Web Visualization of Trajectory Data using Web Open Source Visualization Libraries

Nguyen Hoang Long

March, 2010

Web Visualization of Trajectory Data using Web Open Source Visualization Library

by

Nguyen Hoang Long

Thesis submitted to the International Institute for Geo-information Science and Earth Observation in partial fulfilment of the requirements for the degree of Master of Science in Geo-information Science and Earth Observation, Specialisation: Geoinformatic.

Thesis Assessment Board:

Chair: Prof.Dr. M.J. Kraak

External examiner: Dr. G. Andrienko

Supervisor: Dr. U.D. Turdukulov

Second supervisor: Ms. Dr. C.A. Blok

INTERNATIONAL INSTITUTE FOR GEO-INFORMATION SCIENCE AND EARTH OBSERVATION

ENSCHEDE, THE NETHERLANDS

… Dedicated to my family …

Disclaimer

This document describes work undertaken as part of a programme of study at the International Institute for Geo-information Science and Earth Observation. All views and opinions expressed therein remain the sole responsibility of the author, and do not necessarily represent those of the institute.

Abstract

Our world is not static. Changes occur in all phenomena in our world over time. Understanding these dynamics to discover spatio-temporal patterns, relationships and trends is an important step to the solution of many global, regional, local problems. Trajectory data is an example of dynamic spatio-temporal data. The increase in number and size of trajectory data challenges researchers to develop a tool for visually exploring the information that the trajectory data contains. Web services with open source visualization libraries offer the capability to build a web application with many explorative functionalities. Nevertheless, the question which web library is the most suitable for visual exploration of trajectory data has not been answered yet. The selection of suitable web visualization library or combination of libraries may be an approach for visual exploration of the trajectory data with multivariate attributes.

This research proposes a conceptual model for visual exploration of trajectory data with multivariate attributes by using timemap.js web visualization library. A prototype for visual exploration of iceberg data as a case study was designed and implemented. Our design was based on the related literatures about trajectory data as well as iceberg data and the Visual Information Seeking Mantra "Overview, zoom and filter, details on demand" that was proposed by Shneiderman.

The result of this research is a prototype for visual exploration of iceberg data using timemap.js visualization library. The prototype provides 2 views that are linked together (map view and timeline). The prototype has the capability to display overview of each trajectory and the details of each trajectory different zoom levels. It also provides an animation function and some filtering functions. Colour (for existential changes and trajectories) and size (for size changes) were the main visual variables used in the design of the prototype.

Finally, usability testing was conducted in this research by using eye tracking and questionnaire. It dealt with the assessment of the prototype from an effectiveness, efficiency and satisfaction perspective. Furthermore, by using eye tracking the information about where users looked at most while exploring the iceberg data are given. The results may be valuable for further research.

Key words: Client, server, database, web services, web library, PHP, timemap.js, animation, timeline, Visual Information Seeking Mantra

Acknowledgements

I want to thank my parents and my younger sister who always support and encourage me throughout my studies.

I would like to express my sincere gratitude and admiration to my first supervisor Dr. Ulanbek Turdukulov and my second supervisor Dr. Ms Connie Blok for the invaluable advice; guidance and support throughout the research period.

I want to thank Vietnamese government for sponsoring me to study at ITC

I want to thank Dr. Corné van Elzakker for supporting me in the use of eye tracking in my research.

I want to thank Matthijs Noordzij and Malte Risto of the UT for use of the lab and explanations of the hard- and software for ET

I want thank Lizda Iswari my classmate for sharing her works with me.

I want thank Rozita my classmate for conducting the usability test with me.

I want thank Emmanuel Adetutu, Irma Kveladze, Adam Klpkemei, Qiuju Zhang, Tom Mank, Li Xia and Nazanin Sepehri for participating our usability test.

I want thanks Nguyen Duc Ha and Nguyen Thanh Nga for helping me during the study time in ITC

I want thank Nick Rabinowitz who has developed the timemap.js libraries for his supports and sharing the code

Finally, I want to say thanks to all of my Vietnamese friends in ITC as well as my GFM friends for all the moments we shared together.

Table of contents

Contents

List of figures

List of tables

List of abbreviations

DBMS	Database management system
GUI	Graphical User Interface
NIC	National Ice Center
SQL	Sequence query language
WMS	Web Mapping Services
WSs	Web Services

1. Introduction

1.1. Background and problem statement

Our world is not static. Changes occur in all phenomena in our world over time. Understanding these dynamics to discover spatio-temporal patterns, relationships and trends is an important step to the solution of many global, regional and local problems (e.g. global warming, pollution) (Blok, 2005).

Trajectory data of moving objects, such as cars, hurricanes, icebergs, humans are recently gaining attention. They play a crucial role in decision-making processes in several application domains. Trajectory data are spatio-temporal data (Bogorny and Alvares, 2009). Trajectory data are commonly represented as sampled points in the form (*Id, x, y, t*), where *Id* is the trajectory identifier and *x, y* represents spatial coordinates at a certain time *t* (Bogorny and Wachowicz, 2009). Moving objects do not only have these attributes, but they also can be coupled with other attributes, such as size, shape, temperature, etc. Those attributes can be derived by integrating with other data, such as grid data. Furthermore, trajectory data have little or no semantic information (Alvares et al., 2007).

Due to the increase in capacity to record trajectory data provided by modern technologies (especially GPS and remote sensing), the number and size of trajectory data are increasing rapidly. In many applications, useful information can only be extracted from trajectory data if the use context is considered (Spaccapietra et al., 2008). According to Spaccapietra et al. (2008), analyzing trajectory data plays a important part in understanding and managing complex phenomena which involve object movements. From a GIScience point of view, visual exploration is one of the most useful approaches for analyzing and understanding spatio-temporal data. Traditionally, during exploration, the questions "what", "where" and "when" are often asked by the users (Peuquet, 1994, Li et al., 2007). However, visualizing and analyzing spatio-temporal data is not an easy task (Compieta et al., 2007).

The data about iceberg movements of Antarctica is an example of trajectory data. Iceberg data are related to one of the biggest problems that human have to face, in this century: global warming, which causes polar ice shelves to smaller and fall apart. Since the last decades of the 20th century, international organizations and scientists have been studying movement of Antarctic icebergs. The National/Naval Ice Data Center has a database of tracked Icebergs from 1976 to present (URL 1). The collected data attributes include the id of the iceberg, latitude, longitude, recorded time, size of the iceberg and the name of the satellite that recorded that iceberg. Several groups are interested in icebergs, including climatologists, oceanographers, navigation and petrol companies, universities and institutes. For example, for climatologists, iceberg numbers and their life cycles can be considered as climate indicators; or navigation companies are interested in iceberg positions because of the threat that icebergs may cause for their ships.

Due to the rapid development of the internet in the recent years, Web Services (WSs) now are among the most powerful tools to collect and disseminate data and specifically geo-data. In general, WSs are a set of related functionalities that can be programmatically obtained through the Web. These

functionalities represent the various operations offered by the Web service (Guermouche et al., 2008). According to Mc Kee (2004) “Web Services are Web accessible applications and application components that exchange data, share tasks, and automate processes over the Internet”. Accessing geo-data through a web browser can enhance the accessibility and re-usability. Users can easily access geo-data with an internet connection at any time and no specific software is needed (apart from a web browser). Therefore, WSs can reduce the price of software integration and data-sharing (McKee, 2004)

Researchers who want to build a WS for visual exploration of trajectory data are confronted with some general issues when visualizing trajectory data on the web. These are, for instances: how to visualize trajectory data on the client side? What views and interactions are needed? What server side and database supports are needed?

Blok et al. (2009) described the implementation a prototype for visualizing iceberg data by using SIMILE AJAX and Google API’s. This prototype provides several data views linked together (e.g., Map View, Timeline, Scatter plot). It allowed users to gain overview, zoom-in, zoom-out, filter, and view details on demand. However, there were still some limitations in this prototype. For example, lack of supported functionalities from the server sides and the database (e.g. giving limitations in data updating and data storing on the server side), ineffective filtering and querying mechanisms. This prototype could not integrate Google Earth API in Exhibit’s map view. Movement paths could only be displayed by points in Google Map. The data input was limited to 10,000 records.

Recently, there are many web interactive visualization libraries available on the internet. These include java script libraries such as Openlayers (URL 2), SIMILE (URL 3), Timemap (URL 4), Google MAP API (URL 5), Processing (URL 6), ESRI JavaScript API (URL 7). They offer many build-in functions for visualizing data on the client side. Each of these libraries has its own advantages and disadvantages. Our research aims to investigate those web visualization libraries, choose the most suitable library (libraries) for visualizing trajectory data while considering the characteristics of the data and user tasks (iceberg data are used as a case study).

1.2. Research identification

1.2.1 Objective

The main objective of this research is to propose a framework for visual exploration of trajectory data on the web using existing visualization libraries

1.2.2 Sub-objectives

The main objective can be reached by attaining the following sub-objectives:

1. To understand the users’ requirements for exploring trajectory data.
2. To translate user requirements into functionality and representations for visual exploration of trajectory data, also taking user scenarios into account.
3. To gain an overview of relevant existing open source web visualization libraries for visualizing trajectory data that are coupled with multivariate attributes.
4. To identify required functionalities on the server side and the database.

5. To design the GUI (Graphical User Interface) and implement a research prototype for visual exploration of trajectory data considering the required views and functions for visual exploration of trajectories
6. To evaluate the functionality of the prototype.

1.2.3 Research questions

1. To understand the users' requirements for exploring trajectory data
    - What are the characteristics of trajectory data, iceberg data?
2. To translate user requirements into functionalities and representations to visual explore trajectory data, also taking user scenarios into account.
    - Which tasks do users want to do in exploring trajectory data, iceberg data?
    - How to translate user requirements into functionalities and representations?
3. To gain an overview of relevant existing open source web visualization libraries for visualizing trajectory data with multivariate attributes
    - What are the criteria for choosing suitable web visualization library?
    - Which open source web library (libraries) is the most suitable for visualizing trajectory data that are coupled with multivariate attributes?
4. To identify required functionality on the server side and the database
    - Which functionality from the server sides and the database are required?
5. To design the GUI and implement a research prototype for visual exploration of trajectory data considering the required views and functions for visual exploration of trajectories
    - How to set up the database to store the trajectory data and connect database with the client side?
    - How to design and implement a research prototype, taking user tasks and data characteristics into account?
6. To evaluate the prototype
    - How to evaluate how the prototype?

1.2.4 Innovation aimed

The innovation aims at providing a framework for visualizing trajectory data by using web visualization library (libraries), taking user tasks and data characteristics into account.

1.2.5 Related Work

As stated before, Blok et al. (2009) reported about research that visualized iceberg data by using SIMILE AJAX and Google API. Huang (2003) proposed TOPMODEL using web map services for modelling an interactive visualization analysis framework. This approach showed the possibility of combining Java with Web Map Service (WMS). Nevertheless, this framework only works with static events not with dynamic movements. Ying et al. (2004) proposed a web-based visualization of geo-spatial data by applying style sheet (Extensible Stylesheet Language Transformation/XSLT) to the Geography Markup Language and generated 2D (Scalable Vector Graphics/SVG) and 3D (Virtual Reality Modeling Language/VRML and eXtensible 3D/X3D) geographical representations. XSLT was used for mapping the traffic data in the virtual environment as a case study. According to this study, the

proposed approach is useful for exploration and analysis of large volume of spatio-temporal data. Conversely, the functions of interactions with dynamic movements over time were still missing in this study. Another study is performed by Andrew Trice (URL9) was, concerned about the implementation of a simple interactive temperature map of US by using Google Maps API for Flash/Flex. This map allowed users to perform some simple interactions, such as zoom-in, zoom-out. However, it did not provide any functionality for dealing with the trajectories. Becker et al (2009) used OGC's standard to storing spatio-temporal data combined with SVG SMIL animation for visual exploration of iceberg data. They provided users various visualization models and parameters (e.g. different animation types, temporal legends and animation speed) to observe moving object's behaviours and detect patterns and characteristics of movement (e.g. speed and direction). However, this prototype has limitation in visualizing large number of objects in a long period. Furthermore, this research did not focus on the appearance and disappearance events. Except speed and direction of object's trajectories, in this research prototype, other characteristics of trajectory data such as travel distances, trajectory duration were not considered.

1.3. Project setup

1.3.1 Methodology

To answer questions posed in section 1.2.3, the following methodology was applied. Figure 1.1 describes the schema of the methodology.

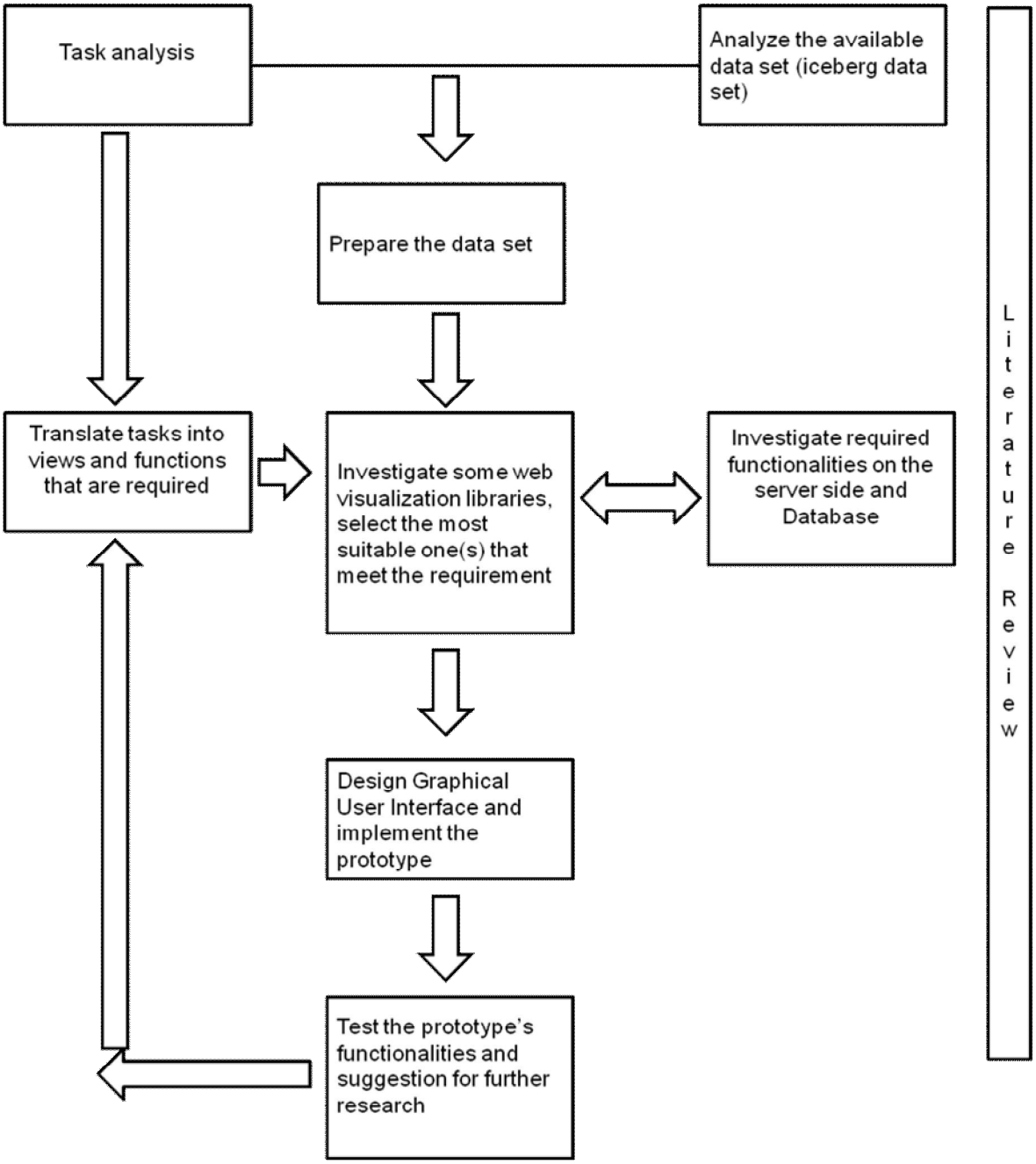

Figure 1.1 Methodology

- *Literature review:* the first task will be to review literature on characteristics of trajectory data and iceberg data. Literature review also will be done in every phases of the research.
- *Task analysis*: by reviewing related literature about which tasks are needed in exploring trajectory data, particularly iceberg data.
- *Translate tasks into views and functions:* in this step, required tasks will be translated into views and functions.
- *Analyze the available data set (iceberg data set):* iceberg data set will be used as my study case. Hence, to get the understanding of these data is very important.

- *Prepare the data set*: iceberg data can be downloaded from the website of National/Naval Ice Data Center in excel format. The data, then, will be cleaned to remove duplicates and errors. After that, some new parameters may be derived further processing existing data. For example, an average speed of each iceberg can be calculated by using the positions and the recorded time.
- *Investigate relevant open source web libraries and select the most suitable one for visualizing trajectory data:* each web visualization library will be investigated by looking at its examples and functionality. The most suitable web visualization library (libraries) will be chosen based on the needed views and functions. However, due to the fact that there are many interactive web visualization libraries, but the limited time, some criteria will be created to limit number the web visualization libraries.
- *Investigate required functionality on the server side and the database:* while considering characteristics of the selected web visualization library (libraries).
- *Design Graphical User Interface and the prototype implementation:* a visualization interface prototype will be implemented.
- *Test the prototype:* finally, the prototype will be tested in order to determine its capabilities and limitations. With regard to the result of this research, some suggestions will be given for further research.

1.4. Thesis structure

This thesis contains six chapters. In the first chapter, the motivation, problem statement, research objectives, research questions and methodology are described.

Chapter 2 will review the related literatures about characteristics of trajectory data and iceberg data as well. Then tasks that are required while exploring trajectory data and iceberg data will be investigated and translated into views and functionalities. Finally, the issue about functions from the database/ server side and the client side will be discussed

In chapter 3, some Web Visualization Libraries will be investigated in order to know their advantages and disadvantages in visualizing trajectory data. Based on the reviewed characteristic, the most suitable web open source visualization library/libraries will be chosen for implementing a visual exploration of trajectory data.

Chapter 4 will describe the implementation process of the prototype steps by steps. Solution to questions of how to make use of web open source visualization libraries, how to connect to the database, as well as how to design a GUI for the prototype will be described.

A short evaluation of functionality of the prototype will be conducted in chapter 5. The evaluation will be based on capabilities and limitations of the prototype as experienced by test participants.

Finally, conclusion and recommendation will be discussed in chapter 6.

2. Data and tasks analysis

In this chapter, first the characteristics of trajectory data and the iceberg data will be reviewed through related literatures. Next, the tasks in visual exploration of trajectory data and iceberg data will be analyzed. Finally, those tasks will be translated into required views and functions.

2.1. Trajectory data and their characteristics

This study focuses on movement of objects along trajectories. Movement of an object is defined as the change in the physical positions over time of an object with respect to geographical space (Andrienko et al., 2008). A trajectory is the path made by an object when it moves. This path cannot be made instantly. It needs a certain amount of time. Therefore, time is an inseparable element of a trajectory (Andrienko et al., 2008), and data set about moving objects include ID, x, y, t fields.

Generally, trajectory data can be divided into two major groups of geo-referenced and non-geo-referenced (Dodge et al., 2008). In other words, some objects move in geographic space so they can be geo-referenced, such as birds, vehicles, humans, icebergs. Other phenomena move in a non-geographic space, such as gaze point movements in eye tracking research. Each of these movements has some similarities; in addition, they also have their own characteristics in data structure, behaviours and nature of movement.

This study is mainly concerned with geo-referenced trajectory data. Some characteristics of geo-referenced movement are described (Andrienko et al., 2008) as follows:

- **Space** is a set of location points, or places, but it does not have origin and orders between points. Thus, it requires reference systems such as geographical coordinates. Depending on the practical purposes, space can be considered as 1-dimensional (linear element), 2-dimensional (element is defined by two coordinates), 3-dimensional (element is defined by three coordinates) or 4-dimensional (element is defined by four coordinates). Comprehensive analysis may require uniform data in the same reference systems. The physical space is continuous and it consists of a number of locations. There are always some locations in between any two different locations. It may be useful if one treats space as a discrete or even finite set of locations while studying trajectory data.

- **"Time** is a continuous set with a linear ordering and distances between the elements, where the elements are moments, or positions in time" (Andrienko et al., 2008). Physical time is not just in linear sequences, but it can also be cyclic, because of earth rotations and revolution. There are two types of cycles: natural cycles and cycles related to human's activities. Temporal cycles can be viewed as a hierarchy of nested cycles – year/month/day-in-month or year/week-in-year/day-in-week. Identifying temporal cycles that are relevant is very important in analyzing trajectory data.

Other characteristics can be divided into two categories, movement-related characteristic and overall characteristic of a single trajectory (Andrienko et al., 2008) . Movement-related characteristics consist of: time, positions of object in space, direction of the movement, change of direction, speed, change of speed, accumulated travel time and distance. Overall characteristics include: geometric shape of the trajectory in space, travelled distance, duration of the trajectory, movement vector or the major direction (from the start position to the destination), mean, median and maximal speed, dynamics or behaviour of the speed.

Apart from examining a single trajectory, users are interested in comparison of two or more trajectories (Andrienko et al., 2008). These include trajectories of different objects (e.g. different cars), trajectories of the same objects but in different times (e.g. trajectories of the same bird but in different seasons), or the same trajectory of the same object but in different times (e.g. trajectory of the same student on the way to school and on the way back home)

Moving objects also have their own characteristics that may influence the trajectory path. Hence, those characteristics need to be considered while analyzing trajectory data. For example: the movement of birds may be depended on their type, health condition or other properties. (Dodge et al., 2008) proposed spatial constrains (networks, barriers, etc) that also affect the trajectories. Objects may move freely in a free environment (e.g. iceberg movements) or they move along specific networks (e.g. cars have to move on the roads).

The trajectory of a moving object may also be influenced by different events and phenomena. For example: movements of people are influenced by weather, legal regulations or traffic jams, and so on. The analysts have to use additional data and background knowledge to take external events or phenomena into account while analyzing trajectory data.

2.1.1. Iceberg data

In this research Antarctic iceberg data are used as a case of trajectory data. Polar regions play an importance role on global climate. Seventy five percent (75%) of the world's fresh water can be found in the Antarctic ice sheet (Collings et al., 2006). Antarctic icebergs are formed by the separation or calving of big masses of ice from ice shelves and glaciers. After they are formed, icebergs normally travel following the ocean currents (Ballantyne and Long, 2002). An iceberg's trajectory is also influenced by wind directions, wind speed, temperature, air temperature, solar radiation, etc (Death et al., 2006). Global sea levels are changing due to ice melting. Iceberg can be considered as an indicator for global climate changes. Generally, it is not feasible and costly to travel to Antarctica to collect data and make observations. The National Ice Center (NIC) started collecting Antarctic iceberg positions from the late 1970s (URL 1). NIC uses a variety of satellite sensors for tracking and monitoring Antarctic icebergs that meet two basic requirements: the icebergs must measure at least 10 nautical miles (nm) along the long axis, and the second requirement is that the icebergs be south of 60s latitude. NIC identifies and names Antarctic icebergs when they meet these two requirements. Icebergs that have been reduced in size below 10nm or that are lost of visual sighting for 30 consecutive days are removed from the database. All iceberg data since 1976 are available in HTML and Excel formats.

