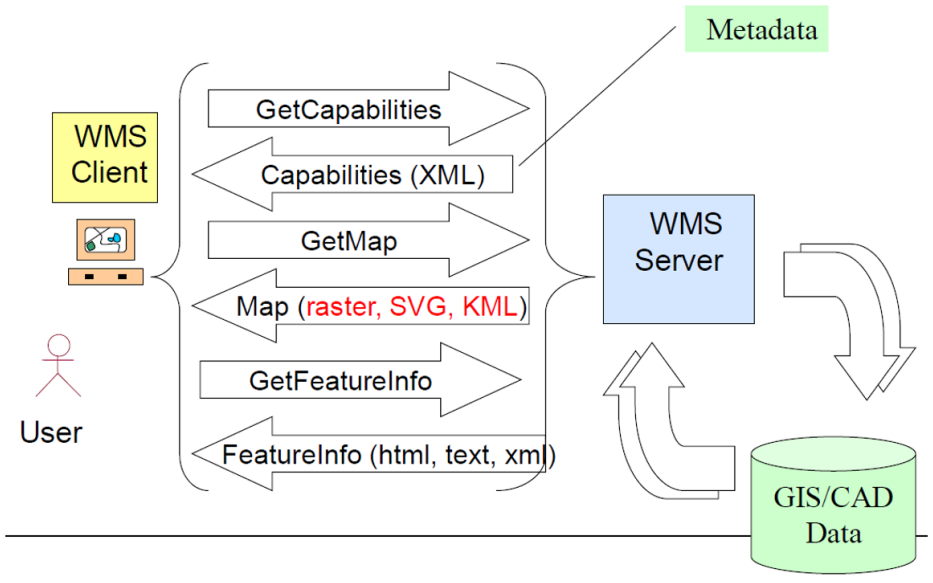


Figure 2.16: A diagram that describes the request and response process of WMS .([van Oosterom and de Vries, 2018])

The most advanced part of the interactive maps are the digital globes that introduces the third dimension on the visualization. The digital globes made the visualizations more immersive for the users and as a result improved their efficiency. While the number of virtual globes is increasing, new technologies that

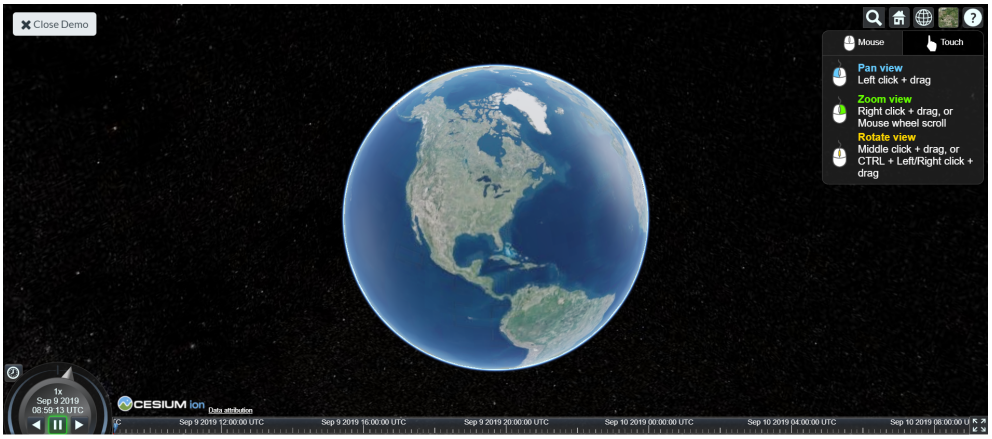


Figure 2.17: A snapshot of the Cesium that is a typical example of a virtual globe (https://cesiumjs.org)

enable open and pervasive development of virtual globes and their applications are rapidly evolving. A number of open-source solutions, e.g., NASA World Wind, Cesium WebGL, Glob3 Mobile and osgEarth, have made the development of virtual globe applications easier [Veenendaal et al., 2017].

2.2.4 Dynamic visualization (Timeseries)

The last category of mapping visualizations is the animated map (time-series). A description of what the animated maps are can be found in Section 2.1.3.The implementation of animated maps on the web at the very early stages of its development was using the animation format of Graphics Interchange Format (GIF) that is one of the two first image formats that were implemented [2]. After the introduction of GIF on the Web, the Adobe Flash was the main animation source on the web and was also used for web mapping animation. Another format that was used for animating maps was the SVG that as its name implies was created for managing vector data. According to [Midtbø, 2005] there are many examples of animated maps but the majority of those examples has been entirely prepared in advance and offer poor analyzing functionality beyond the visual inspection assisted by some interactive tools.

2 https://en.wikipedia.org

3 RELATED WORK

In this chapter, some example web visualizations will be presented related to the main techniques that are been used for the visualization of spatial data. The examples that will be presented are: the **Google Maps**, the **Ocean Motion**, **Discover Magazine**, **Earth Nullschool** , **Google Earth Engine Timelapse** and NOAA **Sea Surface Height Anomaly**. Each one of the applications will add a small stone to reach the goal that is to create an optimal dynamic and interactive visualization for the SSH phenomenon.

3.1 GOOGLE MAPS

The most characteristic and widespread example of an interactive (static) map can be considered the Google Maps (Figure 3.1) but the recent years the visualization has changed from a typical 2D visualization to a virtual globe one. Nevertheless, it is not possible to talk about web cartography without referring to the Google maps.

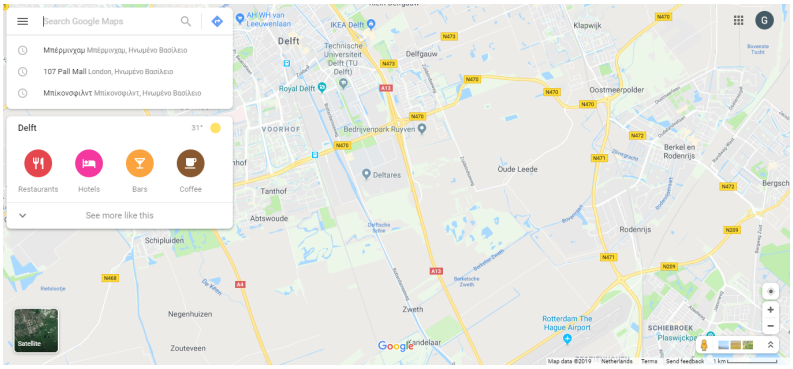


Figure 3.1: Google Maps print screen (https://www.google.nl/maps/)

Despite the many capabilities that this implementation has, the most important ones are the rather smooth zooming and panning capabilities as well as the introduction of the Web Mercator projection.

The developing team of Google Maps was one of the first that introduced the tiled web maps (for raster data format) and many other web map vendors followed their example and nowadays the tiled web maps are considered as the most common technique for the web maps. The conventions used for implementing the map tiles for Google Maps is now used widely from most of the vendors. These conventions include the tile shape and size that is rectangular with dimensions of 256x256 pixels. The numbering of the tiles for the zoom level starts from the 0 in which the whole world fits inside a tile (256x256 pixels). Then, in zoom level 1 the first tile is replaced by exactly 4 tiles of 256x256 pixel size. In general, each tile is replaced from 4 tiles for the next zoom level reducing at the same time the ground resolution (Figure 3.2).

The numbering of the tiles does not refer only to the zoom level but also to the tiles for each zoom level. The numbering for each tile in every tile zoom is described by 2 numbers x, y, with the x referring to the row starting from 0 in 180 degrees longitude and it increases while moving eastwards. The y coordinate is referring to the row starting from 0 in 85.051129 degrees latitude and it increases while moving

Level	# Tiles	Tile width (° of longitudes)	m / pixel (on Equator)	~ Scale (on screen)	Examples of areas to represent
0	1	360	156 412	1:500 million	whole world
1	4	180	78 206	1:250 million	
2	16	90	39 103	1:150 million	subcontinental area
3	64	45	19 551	1:70 million	largest country
4	256	22.5	9 776	1:35 million	
5	1 024	11.25	4 888	1:15 million	large African country
6	4 096	5.625	2 444	1:10 million	large European country
7	16 384	2.813	1 222	1:4 million	small country, US state
8	65 536	1.406	610.984	1:2 million	
9	262 144	0.703	305.492	1:1 million	wide area, large metropolitan area
10	1 048 576	0.352	152.746	1:500 thousand	metropolitan area
11	4 194 304	0.176	76.373	1:250 thousand	city
12	16 777 216	0.088	38.187	1:150 thousand	town, or city district
13	67 108 864	0.044	19.093	1:70 thousand	village, or suburb
14	268 435 456	0.022	9.547	1:35 thousand	
15	1 073 741 824	0.011	4.773	1:15 thousand	small road
16	4 294 967 296	0.005	2.387	1:8 thousand	street
17	17 179 869 184	0.003	1.193	1:4 thousand	block, park, addresses
18	68 719 476 736	0.001	0.596	1:2 thousand	some buildings, trees
19	274 877 906 944	0.0005	0.298	1:1 thousand	local highway and crossing details

Figure 3.2: A table with the number of tiles and the scale of each zoom level (https://wiki.openstreetmap.org/)

southward (Figure 3.3). Since there is not yet an efficient way to create the tiles on the client-side all the rendering is taking place on a server and the tiles are streamed to the client.

Even though the tiling scheme of Google Maps is the widest spread other tiling schemes are being used with most of them having small differences with each other. One of the first standardized tiling schemes was produced by the OSGeo and was named Tiled Map Services (TMS). The main difference of this tiling scheme is the fact that the tile numbering starts from the south of the map (-85.051129 degrees) and increases towards the north (Figure 3.4). A second tiling scheme that has a different encoding of the tiles is the one used in Bing Maps. The new encoding is based on the Quadtree spatial index structure and it is called "quadkeys" and improves the performance in the storage and indexing of the tiles (Figure 3.5).

One last option that has started to increase into popularity is to use bigger tile size and instead of 256x256 pixels to quadruple the size and reach the 512x512 pixels. Mapbox is one of the main vendors that have changed its focus towards these bigger tiles that, according to their blog (https://blog.mapbox.com/), are better suited for the high-resolution displays on one hand, and it costs fewer bytes in size over the network that leads to faster rendering of the maps.

The second important characteristic of this implementation is the introduction of the Web Mercator Projection. The Web Mercator projection is based on the original Mercator projection that for many years was considered as the most common projection in cartography. The Mercator projection is a cylindrical projection that has as output a map that is rectangular, better fitting in the rectangular screens, and also the meridians and the parallels and perpendicular to each other in every place of the world and also are parallel to the borders of the map/screen that help the users to better orient themselves on the map. The other characteristic of the Mercator projection is that due to the cylindrical projection it has big distortion while mov-

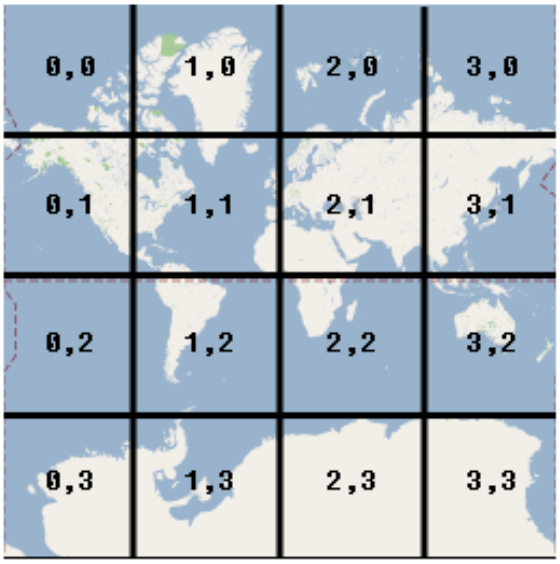


Figure 3.3: Google Maps Tiling Scheme for zoom level 2 (https://developers.google.com/)

ing from the equator either to the north or to the south and that it is con-formal. The Web Mercator uses the WGS 84 ellipsoid but the formulas that are used to project the coordinates on the map are those from the spherical Mercator and no transformation from the ellipsoid to spherical is taking place. The reason for this decision is to reduce the computational cost on the server-side. That leads, on one hand, to have large deviations for the original Mercator, to lose the con-formality (it is retained at small scales) and also the lines of constant course are not straight, but, on the other hand, it retains the characteristics of the cylindrical projection and also if the map is cut at the latitude of around 85 degrees north and south the resulted map is practical a square (Figure 3.6) something that helps with the tilling as explained in the previous paragraph. This projection even though it has been criticized by the Geodesy Subcommittee of the Oil and Gas Producers (aka EPSG) as a not valid projection, over the years it has been accepted as the main projection for web applications with the official identifier: EPSG:3857.

3.2 OCEAN MOTION

The second implementation (Figure 3.7) is a Sea Surface Temperature, Height, Chlorophyll Visualizer, and introduces probably the simplest technique for adding the time dimension on a map using the so-called slideshow. In this implementation, a new image is loaded for every new date with available data (in a specific time interval). The interactivity elements that this implementation is providing are panning in different areas of the world, the change layer selects the date of interest, querying and animation. No zooming capabilities are provided and as a result, fewer tiles are needed comparing with the Google Maps implementation. Due to the lack of the zooming capabilities, it is efficient enough to use an approach like this one (slideshow) for dynamic visualization. On the other hand, most of the modern data-sets are dense enough that there is a need for zooming and if the slideshow technique is used for such application the storage requirements are increasing exponentially.

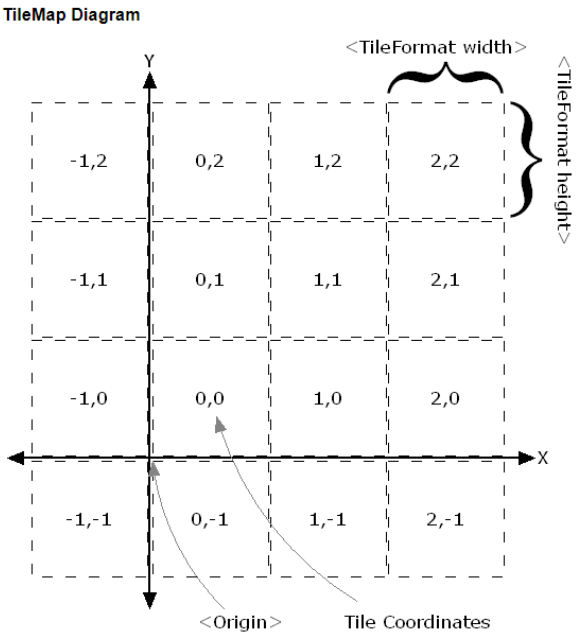


Figure 3.4: The TMS numbering scheme (https://wiki.osgeo.org/)

3.3 DISCOVER MAGAZINE

This implementation (Figure Figure 3.8) is trying to handle more efficiently the storage requirement for a dynamic visualization, which is the use of GIF. The goal of this visualization is to observe the Sea Surface Temperature during El Nino. This format is using the Lemp-Ziv-Welch (LZW) lossless data compression technique to reduce the file size without degrading the visual quality. The main problem with the GIF is the fact that it is a never-ending loop without any possible intervention from the user to select a specific frame. This leads to limited interactivity and less immersive experience for the user.

3.4 EARTH NULLSCHOOL

The Earth Nullschool implementation (Figure 3.9) is considered as one of the best visualizations of the Earth providing multiple interactivity elements like layer selection, zooming and panning, querying and animation. The Earth implementation is a visualization of global weather conditions forecast by supercomputers updated every three hours. The most important characteristic that is introduced in this implementation is the use of a virtual globe. The virtual globes can help the user better understand the spatial data but in this implementation, the time is not represented accurately. The animation that is used is not true dynamic but rather it is static since the value of each phenomenon depicted does not change. The animation is present just to show a trend (e.g. the direction of the wind) and it is mainly a visual trick that helps the user to understand how some phenomena are evolving. There is also the opportunity to jump to different times in the past but this element is handled as in the case of the Ocean Motion and actually, the change can only happen manually without the existence of an animation that could help to have a better general understanding of the phenomenon.