	A	B	C	D	E	F	G	H
1	iceberg	date	latitude	lat_positio	longitude	long_posit	size	satellite
2	A01	1978299	-56	S	-34.2	W	45X25	DMSP
3	A01	1978306	-55	S	-32.3	W	45X25	DMSP
4	A01	1978318	-53.7	S	-35	W	45X20	DMSP
5	A01	1978325	-53.7	S	-35	W	45X20	NOAA
6	A01	1978339	-53.4	S	-34.4	W	45X20	DMSP
7	A01	1978354	-53.9	S	-33.8	W	35X20	DMSP
8	A01	1979011	-53.3	S	-33	W	35X20	DMSP
9	A01	1979017	-53.3	S	-32.8	W	35X20	DMSP
10	A01	1979043	-51.7	S	-31.6	W	20x12	DMSP
11	A01	1979130	-50.2	S	-26.3	W	20x12	DMSP
12	A02	1978319	-58	S	-47.8	W	30x10	NOAA
13	A02	1979015	-58.8	S	-49	W	30x10	DMSP
14	A02	1979045	-58	S	-44.9	W	30x10	DMSP
15	A02	1979067	-58	S	-44.9	W	30x10	NOAA
16	A02	1979080	-57.4	S	-45.9	W	30x10	NOAA
17	A02	1979283	-56	S	-35.8	W	30x10	DMSP
18	A02	1979290	-56.2	S	-34.2	W	30x10	DMSP
19	A02	1979297	-54.6	S	-32.8	W	30x10	NOAA
20	A03	1979040	-58.3	S	-43.9	W	25X10	DMSP
21	A03	1979067	-58.3	S	-43.9	W	25X10	NOAA
22	A03	1979130	-59.5	S	-44.5	W	25X10	DMSP
23	A03	1979172	-59.5	S	-44.5	W	25X10	DMSP

Figure 2.1 Original iceberg data in Excel format [URL1]

Figure 2.1 shows that the collected data attributes include:

- Id of the iceberg
- Recorded time (in Julian like format)
- Iceberg position: latitude and longitude
- Size of the iceberg
- Name of the satellite that recorded that iceberg

Beside those attributes above, icebergs also have other characteristics that can help users in analyzing icebergs behaviours and changes. Based on the attributes above, some movement-related attributes as well as overall attributes can be derived, such as:

- Speed of an iceberg at a particular time
- Acceleration – the change in speed
- Orientation - azimuth cardinal direction
- Travelled distance of an iceberg
- Lifetime (duration) of an iceberg from its first appearance to the last observation.

- Major direction from the start position to the destination
- Mean, median and maximal speed of an iceberg during its lifetime

The Antarctic continent is divided into four quadrants as we can see in Figure 2.2:

- 0-90W quadrant “A” Bellinghausen
- Weddell Sea, 90W-180 quadrant “B” Amundsen
- Eastern Ross Sea, 180-90E quadrant “C” Western Ross Sea
- Wilkesland and 90E-0 quadrant “D” Amery/Eastern Weddell Sea

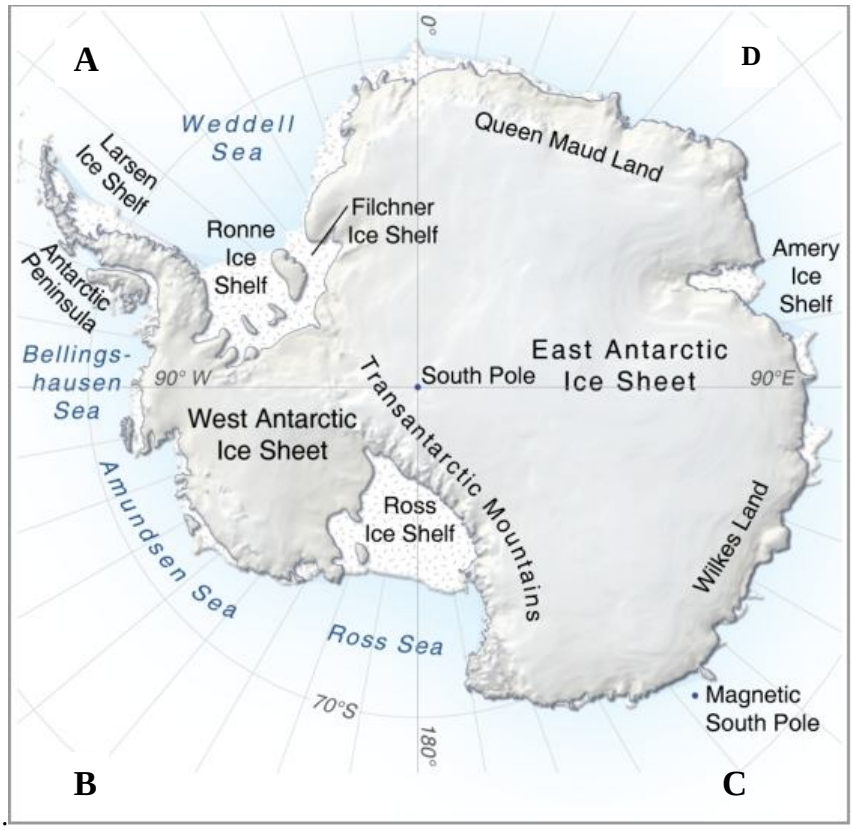

Figure 2.2 Antarctica, topographic map (URL 10)

2.2. Tasks analysis and translation into views, functions and interactions

2.2.1. General tasks

Normally, analyzing trajectory data is important for the two following main reasons: to understand the current patterns and to predict the future based on the understanding of current patterns (Andrienko et al., 2007). To understand patterns, changes have to be considered. According to Blok (2000), spatio-temporal changes can be classified into three kinds:

- Existential changes such as appearances and disappearances.

- Changes of spatial properties: change in location, shape, size, orientation and so on.
- Changes of thematic properties (changes of values of attributes).

To detect changes on spatio-temporal data, Peuquet (1994) distinguishes three components of spatio-temporal data: space (where), time (when) and objects (what) (see Figure 2.3). Due to that, there are three possible basic kinds of questions:

When + where -> what: Users want to know the objects or group of objects at the specific location or set of locations at a given time.

When + what -> where: Users want to know the location or set of locations occupied by a specific object or group of objects at a given time.

Where + what -> when: Users want to know the times that a specific object or group of objects occupied a specific location or set of locations.

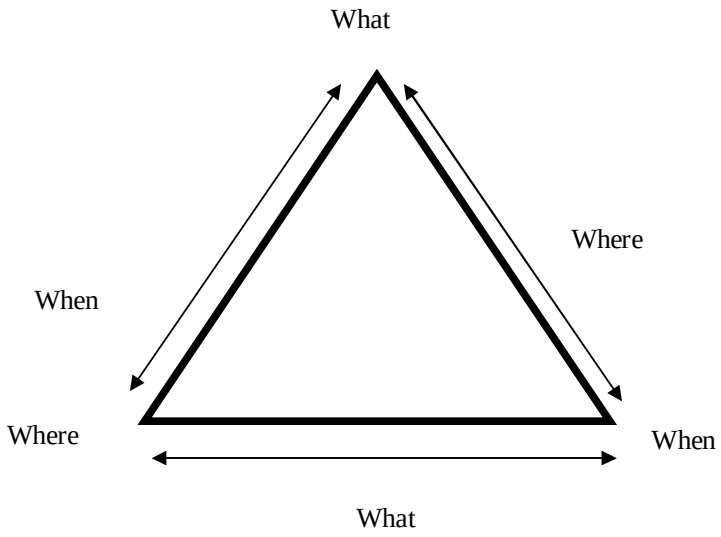

Figure 2.3 Questions while exploring changes on spatio-temporal data (Peuquet, 1994)

To answer the above questions, users need something like a view where users can find information to answer the questions. In this case, a map view may help users answer to where question. A temporal view may help users answer to when question. An attribute view may help users answer to what question. As shown in the Figure 2.3 questions what, where and when are related to each other. Hence, the views that were discussed above should be also linked together to give answers to the questions.

Based on these three basic questions, there are many questions related to moving objects that can be easily found in literature, for example, in (Andrienko et al., 2008):

- What objects were present in location *p* at time *t?*
- At what moments (if any) did object *o* visit location *p?*

- How did the location of object *o* change from time t_1 to time t_2?
- What were the relative positions of objects o_1 and o_2 *at time t?*

Guting and Schneider (2005) mentioned some more specific questions:

- Determine trajectories of birds, whales…
- Where are the whales now?
- Which routes are used regularly by trucks?
- Which taxi is closest to the passenger's requested position?
- At what speed does this plane move? What is its top speed?

In order to give answers to those questions above, some generic tasks for visual exploration will be described. Visual exploration is to be considered as an effective approach to provide material for human's perception and plays an important role in analyzing trajectory data. For visual exploration, a visualization of the data is needed. Keim et al (2008) classified visualization into scientific and information visualization. Scientific visualization has developed for visualizing scientific data that have spatial and time references. Information visualization has developed for visualizing abstract data that have no spatial references. Kiem and his colleagues consider visualization is a medium in a semi-automated analytical process, where humans and machines can cooperate to give a better result. This study focuses on scientific visualization, or geovisualization.

Geovisualization is "the use of visual geospatial displays to explore data and through that exploration to generate hypotheses, develop problem solutions and construct knowledge" (Kraak, 2003). From the map point of view, geovisualization is the integration of approaches from scientific visualization, information visualization, cartography, image analysis, exploratory data analysis and GIS to provide theory, methods and tools for visual exploration, analysis, synthesis, and presentation of geospatial data (any data having geospatial referencing) (MacEachren and Kraak, 2001).

As mentioned above, for visual exploration of trajectory data some generic tasks will be described. Furthermore, tasks will be translated into views, functions and interactions. Although, there are many visual design guidelines, the Visual Information Seeking Mantra propagated by Shneiderman (1996): "Overview first, zoom and filter, then detail-on-demand" will in this research be considered a basic guiding principle.

Provide overview of the whole data set: The goal of the overview is to give users the first impression of the data. Because we are dealing with geo-referenced trajectory data, thus to give users an overview of the whole data set to get the overall pattern, all the trajectories should be visualized by lines on a map view. However, if the data contain too many trajectories, users have difficulties to get information when all the trajectories are visualized. Some trajectories have some similar characteristic, such as they may have the same starting points and same destination or they may have similar shapes. Hence, depending on the numbers of the trajectories, Andrienko et al (2007) suggest using a clustering function. Figure 2.4 shows an example visualization of trajectory data after clustered. Clustering can be defined as a process

to group a set of objects into classes of similar objects (Han et al., 2006). After clustering we can visualize representative trajectories (Lee et al., 2007) of each cluster instead of visualize all the trajectories. Summarization or aggregation functions can be also used as an alternative solution for clustering function. However, in some cases, it is required not only to visualize one type of entity, but two or more types of entities (e.g. to visualize movement of persons as well as cars to see the relationship between them). To deal with these requirements, a classification function should be applied to classify entities into numbers of classes. Classification can also be applied to classify entities that have different properties (e.g. classify the entities based on their size). Normally, the numbers of classes should be from 5 to 9 for a quick comprehension.

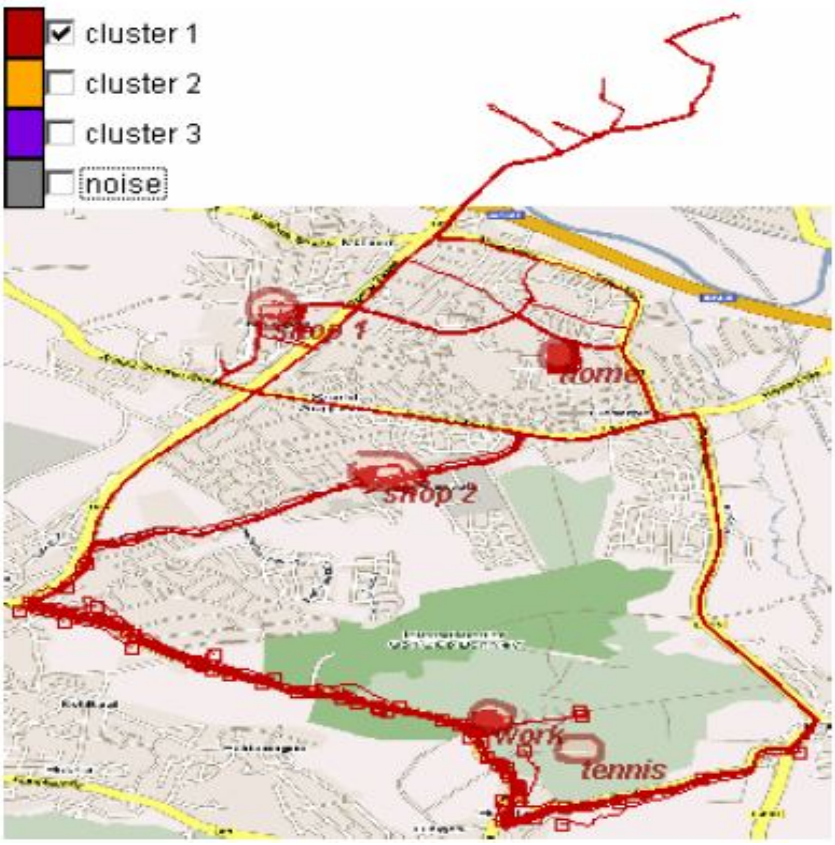

Figure 2.4 Example of visualization trajectory data after clustering (Andrienko et al., 2007)

Spatial Zooming: Users may be interested in some portion of the maps and we need a tool to enable them to focus on a particular region. We can provide a zooming mechanism that let users zoom-in and zoom-out. A useful way to zoom-in is by pointing to the location and double clicks or using the wheel on the mouse to perform the zooming command. Furthermore, in some cases, depending on the zoom factor users may need a panning tool that let them to navigate on the screen.

Filtering/query: While the aim of overview is to give users an idea about the whole data set. The aim of filtering is to focus on the items of interest from a user's perspective. If there are too many trajectories visualized on a map view, users find it very difficult to get the information that they need. In addition, sometimes users are just interested in some particular objects/trajectories, or they want to get information about the patterns at a certain time. They may also be interested in a trajectory of an object

or trajectories of groups of objects that have some characteristics in common at a specific period. In principal, there are two ways to filter and query data:

- Add requested information to what is already displayed on the user's view or highlight them
- Eliminate from user's view the items that do not meet the requirements.

View detail-on-demand: as mentioned before, trajectory data do not only contain data about object *Id*, *x*, *y* and *t*, but objects can also have their own characteristics that users want to know. A simple approach that is normally used in this situation is to click on an item to get a pop-up window with the values of all attributes, or to go to another web page to get the information. For visualizing of trajectory data, it requires at least interactive, dynamically linked spatial, temporal and multi-dimensional attribute views on the data and a brushing tool (mainly for attributes brushing, temporal brushing) (Blok et al., 2009). Linking those views together will provide a coherent image of the data (see Table 2-1).

Tasks	**Required Views**	**Required Functions**
• Overview	• Map view • Temporal view • Attribute view	• Clustering, summarization, aggregation • Classification
• Zoom		• Zoom-in • Zoom-out • Panning
• Filtering/ query		• Add items on user's views • Remove items from user's views • Brushing
• View detail-on-demand		• Pop-up window

Table 2-1 Summary of tasks (based on the Visual Information Seeking Mantra), functions and views

Four main tasks that were discussed above are only the basic tasks in analyzing geographic data in general. To detect changes animation with user controlled interactions (e.g. play, stop, change display speed, etc), timeline, or brushing tools and graphs are considered as useful techniques. In (Dibiase et al., 1992), animation is considered as a useful technique for emphasizing the location of a phenomenon and its attributes, or visualizing change in spatial, temporal, and attribute dimensions However, to deal with some specific tasks that relate to change detection, some other techniques could be applied as well.

Those tasks will be discussed below. Table 2-2 shows the summary of the tasks, required functions and views that focus on detecting the changes.

Emphasize existential change (e.g. appearance and disappearance): information about moving objects' appearances and disappearances is importance. The numbers of appearances or disappearances of objects affect the number of objects that are visualized on the view. Hence, using dynamic visualization variables to emphasize objects (e.g. blinking), or providing dynamic legends (when objects appear or disappear, the legends will be automatically updated) or using visualization variables, such as colour to differentiate the appearance, disappearance and with others, can be applied.

Visualize changes in spatial properties: can be divided to three sub-tasks

- ***Dealing with position changes or tracing trajectory/trajectories:*** tracing trajectories of objects (e.g. in an animation) in a specific period or during their time span and their positions at different moments is one of the main interests of users. Filtering/query/brushing can be also used (e.g. by providing a time-slider). In some cases, it is better if different trajectories can be visualized with different colours. Furthermore, using juxtaposition of maps to represent situations at different moments is another option.

- ***Dealing with shape or/and size changes:*** shape and/or size of objects may change during their movements. Those factors are important in analyzing behaviours of moving objects. Using various visual variables to visualize those characteristics (e.g. size and shape) combined with animation or make use of time-slider may be a solution.

- ***Dealing with height, altitude, volume changes:*** with the development of technologies, the data about moving objects not only contain *x*, *y* but also *z* (altitude): the height and the volume of objects. To visualize those kinds of information, we can make use of 3D environments or providing scatter plot or other graphs linked with the map view. A scatter plot is very useful in analyzing correlation between two quantitative variables (Moore, 2006). For example, we can see the correlation between volume and speed of objects.

Show the changes of thematic properties: sometime, users want to know the trends, such as trends in temperature or speed changes of an object. It is sometime very difficult to visualize on the map. Hence, providing a time plot view linked with the map view is an option.

Visualize external phenomena that influence the trajectory: As mentioned before, there are many external phenomena that may influence trajectories of objects. Hence, it is importance to visualize them as well. It can enhance the abilities of understanding moving objects' behaviours and patterns. Therefore, solution is to provide suitable dynamic or static representation of the external phenomena.

Tasks	Required Views	Required Functions

• *Emphasize existential change*
• *Visualize changes of spatial properties*
• *Show the changes of thematic properties*
• *Visualize external phenomena that influence the trajectory*
	• Map view
	  • Visualize trajectories by lines
	  • Use visualization variables
	  • Use dynamic visualization variables
	  • Use dynamic legends
	  • Use connecting lines (for visualizing iceberg calving)
	  • Using juxtaposition of maps
	• Temporal view/ Timeline
	• Animation
	• Attribute view (scatter plot, time series plot, etc.)
	• 3D view
		• Zoom-in
		• Zoom-out
		• Panning
		• Play
		• Stop
		• Change speed
		• Brushing
		• Rotating
		• Display trend line

Table 2-2 General tasks, views and functions (focusing in the changes)

2.2.2. Specific tasks with iceberg data

Like other trajectory data, questions about iceberg data are also related to "what", "where" and "when". For example:

- Where is the particular iceberg at a certain time?
- Which icebergs are at location *p* at moment *t*?
- What is the size of the iceberg at location *p* at moment *t*?
- What are the trajectories of icebergs?

Execution of tasks that we discussed above can give answers to those questions. However, each type of moving objects may have its own special characteristics and behaviour. For instance, icebergs data have a special behaviour, calving or split of an iceberg into two or more entities. For example: two icebergs, A20A and A20B are resulted from A20. In other words, A20 is a parent, while A20A and A20B are children. Understanding the relations between "parent" and "children" is importance for analyzing iceberg data. An iceberg split means that the parent iceberg will disappear and two or more new

icebergs children will appear. If we simply use dynamic legends and blinking objects, it will be not sufficient to show the parent-child relationship. To solve this problem, we suggest using connecting lines between “parent” and “children” as an addition.

Another issue with the iceberg data that we have to consider is temporal irregularity. Iceberg positions are recorded every 1-5 days (Ballantyne and Long, 2002). Due to the difficulties to distinguish icebergs from surrounding sea ice under certain conditions, there are some observational gaps in the dataset. This will lead to uncertainty when users want to know the positions of icebergs on a day with missing observations. Currently, data on the parent-child relation is missing along with the exact times when an event, i.e., split had occurred. Using an interpolation function may be a solution for filling in these gaps within a datasets.

The required views and functions in visual exploration of iceberg data are described in the Table 2-3 below:

Tasks	**Required Views**	**Required Functions**

Visual exploration of iceberg (see tables 2.1 and 2.2)	• Map view
	  • Visualize trajectories by lines
	  • Use visualization variables
	  • Use dynamic visualization variables
	  • Use dynamic legends
	  • Use connecting lines to visualize parent-child relationships
	  • Using juxtaposition of maps
	• Animation
	• Temporal view/Timeline
	• 3D view
	• Attribute view(s) (e.g. Scatter plot, time plot)
		• Zoom-in
		• Zoom-out
		• Panning
		• Rotating
		• Clustering
		• Classification
		• Add items (highlight items on user’s views)
		• Remove items from user’s views
		• Brushing
		• Play
		• Stop
		• Change speed
		• Interpolation
		• Display trend line
		• Pop up window

Table 2-3 Table of required views and functions in visual exploration iceberg data

In this research, we only concentrate on three particular events:

- Calving/splits
- Size changes
- Appearances/disappearances

In addition to the events above, we need to focus on giving users an overview of iceberg data as well as view details on demand. However, as mentioned before trajectory data usually are very big, it is very difficult to see all the information in a glance. According to Ogao and Kraak (2002) animated map "enable one to deal with real world processes as a whole rather than as instances of time". Table 2-4 shows all the views and functions that will be implemented in the prototype later.

Tasks	**Required Views**	**Required Functions**

Visual exploration of iceberg	• Map view
	  • Visualize trajectories by lines
	  • Use visualization variables for events
	  • Use dynamic visualization variables
	  • Use dynamic legends
	  • Use connecting lines to visualize parent-child relationships
	• Animation
	• Temporal view/Timeline
	• Attribute view(s)
		• Zoom-in
		• Zoom-out
		• Panning
		• Pop up window
		• Clustering
		• Classification
		• Add items (highlight items on user's views)
		• Remove items from user's views
		• Brushing
		• Play
		• Stop
		• Change speed

Table 2-4 List of the required views and functions in visual exploration iceberg data will be created in this research

2.3. Server side and database support

In client and server systems, database are stored and controlled by the server, users send requests and receive information from the server via a client. Some functions can be performed on both server and client side. There is a trade-off between performing functions on the database server and on the client

side. For example: classification function can be implemented on both client side or on server side. In case of client side and the data are very big, the client front-end application has to down load all the data from the server and then perform the classification function on this data. It will take a lot of resource in client side. On the other hand, in case of small data, clients can easily down load from the server and do the processing in clients side. It can reduce the tasks that have to be performed on the server side. If there are too many clients send requests to the server simultaneously and the capabilities of the server are limited. Server may be down or it cannot work properly.

Trajectory data sets usually are very big. Hence, we suggest creating some supported functions from the server side to increase the performance on the clients such as interpolation, clustering and classification.

According to (URL 11) some main functions of client and server are described.

a. Functions of the Database Server

Database are stored and controlled by the server. It allows clients to request and extract information from the database. The main functions of database server are:

- Managing the database stored inside it.
- Receiving requests from client, respond to the clients.
- Managing the database recovery, security.
- Database access, user management, and all control functions are handled by the database server.

b. Functions of the Client

Clients use front-end application to communicate with the server. The main functions of clients are:

- It manages the GUI through front-end application and some data manipulation functions.
- It sends the queries/requests to the database server.
- It interprets the results and presents or displays the results to user.
- Client can also process further the queries results to get the information that users need.

2.4. Summary

In this chapter, we have discussed trajectory data as well as iceberg data and their characteristics. Further, general tasks that are required in visual exploration of trajectory data and some specific tasks associated with iceberg data have been discussed. Then, all the discussed tasks were translated into views and functions. In the next chapter, we will investigate some web visualization libraries; look at their characteristics, advantages as well as disadvantages. We will choose the most suitable web visualization library/libraries based on their characteristics and required views and functions above.

3. Web Visualization Libraries

Many web visualization libraries are recently available on the internet. However, since time is limited, we had to focus on a subset, which originally consisted of ten candidate libraries. Based on the required functions and views discussed in chapter 2, we created some criteria that are described in APENDIX A.1 We then investigated which criteria could be met by the libraries, and decided to limit further investigation to six web visualization libraries, namely ArcGIS JavaScript API, Processing, OpenLayers, Google Maps API, SIMILE and Timemap.js.

This chapter describes characteristics, advantages as well as disadvantages of six web visualization libraries. After that, depending on the function that those libraries offer, and in line with the required functions and views that were discussed in chapter 2, we choose the library that offers us most of the required functions and views. Hence, we do not have to build those functions and views from the beginning.

3.1. ArcGIS JavaScript API

The ArcGIS JavaScript API is a web lightweight visualization library for creating GIS maps and tasks in Web applications. This JavaScript API is developed by ESRI, and hosted on ArcGIS Online. The ArcGIS JavaScript API offers users an easy way to build and deploy their applications.

With ArcGIS JavaScript API, users can access information from different sources in one application. One common reason why users might do this is to display some of user's information on top of a base map from ArcGIS Online or from other web sources. In another scenario, users might use a model on one server and display the results on a map from a different server. This combination of information from multiple web sources is sometimes called a "mashup".

ArcGIS JavaScript API offers users to embed maps and tasks from the ArcGIS Server into their web applications. It provides the capability to display data on an interactive map in any projection. Users can do zooming and panning, search and display the results on the map. It also allows users to draw graphic features, and provides pop-up windows. Furthermore, ArcGIS JavaScript API also provides the animation functions such as start and pause. However, it does not provide the slider for controlling the speed of the animation.

ArcGIS JavaScript API is based on the Dojo JavaScript Toolkit. It allows users to access to Dojo widgets (dijits) and other JavaScript tools. It provides the possibility to integrate with other libraries, such as the Google Chart API.

However, a big disadvantage of ArcGIS JavaScript API is that users need an ArcGIS Server to use of this API. Figure 3.1 shows an example of ESRI Java script API. Moreover, the animation is not easy to build.

Figure 3.1 Example of ESRI Java script API (ULR 12)

3.2. Processing

According to (URL 6), Processing is an open project created by Ben Fry and Casey Reas. It was originally intended to be used in Java run-time environment. In 2008, John Regig created 2D context of Processing to JavaScript for web-based applications. Processing is used by many users (e.g. students, artists, designers, researchers, and hobbyists) for studying, prototyping, and production. It runs in most of the browsers (e.g. FireFox, Safari, Opera, Google Chrome as well as Internet Explorer) using explorer Canvas. Processing is free to download at [ULR6]. It has different version for GNU/Linux, Mac OS X, and Windows.

Processing.js is open source libraries. It allows users to display data on the map view. It offers users functions to create web's animation, image and interaction without using Flash or Java applets. Processing.js is lightweight, easy to learn. The Processing syntax is simple. Although processing.js has capabilities in visualizing data on a map view and animation, however the functions for interaction with map that are required for our purpose (e.g. zooming, panning, pop up window, etc.) are missing. Figure 3.2 gives an example of using processing.js. In this example, we can see it has animation function but not interactive functions, such as zooming and panning.

Figure 3.2 Example of using Processing.js (URL 13)

3.3. OpenLayers

According to [ULR 2], OpenLayers is an open source JavaScript that allows users to display map data in web browsers, with no server-side dependencies. It facilitates users to display map tiles and markers loaded from any source. Open Layers is released under a BSD-style License. Like the Google Maps API, Open Layers implements a JavaScript Application Programming Interface for creating rich web-based geographic applications. It is developed by Open Source software community.

OpenLayers uses industry-standard methods for geographic data access (e.g. the OpenGIS Consortium's Web Mapping Service and Web Feature Service protocols). OpenLayers separates map tools and map data to allow users make use of all the tools on all the data sources.

By looking at examples that are available on the internet, we can clearly see the capabilities of OpenLayers. OpenLayers provides many capabilities including visualizing user's data on a map view, interactive functions (e.g. zooming, panning) with the map, filtering/querying, display the geometry of data (e.g. points, lines and polygons), animation, items clustering, etc.

Nevertheless, it does not provide timeline. The pop-up window and the filtering functions in OpenLayers are not easy to build. Furthermore, the clustering algorithm that is used is pixel based and does not consider about time. OpenLayers displays animation of a sequence of layers or images. Figure 3.3 shows an example of animation in Open Layers. However, for each animation steps, we have to create a layer class. Moreover, Openlayers does not provide the temporal brushing for large time-series data.

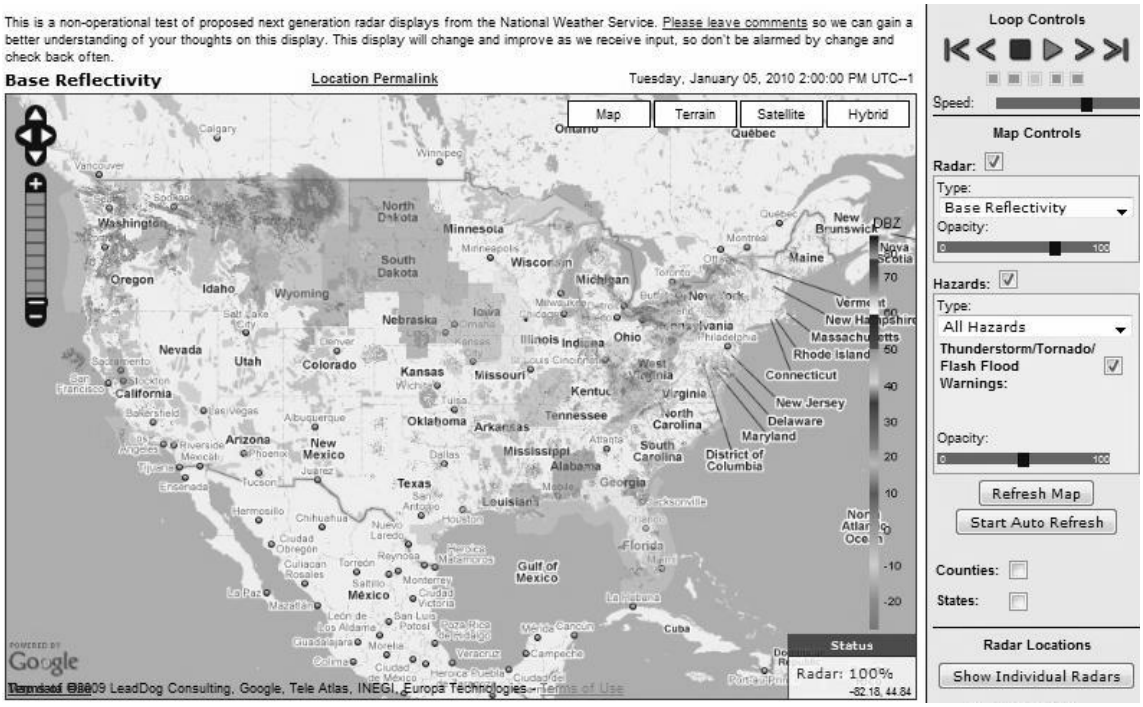

Figure 3.3 Example of animation using OpenLayers (URL 14)

3.4. Google Maps API

The Google Maps API is a JavaScript web library that allows users embed Google map onto their web pages. The Google Maps API is free and available for any website.

The Google Maps API supports most recent web browsers (e.g. Internet Explorer (Windows), FireFox (Windows|Mac|Linux), Safari (Mac|iPhone), and Chrome (Windows)). It requires an API key to embed Google Map onto a user's web pages. Users can get the API key for free at Google web site.

The Google Maps API enables users to interact with three kinds of layers: map, satellite and hybrid. It provides capabilities to display points, lines and polygons in different sizes and colours on the map view. Users can overlay KML, KMZ and GeoRSS files on top of the map. It also provides a number of functions for manipulating maps, such as zooming, filtering and pop-up window. Furthermore, Google Maps API provides markermanager.js that allows users to handle large number of markers. Figure 3.4 gives an example of handling large number of markers using the Google Maps API and markermanager.js. Nevertheless, the Google Maps API has limitations in animation, and timeline.

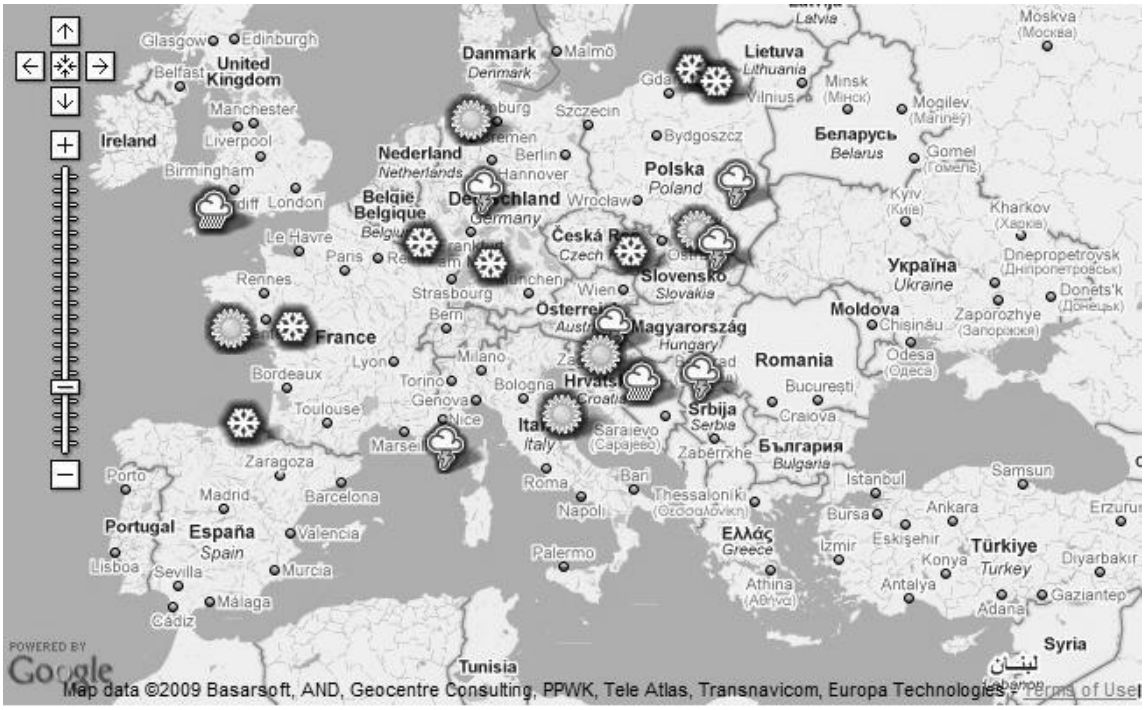

Figure 3.4 Example of handling large number of markers using the Google Maps API and markermanager.js (URL 15)

3.5. SIMILE

SIMILE stands for Semantic Interoperability of Metadata and Information in unlike Environments. SIMILE aims at creating robust, open source tools that enable users to access, manage, visualize and reuse digital assets.

SIMILE is supported by AJAX (Asynchronous JavaScript and XML). AJAX provides a capacity to get data from the server asynchronously without disturbing the displays and behaviours of the existing page.

SIMILE Ajax is a JavaScript library. It is slim, fast and can run on most of the recent browsers (e.g. FireFox, Opera, and Internet Explorer). SIMILE Ajax consists of about 20 sub projects. Table 3-1 shows all the sub-projects of SIMILE.

Project	**Description**
Appalachian	Allows users to manage and use several OpenIDs to ease the login parts of the browsing experience.
Babel	Is a data converter services. It allows users to convert between various data formats, especially, convert data into the Exhibit JSON format.
Citeline	A web application to facilitate the web publishing of bibliographies and citation collections as interactive exhibits and facilitate the sharing of this type of data.
HTTPTracer	An application that is between your HTTP client and your HTTP server. It sniffs all the communication that goes on between the two.
Java Firefox Extension	
jsTeX	Java script library providing tools for interpreting some basic TeX encodings

	and transform them into HTML.
Longwell	Allows users to extract data from different sources and mix them together. It turns a browser into a mash up platform.
Piggy Bank	Allows users to make use of available information on the web in more efficient and flexible ways that are not offered by the original sources.
RDFizers	
Referee	Referee reads web server logs, crawls your referrers (the links that point to your pages) and extract metadata from those pages and text around the links that pointed to your pages
Solvent	A Firefox extension that assist users in writing JavaScript screen scrapers for Piggy Bank
Semantic Bank	Lets users persist, share and publish data that are collected by individuals, groups or communities.
Zotz	A Firefox add-on that allows users to publish citations from your Zotero to an Exhibit (via Citeline) in only one step
Timeline (graduated)	Provide timeline view for temporal data visualization
Timeplot (graduated)	A time series plotting widget
Exhibit (graduated)	Allows users to create their web pages with support for sorting, filtering, and rich visualizations.
Gadget (graduated)	Is useful in exploration, migration, cleanup, evaluation schema emergence of large quantities of XML data
Welkin(graduated)	A graphical graph visualize powered by RDF data. It can display graphs with a real-time interactive visualization
Fresnel (graduated)	A vocabulary for displaying RDF
Seek (graduated)	Allows users to search through their email more effectively.

Table 3-1 Sub-projects of SIMILE

SIMILE allows users to display items with different sizes and colours on the map and the timeline simultaneously. It allows users to interact with the map view (e.g. do zooming and panning) SIMILE also provides the time plot and scatter plot. It provides pop-up window for the attribute view. These views can be linked together. SIMILE allows users to do filtering, brushing and searching by using Exhibit facet. Figure 3.5 shows an example of SIMILE Exhibit. However, SIMILE does not support users in displaying line objects or drawing lines. It also does not provide animation function.

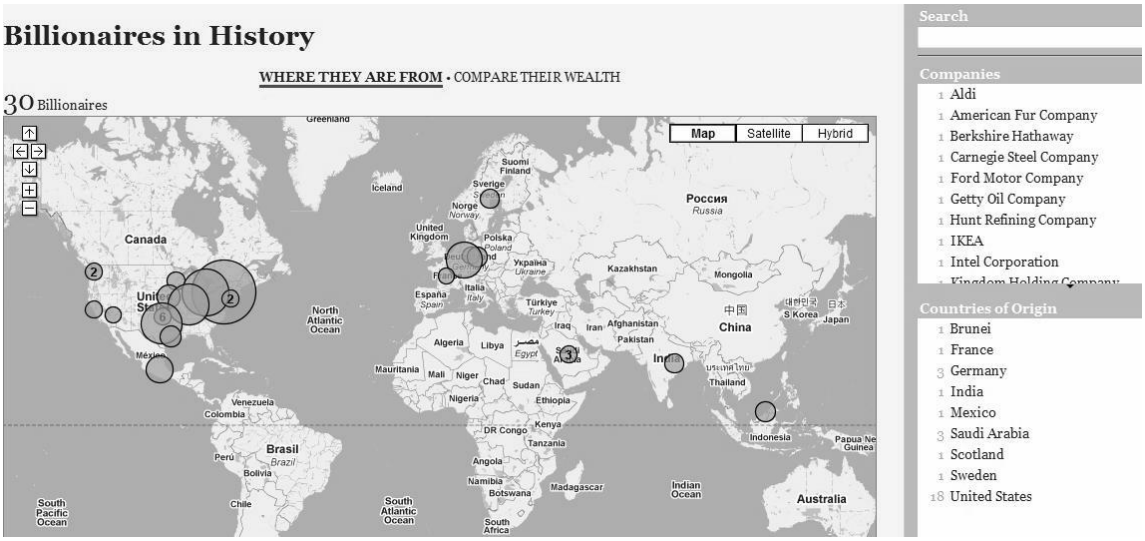

Figure 3.5 SIMILE EXHIBIT Examples: Billionaires in History (URL 16)

3.6. Timemap.js

Timemap.js is an open source JavaScript library. It allows user to combine Google map with SIMILE timeline. As a result, it provides functions from both Google Maps API and SIMILE timeline. Like other JavaScript libraries, timemap.js can run on most of the recent web browsers (e.g. IE, FireFox, Chrome, etc).

Timemap.js allows users to display points, lines on the map and timeline simultaneously. The timeline view is linked with the map view. This library allows user to load multiple datasets in JSON, KML, or GeoRSS format. Timemap.js only displays items in the visible range of the timeline on the map. Timemap.js also enables users to do filtering and displays different objects with different icons and colours. Nevertheless, the filtering function that is provided by timemap.js is very simple. It does not allow users to do filtering several objects at the same time. Although timemap.js does not support animation, we can simulate animation by give to the option to move (pan) the timeline manually. In this case, timeline can be considered as a linear temporal legend. Figure 3.6 shows an example of Timemap.js using KML.

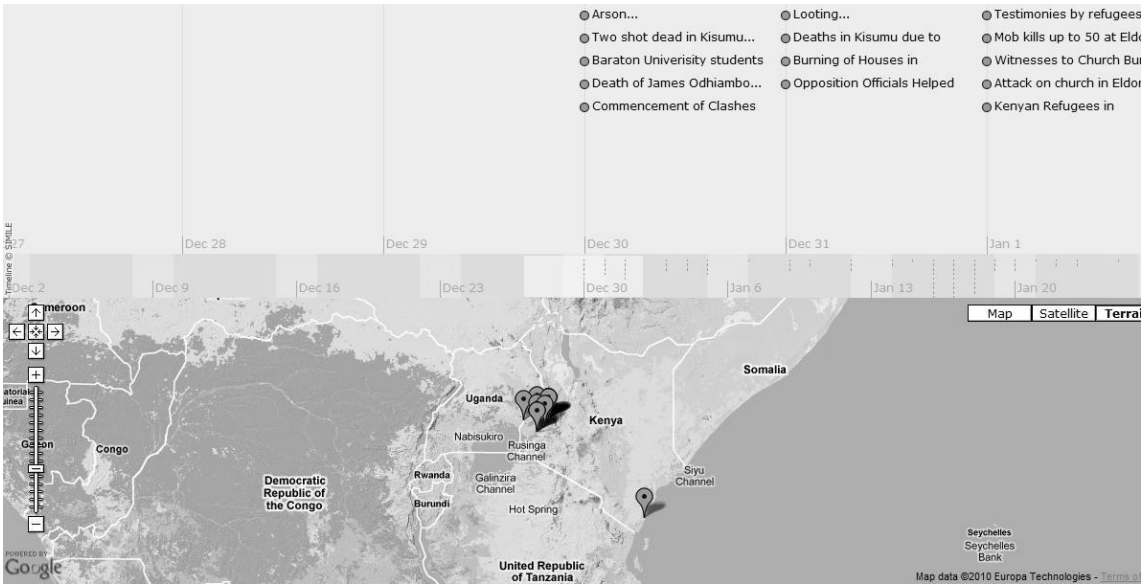

Figure 3.6 Example of timemap.js using KML file (URL 17)

3.7. Conclusion

In this chapter, we have investigated several open source web visualization libraries. After considering their characteristics, advantages as well as disadvantages, we decided to choose timemap.js for our purpose, because timemap.js allows us to combine Google Maps API and SIMILE Timeline, hence it inherits the functionalities of both of them. Furthermore, we see a possibility to make animation by using timemap.js. In the next chapter, design and implementation processes of the prototype will be described step by step.

4. Design and Implementation of the prototype

The previous chapter reviewed visualization libraries and timemap.js was chosen as the most suitable candidate. However, timemap.js misses a few of the required functionalities that need to be embedded.

This chapter describes the process of implementation of our prototype step by step. First, the iceberg data are stored in the PostgreSQL. Then PHP was chosen to build the middle ware between database and client side. Finally, the GUI of the prototype was created by using "timemap.js".

4.1. Conceptual model of the prototype

In this section, the conceptual model of our prototype is presented. A conceptual model is defined in (URL 18) as:"a type of diagram which shows a set of relationships between factors that are believed to impact or lead to a target condition; a diagram that defines theoretical entities, objects, or conditions of a system and the relationships between them".

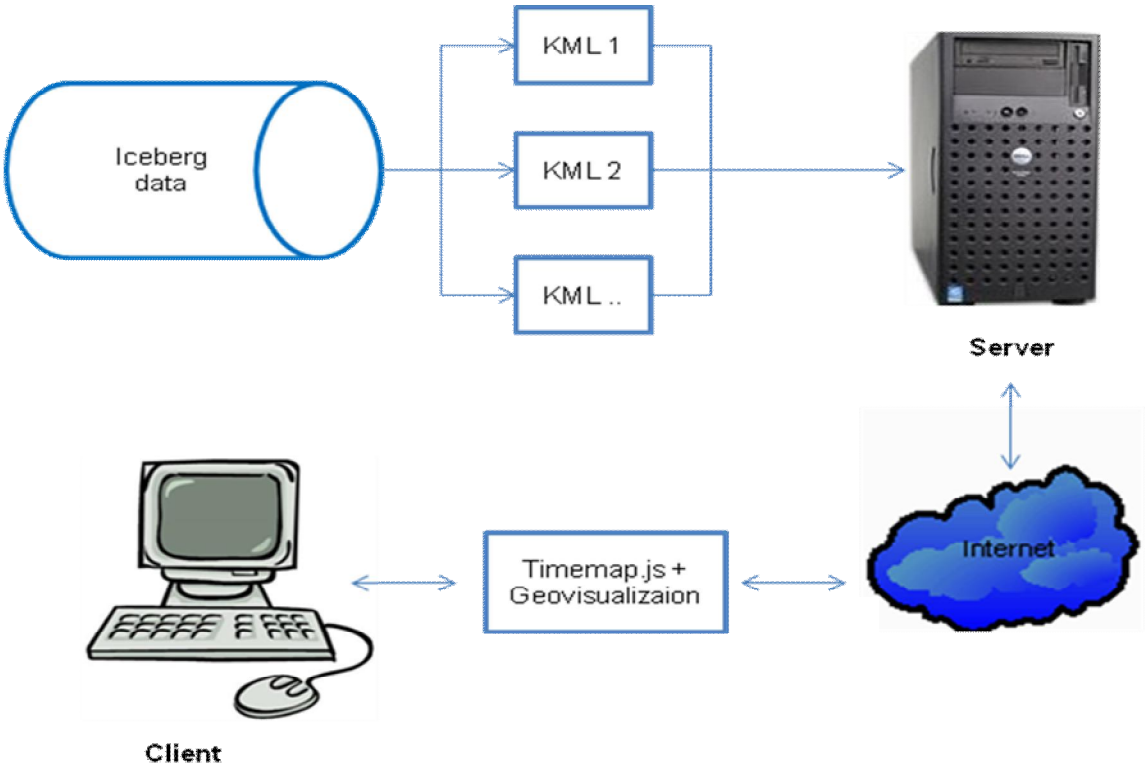

Figure 4.1 Conceptual model of visualizing iceberg data set using timemap.js

Figure 4.1 presents the conceptual model of the prototype and its components. The model shows the relationships from the database to the client/end users.

As mentioned in chapter 2, timemap.js supports different file formats, such as KML, JSON or GeoRSS. KML files were chosen for our purpose. A KML stands for “Keyhole Markup Language”. KML file follows the XML standard.

There are several reasons for choosing KML file. First KML file is human readable file. Users can open it by any text editor and understand it. Second, KML format is supported by many applications, such as Google Earth, Google Maps, Google Maps for mobile devices, NASA WorldWind, ESRI ArcGIS Explorer, Adobe PhotoShop, AutoCAD, etc (URL 19). We can use those applications to check with our prototypes. Finally, a KML file supports geographic features such as points, lines, polylines that are required in our research.

One of the main objectives of this prototype is to provide users with overview as well as detail-on-demand in one interactive map. In some cases, users only focus on the starting point and the destination of a movement. Hence, we intend to give users an overview of each trajectory. At the low zoom levels, the data of the appearances, the disappearances and the major direction (direction from the appearance to the disappearance) are shown to give users an overview of each trajectory. Combine with animation, an overview of the data set could be more or less given to the users. At the high zoom levels, the data about the details of each trajectory are shown. In some cases, when at high zoom levels at some specific area the data that are displayed on the map are still messy, users can perform a function to get another KML file that contains the clustered data of this area. However, we will not focus on clustering function on this research.

In figure 4.1, there are several KML files are generated from the database. Via the server and the internet, those KML files will be transferred to the client and combined with the timemap.js for geovisualization to help users answer the questions related to what, where and when. Those KML files are dynamically generated whenever the database has changed (e.g. update the database or insert new data into the database) or when the users want to use the clustering function a new KML will be generated and transferred to the client. Next, those KML files are processed by the timemap.js for the geovisualization. Finally, an application for visual exploration of iceberg data is created on the client side on the web browser.

4.2. Set up of the database

The first item of the conceptual model of the prototype is the database. The database is one of the important parts in our research. As discussed above, iceberg data will be stored in a DBMS (database management system). Putting data in a DBMS makes the management job easier. In addition, functions can be created in the DBMS for supporting the visualization on the client sides. In this research, PostgreSQL DBMS is used to store the iceberg data set.

4.2.1. PostgreSQL database

PostgreSQL is an object-relational database system. It has the features of a traditional proprietary database system with enhancements to be found in next-generation DBMS systems. PostgreSQL has most features that are present in large proprietary DBMSs, like transactions, sub selects, triggers, views, foreign key referential integrity, and sophisticated locking. PostgreSQL also offers some special

features like user-defined types, inheritance, rules, and multi-version concurrency control to reduce lock contention.

PostgreSQL is free software. It can work on almost all recent operating systems, such as Window, Linux, Mac OS X, etc.

4.2.2. Set up of the iceberg data set

Raw data that are downloaded from the NIC, are in excel files. Generally, the original data have many duplicates, outliers and others errors. We want to thank my classmate Lizda Iswari (Iswari, forthcoming) for sharing the clean data, for providing the table of parent and child relationships and for providing the summary table of each trajectory including the average speed, the lifetime, the travel distance of each trajectory.