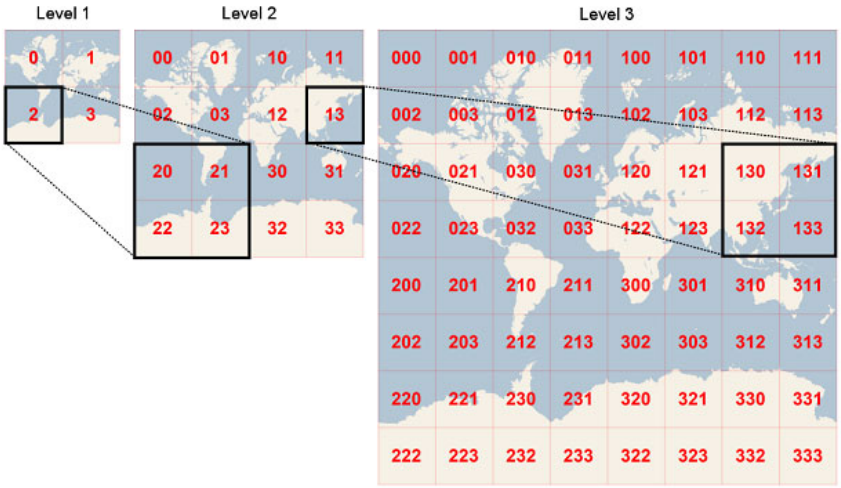


Figure 3.5: Microsoft's Quadkeys encoding (https://docs.microsoft.com/)

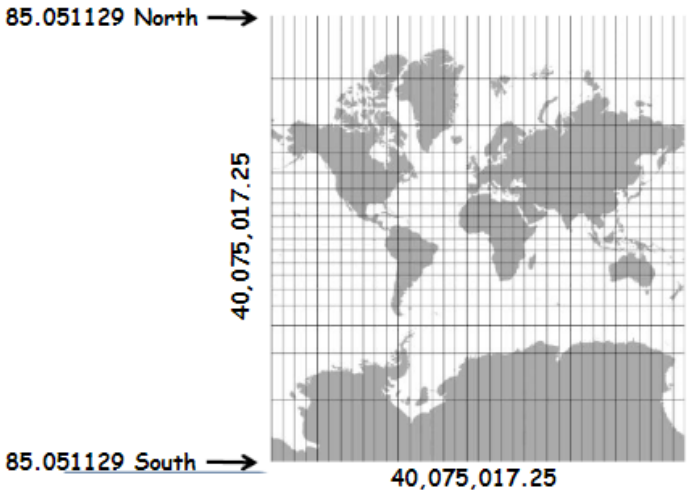


Figure 3.6: The extent of Web Mercator projection. Square shape (approx. 40,000km x 40,000km; equal to the length of the equator line on WGS'84 ellipsoid) (Stefanakis 2017)

3.5 NOAA SEA SURFACE HEIGHT ANOMALY

Continuing on the topic of the virtual globe there is an implementation from the NOAA(Figure 3.10) referring to the SSH Anomaly that handles the time dimension in a more efficient way that the Earth implementation mentioned above and as a matter of fact from any other application mentioned previously. In this visualization instead of using images (.png) as texture as it happens in most cases, it uses a video (.mp4). As a result, it is not only a more accurate representation of the phenomenon but also the user can manipulate the animation. The manipulation of the animation is extended from the start and pause until the speed of the animation and the luminosity. Another useful interactivity element of this visualization is the panning allowing the users to find the area that they are interested in. This implementation even though it is equipped with the best animation among the rest it is far from perfect. The first omission is the absence of zooming capabilities that in this case can be considered as a logical choice since the data-set used is too coarse to that extent that even when the user observes the whole globe it can easily distinguish

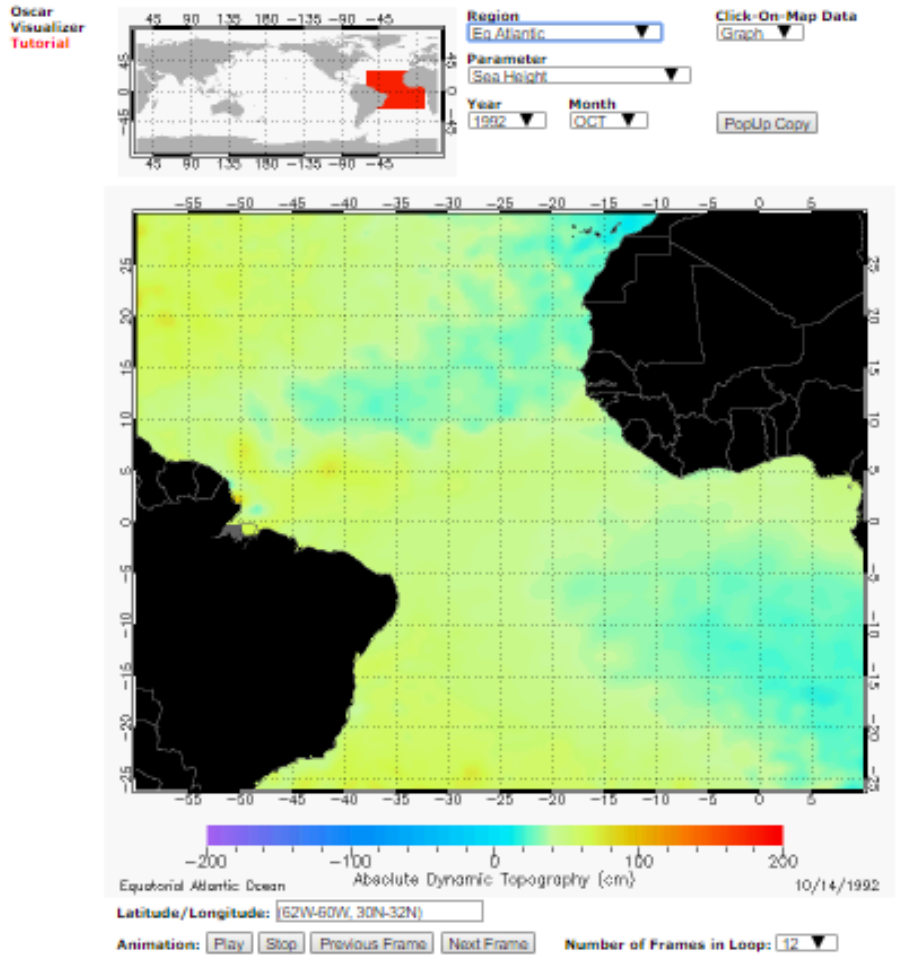


Figure 3.7: Ocean Motion (http://oceanmotion.org)

the different pixels of the video. Also, the user is unable to understand for which period of time the whole animation is referring to as well as each separate frame, so, even though it is possible to select a specific frame from the animation it is of no use since it is not possible to understand to which date it corresponds. Overall, this implementation can give you a general idea of how this phenomenon is changing over time but nothing more.

3.6 GOOGLE EARTH ENGINE TIMELAPSE

The last implementation that will be analyzed is one of the most complete dynamic visualizations that can be found on the web (Figure 3.11). This application is visualizing the differences on the surface of the Earth during the past 34 years using satellite images. It uses videos for the animation and it also has the interactivity elements that can be considered as most important in today's web applications. These elements are the zooming and panning. The important difference of this implementation in comparison with the one from NOAA is the fact that due to the zooming, way more videos should be rendered to have proper worldwide coverage. Actually, this implementation is using a similar technique with the Google Maps in which the map has been organized into tiles, but in this case, the tiles are not images, but they are videos. Another difference is that the tiles are rendered in higher resolution (1424x800 pixels instead of 256x256 pixels) and different aspect ratio that leads to fewer columns than rows, in comparison with the same number in columns

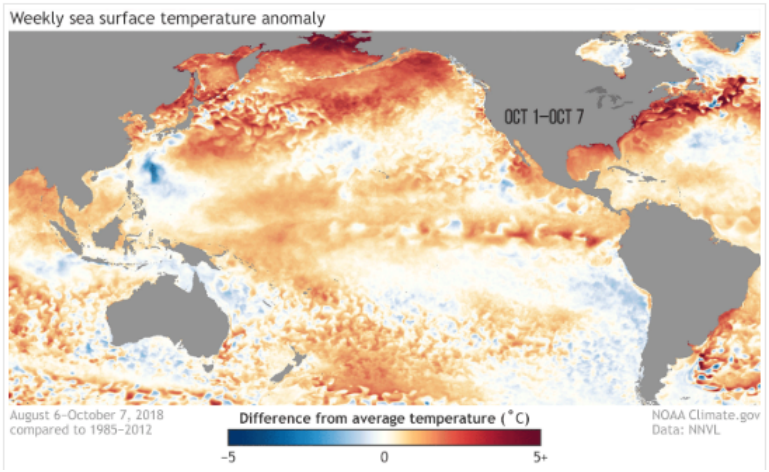


Figure 3.8: This animation shows how sea surface temperatures have departed from the long-term average, from August through early October 2018 (http://blogs.discovermagazine.com/imageo/2018/10/12/visualization-shows-el-nino-brewing-in-pacific/)

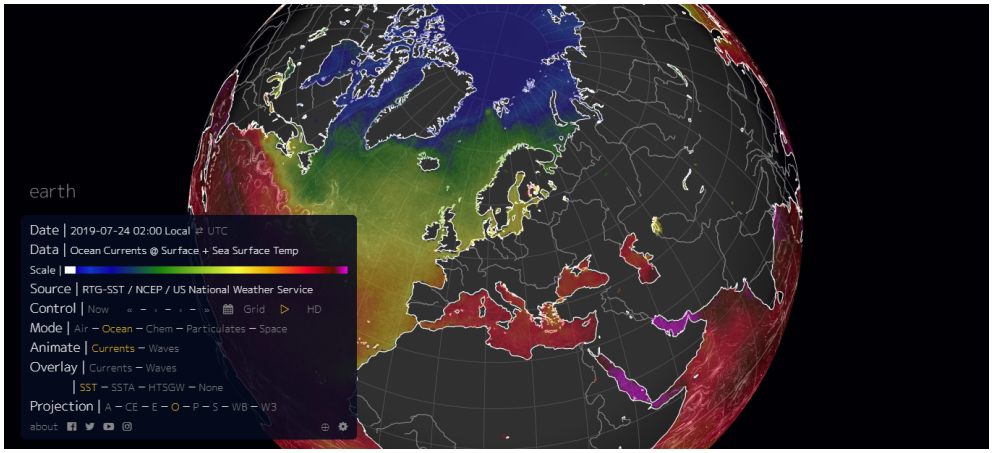


Figure 3.9: The Earth implementation (https://earth.nullschool.net)

and rows that can be found in the most widely used tiling schemes. Two different approaches have been used for this implementation. In the first one the tiles are overlapping (around 75% overlap between the tiles) and every time only one of the tiles is visible with some cases to be able to see at most three. This can help with the synchronization of the videos that it is not even necessary since just the knowledge of the timestamp is enough for the new video(s)'s starting point. In the second approach even though they have produced the overlapping videos like in the previous approach, the overlapped videos are not loaded on the client but only the adjacent ones. The rest of the characteristics of this approach (resolution, number of visible tiles in the viewport) are the same as the first one Figure 3.12. The approach of the overlapping tiles has been withdrawn but through the tile numbering in Figure 3.12 it can be assumed that three extra tiles were added between the two that exists in the current implementation. The problem with the large overlapping tiles is that the number of tiles is increasing exponentially with every new zoom level and as a result, the requirement for storage on the server-side will increase accordingly.

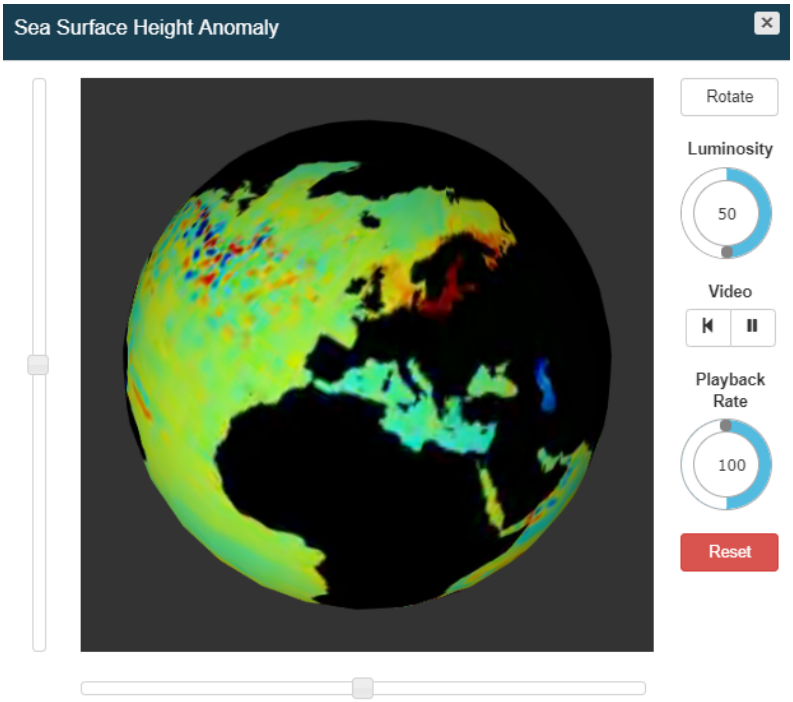


Figure 3.10: The NOAA visualization of the Sea Surface Height (https://sos.noaa.gov/datasets/)

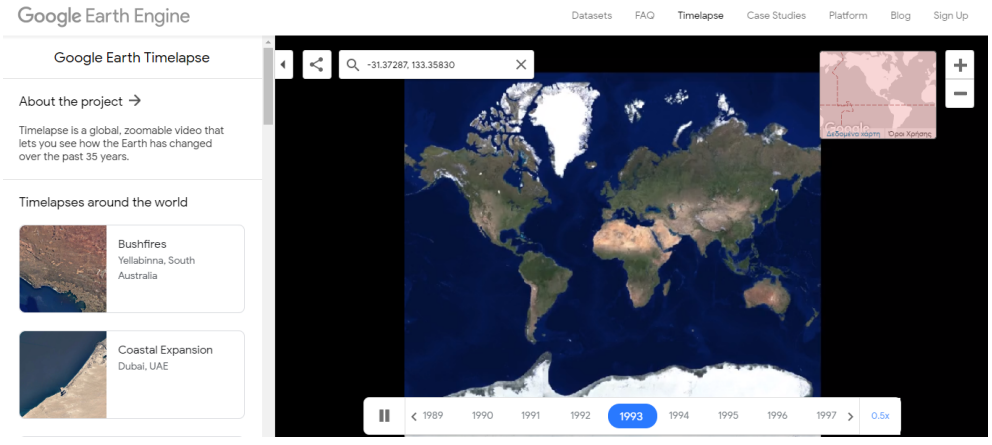


Figure 3.11: The Google Earth Engine Timelapse (https://earthengine.google.com/timelapse/)

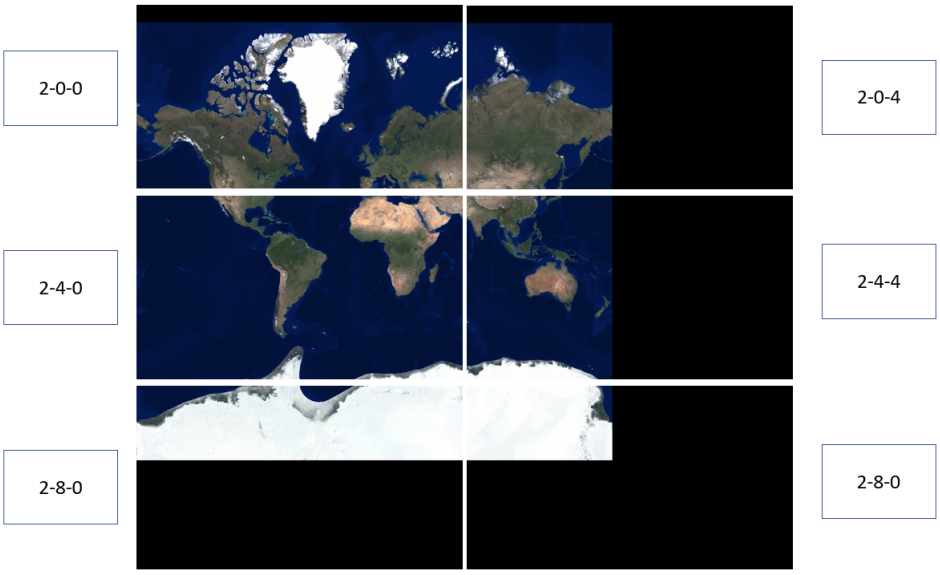


Figure 3.12: The Google Earth Engine Timelapse tiling scheme for zoom level 2

4 METHODOLOGY

In the introductory chapter of this report, the motivation for this thesis has been described as well as the research questions and sub-questions. In this chapter, the methodology to answer the mentioned research questions will be presented. This chapter is organized according to the sub-questions and there will be an attempt to answer them in each of the main sections of this chapter. The first part (Section 4.1) is dedicated to the animation techniques that can be used. The second part (**??**) is dedicated to the different interactivity elements, which ones were selected and, according to the existing technology, how is it possible to implement them, and the last part (**??**) will describe the architecture of the application.

4.1 ANIMATION

There have been three main animation techniques that have been used for visualization on the web. These three animation techniques are the GIF, the SlideShow, and the Video. In this section, there will be a comparison between them with the goal to select the optimal one for this project. The animation that will be used for this project should compile with some requirements to offer to the user the best possible experience as well as to follow the characteristics that can make the implementation optimal.

1. The first 2 requirements for an optimal implementation are about the storage requirements and the streaming speeds something that for the animation can be translated into the size of the files used and ultimately into the **compression** that can be applied for each animation technique.

2. Another important characteristic is the fact that this implementation should be an open-source with any user to be able to have access to it, so, it is important to check the **proprietary status** of its technique.

3. Since the final application that will be the result of this thesis will be interactive, the animation should be also **interactive**.

4. One last characteristic is that the animation should be **compatible with the most popular browsers** available to the users.

The first animation technique that was introduced on the web was the **GIF** and still is considered one of the most common formats used providing up to 4:1 lossless compression. Also, since the GIF was one of the two first image formats compatible with the HTML, after 30 years it is compatible with the most popular browsers. Even though it has some positive characteristics the GIF is far from ideal having some important weaknesses. The first important disadvantage of this technique is the fact that it is limited to only 256 colors or shades of gray. The second one is the fact that the LZW compression technique is proprietary but when the company attempted to collect royalties from the users, they simply stopped using this format. Another important problem that the GIF formats have is that the animation as it is defined in the GIF89a version as well as the Netscape Application Block (NAB) the number of loops or the frame-rate are predefined in the file and the user has no access on them limiting the interactivity of the user with this format.

Using **slideshow** for animation is one of the easiest techniques to implement on the web due to the collaboration of Cascading Style Sheets (CSS)3 and JavaScript. Besides the use of CSS3 for the creation of a slideshow, there is also the WebGL API that can be used for the same purpose. The main difference between the CSS3 and the WebGL is that on one hand the CSS3 is more straight forward in its implementation but on the other hand since WebGL uses the **GPU!** (GPU!) instead of the Central Processing Unit (CPU) is more efficient. One advantage of this technique is the use of image formats that are common in use for web maps since the first years of the web and the first static web maps in the early 1990s. One more important aspect of this kind of animation is that through JavaScript, the users can manipulate the animation adding one more element of interactivity to the final application. As far as the browser compatibility is concerned, the slideshow, since it uses image files it is compatible with the modern browsers, and even though some image file formats are proprietary there are enough formats that are free for the users. The main disadvantage of this technique is the fact that it does not offer any type of compression between frames and as a result, there is a requirement for storage space on the servers.