We wrote a python script to make sequence query language (SQL) to insert iceberg data to the PostgreSQL database (see APPENDIX A.2). The output of the python script is a sql file. This file can be open in PostgreSQL to insert the data. Figure 4.2 shows the subset of iceberg data set inside PostgreSQL. Figure 4.3 shows the subset of the table of parent and child relationships in quadrant A.

	icebergid character var	times character var	lat real	lng real	size real	sizetype character var	appearance character var
1	A01	1978-10-26	-56	-34.2	3858.64	medium	First
2	A01	1978-11-02	-55	-32.3	3858.64	medium	Appearance
3	A01	1978-11-14	-53.7	-35	3086.91	medium	Appearance
4	A01	1978-11-21	-53.7	-35	3086.91	medium	Appearance
5	A01	1978-12-05	-53.4	-34.4	3086.91	medium	Appearance
6	A01	1978-12-20	-53.9	-33.8	2400.93	medium	Appearance
7	A01	1979-01-11	-53.3	-33	2400.93	medium	Appearance
8	A01	1979-01-17	-53.3	-32.8	2400.93	medium	Appearance
9	A01	1979-02-12	-51.7	-31.6	823.18	small	Appearance
10	A01	1979-05-10	-50.2	-26.3	823.18	small	Last
11	A02	1978-11-15	-58	-47.8	1028.97	small	First
12	A02	1979-01-15	-58.8	-49	1028.97	small	Appearance
13	A02	1979-02-14	-58	-44.9	1028.97	small	Appearance
14	A02	1979-03-08	-58	-44.9	1028.97	small	Appearance
15	A02	1979-03-21	-57.4	-45.9	1028.97	small	Appearance
16	A02	1979-10-10	-56	-35.8	1028.97	small	Appearance
17	A02	1979-10-17	-56.2	-34.2	1028.97	small	Appearance
18	A02	1979-10-24	-54.6	-32.8	1028.97	small	Last

Figure 4.2 Iceberg data set in PostgreSQL

As shown in the Figure 4.2 there are two new attributes added to the iceberg data: sizetype and appearance. The former attribute is generated to classify the size of the icebergs. There are three types of size: small - indicates icebergs that have size less than 2000 square kilo-meters, medium – indicates icebergs that have size larger than 2000 square but less than 4000 square kilo-meters, big – indicates icebergs that have size larger than 4000 square kilo-meters. The last attribute in the database is used to specify the appearances, intermediate positions and the disappearances of the icebergs.

	child character var	child_lat real	child_lng real	first_appeara character var	parrent character var	parent_lat real	parent_lng real	last_appeai character v
1	A20A	-55.5	-64.69	1986-09-10	A20	-56	-65.29	1986-08-21
2	A20B	-56.6	-65.29	1986-09-10	A20	-56	-65.29	1986-08-21
3	A21A	-55.8	-64.59	1987-06-25	A21	-59.2	-66.79	1986-12-16
4	A22A	-44.4	-76.6	1994-02-09	A22	-44	-76.6	1994-01-26
5	A22B	-43.2	-76.2	1994-02-09	A22	-44	-76.6	1994-01-26
6	A22C	-39.8	-69.6	1998-09-08	A22A	-45.4	-76.3	1998-08-24
7	A23A	-41.5	-76.2	1991-11-16	A23	-41.3	-76.2	1991-10-27
8	A23B	-39.8	-76.1	1991-11-20	A23A	-41.5	-76.2	1991-11-16
9	A24A	-48.5	-49.7	1992-04-03	A24	-47	-50.69	1992-03-11
10	A24B	-48.1	-49.9	1992-04-03	A24	-47	-50.69	1992-03-11
11	A24C	-48.8	-49.79	1992-04-03	A24	-47	-50.69	1992-03-11
12	A24D	-48.3	-50.79	1992-04-03	A24	-47	-50.69	1992-03-11
13	A27A	-42.5	-75.4	2004-08-17	A27	-42.51	-75.34	2004-08-10
14	A32A	-57.7	-64.9	1995-05-31	A32	-57.7	-64.99	1995-05-24
15	A32B	-57.7	-64.89	1995-05-31	A32	-57.7	-64.99	1995-05-24
16	A35A	-38.9	-76.3	1998-10-04	A35	-38.8	-76.2	1998-09-30
17	A35B	-38.4	-76.2	1998-10-04	A35	-38.8	-76.2	1998-09-30
18	A35C	-39.5	-75.7	1998-12-08	A35A	-39.4	-76.1	1998-12-01

Figure 4.3 Table with parent - child relationship of the icebergs

	icebergid character var	lifetime real	travel_distan real	average_spe real	direction real	appear_lat real	appear_lng real	disappear_la real	disappear_ln real	appear_time character var	disappear_tir character var	appear_size real	disappear_size real
1	A01	196	1313.68	6.70243	85.4244	-34.2	-55.9965	-26.3	-50.1891	1978-10-26	1979-05-10	3863.69	824.254
2	A02	343	1478.16	4.30951	-31.6608	-47.8	-57.9978	-32.8	-54.5953	1978-11-15	1979-10-24	1030.32	1030.32
3	A03	132	144.704	1.09624	144.006	-43.9	-58.2979	-44.5	-59.4985	1979-02-09	1979-06-21	858.597	858.597
4	A04	0	0	0	0	-30.8	-54.1949	-30.8	-54.1949	1979-10-24	1979-10-24	1030.32	1030.32
5	A05	5	48.4061	9.68121	48.9418	-50.4	-61.099	-51	-60.7989	1979-11-01	1979-11-06	515.159	515.159
6	A06	70	530.402	7.57717	7.42478	-60	-63.7995	-53.4	-61.5991	1979-11-08	1980-01-17	858.597	858.597
7	A07	0	0	0	0	-60	-66.9999	-60	-66.9999	1979-11-18	1979-11-18	858.597	858.597
8	A08	42	648.47	15.4398	63.1924	-36.5	-51.6917	-29	-49.4877	1979-12-27	1980-02-07	858.597	858.597
9	A09	0	0	0	0	-31.5	-50.3895	-31.5	-50.3895	1980-01-17	1980-01-17	686.878	686.878
10	A10	0	0	0	0	-39.2	-50.7902	-39.2	-50.7902	1980-02-14	1980-02-14	1030.32	1030.32
11	A11	15	124.046	8.26974	-5.76502	-43.8	-71.6	-44.3	-70.5	1980-05-06	1980-05-21	858.597	858.597
12	A12	71	0	0	0	-60.2	-66.5998	-60.2	-66.5998	1980-07-23	1980-10-02	515.159	515.159
13	A13	168	393.646	2.34313	-8.62004	-27.3	-74	-38.5	-73.8	1980-08-13	1981-01-29	1030.32	1030.32
14	A14	42	0	0	0	-36	-72.3	-36	-72.3	1980-12-17	1981-01-29	2060.63	2060.63
15	A15	132	848.849	6.43068	-40.7752	-47.7	-72.6	-53.8	-66.0998	1981-03-27	1981-08-06	1236.38	1236.38
16	A16	180	1247.57	6.93095	163.197	-50.9	-59.6986	-37.2	-55.2959	1981-11-21	1982-05-20	1803.05	1803.05
17	A17	40	408.45	10.2113	-100.364	-50.2	-64.2996	-49.2	-60.999	1983-04-21	1983-05-31	1974.77	1974.77
18	A19	420	1442.89	3.43544	163.743	-44	-77.9	-58.7	-67.2999	1983-04-21	1984-06-14	700.616	700.616

Figure 4.4 Table with summary of the trajectory data

In Figure 4.4, the summary table with a summary of the trajectory data is shown. In this table, the properties of each trajectory such as the lifetime, the total travel distance, the average speed and the direction are calculated based on the original data (see Figure 4.2 and chapter 2). This table also stores the appearance time, the disappearance time, the size of the iceberg at the appearance time and the size of the icebergs at the disappearance time.

Based on this table, a classification process was performed on some attributes in the data (see the Figure 4.5). For instance, the lifetime attributes are classified into 3 classes, shortlife – indicates the iceberg's lifetime is less than 90 days, mediumlife - indicates the iceberg's lifetime is less than 365 days but more than 90 days and longlife - indicates the iceberg's lifetime is more than 365 days. The travel distance and the average speed were also classified into 3 classes. Distance consists of 3 classes, short distance - indicates the iceberg's total travel distance is less than 1000 kilo-meters, medium distance - indicates the iceberg's total travel distance is less than 3000 kilo-meters but more than 1000 kilo-meters, long distance - indicates the iceberg's total travel distance is more than 3000 kilo-meters. Average speed consists of 3 classes, slow - indicates the iceberg has an average speed of less than 10 kilo-meters per day, normal - indicates the iceberg has an average speed of less than 20 kilo-meters per day but more

than 10 kilo-meters per day, fast - indicates the iceberg has an average speed of more than 20 kilo-meters per day.

appear_size_type character varying(2	disappear_size_type character varying(255)	lifetime_type character varying(25	travel_distance_type character varying(255)	average_speed_type character varying(255
medium	small	mediumlife	mediumdistance	slow
small	small	mediumlife	mediumdistance	slow
small	small	mediumlife	shortdistance	slow
small	small	shortlife	shortdistance	slow
small	small	shortlife	shortdistance	slow
small	small	shortlife	shortdistance	slow
small	small	shortlife	shortdistance	slow
small	small	shortlife	shortdistance	normal
small	small	shortlife	shortdistance	slow
small	small	shortlife	shortdistance	slow
small	small	shortlife	shortdistance	slow
small	small	shortlife	shortdistance	slow
small	small	mediumlife	shortdistance	slow
medium	medium	shortlife	shortdistance	slow
small	small	mediumlife	shortdistance	slow
small	small	mediumlife	mediumdistance	slow
small	small	shortlife	shortdistance	normal

Figure 4.5 Classified attribute from the summary trajectory table

4.3. PHP and the connection to the POSTGRESQL database

One of the most challenging issues that we have to face in this research is how to connect the client with the database. In other words is how to exact data from the database and transfer the data to the client side. PHP and the Apache server are used for this purpose. The model is shown in Figure 4.6. Iceberg data in the database are read by PHP through the Apache server. Then the data will be restored in the KML files. Finally, the KML files will be transferred via the internet to the client side.

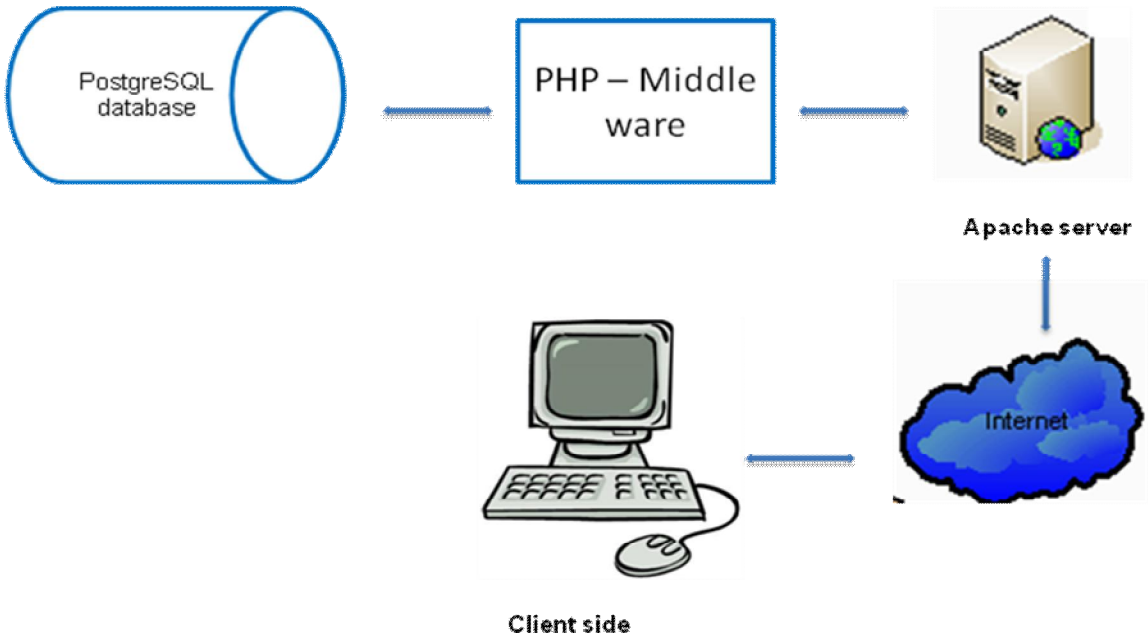

Figure 4.6 Model of connecting with the database using PHP and Apache server

4.3.1. PHP

PHP stands for Hypertext Preprocessor. It is a server side scripting language, like Microsoft's Active server page (ASP). It is open source software and widely used for making dynamic and interactive Web pages. PHP can be downloaded freely from the PHP official web page [URL 20]. It can run on most of popular platform, such as Window, Linux, etc.

PHP files can contain HTML and script inside. PHP files' extensions are ".php", ".php3", or ".phtml". It supports many databases, such as PostgreSQL, MySQL, Oracle, etc. PHP can work with most of the recent servers (Apache, IIS, etc.).

4.3.2. Apache

PHP requires a server application to transcript its code. In our case, we use an Apache server to run PHP. Apache is a powerful, flexible, HTTP/1.1 compliant web server. It supports the latest protocols (HTTP/1.1 (RFC2616)) and is highly configurable and extensible. Since Apache is open-source and free, users or developers can extend it by using the Apache module API. It can operate on most recent operating systems, such as Windows, UNIX, etc.

4.3.3. Using PHP to connect to the PostgreSQL database

To connect to PostgreSQL database using PHP, a `pg_connect` function with connection strings that consists of host name, name of the database, user and password was performed. The following codes describe how to connect to PostgreSQL using PHP:

```
<?php
```

```
//connect to a database named "PostgreSQL" on the host "127.0.0.1"
with a username and password
 $dbh = pg_connect("host=127.0.0.1 dbname=PostgreSQL
user=PostgreSQL password=hoanglong");
 if (!$dbh) {
     die("Error in connection: " . pg_last_error());
 }
```

4.3.4. Using PHP to generate KML file from the PostgreSQL database for visualizing iceberg data

As discussed above, KML files were chosen for this research. However, to extract a KML file from the database and transfer it to the client is a challenge. Hence, there is a need to use PHP as a mid-ware to generate KML from the PostgreSQL database. The KML files that are generated have to be in the structure that is required by timemap.js. The following lines describe a part of a KML file:

```
<kml>
<Document>
<Placemark>
<name>A01</name>
<description>3858.64</description>
<TimeStamp><when>1978-10-25</when></TimeStamp>
<Point><coordinates>-34.2,-56</coordinates></Point>
<tags>A01,medium</tags>
<theme>orange_medium</theme>
</Placemark>
```

Emphasizing the appearance and disappearance of an iceberg is one of the main tasks of our research. There are several ways to emphasize the appearance and disappearance of an iceberg as we discussed in chapter 2. In this research, items in the database are classified into 3 classes: Appearance, Disappearance and Intermediate position inside the KML file. Next, colour is used as a variable to differentiate between the appearances, the disappearances of the iceberg and intermediate positions. The appearance of an iceberg is illustrated by orange dot. A disappearance is illustrated in green colour, while intermediate positions are in blue. As mentioned above, icebergs were also classified in to 3 classes: small, medium and big. Hence, a point has to have two properties, colour and size. To solve this, a JavaScript file named icon.js (see Appendix B.4) that contains all the needed icons was created.

Furthermore, in our prototype we do not only have points objects (icebergs positions at specific times) but also lines connecting the points (movements of the icebergs). Trajectories are classified into two classes: normal movement and split movement. In some exploration tasks such as comparing the trajectories of several icebergs that existed at the same time, it may be useful to differentiate icebergs trajectories by different colours. Hence for normal movement, the colour of each trajectory is chosen randomly between 5 five colours, namely blue, orange, green, purple and yellow. It may partly help to perform tasks above. The red colour is used to illustrate split movements. This colour system is both applied in the timeline and the map view.

In this research, two KML files are generated (see Appendix C.1 and C.2). The first KML file contains the first appearance, disappearance and the major direction of movement of each iceberg. Major direction or major movement is the direction from the starting point to the destination. The second KML file contains the raw data of icebergs and the movements (lines). The line objects are generated based on

the raw data of icebergs. The red lines that illustrate the split events are generated from the parent and child relationship table (see Figure 4.3).

4.4. Improving functionality of timemap.js

There are some limitations in the functionalities that are provided by timemap.js and some required functions that are needed in visual exploration of trajectory data are missing. For example, timemap.js does not provide real animation or the function to control the zoom levels. Hence, the timemap.js library needs to be extended.

4.4.1. Control of the zoom level

To visualize different KML files at different zoom levels, some triggers are required. Whenever the zoom level of the map is changed, a function will be performed. GEventListener function that is provided by Google Map API is suitable for this purpose. Whenever the zoom level of the map reaches a certain zoom level, one KML file is shown and the other will be hided.

```
GEvent.addListener(tm.map, "zoomend", function(oldLevel, newLevel)
{

if (newLevel>=4)
{
tm.each(function() {
tm.datasets.Iceberg2.visible = false; // hide the dataset
tm.datasets.Iceberg1.visible = true; // show the dataset
tm.filter("map"); // filter on map
tm.filter("timeline"); // filter on timeline
tm.timeline.layout(); // update the timeline
});
document.getElementById("details_1").disabled=false;
document.getElementById("overview_1").disabled=true;
}
if (newLevel<4)
{
tm.each(function() {
tm.datasets.Iceberg2.visible = true;
tm.datasets.Iceberg1.visible = false;
document.getElementById("details_1").disabled=true;
document.getElementById("overview_1").disabled=false;
tm.filter("map");
tm.filter("timeline");
tm.timeline.layout();
});
```

4.4.2. Creating animation

Animation is a useful technique in visual exploration of trajectory data (Blok, 2005). However, timemap.js does not provide animation functions such as play, stop and slider to control the speed of animation. It only provides a kind of simulative animation by moving the timeline. Only the items on the visible range of the timeline are shown on the map view. For real animation, creating animation functions is needed, but it is also a big challenge for us. To deal with this problem, we used “JavaScript Timing events” and the setCenterVisibleDate() function provided by timemap.js.

a. JavaScript Timing events

JavaScript allows executing a code after a time interval that is called "JavaScript Timing events". There are two methods in "JavaScript Timing events": the first one is setTimeout() that executes a code after a time interval and the other one is clearTimeout() that clears the setTimeout().

b. SetCenterVisibleDate

Timemap.js provides a function named setCenterVisibleDate(). This function is used to move timeline to a specific time. For example:

```
time= new Date(2009,09,09);
setCenterVisibleDate(time);
```

After executing these codes, the timeline will move to 09-09-2009.

c. Creating animation functions

One of the most important tasks when creating animation functions is providing slider for users to control the speed of the animation. In our project, the slider's code was borrowed from the JavaScript slider that was created by Erik Arvidsson (URL 21). The codes below describe how to create a slider on our web application:

```
<div class="slider" id="slider-1" tabIndex="1">

      <input class="slider-input" id="slider-input-1"/>
</div>
<script type="text/JavaScript">
var    s    =    new    Slider(document.getElementById("slider-1"),
document.getElementById("slider-input-1"));
s.onchange = function () {
      document.getElementById("h-value").value = s.getValue();
      document.getElementById("h-min").value = s.getMinimum();
      document.getElementById("h-max").value = s.getMaximum();
      s.setValue(s.getValue());
      s.setMinimum(s.getMinimum());
      s.setMaximum(s.getMaximum());
};
s.setValue(50);
window.onresize = function () {
      s.recalculate();
};
```

Due to the temporal irregularity of iceberg data, two sliders are created to control the speed of the animation. The first slider controls the speed of the change between two scenes. The second slider controls the time interval step between two scenes. For example: when users set the time interval step to 30, the next scene will be the scene of 30 days after the previous one.

We implemented additional controls by creating a JavaScript file that contains play(), pause() and reset functions. The codes are showed below:

```
// play animation
function play(){
d1= tm.timeline.getBand(0).getCenterVisibleDate();
step=s.getValue();
time=s3.getValue();
time=(time*100)+400;
d1.setDate(d1.getDate()+ step);
```

```
tm.timeline.getBand(0).setCenterVisibleDate(d1);
t=setTimeout("play()",time);
};
// pause animation
function pause(){
clearTimeout(t);
};
// reset animation
function reset(){
      d = tm.eventSource.getEarliestDate();
      tm.timeline.getBand(0).setCenterVisibleDate(d);
};
```

Finally, normally trajectory data are collected over a long time. Iceberg data are collected from 1978 until now. Sometime users just want to focus to a specific time. They can do it by moving the timeline until it reaches the time. If the time space between the recent time and the wanted time is too long, it will take time to move the timeline. Hence, a function "gototime" is provided to let users jump to a specific time.

```
function gototime()
{
      gotime=document.getElementById('time_go').value;
      gotime= new Date(gotime);
      tm.timeline.getBand(0).setCenterVisibleDate(gotime);
};
```

4.4.3. Improving kml.js

Kml.js is a loader file. This is a loader class for KML files. Timemap.js supports all geometry types (e.g. point, polyline, polygon, and overlay) and multiple geometries. By default, the loader only loads some tags in a KML file (e.g. name, theme, description, timespan and timestamp). In our research, for the filtering function, `<tags>` tag needs to be read and stored as a variable.

```
data.options.tags = getTagValue(pm, "tags");
```

4.4.4. Creating filtering functions

After overview, the next stage of the Visual Information Seeking Mantra is zoom and filter. As mentioned in chapter 2, users might want to explore what the behaviour or movement of a particular iceberg during a particular time is. Timemap.js provides the filtering function based on the `<tags>`. We provide several options for users to filter the data such as filter by lifetime, filter by travel distance, filter by average speed, filter by size and filter by name.

After performing this original function, the timemap.js will eliminate from user's view the items that do not meet the requirements. However, the timeline is not changed. Hence, two lines of code are added to move the center of the timeline to the earliest event that meets the requirements. The last two lines of the filtering function did the trick:

```
function setSelectedTag(select) {
var idx = select.selectedIndex;
window.selectedTag = select.options[idx].value;
// run filters
```

```
tm.filter('map');
tm.filter('timeline');
// you'll need this to make the timeline update
tm.timeline.layout();
// set the center of timeline to the first events after filtering
d2 = tm.eventSource.getEarliestDate();
tm.timeline.getBand(0).setCenterVisibleDate(d2);
};
```

Next, some combo boxes were created to give users options for filtering such as a combo box that contains all the name of icebergs. Those combo boxes were dynamically generated by using PHP connecting to the database then get the value for the combo boxes. In the database, many records belong to the same icebergs. Hence, to get a unique value for each iceberg in the combo box, some tricks are used:

```
<?php
$dbh  =  pg_connect("host=127.0.0.1  dbname=postgres  user=postgres
password=hoanglong");
 if (!$dbh) {
     die("Error in connection: " . pg_last_error());
 }
 // execute query
 $sql = "SELECT * FROM iceberg_full_A";
 $result = pg_query($dbh, $sql);
 if (!$result) {
     die("Error in SQL query: " . pg_last_error());
 }
echo "<form name='ge' action='' method='post'>";
echo "<select name='menu' onchange='setSelectedTag(this);'>";
echo "<option selected value=''>Select size</option>";
$os = array();
while ($row = pg_fetch_array($result)) {
            $comb=$row['icebergid'];
            settype($comb,"string");
            $c=$comb;
            if (in_array($c, $os, FALSE)) {
                  echo "";
                  }
            else {echo "<option value=$c>".$c."</option>";
                  array_push($os,$c);}
      }
echo "</select>";
?>
```

4.4.5. Timemap.js interface

Timemap.js is the combination of Google Maps API and SIMILE Timeline. Hence, to make use of timemap.js, we have to include the timemap.js and SIMILE timeline-api.js in our application. There are several versions of SIMILE Timeline, in this research we use SIMILE Timeline version 2.3.0.