Since the introduction of the **video element** in HTML5 the videos are the best option for animation on the web especially when the animation is a product of a rather big number of frames. As a matter of fact, the more frames the more efficient the use of video animation becomes. The main reason for the improved efficiency is due to the encoding techniques that are used that lead to high compression rates retaining at the same time high detailed content. If retaining the best possible quality of the outputted file is of the most important, the most commonly used encodings also provide lossless versions. As it is already mentioned in the previous techniques, since the final application that will be the result of this thesis will be interactive, the animation should be also interactive that means that the user should be able to pause the animation or to move forward or backward in time and even to speed up the animation or slow it down, characteristics that are available in a video formatted animation. The video format even though it complies with most of the requirements of the optimal animation technique it also has a couple of drawbacks. The first disadvantage is the fact that not all video encoding standards are compatible with all the browsers, and the second one is the fact that some of the widest spread encoding standards are proprietary.

In order to have a more clear comparison of the three animation techniques that were presented above, the Table 4.1 was created. In this table, an effort has been made in order to quantify the four requirements for each technique. The scale that was used the symbol **" - "** for the technique that was characterized as insufficient, with **" o "** as average and with **" + "** as excellent.

Animation Technique	Compression	Proprietary Status	Browser Compatibility	Interactivity Options
GIF	+	o	+	-
Slideshow	-	o	o	+
Video	+	o	o	+

Table 4.1: Comparison of the three animation techniques

The results of the comparison presented in Table 4.1 shows that the best-suited animation technique (the one with the highest score overall) for this application is the **Video**. Selecting the video as the best animation technique does not finalize this analysis since the video techniques are providing several different **compression standards** as well as different **containers**. So, a second comparison should take place to select the best encoding and container within the possible selections for the video format.

The most important part of a video is the encoding/compression formats that are a content representation format for storage or transmission of digital video content. The first video format standard was the H.121 developed in 1984 and over the years many more standards have been implemented. Some of the most common formats are the H.262 (MPEG-2 Part 2), MPEG-4 Part 2, H.264 (MPEG-4 Part 10), HEVC (H.265), Theora, VP9, and AV1. Even though the number of formats available is large not all of them are supported by the HTML5 video element. Another support issue that multiple formats faces are the fact that even though they can be supported by the HTML5 they are not supported by several browsers.

The second important part of the video is the container format that is a type of file format for storing digital video data on a computer system. A typical video file contains video data in one of the video encoding formats (e.g. H.264) and audio data in one of the audio encoding formats as well as other important information like synchronization information, subtitles and metadata. Not all the container formats support all the video encoding formats. It is common practice when a developer creates a new encoding format also creates a container format. Just like in the case of the encoding formats there is a large number of formats available but only a small number of them can be supported by HTML5 and the most used browsers. As a matter of fact, the HTML5 supports only three video file formats and those are the OGG, the MP4 and the WebM [1]. These three file formats that are supported by HTML5 leads to the limitation of the video encoding formats that are supported. The OGG supports the Theora compression format, the MP4 supports the H.264 and the HEVC formats and the WebM supports the VP8, the VP9 and AV1 format.

In order to start the comparison between the different file formats and encoding formats it is important to define the characteristics of the optimal implementation. The three characteristics (out of the four) that were defined for the animation in general can also be used in this comparison accompanied by two new ones:

1. the **compression**
2. the **proprietary status**
3. the **compatibility**
4. the **quality**
5. the **encoding/decoding speed**

The comparison between the encoding format and the file formats is not an easy task. So, this comparison starts with the one characteristic that is the most straightforward one and this is the compatibility with the browsers.

In Figure 4.1 it is clear that the combination (codec and file format) that is the most compatible with the browsers is the MP4 with H.264. The second most compatible is the combination of WebM/VP8 and it is closely followed by the OGG/Theora, the WebM/VP9 and the WebM/AV1. In the last place is the MP4/H.265 combination. It is important to mention in this point that the first format that was compatible with the HTML5 video element was the OGG/Theora. But, Apple Inc was worried about the proprietary status of the technologies used for the Theora codec and for the fact that maybe if this codec becomes mainstream then the owners of those to ask for financial resources to permit usage. In a similar situation, it is also the MP4/H.264 format but since it was already used widely Apple Inc believed that most of the owners have all-ready reveled themselves and as a result is safer to use. The MP4 format even though it is the widest spread in use it is not free to use. The last container that was introduced was the WebM with Google (the company responsible for the development of this format) has the license distributed freely for the user. Overall, considering the proprietary status of the three main file formats

1 https://www.w3schools.com

Status of video format support in each web browser

Browser	Operating System	Theora (Ogg)	H.264 (MP4)	HEVC (MP4)	VP8 (WebM)	VP9 (WebM)	AV1 (WebM)
Android browser	Android	Since 2.3[46]	Since 3.0[46]	Since 5.0[46]	Since 2.3[46]	Since 4.4[46]	Android Q Beta
Chromium	Unix-like and Windows	Since r18297[47]	Via FFmpeg[48][49]	No[50]	Since r47759[51]	Since r172738[52]	Yes
Google Chrome	Unix-like, Android, macOS, iOS, and Windows	Since 3.0[53][54]	Since 3.0[54][a]	No[56]	Since 6.0[57][58]	Since 29.0[b]	Since 70[61]
Internet Explorer	Windows	Via OpenCodecs	Since 9.0[62]	No[56]	Via OpenCodecs	No	No
	Windows Phone	No	Since 9.0[63]		No		
	Windows RT		Since 10.0[63]				
Microsoft Edge	Windows 10	Since 17.0 (with Web Media Extensions)[64][65][66]	Since 12.0[67]	Needs hardware decoder[c]	Since 17.0 (supports <video> tag with Web Media Extensions and VP9 Video Extensions)[65]	Only enabled by default if hardware decoder present[70] Since 17.0 (supports <video> tag with Web Media Extensions and VP9 Video Extensions)[64][65][66]	Since 18.0 (with AV1 Video Extension)[71]
	Windows 10 Mobile	No	Since 13.0[72]		Since 15.0 (only via MSE)[73]	Since 14.0 (only via MSE)[74]	No
Konqueror	Unix-like and Windows	Needs OS-level codecs[d]					
Mozilla Firefox	Windows 7+	Since 3.5[75]	Since 21.0[6]	No[56]	Since 4.0[78][79]	Since 28.0[80][81]	Since 65.0[82]
	Windows Vista		Since 22.0[83]				
	Windows XP and N editions		Since 46.0[84]				
	Linux		26.0 (via GStreamer)[7] 43.0 (via FFmpeg)[87]				Since 67
	Android		Since 17.0[88]				in Nightly
	macOS		Since 34.0[89]				Since 66.0
	Firefox OS		Since 1.1[90]				No
Opera Mobile	Android, iOS, Symbian, and Windows Mobile	Since 13.0	Since 11.50	No[91]	Since 15.0	Since 16.0	since 57.0[61]
Opera	macOS, Windows, Linux	Since 10.50[92]	Since 24.0[93]		Since 10.60[94][95]	Yes	since 57.0[61]
Safari	iOS	No	Since 3.1[96]	Since 11[97]	Since 12.1 (only supports WebRTC)[98]	No	No
	macOS	Via Xiph QuickTime Components (macOS 10.11 and earlier)					
GNOME Web	Linux and BSD	Needs OS-level codecs[d]					

Table 4.2: Status of video format support in each web browser (https://en.wikipedia.org)

and the codecs that they support is that the only free for use is the WebM with all its codecs (VP8, VP9, AV1), then with some ubiquity on it's status follows the OGG/Theora and the one that is not free to use is the MP4 with the codecs it supports (H.264, H.265)

The next characteristics that can be compared are compression and quality. These two characteristics are analyzed together since the most important efficiency indicator is the ratio between them. When a new encoding format is introduced the developers are looking to improve the ratio mentioned above (compression/quality) and as a result the newer the format the most efficient the encoding. The most efficient one is considered to be the AV1 that through the literature review and different tests was determined that it is between 20% and 30% more efficient than the VP9 and the H.265 that have about the same efficiency with each other [Ozer, 017a],[Ozer, 017b],[Vatoliny et al., 2018]. The difference increases even more when in the comparison comes in the H.264 that it is around 50% less efficient [Ozer, 017a], [Vatoliny et al., 2018]. Lastly, the Theora and the VP8 encodings have similar results as the H.264 and for some analysts, they are not even as efficient as the H.264 [Ozer, 2010],[Maxwell, 2009].

The last characteristic that the different formats should be compared is the encoding/decoding speed. For the comparison of the encoding speed, the results are reversed compared with the compression/quality one since having the best possible ratio between the compression and quality the encoding becomes more complicated and as a result more time-consuming. For this comparison, the fastest one is the H.264 with comparable times with the Theora and then the VP8, the H.265, and the VP9 follows with the slowest of them all to be the AV1. The decoding speeds follows a different pattern in comparison to the the encoding speed. The fastest is considered to be the decoding of VP9 followed by H.264 and in the last place is the H.265 with the AV1 (https://blogs.gnome.org). It is important to mention in this point that both encoding and decoding speeds are depended on the hardware and software that was used for these processes. The affect of the hardware and software on the decoding speed is even more significant and as a result it is not always easy to define the said speed.

The scores that were used in the Table 4.3 have the same logic as in the case of the Table 4.1. As an overall score, the best results are for the WebM/VP9 that does not excel in any category but is average in most of them. The worst score goes for the MP4/H.265 that is not recommended for use in a web application, at least as for the moment of the writing of this thesis. The other four combinations have the

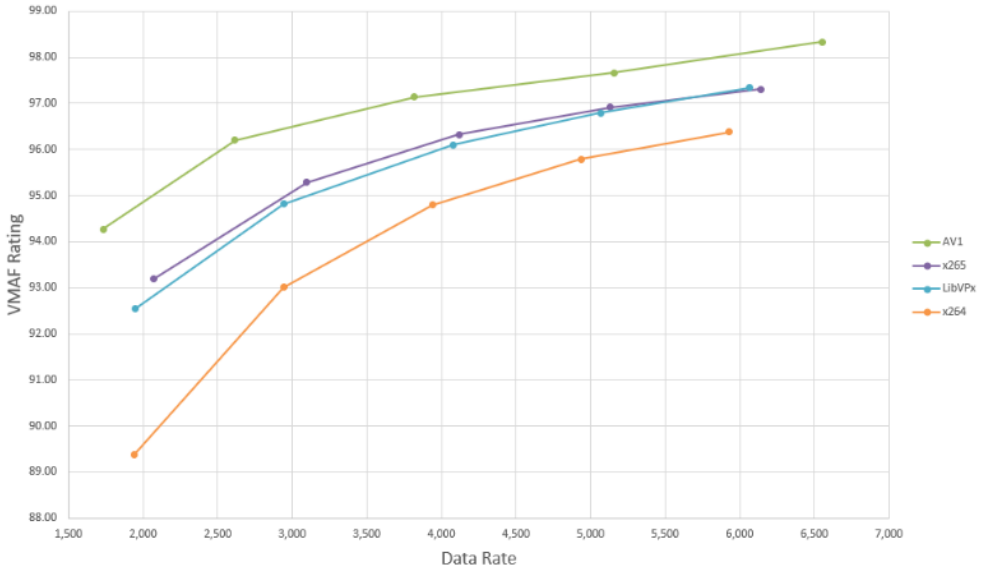


Figure 4.1: Average VMAF quality scores for 4 test clips. (https://www.streamingmedia.com/)

Format Combination (File format/-Codec)	Compression and Quality	Proprietary Status	Browser Compatibility	Encoding/ Decoding Speed
OGG/Theora	-	o	o	+
MP4/H.264	-	-	+	+
MP4/H.265	o	-	-	o
WebM/VP8	-	+	o	o
WebM/VP9	o	+	o	o
WebM/AV1	+	+	-	-

Table 4.3: Comparison of the video formats

same total score but not all of them can be considered as equally appropriate for a web application. From the four it is important to distinguish the MP4/H.264 and the WebM/AV1. The first combination can be considered extremely useful for a web developer since it is supported by all the browsers and if the key characteristic is to create the videos in only one format and never consider compatibility issues then the MP4/H.264 is the format to choose. On the other hand, if the efficiency is the most important characteristic as well as the future-proofing of the application then the WebM/AV1 is the one to choose.

4.2 INTERACTIVITY ELEMENTS

As it has already been established, the advancements in computing have given the opportunity not only to create the GISs but also to produce advancements in the field of interactivity. According to [Roth, 2013] through cartography the mapmakers are trying to communicate with the user some insights they have on a phenomenon and at the same time, the users can apply their knowledge and experience in order to extract their own information for the mapped phenomenon. The layers of information that are available in every given map can be more than one and through the different interactivity elements, the user is able to gain access to as many layers

as possible in a more efficient way. In this section, there will be presented the four interactivity elements that are considered as the most fitting for this application. The selection of the four interactivity elements was a result of a literature review, a comparison between the existent applications and a consideration of the dataset's characteristics. For every element, there will be a sort motivation, a reference on the existing technologies that can help implement each element and a comparison between these technologies in order to select the most fitting one for the application.

4.2.1 Handling the x and y coordinates

In Chapter 2 there was an extended reference on the spatial/geographic data, the maps and the different variations that they exist with the main focus being on the part of the web maps. The web maps have been divided into three main categories that are the static, the interactive and the dynamic ones. The resolution of each of these map categories is the one that will determine the level of details that can be represented on a map. The data that are available nowadays can reach a high level of details and of course most of them have a global coverage. It has been established for many years (since Google Maps) now that the two characteristics of the data (high detail and global coverage) can co-exist in the modern web application and the user should not give up one for the other. As a result, the widest spread interactivity elements that have been introduced into the web maps are the zooming (provides different scales for the map)and panning (changes the position of the center of the map) that are the elements that can help the user to interact with the 2D space in the most efficient way.

The zoom levels for a common web mapping application can reach as many as 23 with a ground resolution of 0.01870 meters per pixel and a scale of 1:70.53. The production of a global raster map with a scale of 1:70 will require an image of prohibitively large size to be displayed. This problem becomes even more prominent when the application that needs to load this map should be on the web that for the average user would have been impossible to load. The problems with a single huge map do not stop only on the size of the outputted map but extend also in other cartographic related elements like the names of the locations on the map, the representation of important facilities, etc. The solution for this problem is not something new, it has been implemented since the era of paper maps and this solution is to use tile maps that represent a specific area and that when combining them the user is able to observe a larger area. This technique was used for the creation of Google Maps that is the first major application that utilizes the tilling technique for the zooming and the panning of the map.

Even though many different implementations have been proposed over the years two of them are the most relevant with this application and those are the tilling scheme of the Google Maps (with its different versions) and the Google Earth Engine Timelapse one that is based on the Carnegie Mellon University CREATE Lab's Time Machine library (with the two different versions).

These two tilling schemes are used as references since they are the ones that have important differences but also they have been used successfully in web applications (like the ones described in the related work chapter). The two main differences between the two tilling schemes are the size of the tiles and the overlap between the tiles. As has already been mentioned the Google Maps tilling schemes rely on 256x256 pixels size while the Time Machine scheme relies on large 1480x800 pixels tiles. This difference leads on one hand to smaller file sizes for the Google Maps approach but also leads to the need for way more tiles to be loaded at the same time in comparison with the Time Machine approach. As a matter of fact, it is common for a typical size viewfinder to display as many as 16 tiles for the Google Maps scheme but only 1 to at most 3 tiles for the Time Machine one. MapBox, one of the biggest map vendors that are using the Google Maps approach is proposing to use bigger tiles (512x512 pixel) in order to compensate for the higher resolution

screens, that are common to modern PCs and mobile devices, but also for the fact that the bigger tiles can be more efficient for the network, rendering faster than the numerous smaller tiles [Lyzi, 2016].

The second part of the comparison is about the overlap of the tiles. While in Google Maps approach the tiles do not overlap, in the Time Machine's first approach did overlap (while in the latest implementation do not). As was discussed in the previous paragraph (about the size of the tiles) having bigger tiles has the advantage that the application requires fewer tiles to be created on the server side and streamed to the client. But, having an overlap (as big as 75%) leads to the need not only to create more tiles but also to streaming them even though there is no need to display all of them. The only reason that can help have overlapping tiles is to better synchronize the videos that it is the biggest issue, that needs to be resolved efficiently and reliably, in having video tiles.

One more issue that needs to be addressed is the shape of the tiles. In the Google Maps tilling scheme, square tiles are being used while in the Time Machine tilling scheme the tiles are rectangular. Some may argue that the rectangular tiles are more efficient due to the fact that the screen is also rectangular. Even though there is an undeniable logic behind the use of rectangular tiles, there is the problem that nor all screens have the same aspect ratio especially when it comes to mobile devices. Having rectangular tiles can be more efficient for some applications (mainly for desktop-oriented applications) but for other applications are not (mobile-oriented applications).

In order to select the best tilling scheme it is important to specify on one hand the different options and on the other hand, the characteristics that each of the scheme should in order to be defined as optimal. The schemes that are going to be compared are the Google Maps scheme with small tiles (256x256 pixels Figure 3.3) and no overlap, the Mapbox scheme that has bigger tiles (512x512 pixels), the Time Machine scheme with the overlapping (75%) large rectangular tiles (1480x800pixels) and the last option is the newer variation of the Time Machine with the same large tiles but without overlap (Figure 3.12). Since the different tilling schemes have been defined, the characteristics of these schemes that the comparison will be based on should be defined too. The first characteristic is the requirement for **storage space**, the second is the **network efficiency**, the third one has to do with the **Existing Technology Compatibility** and the last one is the **synchronization efficiency and reliability** .

The storage space can be considered a rather controversial topic for a web application nowadays. In the application presented in this thesis, due to the fact that the data-set used is rather coarse in order to provide extra information for more than four zooming levels, the number of the videos required is not large enough to consider the storage an issue. The maximum storage space needed for this application is a few hundred megabytes that for the modern servers can be considered negligible. But this application should be also able to utilize a denser data-set without changing the whole approach and as a result, the tile storage can be considered crucial for the efficiency of the application. The comparison is quite straight forward for the first two schemes that use the same number of tiles with the bigger tiles to need more space to e stored. The two-time machine options do not have a difference in tile size but in the number of tiles leaving the option with the large overlapping tiles to be the least efficient one. The most efficient on is the Google Maps scheme with the small tiles and then the Mapbox one closely followed by the last version of the Time Machine that even though it has around four times larger tile than the Mapbox one it needs fewer tiles for every zoom level. The number of tiles changes since the Mapbox tiles are square while the Tile Machine ones are rectangular and as an example for the zoom level 1 the Mapbox requires the typical number of four tiles while the Time Machine only two.

Typically when a web mapping application is designed one of the first decisions is the size of the viewfinder or the "place" that the map will be displayed. The

viewfinder can have a standard size in pixel or it can be a percentage of the size of the screen. Either way, the size of the viewfinder specifies the number of tiles that need to be rendered. Having bigger tiles means fewer tiles to be rendered. For example for one Mapbox tile four Google Maps tiles are needed and due to the video compression the bigger tile is about 15% smaller in size (bytes) than the four smaller ones, and as a result, the bigger tiles are more efficient for the network. Due to the above characteristic somebody could propose that the bigger the tile the better for the application and the network, but the most important part in designing web applications is the user experience. Having smaller tiles it will take more time to cover the viewfinder but it will start loading the tiles one by one starting from the center and the user will have at least "some" map to interact with until the rest of the tiles finish loading.

One important aspect of a proposal of a new methodology for the solution of a problem is whether it is possible to be integrated into the existing technology. If it is easy to be integrated into the existing technology, the chances to be established and used within the community increases. Considering the case of the tilling schemes the existing technology that the proposed methodology should be integrated in the web servers/JavaScript libraries that are being used for serving the static tiles. Most of these servers are designed to mainly serve the small square tiles (256x256 pixels) of the Google Maps and with one parameter change to serve also the bigger tiles of the Mapbox. Serving tiles like the ones of the Time Machine is possible but for most of the softwares and the developers is not something that straight forward to accomplice.

The last and probably the most important characteristic that every tilling scheme should have is the efficient and reliable synchronization of the video tiles in order to ensure the best possible experience for the user. The fewer the videos that need to be synchronized the easiest and more accurate the synchronization becomes. Having at most 4 videos to synchronize is way more accurate than having 16 videos to synchronise since the seeking (for loops) for every video element that is loaded and the setting of the proper timestamps that need to be the same for every video in every given time is not an easy task. The for loops used take some time to be completed (when the number of tiles is increasing) and that can affect the reliability of the synchronization process, a few milliseconds difference and the neighbor-hooding tiles displaying different frames (depends of course on the frame rate) making the experience for the user confusing and unsatisfying. So the fewer the tiles the fastest the seeking and the most reliable the synchronization of the tiles becomes.

4.2.2 Handling the z coordinate

The SSH is a phenomenon that is focusing on the changes in the third dimension of the space (z) that is the height. As it is already mentioned in the scope section of the Introduction, the third dimension will be described with the color on the map. Each user has its way to interpret the colors and what he/she consider as higher value or lower even though through the years some standards have been established in the cartography. Furthermore, the extent of the different pallets that are available nowadays can help the users to better understand the phenomenon and they should be able to select the one that better suits them. So, the second interactivity element that will be added in the application and will be described in this section, is the representation of the third dimension of the data-set with the use of color in order to give the users the opportunity to intuitively understand the changes of the values in the visualization and if there is the case that the original visualization is not satisfying enough for them, to give them the opportunity to select which color pallet is the most suitable for them, after all the goal of this application is to communicate to the users the changes that take place all over the world in the Sea Surface Height phenomenon.

Before starting about changing the color of the tile it is important to be mentioned the original color that every video is encoded in. The videos are encoded in a gray-scale that provides three main advantages. The first advantage of the gray-scale is the fact that the file size of each video is smaller in comparison with an Red, Green, Blue, alpha (RGBa) encoding since there is less information to be encoded. The file size is important not only for the storage requirements but also for the network efficiency since smaller file size leads to faster rendering. The second advantage of the gray-scale is related to the fact that it can effectively depict the differences in multi-dimensional phenomena. The last advantage is the fact that can easily be transformed into an RGBa encoding reducing the calculations needed in order to achieve the different pallets.

In order to achieve the color changing in a video element, there are to main techniques. The first technique is to apply filters on the video with the use of the CSS3. This technique is rather straight forward since the only thing that the developer has to do is to apply a CSS style on the video. This CSS style will contain the filter(s) that need to be applied. The number of filters that can be applied are numerous and some of the most widely used are the blur, the hue, the saturation, the brightness, the sepia, the gray-scale, the invert, and the opacity. All of these filters even though can be helpful in some cases but for a map is not that useful. The pallets that are being used in a typical map contain interpolated colors that can, in many cases, be a result of three different colors. These situations are rather difficult to be achieved by the CSS filters even though the developer has the opportunity to create his/her own filter.

The second technique utilizes the canvas element and more specifically its capability to draw on it the frames of the video. It is possible to extra the frames from the video and saves it as an array with every pixel being described of four values for the Red, Greed, Blue, and Alpha. Then it is possible to manipulate the values in the array for every pixel something that gives the developer excellent control in order to apply the color pallet he/she thinks it is best fitted for the visualization. There are two different APIs that can help the developer to manipulate the canvas element and these are the Canvas API and The WebGL API. even though both of these APIs can achieve almost the same results, they are completely different in the way they achieve it. The Canvas API provides the means to draw mainly 2D graphics on the client using the JavaScript and the Canvas element. The methodology that is used in order to change the color with the help of Canvas API includes the use of two different Canvas elements as well as the video element. The Canvas API uses the CPU of the computer and since the CPU is responsible also for other tasks that run at the same time as the rendering, it is possible to lead to some delays. The WebGL API has been developed for drawing not only 2D graphics but also Three dimensions (3D) graphics. WebGL also uses the canvas element and it is based on the OpenGL ES 2.0 but with the ability to use the HTML elements. The WebGL is a more powerful tool in comparison with the Canvas API but these extra capabilities come with extra complexity on the coding. With WebGL, it is possible to draw both 2D and 3D graphics but always the "drawing" is taking place in a 3D space. Since both APIs can achieve the color changing of the videos the decision on which to be used relies upon the fact that the WebGL API is a way more complicated approach than the canvas API and even though it uses the GPU for the rendering and it is way faster than the Canvas API the simplicity of the latter is the one that makes the difference.

The last part that needs to be mentioned is about the proper selection of the color pallets that will be used (Section 2.1.2). Even though the users will have the opportunity to select the pallet that they prefer, the selection will be among a finite number of pre-selected options. These different color options need to be selected carefully to help the users in decision making. The color picking for a map application is not an easy task and even though throughout the years some standards have been established within the map makers in order to apply specific

colors in specific map elements, as well as the fact that many users due to long years of interaction with maps have developed an understanding of the said standards, it is tricky to predict the perception of every user for every pallet.

4.2.3 Handling the Time Dimension

The last dimension that is important to handle the application is the time. The passing of time affects how a natural phenomenon will evolve and the SSH, as it has already been mentioned in the introduction, is one that is heavily dependable on the time. So, to make as clear as possible to the users how this phenomenon is changing it is important not only to show them how it changes over time but also how it changes while using different time scale. The users will have the opportunity to select the time scale [how dense the data-set that they inspect is (5days, monthly, yearly)] as well as to be able to select which time spam it is of their interest.

Defining how dense or coarse the data are while inspecting them is not a new concept. The zooming capabilities that most of the mapping applications provide is exactly that. The first zoom level in a map (level 0) is the most coarse and for example the whole world fits in a 256x256 pixels tile that translates in 156543.03m/px (spatial resolution). While the zoom level increases the spatial resolution decreases to 78271.52m/px, 39135.76m/px etc until it reaches the maximum zoom level eg. 18 with spatial resolution of 0.5972m/px. The same can also happen with time. It is possible to create zoom levels in time like in the case of space but now instead of pixel there are the frames of the video and instead of spatial resolution there is a temporal resolution. Like in the case of the spatial zoom the temporal zoom when the zoom level increases the temporal resolution per frame will be reduced. The difference with the time is that the information provided in any new level should have a meaning for the user. For example, if the whole time period that is covered by the data-set is 25 years the first zoom level (zoom level 0) should be a map with the average values of these 25 years. If a similar logic is used as in the space division the next zoom level (zoom level 1) should be a "video" with two frames that each frame represents, for instance, the average value of every 12.5 years that is half the value of the first zoom level, then (zoom level 2) it will be a video with four frames for every 6.25 years, then (zoom level 3) a video with 8 frames for every 3.125 years, etc temporal values that have no real meaning for the user. So, the next level (zoom level 1) could have been every 10 years to help the user better integrate the idea of temporal resolution of the phenomenon in a time span that is meaningful for him/her. The problem is solved by using the predefined divisions of time that are, for example, day/month/year/decade/eon, etc. The problem with these divisions is that numerically are rather arbitrary especially for the scale that the users are more familiar with (day/month/year). Nevertheless, since the goal of the application is to help the user better understand how the Sea Surface Height phenomenon evolves there is no alternative but to use these divisions even though it complicates the computations.

In order to achieve the zooming in time there are two main alternatives and these are the creation of every temporal zoom level **on the server** or to have on the server only the temporally denser data-set and then to try to produce the extra zoom levels **on the client**. The first option, on one hand can provide reliable results due to the available tools, but with requirement of extra storage space for every zoom level. The second option provides more efficient storage management but it adds complexity on the calculations on the client, it is less reliable and can use extra computational resources (CPU) slowing down the application as a whole.

4.2.4 Querying Capabilities

One last element that is important to be added has to do with the inspection of the visualized data. The different palettes that can be used in order to visualize the phenomenon can give the users a general idea of how the SSH changes over space and time. Many users want to inspect a specific area and even though the zooming and panning elements of the application can help, still there are some cases that color is not enough. In most cases even having a legend cannot help understand the true values of the phenomenon since the users cannot easily distinguish the limits of every color used in a pallet, so, it is important to give them the opportunity to click on a specific point on the map and to be able to see the numerical value of the surface height for this point. So, this section will be described the methodology on how the querying can be achieved.

In order to achieve the querying option for the application, the canvas element is used as in the case of the color-changing described in Section 4.2.3. The similarities with the coloring do not stop only on the use of the canvas element but also they are extended in the use of getImageData functionality of the canvas that results in the array of the RGBa values for every pixel. The difference, in this case, is that the whole image is not needed but only the pixel that the cursor is clicking on or hovering over. The similarities of this technique with the coloring technique can help the developer to implement both interactivity elements with minimum requirements for extra coding. The last step is to transform the RGBa values back to the height values interpolating between the minimum and maximum values of the data-set.

4.3 SYSTEM ARCHITECTURE

The last part of the methodology is related to the architecture of the application. As the architecture of a system is considered the description of all the components of the system, how they are connected and in general how the data flows within the system from the producer to the analyst and ultimately to the user. Every web application consists of two main components for their architecture described as server-side and client-side. The processes that are taking place on the server-side have the characteristic that does not use the user's system for the computation and in many cases includes processes that have already been pre-computed since these are computationally intensive. The client-side of the application uses for the computations, the user's system and since the developer cannot rely on the power of every user's system to have an efficient application, it is advisable to compute on the client-side mainly simple tasks to have a more responsive application. The most important task of the client is the hosting of the interface with which the user interacts.

Since the goal of this application is to visualize data, the first step is to acquire the data. For this thesis, the data source is the NASA that provides data acquired by satellite imagery and is encoded into a NetCDF formatted data-set (the data-sets used for this application will be described extensively in the next chapter). This application even though it uses NetCDF formatted data it is possible to also use other formats that include the time dimension. The next step is about the pre-processing of the original data-set to be transformed into a more usable format. The pre-processing face is possible due to the use of different software tools that can vary depending on the type of the original data. For this thesis, the goal of the pre-processing phase is to produce the video tiles organized into directories according to the tilling scheme used. Also in this stage, the creation of the extra directories for the zooming in time is taking place. All the processes described until this point are taking place in the so-called server-side.

The next component of every web-based application is the web server, a software that handles the requests and the responses between the server and the client. One example for the webserver related with this application is when the user is zooming in the map the client through the web server requests new tiles to be loaded from the server and the server will find the proper directory that the tiles are stored and it will return as a response the tiles to the client in order to be rendered.

When the tiles are rendered into the client they are stored as HTML elements that now the client can manipulate. One example of a process that takes place in the client is the handling of the z dimensions that actually means the color changing of the video. A second example that also takes place on the client is the querying capabilities described in the previous section. In the diagram that follows (Figure 4.2) the architecture of the system is described graphically.

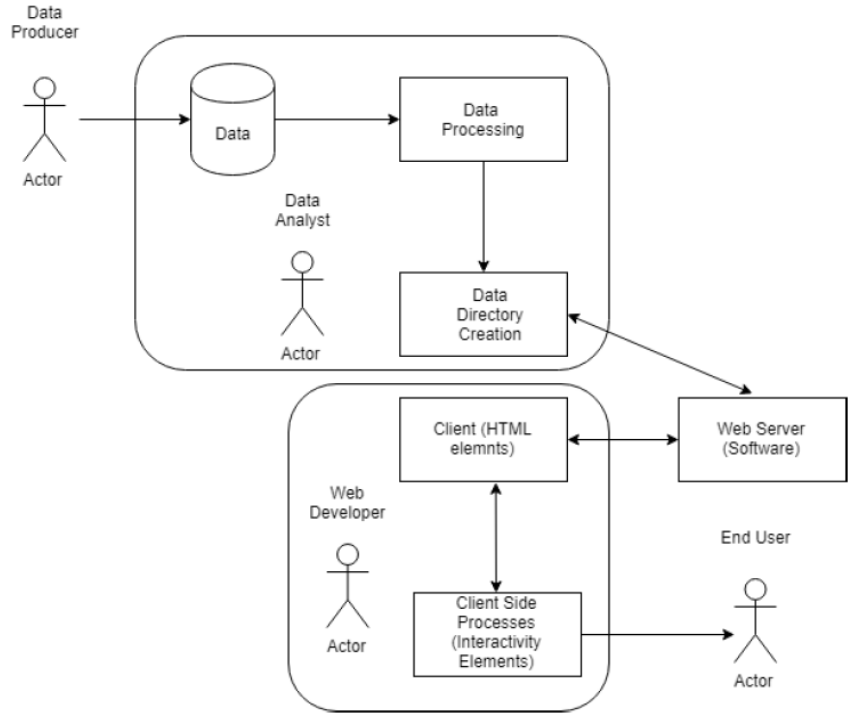


Figure 4.2: The application's system architecture proposal.

5 IMPLEMENTATION/RESULTS

In this chapter, there will be a detailed description of the implementation of the methodology that was proposed in Chapter 4. This chapter will be organized similarly as the Chapter 4 with the main difference being the addition of a first section related to the data-set that was is used for this application. Every other section will be divided into three sub-sections, with the first one being related to the tools used, the second one with the description of the details of the implementation and the last one will focus on the results.

5.1 DATA-SET USED

The first section is about the data-set used since it is not possible to talk about visualizations without firstly to refer to the data that are visualized. The data that are visualized is about the Sea Surface Height Anomaly of the earth's oceans. As it is described in the motivation chapter, the Sea Surface Height is a phenomenon that many researchers are interested in and there are many different data-sets related to this topic. Most related data-sets are using the term "Anomaly" in which, according to NASA, the main parameter of this product, represents the difference between the best estimate of the sea surface height and a mean sea surface. The sea surface height used was corrected for atmospheric effects (ionosphere, wet and dry troposphere), effects due to surface conditions (electromagnetic bias), and other contributions (ocean tides, pole tide, and inverse barometer) [1].

The main data-set that will be used is the "JPL MEaSUREs Gridded Sea Surface Height Anomalies Version 1609" provided by NASA. This data-set contains gridded Sea Surface Height Anomalies SSHA above a mean sea surface in a Network Common Data Form) (NetCDF) format with a temporal resolution of 5 days. The time – span is from 1992/10/01 and reaches until the present. In order to contain fully corrected heights, it is delayed from 1 to 3 months (NASA also provides a data-set without any delay, but this data-set is less accurate). The spatial resolution is of 0.17 degrees in both latitude and longitude that represents about 18.5km for the latitude and for the longitude this number varies depending on the latitude. The spatial resolution is not extremely high but since this project aims to create a visualization on a global scale this resolution is adequate for providing zooming capabilities of four zoom levels. This NASA product is a result of the combination of multiple sensors and these sensors are:

- JASON-1 / POSEIDON-2 (Nadir pointing Radar altimeter using C band (4 – 8 GHz) and Ku band (12–18 GHz) for measuring height above sea surface.)
- JASON-1 / JMR (measures water vapor along altimeter path to correct for pulse delay)
- TOPEX/POSEIDON / POSEIDON ALTIMETER (The NASA-built Nadir pointing Radar Altimeter using C band (5.3 GHz) and Ku band (13.6 GHz) for measuring height above sea surface)
- TOPEX/POSEIDON / TOPEX MICROWAVE RADIOMETER (operating at 18, 21, and 37 GHz was used to correct for atmospheric wet path delay.)

1 http://www.altimetry.info

- OSTM/Jason-2 / POSEIDON-3 (Nadir pointing Radar altimeter using C band and Ku band for measuring height above sea surface.)
- OSTM/Jason-2 / AMR (Advance Microwave Radiometer measures the 18.7 GHz, 23.8 GHz and 34.0 GHz sea surface microwave brightness temperatures.)
- Jason-1 Geodetic / POSEIDON-2 (Jason-1 was moved to a lower orbit and began its geodetic mission on 7 May 2012. The core payloads were switched ON on May 4th and after some POSEIDON-2 radar (PRF) adjustments, the mission was resumed on May 7th) In this new operational phase the Jason-1 mission is in a drifting geodetic orbit.)
- Jason-1 Geodetic / JMR (measures water vapor along altimeter path to correct for pulse delay)

(https://podaac.jpl.nasa.gov)

According to the latest announcement from NASA this data-set has been retired and will not be maintained in the future. The data-set will be still available for archive purposes but the access to it by the users is limited. NASA is providing other data-sets that are also measuring the SSHA in a NetCDF format and can be used in similar applications. ("OSTM GPS based orbit and SSHA OGDR", "SARAL Near-Real-Time Value-added Operational Geophysical Data Record Sea Surface Height Anomaly")

This data-set, as already has been mentioned, is in NetCDF format. According to the web page of the Unidata (the program responsible for the NetCDF software, standards development, updates, etc.) NetCDF is a set of interfaces for array-oriented data access and a freely-distributed collection of data access libraries for C, Fortran, C++, Java, and other languages. The NetCDF libraries support a machine-independent format for representing scientific data. Together, the interfaces, libraries, and format support the creation, access, and sharing of scientific data.[2]. The NetCDF is a binary format with the Classic and 64-bit Offset versions to be an international standard of OGC.