First, in the <head> tag of our page, we include the Google maps script (the API key from Google), the SIMILE Timeline script, and the timemap.js script.

```
<script
src="http://maps.google.com/maps?file=api&amp;v=2&amp;key=ABQIAAAAS
```

```
I0kCI-
azC8RgbOZzWc3VRRarOQe_TKf_51Omf6UUSOFm7EABRRhO0PO4nBAO9FCmVDuowVwRO
Lo3w" type="text/JavaScript"></script>

<script type="text/JavaScript"
src="../../timeline_2.3.0/src/webapp/api/timeline-api.js"></script>
<script src="../timemap1.6.js" type="text/JavaScript"></script>
```

After that, we had to set up the time map initial, defining the parameter to initialize the map and timeline.

```
<script type="text/JavaScript">
var tm;
function onLoad() {
tm = TimeMap.init({
mapId: "map", // Id of map div element (required)
timelineId: "timeline", // Id of timeline div element (required)
options: {
eventIconPath: "../images/",
},
datasets: [
{
id: "Iceberg1",
type: "kml",
options: {
type:"kml",
url: "kml_final.php" // KML file to load
}
},
```

Timemap.js requires defining the bands of the timeline before the timeline can start. There are two bands on the timeline. The iceberg positions are recorded every 1-5 days (Ballantyne and Long, 2002). Hence, the top band's time interval is set in month. Antarctic iceberg positions are collected from the late 1970s until now (see chapter 2). Therefore, we decided to set the bottom band to decade to give the users an idea about the time span for the whole data set. Figure 4.7 shows the timeline of the system.

```
bandIntervals: [

Timeline.DateTime.MONTH, Timeline.DateTime.DECADE
],
```

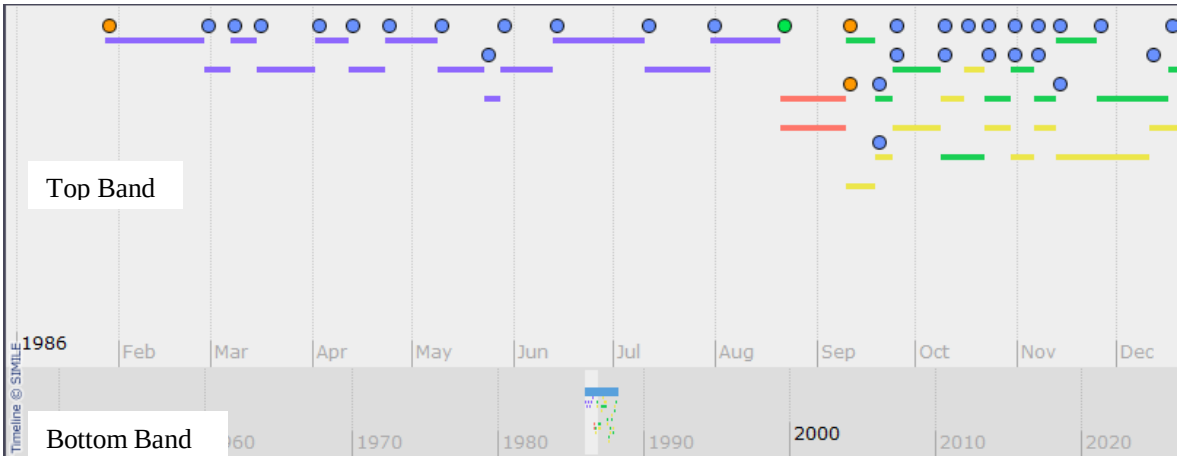

Figure 4.7 The timeline

To set the starting time of the timeline to the earliest event and to make visible only one KML file at the start of the application requires adding a datadisplayFunction() function.

```
dataDisplayedFunction: function(tm) {
tm.datasets.Iceberg1.visible = false;
date_ini=tm.eventSource.getEarliestDate();
tm.timeline.getBand(0).setCenterVisibleDate(date_ini);
tm.filter("map");
tm.filter("timeline");
tm.timeline.layout();
}
```

Next, the size of the timeline view and the map view need to be defined before the application can start. They are set inside the <style> tag.

```
div#timelinecontainer{ height: 310px; }
div#mapcontainer{ height: 300px; }
```

4.4.6. Improving the timeline

A limitation of the SIMILE timeline is events of an iceberg on the timeline are positioned randomly (as shown in Figure 4.7). It makes it difficult for users to get information. Hence, the timeline should be improved.

To solve this problem, the SIMILE timeline API has to be dig. During the investigation of the SIMILE timeline API, we found that the JavaScript file named “original-painter.js” contains the JavaScript that controls the positions of all the events on the timeline. The objective was to connect all events of the same iceberg by one line. It will not only help users get the information easier, but also give them information about the lifetime of icebergs. The length of the line between appearance and disappearance on the timeline can be considered as the lifetime of an iceberg. Because the width of the timeline was set to 300px, thus the timeline is separate into 10 lines to fit it. Every event that belongs to the same iceberg is displayed on the same line. Figure 4.8 shows the improved timeline. Some JavaScript lines of code are added into the original-painter.js to do the trick:

```
for (i=0;i<=60;i++)
```

```
        {
        m=(i+1)%11;
        if (evt._text==myIce[i] && m==1)
        {
              t=0;
        }
        else if (evt._text==myIce[i] && m==2)
        {
              t=20;
        }
        …
        }
        else if (evt._text==myIce[i] && m==10)
        {
              t=180;
        }
        else if (evt._text==myIce[i] && m==0)
        {
              t=200;
        }
        }
…
// for points
iconDiv.style.top = t + "px";
…
// for label
labelDiv.style.top = t + 10 + "px";
…
// for lines
tapeDiv.style.left = startPixel + 8 + "px";
tapeDiv.style.width = tapeWidth - 12 + "px";
tapeDiv.style.height = tapeHeight + "px";
tapeDiv.style.top = t + 5 + "px";
```

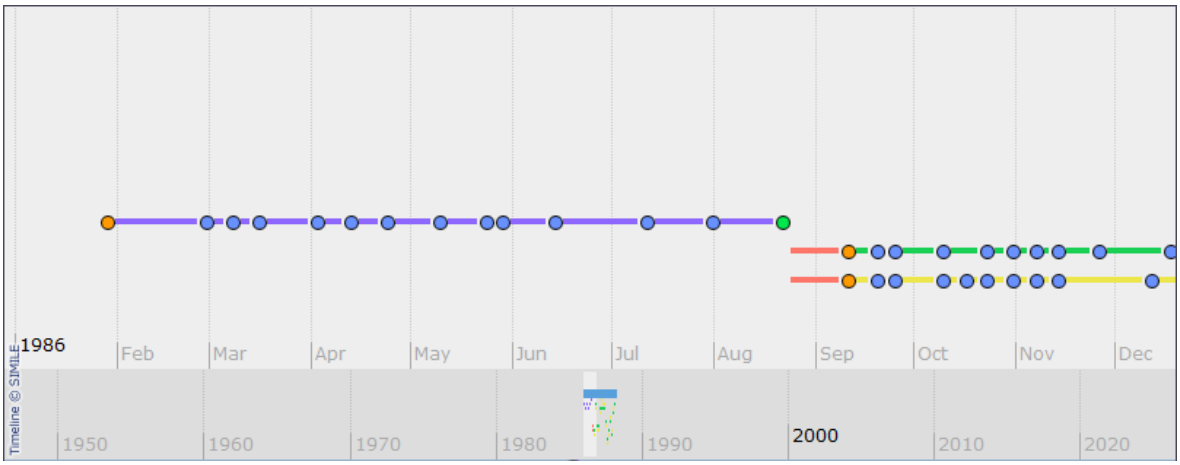

Figure 4.8 Improved timeline

4.4.7. Improving the infowindow

Timemap.js provides pop-up window for users to see the detail of each item in the map view. Originally, the pop-up window displays all the information in the `<description>` tag as shown in Figure 4.9. All the information is displayed on the same line and if out of space, the string will jump to the next row. To give a better view to the users, the information layout should be controlled. To solve this, some

codes in the timemap.js JavaScript file are modified. The result after modifying the code is shown on Figure 4.10.

```
TimeMapItem.openInfoWindowBasic = function() {
    var html = this.opts.infoHtml;
    // create info window
    if (html === "") {
        html = '<div class="infotitle">' + this.opts.title +
'</div>';
        if (this.opts.description !== "") {
                des=this.opts.description;
                description=des.split(";");
                de=0;
                while (de < description.length)
                 {
                html += '<div class="infodescription">' +
description[de] + '</div>';
                de+=1;
                }
        }
            lo=this.getInfoPoint();
            html += '<div class="info"> position: ' + lo +
'</div>';
    }
```

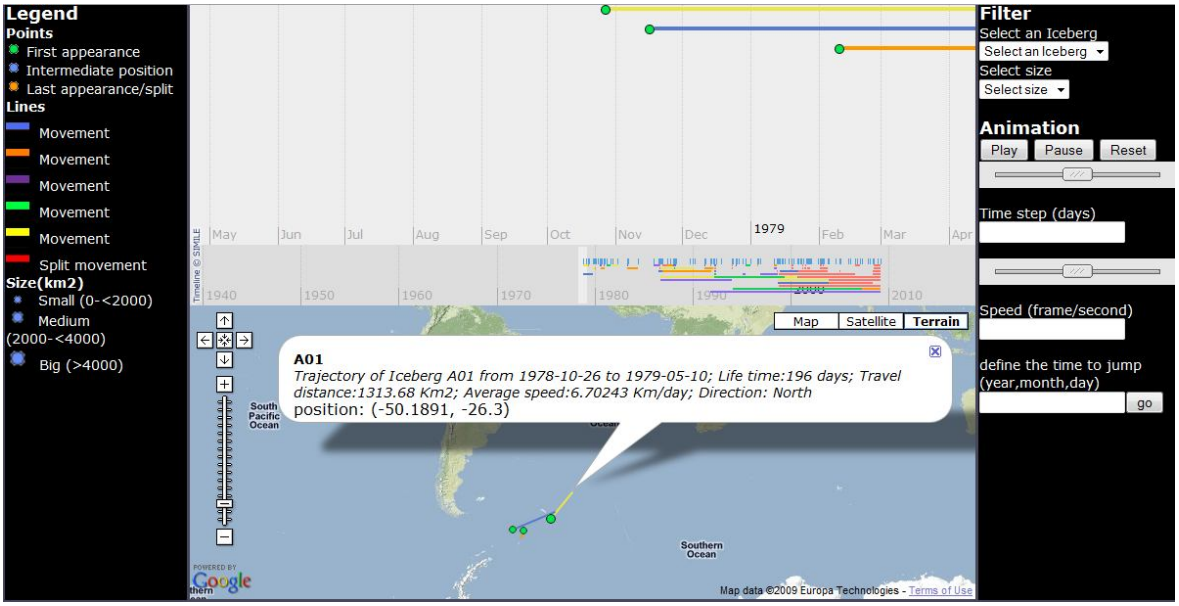

Figure 4.9 Original pop-up window

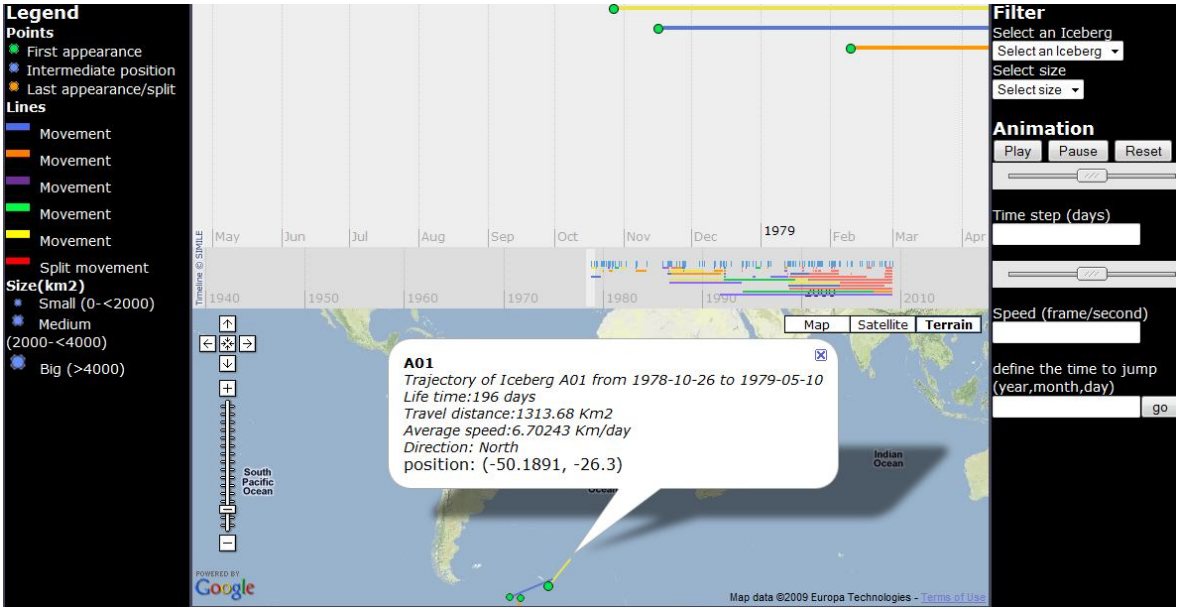

Figure 4.10 Improved pop-up window

4.4.8. Integrating components

The result of integrating above components using HTML code, PHP code, JavaScript code and Cascading Style Sheets is shown in Figure 4.11. The complete code for the prototype is shown in the Appendix C.3.

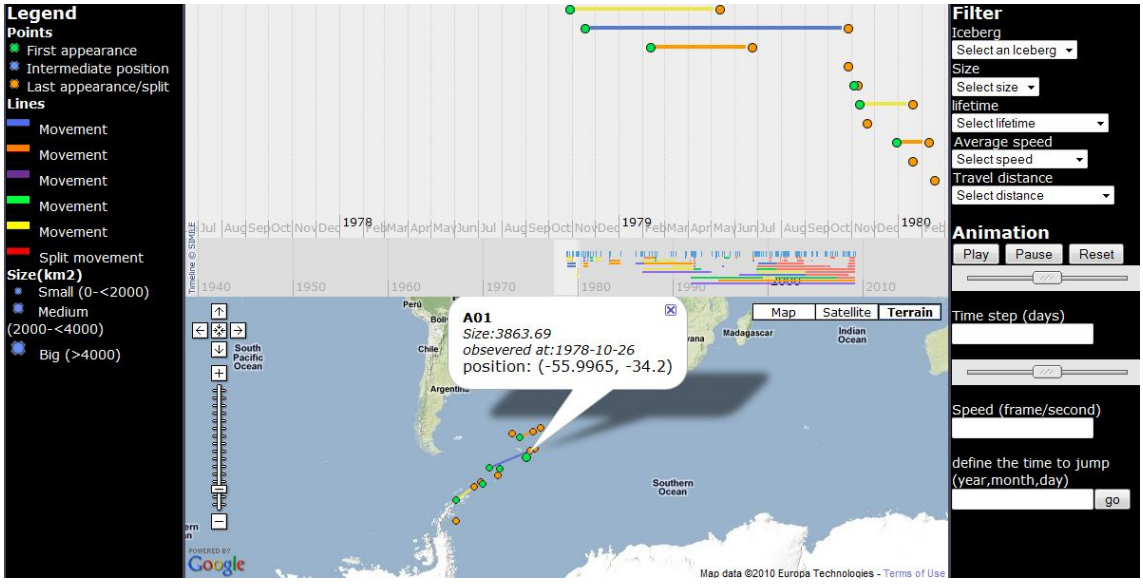

Figure 4.11 Final GUI of the prototype

4.5. Summary

The implementation process of the prototype for visualizing iceberg data as a case is presented in this chapter. The implementation process was based on the tasks that were discussed in chapter 2 as well as on the functionalities of the timemap.js. Most of the views (e.g. map view, timeline, animation, attribute

view) and functions (e.g. zoom, pan, filter, play, stop, etc.) that were discussed in chapter 2 were implemented. However, to emphasize the appearance and disappearance events we used colour variables instead of using dynamic visualization variables or dynamic legend. Improvements of the functionalities as well as improvement the GUI of the original timemap.js are described as well. Nevertheless, the functionalities of the prototype should be tested and evaluated. The next chapter will describe evaluation of the prototype.

5. Evaluation of the prototype

The previous chapter described the design and implementation of the prototype. This chapter describes the usability testing of the prototype.

5.1. Objective and selection of methods

As described before in chapter 4, our prototype provides a map view, timeline, legends and some functions for users to explore iceberg data. The main objective of the usability testing described here is to discover what works well, what has not be used and what has to be improved; such an early evaluation is usually followed by another evaluation after improvements have been made, but within this research it is the end phase that leads to conclusions and recommendations.

According to the standard ISO 9241-11 of the International Standardization Organization (URL 22), usability is defined as "The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use".

There are several usability testing methods such as focus group, interview, think aloud, video recording, etc. In this research, to assess the prototype from the usability perspectives that are mentioned above, several methods are used together, eye tracking, a questionnaire and audio/video recording.

The questionnaire was chosen due to its characteristics. A questionnaire is a simple method for collecting data from the individual participant by giving a set of questions. At the evaluation stages, questionnaires can give answers about the satisfaction of the users with the application, problems encountered, ideas for improvement, etc. Generally, a questionnaire can be provided in four different ways: as handout, post mail, email or web-based. In this research, the questions will be given to the participant by handout.

Fortunately, we were provided an opportunity for using eye tracking in our research. Eye tracking is used widely in usability research in recent years (Manhartsberger and Zellhofer, 2005). However, using eye tracking data for test the usability of the prototype was not our main objective. We hope that we can extract some extra information about the usability issues of our prototype such as whether or not animation has been used to answer questions about changes. From a usability point of view, eye tracking is also useful for testing hypotheses about design (Cooke, 2005). Eye tracking can also reveal hidden cognitive processes. For example, certain usability testing problems can only be analyzed by recognizing when, where and at what users have looked, e.g.: the reason that users fail to spot a particular element may that another element distracts them. Çöltekin et al (2009) proved that eye tracking can enhance usability testing by combining with others traditional techniques to collect user's information such as interview and questionnaire.

Furthermore, a video recorder that recorded the computer screen is also used in this research. Due to the lack of experiences in conducting an eye tracking test, video data has served as back-up source, in case we have problem with the eye tracking.

5.2. Eye tracking

Poole and Ball (2005) define eye tracking as "a technique whereby an individual's eye movements are measured so that the researcher knows both where a person is looking at any given time and the sequence in which their eyes are shifting from one location to another". Eye tracking allows experimenters to determine eye movement and eye-fixation patterns of a person. It provides a valuable source of information about user's behaviours. Eye tracking can reveal information about human behaviours or usability problems which may not be gained by conventional usability data collection methods (Cooke, 2005).

According to DeSantis et al (2005) there are two types of eye tracking systems: head mounted eye tracking systems and stationed eye tracking systems. In this research, we used a stationary system. This system allows participants to move their head freely during the test.

The eye tracking system typically records eye location (horizontal and vertical coordinate), the time and the duration of each sample. Depending on the sampling rate, and the duration of the session, the amount of data may become very large. In data analysis, the first step is normally to differentiate between fixations and saccades. Fortunately, nowadays, there is some software available that provides analysis tools for extracting the fixations and saccades from the data. In our testing session, FaceLAB version 4.5 was used for the calibration and recording of the data. GazeTracker was used for data analysis and visualization of the eye tracking data. FaceLAB and GazeTracker were run in two different computers. The prototype was on the same computer as GazeTracker. The eye tracking system that was used in this research is shown in Figure 5.1.Two cameras were used to track the participant's eyes (see number 2 and 4 in the figure). 3 infrared lights were used to recognize the participant's head (see number 1, 3 and 5 in the figure).

Figure 5.1 Eye tracking system

5.3. Conducting the test

We cooperated with one of my classmates, Rozita Razeghi who does Msc research with the title "Usability of eye tracking as an user research technique in geo-information processing and dissemination" (Razeghi, forthcoming).

5.3.1. Objective and questions for test the prototype

The objective of the test is to evaluate the usability of a prototype design solution for the visual exploration of iceberg data using timemap.js library. The evaluation is based on the use of functions and views while the participants tried to answer questions. The questions are separated into 3 groups according to the Visual Information Seeking Mantra "overview, zoom and filter, details on demand":

Group 1: Questions related to overview

Q1: In which zone, were there more icebergs appearances, zone 1 (inside the blue box) or zone 2 (inside the yellow box) (see Figure 5.2)?

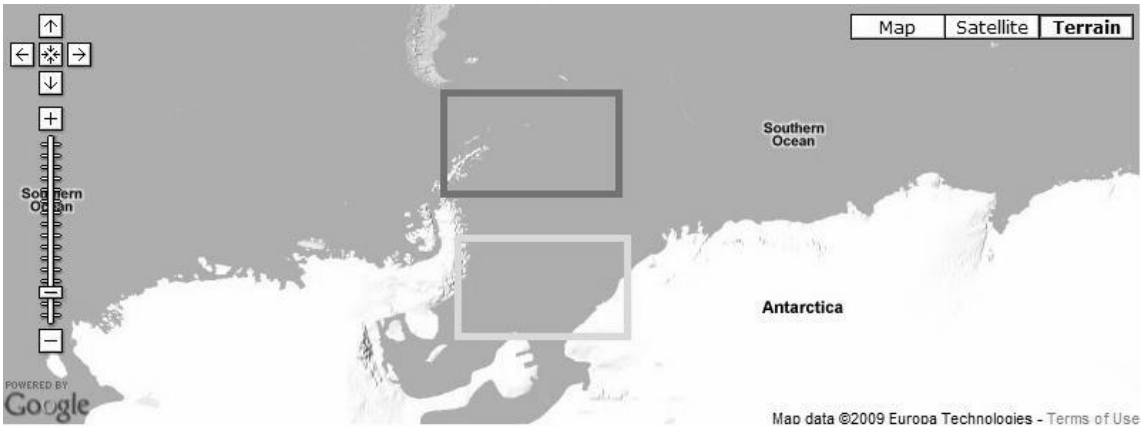

Figure 5.2 Question 1

Q2: In which year were more appearances of icebergs?
Q3: Which iceberg has the longest lifetime?

Group 2: Filtering and zooming

Q1: List 3 icebergs that had slow average speed (less than 10 km/per day)?
Q2: Where is the iceberg A01 at 11/01/1979? (Provide the latitude and longitude position)
Q3: Compare the sizes of icebergs A20A and A20B at their appearances. Is A20A bigger than A20B?

Group 3: Details on demand

Q1: Where and when did the iceberg A15 disappear? (Provide the latitude and longitude position)
Q2: How much did the size of iceberg A19 decrease between its appearance and its disappearance?
Q3: Did iceberg A20A's movement cross the movement of A20B?

5.3.2. Test procedure

Firstly, we sent emails to ITC staff in the Geo-Information Processing Department and Msc students from the Geoinformatic 2008 course to ask whether they want to be a test participant.

Before the test, a questionnaire was given to the participants to ask them some personal information (see Appendix E1). The purpose of the questionnaire is to know about the participant's background. It will help for the data analysis later.

After receiving response from participants, eight participants performed the test: 1 Msc student and 1 Master student from the Geoinformatic course, 1 Msc student from the Water Resources Management Department and 3 PHD students for Geo-Information Processing department. They have different levels of knowledge about the trajectory data and the iceberg data. They also have different experiences with using an interactive map.

For the usability testing, we installed the Apache server as well as PostgiSLQ on the computer that would be used for the test. Due to the fact that many data are used, the computer performs slowly. As mentioned before, the computer does not only run the prototype but also the GazeTracker software.

GazeTracker is considered a heavy software (it needs many of computer's resources). Hence, we decided that only iceberg data for ten years (from 1978 to 1988) are used for the test.

The usability testing session was executed on the 1st of Feb 2010 in a testing room at Twente University. The session started at 9 A.M and finished at 6 P.M. Each participant did the test individually. The average time spent by a test participant is about 1 hour. The whole test contains six steps:

1) Welcome the participants and give an introduction of the test and the prototype.
2) Ask the participant if he/she has any question about the procedure or the prototype.
3) Participants have about 5-15 minutes to get familiar with the prototype.
4) Next, we do the calibration for the eye tracking system. Participants execute the tasks to answer the questions. Questions are written down on a paper that will be stuck near the computer screen. During the test, participants could move their head a little bit. However, they could not move their body or leave their chair. When the participants find the answer for each question, they will say aloud the answer.
5) The data about eye's gazes, eye's fixations and the voice of the participants are recorded.
6) Finally, a short questionnaire is conducted.

5.4. Results

Due to the limitations of the system, during the data saving process, we lost two participants data.

5.4.1. Effectiveness

Effectiveness of the prototype can be assessed by answering the question: can users correctly complete tasks with the prototype. Table 5-1 shows the summary of the correctness of the answers that were provided by six participants. Most of the answers are correct. Question 3 seems to be the most difficult one, 4 participants gave the wrong answers to this question. The reasons why participants gave wrong answers may be revealed by analyzing eye tracking data or the recorded video.