Even though NetCDF is a very efficient way to store array data, there are not many applications that can load a data-set in NetCDF format and to be able to visualize it right away. As a result, it is quite common to transform the data-set to an image format (e.g. tiff). This is also the case for the present application, and the programming language used for the transformation of the NetCDF data-set into images in TIFF format is the Python and more specific the library GDAL that is a translator library for raster and vector geospatial data formats that is released under an X/MIT style Open Source License by the Open Source Geospatial Foundation [3]. The resulted data-set is an image collection with every image represents the global data of Sea Surface Height Anomaly for every five days that is the temporal resolution of the original data-set. The newly created image collection then was uploaded to the Google Earth Engine that is an online software for the processing of spatial data (more information about this software in a later section).

For the last part of this section, there will be a reference on some statistics and facts about the data-set. The first interesting fact is about the file size of the original data-set (in the NetCDF format) that was a little over the 165 MB when the size of the image collection was more than 12 GB. The large difference in the file sizes indicates the compression capabilities of the NetCDF format over the simple image formats. Some other facts that can be interesting are that the number of images in the image collection is 1878 with the first image to refer to the data of SSHA of the 20/10/1992 while the last one refers to the data of the 18/06/2018. In order to have only full years of data the data that were used have as starting date 01/01/1993 and

2 https://www.unidata.ucar.edu/software/netcdf/
3 https://gdal.org/

Image: ssh_grids_v1609_1992100212

Image ID
users/fbaart/ssh_grids_v1609/ssh_grids_v1609_1992100212

Date
1992-10-02 12:00:00 UTC

Size
6.55MB

Last modified
2018-07-11 06:51:30 UTC

Property (1)

Band (1)

Index	*Name*	*Type*	*Dimensions*
0	b1	float	2160x960 px

Figure 5.1: The first image of the data-set as described in the Google Earth Engine

as the last one 01/01/2018. Finally, as it is visible on the Figure 5.1 the dimensions of every image are 2160x960 pixels while the average file size per image is 6.5 MB.

5.2 ANIMATION

In the same section in the methodology chapter, there was an extensive reference on the three more common animation techniques for the web, a comparison between them and in the end the decision of using Videos as the most efficient and effective animation technique in the context of this thesis. The decision making about the animations did not stop there since the video can have many different codecs and containers with different positives and negatives. In this section, there will be a description of how to create the videos, what tools will be used and a comparison between the different options considering the file size, the encoding speed, and the quality.

5.2.1 Tools Used

A video is a series of images (frames) encoded together to be presented one after the other, in a specific order, in a specified time interval. The images that were used to produce the videos were a product of the Google Earth Engine software that will be described Section 5.3. In this section, the focus will be on the encoding of the videos and as a result, there will a description of the FFmpeg that is a complete, cross-platform solution to record, convert and stream audio and video [4]. FFmpeg is designed for command-line-based processing of video and audio files and widely used for format transcoding, basic editing, video scaling, video post-production effects, and standards compliance. The FFmpeg project started in 2000 and since then it has become part of the workflow of many software, and its libraries are used in the core of many media players. Even though the project has created its codecs and containers, it also supports most of (if not all) the rest of the codecs and containers for both audio and video. In the context of this thesis, there is no audio to encode and as a result, only the video formats are of interest. All six of the different formats that were compared in the methodology part are supported by the FFmpeg even though the main focus of the analysis will be about the mp4/H.264

4 https://ffmpeg.org

and the WebM/VP9 that are the most commonly used and as it was concluded in Section 4.1 the best options for this thesis. Finally, FFmpeg provides a large number of filters for many different purposes and a detailed wiki [5] in which the user can find the command that he/she needs.

Although FFmpeg is the main tool used in the video creation process it is important to also refer to the subprocess module of Python that is used to run new applications or programs through Python code by creating new processes. It also helps to obtain the input/output/error pipes as well as the exit codes of various commands. With Python, it is possible to handle a large number of images and directories that work as the input for the video creation, but this process (the video creation) should be repeated multiple times and with the simple use of FFmpeg through the command line is not an easy task. For this reason, the subprocess module handles all the repetitive processes and calls the FFmpeg only for the encoding of the videos.

5.2.2 Video Creation Process

In this section, there will be only referenced on the video encoding part and not on the organization of the directories and the images where produced since this is a part that will be described in the tilling scheme section.

In order to produce a video from images the first step is to have all the images stored in a folder in the correct order (Figure 5.3). The images that will be used as the frames of the video are stored into directories with identical structure and with identical naming with every directory representing a frame of the video. The only difference between the directories is the out-most folder that is named with an ascending number(Figure 5.2).

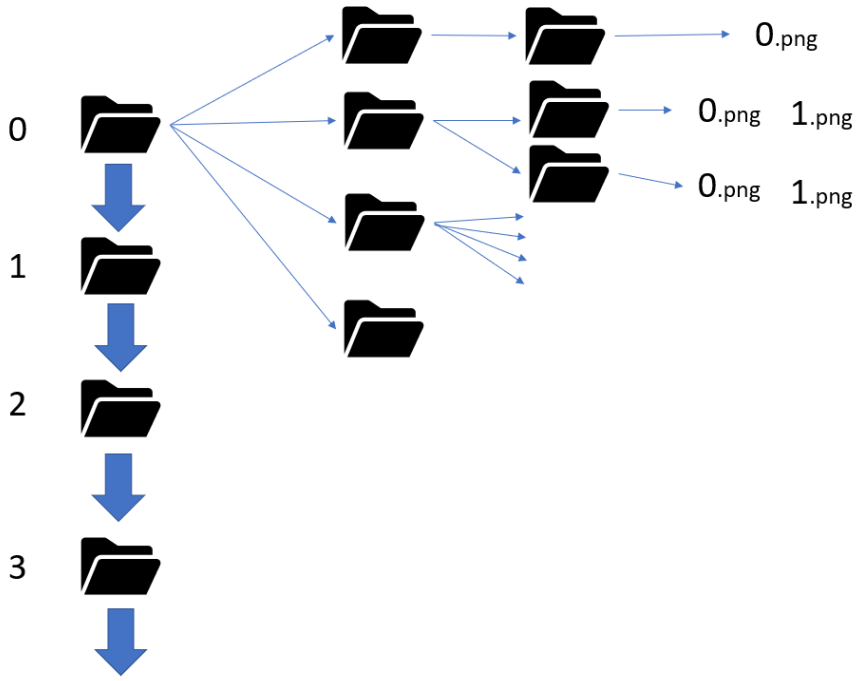


Figure 5.2: A diagram that shows the organizations of the images

The goal is to end up with just one directory similar with the original with the only difference being that in the position of the images (0.png, 1.png, etc in Figure 5.2) there will be folders with the same name and inside these folders there will be the images enumerated according to the out-most folder (Figure 5.3).

5 https://trac.ffmpeg.org/wiki

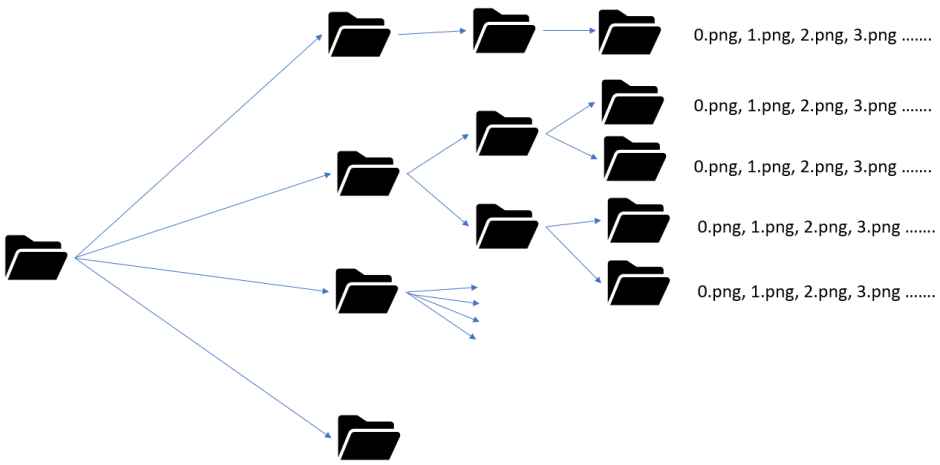


Figure 5.3: A diagram that shows the directory right before the creation of the videos

After the creation of the proper directory with the images (Figure 5.3) it is the time for the subprocess and the FFmpeg to produce the videos. The use of the subprocess is really simple since the main input is the FFmpeg command as a string. As soon as the string is formulated the subprocess is called with the argument being the formulated sting.

Off-course, the tricky part is to formulate the proper FFmpeg command and since it does not run in the Python interpreter there is no error message if the command is incorrect and the only way to see if something is wrong is by checking the output. The FFmpeg command has a certain number of arguments that need to be correctly executed.

- Call FFmpeg (ffmpeg.exe)
- Specify the framerate (-framerate)
- Specify the input that in our case is the folder that contains the images (-i)
- Specify the video codec that will be used in order to encode the video (-codec:v)
- Specify the quality that is described by a number (crf) between 0 and 63 with the smaller number meaning higher quality. The typical values are between 15 and 35. Values outside this range rarely have a reason to be used with the really small values to be related to the lossless encoding that results in really high-quality videos but also really big in size something that leads to limited usability. (-crf)
- In the case of this thesis is important to respecify the framerate to avoid corrupted videos (-r)
- The last argument that needs to be specified is the output file with the correct indication for the file container (eg. mp4 or webm) and the proper name that should be the same as the one in the original directories with images.

One last detail that needs to be taken care of is the removal of the folder with the images from which the video was created. A command in FFmpeg that can be used looks like:

'ffmpeg.exe -framerate 10 -i videoTilesPerMonth$_m$p4/0/0/%03d.png $-$ codec : vlibx264 $-$ crf22 $-$ r10videoTilesPerMonth$_m$p4/0/0/0.mp4'

5.2.3 Animation Results

In this last part some numbers and facts descriptive of the video creation process described above.

The first interesting number is the file size of the images that were used as input. In the case that the data-set is aggregated with per month data the file size is 281 MB while the videos were more than ten times smaller in size, something that indicates how efficient the video encoding can be. More specific the file size for the two formats that were selected to be used in the application mp4/H.264 and WebM/VP9, for framerate of 15 frames per second (fps) and crf value equal with 22, is 25 MB and 20.7 MB respectively. The file size for both formats is really small but when the zoom levels will be increased into more than four the file sizes will be increased exponential and the improved efficiency of the VP9 codec will be even more evident. The improved efficiency of the VP9 codec comes at a cost since the time that needs in order to encode the videos is more than three times longer than that of the H.264 (the actual number is irrelevant in this case since it depends on the power of the processor of the computer). The decoding speed would be also interesting to be measured but the measurement has two main problems. The first is the same as in the case of the encoding and is the fact that it is depended on the hardware and software that the user uses. The second problem is that there is not an easy way to measure the decoding time for an HTML5 video element. The final numbers that need to be mentioned are related to the quality of each encoding and these numbers are the bitrate of each video with higher value meaning higher quality. The bitrate of the videos is not always constant and varies within the different sections of the video (Through the FFmpeg it is possible it set a constant bitrate but it is time-consuming and for many cases does not improve the final result significantly). Returning to the comparison between the VP9 and the H.264 the bitrate is around 330 kbps and 270 kbps respectively demonstrating once again the superiority of the former in comparison to the latter. The comparison described above can be summarized in Table 5.1

Comparison Element	**WebM/VP9**	**mp4/H.264**
Example File size (KB)	515	863
Bitrate (kbps)	330	270
Example Encoding Time (s)	1467	427

Table 5.1: Comparison between the WebM/VP9 and the mp4/H.264

5.3 HANDLING THE X AND Y COORDINATES

This section is about the first of the interactivity elements that were analyzed in the methodology chapter. The X and Y coordinates are linked with the zooming and panning capabilities of the interactive maps. The most commonly used solution to this problem is the introduction of tiles and since the Google Maps many different tilling schemes have been introduced but in the end, this technology has been standardized around the implementation of Google Maps. The Typical size of a tile is 256x256 pixel while some newer approaches introduce the 512x512 pixel tiles (MapBox) due to the improved network efficiency and for the fact that the screens are

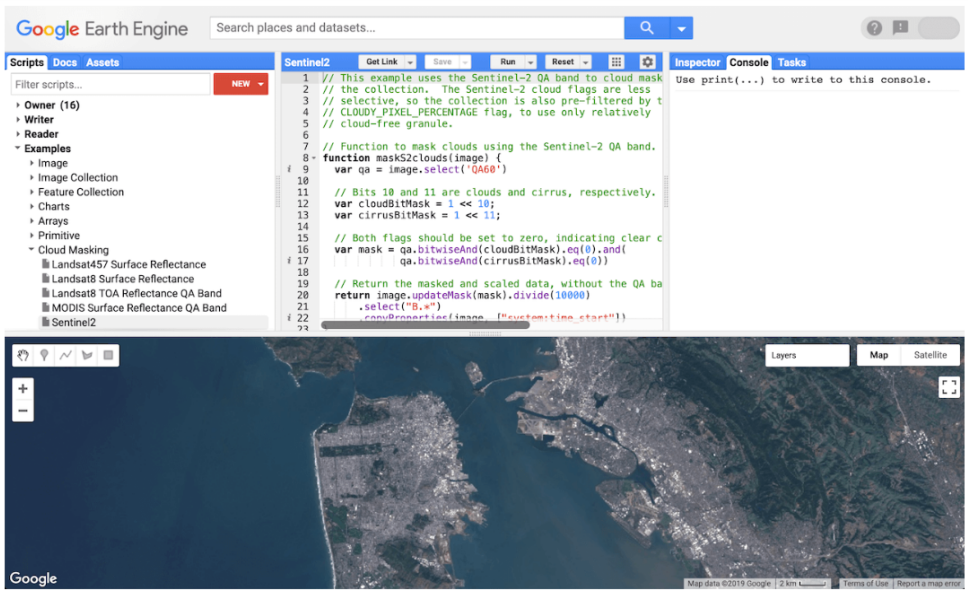


Figure 5.4: The Google Earth Engine Code Editor

getting higher resolution with every year. The only other good example of video tiles comes from the Carnegie Mellon University CREATE Lab's Time Machine library that even though it is the only one that is operational and reliable enough is not an elegant solution to the problem (see section 4.2.1). This section will begin with the analysis of the tools that were used on one hand for the creation of the tiles and on the other hand for the implementation on the web. The next sub-section will be about the implementation and the last one about some interesting results related to the tilling.

5.3.1 Tools Used

The first tool that will be described is one that has already been referred to in the previous sections of this chapter and it is non-other than the Google Earth Engine. In the website of the Google Earth Engine [6] is explained that Earth Engine is a platform for scientific analysis and visualization of geospatial datasets, for academic, non-profit, business and government users, while it hosts satellite imagery and stores it in a public data archive that includes historical earth images going back more than forty years. The images, ingested daily, are then made available for global-scale data mining. One last characteristic of the Earth Engine is that provides APIs and other tools to enable the analysis of large data-sets. There are two APIS for the Earth Engine. The first one is based on the JavaScript programming language and is accompanied by a web-based code editor and the second one is based on the Python programming language. Due to the web-based code editor of the JavaScript API is used more widely this was the one that was used in the context of this thesis project.

In the code editor (Figure 5.4) and on the left side it is visible the part in which the user can find the scripts that he/she has created, the docs tab in which the documentation for every function that the API contains and lastly the assets tab in which the user can upload its data-sets. In the middle is the main body of the code editor in which the user can create and modify his/her scripts. In this point it is important to highlight the fact that even though the code editor is using the JavaScript programming language the API provides a large number of extra functions focused in the analysis of spatial data and also it has a different logic in

6 https://earthengine.google.com/faq/

programming compared with the simple JavaScript and is closer to the logic used in the big data analysis (eg. instead of using "for" loops it uses "map"). Concluding with the description of the script editor, on the top of the main body there is a search bar in order to look for the data-set that you need (among the satellite imagery that the Earth Engine hosts), on the right side there is the panel for the outputs of the scripts (if it is numerical or a file that can be stored in the user's drive or cloud) and on the lower part there is a map that is displayed the visualized outputs of the analysis.

The second tool that was briefly used in the first stages of the implementation is the Mapbox API. Mapbox is the location data platform for mobile and web applications. It provides building blocks to add location features like maps, search, and navigation into any experience you create [7]. The mapbox provides also a web-based editor of spatial data but in this case, is more similar to a GISsoftware in which the analysis is not based on scripts that the user creates but rather the functions have been encoded into buttons of an interface. Mapbox also provides two JavaScript libraries (mapbox js and mapbox gl) for creating and handling maps on the web. the mapbox js will handle basic tasks and is comparable with the Leaflet and the Google Maps API. Finally, the mapbox gl is a library that uses WebGL to render interactive maps.

The last tool used is Leaflet that is the leading open-source JavaScript library for mobile-friendly interactive maps. It is lightweight and according to the developers it is designed with simplicity, performance and usability in mind three characteristics that were appreciated in this thesis project and was selected in the end for the application instead of the mapbox js.

5.3.2 Implementation (Tile Creation)

sec:anime-r

- (2*x, 2*y)
- (2*x, 2*y+1)
- (2*x+1, 2*y)
- (2*x+1, 2*y+1)

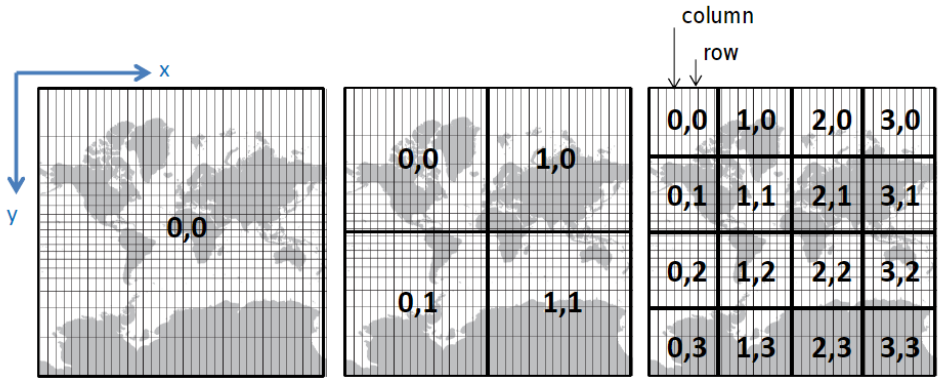


Figure 5.5: Tile numbering for the Google maps tilling scheme [Stefanakis, 2017]

In Figure 5.5 are visible the tile coordinates that needed in order to produce the bigger tile and how the above equations are applied (eg. for the tile in zoom level 1 with coordinates (1,1) the tiles that are needed are: (2,2),(3,2),(2,3),(3,3)). The merge of the 256x256 tiles to produce the 512x512 tiles where possible through python and more specific the pillow library.

7 mapbox.com/about/company/

5.3.3 Implementation (Web)

After the creation of the tiles on the server is the step that these tiles are loaded on the browser for the user to interact with. The handling of the tiles is a result of the use of the leaflet library and more specifically the methods of ”tileLayer” and ”gridLayer”. In this project the gridLayer method was used since it is more generic and easier to manipulate. The gridLayer contains many different options that can be changed from the developers according to their needs. Some of the more important options that can be changed are:

- tile size
- minimum zoom
- maximum zoom
- bounds (the boundaries (latitude/longitude) in which the tiles will load)
- keepbuffer (how many rows or columns of tiles will be kept before unloading while panning)

This method also contains the function createTile that is the one that can be manipulated in order to add for example the video element instead of the images or a canvas element even a ”div” with multiple HTML elements. In order to set up the createTile function two things are actually needed:

- to create the HTML element (eg. document.createElement('video'))
- and to provide the source of the element

While specifying the source of the tiles need to be in a format that can easily be used from the browser in to find in the server the correct file as fast as possible. In the case of the Google Maps tilling scheme, the source (that is actually a path in the server) looks like:

”videoTiles/'+ coords.z + '/' + coords.x + '/' + coords.y + '.mp4”

where the coord.z refers to the zoom level.

All the process that was described in the previous sub-section leads to this simple path that can be read efficiently by the browser and the leaflet library that is responsible for the tilling. The map pane is where the tiles are added and while the user is panning the translation of the center of the pane in comparison with the original position, in pixels, is measured in order to specify which tiles need to be added or removed from the pane. The last part is to add this new ”gridLayer” that was created into a map element as part of its active layers.

With a rather simple function of the Leaflet library, it is possible to load the video tiles on the browser. Unfortunately, it is almost impossible to load all the tiles at exactly the same time while zooming and of course, there is no need for discussion on the topic while panning. Since the videos are loading at different times they are not possible to render the same frame at the same time and the result is a group of videos that play randomly resulting in a confusing and unpleasant experience. For this reason, there is the need of defining a way to synchronize the videos in order all the videos in the group to render the same (or almost the same) frame leading to a smooth result that in perfect conditions looks like that there is only one video instead of multiple.

The synchronization of the videos especially when they load at different times and almost always new videos are rendered is not an easy task. The reasons why this is difficult are multiple and can begin from the performance of each device that can struggle to render multiple videos at the same time, it can also be related

with the fact that the browsers do not follow the frame rate of the videos to the millisecond since they take into consideration the refresh rate of the screen too, and can reach t the fact that the main libraries that could help with the synchronization (popcorn.js, mediagroup.js) have been discontinued from the browsers.

Many different approaches were implemented during the course of this project but most of them failed. The one that is considered as closer to the ideal result is the so-called timing object. The timing object is a very simple object, essentially an advanced stopwatch. If started, its value changes predictably in time, until at some point later, it is paused, or perhaps reset. It may be queried for its value at any time. Such deterministic behavior is required for reliable distributed synchronization. In terms of implementation, the timing object is a fairly thin wrapping around a monotonic system clock (integration with online timing resources adds complexity). The precision of the timing object is limited by the underlying system clock. [8]. The proposed timing model is one where timed components are connected to timing objects, thereby accepting the timing object as director of timed operation. In other words, the timing object is the master, and the timed component is the slave.

In more practical matters in order to use the timing object in the application, it needed just to set up a timing object:

```
var to = new TIMINGSRC.TimingObject();
```

It is important here to add that the above expression is the simplest format and it is possible for the developer to intervene and to set different parameters like the time range [0, 30] for the timing object and to enable the loop option on the mediaSync(loop: true). and then to utilize the 'tileload' function of Leaflet in order to synchronize every new tile (video element) with the timing object:

```
var sync = MCorp.mediaSync(tile, to);
```

In order to optimize the synchronization process the "sync" is triggered when the tile has loaded (tile.onloadeddata functionality of JavaScript) as well as due to the fact that some times the video where not "sync"ing with the timing object, a keep in sync function was created that was triggered with a time interval and it was looking in the Document Object Model (DOM) for all the video elements and forced them to "sync".

5.3.4 Tilling Scheme Results

The results part of the chapter will have two parts like in the case of the implementation one for the tilling scheme and the second one for the web.

In the first part, the main results are the maps that were the output from the Earth Engine. The comparison in file size can be between the small tiles 256x256 and the bigger ones 512x512 that the bigger tiles end up being slightly bigger than the small ones. In numbers, the small tiles folder size is 281 MB while the big tiles folder is 283 MB. The interesting part is when these images are transformed into video tiles. In this case, the folder the videos were created with the exact same parameters and encoding and with the only difference being the tile size and the different directory structure. The folder sizes are reverse with the 512x512 being smaller at 27.5 MB while the 256x256 being 31.5 MB. This difference is more important in this case (in comparison with the case of the images) because on one hand the 3 MB in 30 is 10% difference that is considered especially when extra zooming levels are added, and on the other hand the true storage requirements in an application like this one are related with the video tiles since these are the ones that are needed.

8 https://webtiming.github.io/timingobject/the-timing-object

Figure 5.6: The interface of the application with the videos synchronized (zoom level 1).

In Figure 5.6 is shown the explanation of the first generic interface that it was created for the application. There are two categories main of categories of buttons: the video related (start, pause, reset) and the map related (zoom in, zoom out). For the panning of the map, like in most similar applications, the "mouse" of the computer is required with which the user clicks (left click) on the map and without releasing the button move the mouse. In the images of the description of the interface, the map is tiled but due to the synchronization, the edges of the tiles are not detectable.

In figures Figure 5.7 and Figure 5.8 due to the enabling of the videos controls the tiles are visible for both tile sizes (256x256 pixels and 512x512 pixels).

The second part of the results is about the web and how the video tiles can perform depending on the tile size. As some general remarks can be noticed that four smaller videos are bigger in size than one larger. For zoom level 1, the four small videos have a combined size of 2.9 MB while the larger one has 2.5 MB a considerable difference for the load on the network. The other big difference is in the synchronization that is not something that it is easily measured but it is rather straight forward the fact that the fewer the videos the easier is for the browser to handle them. Another factor that helps the synchronization is the creation of videos with a rather low frame rate (eg. 10fps) that increases the time interval between the frames and as a results the "timeUpdate" function that is triggered every 15 to

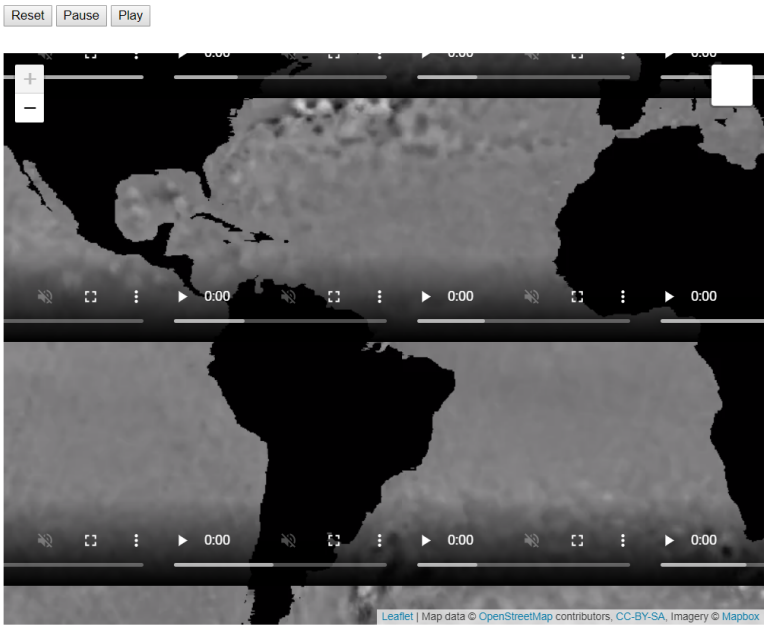


Figure 5.7: The interface of the application for256x256 pixels tiles (zoom level 1)

250ms want to miss a frame and the possibilities for displaying the "wrong" frame will be reduced.

In order to present some numerical results, some parameters need to be defined. The first parameter is about the fact that the web users will pay attention to a web page for average no more than 15 seconds that means that whichever element is considered important for the web application need to load faster than 15 seconds to keep the attention of the user. A second parameter that needs to be defined is that, as has already been mentioned the process that takes place on the client is using the user's systems and as a result these numbers are subjective. Finally, the parameters that are of interest and would be measured are:

- Loading time of zoom in (from level 1 to level 2)
- Loading time of zoom out (from level 3 to level 2)
- Loading time of panning one row (zoom level 3)

The comparison will take place between the 256x256 pixels version and the 512x512 pixels one. The comparison of mp4 and WebM implementations even though they were considered to be added in the results but the resulting numbers were always too close and consequently it was decided that adding one of them is enough. Before the demonstration of the results, it is important to specify the system's hardware details since the results heavily rely upon them.

- Intel Core I7-4810MQ CPU
- 16GB Random Access Memory (RAM) ddr3 1600MHz
- 256GB Solid State Disk (SSD)
- 60Hz Screen 1920x1080 pixels resolution

The test results demonstrate that the time needed to perform the main task of the application is rather small and all of them are less than 2 seconds. It is clear that even though the differences are small the bigger tiles (512) are faster in rendering the tiles than the smaller ones (256).

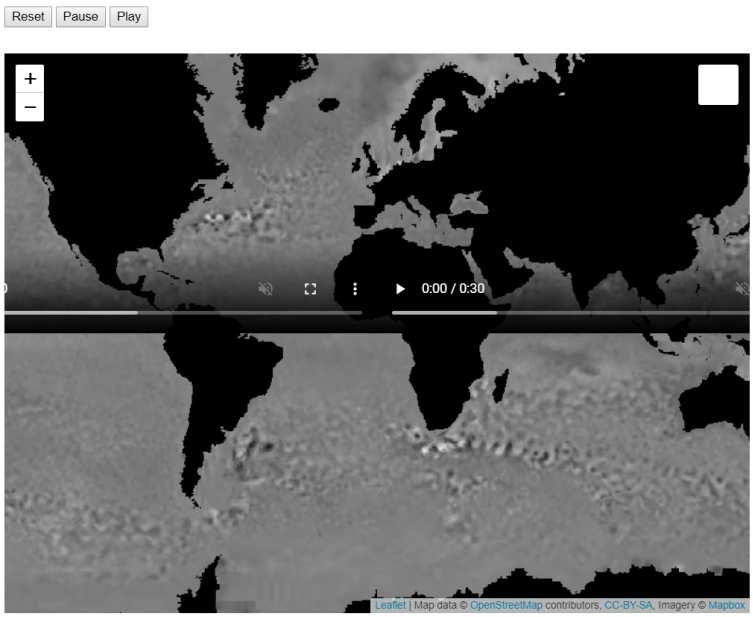


Figure 5.8: The interface of the application for 512x512 pixels tiles (zoom level 1)

Comparison Element	**256**	**512**
Loading time of zoom in (from level 1 to level 2)	1.8s	1.5s
Loading time of zoom out (from level 3 to level 2)	1.5s	1.1s
Loading time of panning one row (zoom level 3)	1.9s	1.1s

Table 5.2: Comparison between the 256x256 pixels version of the application and the 512x512 pixels version of the application

5.4 HANDLING THE Z COORDINATE

It is not possible to write about the Sea Surface Height and not to talk about the representation of the height in this visualization. From the introduction, it is clear that the height will be represented with different colors depending on the values. Also, since not all users can interpret the colors in the same way it was proposed to give to the users the possibility to select the pallet that they prefer. In section 4.2.2 it was analyzed the two different methodologies that can be followed as well as a short description of the APIs that can be used. This section will have the same structure as the previous ones and will start with the tools used part, followed by the implementation and concluded with the results.

5.4.1 Tools Used

The tools used are referring mainly to the two APIs that were mentioned in section 4.2.2 and these are the Canvas API and the WebGL API. The Canvas API is really simple to use and among other things, it can be used for animation, game graphics, data visualization, photo manipulation, and real-time video processing. Since there are not many things to talk about the Canvas API there will be a reference on the WebGL that can be considered as an extremely good alternative especially if the performance is the goal for the web application.

The most important component of WebGL is the shaders. The shaders are programs, written using the OpenGL ES Shading Language (GLSL), that take information about the vertices that make up a shape and generates the data needed to render the pixels onto the screen: namely, the positions of the pixels and their colors. Two shader functions run when drawing WebGL content: the vertex shader that is responsible for the shapes that consist of vertices and the fragment shader that is responsible for the color of the pixels in the shape. The vertex shader and the fragment shader are written in GLSL and pass the text of the code into WebGL to be compiled for execution on the Graphics processing unit (GPU). Together, a set of vertex and fragment shaders is called a shader program. In order to implement the color changing with WebGL canvas is needed and the video element. In the fragment shaders, it is also possible to use video as a texture and this mode is used for this methodology.

The last tool/library that was used is the D3.js that is a JavaScript library for visualizing data using web standards. D3 helps you bring data to life using SVG, Canvas, and HTML. "D3 combines powerful visualization and interaction techniques with a data-driven approach to DOM manipulation, giving you the full capabilities of modern browsers and the freedom to design the right visual interface for your data" [9].

5.4.2 Implementation

Until now in the implementation parts the discussion has been revolved around the video elements and how to produce, load and synchronize them. After the solution of these problems, it is time to use the videos as inputs to implement the coloring part of the application.

The first task of the implementation is to add the two extra HTML elements while creating the tiles. These two elements are canvases and since these two canvases are bound together with the video element it is necessary to create a div element and then to append the video and canvas elements as "children". In this next part is where the d3.js is used to produce the color maps.

- Select the pallet (eg. Spectral)
- Set the interpolation within the colors of the pallet
- Define the type of the scale (eg. sequential)
- Produce the new colors with d3.rgb

Then starts the process that have been already referenced with the drawing of the video frame in the first canas element (ctx1.drawImage(video)), get the image data from the first canvas (ctx1.getImageData) and save them into an array. With a for loop traverse the array every four elements (red, green, blue, alpha) that corresponds in every pixel's values and with the colors created from the d3.rgb (color = this.rgbs) set the new color for every pixel:

data[i] = color.r data[i+1] = color.g data[i+2] = color.b

The last step is take the changed array with the new colors and to draw it in the second canvas (ctx2.putImageData) that is the only element that is actually visible to the users (for both the video and the first canvas the "display" is set to "none").

Since it has been discussed, the color pallets that are most commonly used in the visualizations of the Sea Surface Height are having blue as the low values and red

9 https://d3js.org/

as high values. This color pallet can be changed a little using for example green for low values or brown for high values. It is also possible to use only one color and that being the blue with almost white being for the low values and deep blue for the high values.

5.4.3 Results

Just like in the case of the Section 5.3 the Figure 5.9 is presenting the two extra buttons that where added in order to change the color pallets. In Figure 5.9 is also visible the color pallet instead of just the grayscale that it was used in Figure 5.6. In Figure 5.10 is used the RdBu color pallet.