	Person 1	Person 2	Person 3	Person 4	Person 5	Person$	Overall
Question 1	R	R	R	R	R	R	100%
Question 2	R	R	R	R	R	R	100%
Question 3	W	W	W	W	R	R	33.3%
Question 4	R	R	R	R	R	R	100%
Question 5	R	R	R	R	R	R	100%
Question 6	R	R	R	R	R	R	100%
Question 7	R	R	R	R	R	R	100%
Question 8	R	R	R	R	R	R	100%
Question 9	R	W	R	R	R	R	83.3%
Overall	89%	78%	89%	89%	100%	100%	

Table 5-1 Summary of the correctness of the answers provided by six participants

Table 5-2 shows the average correctness of the answers for 3 question groups. In this test, participants had to answers to all the provided questions. The overall correctness of all the answers is 90.74%. The average correctness reflects the effectiveness of the prototype.

Group	Related questions	Average correctness
G1-Overview	1-3	77.80%
G2-Zoom and filter	4-6	100%
G3-Details on demand	7-9	94.40%

Table 5-2 Summary of the average correctness of the answers of each question group

5.4.2. Efficiency

Efficiency refers to the amount of effort that users need to achieve their purpose. In this research, we provided participants questions about the difficulty of each group tasks in the post-test questionnaire (see Appendix D.1). The levels of difficult of each question group are rated by the participants on a scale from 1 to 5 (see Table 5-3). With questions of group 1 (overview) and group 2 (zoom and filter), the average score is 4. Group 3 (details on demand) has the lowest average score: 3.83. The reason might be more actions were required (e.g. do zoom to see the details data, click on more than 1 item to get the information, do a calculation, etc.) to find answers for the question in group 3. The overall score is 3.94.

Test person	Overview	Zoom and filter	Detail on Demand
Test person 1	5	4	4
Test person 2	4	3	4
Test person 3	4	4	2
Test person 4	4	4	4
Test person 5	3	4	4
Test person 6	4	5	5
Overall	4	4	3.83

Table 5-3 Summary of the level of difficulty experienced in each group of questions (1-very difficult; 2-difficult; 3-normal; 4-easy; 5-very easy)

5.4.3. User's satisfaction

Satisfaction is the opinion of the users about the application, is the application good or bad. After finishing all the tasks, a short questionnaire was provided to the participants to get information about their satisfaction with the prototype (see Appendix D.2). In the questionnaire, the satisfaction levels are classified into five levels. 4.33 is the average score of the satisfaction levels of all participants (see Table 5-4). The average score reflects the satisfaction of the users of the prototype.

Test person	Satisfaction level
Test person 1	4
Test person 2	4
Test person 3	5

Test person 4	4
Test person 5	4
Test person 6	5
Overall	4.33

Table 5-4 Summary of user's satisfaction (1- very poor; 2- poor; 3-normal; 4-good; 5-very good)

5.4.4. Eye tracking Data

As mentioned above eye tracking data can give information that many other usability testing methods cannot give. There are several outputs that can be extracted from the eye tracking data. According to Poole and Ball (2005) "the process of inferring useful information from eye movement recordings involves the Human Computer Interaction researcher defining "areas of interest" over certain parts of a display or interface under evaluation, and analyzing the eye movements which fall within such areas". Hence, it can give an objective evaluation about the visibility, meaningfulness and position of the specific elements of the interface. Based on that, the design of the interface can be improved (Goldberg and Kotval, 1999). In this research, we defined 4 look zones or areas of interest. The first zone is the timeline, the second one is the map view, the third zone is the functions zone and finally the legend zone. The reason why only 4 zones were defined is because of the relatively low accuracy of the system at Twente University. With small elements such as a play button in the prototype, in the test, participants may look at this button, but the output data may not show that the participants looked at the button.

Figure 5.3 shows the average of the total test time spent in each look zone (the time that participants spend on each zone in seconds and the percentage of the total time). The sum of the percentages in this figure is not 100%, because, in the test, participants did not only look on the screen but sometimes they had to look at the questions on the paper that was stick near the computer's screen. Furthermore, some data also might get lost due to an error of the eye tracking system (e.g. while participants look at a position near the frame of the computer's screen, the eye tracking system may record the eye gaze position as being outside the screen).

Before the test, we expected that participants will spend most of their attention on the timeline and the map view, because most of the questions that were provided required the participants to find the information in both the map view and the timeline. Results show that participants spent most of their attention on the map view (19.5%), while the timeline attracted 17.08% of total times. The amount of time that participants spent on the map view is similar to the time spent on the timeline. The results of the eye tracking data are therefore more or less similar to our expectation. That means the timeline and map view both are useful in getting the information to answer the questions.

The legend zone did attract the participant's gaze by only 0.53%. Before the real test, participants had sometimes to get familiar with the legend. It still proves that the legends are quite easy to remember.

The participants could use functions to find the answers quickly. Figure 5.3 shows that the amount of time that participants spent on the functions zone (13.57%) is more or less equal to 75% of the amount of times that they spent on the timeline (17.08%). We do not expect participants to pay so much attention to the function zone as they spent during the test. Because, normally when participants want to perform filtering functions or use the animation function, they should just look at the function zone and

click on the functions of interest. The process should go very fast, particularly because the participants already familiarized with the prototype. According to Poole and Ball (Poole and Ball, 2005), if participants pay attention at one item longer than expected, it indicates that this item may have lacks of meaningfulness and probably needs to be redesigned. Based on the result, we might think that the functions interface may are some difficulties for users to understand. A problem might be the title of the filtering function (e.g. average speed, lifetime). Furthermore, some filtering options (e.g. lifetime, average speed, travel distance) can only be applied to gain an overview of the data set, which might confuse the participants. Nevertheless, this issue may not happen when users are more familiar with the prototype.

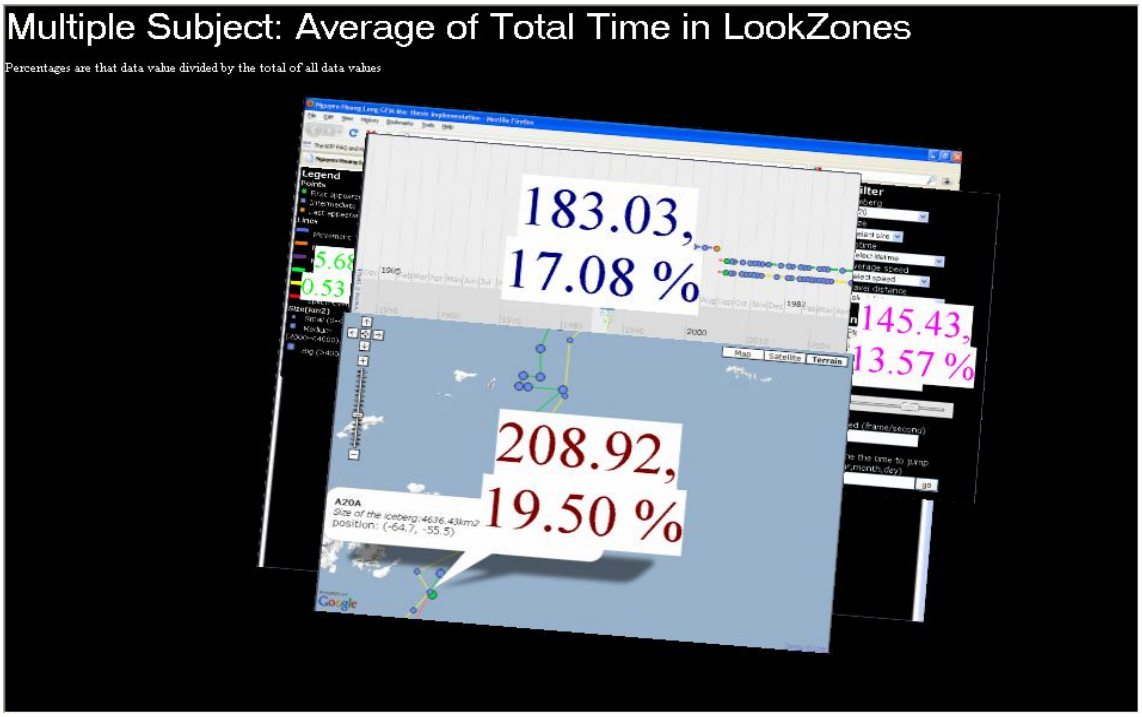

Figure 5.3 Summary of average total time in each Look Zones

5.4.5. Video recorded data

As mentioned before, we also audio/video recorded the test as back-up for Eye tracking. Given some problems with the eye tracking system, this proved to be useful (see below). While analyzing the recordings, we found some issues related to the prototype. Firstly, with the overview questions, we expect that participants would use the animation functions to find answers. However, some participants did not use the animation functions. They only used their mouse for dragging the timeline. The reason for this issue might be because in the test, only data of ten years are provided. Hence, participants could drag the time line to the time they wanted. However, if the time span is longer, participants who did not use the animation now will probably consider using the animation instead of dragging the timeline by mouse manually. Furthermore, none of events on the timeline is labelled. If events on the timeline are labelled, participants do not have to click to identify the item's name.

By watching the video, the reason why 4 participants gave wrong answers to question 3 might be lack of experience of some participants with long time series data. The participants only looked at a subset of the data set instead of looking at the whole data set. As mentioned before, our prototype displays

different data sets at different zoom levels, the boundary between these two zoom levels is fuzzy, and participants may not have noticed the difference, since it showed that participant who gave the wrong answer to question 9, did not zoom in to view the detailed level.

5.4.6. Limitation of the test

After finishing the test, we realized some limitations of the test procedure as well as limitation of the eye tracking system.

Firstly, during the test section, we realized that the GazeTracker software needs a powerful computer to run smoothly. However, in this research the hardware of the system as available in the lab is not really good (the computer's configuration is a Pentium IV, 3.0 GHz and 1,5 Gb of RAM). As mentioned above, the prototype on the same computer as GazeTracker. Hence, when GazeTracker is running, the performance of the prototype was very slow. Users some time felt uncomfortable, lost their patience and some time they did not look at the screen anymore. The results of the test were affected (e.g. biased Eye Tracking results) by the slow system, especially, with the recorded time spent on each task and the user's gaze data. For this reason, we decided to disregard data about time spent on tasks. Furthermore, two participants' data were lost due to the limitations of the systems.

Next, due to the accuracy of the eye tracking system that was used, we could not test whether participants looked at small buttons or not. Furthermore, video the exported from the system seemed to record events on the screen randomly. As a result, the video recording with the eye's cursors as overlay on the prototype is very difficult to analyze. Finally, due to the limited of time and the bias in the eye tracking data, data analysis (e.g. of the excel outputs) could not be fully done.

The purpose of the test is not testing how fast the participants could get familiar with the prototype. Since the prototype has some functionalities that need time to get familiar with, it may be better if participants would be given more time to become familiar with the prototype before the real test.

5.5. Conclusion

In this chapter, a usability testing process was described. Usability testing was conducted using Eye tracking, questionnaires and audio/video recording. The usability testing involved the assessment of the effectiveness and the efficiency of the prototype as well as the user's satisfaction. Furthermore, from eye tracking data and audio/video data, some limitations of the prototype were revealed such as problem with function design or the issue of changing zoom levels. The next chapter will present the conclusions as well as the recommendations for further research.

6. Conclusion and Recommendation

The previous chapter describes the evaluation of the prototype. A usability testing was conducted by combining different usability testing methods, eye tracking and a questionnaire. This chapter will describe the conclusions of the thesis and the recommendation for further research.

6.1. Conclusion

The main objective of this research is to propose a framework for visual exploration of trajectory data on the web using existing visualization libraries. The main objective can be achieved by answering all the research questions as follows:

- What are the characteristics of trajectory data, iceberg data?

The characteristics of trajectory data were reviewed through relevant literatures. The characteristic of icebergs data were also reviewed in chapter 2.

- Which tasks do users want to do in exploring trajectory data, iceberg data?

Based on the changes of the phenomena that proposed by Blok (Blok, 2005), the What-Where-When questions proposed by Peuquet (1994) and the Visual Information Seeking Mantra proposed by Shneiderman (1996), the tasks in visual exploration of trajectory data and icebergs data in particular were analyzed in chapter 2.

- How to translate user requirements into functionalities and representations?

The tasks as discussed above were analyzed combine with reviewed related literatures to translate into required views and functions.

- What are the criteria for choosing suitable web visualization library?

Many web visualization libraries are available on the internet. However, since time is limited, we had to focus on a subset, which originally consisted of ten candidate libraries. Based on the required functions and views discussed in chapter 2, we created some criteria that are described in APENDIX A.1 We then investigated which criteria could be met by the libraries, and decided to limit further investigation to six web visualization libraries, namely ArcGIS JavaScript API, Processing, OpenLayers, Google Maps API, SIMILE and Timemap.js in chapter 3.

- Which open source web library (libraries) is the most suitable for visualizing trajectory data with multivariate attributes?

After considering the characteristics of each library, timemap.js was chosen for our purpose. Timemap.js inherits the functionalities of both Google Maps API and SIMILE timeline. Hence, it

provides more functioned and viewed in visual exploration of trajectory data that are discussed in chapter 2 in comparison with the other libraries.

- Which functionality from the server sides and the database are required?

The main functionalities of function on the database/server side and the function on the client side were reviewed. After considering their characteristics, some functions such as classification and clustering are suggested to perform on the server side to increase the performance of the system. Nevertheless, in this research we did not focus on creating the functions on the server side and the database.

- How to set up the database to store the trajectory data and connect the database with the client side?

Iceberg data were stored in PostgreSQL database. The original data is in excel format. Then python scripts were used to make the sql file that needed to insert data to the PosgreSQL. Next, PHP and Apache server were chosen to transfer data from the database to the client. PostgreSQL, PHP and Apache server are open source software.

- How to design and implement a research prototype, taking user tasks and data characteristics into account?

In the conceptual model of the prototype, we proposed to display different data set or KML files at different zoom levels. At the low zoom level the KML file that consists of the appearance positions of icebergs, the disappearance positions of the icebergs and the major directions (direction from the appearance position to the disappearance position) is displayed. At the higher zoom level, the KML file that consists of all the iceberg data is displayed.

In the implementation stage, a web application that uses timemap.js for geovisualization was implemented for visual exploration of trajectory data, using iceberg data as a case. The prototype provides users an overview of each trajectory. At the low zoom levels, the data of the appearances, the disappearances and the major direction (direction from the appearance to the disappearance) are shown to give users an overview of each trajectory. Combine with animation, an overview of the data set could be more or less given to the users. If users are interested in details of any trajectory, they can use perform zoom in function. At the high zoom levels, the data about the details of each trajectory are shown. Users can also pan the map view and the timeline and filter the data. Timemap.js does not provide an animation function. Hence, we extended the functionalities of the timemap.js to provide users animation functions such as play, pause, restart animation, a function to jump to a specific time and a speed control. Since the iceberg data have temporal irregularities, there are two speed sliders implemented. One controls the speed of the animation. The other one controls the time step between two frames. Colour (for existential changes and trajectories) and size (for size changes) were the main visual variables used in the design of the prototype.

A problem of the SIMILE timeline is that events on the timeline are positioned randomly. It makes it difficult for users to get the information. We improved this by putting all events that belong to the same objects on the same line. It may help users in getting the information of a particular iceberg and focus on objects of interest.

In the conceptual model of the system, we suggested to have a function that generates a new KML file that contains the data after clustering when users want to perform the clustering function on the events that are displayed on their map view. However, due to the limited of time, this function has not implemented yet.

- How to evaluate how the prototype?

A usability testing was conducted by using eye tracking and questionnaire in chapter 5. It described the assessment of the prototype from the effectiveness and satisfaction perspective. Furthermore, from eye tracking data and audio/video data, some limitations of the prototype were revealed such as problem with function design or the issue of changing zoom levels. Due to the limitation of the eye tracking system, the result of the usability testing was not good in all aspects. Nevertheless, this evaluation results may be valuable for further research.

6.2. Limitations of the prototype

During the evaluation, the prototype proved that it be able to help users in exploring the iceberg data (see chapter 5). However, besides the limitations that were revealed during the evaluation that was described in chapter 5 such as the confusion problem with changing zoom levels or the problem with the function design. As developers of the prototype, we realize that the prototype still has some other limitations.

The first limitation is related to the timeline. Depending on the width of the timeline, there will be some overlap when many events (more than ten in our case) have to be displayed simultaneously on the timeline.

The second limitation is related to the number of objects that can be displayed. When the numbers of events is larger than 5000, it will affect the performance of the prototype (slow down the prototype). The computer configuration is: Intel core 2 Duo processor, 4 GB DDR2.

Thirdly, the filtering function of the prototype only let users choose filtering options one by one. For example, users can only filter one iceberg at one time; they cannot choose several icebergs at the same time. Another limitation related to the filtering function is that users cannot filter groups of icebergs other than a whole family of icebergs (for example, if users choose A20, it will display A20, A20A and A20B).

Finally, there are some bugs in SIMILE timeline API. They make the events on the timeline sometimes disappear.

6.3. Recommendation

The main research problems are solved. Nevertheless, some visualization features of the prototype can be improved or developed. The recommendations that we suggest are related to improvement of the limitations discussed above and improvement of the visualization features.

Firstly, as mentioned above the filtering functions in our prototype still has limitations. Hence, Combining with SIMILE facets can be a solution. SIMILE EXHIBIT provides the facet functionality that is very useful for filtering the data (see 3.5). We suggest combining SIMILE EXHIBIT facets with the timemap.js to provide users with a betters filtering mechanism.

Secondly, to deal with the fuzzy boundary of the zoom levels, a colour system should be applied on the zoom level bar. For example, red can be used to indicate that zoom levels for data overview will be displayed, and blue for detailed zoom levels.

Next, as mentioned before, some bugs cause the disappearances of the events. Further works should also concentrate on this problem.

Labelling the first appearance and disappearance of an iceberg on the timeline should be done. Our system could not display the label of the events on the timeline due to some problem with the code. Labelling events on the timeline will help users to recognize the events easier. The purpose of labelling the events is to help users know what the name of the iceberg is without clicking to see the details. To meet this purpose, it is not necessary to label all the events on the timeline. We suggest labelling the appearance and disappearance events only.

Another problem that should be improved is the limitations of the space on the timeline. If there are about 10 events simultaneously, there will be some overlap on the timeline. This problem could be solved by providing a zoom levels on the timeline and by clustering of events that have a similar properties.

Furthermore, the timeline can be improved by giving the option for users to change time interval of the top band of the timeline. In some cases, users may want to see events that are spread in a longer time or they want to display all the events at one time. To meet this requirement, a function that allows the band of the timeline changes according to the user's requirements such as change the time interval of the top band of the timeline from month to year.

Other suggestions are:

- Make use of a line with an arrow to show a direction instead of the simple line to indicate the iceberg's movement. This may help users in getting the direction of the movement when trajectories of object are close and intersect.

- Apply the research prototype to a frequent temporal data to test the possibility of giving users ideas about the speed of moving objects. Given the temporal irregularity of iceberg data, it is difficult for users to compare the speed of different objects.

- Further testing for the usability is necessary to fully evaluate the functionalities of the system. In our usability testing section, we provide some questions for users to find answers. To answer those questions, users not have to use all the provided functions. Hence, further usability testing is recommended.

- Combine the prototype with functions of the database to get clustered data. In some scenarios, too many objects are displayed on the map view. When users click the button that performs the clustering function, it will trigger a process of getting the new data. First, it will get the bounding

box of the area that is displayed on the map and the visible range on the timeline. Then a new KML file will be generated from the database. This KML file stores the data of those items in the area in the visible time after clustering. Finally, this KML file will be displayed instead of the other KML file.

Finally, as discussed in the chapter 2, icebergs movements are influenced by some external phenomena such as ocean currents, bathymetry, sea surface temperate, etc. Our prototype does not have the capability to visualize those external phenomena. Further works should focus on this issue.

7. Appendix

7.1. Appendix A

A.1 Table for choosing the web visualization libraries

Web Libraries	Programming language	Map View	Attribute View	Temporal View	Interaction	Filter	Animation
SIMILE	JavaScript	YES	YES	YES	YES	YES	NO
Processing	JavaScript	YES	NO	YES	NO	NO	YES
Google Map API	JavaScript	YES	YES	NO	YES	YES	NO
OpenLayers	JavaScript	YES	YES	YES	YES	YES	YES
Timemap.js	JavaScript	YES	YES	YES	YES	YES	YES
ESRI JavaScript API	JavaScript	YES	YES	YES	YES	YES	YES
Google Chart API	JavaScript	NO	NO	NO	YES	NO	NO
Jfree	Java	NO	NO	NO	NO	NO	NO
Pchart	PHP	NO	NO	NO	NO	NO	NO
Timemap	Java	YES	YES	YES	YES	NO	YES

A.2 Python script that make the SQL to insert iceberg data to the PostgreSQL database

```
f1 = open("iceberg_fix.txt", "r")
f2 = open("insert_fix.sql", "w")
line=f1.readline()
line=f1.readline()
line=line.split()
#name
for i in range(0,1971):
    f2.write("INSERT INTO icebergs VALUES ('")
    f2.write(line[0])
    f2.write("','")
```

```
    time=line[2].split("-")
    f2.write(time[2])
    f2.write("-")
    f2.write(time[1])
    f2.write("-")
    f2.write(time[0])
    f2.write("',")
    f2.write(line[3])
    f2.write(",")
    f2.write(line[4])
    f2.write(",")
    f2.write(line[6])
    f2.write(",'")
    f2.write(line[7])
    f2.write("','")
    f2.write(line[8])
    f2.write("');\n")
    line=f1.readline()
    line=line.split()
f1.close()
f2.close()
```