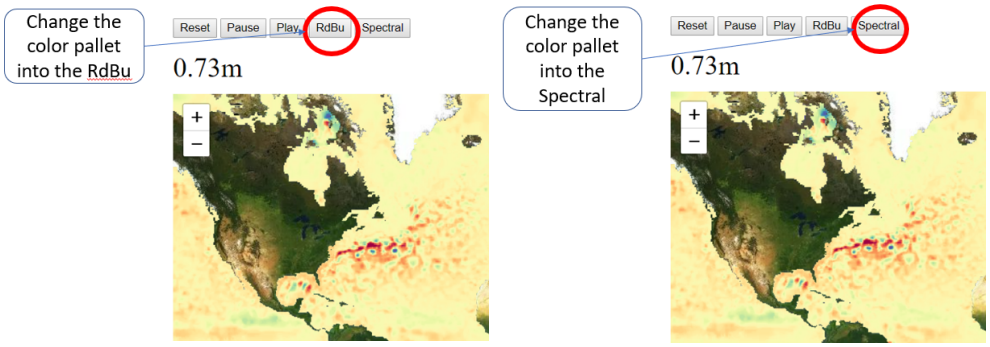


Figure 5.9: The interface of the application with the videos synchronized and the addition of the coloring buttons (zoom level 1). The pallet that is used is the spectral

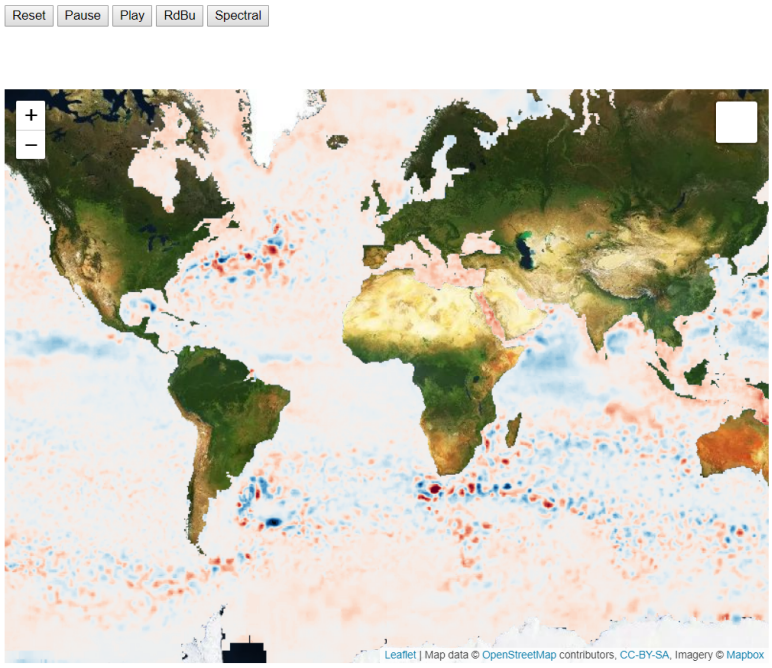


Figure 5.10: The interface of the application with the color pallet set to RdBu (zoom level 1)

The comparison between the different video tile formats will have the same characteristics as the one on Section 5.3. The measuring of the color changing process was also considered but every time the change was taking place almost instantaneous and as a result there is no reason to measure it.

The numbers in Table 5.3 even though they are subjective it show a general pattern, that was present also in Table 5.2, the performance of the application has a

Comparison Element	256	512
Loading time of zoom in (from level 1 to level 2)	1.928s	1.5s
Loading time of zoom out (from level 3 to level 2)	3.9s	2.8s
Loading time of panning one row (zoom level 3)	2.7s	1.6s

Table 5.3: Comparison between the 256x256 pixels version of the application and the 512x512 pixels version of the application while adding the coloring process in the application.

considerable difference in the performance between the small and the big tiles with the bigger ones being more efficient. The difference in performance is also visible when comparing the grayscale tiles and the colored one with the second ones being slower. Also, it is important to notice that the numbers in seconds for every performed task is considerably smaller than the 15 seconds that is considered the maximum for the user to pay attention. Even though, this technique is the slowest between the alternatives it is fast enough for this application. In the case that more videos should be rendered then maybe the use of WebGL API will be more efficient (even though more videos can cause problems with the previous part of the application and is not advisable to use).

5.5 HANDLING THE TIME DIMENSION (TEMPORAL SCALE)

The handling of time is considered as one of the key characteristics of this thesis project. Using a video as an input a major factor in presenting to the user the passing of time. But the concept of handling the time is mainly related to the term ”zooming in time”. As it has already been described how the zooming in time works (section 4.2.3) the focus will be on how to create the three ”zoom” levels: per year, per month and per five days. The zooming in time is selected to be implemented by creating different video-sets for every zoom level. So there is a directory in the server that it contains videos that each frame represents the mean value of Sea Surface Height Anomaly for a year, another directory with videos that each frame represents the mean value of Sea Surface Height Anomaly of a month and a third directory that contains videos with non aggregated values for every frame.

This section will have a different structure than the previous ones of the chapter since it is heavily related to Section 5.4. The part of the tools used is omitted since all the directories have been created from the Google Earth Engine with the only difference being that before the exporting map one mar filter is applied. This filter is the calendar range (ee.Filter.calendarRange). Every image of the image collection is described by the date that it was created and with the calendar range function, it is possible to select all the images that were created each month or each year. Then there are numerous aggregation options like the mean, the median the sum, etc. In the context of this thesis, it was used the mean value. A problem that came up during the creation of the ”zoom” levels was the fact that the data-set was too big (1825 images) to export all the maps for every 5 days case. The problems were found in the Earth Engine API that could not compute the maps. For this reason, it was decided to cut the data-set into 5 parts one for every 5 years. Each part of the data-set will contain 365 images that are slightly bigger than the monthly one that has 300 images. Another reason that it was decided to cut the data-set into smaller parts was the fact that the resulted video for the 1825 frames it would have been too big to be loaded and rendered efficiently on the browser.

The new issue that needs to be solved is that to give to the user the opportunity to inspect the whole data-set the videos from each part need to play one after the other. This is possible in JavaScript since the video element will be still only one but the source of the video will change. The new code will be added in the create tile part since it is the one that creates the video elements. So, the sources (file paths) of the videos will be stored into an array and the timing object will be used to as trigger to change the source. More specific there will be a function that will be triggered on the time update and it will check if the position of the timing object will reach the duration of the video. If the duration of the video is reached an index will be increased by one and will point to the index of the array with the source of the next part. Finally when the index will have reached the length of the array then it will be re-sated and will start over with the first source for the videos.

The technique that implements the sequential play of the video-sets works fluently enough even for as many as 16 videos that are loading at the same time. For the user when the sources are changing it will look as small lag that normally it is less than a second and looks similar as in the case that the videos play in a loop.

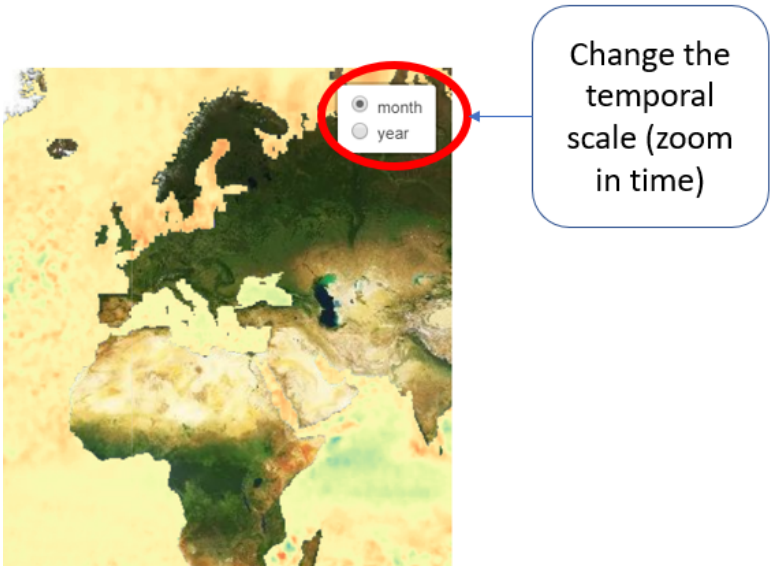


Figure 5.11: The interface of the application with the zoom in time functionality

The Figure 5.11 is showing the interactivity element of zooming in time as it is implemented in the interface.

5.6 QUERYING CAPABILITIES

The querying capabilities in an application is really important can allow the user to inspect the visualized phenomenon in more depth. The color pallets that are being used can give a general idea of how a phenomenon changes but every quantitative phenomenon is better to also be represented with its numerical value. The combination of the numerical value and the color can help the users understand how a phenomenon change and which change is considered significant and which not so.

In order to achieve the querying capabilities, the use of the Canvas API and the d3.js library is required. The ”tools used” part will be omitted since both tools have already been explained. This technique will start from the point that the color-changing process has finished (Section 5.4) with the second canvas element being colored with the new pallet. Firstly, an event listener is being created that can be even the move over or the click that triggers a coloring picking function. Then

according to the ”mouse” position the pair of coordinates of the pixel underneath will be extracted, the data will be extracted from the canvas (ctx2.getImageData) but this time not for the whole image but only the specific pixel. Then, the average value of the red, green, blue of the pixel will be extracted and since this average can have values between 0 and 255 it is easy to determine the value as the percentage of the maximum. The last step is to set up an interpolation between the minimum and maximum values of the original data-set (-1,1) with d3.js (d3.interpolateNumber(-1, 1)) and access the real value.

This method for querying the map is rather straight forward to implement and also utilizes the tools and elements that have already been used in previous steps of the implementation of the application.

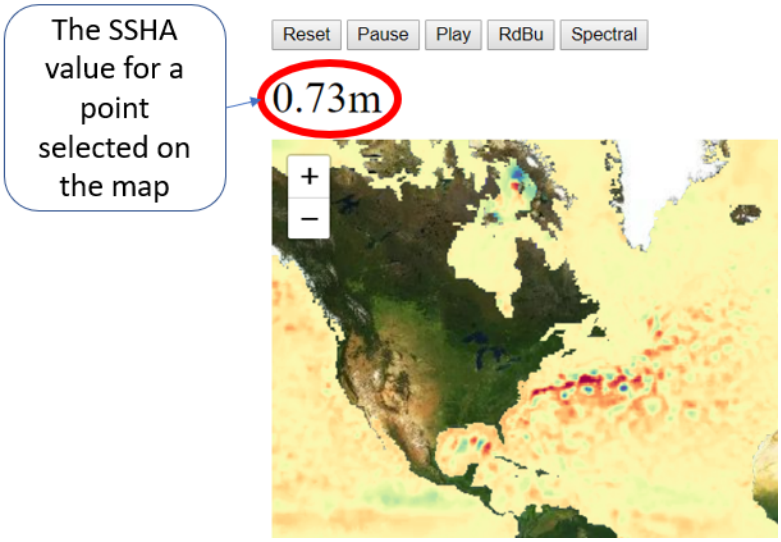


Figure 5.12: The interface of the application with the querying capabilities added

5.7 SYSTEM ARCHITECTURE

The system architecture is the one that describes the flow of the data, from the producer to the user. In Section 4.3 it was described a somewhat general system architecture scheme and at this point, it is the time to be explained more in-depth. The main parts of the architecture of a web-based application are the ”server-side” and the ”client-side”.

The analysis will start from the server-side and more specific with the downloading of the original data-set from NASA named “JPL MEaSUREs Gridded Sea Surface Height Anomalies Version 1609”. The first process is about the transformation of the data-set from NetCDF into tiff format (images), creating in this way a new data-set. The new data-set is uploaded into the Earth Engine to pass through the second face of processing (creating ”maps” for every zoom level in space and time). This new process creates multiple image directories (seven in number) that will be used as input in the final process that takes place on the server. This last process is about the use of the images created in the previous step to create videos with the use of FFmpeg through Python programming. All the processes used until this point are meant to be used only once and all the intermediate data-sets that were created are of no real use in the application. The goal from the start was to create the seven directories that contain the videos that will be used for the visualization on the browser.

The web server that was used is the Apache Tomcat. this server will handle the requests from the client (browser) to the server and it will respond with the

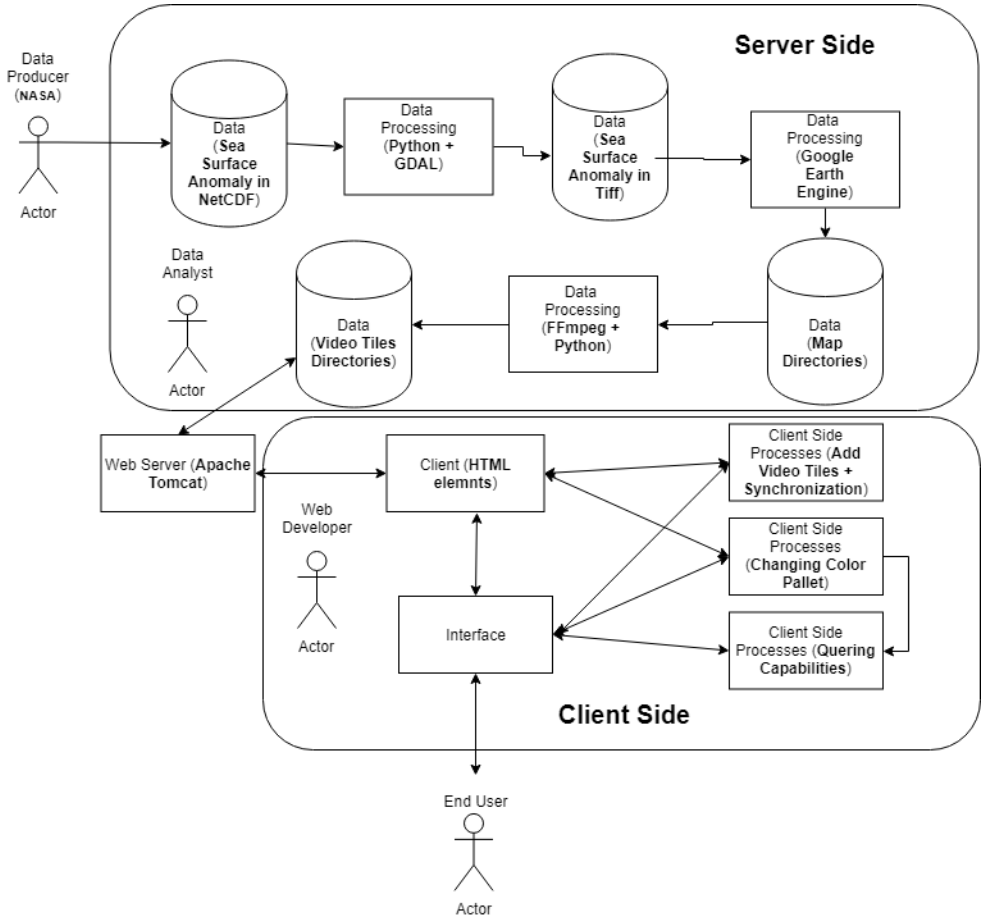


Figure 5.13: The architecture of the application

appropriate files (for this application the responses are the video tiles). Applications that are handling spatial information are also using the Geoserver. The Geoserver implements the OGC standards but there is not one that can handle the video tiles used in this application.

The other big part of the architecture is referring to the client-side of the application. A client can handle several elements specified in HTML. In the case of this application, the HTML elements are the video tiles that are loaded from the server through the web server and the canvas element that will be used to achieve the coloring of the maps and the querying. The client is also the one that renders the interface of the application with which the user can interact with. The client is not static just rendering HTML elements to the users but with the help of JavaScript and its libraries, it is also possible to perform some processes that can be triggered either automatically from the browser itself or by the user that can press a button or use a slider, etc. The efficiency of the processes that are computed on a browser is relying on the user's system and as a result, the experience can vary among them. The first process that is taking place on the client and it is probably the most important one is the decision on which tiles should be loaded and in which position should be placed. The second process can also be included in the first one and it is the one that helps with the synchronization of the video tiles. The last process that is taking place on the client is the coloring and the querying that, like in the case with the first two processes, are connected and the second cannot exist without the first one.

The architecture section is the last one of the implementation/results chapter and this means that the prototype has reached its final form. The application can be found online on *https://giorgosdimo.github.io/MSc-Thesis/* for anyone that wants to

check all the above-mentioned characteristics live. This application is not perfect. The interface is generic and the performance needs enhancement but the issues and their possible solutions are part of the discussion in the that follows.

6 CONCLUSIONS AND FUTURE WORK

This chapter answers the research questions that were posed in the introduction chapter since they reflect the goals of this thesis. Starting from the sub-questions first and, in this way, covering all the different parts of the application will help to gather the information required to answer the main research question and actually how close the final result is to the expected one. This chapter will also include a section that will discuss the contributions of this thesis to the field of Geomatics and a discussion about the parts that were tackled successfully, the shortcomings and which decisions should have been different and why. The last section will be about future work and which technologies will be interesting to be added to such an application.

6.1 CONCLUSIONS

This section will be divided into four subsections that is equal with the number of the sub-question plus the main research question.

6.1.1 Sea surface height is a dynamic phenomenon (2.5D + time), what technique of animation should be used and why?

The sea surface height is alike most of the physical phenomena are dynamic meaning that change over time. Also, in most cases, the changes that take place in the world are not leaner and as a result, a simple trend cannot fully explain the changes of the phenomenon on one hand and on the other hand, statistical values can demonstrate only part of the truth. As a result, it is important to show to the user the data as raw as possible and one way is to create an animation that shows the exact values for each capturing date. So, the animation technique that was selected (Section 4.1) to be implemented is the video one and more specific the VP9 encoding in a WebM file format. A second option that can be considered as viable is the mp4/H.264 that even though it is the worst performer than the VP9/WebM it has the plus that it is supported by all the modern browsers and even most of the older ones. Overall using a video VP9/WebM visualization, a format that was actually created for the web, provides better storage performance in high quality and reliable output.

6.1.2 What elements of interactivity are relevant to a web mapping application and which ones should be implemented?

Since the advancements in computing systems, cartography is a science that utilized the new technology to a great degree introducing, on one hand, the Geographic Information Systems but also the interactive maps. The interactive maps can help the user better understand the phenomenon that is visualized since the experience of interacting with the map is more immersive. A second reason that the interactive maps are really important is that it allowed the map producer to better utilize and visualize the spatial data and as a result to bring to the user even more information

to consume. For the above reason, it was considered important to introduce some interactivity elements into the application.

1. *Handling the x and y coordinates*

   The first and the most important aspect that needs to be considered on a map is how to handle the horizontal plane. Most application is 2D representations of the world and as a result, the horizontal plane is the one that stands out in a visualization. the two interactivity elements that are used in this case are called "zoom" and "pan". The zooming and panning in web applications is something new and as a result, there have been proposals and implementations on the methodologies that can be used. The methodology that it is mostly used online and as a result in this application as well as the use of tiles and more specific to use the tilling scheme of Google Maps. The Google Maps tilling scheme organizes the directories of the saved tiles in the server in such a way that the webserver to be able to send the responses as efficient as possible. Most of the web applications that are using a tilling scheme such as the Google Maps, are using tiles 256x256 pixels in size but, the advancements in technology has led to the creation of screen with really high resolution that results to extreme numbers of tiles to be rendered at the same time and also considering the fact that most browsers have limited amount of elements that can be streamed at the same time, it can lead to lag-ish applications. The solution to the problem is to render bigger tiles (512x512 pixels in size) that reduces, on one hand, the number of tiles that need to be rendered and on the other hand, it reduces the load on the network. Overall, for the zooming and panning the best option is the use of a tilling scheme like the one of the Google Maps with tiles 512x512 pixels in size something that can be achieved simply enough through the Google Earth Engine (export map) for the server-side of the process and through the Leaflet JavaScript for the client-side. The final remark that need to be made, considering the zooming and panning, is the fact that since the tiles in this application are videos in order to offer a decent experience the least that can be done is to synchronize the video in order all of them to represent the same instance in time something that can be achieved with the use of the timing object technology.