7.2. Appendix B

D.1 kml.js

```
 * Timemap.js Copyright 2008 Nick Rabinowitz.
 * Licensed under the MIT License (see LICENSE.txt)
 */
 /**
 * @fileOverview
 * KML Loader
 *
 * @author Nick Rabinowitz (www.nickrabinowitz.com)
 */
/*globals GXml, TimeMap */

/**
 * @class
 * KML loader factory - inherits from remote loader
 *
 * <p>This is a loader class for KML files. Currently supports all
geometry
 * types (point, polyline, polygon, and overlay) and multiple
geometries.</p>
 *
 * @example Usage in TimeMap.init():

    datasets: [
        {
            title: "KML Dataset",
            type: "kml",
            options: {
                url: "mydata.kml"   // Must be local
            }
        }
    ]
 *
 * @param {Object} options          All options for the
loader:<pre>
 *   {Array} url                        URL of KML file to load
(NB: must be local address)
 *   {Function} preloadFunction         Function to call on data
before loading
 *   {Function} transformFunction       Function to call on
individual items before loading
 * </pre>
 * @return {TimeMap.loaders.remote} Remote loader configured for
KML
 */
TimeMap.loaders.kml = function(options) {
    var loader = new TimeMap.loaders.remote(options);
    loader.parse = TimeMap.loaders.kml.parse;
    return loader;
}
/**
 * Static function to parse KML with time data.
 *
 * @param {XML string} kml      KML to be parsed
 * @return {TimeMapItem Array}  Array of TimeMapItems
 */
```

```
TimeMap.loaders.kml.parse = function(kml) {
    var items = [], data, kmlnode, placemarks, pm, i, j;
    kmlnode = GXml.parse(kml);

    // get TimeMap utilty functions
    // assigning to variables should compress better
    var util = TimeMap.util;
    var getTagValue = util.getTagValue,
        getNodeList = util.getNodeList,
        makePoint = util.makePoint,
        makePoly = util.makePoly,
        formatDate = util.formatDate;

    // recursive time data search
    var findNodeTime = function(n, data) {
        var check = false;
        // look for instant timestamp
        var nList = getNodeList(n, "TimeStamp");
        if (nList.length > 0) {
            data.start = getTagValue(nList[0], "when");
            check = true;
        }
        // otherwise look for span
        else {
            nList = getNodeList(n, "TimeSpan");
            if (nList.length > 0) {
                data.start = getTagValue(nList[0], "begin");
                data.end = getTagValue(nList[0], "end") ||
                    // unbounded spans end at the present time
                    formatDate(new Date());
                check = true;
            }
        }
        // try looking recursively at parent nodes
        if (!check) {
            var pn = n.parentNode;
            if (pn.nodeName == "Folder" || pn.nodeName=="Document")
{
                findNodeTime(pn, data);
            }
            pn = null;
        }
    };
    // look for placemarks
    placemarks = getNodeList(kmlnode, "Placemark");
    for (i=0; i<placemarks.length; i++) {
        pm = placemarks[i];
        data = { options: {} };
        // get title & description
        data.title = getTagValue(pm, "name");
            data.index = getTagValue(pm, "index");
        data.options.description = getTagValue(pm, "description");
            data.options.tags = getTagValue(pm, "tags");
            data.options.lineWeight = getTagValue(pm,
"lineWeight");
            data.options.theme = getTagValue(pm, "theme");
        // get time information
        findNodeTime(pm, data);
        // find placemark(s)
```

```
        var nList, coords, pmobj;
        data.placemarks = [];
        // look for marker
        nList = getNodeList(pm, "Point");
        for (j=0; j<nList.length; j++) {
            pmobj = { point: {} };
            // get lat/lon
            coords = getTagValue(nList[j], "coordinates");
            pmobj.point = makePoint(coords, 1);
            data.placemarks.push(pmobj);
        }
        // look for polylines
        nList = getNodeList(pm, "LineString");
        for (j=0; j<nList.length; j++) {
            pmobj = { polyline: [] };
            // get lat/lon
            coords = getTagValue(nList[j], "coordinates");
            pmobj.polyline = makePoly(coords, 1);
            data.placemarks.push(pmobj);
        }
        // look for polygons
        nList = getNodeList(pm, "Polygon");
        for (j=0; j<nList.length; j++) {
            pmobj = { polygon: [] };
            // get lat/lon
            coords = getTagValue(nList[j], "coordinates");
            pmobj.polygon = makePoly(coords, 1);
            // XXX: worth closing unclosed polygons?
            data.placemarks.push(pmobj);
        }
        items.push(data);
    }

    // look for ground overlays
    placemarks = getNodeList(kmlnode, "GroundOverlay");
    for (i=0; i<placemarks.length; i++) {
        pm = placemarks[i];
        data = { options: {}, overlay: {} };
        // get title & description
        data.title = getTagValue(pm, "name");
            data.index = getTagValue(pm, "index");
        data.options.description = getTagValue(pm, "description");
            data.options.tags = getTagValue(pm, "tags");
            data.options.lineWeight = getTagValue(pm,
"lineWeight");
            data.options.theme = getTagValue(pm, "theme");
        // get time information
        findNodeTime(pm, data);
        // get image
        nList = getNodeList(pm, "Icon");
        data.overlay.image = getTagValue(nList[0], "href");
        // get coordinates
        nList = getNodeList(pm, "LatLonBox");
        data.overlay.north = getTagValue(nList[0], "north");
        data.overlay.south = getTagValue(nList[0], "south");
        data.overlay.east = getTagValue(nList[0], "east");
        data.overlay.west = getTagValue(nList[0], "west");
        items.push(data);
    }
```

```
    // clean up
    kmlnode = null;
    placemarks = null;
    pm = null;
    nList = null;
    return items;
};
```

D.2 animation.js

```
// play animation
function play(){
d1= tm.timeline.getBand(0).getCenterVisibleDate();
step=s.getValue();
time=s3.getValue();
time=((100-time)*10)+400;
d1.setDate(d1.getDate()+ step);
tm.timeline.getBand(0).setCenterVisibleDate(d1);
t=setTimeout("play()",time);
};
// stop animation
function pause(){
clearTimeout(t);
};
// reset
function reset()
{
      d = tm.eventSource.getEarliestDate();
      tm.timeline.getBand(0).setMaxVisibleDate(d);
};
```

D.3 funtion.js

```
// filter function for tags
TimeMap.filters.hasSelectedTag = function(item) {
// if no tag was selected, every item should pass the filter
if (!window.selectedTag) {
return true;
}
if (item.opts.tags) {
// look for selected tag
if (item.opts.tags.indexOf(window.selectedTag) >= 0) {
// tag found, item passes
return true;
} else {
// indexOf() returned -1, so the tag wasn't found
return false;
}
}
else {
// item didn't have any tags
return false;
}
};
// onChange handler for pulldown menu
function setSelectedTag(select) {
var idx = select.selectedIndex;
window.selectedTag = select.options[idx].value;
// run filters
tm.filter('map');
tm.filter('timeline');
```

```
// you'll need this to make the timeline update
tm.timeline.layout();
d2 = tm.eventSource.getEarliestDate();
tm.timeline.getBand(0).setCenterVisibleDate(d2);
};
//funtion jump to specific time
function gototime()
{
      gotime=document.getElementById('time_go').value;
      z1=tm.map.getZoom();
      gotime= new Date(gotime);
      tm.timeline.getBand(0).setCenterVisibleDate(gotime);
      tm.map.setZoom(z1);
      tm.timeline.layout();
};
```

D.4 icon.js

```
// medium blue icon
var Test = new GIcon();
Test.image = "../images/blue-circle.png";
Test.colour = "#5A7ACF";
Test.iconSize = new GSize(16,16);
Test.iconAnchor = new GPoint(8,8);
Test.infoWindowAnchor = new GPoint(8,8); TimeMap.themes.blue_medium
= new TimeMapTheme({ icon:Test, eventIconImage: "../images/blue-
circle.png"
// other custom settings, such as event colour, here
});
// huge blue icon
var Test = new GIcon();
Test.image = "../images/blue-circle.png";
Test.colour = "#5A7ACF";
Test.iconSize = new GSize(24,24);
Test.iconAnchor = new GPoint(12,12);
Test.infoWindowAnchor = new GPoint(8,8); TimeMap.themes.blue_huge =
new TimeMapTheme({ icon:Test, eventIconImage: "../images/blue-
circle.png"
// other custom settings, such as event colour, here
});
// small blue icon
var Test = new GIcon();
Test.image = "../images/blue-circle.png";
Test.colour = "#5A7ACF";
Test.iconSize = new GSize(12,12);
Test.iconAnchor = new GPoint(6,6);
Test.infoWindowAnchor = new GPoint(8,8); TimeMap.themes.blue_small
= new TimeMapTheme({ icon:Test, eventIconImage: "../images/blue-
circle.png"
// other custom settings, such as event colour, here
});
// medium orange icon
var Test = new GIcon();
Test.image = "../images/orange-circle.png";
Test.colour = "#FF9900";
Test.iconSize = new GSize(16,16);
eventIconImage: "../images/orange-circle.png";
Test.iconAnchor = new GPoint(8,8);
Test.infoWindowAnchor = new GPoint(8,8);
TimeMap.themes.orange_medium = new TimeMapTheme({ icon:Test,
eventIconImage: "../images/orange-circle.png"
```

```
// other custom settings, such as event colour, here
});
// huge orange icon
var Test = new GIcon();
Test.image = "../images/orange-circle.png";
Test.colour = "#FF9900";
Test.iconSize = new GSize(24,24);
Test.iconAnchor = new GPoint(12,12);
Test.infoWindowAnchor = new GPoint(8,8); TimeMap.themes.orange_huge
= new TimeMapTheme({ icon:Test, eventIconImage: "../images/orange-
circle.png"
// other custom settings, such as event colour, here
});
// small orange icon
var Test = new GIcon();
Test.image = "../images/orange-circle.png";
Test.colour = "#FF9900";
Test.iconSize = new GSize(12,12);
Test.iconAnchor = new GPoint(6,6);
Test.infoWindowAnchor = new GPoint(8,8);
TimeMap.themes.orange_small = new TimeMapTheme({ icon:Test,
eventIconImage: "../images/orange-circle.png"
// other custom settings, such as event colour, here
});
// medium green icon
var Test = new GIcon();
Test.image = "../images/green-circle.png";
Test.colour = "#5A7ACF";
Test.iconSize = new GSize(16,16);
Test.iconAnchor = new GPoint(8,8);
Test.infoWindowAnchor = new GPoint(8,8);
TimeMap.themes.green_medium = new TimeMapTheme({ icon:Test,
eventIconImage: "../images/green-circle.png"
// other custom settings, such as event colour, here
});
// huge green icon
var Test = new GIcon();
Test.image = "../images/green-circle.png";
Test.colour = "#5A7ACF";
Test.iconSize = new GSize(24,24);
Test.iconAnchor = new GPoint(8,8);
Test.infoWindowAnchor = new GPoint(8,8); TimeMap.themes.green_huge
= new TimeMapTheme({ icon:Test, eventIconImage: "../images/green-
circle.png"
// other custom settings, such as event colour, here
});
// small green icon
var Test = new GIcon();
Test.image = "../images/green-circle.png";
Test.colour = "#5A7ACF";
Test.iconSize = new GSize(8,8);
Test.iconAnchor = new GPoint(4,4);
Test.infoWindowAnchor = new GPoint(8,8); TimeMap.themes.green_small
= new TimeMapTheme({ icon:Test, eventIconImage: "../images/green-
circle.png"
// other custom settings, such as event colour, here
});
```

7.3. Appendix C

C.1 Iceberg_kml2.php

```
<?php
$dbh = pg_connect("host=127.0.0.1 dbname=postgres user=postgres
password=hoanglong");
 if (!$dbh) {
     die("Error in connection: " . pg_last_error());
 }
 // execute query
 $sql = "SELECT * FROM iceberg_full_A";
 $result = pg_query($dbh, $sql);
 if (!$result) {
     die("Error in SQL query: " . pg_last_error());
 }
$os = array("Mac", "NT", "Irix", "Linux");
while ($row = pg_fetch_array($result)) {
            $comb=$row['icebergid'];
            settype($comb,"string");
            $c=$comb;
            if (in_array($c, $os, FALSE)) {
                  echo "";
                  }
            else {array_push($os,$c);}
      }
?>
<?php
 $dbh = pg_connect("host=127.0.0.1 dbname=postgres user=postgres
password=hoanglong");
 //connect to a database named "mary" on the host "sheep" with a
username and password
 if (!$dbh) {
     die("Error in connection: " . pg_last_error());
 }
 // execute query
 $sql = "SELECT * FROM iceberg_full_a";
 $result = pg_query($dbh, $sql);
 if (!$result) {
     die("Error in SQL query: " . pg_last_error());
 }
// Creates an array of strings to hold the lines of the KML file.
$kml = array('<?xml version="1.0" encoding="UTF-8"?>');
$kml[] = '<kml xmlns="http://earth.google.com/kml/2.1">';
$kml[] = ' <Document>';
$oldid="A";
$oldpoint = "A";
$n=0;
// Iterates through the rows, printing a node for each row.
while ($row = pg_fetch_array($result))
{
  $kml[] = ' <Placemark>';
  $kml[] = ' <name>'. $row['icebergid'] .'</name>';
  $kml[] = ' <description>Size of the iceberg:' .
htmlentities($row['size']) . 'km2 observed at: ' . $row['times'].
'</description>';
  $kml[] = ' <TimeStamp>';
  $kml[] = ' <when>'. $row['times'] .'</when>';
  $kml[] = ' </TimeStamp>';
```

```
  $kml[] = ' <Point>';
  $kml[] = ' <coordinates>' . $row['lng'] . ','  . $row['lat'] .
'</coordinates>';
  $kml[] = ' </Point>';
  $kml[] = '<tags>' . $row['icebergid'] . "," . $row['sizetype'].
'</tags>';
  if ($row['appearance']=="First")
      {
        if ($row['sizetype'] == "small")
            {
            $kml[] = '<theme>orange_small</theme>';
            $oldpoint=$row['icebergid'];
            }
        else if ($row['sizetype'] == "medium")
            {
            $kml[] = '<theme>orange_medium</theme>';
            $oldpoint=$row['icebergid'];
            }
        else if ($row['sizetype'] == "big")
            {
            $kml[] = '<theme>orange_huge</theme>';
            $oldpoint=$row['icebergid'];
            }
      }
else if ( $row['appearance']=="Last")
      {
if ($row['sizetype'] == "small")
            {
            $kml[] = '<theme>green_small</theme>';
            $oldpoint=$row['icebergid'];
            }
        else if ($row['sizetype'] == "medium")
            {
            $kml[] = '<theme>green_medium</theme>';
            $oldpoint=$row['icebergid'];
            }
        else if ($row['sizetype'] == "big")
            {
            $kml[] = '<theme>green_huge</theme>';
            $oldpoint=$row['icebergid'];
            }
      }
  else if ($row['icebergid']==$oldpoint)
        {
        if ($row['sizetype'] == "small")
            {
            $kml[] = '<theme>blue_small</theme>';
            $oldpoint=$row['icebergid'];
            }
        else if ($row['sizetype'] == "medium")
            {
            $kml[] = '<theme>blue_medium</theme>';
            $oldpoint=$row['icebergid'];
            }
        else if ($row['sizetype'] == "big")
            {
            $kml[] = '<theme>blue_huge</theme>';
            $oldpoint=$row['icebergid'];
            }
```

```
  }
  $kml[] = ' </Placemark>';
  $n++;
      $newid=$row['icebergid'];
      $newtime=$row['times'];
      $newlng=$row['lng'];
      $newlat=$row['lat'];
      if ($newid==$oldid && $n>=2)
            {
            $kml[] = ' <Placemark>';
            $kml[] = ' <name>'. $row['icebergid'] .' </name>';
            $kml[] = ' <description>Trajectory of Iceberg ' .
$row['icebergid'] . ' from ' . $oldtime . ' to ' . $newtime .
'</description>';
            $kml[] = ' <TimeSpan>';
            $kml[] = ' <begin>'. $oldtime .'</begin>';
            $kml[] = ' <end>'. $newtime .'</end>';
            $kml[] = ' </TimeSpan>';
            $kml[] = ' <LineString>';
            $kml[] = ' <coordinates>'. $oldlng . ',' . $oldlat .
"\n" . $row['lng'] . ','  . $row['lat'] . '</coordinates>';
            $kml[] = ' </LineString>';
            $kml[] = ' <tags>' . $row['icebergid'] . '</tags>';
for ($i=1; $i<=60; $i++)
      {
      $m=($i+1)%5;
      if ($row['icebergid']==$os[$i] && $m==1)
            {
            $kml[] = ' <theme>blue</theme>';
            }
      elseif ($row['icebergid']==$os[$i] && $m==2)
            {
            $kml[] = ' <theme>orange</theme>';
            }
      elseif ($row['icebergid']==$os[$i] && $m==3)
            {
            $kml[] = ' <theme>purple</theme>';
            }
      elseif ($row['icebergid']==$os[$i] && $m==4)
            {
            $kml[] = ' <theme>green</theme>';
            }
      elseif ($row['icebergid']==$os[$i] && $m==0)
            {
            $kml[] = ' <theme>yellow</theme>';
            }
      }
            $kml[] = ' </Placemark>';
                                          }
      $oldid=$newid;
      $oldlng=$newlng;
      $oldlat=$newlat;
      $oldtime=$newtime;
}
 // execute query
 $sql1 = "SELECT * FROM parent_child";
 $result_1 = pg_query($dbh, $sql1);
 if (!$result_1) {
     die("Error in SQL query: " . pg_last_error());
```

```
 }
while ($row = pg_fetch_array($result_1))
{
            $kml[] = ' <Placemark>';
            $kml[] = ' <name>'. $row['child'] .'</name>';
            $kml[] = ' <description> Iceberg ' . $row['parrent'] .
' split at ' . $row['last_appearance'] . '</description>';
            $kml[] = ' <TimeSpan>';
            $kml[] = ' <begin>'. $row['last_appearance']
.'</begin>';
            $kml[] = ' <end>'. $row['first_appearance'] .'</end>';
            $kml[] = ' </TimeSpan>';
            $kml[] = ' <LineString>';
            $kml[] = ' <coordinates>'. $row['parent_lat'] . ',' .
$row['parent_lng'] . "\n" . $row['child_lat'] . ','  .
$row['child_lng'] . '</coordinates>';
            $kml[] = ' </LineString>';
            $kml[] = ' <tags>'. $row['parrent'] .'</tags>';
            $kml[] = ' <theme>red</theme>';
            $kml[] = ' </Placemark>';
}
// End XML file
$kml[] = ' </Document>';
$kml[] = '</kml>';
$kmlOutput = join("\n", $kml);
header('Content-type: application/vnd.google-earth.kml+xml');
echo $kmlOutput;
?>
```

C.2 Iceberg_kml1.php

```
<?php
$dbh = pg_connect("host=127.0.0.1 dbname=postgres user=postgres
password=hoanglong");
 if (!$dbh) {
     die("Error in connection: " . pg_last_error());
 }
 // execute query
 $sql = "SELECT * FROM iceberg_full_A";
 $result = pg_query($dbh, $sql);
 if (!$result) {
     die("Error in SQL query: " . pg_last_error());
 }
$os = array("Mac", "NT", "Irix", "Linux");
while ($row = pg_fetch_array($result)) {
            $comb=$row['icebergid'];
            settype($comb,"string");
            $c=$comb;
            if (in_array($c, $os, FALSE)) {
                  echo "";
                  }
            else {array_push($os,$c);}
      }
?>
<?php
 $dbh = pg_connect("host=127.0.0.1 dbname=postgres user=postgres
password=hoanglong");
 //connect to a database named "mary" on the host "sheep" with a
username and password
 if (!$dbh) {
```

```
    die("Error in connection: " . pg_last_error());
 }

 // execute query
 $sql = "SELECT * FROM iceberg_full_a";
 $result = pg_query($dbh, $sql);
 if (!$result) {
    die("Error in SQL query: " . pg_last_error());
 }

// Creates an array of strings to hold the lines of the KML file.
$kml = array('<?xml version="1.0" encoding="UTF-8"?>');
$kml[] = '<kml xmlns="http://earth.google.com/kml/2.1">';
$kml[] = ' <Document>';
$oldid="A";
$oldpoint = "A";
$n=0;
// Iterates through the rows, printing a node for each row.
while ($row = pg_fetch_array($result))
{
if ($row['appearance']=="First" || $row['appearance']=="Last")
      {
  $kml[] = ' <Placemark>';

  $kml[] = ' <name>'. $row['icebergid'] .'</name>';
  $kml[] = ' <description>Size of the iceberg:' .
htmlentities($row['size']) . 'km2 observed at: ' . $row['times'].
'</description>';
  $kml[] = ' <TimeStamp>';
  $kml[] = ' <when>'. $row['times'] .'</when>';
  $kml[] = ' </TimeStamp>';
  $kml[] = ' <Point>';
  $kml[] = ' <coordinates>' . $row['lng'] . ','  . $row['lat'] .
'</coordinates>';
  $kml[] = ' </Point>';
  $kml[] = '<tags>' . $row['icebergid'] . "," . $row['sizetype'].
'</tags>';
  if ($row['appearance']=="First")
      {
        if ($row['sizetype'] == "small")
            {
            $kml[] = '<theme>orange_small</theme>';
            $oldpoint=$row['icebergid'];
            }
        else if ($row['sizetype'] == "medium")
            {
            $kml[] = '<theme>orange_medium</theme>';
            $oldpoint=$row['icebergid'];
            }
        else if ($row['sizetype'] == "big")
            {
            $kml[] = '<theme>orange_huge</theme>';
            $oldpoint=$row['icebergid'];
            }
      }
else if ( $row['appearance']=="Last")
      {
if ($row['sizetype'] == "small")
            {
```

```
            $kml[] = '<theme>green_small</theme>';
            $oldpoint=$row['icebergid'];
            }
        else if ($row['sizetype'] == "medium")
            {
            $kml[] = '<theme>green_medium</theme>';
            $oldpoint=$row['icebergid'];
            }
        else if ($row['sizetype'] == "big")
            {
            $kml[] = '<theme>green_huge</theme>';
            $oldpoint=$row['icebergid'];
            }
      }
  else if ($row['icebergid']==$oldpoint)
        {
        if ($row['sizetype'] == "small")
            {
            $kml[] = '<theme>blue_small</theme>';
            $oldpoint=$row['icebergid'];
            }
        else if ($row['sizetype'] == "medium")
            {
            $kml[] = '<theme>blue_medium</theme>';
            $oldpoint=$row['icebergid'];
            }
        else if ($row['sizetype'] == "big")
            {
            $kml[] = '<theme>blue_huge</theme>';
            $oldpoint=$row['icebergid'];
            }
  }
  $kml[] = ' </Placemark>';
      }
  $n++;
  if ($row['appearance']=="First")
      {
      $oldid=$row['icebergid'];
      $oldtime=$row['times'];
      $oldlng=$row['lng'];
      $oldlat=$row['lat'];
      }
  if ($row['appearance']=="Last" && $row['icebergid']!="A04" &&
$row['icebergid']!="A07" && $row['icebergid']!="A09" &&
$row['icebergid']!="A10")
            {
            $kml[] = ' <Placemark>';
            $kml[] = ' <name>'. $row['icebergid'] .' </name>';
            $kml[] = ' <description>Trajectory of Iceberg ' .
$row['icebergid'] . ' from ' . $oldtime . ' to ' . $row['times'] .
'</description>';
            $kml[] = ' <TimeSpan>';
            $kml[] = ' <begin>'. $oldtime .'</begin>';
            $kml[] = ' <end>'. $row['times'] .'</end>';
            $kml[] = ' </TimeSpan>';
            $kml[] = ' <LineString>';
            $kml[] = ' <coordinates>'. $oldlng . ',' . $oldlat .
"\n" . $row['lng'] . ','  . $row['lat'] . '</coordinates>';
            $kml[] = ' </LineString>';
```

```
            $kml[] = ' <tags>' . $row['icebergid'] . '</tags>';
for ($i=1; $i<=60; $i++)
      {
      $m=($i+1)%5;
      if ($row['icebergid']==$os[$i] && $m==1)
            {
            $kml[] = ' <theme>blue</theme>';
            }
      elseif ($row['icebergid']==$os[$i] && $m==2)
            {
            $kml[] = ' <theme>orange</theme>';
            }
      elseif ($row['icebergid']==$os[$i] && $m==3)
            {
            $kml[] = ' <theme>purple</theme>';
            }
      elseif ($row['icebergid']==$os[$i] && $m==4)
            {
            $kml[] = ' <theme>green</theme>';
            }
      elseif ($row['icebergid']==$os[$i] && $m==0)
            {
            $kml[] = ' <theme>yellow</theme>';
            }
      }
            $kml[] = ' </Placemark>';
                                          }
}
 // execute query
 $sql1 = "SELECT * FROM parent_child";
 $result_1 = pg_query($dbh, $sql1);
 if (!$result_1) {
     die("Error in SQL query: " . pg_last_error());
 }
while ($row = pg_fetch_array($result_1))
{
            $kml[] = ' <Placemark>';
            $kml[] = ' <name>'. $row['child'] .'</name>';
            $kml[] = ' <description> Iceberg ' . $row['parrent'] .
' split at ' . $row['last_appearance'] . '</description>';
            $kml[] = ' <TimeSpan>';
            $kml[] = ' <begin>'. $row['last_appearance']
.'</begin>';
            $kml[] = ' <end>'. $row['first_appearance'] .'</end>';
            $kml[] = ' </TimeSpan>';
            $kml[] = ' <LineString>';
            $kml[] = ' <coordinates>'. $row['parent_lat'] . ',' .
$row['parent_lng'] . "\n" . $row['child_lat'] . ','  .
$row['child_lng'] . '</coordinates>';
            $kml[] = ' </LineString>';
            $kml[] = ' <tags>'. $row['parrent'] .'</tags>';
            $kml[] = ' <theme>red</theme>';
            $kml[] = ' </Placemark>';
}
// End XML file
$kml[] = ' </Document>';
$kml[] = '</kml>';
$kmlOutput = join("\n", $kml);
header('Content-type: application/vnd.google-earth.kml+xml');
```

```
echo $kmlOutput;
?>
```

C.3 thesis.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"><head>
<meta http-equiv="content-type" content="text/html; charset=utf-8"
/><title>Nguyen Hoang Long GFM Msc thesis implementation</title>
<script
src="http://maps.google.com/maps?file=api&amp;v=2&amp;key=ABQIAAAAS
I0kCI-
azC8RgbOZzWc3VRRarOQe_TKf_51Omf6UUSOFm7EABRRhO0PO4nBAO9FCmVDuowVwRO
Lo3w" type="text/JavaScript"></script>
<script type="text/JavaScript"
src="../../timeline_2.3.0/src/webapp/api/timeline-api.js"></script>
<script src="../timemap1.6.js" type="text/JavaScript"></script>
<script src="../markermanager.js" type="text/JavaScript"></script>
<script src="../loaders/kml.js" type="text/JavaScript"></script>
<script src="../manipulation.js" type="text/JavaScript"></script>
<script src="../loaders/progressive.js"
type="text/JavaScript"></script>
<script type="text/JavaScript" src="js/range.js"></script>
<script type="text/JavaScript" src="js/timer.js"></script>
<script type="text/JavaScript" src="js/slider.js"></script>
<script type="text/JavaScript" src="js/animation.js"></script>
<script type="text/JavaScript" src="js/icon.js"></script>
<script type="text/JavaScript" src="js/functions.js"></script>
<script type="text/JavaScript">
//for the array in original_painter
icear= "new" + "<?php echo($icearr);?>";
</script>
<script type="text/JavaScript">
//<![CDATA[
function getQueryString( sProp ) {
      var re = new RegExp( sProp + "=([^\\&]*)", "i" );
      var a = re.exec( document.location.search );
      if ( a == null )
            return "";
      return a[1];
};
function changeCssFile( sCssFile ) {
      var loc = String(document.location);
      var search = document.location.search;
      if ( search != "" )
            loc = loc.replace( search, "" );
      loc = loc + "?css=" + sCssFile;
      document.location.replace( loc );
}

var cssFile = getQueryString( "css" );
if ( cssFile == "" )
      cssFile = "css/bluecurve/bluecurve.css";

document.write("<link type=\"text/css\" rel=\"StyleSheet\" href=\""
+ cssFile + "\" />" );
//]]>
</script>
<script type="text/JavaScript">
```

```
var tm;
function onLoad() {
tm = TimeMap.init({
mapId: "map", // Id of map div element (required)
timelineId: "timeline", // Id of timeline div element (required)
options: {
eventIconPath: "../images/",
},
datasets: [
{
id: "Iceberg1",
type: "kml", // Data to be loaded in KML - must be a local URL
options: {
url: "iceberg_kml2.php" // KML file to load
}
},
{
id: "Iceberg2",
type: "kml", // Data to be loaded in KML - must be a local URL
options: {
url: "iceberg_trajectory.php" // KML file to load
}
},
],
bandIntervals: [
Timeline.DateTime.MONTH, Timeline.DateTime.DECADE
],
dataDisplayedFunction: function(tm) {
tm.datasets.Iceberg1.visible = false;
date_ini=tm.eventSource.getEarliestDate();
tm.timeline.getBand(0).setCenterVisibleDate(date_ini);
tm.filter("map");
tm.filter("timeline");
tm.timeline.layout();
}
// make pathlines. This is accomplished by adding a new filter
chain with a new filter
});
tm.addFilter("map", TimeMap.filters.hasSelectedTag); // hide map
markers on fail
tm.addFilter("timeline", TimeMap.filters.hasSelectedTag); // hide
timeline events on fail
tm.addFilter("map", TimeMap.filters.hasSelectedTitle); // hide map
markers on fail
tm.addFilter("timeline", TimeMap.filters.hasSelectedTitle); // hide
timeline events on fail
GEvent.addListener(tm.map, "zoomend", function(oldLevel, newLevel)
{
if (newLevel>=5)
{
tm.each(function() {
tm.datasets.Iceberg2.visible = false;
tm.datasets.Iceberg1.visible = true;
tm.filter("map");
tm.filter("timeline");
tm.timeline.layout();
});
}
if (newLevel<5)
```

```
{
tm.each(function() {
tm.datasets.Iceberg2.visible = true;
tm.datasets.Iceberg1.visible = false;
tm.filter("map");
tm.filter("timeline");
tm.timeline.layout();
});
}
});
// set invisible for the datasets at start zoom level
}
// event
// function cluster test
function cluster(){
if (5>4)
{
tm.each(function() {
tm.datasets.Iceberg2.visible = true;
tm.datasets.Iceberg1.visible = false;
tm.timeline.getBand(0).setCenterVisibleDate(d1);
tm.filter("map");
tm.filter("timeline");
tm.timeline.layout();
});
}
};
function uncluster(){
if (5>4)
{
tm.each(function() {
tm.datasets.Iceberg2.visible = false;
tm.datasets.Iceberg1.visible = true;
tm.filter("map");
tm.filter("timeline");
tm.timeline.layout();
});
}
};
// new filter
// filter function for title
TimeMap.filters.hasSelectedTitle = function(item) {
// if no tag was selected, every item should pass the filter
if (!window.selectedTitle) {
return true;
}
if (item.title==window.selectedTitle) {
// tag found, item passes
return true;
} else {
// indexOf() returned -1, so the tag wasn't found
return false;
}
};
// setSelect
function setSelectedTitle() {
window.selectedTitle = A11;
// run filters
tm.filter('map');
```

```
tm.filter('timeline');
// you'll need this to make the timeline update
tm.timeline.layout();
};
function refresh() {
ne=tm.eventSource.getCount();
alert(ne);
tm.timeline.layout();
};
intervals="WEEK";
</script>
<link href="examples.css" type="text/css" rel="stylesheet" />
<style>
div#timelinecontainer{ height: 310px; }
div#mapcontainer{ height: 300px; }
    .timeline-event-label {
        margin-top: -2px;
        font-size: 0px;
    }
.body { font-size: 14px; colour: white; }
.header { font-size: 18px; colour: white;}
</style>
</head>
<body onload="onLoad();" onunload="GUnload();">
<div id="help">
<h1>Iceberg</h1>
</div>
<table style="background-colour: rgb(71, 71, 89); width: 100%;
height: 610px;">
<tbody>
<tr>
<td style="width: 15%; background-colour: rgb(0, 0, 0); vertical-
align: top;">
<div><span class="header"><b>Legend</b></span></div>
<div><span class="body"><b>Points</b></span></div>
<div><img src="../images/green-circle.png" /><span class="body">
First appearance</span></div>
<div><img src="../images/blue-circle.png" /><span class="body">
Intermediate position</span></div>
<div><img src="../images/orange-circle.png" /><span class="body">
Last appearance/split</span></div>
<div><span class="body"><b>Lines</b></span></div>
<div><img src="../images/movement1.png" width=24px
height=24px/><span class="body">  Movement</span></div>
<div><img src="../images/movement2.png" width=24px
height=24px/><span class="body">  Movement</span></div>
<div><img src="../images/movement3.png" width=24px
height=24px/><span class="body">  Movement</span></div>
<div><img src="../images/movement4.png" width=24px
height=24px/><span class="body">  Movement</span></div>
<div><img src="../images/movement5.png" width=24px
height=24px/><span class="body">  Movement</span></div>
<div><img src="../images/thinredline.png" width=24px
height=24px/><span class="body">  Split movement</span></div>
<div><span class="body"><b>Size(km2)</b></span></div>
<div><img src="../images/blue-circle.png" style="margin-left:6px;"
width=12px height=12px/><span class="body" style="margin-
left:0px;">   Small (0-<2000)</span></div>
```

```
<div><img src="../images/blue-circle.png" style="margin-left:3px;"
width=18px height=18px/><span class="body" style="margin-
left:2px;">  Medium (2000-<4000) </span></div>
<div><img src="../images/blue-circle.png" width=24px
height=24px/><span class="body">  Big (>4000)</span></div>
<div>a </div>
</td>
<td style="height: 100%;">
<div id="timelinecontainer">
<div id="timeline"></div>
</div>
<div id="mapcontainer">
<div id="map"></div>
</div>
</td>
<td style="width: 20%; background-colour: rgb(0, 0, 0); vertical-
align: top;">
<div><span class="header"><b>Filter</b></span></div>
<div><span class="body">Iceberg</span>
<?php
$dbh = pg_connect("host=127.0.0.1 dbname=postgres user=postgres
password=hoanglong");
 if (!$dbh) {
     die("Error in connection: " . pg_last_error());
 }
 // execute query
 $sql = "SELECT * FROM icebergs";
 $result = pg_query($dbh, $sql);
 if (!$result) {
     die("Error in SQL query: " . pg_last_error());
 }
echo "<form name='ge' action='' method='post'>";
echo "<select name='menu' id='details_1'
onchange='setSelectedTag(this);'>";
echo "<option selected value=''>Select an Iceberg</option>";
$os = array("Mac", "NT", "Irix", "Linux");
while ($row = pg_fetch_array($result)) {
            $comb=$row['icebergid'];
            settype($comb,"string");
            $c=$comb;
            if (in_array($c, $os, FALSE)) {
                  echo "";
                  }
            else {echo "<option value=$c>".$c."</option>";
                  array_push($os,$c);}
      }
echo "</select>";
?>
</div>
<div><span
class="body">Size          
     </span></div>
<?php
$dbh = pg_connect("host=127.0.0.1 dbname=postgres user=postgres
password=hoanglong");
 if (!$dbh) {
     die("Error in connection: " . pg_last_error());
 }
 // execute query
```

```
 $sql = "SELECT * FROM icebergs";
 $result = pg_query($dbh, $sql);
 if (!$result) {
     die("Error in SQL query: " . pg_last_error());
 }
echo "<form name='ge' action='' method='post'>";
echo "<select name='menu' onchange='setSelectedTag(this);'>";
echo "<option selected value=''>Select size</option>";
$os = array("Mac", "NT", "Irix", "Linux");
while ($row = pg_fetch_array($result)) {
            $comb=$row['sizetype'];
            settype($comb,"string");
            $c=$comb;
            if (in_array($c, $os, FALSE)) {
                  echo "";
                  }
            else {echo "<option value=$c>".$c."</option>";
                  array_push($os,$c);}
      }
echo "</select>";
?>
</div>
<div><span
class="body">lifetime         &
nbsp </span>
<form name="ge" action='' method="post">
<select name="direction" onchange="setSelectedTag(this);">
<option selected value="">Select lifetime</option>;
<option selected value="shortlife">short (0-<90 days)</option>;
<option selected value="mediumlife">medium (90-<365 days)</option>;
<option selected value="longlife">long (>365 days)</option>;
</select>
</div>
<div><span class="body" style="margin-left:2px;">Average
speed      </span>
<form name="ge" action='' method="post">
<select name="direction" onchange="setSelectedTag(this);">
<option selected value="">Select speed</option>;
<option selected value="slow">slow (0-<10km2)</option>;
<option selected value="normal">normal (10-<20km2)</option>;
<option selected value="fast">fast (>20km2)</option>;
</select>
</div>
<div><span class="body" style="margin-left:1px;">Travel
distance</span>
<form name="ge" action='' method="post">
<select name="direction" onchange="setSelectedTag(this);">
<option selected value="">Select distance</option>;
<option selected value="shortdistance">short(0-<1000km)</option>;
<option selected value="mediumdistance">medium (1000-
<3000km)</option>;
<option selected value="longdistance">long (>3000km)</option>;
</select>
</div>
<div>a</div>
<div><span class="header"><b>Animation</b></span></div>
<input value="Play" onclick="play()" type="button" />
<input value="Pause" onclick="pause()" type="button" />
<input value="Reset" onclick="reset_animation()" type="button" />
```

```
<div class="slider" id="slider-1" tabIndex="1">
      <input class="slider-input" id="slider-input-1"/>
</div>
<script type="text/JavaScript">
var s = new Slider(document.getElementById("slider-1"),
document.getElementById("slider-input-1"));
s.onchange = function () {
      document.getElementById("h-value").value = s.getValue();
      document.getElementById("h-min").value = s.getMinimum();
      document.getElementById("h-max").value = s.getMaximum();
      s.setValue(s.getValue());
      s.setMinimum(s.getMinimum());
      s.setMaximum(s.getMaximum());
};
s.setValue(50);
window.onresize = function () {
      s.recalculate();
};
</script>
<p>
<span class="body">Time step (days)</span> <input id="h-value"
onchange="s.setValue(parseInt(this.value))"/>
</p>
<div class="slider" id="slider-2" tabIndex="2">
      <input class="slider-input" id="slider-input-2"/>
</div>
<script type="text/JavaScript">

var s3 = new Slider(document.getElementById("slider-2"),
document.getElementById("slider-input-2"));
s3.onchange = function () {
      document.getElementById("time-value").value = s3.getValue();
      document.getElementById("h-min").value = s3.getMinimum();
      document.getElementById("h-max").value = s3.getMaximum();
      s3.setValue(s.getValue());
      s3.setMinimum(s.getMinimum());
      s3.setMaximum(s.getMaximum());
};
s3.setValue(50);

window.onresize = function () {
      s3.recalculate();
};
</script>
<p>
<span class="body">Speed (frame/second)</span> <input id="time-
value" onchange="s3.setValue(parseInt(this.value))"/>
</p>
<div><span class="body">define the time to jump
(year,month,day)</span></div>
<div><input id="time_go"/><input value="go" onclick="gototime()"
type="button" />
</td>
</tr>
</tbody>
</table>
</body></html>
```

Appendix D

D.1 Pre-test questionnaire

1. Personal information
   a) Name / Surname:
   b) Sex:
   c) Nationality:
   d) Education
2. How familiar are you with trajectory data or movement data? Please indicate your familiarity rate based on the template:
   □ Poor □ Modest □ Good
3. How much are you know about icebergs and their characteristics?
   □ Poor □ Modest □ Good
4. How familiar are you with using an interactive map?
   □ Poor □ Modest □ Good
5. How familiar are you with using animation?
   □ Poor □ Modest □ Good
6. How familiar are you with using websites application?
   □ Poor □ Modest □ Good

D.2 Questionnaire after the test

Q1: Rate how easy or difficult in finding the answers for those questions above?
(1 = very difficult; 2 = difficult; 3 = normal; 4 = easy; 5 = very easy)
For over view:

1	2	3	4	5

For zoom and filter:

1	2	3	4	5

For details on demand

1	2	3	4	5

Q2: please rate your satisfaction of the prototype?
(1 = very poor; 2 = poor; 3 = normal; 4 = good; 5 = very good)

1	2	3	4	5

8. References

ALVARES, L. O., BOGORNY, V., KUIJPERS, B., MACEDO, J. A. F. D., MOELANS, B. & VAISMAN, A. (2007) A model for enriching trajectories with semantic geographical information. *Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems.* Seattle, Washington, ACM.

ANDRIENKO, G., ANDRIENKO, N. & WROBEL, S. (2007) Visual analytics tools for analysis of movement data. *SIGKDD Explor. Newsl.,* 9**,** 38-46.

ANDRIENKO, N., ANDRIENKO, G., PELEKIS, N. & SPACCAPIETRA, S. (2008) Basic Concepts of Movement Data. *Mobility, Data Mining and Privacy.* Springer Berlin Heidelberg.

BALLANTYNE, J. & LONG, D. G. (2002) A multidecadal study of the number of Antarctic icebergs using scatterometer data. *Geoscience and Remote Sensing Symposium, 2002. IGARSS '02. 2002 IEEE International.*

BECKER, T., KÖBBEN, B. & BLOK, C. A. (2009) Timemapper : visualizing moving object data using WMS time and SVG SMIL interactive animations. *In: Proceedings SVGOpen 2009: 7th international conference on scalable vector graphics***,** 13.

BLOK, C. A. (2000) Monitoring change : characteristics of dynamic geo - spatial phenomena for visual exploration. *In: Spatial Cognition II : an interdisciplinary approach to representing and processing spatial knowledge / Ch. Freksa ... [et al.] (eds.) Berlin : Springer verlag, 2000. ISBN 3-540-64603-5 (Lecture Notes in Artificial Intelligence : 1404) pp. 16-30.*

BLOK, C. A. (2005) Dynamic visualization variables in animation to support monitoring of spatial phenomena. *ITC Dissertation;119.* Utrecht, Enschede, Universiteit Utrecht, ITC.

BLOK, C. A., TURDUKULOV, U., KOBBEN, B. & CALLE POMARES, J. L. (2009) Web-based visual exploration and error detection in large data sets: Antarctic iceberg tracking data as a case.

BOGORNY, V. & ALVARES, L. O. (2009) Semantic Trajectory Data Mining: a User Driven Approach. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany.

BOGORNY, V. & WACHOWICZ, M. (2009) A Framework for Context-Aware Trajectory. *Data Mining for Business Applications.*

COLLINGS, A., WILLIAMS, R. N., YOUNG, N. & HYLAND, G. (2006) A semiautomated line tracing technique for monitoring ice margins in Antarctic images. *International Journal of Remote Sensing,* 27**,** 433-448.

ÇÖLTEKIN, A., HEIL, B., GARLANDINI, S. & FABRIKANT, S. I. (2009) Evaluating the effectiveness of interactive map interface designs: A case study integrating usability metrics with eye-movement analysis. *Cartography and Geographic Information Science,* 36**,** 5-17.

COMPIETA, P., DI MARTINO, S., BERTOLOTTO, M., FERRUCCI, F. & KECHADI, T. (2007) Exploratory spatio-temporal data mining and visualization. *Journal of Visual Languages & Computing,* 18**,** 255-279.

COOKE, L. (2005) Eye tracking: How it works and how it relates to usability. *Technical Communication,* 52**,** 456-463.

DEATH, R., SIEGERT, M. J., BIGG, G. R. & WADLEY, M. R. (2006) Modelling iceberg trajectories, sedimentation rates and meltwater input to the ocean from the Eurasian Ice Sheet at the Last Glacial Maximum. *Palaeogeography, Palaeoclimatology, Palaeoecology,* 236**,** 135-150.

DESANTIS, R., ZHOU, Q. & RAMEY, J. (2005) A Comparison of Eye Tracking Tools in Usability Testing. *STC Proceedings*.

DIBIASE, D., MACEACHREN, A., KRYGIER, J. & REEVES, C. (1992) Animation and the Role of Map Design in Scientific Visualization. *Cartography and Geographic Information Systems,* 19**,** 201-214.

DODGE, S., WEIBEL, R. & LAUTENSCHUTZ, A.-K. (2008) Taking a Systematic Look at Movement: Towards a Taxonomy of Movement Patterns. *Information Visualization,* 7.

GOLDBERG, J. & KOTVAL, X. (1999) Computer interface evaluation using eye movements: methods and constructs. *International Journal of Industrial Ergonomics,* 24**,** 631-645.

GUERMOUCHE, N., PERRIN, O. & RINGEISSEN, C. (2008) Timed Specification For Web Services Compatibility Analysis. *Electron. Notes Theor. Comput. Sci.,* 200**,** 155-170.

GUTING, R. H. & SCHNEIDER, M. (2005) *Moving objects databases,* Amsterdam etc., Elsevier Morgan Kaufmann.

HAN, J., KAMBER, M. & PEI, J. (2006) *Data Mining: Concepts and Techniques, Second Edition (The Morgan Kaufmann Series in Data Management Systems),* Morgan Kaufmann.

HUANG, B. (2003) Web-based dynamic and interactive environmental visualization. *Computers, Environment and Urban Systems,* 27**,** 623-636.

ISWARI, L. (forthcoming) Back-end support for trajectory management using open source software. Enschede, ITC.

KEIM, D., ANDRIENKO, G., FEKETE, J.-D., GÖRG, C., KOHLHAMMER, J. & MELANÇON, G. (2008) Visual Analytics: Definition, Process, and Challenges. *Information Visualization.*

KRAAK, M. J. (2003) Geovisualization illustrated. *ISPRS journal of photogrammetry and remote sensing,* 57.

LEE, J.-G., HAN, J. & WHANG, K.-Y. (2007) Trajectory clustering: a partition-and-group framework. *Proceedings of the 2007 ACM SIGMOD international conference on Management of data.* Beijing, China, ACM.

LI, X., KRAAK, M. J. & MA, Z. (2007) Towards visual representations to express uncertainty in temporal geodata. *In: ICC 2007 : Proceedings of the 23nd international cartographic conference ICC: Cartography for everyone and for you, 4-10 August 2007 Moscow, Russia. Moscow: International Cartographic Association (ICA), 2007. ISBN: 0-958-46093-0. 14 p.*

MACEACHREN, A. M. & KRAAK, M. J. (2001) Research challenges in geovisualization. *Cartography and geographic information science,* 28.

MANHARTSBERGER, M. & ZELLHOFER, N. (2005) Eye tracking in usability research: What users really see. OCG publication.

MCKEE, L. (2004) The Spatial Web. Open Geospatial Consortium. [cited on 5/6 2009]; Availble from: http://www.opengeospatial.org/pressroom/papers

MOORE, D. S. (2006) *Introduction to the practice of statistics,* New York, Freeman.

OGAO, P. J. & KRAAK, M. J. (2002) Defining visualization operations for temporal cartographic animation design. *International Journal of Applied Earth Observation and Geoinformation,* 4**,** 23-31.

PEUQUET, D. J. (1994) It's About Time: A Conceptual Framework for the Representation of Temporal Dynamics in Geographic Information Systems. *Annals of the Association of American Geographers,* 84**,** 441-461.

POOLE, A. & BALL, L. (2005) Eye tracking in human-computer interaction and usability research. *Encyclopedia of human computer interaction. Idea Group, Pennsylvania***,** 211-219.

RAZEGHI, R. (forthcoming) Usability of eye tracking as an user research technique in geo-information processing and dissemination. Enschede, ITC.

SHNEIDERMAN, B. (1996) The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. *Proceedings of the 1996 IEEE Symposium on Visual Languages.* IEEE Computer Society.

SPACCAPIETRA, S., PARENT, C., DAMIANI, M. L., DE MACEDO, J. A., PORTO, F. & VANGENOT, C. (2008) A conceptual view on trajectories. *Data & Knowledge Engineering,* 65**,** 126-146.

YING, J., GRACANIN, D. & LU, C.-T. (2004) Web visualization of geo-spatial data using SVG and VRML/X3D. *Third International Conference on Image and Graphics, 2004. Proceedings.* .

URL's:

1. National/Naval Ice Center. Antarctic Icebergs current positions. 2009 [cited 2009 2/6]; Available from: http://www.natice.noaa.gov/products/iceberg/.

2. OPENLAYERS. 2009 [cited 2009 10/8]; Available from: http://www.openlayers.org/.

3. SIMILE. 2009 [cited 2009 10/8]; Available from: http://code.google.com/p/simile-widgets/.

4. TIMEMAP. 2009 [cited 2009 10/8]; Available from: http://code.google.com/p/timemap/.

5. Google MAP API. 2009; Available from: http://code.google.com/apis/maps/.

6. Processing. 2009 [cited 2009 10/8]; Available from: http://processingjs.org/.

7. ESRI JavaScript API. 2009 [cited 2009 10/8]; Available from: http://resources.esri.com/arcgisserver/apis/JavaScript/arcgis/.

8. List of 22 action script 3.0 API's. 2009 [cited 2009 15/8]; Available from: http://seantheflexguy.com/blog/2007/08/13/list-of-22-actionscript-30-apis/.

9. Visualizing graphic data set | insideRIA. 2009 [cited 2009 15/8]; Available from: http://www.insideria.com/2008/06/working-with-geogrpahic-data.html

10. Antarctica, topographic map. 2009 [cited 2009 15/9]; Available from: http://maps.grida.no/go/graphic/antarctica-topographic-map
11. Function of Client and Server environment. [cited 2009 15/10]; Available from: http://www.computerfreetips.com/Datebase_tips/sql_server/Administration/sql_server_data.html

12. ESRI JavaScript API example, non geographic info window. 2009 [cited 2009 11/11]; Available from: http://resources.esri.com/help/9.3/arcgisserver/apis/JavaScript/arcgis/demos/map/map_infowindow.html

13. Processing.js exhibition. 2009 [cited 2009 13/11]; Available from: http://Processingjs.org/exhibition

14. National weather service RIDGEII radar display. 2009 [cited 2009 15/11]; Available from: http://radar.srh.noaa.gov/

15. Google Maps API example – Random Weather Map. 2009 [cited 2009 17/11]; Available from: http://gmaps-utility-library.googlecode.com/svn/trunk/markermanager/release/examples/weather_map.htmlr/release/examples/weather_map.html

16. SIMILE Widgets Exhibit Examples, Billionaires. 2009 [cited 2009 19/11]; Available from: http://simile-widgets.org/exhibit/examples/billionaires/billionaires.html

17. Timemap.js KML example. 2009 [cited 2009 21/11]; Available from: http://timemap.googlecode.com/svn/trunk/examples/kenya.html

18. Conceptual model definition. 2009 [cited 2009 07/12]; Available from: http://dictionary.reference.com/browse/conceptual+model

19. KML documentation introduction. 2009 [cited 2009 10/12]; Available from: http://code.google.com/apis/kml/documentation/

20. PHP official website. 2009 [cited 200915/12]; Available from: http://php.net/

21. Slider (WebFX). 2009 [cited 2010 27/12]; Available from: http://webfx.eae.net/dhtml/slider/slider.html

22. Definition of usability. 2010 [cited 2010 10/01]; Available from: http://www.usabilitynet.org/management/b_what.htm