2. *Handling the z coordinate*

   The sea surface height phenomenon, as it is implied by the name, mainly focuses on the third coordinate (z coordinate) and consequently, it is not possible to not handle it properly in this application. From the beginning of this project, it was decided that the height value will be represented with color. It will be easy to produce images, and consequently the videos, with a nice color pallet and then use this as an input for the tilling of the map (see handling the x and y coordinates) but the colors are all about the perception of the users and as a result it was decided to give them the opportunity to select (among a list of pre-selected options) which pallet fits them the best and use that one while trying to understand how the phenomenon changes through time and space. The original data-set was decided to be represented into a grayscale that one hand reduces the size of the encoded image and videos and on the other hand, it is easier to manipulate latter. The coloring process was decided to be implemented with the use of the Canvas API that is specifically designed for handling 2D drawings. The new colors (pallets) that are being used are produced through the d3.js library and then used by the Canvas API (that uses canvas elements to achieve it) to change the color of the original video.

3. *Handling the Time Dimension*

   The user when is trying to understand a phenomenon maybe he/she prefers to have at first a general overview (worldwide) and then to focus on more

detailed data, for example, a specific area that is of interest, something that is possible through the zooming in space (spatial scale). Similar behavior can be observed also for the time dimension that a user can be interested in having a more general overview of a phenomenon over the past 25 years or he/she can be also interested for more detailed data, for example, a specific year, that is also possible with zooming in time (temporal scale). The coarse data are represented as the average value per year, then there is an intermediate zoom level that is the mean value per month and lastly, there is the highest detail possible that in the case of the Sea Surface Height Anomaly data-set is per 5 days. The solution that it was used, Unlike the color-changing part, was the easiest one in simply creating a new directory for each zoom level. The tricky part was the handling of the most detailed zoom level that is actually the whole data-set (1825 images) which on one hand it could not be computed as a whole in Google Earth Engine but also the size of the final video that would have been too big to be loaded and rendered efficiently on the client. For this reason, this zoom level was divided into five different directories that it was called and played on the client consecutively. So, the user can zoom in time with a click of a button that calls a function to change the source of the video element.

4. *Querying Capabilities*

   The last interactivity element that it was decided to be added in the application it was the querying capabilities. As it has already been explained the sea surface height values are being represented by the color added with the canvas API. Even though the color is considered as one of the main techniques for communicating information in cartography it is not always descriptive enough. The color is better used in order to provide a general idea of a phenomenon especially since it relies on the perception of the user. A large number of the users have a hard time distinguishing color hues that are close in the spectrum and as a result, it is difficult for them to distinguish what the actual values are. For that reason, it was introduced the querying option that the user with a click can see the actual numerical value of the height for a specific point on the surface of the Earth. This can be achieved since it is possible to translate the colors into RGBa values (Canvas API) and after interpolating the true values (in meters) to find the one that corresponds to the specific hue.

6.1.3 What type of architecture is more appropriate for implementation with these characteristics?

Every web application has as a goal to provide the users with data that are not (normally) stored on the HTML document that is the basis of the application, but it is retrieved from a server that is stored. The data that is stored on a server can be either produced on it with a specified process or it can also be retrieved from another source. There is also the possibility that this information that is presented to the user is processed on the client before reaching its final form. All this flow of information is described in the system architecture and it is considered as one of the most important aspects of the application. The system architecture is divided into two main components that are named server-side and client-side. For this application the even though many processes took place on the server-side since it was a one-time computation of the video directories it can be considered that it plays the role of the storage of the video tiles. In order to connect the server with the client, there is the webserver that is a software that "serves" the requests for data (in this case video tiles) of the client from the server. The application developed for this thesis is mainly client-based since most of the processes (interactivity elements) happen on the browser something that requires enough processing power on the user's computers.

6.1.4 *Main research question:* What is an optimal WebGIS-architecture for making an interactive - dynamic visualization of the sea-surface height phenomenon?

The term optimal is rather strong on one hand but on the other hand, it has room for different interpretations. Through the sub-questions that were answered in the previous sub-sections, it is possible to extract one opinion on what can someone interpret as an optimal architecture. The only other relevant application that is available online is the Google Earth Engine Timelapse that uses a unique tilling scheme and tile size that is difficult to implement, limited interactivity elements and bad storage managements storing unnecessary overlapping tiles. The key characteristics that are trying to improve the disadvantages of the Google Earth Engine Timelapse implementation are:

- The use of videos as animations that provides good data compression with good quality
- Use a simple tilling scheme (Google Maps) widely used in the industry with an easy application through the use of already created javascript libraries.
- Big rectangular tiles (512x512) are improving the efficiency of the application but not too big that affects the user's experience.
- Combine the coloring with the querying in order to help the user better understand the visualized phenomenon.
- Handling the time dimension (zooming) is something that is not yet common on similar applications and can provide a deeper understanding of the visualized phenomenon.

Hopefully, this research is considered to add a few stepping stones towards the correct direction and other researchers to consider that it is close to what they think as optimal implementation (*https://giorgosdimo.github.io/MSc-Thesis/*). it is recognized that some parts can be improved but these aspects of the research will be analyzed in the discussion section that follows.

6.2 CONTRIBUTION TO THE FIELD OF GEOMATICS

The field of Geomatics is the one responsible for handling spatial data. A large category of spatial data is the environmental data that is not effective enough to analyze them ignoring the time dimension. The use of video elements for the inclusion of time in the visualization is not an uncommon technique while analyzing this type of data but, the use of videos as tiles in a web map tiled application is not something that can easily be found and only the Time Machine implementations can be found online (eg. Google Earth Engine Timelapse). The try to use visualization techniques conventional for static maps (tilling scheme) is something that never been tried before, even though it should have been since it can help with the interoperability of this kind of application. The use of 512x512 pixels in size tiles is rather common nowadays (Mapbox) but not for the case of videos. The use of such tiles (the methodology can work with smaller tiles (256x256)too but it is not advisable) leads to another first of this approach that is the try to synchronize the video tiles to visualize almost fluid passing of time on a browser. The use of different spatial scales in mapping applications is really common (through zooming and panning in space), but the use of temporal scales (zooming in time) is something new. The final part of the firsts for this research is the use of more complex interactivity elements

Figure 6.1: An example of corrupted tiles

than just the zooming and panning that have been tried in the Time Machine approach too. Hopefully, in the future, some more refined version of the methodology used in this application to pass through the process of standardization and just like there is a Web Map Tiled Services (WMTS) of OGC or a TMS of OsGeo to also exist a Web Video Map Services (WVMS) for online tiled video maps.

6.3 DISCUSSION

In this section, the goal is to pinpoint the parts of this research that the decisions that were made can be described as controversial.

The video creation technique even though it was able to produce reliable videos as results, there were also times that the result either they were corrupted from the beginning (as soon as they were created through the process described in section 5.2) or they were appeared corrupted when they were loaded on the browser (Figure 6.1).

Such cases appeared mainly when mp4/H.264 formatted video tiles were used and this is one of the reasons that the WebM/VP9 was suggested, nevertheless some of the browsers do not support the WebM/VP9 format and as a result, it is possible to have to handle mp4/H.264 formatted tiles with the dangerous of some times when they load to appear corrupted.

Probably the biggest struggle of this project was related to the synchronization of the video tiles. The use of the timing object fixed the large part of the problem but still, it is not 100% reliable. The most important problem of the timing object is the fact that the tiles that are connected with it are unable to be properly deleted when zooming and panning and as a result, the sum of all the videos that have been loaded are still running in the background slowing down the application considerably. There are cases that some videos need a second or two until they get truly

synchronized. Another issue that appears from time to time is that some videos will freeze while all the neighborhooding videos continue to play. This problem can be corrected most of the time by pausing and start playing again. This issue is not just with this application but with similar implementations all over the web. At the moment there is not an absolutely reliable technique that will make sure that from the moment of the loading of the video all the videos will be perfectly synchronized and never lose it.

The process that contributes the highest load on the user's computer is the coloring one and more specific the reading of the data from the first canvas. The lag that appears, in this case, can, on one hand, slow down the visualization and on the other hand, can cause visible desynchronization of the canvases (not video anymore). A solution to this problem can be the use of any of the other two techniques that have already been discussed in section 2.4. Of the two the most efficient one, although the most challenging one, is the use of WebGL API. Related with the same topic is the limitation that appears on the querying capabilities that require the use of a canvas something that adds complexity to the system.

One last issue noticed is that since the architecture of the application almost all the processes related with the interactivity elements are computed on the client and as a result appears to load the CPU significantly reaching at some points even the 20% mark that even though it is not considered terrible, if it is combined with other processes that run on the users computer it can cause overload. So, maybe some of the processes need to be "relocated" to other components of the application, for example, the server, or to other components of the user's computer like the GPU through the use of WebGL API.

6.4 FUTURE WORK

This section introduces some ideas for the expansion of the present project. These ideas will be divided into three main parts. The first part of suggestions is about ideas that were not implemented due to time and technology constraints or ideas that can help to have a bug free properly working prototype. The second part is about improving the working prototype introducing ideas that can improve the performance and the experience for the user in general. The last part is about ideas that are drastically different from the proposed implementation that can be considered as new applications with fundamental changes in areas of the project such as the data format and the architecture of the system.

6.4.1 Towards a Reliable Working Prototype

- The bigger issue with the implementation presented in this thesis is the synchronization. This issue even though it is solved almost entirely it still has some problems with the most important being the inability to disconnect the video tiles from the timing object something that slows down the application. The proposed idea is to communicate with the developing team of the timing object to provide some new insights on how this issue can be solved and how to improve the reliability of the video synchronization as a whole.

- Supposedly the technology issues derived from the timing object have been solved, the next step is to create a proper, user-friendly interface to be able to disseminate the ideas surrounding this project more effectively.

- The last proposal that belongs to this first part is related with the performance of a usability test in order to determine how users are perceiving the concepts introduced by this projects, which ones are considered more useful

and maybe to have an insight on what other options the user would like to be implemented in such an application.

6.4.2 Improving the Working Prototype

- The first idea that can be implemented in order to improve the performance of the application is the use of WebGL API as the main platform on which the interactivity elements will be computed. The WebGL API will introduce the use of GPU as the main computational unit something that can improve the performance of the application dramatically.
- An idea that can improve the user experience considerably is related to the improvement of the querying capabilities of the application. Until this point, the "click" on the map shows the value of the SSHA for a specific moment in time something that it is useful but it does not help the user to fully understand how the phenomenon changes. Producing, for example, a diagram that will be updated through time with the new values of the SSHA is providing a complete overview of the specified point in space. Another idea is to select a polygon and not just a point in order to have an improved overview of an area. One last idea related to the querying capabilities is to allow the user to export the resulting diagrams/histograms for his/her personal use.
- In this part of the future work also belongs to the proposal for the WVMS. Since through the above-mentioned ideas the prototype is considered to be in its best possible iteration then it is also ready to be standardized in order to be implemented by the community.

6.4.3 New Applications Ideas

- Since the phenomena that are similar to the SSHA are, in most cases, 3D, an idea is to use for the third dimension instead of just color a proper elevation difference something that may help the user better understand these three-dimensional phenomena.
- This thesis project is using raster data as data format and even though is efficient enough it has some drawbacks. One drawback is related to the fact that the raster data format is a storage requirement heavy format, especially when compared with vector data. Also, in order to be efficient in the animation part, it requires the use of video that has to be pre-computed on a server increasing the complexity of the computation on the server-side. As a result, a proposed idea is to use vector data format for such visualizations using SVG file format and WebGL or CSS3 for the animations on the client-side.
- Even though the proposed system architecture is leaning more towards a client-based approach it is possible to become even more "client-based". In such an architecture the input will be the NetCDF data from NASA and will be loaded, decoded and animated on the client-side without any pre-processing on the server-side. It will also be interesting to increase the performance of such an application by not loading the whole data-set when the application is launched but to load only the part of the data that are useful for the rendering of the animation with the specified characteristics (e.g. spatial/temporal scale).
- There is a trend in web visualizations (Chapter 3) in using virtual globes instead of a simple 2D map. The virtual globes are making the experience of the user more immersive and as a result, can be used for applications such as the one presented in this thesis project.

BIBLIOGRAPHY

Adnan, M., Singleton, A., and Longley, P. (2010). Developing efficient web-based gis applications.

Al-Qurabat, A. (2015). Web geographic information system. https://www.researchgate.net/publication/320357022_Web_Geographic_Information_System. [Online; accessed 24-October-2019].

Bláha, J. and Sterba, Z. (2014). Colour contrast in cartographic works using the principles of johannes itten. *Cartographic Journal The*, 51:203–213.

Borunda, A. (2019). A heat wave is turning greenland's ice to slush. that's bad news. https://www.nationalgeographic.com/environment/2019/07/greenland-melting-second-time-this-summer-bad/. [Online; accessed 24-October-2019].

cartography2.org (2018). A history of animated maps. https://www.cartography2.org/a-history-of-animated-maps/. [Online; accessed 28-October-2019].

Choudhury, N. (2014). World wide web and its journey from web 1.0 to web 4.0. *International Journal of Computer Science and Information Technologies*, 5(6):8096–8100.

Christophe, S., Christine, Z., and Roussaffa, H. (2011). Colours harmony in cartography. *25th International Cartographic Conference (ICC2011)*.

Clarke, K. C. (1986). Advances in geographic information systems. *Computers, environment and urban systems*, 10(3-4):175–184.

ESRI (2019). History of gis.

Friendly, M. (2006). A brief history of data visualization. In Chen, C., Härdle, W., and Unwin, A., editors, *Handbook of Computational Statistics: Data Visualization*, volume III, pages ???–??? Springer-Verlag, Heidelberg. (In press).

Friendly, M. and Denis, D. J. (2001). Milestones in the history of thematic cartography, statistical graphics, and data visualization. *URL http://www. datavis. ca/milestones*, 32:13.

Goodchild, M. F. and Haining, R. P. (2004). Gis and spatial data analysis: Converging perspectives. *Papers in Regional Science*, 83(1):363–385.

Harrell, J. and Brown, V. (1992). The world's oldest surviving geological map – the 1150 bc turin papyrus from egypt. *Journal of Geology*, 100:3–18.

Harrower, M., Fabrikant, S. I., and Dodge, M. (2008). The role of map animation in geographic visualization.

Hauser, H., Rheingans, P., and Scheuermann, G. (2018). Foundations of data visualization (dagstuhl seminar 18041). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Huisman, O. and De By, R. (2009). Principles of geographic information systems. *ITC Educational Textbook Series*, 1:17.

Kaplan, S. (2019). One of alaska's warmest springs on record is causing a dangerous thaw. https://www.washingtonpost.com/science/2019/04/19/one-alaskas-warmest-springs-record-is-causing-dangerous-thaw/. [Online; accessed 24-October-2019].

Lyzi, D. (2016). 512 map tiles. https://blog.mapbox.com/512-map-tiles-cb5bfd6e72ba. [Online; accessed 28-October-2019].

MacEachren, A. M. (1998). Cartography, gis and the world wide web. *Progress in Human Geography*, 22(4):575–585.

Maxwell, G. (2009). Youtube / ogg/theora comparison. https://people.xiph.org/~greg/video/ytcompare/comparison.html. [Online; accessed 28-October-2019].

Midtbø, T. (2005). Interactive cartographic animations–analysing functionality in a web environment.

Ozer, J. (2010). Vp8 vs. h.264. https://www.streamingmedia.com/conferences/west2010/presentations/SMWest-2010-H264-VP8.pdf. [Online; accessed 28-October-2019].

Ozer, J. (2017a). Hevc: Rating the contenders. https://streaminglearningcenter.com/wp-content/uploads/2017/05/Comparing_Best_HEVC_Codec.pdf. [Online; accessed 28-October-2019].

Ozer, J. (2017b). Netflix on av1. https://streaminglearningcenter.com/codecs/netflix-on-av1.htmlf. [Online; accessed 28-October-2019].

Peterson, M. (1999). Active legends for interactive cartographic animation. *International Journal of Geographical Information Science*, 13:375–383.

Roth, R. E. (2013). Interactive maps: What we know and what we need to know. *Journal of Spatial Information Science*, 2013(6):59–115.

Stefanakis, E. (2017). Web mercator and raster tile maps: two cornerstones of online map service providers. *GEOMATICA*, 71:100–109.

Stewart, R. H. (2008). *Introduction to physical oceanography*. Texas A & M University College Station.

van Oosterom, P. and de Vries, M. (2018). Web map/feature services (wms/wfs).

Vatoliny, D., Kulikov, D., Erofeev, M., Dolganov, S., and Zvezdakov, S. (2018). Msu codec comparison 2017. http://compression.ru/video/codec_comparison/hevc_2017/MSU_HEVC_comparison_2017_P5_HQ_encoders.pdf. [Online; accessed 28-October-2019].

Veenendaal, B., Brovelli, M. A., and Li, S. (2017). Review of web mapping: Eras, trends and directions. *ISPRS International Journal of Geo-Information*, 6(10):317.

Weeman, K. and Patrick, L. (2018). New study finds sea level rise accelerating. https://climate.nasa.gov/news/2680/new-study-finds-sea-level-rise-accelerating/. [Online; accessed 24-October-2019].

Whitehouse, D. (2000). Ice age star map discovered. http://news.bbc.co.uk/2/hi/science/nature/871930.stm. [Online; accessed 24-October-2019].

COLOPHON

This document was typeset using LaTeX. The document layout was generated using the `arsclassica` package by Lorenzo Pantieri, which is an adaption of the original `classicthesis` package from André Miede.